UNIVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA, SISTEMAS Y TELECOMUNICACIONES

PROGRAMA DE INGENIERÍA EN TELECOMUNICACIONES

TRABAJO DE GRADO PARA OPTAR EL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

TÍTULO:

IMPLEMENTACIÓN DE UNA AUTORIDAD DE CERTIFICACIÓN PARA LA AUTENTICACIÓN FUERTE DE LOS USUARIOS ANTE UNA BASE DE DATOS DE INFORMACIÓN ACADÉMICA

Autor:

JOHRMAN DE JESÚS VIDES NIÑO

Director:

Ph.D. ISABEL CRISTINA SATIZÁBAL ECHAVARRÍA

PAMPLONA-COLOMBIA

ENERO de 2016

UNIVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA, SISTEMAS Y TELECOMUNICACIONES

PROGRAMA DE INGENIERÍA EN TELECOMUNICACIONES

TRABAJO DE GRADO PARA OPTAR EL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

TÍTULO:

IMPLEMENTACIÓN DE UNA AUTORIDAD DE CERTIFICACIÓN PARA LA AUTENTICACIÓN FUERTE DE LOS USUARIOS ANTE UNA BASE DE DATOS DE INFORMACIÓN ACADÉMICA

Autor:

JOHRMAN DE JESÚS VIDES NIÑO

Director:

PhD. ISABEL CRISTINA SATIZÁBAL ECHAVARRÍA

JURADO CALIFICADOR:
M.Sc. LUIS ENRIQUE MENDOZA
M.Sc. WILLIAM VILLAMIZAR ROZO
M.Sc. JOSE DEL CARMEN SANTIAGO GUEVARA

PAMPLONA-COLOMBIA

ENERO de 2016

UNIVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA, SISTEMAS Y TELECOMUNICACIONES

PROGRAMA DE INGENIERÍA EN TELECOMUNICACIONES

TRABAJO PRESENTADO PARA OPTAR POR ÉL TITULO DE INGENIERO EN TELECOMUNICACIONES

TEMA:

IMPLEMENTACIÓN DE UNA AUTORIDAD DE CERTIFICACIÓN PARA LA AUTENTICACIÓN FUERTE DE LOS USUARIOS ANTE UNA BASE DE DATOS DE INFORMACIÓN ACADÉMICA

FECHA DE INICIO DEL TRABAJO: NOVIEMBRE 2014

FECHA DE TERMINACION DEL TRABAJO: ENERO de 2016

NOMBRES Y FIRMAS DE AUTORIZACIÓN PARA LA SUSTENTACION:

_____
JOHRMAN DE JESÚS VIDES NIÑO
AUTOR

_____ _____
Ph.D. ISABEL CRISTINA SATIZÁBAL **M.Sc. WILLIAM VILLAMIZAR ROZO**
DIRECTOR **DIRECTOR DEL PROGRAMA**

JURADO CALIFICADOR:

_____ _____
M.Sc. LUIS ENRIQUE MENDOZA **M.Sc. WILLIAM VILLAMIZAR ROZO**

_____
M.Sc. JOSE DEL CARMEN SANTIAGO GUEVARA

PAMPLONA N. S. COLOMBIA

ENERO de 2016

DEDICATORIA

Antes que todo debo agradecer a Dios por permitir todos los medios para hoy poder aspirar a un grado profesional, pero que también me ha hecho crecer como persona. Él sabe que yo fácilmente puedo ganarme el trofeo al terrenal más incrédulo, pero de cierto tiempo para acá, hay cosas que no las puedo responder usando las teorías de Darwin o con las leyes de Newton o Einstein; vi lo oscuro como sólo Kirchhoff ha querido verlo y no imaginárselo con su teoría del cuerpo negro, y estoy seguro que si existe ese cuerpo que no refleja nada de luz, debe existir el cuerpo blanco que la refleja toda, pero he sido tan ciego que no lo puedo ver, sólo espero estar preparado para el día que mi terca mente lo quiera entender.

Agradecer a dos a dos personas muy importantes en mi vida y que gracias a ellos he logrado alcanzar las metas que me he propuesto, que han esto allí para no desviarme del camino; a mis "papás", Daniel Vides Gordon y Cristina Niño Navarro gracias por confiar y creer siempre en mí, porque más que un logro mío, es suyo, gracias por todos los esfuerzos que hicieron los dos sacarnos a mis hermanos y mí, porque yo sé todo lo que les costó y les ha costado, brindarnos lo necesario para darnos armas y que podamos salir delante y valernos por nosotros mismo, no tengo palabras para agradecerles, gracias, los amo.

También quiero agradecerle a mis hermanos Marvin de Jesús Vides Niño y Cristina Isabel Vides Niño por cuidar a mis papás mientras yo estaba por fuera, por los rasguños, abrazos, patadas, besos… ellos, los menores, los que me motivan a darle duro para ser mejor, y un buen ejemplo a ellos…. En la buena pelaos.

Quiero extender estos agradecimientos, en general a toda mi familia, paterna y materna, siempre han esto ahí, apoyando dando fuerza y moral, que si se puede salir adelante, que con disciplina se logran las cosas, soy afortunado de tenerlos como familia. Gracias abuel@s, ti@s, prim@s… Gracias.

No puedo irme sin agradecerle a Yakira Bimber, mi novia, mi amiga, gracias por tu apoyo y compresión, por sacar esas excusas para solo estar conmigo aunque no te importa desvelarte para solo darme un beso y mantenerme despierto mientras sacaba esta tesis cuando el café ya no hacia efecto. Aunque cursi y pasado de moda pero… ♫♪ Ya encontré, ya encontré la mujer que por tanto tiempo había esperado…y es que esa carita ese pelo esa cinturita esos ojitos tan bonitos esa boquita rosadita, me hace sentir el dueño del mundo♪♫

A mis compañer@s de la U, l@s que siempre sonsacaban para cualquier locura, que también juntos trasnochábamos para terminar perdiendo un parcial, gracias a tod@s ell@s, esto nunca vuelve y que honor haber quemado esa etapa con uds. Gracias a Dios por toparlos en mis caminos.

AGRADECIMIENTOS

A toda la planta docente del programa Ingeniería en Telecomunicaciones quienes durante poco más de 4 año me formaron como profesional y como persona, siempre con sus consejos y experiencias de vida que nos hacen reflexionar, no tengo palabras para agradecer las enseñanzas que me ayudan cumplir metas, sueños y lograr todas esas grandes ambiciones.

A mi tutora de tesis, la PhD. Isabel C Satizábal, por sus enseñanzas, no solo en lo académico, sino a esas lecciones que como estudiante y persona no se pueden olvidar, y que en definitiva lo hacen crecer a uno; en lo académico no veo como darle agradecerle pues gracias a esa dedicación que le pone a sus clases, me han hecho decidirme por una línea de postgrado a fin a las telemáticas.

RESUMEN

En una base de datos se encuentra almacenada información que debe resguardarse celosamente, como: datos personales, calificaciones de estudiantes, registro de asignaturas aprobadas y reprobadas, entre otros. Por ello, los problemas que puede tener una organización, que no ofrezca la seguridad necesaria para preservar estos datos, no solo pueden ser legales, sino que pueden afectar su imagen ante la sociedad. El método de autenticación usuario/contraseña puede ser una solución, pero esta es una autenticación débil, ya que la contraseña (algo que se conoce) y el usuario, pueden adivinarse, hallarse (ataque por fuerza bruta) o capturarse (leyendo los paquetes sí la comunicación no está cifrada o con *keylogger*[1]). Utilizar un método de autenticación fuerte, que incluya dos o más factores de autenticación (algo que se conoce, algo que se tiene, algo que se es), permite brindar mayor seguridad y dificulta la labor a los atacantes.

La Universidad de Pamplona utiliza autenticación usuario/contraseña para acceder a la información académica de los estudiantes, por lo que el sistema puede verse comprometido a través de varios tipos de ataques. En este proyecto se crea una Infraestructura de Clave Pública (PKI) a fin de implementar un método de autenticación fuerte, utilizando certificados digitales, para acceder a una base de datos de información académica. La PKI se conforma de: una Autoridad de Registro (RA) que se encarga de registrar a los estudiantes (usuarios) en el sistema, generar las solicitudes de certificados y entregarles la tarjeta inteligente con sus claves y certificado a los usuarios, protegida por un PIN (Número de Indentificación Personal); una Autoridad de Certificación (CA) que recibe las solicitudes de certificados de la RA y vincula la clave pública de cada estudiante con su identidad a través de la firma digital de los certificados digitales. También se encarga de revocar los certificados; una Autoridad de Validación (VA) que se encarga de verificar si los certificados son válidos (no han sido revocados y/o no han expirado); y las Entidades Finales (EE) que son los estudiantes o usuarios del sistema, quienes para autenticarse ante la base de datos de información

[1] *Keylogger*: Es un tipo de software o un dispositivo hardware específico que se encarga de registrar las pulsaciones que se realizan en el teclado, para posteriormente memorizarlas en un fichero o enviarlas a través de internet.

académica deben introducir el PIN de su tarjeta y comprobar la posesión de su clave privada, respondiendo correctamente a un reto, aunque este proceso es transparente para los usuarios. Se simuló una base de datos de información académica, con funcionalidades básicas para verificar el acceso exitoso a esta tras la autenticación de doble factor.

Además se ejecutaron las mismas pruebas que el sistema actual de la Universidad de Pamplona no resistiría, como lo fueron Sniffing, robo identidad, suplantación de identidad e incluso al autor después haber implementado todo el sistema no puedo violentar su propio sistema de autenticación, lo que da por entender, que puede aumentar la seguridad respecto a la autenticación tradicional, como las usadas por nuestro alma mater; las pruebas de seguridad fueron el gran pilar para este proyecto, pero… ¿qué pasa si tenemos el sistema más robusto con un desempeño nefasto? Sería una implementación inviable, por ello, se llevó todos los software creados a un punto de estrés para evaluar su desempeño bajo esta condición y los resultados fueron favorables con tiempos de 8,78 segundos para crear una par de claves en una tarjeta inteligente el bastante prometedor, e incluso los tiempos de lectura fueron aún más bajos aprox 2,528 segundos para acceder al contenido de una tarjeta inteligente, donde todo internamente está cifrado excepto la zona pública, esto sumando al alto nivel de usabilidad dadas las pruebas donde un grupo de 3 personas con conocimientos básicos de computación lograron el manejo de todos los software en promedio en 1,19 segundos esto debido al uso de imágenes características al función y fácil acceso a todas las funciones.

ABSTRACT

Its important to preserve suspiciously a basedate, because of the information it can store, as: private dates, student's grades, register of approved or reproved subjects, among other informations. Due to this, problems that an organization could have, if it doesn't offer a safety security for preserving this basedate that not only can be legal, or else it can affect its image in the face of the society, since its information could not be reliable. The method of authentication user/password can be a solution, but this one is a weak authentication, since the password and the user can be guessed, moreover it can be found assault for brute force or be captured eading the packages, if the comunication is not coded or using keylogger[2]. Use a method of strong authentication, involving two or more factors ofauthentication (something you know, something you have, something that is), allowsto provide greater security and difficult work to the attackers.

The University of Pamplona used authentication username/password to access academic information of students, so that the system can be compromised through various types of attacks. This project creates an infrastructure of public key (PKI) in order to implement a strong authentication method, using digital certificates, to access a database of academic information. The PKI is made up of: an authority record (RA) which is responsible for registering students (users) in the system, generate certificate requests and issuing them with smart card with your keys and certificate users, protected by a PIN (Personal identification number); an authority of certification (CA) that receives requests for certificates of RA and binds the public key of each student with their identity through the digital signature of the digital certificates. Also responsible for revoking certificates; a validation authority (VA) which is responsible for verifying whether the certificates are valid (they have not been revoked or they have not expired); and the entities end (EE) who are students or users of the system who to authenticate to the database of academic information should enter the PIN of your card and verify possession of your password private, properly responding to a challenge, although this process is transparent to users. A database of academic information, with basic functionality to verify successful access to this after the two factor authentication was simulated.

[2] *Keylogger or keystroke:* It is a type software or specific hardware, which is in charge of register keystrokes that are realized in the keypad, it is subsequently memorized in a file or sent it on Internet.

In addition the same evidence that the current system of the University of Pamplona not resist, as they were Sniffing, spoofing, identity theft and even to the author then be implemented system-wide not can violate its own authentication system, which gives understanding, that can increase the security with respect to the authentication of the traditional, such as those used by our alma mater; the safety tests were the great pillar for this project, but... What if we have more robust system with a disastrous performance? It would be a feasible implementation, therefore all software was created to evaluate its performance under this condition to a point ofstress outcomes were favorable with times of 8.78 seconds to create a key pair in asmart card the very promising and even times of reading were even lower approximately 2,528 seconds to access the contents of a smart card where everything is internally coded except the public area, this adding to the high level of usability given evidence where a group of 3 people with basic knowledge of computing were able management of all software on average in 1.19 seconds this due to the use offeatures images to the function and easy access to all functions.

CONTENIDO

LISTA DE FIGURAS

Anexo A

LISTA DE TABLAS

GLOSARIO

Applet: Componente de un aplicación que se ejecuta n el contexto de otro programa, por ejemplo en una navegador web.

Archivo JAR: Java Archive File. Es un archivo comprimido que contiene un conjunto de archivos y clases en Java.

Ataque: Intento de traspasar el control de seguridad de un sistema.

Autenticación: Es el proceso de verificar la identidad de un usuario, equipo u otra entidad en un sistema computación, usualmente como prerrequisito para permitir acceso para recursos en un sistema.

Autorización: Es el proceso por el cual el acceso para un método o recurso es determinado. La Autorización depende sobre la determinación si el principal asociado con un requerimiento a través de la autenticación está en una rol de seguridad dado.

Bug: una propiedad del software o hardware que causa una mal funcionamiento.

HTTP: *HyperText Transfer Protocol* o Protocolo de Transferencia de Hipertexto. Es el protocolo usado en cada transacción de la Web (WWW)

Modo promiscuo: normalmente interfaz Ethernet que permite leer toda la información sin importar su destino, aplicable a un segmento de red.

Capítulo 1

INTRODUCCIÓN

1.1 Planteamiento del Problema

La autenticación es un proceso necesario cuando dos entidades, que participan en un intercambio de mensajes o información sobre una red y/o Internet, desean estar seguros de la identidad de su contraparte [1]. Por otro lado, cada vez más personas y organizaciones están utilizando Internet, para intercambiar información, ya sea poco relevante o muy importante, como para realizar pagos o transacciones bancarias, inicios de sesión, envíos de correos electrónicos, etc. [2]. En otras palabras, autenticar es verificar que otra persona o entidad es quien dice ser.

Probablemente, la forma más básica de autenticación de usuarios, y la más utilizada, es una combinación de usuario/contraseña. Este tipo de autenticación es débil al basarse en un solo factor (algo que se conoce, la contraseña), además los usuarios eligen contraseñas fáciles de recordar y cortas, lo que contribuye a que sea un método poco seguro pues para que tenga un nivel de seguridad aceptable las contraseñas deberían utilizar un conjunto amplio de caracteres (mayúsculas, minúsculas, números, caracteres especiales), tener una longitud mínima de 8 caracteres, limitar el número de intentos de ingreso y ser cambiadas cada cierto tiempo, sin embargo, a un ser humano le es difícil recordar varias contraseñas con estas características. Además, a medida que avanza la tecnología, los ataques contra este tipo de autenticación son cada vez más fáciles de implementar; ataques como *Sniffing*[3] pueden ser efectivos cuando la comunicación no está cifrada, además con computadores de alta capacidad de procesamiento se pueden realizar ataques de diccionario o por fuerza bruta para obtener las contraseñas.

[3] *Sniffing:* Método que captura y almacena los paquetes de datos que viajan por la red, para su posterior análisis [70].

Las bases de datos de información académica almacenan el historial académico de los estudiantes (notas), sus horarios, datos personales de los estudiantes y docentes, etc. Esta información es de carácter privado y sólo puede ser modificada y visualizada por determinadas personas, por ejemplo, las notas de una determinada materia sólo pueden ser modificadas por el docente de dicha materia y cada estudiante sólo puede visualizar sus notas. Por ello, se debe proteger celosamente la confidencialidad e integridad de esta información. Sin embargo, el método de autenticación que se suele utilizar para acceder a este tipo de bases de datos es el tradicional usuario/contraseña que, cómo se mencionó anteriormente, presenta diferentes problemas de seguridad, lo que puede acarrear problemas como la modificación de notas por personas no autorizadas, el robo o la eliminación del historial académico de un estudiante, etc.

Ante los problemas presentados se hace necesario la implementación de un sistema de autenticación fuerte, que resulte de la unión de dos o más factores de autenticación (algo que se conoce, algo que se es, algo que se tiene).

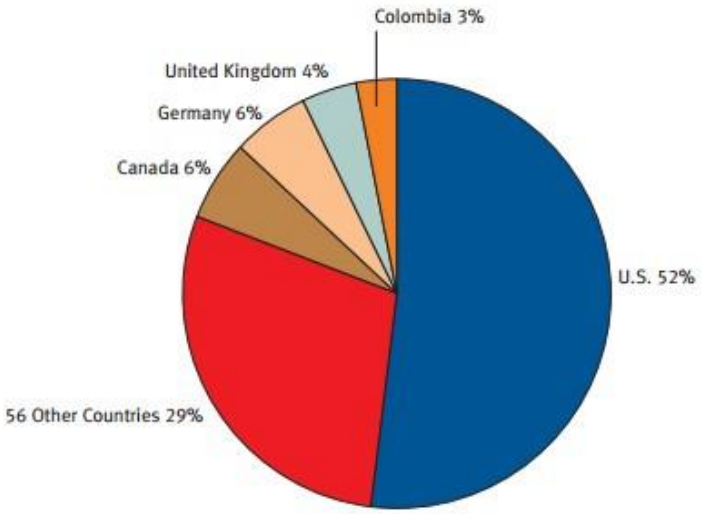
1.2 Justificación

Según organismos internacionales como la *RSA Security*[4], en el mes de enero de 2013, Colombia fue el quinto país más golpeado por ataques de robo de identidad [3]. Como lo muestra la Figura 1-1, en enero de 2013, Colombia tuvo el 3% de los ataques de *phising* a nivel mundial, superado por EEUU con 52%, Canadá y Alemania ambos con 6% e Inglaterra con 4%.

Por otro lado, a nivel Latinoamericano y nacional, como lo revelan la V Encuesta Latinoamérica de Seguridad de la Información [4] y la XII Encuesta Nacional de Seguridad [5] , respectivamente, de 2012 a 2013 se presentó un incremento en los ataques de suplantación de identidad y los accesos no autorizados, pues los ataques de suplantación de identidad aumentaron del 6% al 10% y los accesos no autorizados del 18% al 26%, lo que refleja falencias en los métodos de autenticación utilizados.

[4] *RSA Security:* Filial de *EMC Corporation* (sigla recursiva a partir de sus fundadores Richard Egan y Roger Marino; la C proviene de *Computer*, para ajustarla a la famosa fórmula de Einstein), dedicada a la criptografía y al software de seguridad. El nombre RSA proviene de sus fundadores, Ron Rivest, Adi Shamir y Len Adleman.

Figura 1-1 Ataques de Phising a nivel mundial



Fuente: Reporte de Fraude *Online,* RSA [4]

Actualmente, el sistema de acceso al campus TI de la plataforma HermeSoft, reduce la entropía[5] de las claves al convertir cualquier contraseña a solo números y un atacante puede logearse el número de veces que desee, pues no se limita el número de intentos al iniciar sesión, haciendo posible ataques de diccionario y por fuerza bruta. Además, esta información viaja por un canal que no está cifrado, es decir, viaja en claro, lo que puede conllevar a ataques, como el uso de un *Screenlogger*[6] en el computador de la víctima, para capturar la secuencia de números que introduce cada vez que ingresa al campus TI y la correspondencia de los caracteres con los números, o el uso de un analizador de paquetes (*Sniffer*) en la red de la víctima, para interceptar los paquetes y obtener la secuencia de números que introduce un usuario determinado y el código fuente de la página.

De esta manera, el sistema actual ofrece al usuario común una percepción falsa de seguridad, que podría verse defraudada fácilmente, pues ya se han presentado denuncias por modificación irregular de notas en el pasado[7]. Por dichas razones,

[5] *Entropía de Clave:* Es el número de los posibles valores que puede tomar una clave, lo que determina la fortaleza de la misma [71]

[6] *Screenlogger*, Software que registra la actividad de un computador basado en tomas de captura de pantalla (pantallazo). Puede configurarse para que inicie su registro con ciertas páginas WEB, hacer tomas con cada clic, etc. [72].

[7] Basado en el informe periodístico de Noticias UNO, donde supuestamente se presentaron irregularidades con la calificación de un estudiante, sin el consentimiento de la profesora de la materia [73].

se propone la implementación de una Autoridad de Certificación (CA - *Certification Authority* -), que combinada con criptografía de clave pública, permitirá el uso de un mecanismo de autenticación fuerte, basado en dos factores: algo que se tiene (una tarjeta inteligente) y algo que se conoce (una clave), que beneficie a todos los usuarios de la base de datos de información académica (docentes, estudiantes y administrativos), pues permitirá que la información de los usuarios viaje cifrada y que sea mucho más difícil para un atacante perpetrar cualquier tipo de ataque.

La efectividad del nuevo método de autenticación se probará a nivel de laboratorio. Dependiendo de los resultados de estas pruebas se podrá presentar como alternativa de autenticación a la Universidad de Pamplona.

1.3 Delimitación

1.3.1 Objetivo General

Implementar una autoridad de certificación que permita emitir y revocar los certificados digitales de los usuarios de una base de datos de información académica, para contar con un mecanismo de autenticación fuerte que evite el robo de identidad y accesos no autorizados.

1.3.2 Objetivos Específicos

- Diseñar una aplicación que permita realizar el registro presencial de los usuarios del servidor de base de datos de información académica, generar el par de claves, emitir y revocar los certificados digitales y entregarlos a los usuarios en una tarjeta inteligente
- Implementar la aplicación diseñada para poder realizar el registro de los usuarios al sistema y emitir las tarjetas inteligentes con los certificados y las claves
- Diseñar una aplicación Web que permita acceder a la base de datos de información académica utilizando un mecanismo de autenticación fuerte que involucre el empleo de criptografía de clave pública y el uso de tarjetas inteligentes
- Implementar la aplicación Web diseñada, de manera que los usuarios puedan autenticarse ante la base de datos de información académica utilizando sus tarjetas inteligentes.

- Probar la aplicación de registro implementada, evaluando su amigabilidad, su seguridad en la generación y borrado de claves, y su buen desempeño en la copia de los certificados y las claves en las tarjetas inteligentes.
- Probar el mecanismo de autenticación de la base de datos de información académica, comprobando su buen desempeño y su resistencia a ataques de *sniffing*, robo de identidad y suplantación de identidad.

1.3.3 Acotaciones

Las aplicaciones que se van desarrollar en este proyecto serán probadas a nivel de laboratorio y se creará una base de datos de información académica de prueba que permita evaluar el mecanismo de autenticación fuerte a implementar.

Las herramientas de software a utilizar en el proyecto son:

- Kit que incluya herramientas para el desarrollo, depuración y monitoreo de aplicaciones Java.
- Entorno de desarrollo integrado, hecho principalmente para el lenguaje de programación Java (Editor).
- Un sistema de gestión de bases de datos relacional orientado a objetos.
- Un paquete de herramientas, lo suficientemente robustas para la administración y bibliotecas relacionadas con criptografía, que además suministre funciones criptográficas a otros lenguajes de programación, para la creación de los certificados digitales.
- Herramientas computacionales proporcionadas por la empresa ACS Ltd[8], para la Lectura y escritura de los certificados en las tarjetas inteligentes.

En relación con el hardware:

- ACOS5-64: De *Advanced Card System Ltd*, es una Tarjeta inteligente criptográfica que es compatible con las normas ISO7816-1/2/3/4/8/9.

[8] *ACS Ltd* (*Advanced Card Systems*): Es una empresa desarrolladora de tarjetas inteligentes, lectores de tarjetas inteligentes y productos relacionados-. Los distribuye a más de cien países.

- ACR38: De *Advanced Card System Ltd*, Lector y escritor de tarjeta inteligente.

- CryptoMate64: De *Advanced Card System Ltd*, es una USB criptográfica.

Capítulo 2

MARCO TEÓRICO

2.1 Criptografía Asimétrica

La criptografía simétrica o de clave secreta consiste en utilizar una misma clave para cifrar y descifrar los mensajes. Sin embargo, dicha clave debe conocerla tanto el transmisor como el emisor de los mensajes antes de iniciar la comunicación, lo que se llama "problema de distribución de claves". La criptografía asimétrica permite solucionar dicho problema.

El primer sistema de cifrado asimétrico o de clave pública conocido[9] fue formulado en 1976 por W. Diffie y M. Hellman, quienes influenciados por la tesis de R. Merkle sobre la distribución de claves secretas, dieron a conocer un método práctico para establecer una clave secreta compartida a través de un canal de comunicación inseguro.

La idea del cifrado asimétrico, es que la clave de cifrado es diferente a la clave de descifrado; al lograrlo, se cumple con uno de los seis principios de Kerckhoffs [6]: "la seguridad del sistema reposa sobre la clave y no sobre el algoritmo". En el caso de la criptografía asimétrica, la seguridad reposa sobre la clave privada. La

[9] En 1970 James H. Elis, un matemático británico, ideó una forma para cifrar un mensaje con clave conocida y descifrarla con una privada, pero la rigurosidad del método planteado no permitió ponerlo en práctica [75]. Por otro lado, Clifford Cocks, matemático de profesión, inventó en 1973 lo que hoy en día se conoce como algoritmo RSA, ofreciendo un método práctico para su implementación, y en 1974 otro matemático Malcolm J. Williamson desarrolló lo que hoy se conoce como el algoritmo Diffie-Hellman. Ninguno de los métodos anteriores tuvo un objeto de uso práctico en los usuarios comunes, pero si a nivel militar por lo que estos fueron clasificados por sus gobiernos, esto sumado a su temprana invención, por lo que no se hicieron de conocimiento público [76].

clave pública, como su nombre lo indica, es pública y esta debe estar accesible para cualquier sistema, recurso o persona (desde ahora se llamará *CPub*). En contraposición la clave privada (*CPriv*) se debe mantener y/o almacenar en secreto, y sólo su propietario debe conocerla.

La criptografía asimétrica soluciona los problemas típicos de la criptografía simétrica [7]:

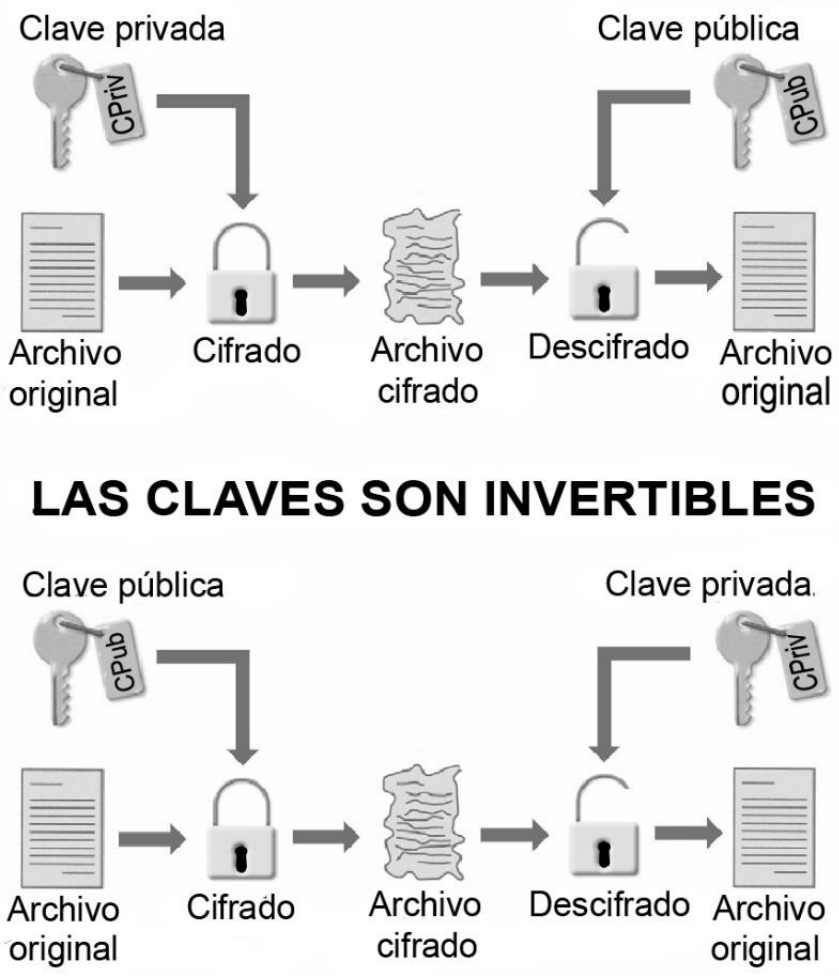
- El intercambiar las claves ya no es una acción crítica, para ello está la criptografía asimétrica. En los casos donde se deba establecer una comunicación privada por tiempo prolongado, se debe establecer una clave de sesión, es decir, una clave simétrica, pues el cifrado de mensajes con claves asimétricas involucraría un coste computacional no admisible, por ello se usa la criptográfica asimétrica para compartir una clave simétrica que luego podrá ser usada como clave de sesión y los mensajes serán cifrados con esta, ahorrando potencia computacional. La clave de sesión debe permanecer secreta, sólo la deben conocer los participantes en la comunicación, y para intercambiar esta clave usando criptografía asimétrica se suele usar el algoritmo Diffie-Hellman.

- En la criptografía simétrica, la autenticidad es probada por el nivel de seguridad con el que se almacena la clave, en la criptografía asimétrica el problema de autenticidad se resuelve de inmediato (al menos parcialmente), ya que sólo el poseedor de la clave privada puede generar un mensaje cifrado, que sólo podrá ser descifrado con su correspondiente clave pública.

- Del mismo modo, se resuelve el problema de privacidad, ya que con un algoritmo asimétrico, desde el punto de vista de las claves (lo que se cifra con la clave privada deber ser descifrado con la clave pública, y viceversa), es suficiente con cifrar un mensaje con la clave pública del receptor, de modo que sólo el poseedor de la clave privada correspondiente podrá leerlo, como se muestra en la Figura 2-1.

- Por último, cada persona o entidad deber poseer un único par de claves (privada y pública): las claves públicas serán almacenas en registros especiales de acceso público (repositorios de certificados), de los que se pueden descargar en un momento dado y las claves privadas guardadas celosamente en un servidor o en una tarjeta inteligente, es decir, en una ubicación segura.

Figura 2-1 Criptografía de Clave Asimétrica



Fuente: Tutorial Firma Electrónica [8].

2.1.1 Algoritmos de Cifrado Asimétrico

Los algoritmos de cifrado asimétrico deben cumplir cuatro (4) condiciones o reglas mínimas que son [9]:

Dada una clave pública $CPub$, una clave privada $CPriv$ y el mensaje M.

1. $CPriv\big(CPub(M)\big) = CPub\big(CPriv(M)\big) = M$ para cualquier mensaje M.
2. Todas las parejas $(CPriv, CPub)$ son diferentes.
3. Deducir $CPriv$ de $CPub$ debe ser tan difícil como obtener M de $CPub(M)$ sin conocer $CPriv$, o M de $CPriv(M)$ sin conocer $CPub$.
4. Las claves $CPriv$ y $CPub$ deben ser fáciles de generar.

La primera condición garantiza la consistencia de cualquier sistema de cifrado asimétrico, la segunda y la tercera regla garantizarán la seguridad del sistema por lo que jamás pueden haber dos o más parejas de claves iguales y el conocimiento de la clave pública no permitirá obtener ninguna información sobre la correspondiente clave privada, ni descifrar el texto que con ella se ha cifrado [7], por lo cual se usan problemas matemáticos difíciles de resolver como la factorización[10], logaritmos discretos[11], mochila tramposa[12], entre otros [10]; y la cuarta condición o regla garantizará la viabilidad y factibilidad del sistema criptográfico.

A pesar que Diffie y Hellman fueron los primeros en formalizar el problema, fueron incapaces de encontrar un sistema que pueda resolver el problema [11] por lo que se usa como un protocolo de intercambio de claves. Ya que el problema de las condiciones o reglas permanecía vigente, esto dio lugar a la creación del algoritmo RSA por R. Rivest, A. Shamir y L. Adleman, este esquema, que se hizo famoso como el primer *sistema de cifrado de clave pública*, se basa en la aplicación de algoritmos aritméticos para enteros grandes, y fue publicado como el estándar PKCS # 1, distribuido de forma privada en 1991 [12].

Los algoritmos de cifrado asimétrico más usados en la actualidad, son: DH [13] [14], RSA y DSA [15] [16] [17].

2.1.1.1 RSA – Rivest Shamir Adleman

El algoritmo RSA fue desarrollado en 1977 en los laboratorios del MIT (*Massachusetts Institute of Technology* – Instituto Tecnológico de Massachusetts) por Ronald Rivest, Adi Shamir y Leonard Adleman, y el nombre del mismo se debe a las iniciales de los apellidos de sus creadores [18]. El algoritmo RSA quizás sea el más sencillo de comprender e implementar [19]. Este algoritmo se basa en la dificultad para factorizar números grandes semiprimos[13] [20], y así hallar los dos números primos generadores que multiplicados entre sí, proporcionen el número semiprimo dado. A partir del producto de estos dos números primos se calculan las claves privada y pública. Las claves ($CPriv$, $CPub$) para el algoritmo RSA se generan de la siguiente manera [21]:

1. Se eligen al azar dos números primos distintos p y q. Recomendaciones,
    - Primero elegir q y luego p. Se aconseja que al elegir p no esté en el rango $q < p < 2q$.
    - Elegir números grandes, co$n$ una longitud mayor a 520 bit$s$, es decir 155 dígitos $decimales\ o\ 128$ dígitos hexadecimales. Al tomar en cuenta esta

[10] *Factorización:* descomponer un número grande en sus factores primos. [13]

[11] *Logaritmo Discreto:* obtener el exponente al que ha sido elevada una base para dar un resultado. ibíd

[12] *Mochila Tramposa:* obtener los sumandos que han dado origen a una suma. ibíd

[13] *Número Semiprimo:* Es un número entero, obtenido a partir de la multiplicación de dos números primos.

recomendación el producto de q y p creará un número cuyo tamaño será mayor a 1024 bits (309 dígitos decimales o 256 dígitos hexadecimales).

2. Hallar $n = p \cdot q$
    - n es el módulo para la $CPriv$ y $CPub$. Su longitud, generalmente expresada en bits, es la longitud de la clave.
    - Como recomendación n debe tener una longitud mayor a 1024 bits (309 dígitos decimales o 256 dígitos hexadecimales) para garantizar una clave fuerte que no sea susceptible a ataques de factorización en un futuro muy cercano. El uso de claves de menor longitud a 1024 bits es inseguro, pues a lo largo de los años, la potencia computacional ha aumentado logrando factorizar números grandes en menor tiempo [22]. Finalizando el año 2009, con el uso de computadoras ordinarias para la época[14] un equipo de investigadores logró factorizar una clave RSA de 768 bits en dos años de trabajo [23]. Así se evidencia que los usuarios comunes cada vez tienen mayor disposición de potencia computacional, por ello, las claves de 1024 bits serán seguras por unos pocos años. Según [24], en el año 2020 será posible factorizar claves de 1024 bits.
3. Se calcula $\varphi(n) = \varphi(p) \cdot \varphi(q) = (p - 1)(q - 1)$, donde φ es la función *phi* de Euler[15]. Este valor se mantiene privado.
4. Se escoge un número e tal que $1 < e < \varphi(n)$ y $MCD(e, \varphi(n)) = 1$. Por lo que e y $\varphi(n)$ son números coprimos[16].
    - (e, n) será la clave pública $CPub$.
    - Con e se tendrá un multiplicador modular inverso[17]. $e\ (mod\ \varphi(n))$.
    - e también es conocido como exponente público.
    - e debe tener una longitud de bits[18] corta y un peso de Hamming[19] pequeño.
    - Si el exponente de codificación e es demasiado bajo, existe la posibilidad de que un atacante pueda recuperar la clave privada a partir de la clave pública usando el *Ataque de Coppersmith*[20] [25]. Se

[14] El equipo de investigadores usó computación matricial y una computadora con procesador 2.2 GHz AMD Opteron y 2GB de memoria RAM.

[15] *Función Phi de Euler* $\varphi(n)$: Es una función aritmética que cuenta los números enteros positivos menores o iguales a n, que son semiprimos de n.

[16] *Coprimos:* Dos números enteros son primos entre sí (o primos relativos o coprimos), si no tienen ningún factor primo en común, o, dicho de otra manera, si no tienen otro divisor común más que 1 y -1. Por ejemplo, 6 y 35 son primos entre sí, pero 6 y 27 no lo son porque ambos son divisibles por 3. [79]

[17] El multiplicador inverso de n mod p existe si y sólo si n y p son semiprimos, es decir, si $\mathrm{MCD}(n, p) = 1$. El multiplicador modular inverso de un entero *n* módulo *p* es un entero *m* tal que $n^{-1} \equiv m \pmod{p}$ [89]

[18] *Longitud de Bits:* Número de dígitos binarios necesarios para representar un número entero. [78]

[19] *Peso de Hamming:* Número de bits de una cadena, diferentes de cero. Ej.: Cadena = 10001111001011, tiene un peso de Hamming de 8.

[20] *Ataque de Coppersmith:* Describe varios tipos de ataques en contra de una clave pública RSA basados en el teorema de Coppersmith donde se ataca los pequeño exponentes modulares del algoritmo.

recomienda que al elegir el exponente público este también resulte ser un número primo de Fermat[21]. El RFC 4871 recomienda usar como máximo el exponente público $e = 65537\ (0x10001)$ [26] debido a que es lo suficientemente grande como para evitar los ataques por el algoritmo de Coppersmith y adecuadamente pequeño para realizar operaciones criptográficas en un tiempo extremadamente rápido.

5. Se determina $d \equiv e^{-1}\ (mod\ \varphi(n))$, es decir que d es el multiplicador modular inverso de e.
   - (d, n) será la clave privada $CPriv$.
   - d es el exponente privado y por tanto, tiene que mantenerse en secreto.
   - Se recomienda comprobar que el exponente privado no cumpla la expresión $d < (1/3)(\sqrt[4]{n})$, pues junto con la condición del primer paso $(q < p < 2q)$ implicaría que el exponente privado es pequeño y supondría un ruptura del sistema de cifrado usando el *ataque Wiener*[22] [27].
6. Una vez obtenidas el par de claves $CPriv = (d, n)$ y $CPub = (e, n)$, se dispondrá de ellas para el cifrado y descifrado de mensajes. La codificación se lleva a cabo según la expresión $C \equiv M^e (mod\ n)$, donde C es el mensaje cifrado, M el mensaje en claro, e exponente público y n un número semiprimo. Mientras la decodificación se hará de la siguiente forma $M \equiv C^d\ (mod\ n)$, donde M es el mensaje en claro recuperado, C el mensaje codificado y d el exponente privado.

Un posible atacante, si quiere recuperar la clave privada a partir de la pública, debe conocer los factores p y q de n, el cual hace parte de la clave pública, y esto es un problema computacionalmente intratable, pues deberá factorizar n que es un número semiprimo de gran tamaño, hasta hallar los únicos dos números primos cuyo producto lo genere. La seguridad del algoritmo RSA se basa en la dificultad de factorizar números semiprimos grandes, por ello, las claves deben ser lo suficientemente largas para garantizar la consistencia del algoritmo.

Debido que el algoritmo RSA será usado en la implementación de esta tesis, a continuación se muestra un ejemplo práctico de cifrado y descifrado usando el algoritmo RSA. Este se realiza con valores pequeños (contrario a lo que se hace a nivel práctico) para lograr su fácil entendimiento y verificar que efectivamente funciona.

1. Se eligen dos números primos distintos, tales como

[21] *Número Primo de Fermat:* Número primo entero positivo de la forma $2\text{^}(2\text{^}N\ ) + 1$ donde N es un número no negativo entero. Los primeros números de Fermat son: 3, 5, 17, 257, 65537, 4294967297, 18446744073709551617…

[22] A*taque Wiener*: Es un tipo de ataque contra el algoritmo RSA. El ataque utiliza el método de fracción continua para hallar la clave privada cuando el exponente privado es pequeño [27].

$q = 53$ y $p = 107$

2. Se calcula $n = pq$
   $n = 53x107 = 5671$
3. Se obtiene $\varphi(n)$
   $\varphi(n) = (p-1)(q-1)$
   $\varphi(5671) = (107-1)(53-1) = 5512$
4. Se escoge el exponente público e. Según el RFC 4871 [26] $e = 65537$, pero no se estaría cumpliendo la condición $1 < e < \varphi(n) \rightarrow 1 < e < 5512$, por tanto debemos usar el anterior número primo de Fermat 257. El exponente público sería $e = 257$. La clave pública estaría dada por $CPub = (e, n) = (257, 5671)$.
5. Se calcula el exponente privado $d \equiv e^{-1} mod\ \varphi(n) \rightarrow d \equiv 257^{-1} mod\ 5512$
   $d \equiv 4697$. La clave privada estaría dada por $CPriv = (d, n) = (4697{,}5671)$
6. Una vez calculado el par de claves, el remitente poseedor de la clave pública $CPub = (257, 5671)$ procederá a cifrar un mensaje en claro $M = 5451$ con $C \equiv M^e (mod\ n) \rightarrow C \equiv 5451^{257} (mod\ 5671) \rightarrow C \equiv 3077$
   Donde $C \equiv 3077$ es el mensaje cifrado, el cual es enviado al poseedor de la correspondiente clave privada, que efectuará la siguiente operación $M \equiv C^d\ (mod\ n) \rightarrow M \equiv 3077^{4697} mod\ 5671 \rightarrow m \equiv 5451$. Como se puede observar el mensaje fue descifrado exitosamente.

En el ejemplo se cifró el mensaje con la clave pública del emisor, para ofrecer confidencialidad, aunque también es posible cifrarlo con la clave privada, pero esta acción no tendría ningún sentido pues cualquiera con la clave pública del emisor podría descifrarlo.

Aunque el algoritmo RSA es bastante seguro teóricamente, existen aristas débiles, debidas a la forma de utilizarlo. Estos puntos débiles pueden ser aprovechados por un atacante. Las vulnerabilidades que presenta son:

- **Ataque de Módulo Común:** Algún administrador podría pensar que, generando una sola vez p y q, sería más rápido generar tantos pares de claves como quiera. Este administrador se verá forzado a modificar el exponente público e cada vez que vaya a crear el par de claves. El producto de $n = pq$ será igual en todas las claves de los usuarios, es decir, el módulo será común. Sin embargo, si el administrador lo hace así, un atacante podrá decodificar los mensajes cifrados sin necesidad de la clave privada (este método funciona sólo si los exponentes públicos son primos relativos o coprimos pero con un exponente público generado al azar, las probabilidades que haya dos primos relativos son altas) mediante el Algoritmo Extendido de Euclides [28]. Por lo tanto, se deben generar p y q diferentes para cada par de claves.

- **Cifrar y Firmar:** Con el algoritmo RSA, cuando se quiera dar confidencialidad, autenticidad y no repudio a un mensaje nunca se debe cifrar y después firmar,

por el contrario se tiene que firmar primero y luego cifrar [29]. Si Alicia va a iniciar sesión en un servidor, y envía su contraseña cifrada con la clave pública del servidor, agregando después su firma digital, ella está incurriendo en una mala práctica, pues si su mensaje es interceptado por el camino, el atacante puede remover la firma digital de Alicia y agregar otra firma, por lo que el servidor le dará el inicio de sesión al atacante, pero si Alicia hubiese firmado digitalmente su contraseña y luego la hubiese cifrado con la clave pública del servidor, el atacante para introducir el criptograma deberá descifrar el mensaje sin conocer la clave privada, es decir, se enfrenta al guerrero más grande del algoritmo RSA, la factorización.

- **Claves Demasiado Cortas:** Actualmente se considera segura una clave RSA con una longitud de *n* de al menos 1024 bits [22], por ahora, ya que la potencia computacional va aumentar con el pasar de los años. Por ejemplo, en el año de 1996, se recomendaba el uso de claves con longitud 512 bits, pero en mayo de 1999, Adi Shamir presentó un dispositivo llamado Twinkle [30], aún no construido, que podría ser incorporado en ordenadores de bajo coste y pondría en serio peligro los mensajes cifrados con claves de 512 bits o menos. Por ello el artículo [ibíd] presentado le entregaba la batuta a las claves de 768 bits, como nueva recomendación pero estás también se rompieron en 2009 [23], y probablemente en los próximos años se logre descomponer un clave de 1024 bits. Así teniendo en cuenta los avances de la tecnología, y suponiendo que el algoritmos RSA no sea roto analíticamente, se debe escoger la longitud de la clave en función del tiempo que se quiera que la información permanezca privada, por ello para servidores se recomienda usar claves de 4096 bits [31], pues claves de esta longitud no podrán ser descompuestas en el futuro cercano.

2.2 Funciones de Hash

En esta sección se habla de las funciones de *hash* conocidas como MDC (Códigos de detección de modificaciones - *Modication Detection Codes*)[23], que van a permitir crear firmas digitales.

El término *hash* proviene, aparentemente, de la analogía con el significado estándar (en inglés) de dicha palabra: 'picadillo', mezclar. Una función de *hash* es unidireccional y tiene como parámetro de entrada una cadena de cualquier longitud (algunos tipos de algoritmos están limitados), la función de *hash* la

[23] *MDC* (*Modication Detection Codes*): Función de hash criptográfica sin clave secreta que puede ser usada para detectar cualquier modificación de la cadena a la cual se aplique, ya sea esta modificación accidental o malintencionada. Por tanto las MDC permiten proteger la integridad de la información [81]

convierte (mapea) a un rango de salida finito con longitud fija, esto se conoce como la huella digital del archivo de entrada, y por tanto la cadena de *hash* resultante debe ser única, y ninguna otra cadena, con al menos un bit alterado puede generar el mismo *hash*. Esta es una de la propiedades fundamentales de las funciones de *hash* [32]. Además estas funciones deben ser capaces de soportar todos los tipos de ataque por criptoanálisis conocidos. Como mínimo, debe tener las siguientes propiedades [33]:

- **Bajo Costo Computacional para Calcular un Resumen:** Esta es una propiedad de todas las funciones unidireccionales, y las funciones de *hash* no son la excepción. Esta propiedad garantiza la factibilidad de un sistema que use las funciones de *hash* para calcular el resumen, además es ideal para los equipos con limitaciones en potencia de cálculo como las tarjetas inteligentes.
- **Resistencia a Pre-imagen:** Con esta propiedad se garantiza que no se pueda hallar la cadena resumida (mensaje original) a partir de una cadena de *hash.*
- **Resistencia a Segunda Pre-imagen:** Dada una entrada m_1 deberá ser difícil encontrar una entrada diferente m_1' de tal manera que $HASH(m_1) = HASH(m_1')$. Así se garantizará que un valor de *hash* no sea duplicado, por cadenas de entrada diferentes.
- **Resistencia a Colisiones:** Al igual que la propiedad anterior, debe ser computacionalmente imposible hallar valores distintos $m_1 \neq m_2$, de tal manera que $HASH(m_1) = HASH(m_2)$. Por tanto la propiedad de resistencia a segunda pre-imagen está contenida en la resistencia a colisiones, mientras que la primera se refiere a cadenas (m_1' , m_1) que varían una a la otra con pequeños cambios, y la segunda con cadenas (m_1, m_2) totalmente diferentes, y por tanto siendo resistentes a la paradoja de cumpleaños[24].

Existen varios tipos de funciones de hash, entre ellos MD5 [34] [35] [36] [37], que produce una cadena de salida de 128 bits y cayó en desuso por el gran número de colisiones que presenta, y SHA (*Secure Hash Algorithm* - Algoritmo de Hash Seguro), que hace alusión a una familia de cinco diferentes funciones de *hash* desarrolladas en 1993 por la NSA (*National Security Agency* - Agencia Nacional de Seguridad) y publicadas por el NIST (*National Institute of Standards and Technology* - Instituto Nacional de Estándares y Tecnología) en 1995 [38].

Los algoritmos de la familia SHA son: SHA-1, SHA-224, SHA-256, SHA-384 y SHA-512 [39], los últimos 4 pertenecen a la subfamilia SHA-2, para distinguirlos de la primera. SHA-1 realiza un resumen del mensaje de sólo 160 bits, mientras que las otras producen el resumen con una longitud en bits igual al número indicado en

[24] *Paradoja de Cumpleaños:* En teoría de probabilidad, se refiere a encontrar la probabilidad de que en un grupo de n personas, hay por lo menos dos personas que cumplen años el mismo día. Esta paradoja de lugar a un ataque criptográfico conocido como ataque de cumpleaños, donde se explotan las posibilidades de hallar dos cadenas que resulten con el mismo valor de hash o resumen. [82]

el acrónimo. El algoritmo SHA-1 es el más común de la familia y se utiliza en numerosas aplicaciones y protocolos, aunque su seguridad fue comprometida en el 2005, cuando criptoanalistas[25] chinos demostraron que no era resistente a colisiones [40], desde entonces no se recomienda su uso.

A pesar de que todavía no se conocen ataques contra las variantes de SHA-2, que es un algoritmo similar al de SHA-1 [39], se han hecho esfuerzos para desarrollar algoritmos de *hash* alternativos y sobretodo más seguros. El 2 de noviembre de 2007 se anunció en las oficinas del *Federal Register*, la competencia para la elección de una nueva función SHA, que sería conocida como SHA-3 y pretendería reemplazar a SHA-2, aunque no se han demostrado ataques realmente significativos. El 2 de octubre de 2012 fue anunciado el ganador del concurso público, el algoritmo Keccak, concebido por un grupo de criptógrafos italianos y belgas, tres años más tarde (05 de agosto 2015), este algoritmo se convierte oficialmente en el nuevo miembro de la familia SHA, bajo el nombre SHA-3.

En este trabajo de grado se utiliza SHA-2 (específicamente SHA-512), por lo que se describe a continuación.

2.2.1 SHA-2

SHA-2 está conformado por cuatro funciones de *hash:* SHA-224, SHA-256, SHA-384 y SHA-512. Estas funciones generan resúmenes de 224, 256, 386 y 512 bits respectivamente. Fueron desarrolladas por la NSA y publicadas por el NIST en 2001. SHA-256 y SHA-512 trabajan respectivamente con 32 y 64 bits de datos, utilizando un número diferente de rotaciones y constantes adicionales, pero su estructura es prácticamente igual [41]. Los algoritmos SHA-224 y SHA-384 solamente son versiones truncadas de las dos anteriores, con un resumen calculado con diferentes valores iniciales. Todas estas variantes han sido patentadas por el gobierno de Estados Unidos pero han sido liberadas bajo licencia libre para que cualquiera pueda acceder a ellas, para su estudio y uso.

SHA-2 incluye cambios sustanciales a SHA-1, tanto así que aún no se ha revelado un ataque práctico contra esta función, y en la actualidad se considera seguro [41]. La seguridad del algoritmo depende por completo de la capacidad de producir un valor diferente para cada entrada específica. A pesar que SHA-2 es presumiblemente más seguro que SHA-1, su uso no fue masivo después de su lanzamiento, esto probablemente se debió al hecho de que SHA-2 no era

[25] *Criptoanalista:* Se dedica al estudio de sistemas criptográficos con el fin de encontrar debilidades en los sistemas y romper su seguridad sin el conocimiento de información secreta.

soportado por sistemas operativos como *Windows XP Service Pack 2* el cual era el más usado para la época [42].

2.3 Autenticación

La autenticación es el proceso por el cual un sistema informático, computador, software o usuario, comprueba de manera segura cierta característica de un objeto, comúnmente la identidad, de otro equipo, software, usuario o datos [43]. Nótese que hay que "autenticar, para poder autorizar", por ende, el término autenticación es distinto de autorización, mientras que la segunda es el proceso de dar a los usuarios el acceso a los objetos del sistema con base en su identidad, la autenticación se limita a confirmar la identidad. Existen dos grandes grupos dentro de los métodos de autenticación: la autenticación de usuarios y la autenticación de mensajes.

2.3.1 Autenticación de Mensajes

Es establecer si un mensaje proviene o fue generado efectivamente por una fuente determinada. El mecanismo mediante el cual se implementa este servicio utilizando criptografía asimétrica es la firma digital, y el utilizado en criptografía simétrica se llama Código de Autenticación de Mensajes (MAC - *Message Authentication Code*-). A continuación se describen las firmas digitales, por ser este el método de autenticación de mensajes utilizado en el presente trabajo de grado.

2.3.1.1 Firma Digital

La firma digital es una tecnología matemático-informática que permite la creación y gestión segura de documentos digitales. Los documentos firmados digitalmente permiten verificar el origen y la integridad del contenido desde el momento de su creación [44].

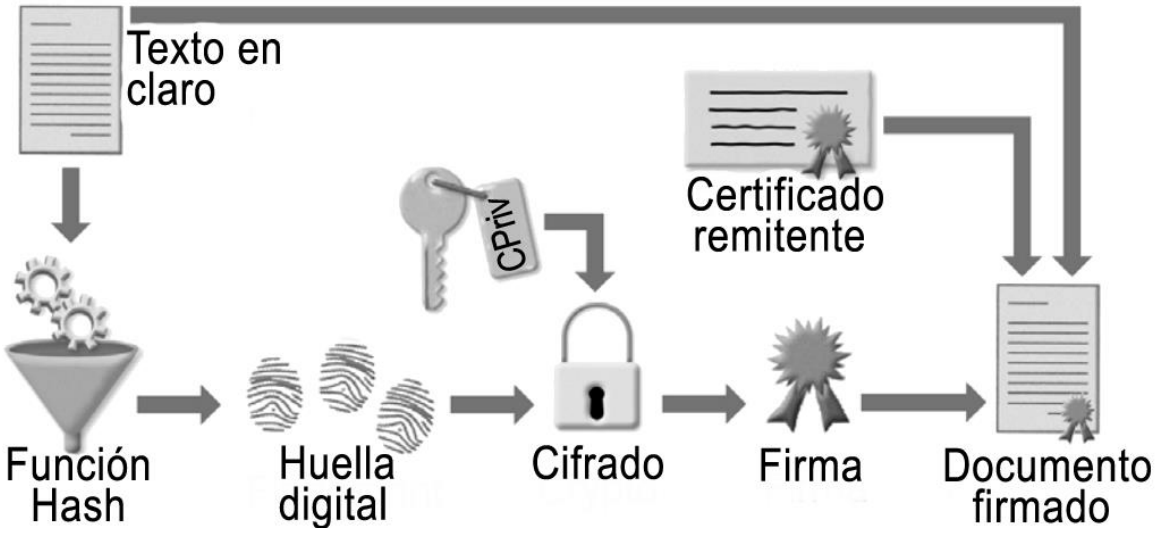
La firma digital incluye dos conceptos ya conocidos: la criptográfica asimétrica y las funciones de *hash* o resumen. En la Figura 2-2 se muestra el proceso para firmar digitalmente cualquier documento digital.

El proceso implica tres etapas principales [44]:

- **Generación del Resumen del Documento:** El mensaje original (texto en claro) es procesado por una función hash para hallar su resumen (huella digital).
- **Generación de la Firma:** El resumen del mensaje se cifra con la clave privada del remitente ($CPriv$) para crear la firma digital.

- **Firmado del Documento:** El documento en claro se firma adjuntando la firma obtenida anteriormente.

Figura 2-2 Proceso de firma de un documento digital



Fuente: Creación propia.

De esta manera, no se cifra el mensaje completo, solo se halla el valor de *hash* del mismo y se cifra este valor con la clave privada para generar la firma digital, así la operación será mucho más rápida, pues toma más tiempo cifrar un mensaje de gran longitud de bits que uno de pocos bits como el valor de *hash.*

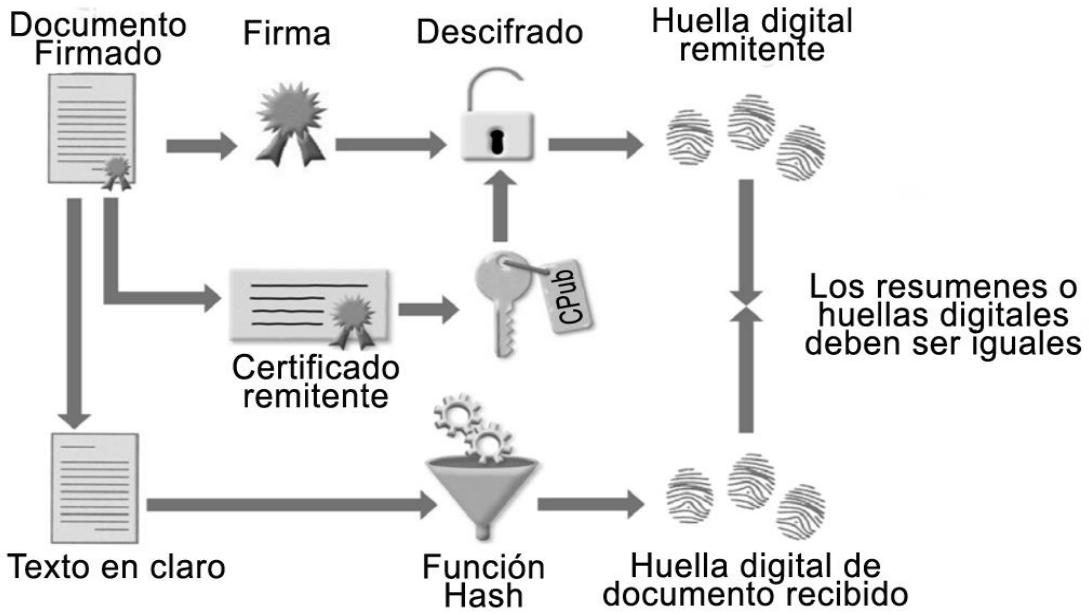
El documento firmado se envía al destinario que, a su vez, tendrá que realizar las comprobaciones para asegurarse de la validez de la firma y de la integridad del documento. Las operaciones que este debe hacer son las que se muestran en la Figura 2-3. El proceso de verificación de la firma digital también incluye algunas etapas, las cuales son [44]:

- El destinatario separa la firma del documento firmado, quedando un texto en claro y la firma que se procesarán por separado.
- Al texto en claro se le calcula el resumen o huella digital.
- La firma se descifra con la clave pública del remitente (el certificado contiene la clave pública) para así hallar la huella digital del documento original.
- Las dos huellas halladas anteriormente se comparan entre si y en este punto habrá dos posibilidades:
    - Los resúmenes o huellas digitales son iguales, por lo tanto el documento es auténtico (solo ese remitente específico pudo haber cifrado el mensaje, pues es el único que posee la clave privada), hay no repudio

de origen (el remitente no podrá negar que ha hecho la operación), y es íntegro (no fue alterado en su camino al receptor).

- Los resúmenes o huellas digitales son diferentes, lo que significa que el documento se ha modificado (intencional o accidentalmente) después que fue firmado por el remitente, es decir, no es un documento confiable.

Figura 2-3 Proceso de verificación de la firma en un documento digital



Fuente: Creación propia.

Este tipo de firma tiene todas las garantías de seguridad, siempre y cuando se tengan en cuenta las recomendaciones mínimas del uso de la criptografía asimétrica.

2.3.2 Autenticación de Usuario

Consiste en verificar la identidad de un usuario. Para ello, el sistema puede optar por solicitar al usuario una o varias credenciales que confirmen su identidad, estas credenciales se pueden clasificar en tres categorías [45]:

- **Algo que se conoce:** En este caso, el usuario debe comprobar al sistema que conoce un secreto, por ejemplo, una contraseña. Por lo general, el nombre de usuario es una cadena de caracteres formada por: un nombre, una dirección de correo electrónico o un apodo, mientras que la contraseña es una cadena que no suele tener un significado particular. La contraseña debe ser lo suficientemente fuerte como para no ser adivinada, pero no demasiado difícil como para no ser recordada.

- **Algo que se tiene:** El usuario requiere para identificarse algún tipo de dispositivo, como: una tarjeta magnética, una tarjeta inteligente, un generador de contraseñas, una llave electrónica, etc.
- **Algo que se es:** Se requiere comprobar alguna característica fisiológica del usuario (biometría), por ejemplo: las huellas digitales, la voz, la retina, el iris, etc.

Cada uno de los tres métodos mencionados anteriormente tiene sus propias ventajas y desventajas, en términos de eficiencia, seguridad, costos computacionales y costos económicos.

La forma más fácil de las tres es la autenticación usuario/contraseña, es decir, basada en algo que se conoce. La seguridad de este método radica en la longitud y recursividad de la contraseña, la combinación de mayúsculas, minúsculas y caracteres especiales, pues un usuario que configure:

- Una clave corta, es susceptible a un ataque por fuerza bruta.
- Una clave fácil (entiéndase por clave fácil la que está compuesta por fechas o nombres importantes para el usuario, o por una palabra común) es susceptible a un ataque por diccionario o simplemente a que sea adivinada.

Por lo general, los usuarios manejan dos o más cuentas en la cuales se autentican de esta forma, y lo recomendado, en este caso, es manejar una contraseña distinta en cada cuenta, pero es difícil recordar tantas contraseñas, sobre todo si son largas o con combinación de caracteres, por lo que más de un usuario opta por configurar la misma contraseña en varias cuentas.

Los sistemas pueden garantizar que nuestra contraseña viaja a través de la red de forma segura (usando comunicación cifrada, por ejemplo, SSL (*Secure Socket Layer*)), y que su almacenamiento también es seguro (guardando en las bases de datos el resumen o huella digital de la contraseña, no la contraseña en claro), pero siempre este tipo de autenticación depende del buen uso que le dé el usuario.

La autenticación por algo que se tiene, es más fuerte que la anterior, pero es más compleja, pues se deben tener los dispositivos o tarjetas que demuestren nuestra identidad (estos dispositivos o tarjetas, contienen información del usuario y variables que interactúan con el sistema), además el coste computacional y de implementación se eleva al tener que manejar periféricos y realizar operaciones transparentes al usuario pues, generalmente, el sistema envía un reto (desafío) y la tarjeta o dispositivo lo descifra [45].

Si la información a la que se va acceder es más delicada y restringida, se opta por sistemas que soliciten al usuario sus huellas dactilares, iris, forma de la mano, etc. Estos son más engorrosos para el usuario, pues debe acceder físicamente a los módulos que leerán su información biométrica.

Como se ve, el nivel de seguridad depende del coste computacional y económico que tenga el sistema, aun así se pueden combinar estos factores al momento de autenticar obteniendo dos categorías de acuerdo al nivel de seguridad que ofrecen [45]:

- **Autenticación Débil:** También llamada autenticación de un solo factor. Es aquella que requiere sólo un factor (por ejemplo, una contraseña) para autenticar a un usuario. El sistema compara la contraseña con la base de datos de nombres de usuario y contraseñas y luego autentica al usuario si coinciden. Esta es la forma más simple, pero más débil de autenticación, porque las contraseñas de los usuarios tienden a ser débiles. La autenticación de un solo factor también puede implicar el uso de una tarjeta como medio para abrir una puerta cerrada o acceder a un sistema, pero esta solo contiene una clave que por lo general es larga.
- **Autenticación Fuerte:** Es el tipo de autenticación de dos o más factores, que normalmente combina dos tipos de autenticación de un solo factor, como algo que se conoce y algo que se tiene. Por ejemplo, los cajeros automáticos requieren autenticación de dos factores: el usuario inserta una tarjeta física en el equipo y luego digita un PIN. Para este tipo de autenticación un factor debe ser físico, como una tarjeta inteligente o un factor biométrico, y el segundo factor debería ser una contraseña. También se pueden combinar los tres factores para pasar a una autenticación muy fuerte. En algunos mecanismos de autenticación fuerte basados en criptografía asimétrica, el usuario debe demostrar que posee la clave privada, bien sea descifrando un reto, que el sistema ha cifrado con su clave pública o bien agregando su firma digital al reto.

2.4 Infraestructura de Clave Pública (PKI)

La clave pública debe ir acompañada de una serie de datos adicionales que enlazan la clave con su legítimo propietario. Esto deja abierto un problema, la confianza. A continuación se detalla la siguiente situación:

Sean Alicia y Bob usuarios que quieren compartir un mensaje (factura de compra) y cada uno posee un par de claves.

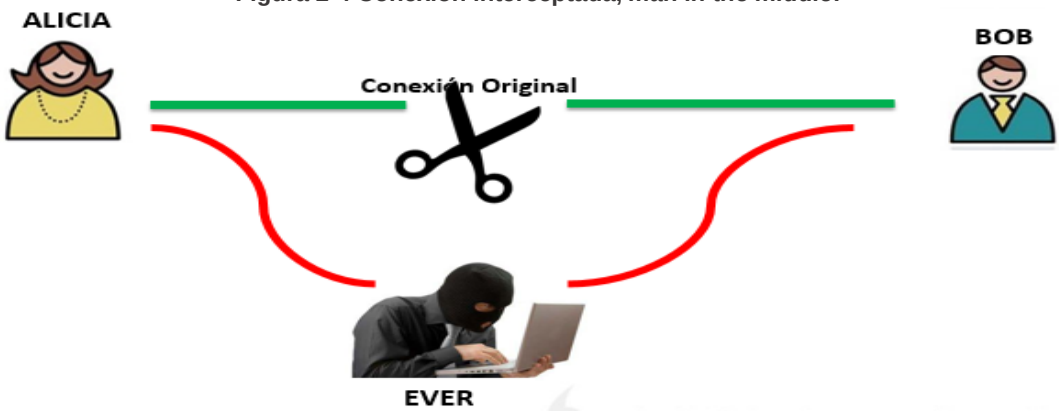
I. Alicia quiere enviar una factura de compra a Bob, donde dice que este debe consignar a Alicia $100 a la cuenta bancaria 123-4.
II. Alicia firma digitalmente la factura (esto consiste en cifrar el resumen o huella digital del mensaje –factura- con su clave privada) y agrega su clave pública para que Bob pueda comprobar la integridad y autenticidad de la factura.
III. Bob recibe la factura firmada digitalmente por Alicia.
IV. Bob descifra con la clave pública el resumen que Alicia envió cifrado con su clave privada.

V. Bob calcula el resumen de la factura, y compara dicho resumen con el resumen que Alicia le envió.
VI. El resumen que Bob calculó resulta ser igual al que Alicia le envió, por tanto Bob está seguro que Alicia fue quien envió esa factura, y se dirige a su banco a realizar la transferencia bancaria.

De momento no se ha dicho nada nuevo, solo se enunció una posible situación entre dos interlocutores. Nótese que de IV a VI se volvió a describir la verificación de una firma digital.

En la situación anteriormente descrita, se agregará otro "jugador". Ever es un atacante que quiere interceptar la comunicación para alterar la factura de Alicia y cambiar la cuenta bancaria por una propia, de manera que cuando Bob haga la consignación bancaria a Alicia le envíe el dinero a Ever y no a Alicia, en otras palabras, Ever va a realizar un robo. Ever puede interferir el enlace de comunicación entre Alicia y Bob para hacerles creer que están hablando con su contraparte. Como se aprecia en la Figura 2-4; cuando Alicia envía la factura firmada digitalmente a Bob, Ever puede interceptar esa factura, remover la firma digital de Alicia, modificar el número de cuenta bancaria e insertar su firma digital en la factura modificada para así enviársela a Bob. Este verificará la integridad y la autenticidad del mensaje y pensará que dicha factura la ha enviado Alicia, procediendo a realizar la transferencia bancaria creyendo que le está consignando a Alicia, pero en realidad lo está haciendo a Ever; en este punto el atacante se habrá salido con la suya.

Figura 2-4 Conexión interceptada, *man in the middle*.



Fuente: Creación propia

El tipo de ataque que se acaba de presentar se conoce como *man in the middle*. Si Bob se hubiese preguntado «*¿realmente la clave pública que viene con la firma digital es de Alicia?*» posiblemente se hubiese abstenido de realizar la consignación bancaria y se habría evitado el fraude. Nótese como una duda (desconfianza) puede evitar inconvenientes.

La solución para la anterior situación es ofrecer confianza a Bob, y ese es el objetivo de una PKI (*Public Key Infrastructure* -Infraestructura de Clave Pública-), a través de entidades conocidas jurídicamente como, prestadores de servicios de certificación[26] [46], que usan certificados digitales para garantizar la identidad de los usuarios. Estos certificados digitales son documentos con la clave pública de un usuario firmados digitalmente por una un "tercera persona"[27]. Esta "tercera persona" es quien brinda la confianza a uno o varios usuarios sobre la legitimidad de la clave pública usada por estos. Así cuando Bob tenga dudas sobre la clave pública que llegó en la anterior factura, solo tendrá que preguntarle a esta "tercera persona" y esta le dará su veredicto si realmente le pertenece a Alicia o no. En la Figura 2-5, se aprecia la situación narrada al principio de esta sección, pero ahora Bob desconfía de la clave pública que recibe, por ello:

I. Alicia primero deberá dar a conocer su clave pública a la "tercera persona" para así poder acreditar esta clave
II. La "tercera persona" le responde a Alicia que ya tiene conocimiento de su clave pública, le hace entrega de un certificado digital firmado (contiene la clave pública de Alicia, la firma digital de la tercera persona, y los datos de Alicia), y además almacena dicho certificado en una base de datos.
III. Alicia procede a firmar con su clave privada la factura y la anexa con el certificado digital que recibió de la "tercera persona".
IV. Bob recibe la factura firmada por Alicia y procede a verificarla, pero Bob desconfía de la clave pública que ha recibido y observa que esa clave pública está avalada por la "tercera persona". Bob se dirige a la "tercera persona" para preguntarle si realmente la clave pública que recibió es de Alicia. La "tercera persona" chequea su base de datos y verifica que la clave pública que Bob envió a revisar pertenece a Alicia.
V. La "tercera persona" responde a Bob, comunicándole que la clave pública realmente pertenece a Alicia.
VI. Bob seguro que ha recibido un factura que ha sido enviada por Alicia, se dirige a su banco a realizar la transferencia.

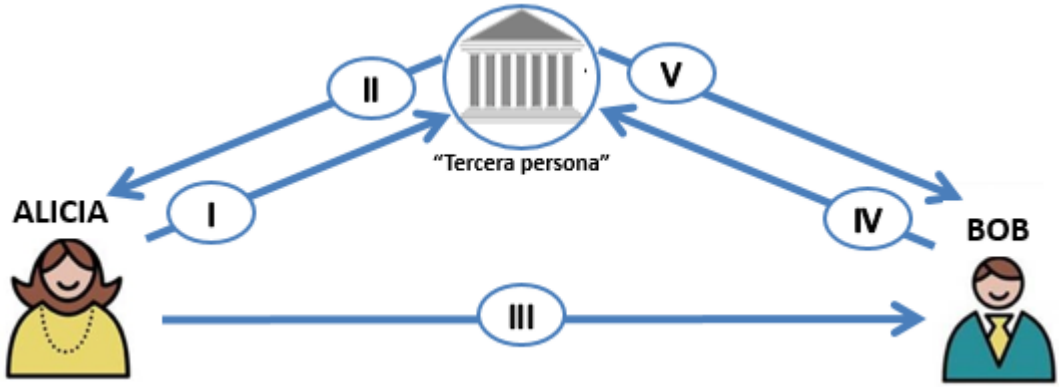
En la anterior situación, Ever, el atacante, intercepta la factura de Alicia, y nota que la firma digital que contiene la factura, ahora viene con un certificado digital. Ever es un atacante perspicaz y también va a crear un certificado digital, la idea es engañar a Bob. Ever contacta a una "cuarta persona" la cual será su secuaz y le pide que firme su clave pública. La "cuarta persona" cómplice del fraude a realizar, sabe que si alguien pregunta por esa clave pública, él debe engañarla y

[26] *Prestador de Servicios de Certificación:* Persona, física o jurídica, que expide certificados electrónicos o que presta otros servicios en relación con la firma electrónica. Ejemplo: VeriSign, Comodo, Certicamara

[27] Se usa "tercera persona" a modo didáctico, pues la emisora de confianza en un sistema PKI es la Autoridad de Certificación.

comunicarle que pertenece a Alicia. Ya con el certificado, Ever firma digitalmente el mensaje y lo agrega a la factura modificada. Bob, recibe la factura, naturalmente desconfía de la clave pública que recibió y se dirige a donde firmaron esa clave. Bob le hace la gran pregunta a la "cuarta persona" y esta responde sin titubear que si es de Alicia. Bob convencido de la autenticidad de dicha clave pública se dirige a su banco a realizar la transferencia del dinero.

Figura 2-5 - Comunicación en base a la confianza



Fuente: creación propia

Como se ha podido apreciar Bob fue nuevamente engañado, a pesar que se ha introducido la figura de "tercera persona" que a efectos de la analogía sería una estructura completa de PKI. ¿Qué falló? Bob no ha debido confiar en "todas las personas", Bob ha debido tener una lista de sus "personas" de confianza, y solo estas deberían tener su credibilidad.

2.4.1 Definición

Una PKI se puede configurar para proporcionar una amplia gama de servicios. En [47] se define como:

> "*El conjunto de hardware, software, gente, políticas y procedimientos necesario para crear, gestionar, guardar, distribuir y revocar certificados digitales, basado en criptografía asimétrica.*"

2.4.2 Componentes

El PKIX-WG (*Public Key Infrastructure X.509 – Work Group*) creado por el IETF (*Internet Engineering Task Force* - Grupo de Trabajo de Ingeniería de Internet -), ha establecido perfiles de certificados X.509, que son unos de los más utilizados

porque se facilita su uso en Internet. Los componentes de una PKI, según las especificaciones PKIX son [48]:

- Autoridad de Certificación (CA)
- Autoridad de Registro (RA)
- Repositorio
- Entidad Final (EE)

2.4.2.1 Autoridad de Registro (RA)

Es quien se encarga de registrar los usuarios en la PKI. Aunque la función de registro se puede implementar directamente en la CA (*Certificaction Authority*), de hecho algunas PKI pequeñas funcionan de esa manera, cuando las soluciones basadas en PKI pronostican un aumento determinado del número de entidades finales y/o las futuras entidades finales están dispersas en un amplio rango geográfico, tiene mucho sentido agregar un componente por separado que se encargue del registro de los usuarios, de lo contrario, la PKI tendría un registro centralizado. El propósito principal de la RA es mejorar la escalabilidad y disminuir los costos de operación de una PKI [46].

El funcionamiento típico de una RA es [45]:

- El usuario solicitante hace llegar a la oficina física de la RA, la documentación necesaria para su plena identificación, además puede manifestar si posee o no un par de claves.
    - Si posee un par de claves: el usuario solicitante deberá crear por sus propios medios una CSR (Solicitud de Firma de Certificado – *Certificate Sign Request*) firmada con su clave privada, para demostrar su posesión.
    - Si no posee un par de claves: la RA creará un par de claves, y dependiendo de la políticas de la PKI, la clave privada se almacenará en un dispositivo criptográfico, como por ejemplo, una tarjeta inteligente, que permitirá operaciones con la clave privada pero nunca su copia o extracción. Después de creadas el par de claves se crea una CSR firmada con la clave privada recién creada.
- Nuevamente de acuerdo a las políticas de la PKI, la CSR creada se podrá firmar por la Autoridad de Registro, para poder demostrar que dicha CSR proviene de un lugar autorizado.
- La CSR es enviada a la Autoridad de Certificación.

2.4.2.2 Autoridad de Certificación (CA)

Es el corazón de una PKI; su función es emitir, revocar[28] y gestionar los certificados digitales. La CA garantiza a los terceros que confían en sus certificados, la relación entre la identidad de un usuario y su clave pública. La confianza de los usuarios en la CA es importante para el funcionamiento del servicio.

El funcionamiento típico de una CA es [45]:

- Luego que la RA ha verificado la información del usuario solicitante, envía la CSR a la CA, y esta, dependiendo de las políticas de la PKI, corrobora nuevamente la información del solicitante y verifica la firma de quien ha enviado la CSR.
- Con base en la información contenida en la CSR se crea el certificado digital, con la firma digital de la Autoridad de Certificación.
- La CA envía copia del certificado creado a una base de datos, llamada repositorio.
- Un certificado puede ser revocado, ya sea por la CA o por solicitud del usuario, quien puede creer que su clave privada fue comprometida (cayó en manos indeseadas). Los certificados revocados se listan por número de serie en la CRL[29] (Lista de Certificados Revocados – *Certificate Revocation List*-) que también está firmada por la Autoridad de Certificación. Si una CA posee una CRL extensa, se hace poco práctica su descarga, por ese motivo existen mecanismos alternativos de consulta de validez de los certificados, como servidores basados en el protocolo OCSP[30] (*Online Certificate Status Protocol*).

Debido a que el certificado de una CA es auto-firmado, se debe instalar en cada equipo este certificado para poder garantizar que dicha CA es de confianza y que todos aquellos certificados firmados por ella son confiables. Es en este punto donde Bob pudo evitar el último fraude, aceptar solo a “tercera persona” como persona de confianza.

---

[28] *Certificado Revocado:* Certificado que no es válido, o es cancelado por la Autoridad de Certificación.

[29] *CRL:* Es una lista de los certificados (o más específicamente, una lista de los números de serie de los certificados) que han sido revocados, y por lo tanto, las entidades finales que presentan aquellos certificados (revocados) ya no deben ser de confianza. Según el RFC 3280, existen dos estados de certificados revocados: Revocado (revoked) y Hold, el primero es revocado de manera irreversible, pero el segundo estado es reversible, es decir, es un revocación temporal. Un certificado expirado no debe estar en la CRL. El RFC 5280 define las razones por las cuales un certificado puede ser revocado.

[30] *OCSP:* Definido en el RFC 2560, es un método para determinar el estado de revocación de un certificado digital usando otros medios que no sean el uso de CRL. Tiene la ventaja de funcionar en línea, evitando descargar las CRLs.

2.4.2.3 Repositorio

Su tarea principal es distribuir los certificados firmados y las CRLs. La distribución de los certificados depende por completo de la PKI con la que se esté trabajando, y puede llevarse a cabo de diferentes formas, dependiendo de la políticas de la PKI.

La integridad de la información que almacenan se logra a través de la firma de la CA. Un repositorio debe centrarse en la disponibilidad y rendimiento del sistema, pues es de acceso a todo público.

2.4.2.4 Entidad Final (EE)

Puede ser un cliente de los servicios brindados por la PKI, un sistema de usuario final, un dispositivo, un proceso o cualquier cosa identificada como el sujeto de un certificado de clave pública.

2.4.3 Certificados de Clave Pública (PKCs)

La distribución de claves públicas (además de la confianza) es un problema crucial en la criptografía asimétrica, el cual se resuelve mediante el uso de certificados digitales. Según [45]:

> *"El certificado digital es un documento en el que una autoridad de certificación […] garantiza que una clave pública y un determinado sujeto están realmente asociados, evitando de esta forma posibles suplantaciones de identidad."*

Por tanto, un certificado de clave pública identifica de forma exclusiva a una entidad. Un tipo de autenticación basada en certificados es fuerte, pues en el proceso de autenticación se debe demostrar el conocimiento de la clave privada correspondiente al certificado digital (algo que se conoce), esta debe estar almacenada en una ubicación, que por lo general es una tarjeta inteligente (algo que se tiene), y el acceso está protegido con una contraseña (algo que se conoce).

El estándar de certificados digitales X.509 es el más usado en la actualidad. Se han desarrollado varias versiones como se aprecia en la Figura 2-6, pero la tercera versión es la que está más difundida en [49]. Este estándar creado por la UIT (Unión Internacional de Telecomunicaciones) hizo su debut con la primera versión en 1988, donde propone un sistema jerárquico de autoridades de certificación para la emisión de certificados y avalar la validez de las claves públicas de los demás. Esta versión contiene 7 campos [50]:

- **Versión del certificado (*version*):** Informa sobre la versión del certificado que se está usando.
- **Número de serie (*serialNumber*):** Se asigna un número entero a cada certificado para facilitar la administración de estos, por ende, cada certificado debe tener un número de serie distinto en una respectiva CA.
- **OID[31] del método usado para firmar el certificado *(signature)*:** Identifica el algoritmo y la función de hash utilizados por la CA para firmar el certificado, por ejemplo, *md5WithRSAEncryption*, *sha512WithRSAEncryption*, etc..
- **Firmante (*issuer*):** Nombre de la CA que ha firmado y expedido el certificado.
- **Validez (*validity*):** El certificado es válido sólo durante un intervalo de tiempo, es decir, no es válido antes de la fecha de activación (*not-valid-before*), ni después de la fecha de expiración (*not-valid-after).*
- **Titular (*subject*):** Asocia la clave pública a la persona que está descrita en este campo. En este campo y en *issuer* la información se proporciona en el formato de nombre distinguido (DN - *Distinguished Name* -) descrito en el RFC 1779 así [51]:
    - CN: Nombre del titular. Ej.: Carlos Martinez Bejarano.
    - O: Organización. Ej.: Universidad de Pamplona
    - OU: Departamento dentro de la organización. Ej.: Facultad de Ingeniería
    - C: Nombre del país en código ISO 3166-1 alfa-2[32]. Ej.: CO
    - ST: Estado, provincia o departamento. Ej.: Bolívar
    - L: Localidad, ciudad o municipio. Ej.: Cartagena
    - E: dirección de correo electrónico. Ej.: carlosmartinez@ejemplo.com
- **Algoritmo que usado para crear la clave pública del titular (*subjectPublicKeyInfo*)**.

Los siete campos del certificado X.509v1 no son suficientes para la mayoría de los casos, por ello, en 1993 se creó una segunda versión de X.509 con la adición de dos campos más, identificadores únicos de sujeto (*subjectUniqueIdentifier*) y de emisor (*issuerUniqueIdentifier*) [50], con el objetivo de evitar homónimos en los certificados, ya sea por parte de un sujeto (usuario) o un emisor (CA). Así *subjectUniqueIdentifier* es un código que debe ser diferente para cada usuario haciendo diferenciar dos certificados con el mismo nombre de diferentes titulares, al igual sucede con el otro campo agregado, donde se asigna un identificador a cada CA para diferenciarlas de aquellas que tengan el mismo nombre.

[31] *OID (Identificador de Objeto - Object identifier):* A cada objeto en particular se le asigna un código, ya sea numéricoo alfanumérico para su fácil identificación. Las partes implicadas se deben poner de acuerdo para fijar estos códigos, ya sea por medio de un estándar o un RFC.

[32] *ISO 3166-1 alfa-2:* Código de letras y/o números para representar el nombre de un país en dos caracteres. Ejemplo: Colombia (CO), Estados Unidos (US), Brasil (BR), Francia (FR), Italia (IT), etc.

La tercera versión definida en el RFC 5280, lanzada en 1996, es la más usada en la actualidad. Esta versión incluye el campo *extensions* [50], que permite la adición de nueva información a la estructura del certificado sin modificar su definición inicial. Algunas extensiones usadas son:

- ***Authority Information Access:*** Esta extensión permite ubicar, por medio de una URL, certificados raíz o intermedios faltantes, puntos de distribución de CRLs , estados de certificados para su validación, entre otros servicios.
- ***Basic Constraints:*** La extensión de las restricciones básicas identifica si el titular del certificado es una entidad emisora o no, y si lo es debe indicar la longitud máxima de los caminos de certificación válidos que incluyen el certificado.
- ***Netscape Cert Type:*** Indica el fin o los fines para los que se puede utilizar el certificado. Esta extensión está descontinuada, pero se tiene en cuenta para maximizar la compatibilidad con navegadores basados en Netscape (por ejemplo Mozilla Firefox). Los usos que ofrece esta extensión son: Certificado de cliente SSL, certificado de servidor SSL, Certificado S/MIME, Certificado para firmar, certificado SSL de CA, certificado S/MIME de CA para firmar.
- ***Key Usage:*** Su uso es igual a la extensión anterior, pero ha recibido el espaldarazo de la IETF. Los posibles usos del certificado son: firma digital, no repudio, cifrado de claves, cifrado de mensajes, establecimiento de claves asimétricas, firmado de certificados, firmado de CRLs, solo para cifrar y por último solo para descifrar.
- ***Netscape Comment:*** Agrega un comentario al certificado.
- ***Subject Key Identifier:*** Usado para ayudar a la construcción de caminos de certificación.
- ***Authority Key Identifier:*** Usado para ayudar a la construcción de caminos de certificación entre las autoridades de certificación.
- ***Extended Key Usage:*** Indica uno o más fines para los que se puede usar la clave pública del certificado. En general, esta extensión aparecerá sólo en certificados de entidad final. Los posibles usos reglamentados son: autenticación de servidor web, autenticación de clientes, algoritmo de cifrado a usar, algoritmo de firmado digital a usar, firmado de código fuente para programas, protección de correos electrónicos, no repudio en e-mails, firma con marca de tiempo, firma de consultas OCSP.

Para finalizar, las extensiones más comunes de los archivos que contienen los certificados son: ".CER", ".PEM", y ".DER".

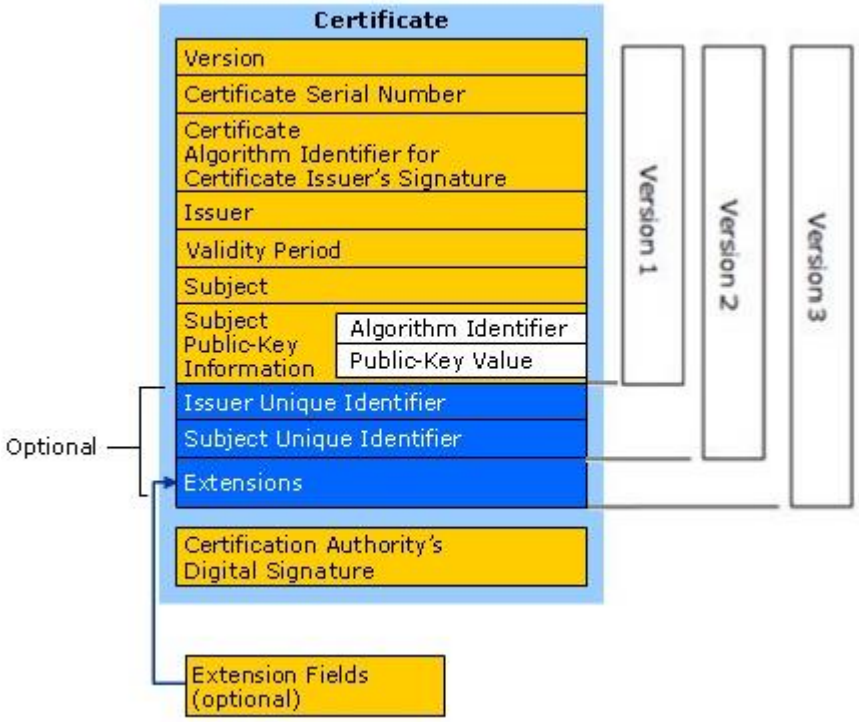
2.4.4 Estándares

Si todas las partes interesadas tienen un certificado digital con una estructura interna distinta, sería casi imposible llegar a la correcta gestión de una PKI, por ello se han desarrollado diversos estándares. A continuación se mencionarán los más usados.

2.4.4.1 Recomendación X.509

Esta recomendación [52] define y estandariza un formato de certificados general y flexible. La amplia difusión de este formato muestra su idoneidad para diversos ambientes y su disponibilidad como estándar internacional justo en el momento en que un gran número de proveedores iniciaban la implementación de sus productos.

Figura 2-6 Estructura del estándar X.509 para un certificado digital



Fuente: How Certificates Work [53].

La utilidad real de X.509 proviene de su poderoso mecanismo de extensiones definido para la versión 3 de los certificados y la versión 2 de las CRLs. Así, el contenido de los certificados y las CRLs puede ser adaptado fácilmente a ambientes específicos, restringiendo o no su utilidad en otros ambientes.

Aunque éste es un estándar internacional, X.509 ha continuado evolucionando en algunos aspectos. Los problemas encontrados a nivel operacional han sido (y continúan siendo) tratados en el estándar a través de reportes formales de defectos y procesos de resolución. También se le han incorporado algunas modificaciones para aclarar el texto o para proporcionar detalles adicionales; un ejemplo de esto es la definición y uso de los certificados de atributos para la gestión de privilegios. [54]

2.4.4.2 Pretty Good Privacy (PGP)

PGP (Privacidad Bastante Buena - *Pretty Good Privacy*) se trata de un proyecto iniciado a principios de los 90 por Phill Zimmerman. PGP está definido en el RFC 4880 y combina técnicas de criptografía simétrica y asimétrica [50]. Esta combinación permite aprovechar lo mejor de cada uno. Mientras que en la recomendación X.509 la gestión de los certificados digitales es centralizada, PGP se basa en la creación de redes de confianza [55]. En una red de confianza no se utiliza como tal una CA, en su lugar cada entidad de la red certifica el vínculo entre las claves públicas y sus propietarios para otras entidades, es decir, cada entidad firma el certificado de la otra entidad a la cual le tiene confianza, así se entreteje una red de confianza lo que en el estándar PGP se conoce formalmente como Web de Confianza; claro está, una entidad también evalúa a su contraparte asignándole un nivel de confianza que puede ser [55]:

- **Confianza Nula:** Establece que una entidad no es de confianza para firmar otros certificados, por lo que los certificados firmados por esta no serían válidos.
- **Confianza Parcial:** Implica que los certificados firmados por una entidad con este nivel, deben estar por lo menos firmados por otra entidad del mismo nivel para poder considerar un certificado como válido.
- **Confianza Total:** Los certificados firmados por estas entidades son considerados válidos.

Aunque PGP soporta los certificados X.509, también posee su propia estructura de certificados digitales, los campos son los siguientes [55]:

- **Número de Versión:** Identifica la versión de PGP utilizada para crear la clave asociada con el certificado.
- **Clave del Titular:** En este campo se agrega la clave pública del titular.
- **Datos Personales:** Nombre del titular del certificado.
- **Firma Digital del Titular:** Se le conoce como auto-firmado, es con el fin de demostrar la posesión de la clave privada.
- **Periodo de Validez:** Incluye el rango de fechas entre las cuales el certificado es válido.
- **Algoritmo de Cifrado Simétrico:** Indica el algoritmo simétrico con el que el propietario del certificado prefiere tener la información cifrada.

2.4.4.3 Public-Key Cryptography Standards (PKCS)

El grupo de estándares de criptografía de clave pública, mejor conocido como PKCS (por su sigla en inglés *Public Key Cryptography Standard*), a lo largo del tiempo se ha convertido en el estándar para el desarrollo y uso de técnicas de cifrado asimétrico usando el algoritmo RSA.

PKCS actualmente, consta de quince normas que definen el cifrado, la sintaxis y las interfaces para la utilización de diversas herramientas criptográficas. Los estándares PKCS en uso son [56]:

- **PKCS#1: *Estándar Criptográfico RSA*:** Define las propiedades matemáticas, el formato de las claves asimétricas RSA (formato ASN.1[33]), el algoritmo básico, el esquema de codificación para realizar el cifrado y descifrado con las claves públicas y privadas RSA, además define la creación y verificación de firmas digitales.
- **PKCS#2: *Estándar de Intercambio de Claves Diffie-Hellman*:** Define un protocolo que permite que dos partes establezcan conjuntamente una clave compartida de sesión (clave simétrica) usando la criptografía asimétrica a través de un canal de comunicaciones inseguro.
- **PKCS#5: *Estándar Criptográfico Basado en Contraseña*:** Recomendaciones para la implementación de criptografía basada en contraseñas, que cubren las funciones de derivación de claves, esquemas de cifrado, esquemas de autenticación de mensajes, y la sintaxis ASN.1 que identifica las técnicas.
- **PKCS#6: *Estándar sobre la Sintaxis de los Certificados Extendidos*:** Define extensiones para el estándar X.509v1, pero un certificado X.509v3 soporta extensiones definidas en su propio estándar, por ello esta norma está obsoleta.
- **PKCS#7: *Estándar sobre la Sintaxis de un Mensaje Cifrado*:** Conjunto de estándares para firmar y cifrar mensajes, normalmente usado para demostrar la autenticidad y privacidad de los datos contenidos en dicho mensaje.
- **PKCS#8: *Estándar sobre la Sintaxis de la Información de la Clave Privada*:** En él se describe la sintaxis para el alojamiento seguro de una clave privada, ya sea en claro o cifrada.
- **PKCS#9: *Estándar sobre los Tipos de Propiedades en Otros Estándares*:** Define los atributos ASN.1 para PKCS #6, PKCS #7, PKCS #8, y PKCS #10.
- **PKCS#10: *Estándar sobre la Sintaxis de la Solicitud de Firma de Certificados*:** En él se define el formato de los mensajes enviados a una

[33] *Abstract Syntax Notation One (ASN.1):* Es un estándar y notación que describe las reglas y estructuras para la representación, codificación, transmisión y decodificación de datos en las telecomunicaciones y las redes de computadores. ASN.1 define la sintaxis abstracta de información, pero no restringe la manera en que la información se codifica. [114]

autoridad de certificación para solicitar la firma de una clave pública. Usualmente, la RA es quien envía estos mensajes, también conocidos como CSR (*Certificate Signing Request*).

- **PKCS#11: *CRYPtographic TOKen Interface (CRYPTOKI):*** Define una API[34] (CRYPTOKI) para los dispositivos que pueden realizar operaciones criptográficas y procesar la información (claves y certificados) relacionada con los procesos de firma y cifrado, donde por características del hardware la clave privada no pueda extraerse, ni copiarse.
- **PKCS#12: *Estándar de Intercambio de Información Personal:*** Define un formato de fichero (PFX) usado comúnmente para almacenar claves privadas con su certificado de clave pública protegido mediante clave simétrica.
- **PKCS#13: *Estándar de Criptografía de Curvas Elípticas***: Especifica muchos aspectos de la criptografía basada en la teoría de curvas elípticas, que todavía está en fase de desarrollo.
- **PKCS#15: *Estándar sobre el Formato de la Información en Dispositivo Criptográfico***: Propone un modelo generalizado para el acceso a los identificadores criptográficos, con el fin de facilitar aún más la compatibilidad entre los fabricantes de hardware. Se utiliza en tarjetas inteligentes.

2.5 SSL/TLS

SSL (Capa de Conexión Segura - *Secure Socket Layer*-) es un protocolo de carácter general, diseñado en 1994 por la empresa *Nestcape Communications Corporation*, y basado en la aplicación conjunta de criptografía simétrica, criptografía asimétrica, certificados digitales y firmas digitales para crear un canal seguro de comunicación a través de Internet. La última versión de SSL es la 3, que es considerada insegura en la actualidad pues ingenieros de Google encontraron graves fallos en este protocolo [57].

SSL y TLS (Seguridad de Capa de Transporte - *Transport Layer Security* -) proporcionan autenticación y privacidad de la información entre extremos, sobre Internet, mediante el uso de criptografía [50].

[34] *API (Application Programming Interface - Interfaz de Programación de Aplicaciones):* Conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizados por otro software como una capa de abstracción, para brindar la capacidad de comunicación entre componentes de software y hardware [115].

2.5.1 Diferencias entre SSL y TLS

TLS es el sucesor de SSL, y algunas de las modificaciones que se realizaron a TLS para mejorarlo con respecto a SSL fueron [58]:

- La construcción del MAC (*Message Authentication Code*) fue modificada ligeramente apareciendo HMAC (*Hash Message Authentication Code*).
- Las implementaciones requirieron incluir soporte para el protocolo de intercambio de claves Diffie-Hellman, para el estándar de firma digital y para el algoritmo Triple-DES.
- Soporte de funciones SHA2, exactamente SHA256 y SHA384.
- Adiciona estandarización en el orden de los mensajes, más mensajes de alerta y más bloques de relleno a los bloques codificados

2.5.2 Funcionamiento

El protocolo SSL se compone de dos partes: un protocolo de negociación (SSL *Handshake Protocol*) y un protocolo de registro (*SSL Record Protocol*). El protocolo de registro especifica la forma de encapsular los datos transmitidos y recibidos, incluyendo los de negociación. Los datos pueden ser comprimidos, cifrados y empaquetados con un código de autenticación de mensajes (MAC-*Message Authentication Code*). Cada registro tiene un campo de *content_type* que especifica el protocolo de nivel superior que se está usando.

Cuando se inicia la conexión, el nivel de registro encapsula el protocolo de negociación (*handshake*), que tiene el campo *content_type*.

El protocolo SSL admite autenticación mutua y simple, en la primera el cliente puede autenticar al servidor y este también lo puede hacer con el cliente, mientras que en la segunda solo un ente, generalmente el servidor, presenta el certificado que le sirve para demostrar su identidad ante el cliente.

Se envían y reciben los siguientes mensajes durante el *handshake* [45]:

- El cliente envía un mensaje *ClientHello* especificando una lista de algoritmos criptográficos, métodos de compresión y la versión del protocolo SSL más alta permitida. Éste también envía bytes aleatorios que serán usados más tarde (llamados *Challenge* o desafío del cliente). Además puede incluir el identificador de la sesión.
- El servidor envía un mensaje *ServerHello*, en el que elige los parámetros de conexión a partir de las opciones ofertadas con anterioridad por el cliente.
- Cuando los parámetros de la conexión son conocidos, cliente y servidor intercambian certificados (dependiendo de las claves públicas de cifrado seleccionadas). Estos certificados son actualmente X.509, pero hay también un

borrador especificando el uso de certificados basados en OpenPGP.5 y OpenSSL.

- Cliente y servidor negocian una clave secreta (simétrica) común llamada *master secret*, posiblemente usando el resultado de un intercambio Diffie-Hellman, o simplemente cifrando una clave secreta con la clave pública que es descifrada con la clave privada de cada uno. Todos los datos de claves restantes son derivados a partir de este *master secret* (y los valores aleatorios generados en el cliente y el servidor), que son pasados a través una función pseudoaleatoria cuidadosamente elegida.

2.6 Ataques

A continuación se describen los ataques que se usarán en este trabajo de grado, los cuales fueron acotados en el anteproyecto.

2.6.1 Ataques Pasivos

Un ataque pasivo intenta conocer o hacer uso de información del sistema, pero no afecta a los recursos del mismo. Los ataques pasivos se dan en forma de ***escucha*** o de observación no autorizada de las transmisiones. El objetivo del oponente es obtener información que se esté transmitiendo. Son muy difíciles de detectar, ya que no implican alteraciones en los datos. Contra estos ataques se debe poner más énfasis en la prevención que en la detección. Para su posible solución es recomendado utilizar cifrado de la información.

2.6.1.1 Sniffing

Un *sniffer* es un programa que trabaja en conjunto con la tarjeta de interfaz de red, para obtener indiscriminadamente todo el tráfico que esté dentro del umbral de audición del sistema que escucha. Y no sólo el tráfico que vaya dirigido a una tarjeta de red, sino a la dirección de difusión de la red 255.255.255.255 (es decir a todas partes) [59].

En la Figura 2-7 se ilustra el proceso del ataque pasivo de monitoreo de la información.

Figura 2-7 Monitoreo no autorizado

Fuente: Categorías de ataques [60].

2.6.2 Ataques Activos

Intentan alterar los recursos del sistema o afectar su funcionamiento. Implican alguna modificación del flujo de datos o la creación de un flujo falso

2.6.2.1 Robo de Identidad

El robo de identidad es la obtención ilegal de contraseñas, claves o cualquier otra credencial que utilice el usuario para identificarse ante un sistema. Existen varios métodos para obtener información personal [61]:

- **Correos Falsos:** Esta técnica le permite a un atacante hacerse pasar por una organización, banco o empresa verdaderos para obtener información de acceso a algún recurso que usted utilice en esa organización, banco o empresa.
- **Personal:** Cualquier persona maliciosa podría obtener información que escuchó o vio que le permita el acceso a algún recurso valioso.
- **Ataque Organizado**: Cualquier atacante podría intentar violar la seguridad de un banco, empresa u organización para obtener información personal de los clientes y luego acceder a algún recurso de esa empresa.
- **Ataque a Servidores de Almacenamiento de Información Online:** El atacante puede tratar de obtener datos de un servidor de almacenamiento de datos en la nube; obteniendo contraseñas, números de identificación, cuentas bancarias, etc.
- **Ataque por Fuerza Bruta:** Prueba todas las combinaciones posibles hasta dar con la clave: Este tipo de ataque en muy primitivo, pero puede ser eficaz si se logra tener alta velocidad de procesamiento para probar muchas claves por segundo, por ello, es la principal razón del aumento de la longitud de

contraseñas conforme a los avances tecnológicos donde cada día salen dispositivos con mayores velocidades para este cracking.

- **Ataque por Diccionario:** Consiste en averiguar una contraseña probando todas las palabras de un diccionario determinado. Este tipo de ataque suele ser más eficiente que un ataque de fuerza bruta, ya que muchos usuarios suelen utilizar una palabra existente en su lengua como contraseña para que sea fácil de recordar, lo cual no es una práctica recomendable.

2.6.2.2 Suplantación de Identidad

Es la apropiación de la identidad de una persona: hacerse pasar por esa persona, asumir su identidad ante otras personas en público o en privado, en general, para acceder a ciertos recursos o la obtención de créditos y otros beneficios en nombre de esa persona. Por otro lado, la suplantación de identidad también es utilizada con el fin de perjudicar a una persona, es decir, difamarlo o manchar su nombre con diversos fines que el criminal busque. El caso más común hoy en día se da cuando un atacante, por medios informáticos o personales, obtiene información personal y la utiliza ilegalmente (ver Figura 2-8) [61]. El ataque de robo de identidad es usualmente el paso previo para realizar un ataque de suplantación de identidad.

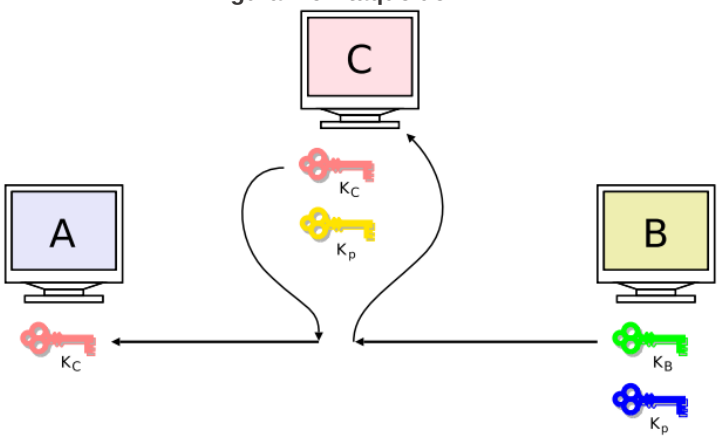
Figura 2-8 Suplantación de Identidad

Fuente: Fundamentos y categorías de ataques [60].

2.6.2.3 Hijacking o Man In The Middle (MITM)

Este es un ataque activo, porque el atacante intenta modificar la información que se intercambia, en este caso, entre el votante y el centro electoral, secuestrando la sección que inició el usuario, es decir, se hace pasar por el usuario ante el servidor, y por el servidor ante el usuario, realizando los procesos como a él le interese, en este caso, podría generar votos a su conveniencia.

Figura 2-9 Ataque de MITM



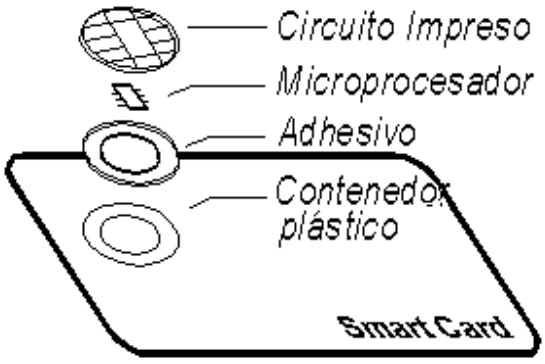
Fuente: Criptografía y seguridad en Computadores [62].

En la **¡Error! No se encuentra el origen de la referencia.** se muestra una forma en donde el ataque activo MITM se evidencia. El atacante, computador C, le envía una clave pública falsa al computador A, el cual cree que es el computador B

2.7 Tarjetas Inteligentes

La característica que distingue a las tarjetas inteligentes de otras tarjetas, radica en el hecho de que contienen un microprocesador capaz de darles un poco de "inteligencia", lo que permite no sólo el mero almacenamiento de los datos, sino también su procesamiento. La Figura 2-10 muestra la estructura física de la tarjeta.

Figura 2-10 Estructura física de una tarjeta inteligente



Fuente: Philips DX smart card reference manual **[63]**.

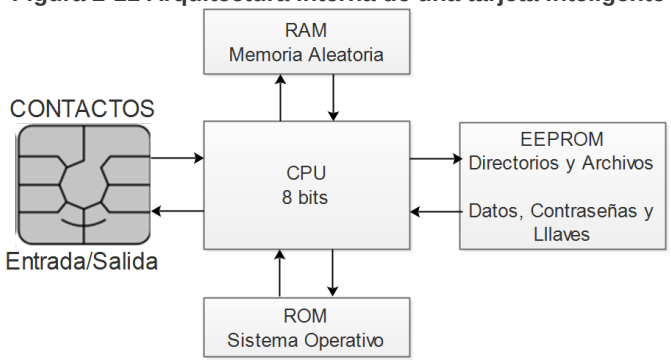
En la práctica, el procesador realiza operaciones con los datos directamente dentro de la tarjeta, y esta es una propiedad que no debe subestimarse, ya que proporciona una enorme contribución a la seguridad para las operaciones con

datos sensibles (datos personales del titular de la tarjeta, certificados de clave pública, claves privadas, etc).

2.7.1 Arquitectura Interna

En la Figura 2-11 se visualizan los componentes de una tarjeta inteligente.

Figura 2-11 Arquitectura interna de una tarjeta inteligente



Fuente: Internal Architecture SC **[63]**.

- **CPU:** Es el procesador central, tipo CISC[35] de 8, 16 o 32 bits, con una velocidad de procesamiento de 3,5 o 5 MHz.
- **ROM:** Es una memoria de sólo lectura cuyas dimensiones varían de 2 a 64 Kbytes. Contiene el sistema operativo de la tarjeta.
- **EEPROM:** Tiene una capacidad de alrededor de 128 Kbytes. Sirve para almacenamiento masivo, así que puede contener datos de los usuarios, claves, certificados, archivos, aplicaciones, etc.
- **RAM:** En la memoria de acceso aleatorio se cargan todas las instrucciones que se ejecutan en la CPU. Rango de memoria de 128 a 1024 bytes.
- **Entrada/Salida:** Interfaz para el intercambio de datos con el exterior.

La estructura que se presentó es básica (de referencia). Dependiendo de las funciones para las que están destinadas algunas tarjetas pueden tener cambios en el esquema descrito. Por ejemplo, en esta tesis se usan tarjetas criptográficas, y los algoritmos deben realizar cálculos complejos por lo que necesitan un procesador más, llamado coprocesador criptográfico.

[35] CISC (Complex Instruction Set Computer - Computador con Conjunto de Instrucciones Complejas).

2.7.2 Tipos de Tarjetas Inteligentes

La primera distinción que puede hacerse es acerca de la presencia o ausencia del microchip:

- **Tarjeta de Memoria:** Sirve para almacenamiento de datos, no tiene en su interior un procesador. Contiene un interruptor lógico de seguridad para el acceso a los datos (autenticación).
- **Tarjeta con Procesador:** También llamada ICC (tarjeta con circuito integrado - *Integrated Circuit Card*), según la ISO. Contiene un microchip que puede almacenar, recuperar y manipular datos. Algunas de ellas también tienen un coprocesador para más operaciones complejas (tarjetas criptográficas).

Otra clasificación de las tarjetas inteligentes se puede hacer de acuerdo a la forma de comunicarse con el exterior, entonces se tiene:

- **Tarjetas Inteligentes de Contactos:** La tarjeta accede por medio de contactos colocados en la superficie de la tarjeta.
- **Tarjeta Inteligente sin Contactos:** El dispositivo es capaz de transmitir y recibir ondas electromagnéticas utilizadas para intercambiar datos. Las más populares son las tarjetas inteligentes RFID.

2.7.3 Estándares

Las tarjetas inteligentes, aunque diferentes en arquitectura, deben cumplir con ciertas normas en cuanto a su apariencia física, protocolos de comunicación, controles, etc. Dos organizaciones en particular (Organización Internacional de Estándares -ISO- y la Comisión Electrotécnica Internacional -IEC-) han establecido una serie de referencias para los fabricantes de tarjetas inteligentes.

2.7.3.1 Estándar ISO/IEC 7816

Todas las características básicas de una tarjeta inteligente están documentadas en la norma ISO/IEC 7816, que incluye las siguientes secciones:

- **ISO/IEC 7816-1:** Características físicas.
- **ISO/IEC 7816-2:** Dimensiones y ubicación de los contactos.
- **ISO/IEC 7816-3:** Protocolos de interfaz eléctrica y de transmisión.
- **ISO/IEC 7816-4:** Organización, seguridad y comandos para el intercambio de información
- **ISO/IEC 7816-5:** Sistema y procedimiento de registro de numeración para los identificadores de aplicación.
- **ISO/IEC 7816-6:** Interoperabilidad en los elementos para el intercambio de datos.

- **ISO/IEC 7816-7:** Interoperabilidad en los comandos de la tarjeta (SCQL).
- **ISO/IEC 7816-8**: Comandos para operaciones de seguridad
- **ISO/IEC 7816-9**: Comandos para la gestión de la tarjeta
- **ISO/IEC 7816-10:** Señales electrónicas para operación síncrona.
- **ISO/IEC 7816-11:** Verificación de la identidad personal a través de métodos biométricos.
- **ISO/IEC 7816-15**: Aplicación de información criptográfica.

Estas normas están en constante evolución como consecuencia de las nuevas tecnologías, o el uso de soluciones particulares creadas por los fabricantes, que luego se pueden convertir en estándar común.

Como ya se sabe, los estándares o normas proporcionan pautas a seguir, sin embargo, se recomienda que cada fabricante siga estas normas, pero no es obligación, por esto, no es raro encontrar tarjetas especiales que implementan funciones no relacionadas con las normas anteriormente descritas (especialmente en el campo de la criptografía) donde un fabricante crea métodos propios para la mejor solución a un problema dado.

2.7.4 Funcionamiento

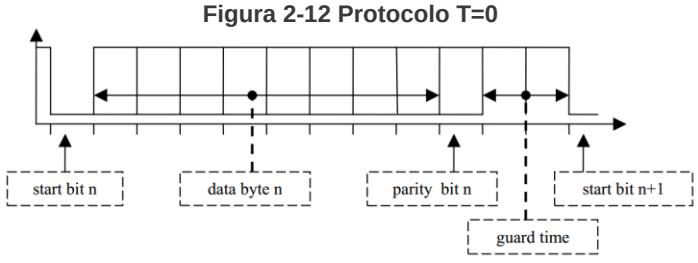
2.7.4.1 Protocolo de transmisión

Actualmente, las tarjetas inteligentes utilizan dos protocolos de transmisión conocidos como $T = 0$ y $T = 1$. Son dos protocolos de comunicaciones en serie asíncronos *half-dúplex*.

2.7.4.1.1 Protocolo $\mathbf{T = 0}$

El $T = 0$ es quizás el más común, porque es ampliamente usado en la estándar GSM. Es un protocolo de comunicación **orientado a byte**, en el que cada unidad de transmisión está precedida por un bit de inicio en «0 lógico» y seguida por un bit de paridad, luego es continuada por un tiempo de guarda que consiste en dos bits en «1 lógico» para anunciar el fin de la transmisión (ver Figura 2-12) [64].

Figura 2-12 Protocolo T=0



Fuente: ***Smart Card Reader Software Implementation*** **[65].**

2.7.4.1.2 Protocolo T = 1

Es un protocolo **orientado a bloque** más complejo que $T = 0$. Se basa en una capa física similar a $T = 0$, pero sin comprobación de paridad. La comprobación de errores se realiza en todo el bloque de datos intercambiado entre la tarjeta inteligente y el IFD (*Interface Device*- Dispositivo de Interfaz), realizando de esta manera la separación de las capas y haciendo $T = 1$ más adecuado para el transporte de datos cifrados [64].

La estructura de un bloque se ilustra en la Figura 2-13.

Figura 2-13 Estructura de un bloque T=1

Prologue field			***Information field***	***Epilogue field***
node address **NAD**	protocol control byte **PCB**	length **LEN**	**APDU**	**EDC**
1 byte	1 byte	1 byte	0... 254 byte	1... 2 byte

Fuente: ***Smart Card Reader Software Implementation*** **[65].**

- **NAD:** Permite controlar la tensión Vpp (2 bits) y codificar las direcciones lógicas de la fuente y el destino de los datos (cada una de 3 bits).
- **PCB:** Contiene la información de varios protocolos de gestión.
- **LEN:** Contiene la longitud en bytes del siguiente campo.
- **APDU (Unidad de Datos de Protocolo de Aplicación - *Application Protocol Data Unit*):** Contiene los datos hacia o desde la capa de aplicación del modelo OSI.
- **EDC (Código de Detección de Error - *Error Detection Code*):** Contiene un código de corrección de errores calculado sobre los campos anteriores.

Con las mismas condiciones generales que $T = 0$ (frecuencia de reloj, y otros parámetros) un análisis de rendimiento indica que la velocidad de transferencia obtenida con $T = 0$ y $T = 1$ son muy similares [66]. $T = 1$ tiene un mecanismo para detectar y corregir errores más fiable, ya que, crea una separación entre niveles, preferibles en determinadas circunstancias (mensajería segura) [67].

2.7.5 Interfaz con las Aplicaciones

La norma PKCS#11 especifica una interfaz de programación de aplicaciones (API), llamada Cryptoki (*Cryptographic Token Interface*), para aquellos dispositivos almacenadores de información y capaces de implementar funciones criptográficas. Cryptoki sigue un enfoque basado en programación orientada a objetos y busca mantenerse la interoperabilidad entre distintos dispositivos y aplicaciones. Es importante señalar que no todos los dispositivos criptográficos son capaces de realizar todas las funciones que ofrece la API. Por tanto, PKCS#11 no especifica

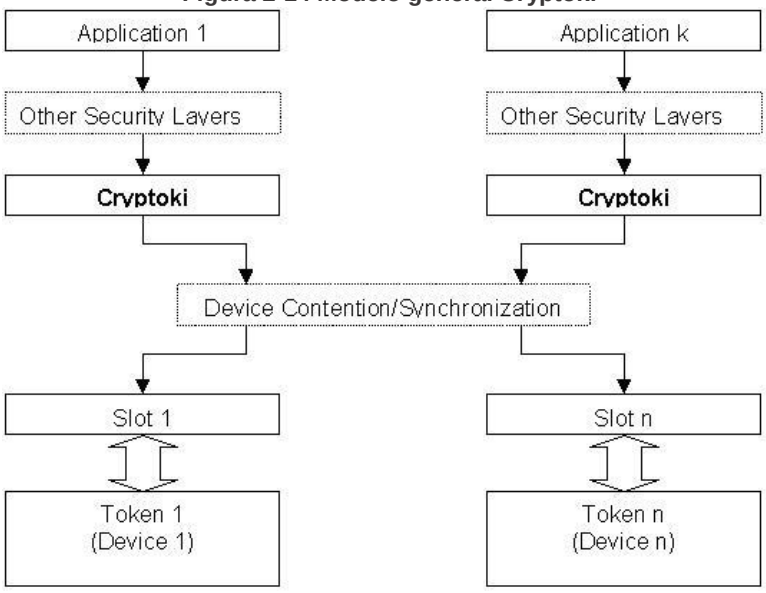
los detalles de la programación, simplemente ofrece una descripción precisa de los tipos de datos y funciones que pueden ser utilizados por un programador.

2.7.5.1 Modelo General de Cryptoki

El modelo general de Cryptoki se ilustra en la Figura 2-14.

Una aplicación no se comunica directamente con el dispositivo físico, lo hace a través de la API Cryptoki accediendo a objetos que representan la abstracción lógica de dicho dispositivo; se pueden destacar dos de estos objetos los *tokens* y los *slots*. Los *tokens* abstraen el dispositivo físico capaz de realizar las operaciones criptográficas (por ejemplo, la tarjeta inteligente), mientras que los *slots*, abstraen los dispositivos físicos lectores. Este modelo es el que hace que todos los dispositivos criptográficos sean lógicamente idénticos, evitando que cada aplicación tenga un protocolo distinto para comunicarse con la tarjeta inteligente, esto es lo que hace posible la interoperabilidad.

Figura 2-14 Modelo general Cryptoki



Fuente: pkcs#11 cryptographic token interface standard [68]

Cryptoki también le permite a una o más aplicaciones conectarse a uno o más *tokens*, gestionando una multi-sesión a través de un dispositivo de contención/sincronización (*Device Contention/Synchronization*).

Cryptoki es implementada usando bibliotecas escritas en ANSI C[36] para que la aplicación pueda conectarse directamente o dinámicamente.

2.7.5.2 Sesiones Cryptoki

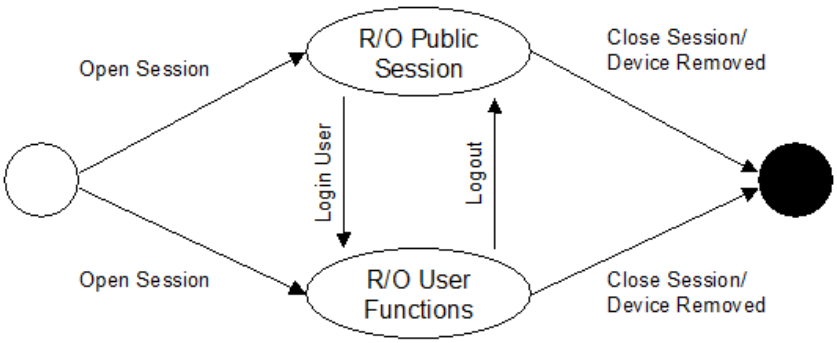
Una vez que la aplicación haya reconocido los *slots* y los *tokens*, se debe iniciar una sesión para que Cryptoki pueda acceder al sistema de archivos, objetos y funciones de la tarjeta inteligente.

La sesión proporciona una conexión lógica entre la aplicación y el *token.* Estas sesiones pueden ser de dos tipos: **R/W** (*Read/Write*) o **R/O** (*Read/Only*). Una aplicación puede crear, leer, escribir y destruir objetos al interior de la tarjeta. Con estas sesiones se imponen permisos a las aplicaciones al usar el sistema de archivos, pues es posible configurar que solo acepte cierto tipo de sesiones, lo que le agrega un *plus* con respecto a la seguridad de la tarjeta.

Una vez que se abre la sesión, la aplicación tiene acceso a todos los objetos públicos y *tokens*. Para acceder a los objetos privados de la tarjeta se necesita autenticar al usuario, para comprobar que es el dueño de la tarjeta. Este procedimiento se hace a través de un inicio de sesión en la que tiene que proporcionar el PIN de la tarjeta. Cuando se cierra una sesión se destruyen todos los objetos creados en su interior, esto también aplica a los objetos que están en uso por otras sesiones.

- **Sesión R/O**: Una sesión de sólo lectura consta de dos estados, y solo puede estar en un estado a la vez, como se ilustra en la Figura 2-15.

Figura 2-15 Estados de sesión R/O



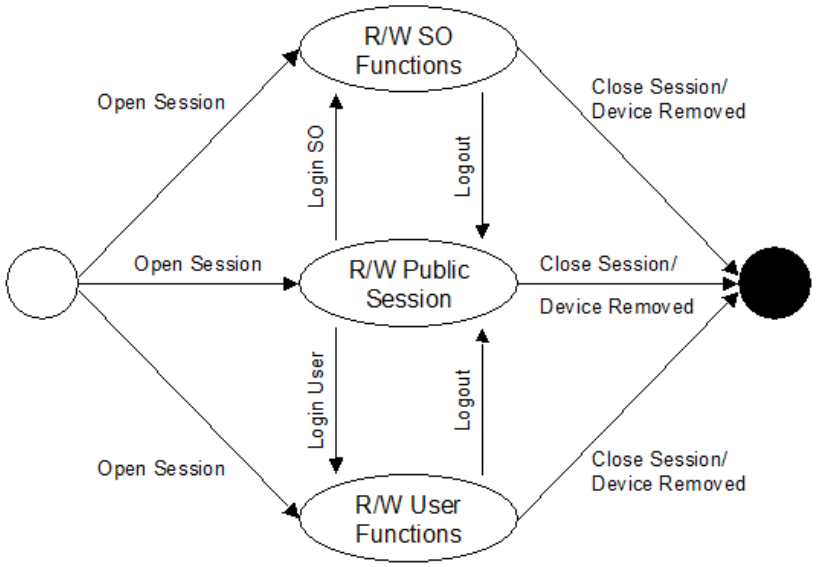
Fuente: PKCS #11 Cryptographic Token Interface Usage Guide Version 2.40 [69].

[36] *ANSI C:* Estándar publicado por el Instituto Nacional Estadounidense de Estándares (ANSI), para el lenguaje de programación C. Se recomienda a los desarrolladores de software en C que cumplan con los requisitos descritos en el documento para facilitar así la portabilidad del código.

Cuando la sesión se abre se puede estar en el estado «R/O *Public Session*» (usuario no autenticado) o el estado «R/O *User Functions*» (usuario autenticado). En el primer caso, la aplicación tiene acceso de sólo lectura a los objetos públicos. En el segundo caso, la aplicación tiene acceso a todos los objetos de la tarjeta (públicos y privados), pero de solo lectura, también se pueden crear objetos pero estos son temporales, son borrados cuando se cierra la sesión.

- **Sesión R/W:** Una sesión de lectura y escritura consta de tres estados.

Figura 2-16 Estados de una sesión R/W



Fuente: PKCS #11 Cryptographic Token Interface Usage Guide Version 2.40 [69].

En base a la Figura 2-16, cuando se inicia la sesión inicialmente se encuentra en ***R/W Public Session***. En este estado el usuario sólo tiene acceso de escritura y lectura a los objetos públicos, por tanto no es necesario autenticarse con el PIN. En el estado de ***R/W User Functions*** se tiene acceso de lectura y escritura a los objetos privados y públicos, es necesaria la autenticación por medio del PIN y es usado por los usuarios normales o promedio. En la sesión ***R/W SO Functions***, la aplicación no podría acceder a los objetos privados, sólo a los públicos; pero a diferencia del primer estado, este puede configurar el PIN del estado anterior; para acceder a este nivel se necesita otro PIN, usado por el *Security Officer*.

Existen dos usuarios en Cryptoki, estos usuarios fueron introducidos en la versión 2.40 de PKCS#11. Un tipo de usuario es el ***Security Officer*** (SO - Oficial de Seguridad), el otro tipo es el ***User*** que se asocia a usuarios normales. Sólo se permite que el ***User*** tenga acceso a los objetos privados, por razones de seguridad, de manera que ninguna otra entidad o persona, a

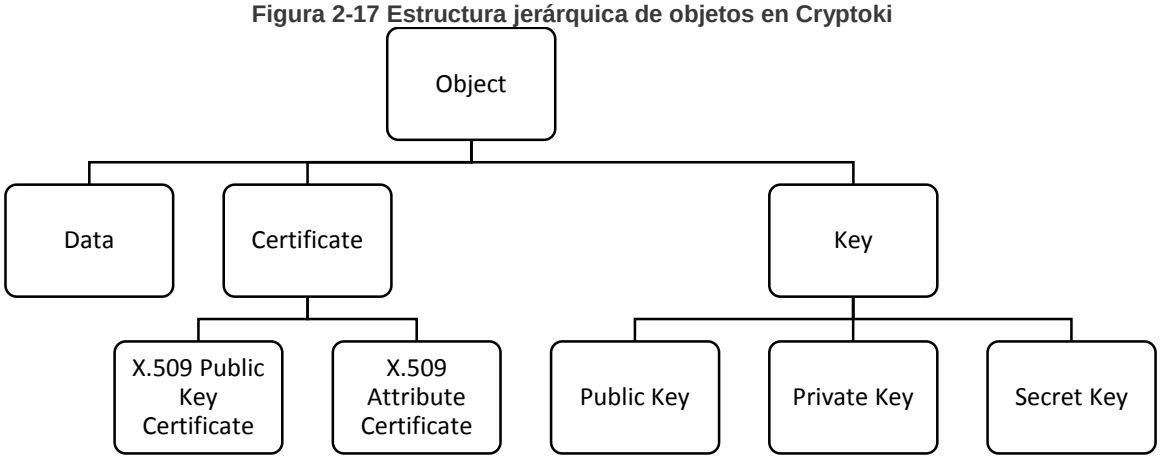
diferencia del usuario propietario, tenga acceso a las claves privadas que son almacenadas en los objetos privados. El usuario SO tiene el rol de acceder a los objetos públicos, inicializar un *token*[37], establecer por primera vez el PIN del usuario normal y cambiarlo con el consentimiento del usuario, pues solicita el PIN antiguo.

- **Eventos de Sesión :** Una sesión puede cambiar su estado actual, como se aprecia en la Figura 2-15 y Figura 2-16. Los eventos responsables del cambio en el estado de la sesión son:
  - ***Login SO*** (*Security Officer*): Autenticación del Administrador.
  - ***Login User***: Autenticación del usuario.
  - ***Logout***: Cierre de sesión por la aplicación, cambia a una sesión pública ya sea que esté en *SO* o *User.*
  - ***Close Session***: La aplicación cierra la sesión actual o todas las sesiones abiertas en el *token*, es decir, la tarjeta.
  - ***Device Removed***: La aplicación desconecta el *token* del *slot*, es decir, desconecta la tarjeta del lector.

2.7.5.3 Objetos

El *token* contiene objetos y funciones de cifrado. Cryptoki define tres clases de objetos: **dato**, **certificado** y **clave.** La jerarquía de objetos se muestra en la Figura 2-17.

Figura 2-17 Estructura jerárquica de objetos en Cryptoki



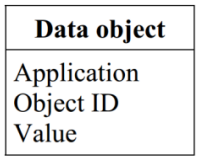
Fuente: Autor

- **Objeto Dato** :Un objeto dato (*data*) contiene *inf*ormación acerca de las aplicaciones o cualquier otra información relacionada al usuario, ya sea el

[37] La inicialización del *token* sólo se realiza cuando la tarjeta es insertada por primera vez en un lector, en otras palabras, la tarjeta está nueva.

nombre del titular, libros favoritos, información de contacto, etc. Suele ser privado o público según la aplicación. Se define en Cryptoki con los atributos que se muestran en la Figura 2-18.

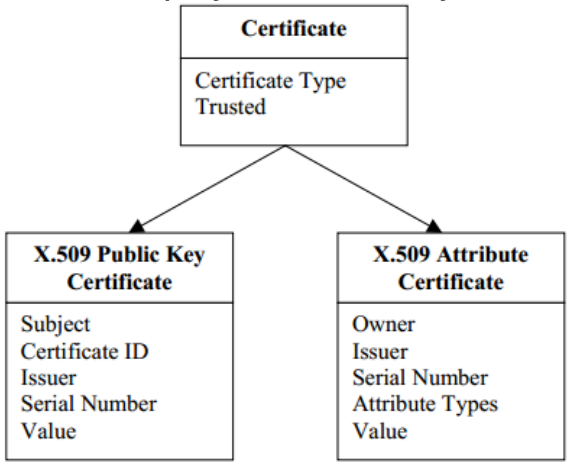
Figura 2-18 Atributo de un objeto Data



Fuente: PKCS #11 Cryptographic Token Interface Usage Guide Version 2.40 [69].

- **Objeto Certificado:** Este objeto puede almacenar certificados digitales junto con sus atributos. Generalmente es un objeto público. En la Figura 2-19 se muestran los atributos de cada una de las subclases de un objeto certificado.

Figura 2-19 Jerarquía y atributos de un objeto *certificate*



Fuente: PKCS #11 Cryptographic Token Interface Usage Guide Version 2.40 [69].

- **Objeto Clave**: Contiene una clave criptográfica, que puede ser pública, privada o secreta. Debe ser obligatoriamente **privado**. Estas claves de cifrado permiten realizar una autenticación. Sus diferentes subclases, con los respectivos atributos, se muestran en la Figura 2-20.

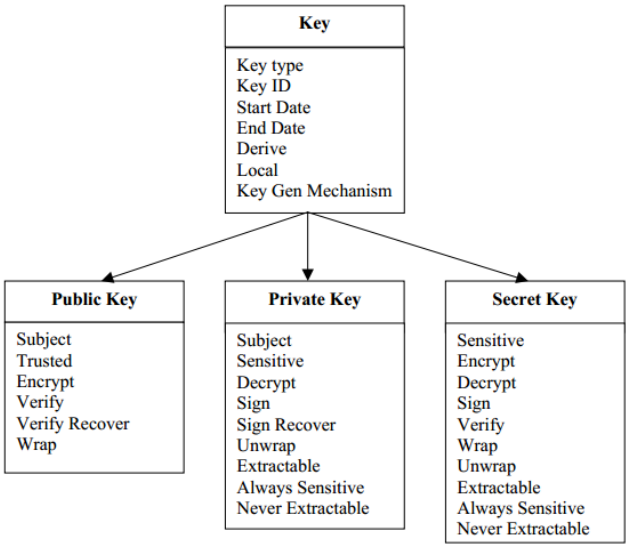
Cryptoki también provee otros tipos de objetos, que proporcionan información adicional sobre, por ejemplo, las características de hardware del dispositivo y los algoritmos criptográficos soportados, los cuales se muestran en la Figura 2-21.

2.7.5.4 Consideraciones en Seguridad de Cryptoki

Cryptoki como API para dispositivos criptográficos proporciona la base para la seguridad dentro de un sistema informático o de comunicaciones. Las dos características principales de la API para facilitar esta tarea son:

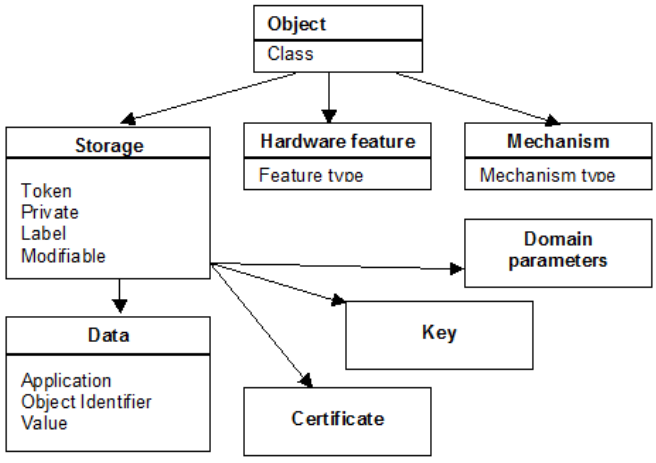
- El acceso a los objetos privados, funciones criptográficas y certificados de usuario requiere un PIN. Así que la mera posesión del dispositivo criptográfico (tarjeta inteligente) no es suficiente para usarlo, pues también se debe saber el PIN.

Figura 2-20 Jerarquía y atributo de un objeto Key



Fuente: PKCS #11 Cryptographic Token Interface Usage Guide Version 2.40 [69].

Figura 2-21 Jerarquía completa de objetos definidos en el estándar Cryptoki



Fuente: PKCS #11 Cryptographic Token Interface Usage Guide Version 2.40 [69].

- Se puede garantizar una protección adicional para las claves privadas o secretas mediante la designación de sus atributos como «sensibles», es decir, el campo «*Sensitive*» en valor «true» lo que significa que no puede ser develado en claro. También se puede establecer «Extractable» en «false» para que nunca el objeto pueda ser extraído de la tarjeta.

Se espera que el acceso a los objetos de tipo sensible, privado, no extraíble sea muy difícil, excepto para la interfaz Cryptoki. Los dispositivos criptográficos generalmente están equipados con un entorno protegido (por ejemplo, una memoria protegida) para almacenar datos de este tipo, pero algunos no tienen esta posibilidad, por lo que deben proteger sus objetos de otra manera. Una solución consiste en cifrar los objetos con una «clave maestra» que se deriva del PIN del usuario.

Pero no es solo proteger los objetos privados, también se deben considerar todas las aristas posibles como, por ejemplo, en el sistema operativo que alberga la aplicación usada con la tarjeta inteligente, es posible que otras aplicaciones intercepten la información que viaja de la aplicación genuina a la tarjeta, pudiéndose obtener el PIN mientras es gestionado por el sistema operativo, o por el canal de comunicación entre la tarjeta y la API; estos ataques pueden tener consecuencias nefastas, como podría ser el pirateo de una sesión y realizar cualquier tipo de operación con el *token* interceptado. No obstante, todo no pueden ser malas noticias, cuando se cambia el atributo de un objeto, ya sea a privado, sensible o no extraíble, la gran mayoría (por no decir todas) las tarjetas criptográficas, suelen copiar ese objeto a una ubicación donde las características del hardware no permiten ciertas operaciones como la sustracción, copia, etc., por tanto, sería muy difícil clonar una tarjeta inteligente.

Por último, se puede concluir que Cryptoki no garantiza la seguridad de todo el sistema, sino que proporciona una buena y sólida base para el desarrollo de aplicaciones que hacen uso de los dispositivos criptográficos.

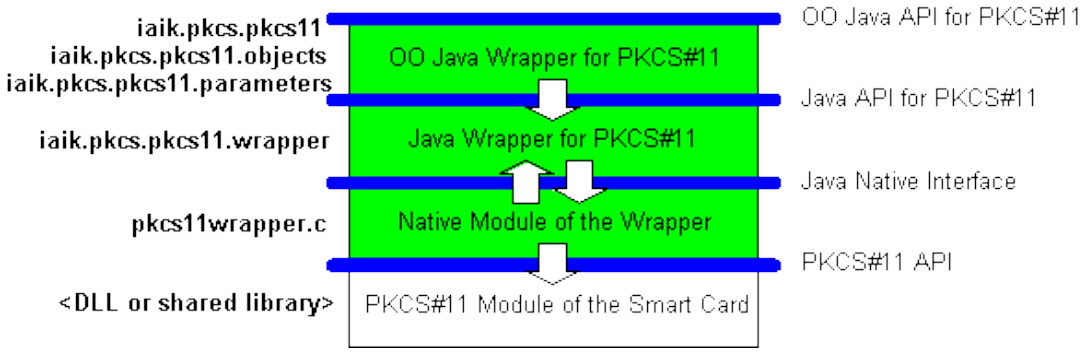
2.7.6 Proveedor de Seguridad IAIK

Es un proveedor de bibliotecas de seguridad, capaces de ofrecer una amplia variedad de funciones criptográficas e implementa una API para el acceso a las tarjetas inteligentes.

En este trabajo se utiliza la implementación basada en Java del estándar PKCS#11 que desarrolló el *Institute for Applied Information Processing and Communications* (IAIK) de la universidad austriaca *Graz University of Technology.* El conjunto de funciones criptográficas que proporciona IAIK son accesibles a

través de las APIs Java JCA[38] (*Java Cryptography Architecture*) y JCE (*Java Cryptography Extention*). El acceso al dispositivo criptográfico se lleva a cabo utilizando las bibliotecas del estándar PKCS#11 (Cryptoki). Cryptoki está escrito originalmente en el lenguaje C y crea un *token* para representar lógicamente a los dispositivos criptográficos. La parte responsable de comunicarse con el *token* es llamada *Wrapper,* utilizando las funciones de Java JNI[39] (*Java Native Interface*). La Figura 2-22 muestra las capas del software IAIK.

Figura 2-22 Estructura del software IAIK



Fuente: IAIK PKCS#11 Wrapper [70].

La capa más baja está ocupada por las bibliotecas de enlace dinámico (DLL – *Dynamic Link Library*) que provee el fabricante del dispositivo criptográfico siguiendo el estándar PKCS#11. La capa inmediatamente superior es necesaria para implementar la capa JNI que permite utilizar las funciones de C por códigos escritos en Java. La capa *JAVA API for PKCS#11* proporciona todos los nombres de las clases, estructuras de datos y métodos en Java correspondientes a la contraparte del PKCS#11 escrito en C. Por último, la capa *OO Java API for PKCS#11* es la encargada de crear la infraestructura de programación orientada a objetos para las aplicaciones que se van a desarrollar.

Para cerrar esta sección, las bibliotecas de IAIK se adaptan al estándar original de Cryptoki, que está escrito en C, para poder ser usadas en aplicaciones Java.

[38] *JCA:* Contiene una arquitectura de tipo "proveedor" con un conjunto de APIs para firmas digitales, resúmenes de mensaje (*hash*), certificados y validación de certificados, cifrado (simétrico/bloque asimétrico/cifrado de flujo), generación de claves privadas y públicas, generación segura de números aleatorios, entre otras. Estas APIs permiten a los desarrolladores integrar fácilmente la seguridad en sus aplicaciones con solo el llamado de una función [121].

[39] *JNI:* Es un *framework* que permite que un programa escrito en Java y ejecutado en la máquina virtual java (JVM) pueda interactuar con programas escritos en otros lenguajes como C, C++ y ensamblador [122].

Capítulo 3

DISEÑO DE LA INFRAESTRUCTURA DE CLAVE PÚBLICA (PKI)

3.1 Diseño de la Aplicación de Registro de Usuarios

La Aplicación de Registro de Usuarios va a permitir el registro de los estudiantes en la PKI y se les entregará a estos una tarjeta inteligente con su par de claves y su certificado.

3.1.1 Requisitos Funcionales

Un requisito funcional define una función del sistema de software o sus componentes. Los requisitos detrás del diseño de la Aplicación de Registro de Usuarios son.

I. El aplicativo debe autenticar a la Autoridad de Registro:
   a. ¿Quién realiza la operación?: La Autoridad de Registro
   b. ¿Qué datos ingresa?: Inserta la tarjeta inteligente e introduce el PIN
   c. ¿Qué hace sistema?: Le envía un desafío cifrado a la RA y verifica que la respuesta a este sea correcta. Si es correcta concede el acceso, de lo contrario se lo deniega.

II. El aplicativo debe permitir registrar a los estudiantes (usuarios):
   a. ¿Quién realiza la operación?: La Autoridad de Registro

b. ¿Qué datos ingresa? Nombre completo, número de documento de identificación, programa, correo electrónico, sede (país, departamento, ciudad)

c. ¿Qué hace el sistema?: Almacena la información del usuario en la Base de Datos de Usuarios.

d. Extensiones:

i. El aplicativo debe verificar que el usuario no esté ya registrado

III. El aplicativo debe generar un par de claves (pública y privada) para cada usuario:

a. ¿Quién realiza la operación?: La Autoridad de Registro

b. ¿Qué datos ingresa? La autoridad de registro inserta la longitud de clave y la tarjeta.

c. ¿Qué hace el sistema?: Genera el par de claves en una tarjeta inteligente virgen con capacidad para operaciones criptográficas y protege la clave privada con un PIN.

IV. El aplicativo debe generar una solicitud de firma de certificado (CSR) por cada usuario registrado:

a. ¿Quién realiza la operación?: La Autoridad de Registro

b. ¿Qué datos ingresa? Se extrae de la Base de Datos de Usuarios el nombre completo, el programa, el número de documento de identidad, la sede y el correo electrónico del usuario

c. ¿Qué hace el sistema?: Genera la solicitud CSR, la firma con la clave privada de la Autoridad de Registro y la envía a la Autoridad de Certificación.

V. El aplicativo debe emitir un certificado digital a cada usuario registrado

a. ¿Quién realiza la operación?: La Autoridad de Certificación

b. ¿Qué datos ingresa? La solicitud CSR que le envió la Autoridad de Registro, el tiempo de validez del certificado y la clave privada para firmar el certificado.

c. ¿Qué hace el sistema?: Genera el certificado, firmándolo digitalmente con la clave privada de la Autoridad de Certificación y lo envía de regreso a la Autoridad de Registro.

d. Extensiones:

i. El aplicativo debe comprobar previamente la firma de la CSR.

ii. Solo puede emitir certificados digitales cuyas solicitudes CSR provengan de la Autoridad de Registro.

VI. El aplicativo debe publicar el certificado del usuario en el repositorio.

a. ¿Quién realiza la operación?: La Autoridad de Certificación.

b. ¿Qué datos ingresa?: Certificado del usuario

c. ¿Qué hace el sistema?: Coloca el certificado del usuario en un repositorio público para que todos los usuarios tengan acceso a él vía Web.

VII. El aplicativo debe guardar el certificado del usuario en la tarjeta inteligente.

a. ¿Quién realiza la operación?: La Autoridad de Registro.

b. ¿Qué datos ingresa?: El certificado digital del usuario y conecta la tarjeta inteligente al computador

c. ¿Qué hace el sistema?: Importa el certificado digital en la tarjeta inteligente.

VIII. El aplicativo debe permitir revocar los certificados digitales

a. ¿Quién realiza la operación?: La Autoridad de Certificación.

b. ¿Qué datos ingresa?: El número de serie del certificado

c. ¿Qué hace el sistema?: Anula la validez del certificado y de su par de claves, por tanto, no podrá ser usado en una autenticación futura.

d. Extensiones:

i. La Autoridad de Certificación debe verificar previamente la validez de la solicitud de revocación que hace el usuario.

IX. El aplicativo debe colocar los certificados revocados en un repositorio público

a. ¿Quién realiza la operación?: La Autoridad de Certificación.

b. ¿Qué datos ingresa?: La lista de certificados revocados (CRL)

c. ¿Qué hace el sistema?: Almacena la CRL en un sitio público accesible vía Web.

X. El aplicativo debe permitir verificar la validez de los certificados

a. ¿Quién realiza la operación?: Los usuarios (operarios y estudiantes) y la Autoridad de Validación

b. ¿Qué datos ingresa?: El número de serie del certificado

c. ¿Qué hace el sistema?: El servidor OCSP (Autoridad de Validación) verifica si el certificado es válido y envía su respuesta firmada al usuario solicitante.

3.1.2 Requisitos No Funcionales

Los requisitos no funcionales describen las restricciones de la aplicación, por ende son atributos relacionados con la calidad, como:

- **Rendimiento**: El aplicativo de Registro de Usuarios debe ejecutarse en equipos que posean por lo menos 2GB de memoria y una velocidad de procesamiento minina de 2GHz, para así evitar que se trunquen los distintos procesos.
- **Usabilidad:** El aplicativo debe ser fácil de usar para los operarios.
- **Escalabilidad:** La base de datos de usuarios debe poder crecer para albergar todos los usuarios que sean necesarios.
- **Seguridad**
    - Solo personal autorizado podrá ejercer el papel de Autoridad de Registro y Autoridad de Certificación. Estos funcionarios son los únicos que poseerán la tarjeta inteligente con la clave de acceso al sistema para registro de usuarios y generación de certificados.
    - Las aplicaciones estarán ejecutándose en espacios de memoria controlados y supervisados por procesos que se encarguen de eliminar cualquier tipo de información almacenada en la memoria RAM, espacios de intercambio (partición swap[40]) o disco duro por medio del uso de herramientas que provee Java en su *Java Machine Virtual.*
    - La clave privada de los diferentes usuarios (operarios y estudiantes) no podrá ser extraída o copiada de la tarjeta inteligente, solo se podrá acceder a ella para realizar las operaciones criptográficas.
    - Todas las respuestas a las solicitudes de estado de certificado del servidor OCSP deben ser firmadas digitalmente por este.
    - Debido a que las Autoridades de Registro y de Certificación están separadas, la comunicación entre estas debe estar cifrada con SSL. Aunque esta comunicación sea indirecta, pues ambas se

[40] El espacio de intercambio es una zona del disco (un fichero o partición) que se usa para guardar las imágenes de los procesos que no han de mantenerse en memoria física. A este espacio se le suele llamar *swap*, del inglés "intercambiar".

intercomunican usando una base de datos, la comunicación con esta base de datos debe estar cifrada.

- Los usuarios deberán tener sus tarjetas inteligentes en un lugar seguro y no revelar su PIN a nadie
- Si algún usuario considera que su clave privada se vio comprometida, es decir, alguien no autorizado accedió a ella, debe solicitar inmediatamente que el certificado sea revocado
- La longitud mínima de las claves de los usuarios debe ser 1024 bits y la de las claves de los operarios debe ser 2048 bits.

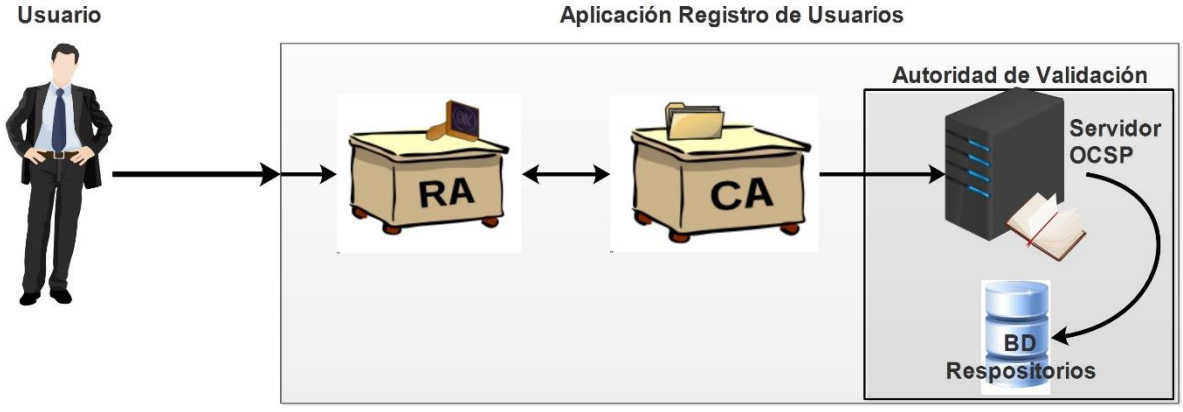
- **Estabilidad**: El aplicativo deberá ser capaz de ejecutar funciones de recuperación en caso de fallos generales, evitando una salida abrupta que obligue al operario a iniciar nuevamente una actividad que haya hecho con anterioridad. Por medio de las pruebas de amigabilidad y desempeño se podrán identificar los procesos inestables los cuales serán controlados.

3.1.3 Desarrollo del Diseño

La aplicación de Registro de Usuarios está dividida en:

- Autoridad de Registro
- Autoridad de Certificación
- Autoridad de Validación

Figura 3-1 Diagrama esquemático Aplicativo Registro de Usuarios



Fuente: Autor

Como se aprecia en la Figura 3-1, en un proceso de registro cotidiano, el usuario se acerca a la Autoridad de Registro. La RA debe primero autenticarse ante el sistema (función I) y comprobar la identidad del estudiante que solicita un certificado digital basado en la política de seguridad, que se describirá más adelante. Luego, la RA tendrá que registrar al estudiante (función II), generar el par de claves del mismo en la tarjeta inteligente (función III), emitir una solicitud para que la CA cree el certificado digital (función IV), y entregar la tarjeta inteligente grabada con el certificado y la clave privada al estudiante (función VII).

La Autoridad de Certificación (CA) es la encargada de comprobar la firma de las solicitudes que recibe de la RA, emitir los certificados y firmarlos, enviarlos luego a la RA (función V) y publicarlos en el repositorio de certificados (función VI). También es la encargada de revocar los certificados, tras comprobar la validez de las solicitudes que reciba (función VIII) y colocarlos en un sitio público (función IX).

El servidor OCSP y la base de datos de repositorios (BD_REPO), no son más que una segregación de la entidad Autoridad de Validación. El OCSP es quien consulta el certificado en la BD_REPO y luego responde indicando si este es válido o no (función X).

Antes de iniciar con la descripción del diseño de las entidades, se muestra cómo sería el proceso general para emitir los certificados, para que se pueda comprender mejor el rol que desempeña cada entidad. En la Figura 3-2 se muestra el diagrama de flujo general de las actividades llevadas a cabo por el sistema, segmentadas de acuerdo a los entes de la PKI.

3.1.3.1 Autoridad de Registro

Para iniciar el Aplicativo de Autoridad de Registro, el operario deberá primero iniciar sesión. En la Figura 3-3 se muestra cómo se autorizará el acceso a los operarios.

Como se puede observar en la Figura 3-3, además de las acciones típicas de cualquier inicio de sesión, se comprueba la conectividad con la Base de Datos de Usuarios (BD_USU), como último requisito para completar el inicio de sesión y lanzar la ventana principal, donde se mostrarán las siguientes opciones:

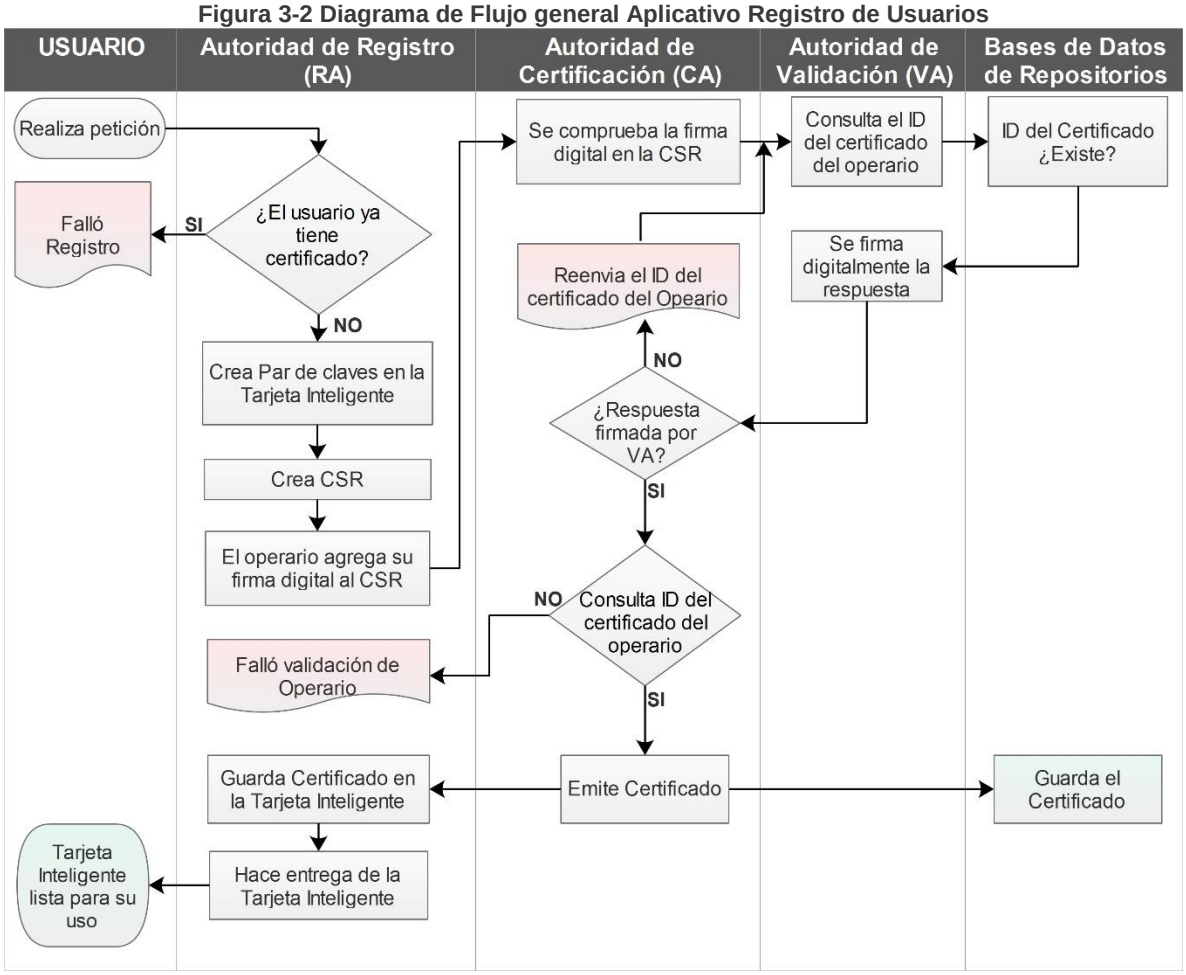
- **SolicitarCertificado:** Para generar las Solicitudes de Firma de los Certificados (CSRs), a partir de los datos del usuario, después de ser validados por el operario. Para registrar al usuario, se usará un formulario, cuyos campos obligatorios son:
    - Número de identificación
    - Nombre
    - Apellido

- Correo electrónico
- Sede (País, Departamento, Ciudad)
- Programa académico

Esta es la función principal de la Autoridad de Registro, y se detalla en la Figura 3-4. Primero, se generan el par de claves en la tarjeta inteligente y después se crea la CSR. Luego, el aplicativo debe firmarla usando la clave privada del operario, esto para garantizar que un posible atacante emita CSRs y la CA las apruebe, por ello, la CA debe corroborar la firma con la que ha llegado la CSR.

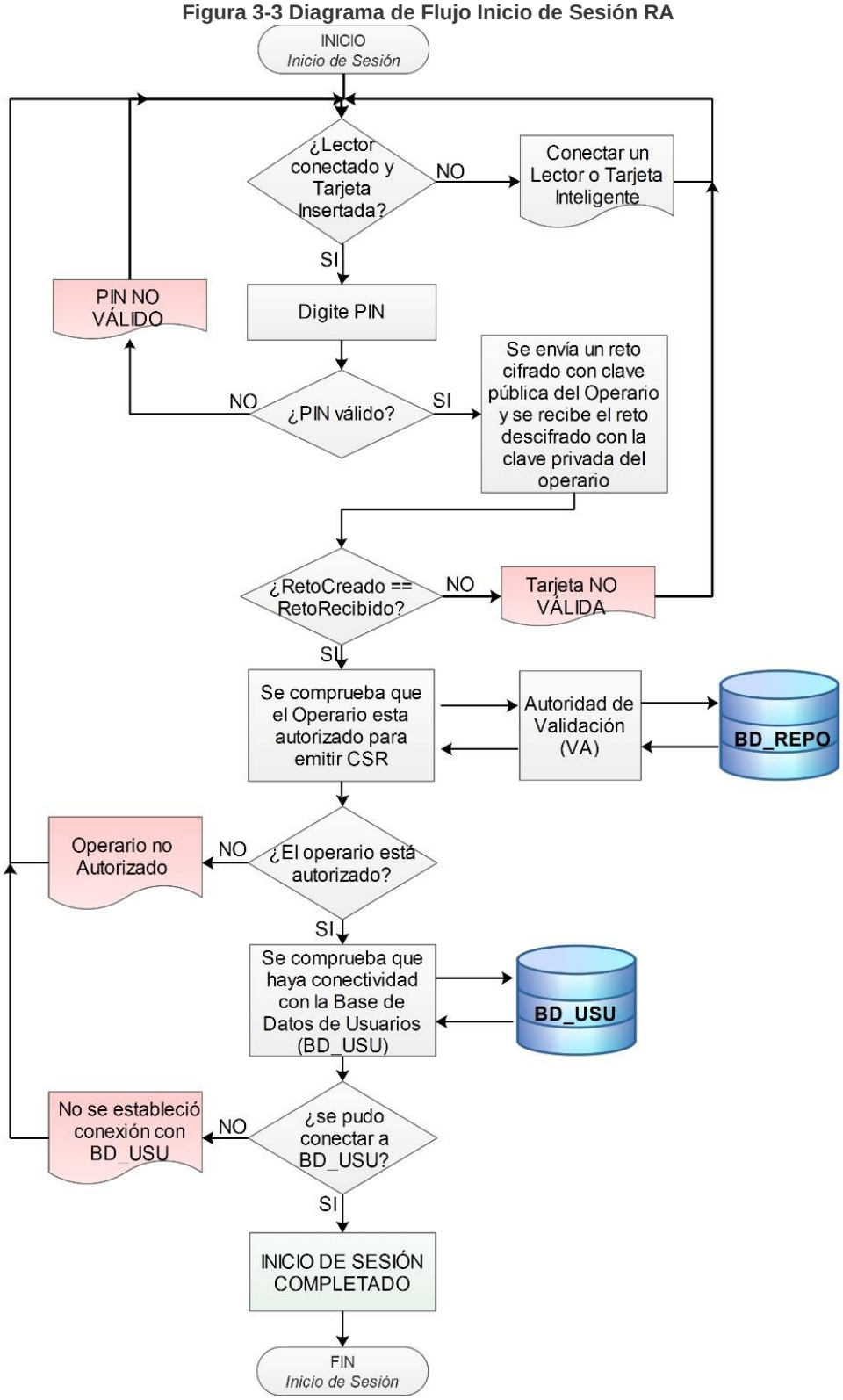
De los campos anteriores, solo el número de identificación no hará parte del certificado.

Figura 3-2 Diagrama de Flujo general Aplicativo Registro de Usuarios



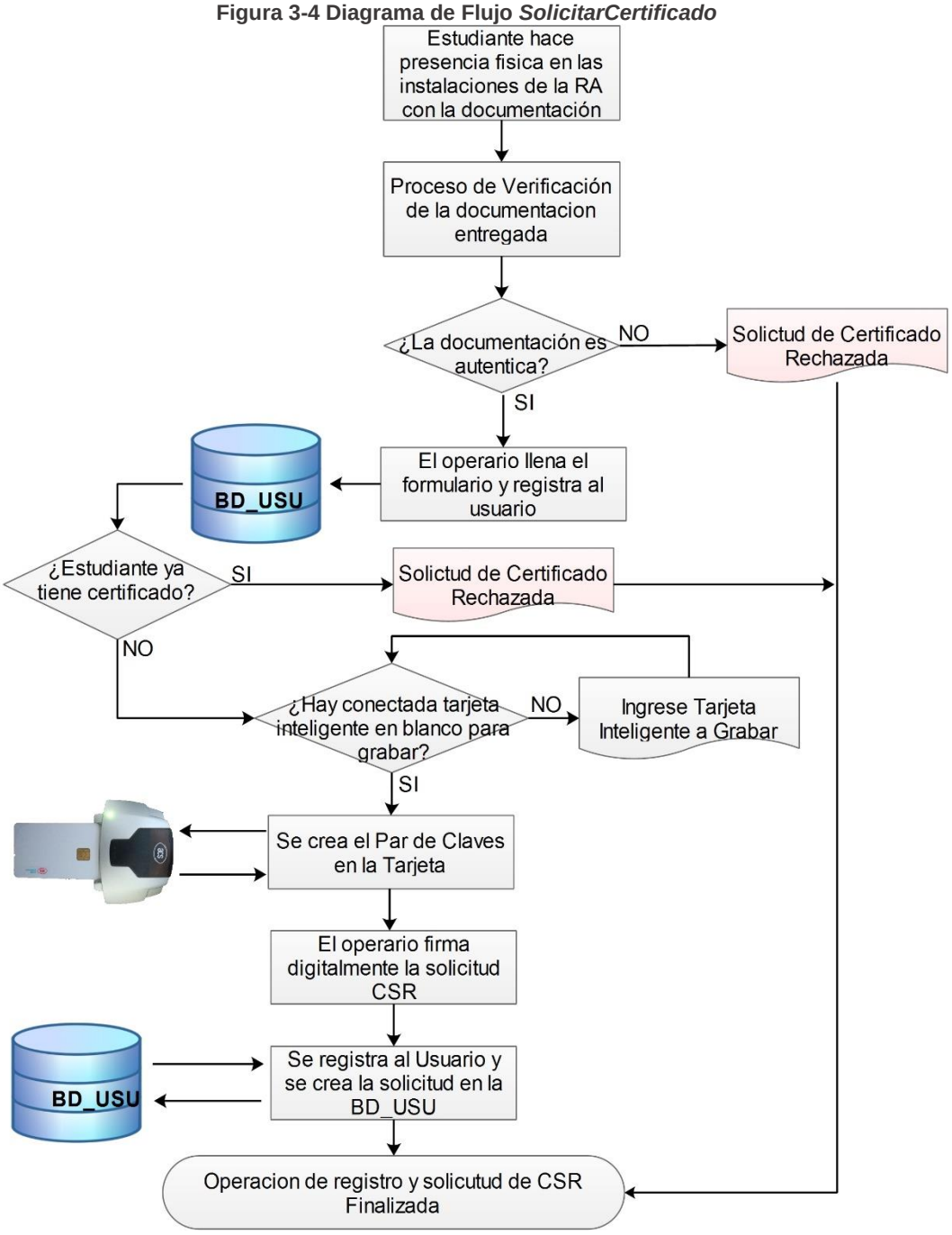
Fuente: Autor

Figura 3-3 Diagrama de Flujo Inicio de Sesión RA



Fuente: Autor

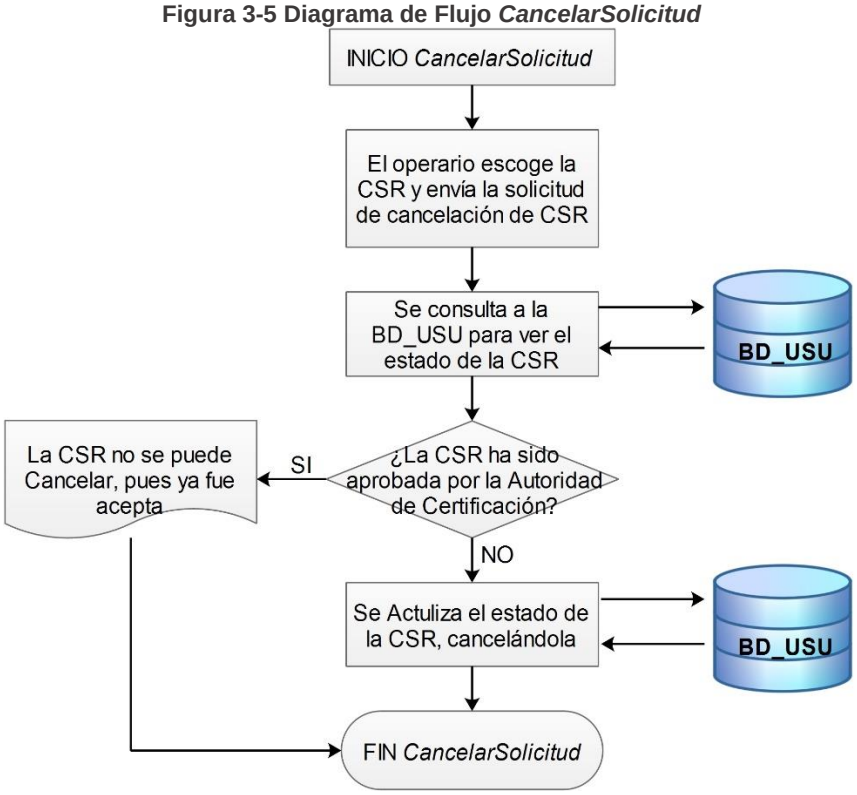
Figura 3-4 Diagrama de Flujo *SolicitarCertificado*



Fuente: Autor

- **CancelarSolicitud:** Para que el operario pueda cancelar una CSR. cuando considere haber cometido un error durante su generación (ver Figura 3-5.). Solo podrán cancelarse aquellas CSR para las que la CA aún no haya emitido un certificado.

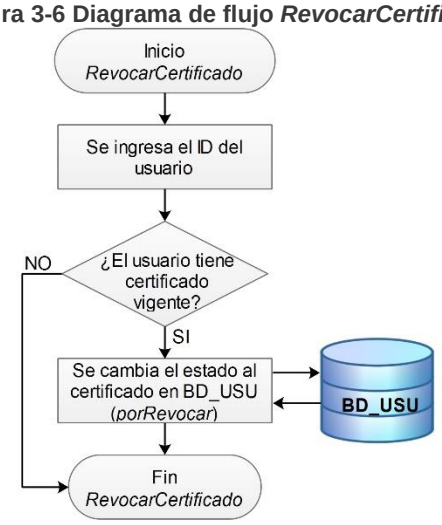
Figura 3-5 Diagrama de Flujo *CancelarSolicitud*



Fuente: Autor

- **SolicitudesEmitidas:** Para listar todas las CSRs emitidas por los operarios y su estado (no firmado, firmado, cancelado o rechazado).
- **SolicitudesAprobadas:** Para listar las CSRs que han sido aprobadas, e importar los certificados a las tarjetas inteligentes en caso que el proceso de importación automático en GenerarSolicitud haya fallado por no haber recibido la aprobación inmediata de la CA.
- **SolicitudesRechazadas:** Para listar todas las CSRs que fueron rechazadas por la CA.
- **SolicitudRevocación:** A través de esta opción el operario puede generar una solictud de revocación de certificados digitales a usuarios que manifiesten su intención de anular el certificado por sospechas que está en riego o su clave privada esté comprometida.

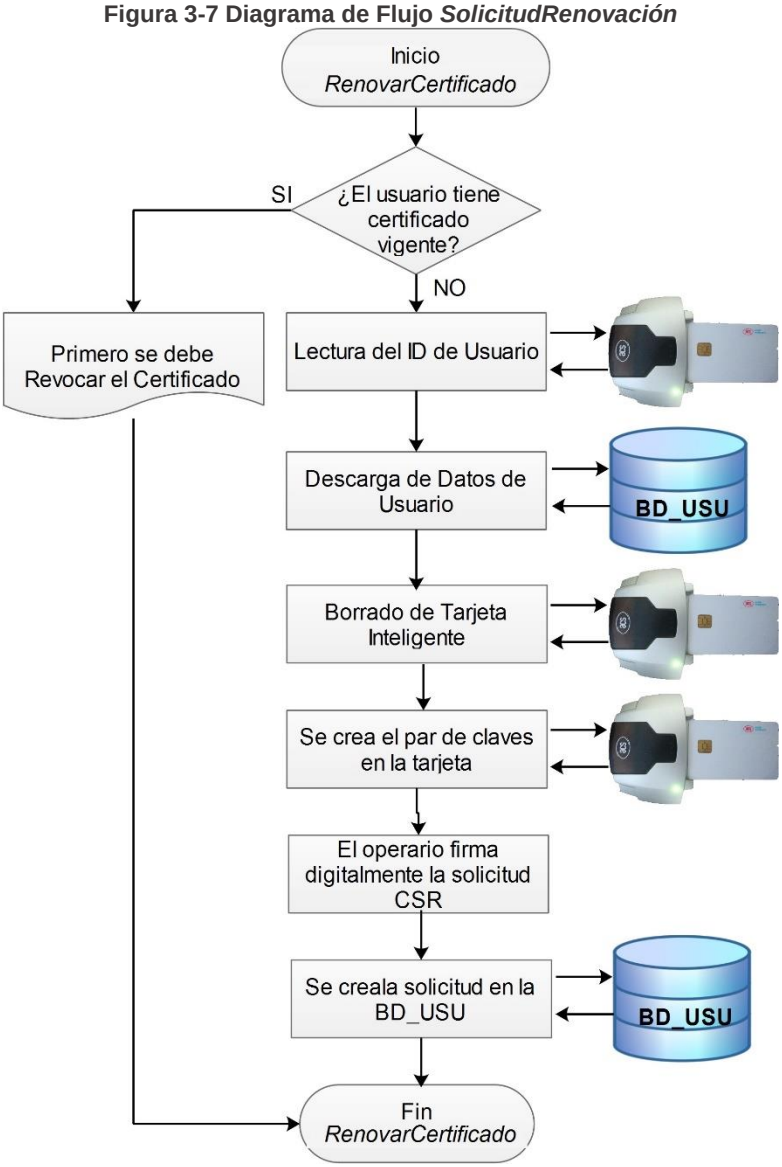
Figura 3-6 Diagrama de flujo *RevocarCertificado*



Fuente: Autor

- **RenovarCertificado:** Por medio de esta opción, el operario puede renovar el certificado digital aquellos usuarios que no tenga su certificado vigente ya sea porque a expirado o a sido revocado por solicitud del usuario. Es necesario que el certificado que haya de renovarse este deba estar revocado para poder efectuar se la operación

Figura 3-7 Diagrama de Flujo *SolicitudRenovación*



Fuente: Autor

3.1.3.2 Autoridad de Certificación

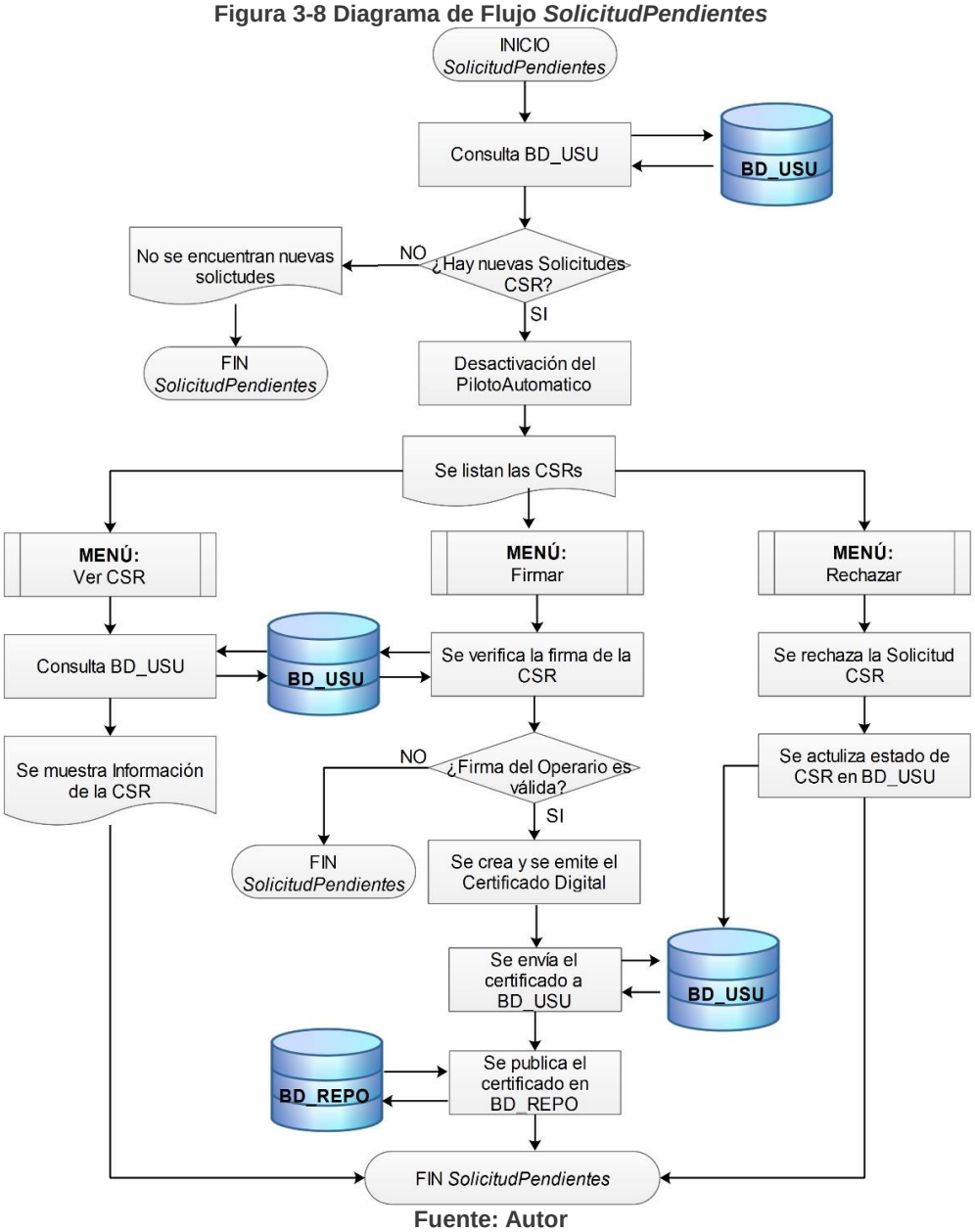
La Autoridad de Certificación es el corazón de una PKI, por tanto, también lo es del aplicativo de Registro de Usuarios. Esta entidad cobra tal importancia, ya que tiene la facultad de ofrecer confianza, pues al firmar digitalmente un certificado, asegura la veracidad de la información contenida en él.

Para cumplir las funciones de la CA, se plantean los siguientes subprocesos:

- **CertificadosFirmados:** Para listar los certificados emitidos por la Autoridad de Certificación.

- **CertificadosRevocados:** Para mostrar los certificados que han sido revocados. Corresponde a una consulta a la Base de Datos de Repositorios.
- **SolicitudPendientes:** Para visualizar las CSRs emitidas por la Autoridad de Registro (ver Figura 3-8). El operario podrá aceptar o rechazar cada solicitud, generar y firmar los certificados. Se deben revisar constantemente las solicitudes para garantizar la rapidez del Aplicativo de Registro de Usuarios. Como se puede ver en la Figura 3-8, el operario (CA) podrá elegir entre tres opciones: Ver CSR, Firmar y Rechazar.

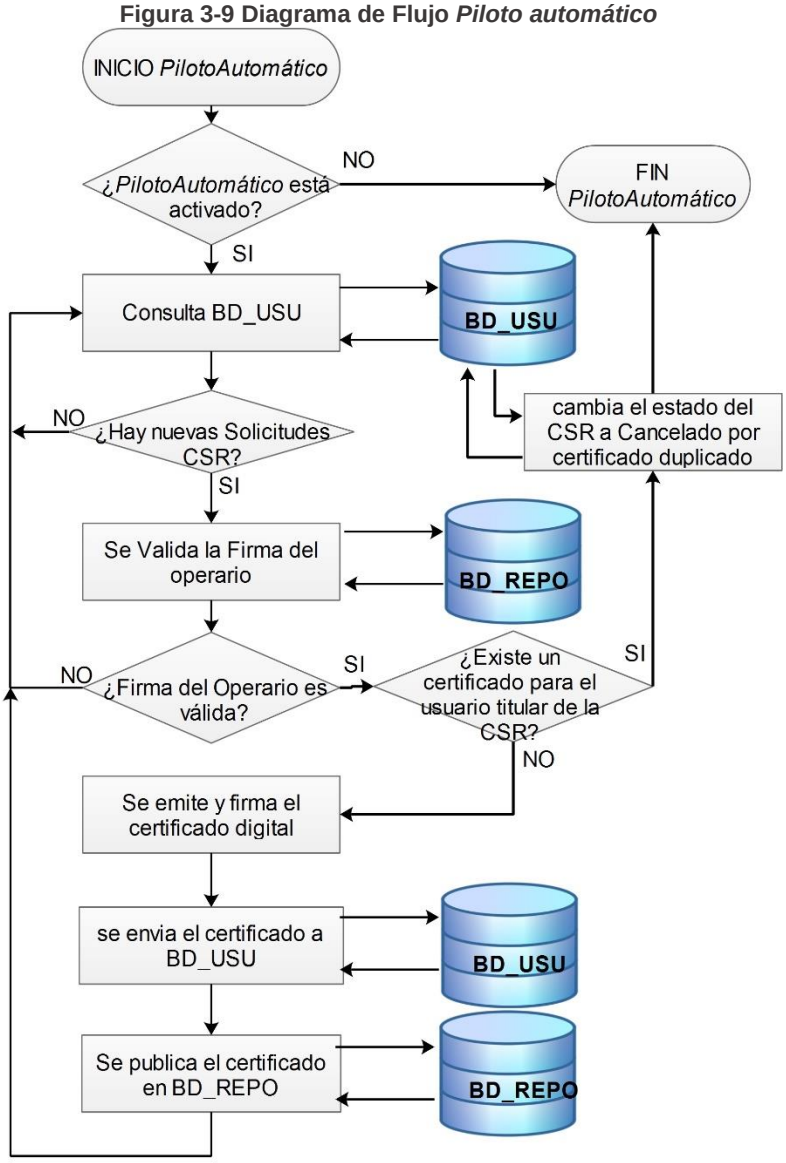
Figura 3-8 Diagrama de Flujo *SolicitudPendientes*



Fuente: Autor

- **PilotoAutomático:** Para analizar de manera automática cada solicitud de firma enviada por la Autoridad de Registro, y en caso de ser aprobada, se emitirá y firmará automáticamente el certificado que le corresponde; pero si la solicitud no cumple con los requisitos será rechazada, como se observa en la Figura 3-9. La activación de esta función automatizará la emisión de certificados en la Autoridad de Certificación, por lo tanto no será necesaria la intervención humana, salvo en caso de supervisión.
- **SolicitudRechazada:** Para listar las CSRs rechazadas.
- **RevocarCertificado**: Funcion automatizada que inicia con el programa y revisa constamente BD_USU en busca de solicitudes de revocación de certificados emtidas por la Autoridad de Registro.
  La renovación de certificados será ejecutada como una nueva solicitud de certificado, por lo que PilotoAutomatico las firmará como si fuesen solicitudes nuevas.

Figura 3-9 Diagrama de Flujo *Piloto automático*



Fuente: Autor

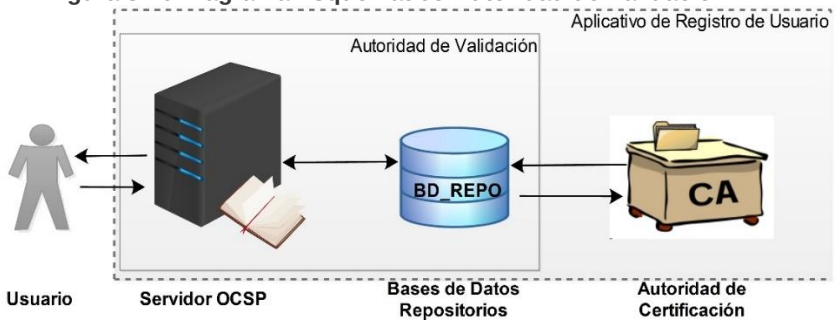
3.1.3.3 Autoridad de Validación

La Autoridad de Validación (VA) tiene como tarea suministrar información sobre la validez de los certificados digitales que hayan sido emitidos por la Autoridad de Certificación. La VA se compone de la Bases de Datos de Repositorios (BD_REPO) y el servidor OCSP (Online Certificate Status Protocol) que será quien firme digitalmente las respuestas a las consultas que hagan los usuarios sobre el estado de los certificados(ver Figura 3-10). Un usuario (que no necesariamente es una persona, pues puede ser cualquier entidad que requiera establecer la validez

de un certificado digital, por ejemplo, un servidor donde se va a autenticar un usuario) realiza una consulta al servidor OCSP y este busca en BD_REPO (solo la Autoridad de Certificación puede modificar, agregar, o actualizar la información de los certificados en BD_REPO), que responderá afirmativa o negativamente sobre la validez del certificado. El servidor OCSP recibirá esta respuesta y la firmará digitalmente.

Una vez entendido el papel de la Autoridad de Validación se presenta en la Figura 3-11, el diagrama de flujo para que este ente funcione y responda las consultas de los usuarios.

Figura 3-10 Diagrama Esquemático Autoridad de Validación



Fuente: Autor

3.1.4 Políticas de Emisión de Certificados

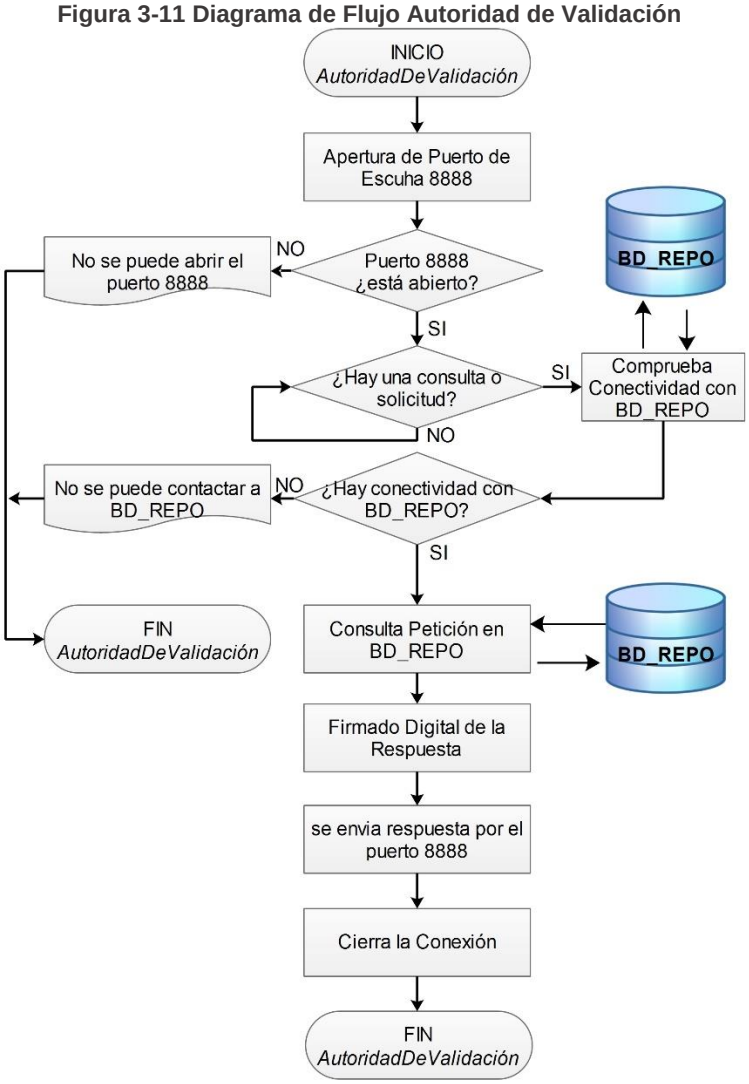
3.1.4.1 Solicitud de Certificado

Para emitir un certificado se exigirá la presentación personal del estudiante, o en caso de no poder, deberá presentarse un representante con poder notariado ante la Autoridad de Registro. El solicitante debe aportar los siguientes documentos:

- Original y copia de la cedula de ciudadanía o documento equivalente que acredite la identidad del solicitante (Tarjeta de Identidad, Visa de estudiante).
- Poder firmado y notariado por el estudiante, autorizando dicha diligencia (opcional).
- Recibo de matrícula financiera para verificar su condición de estudiante.
- El solicitante debe suministrar la siguiente información personal:
    - Número de Identificación
    - Nombres y Apellidos
    - Sede donde estudia (País, Estado, Ciudad)
    - Correo Electrónico

- Programa académico al que pertenece

Figura 3-11 Diagrama de Flujo Autoridad de Validación



Fuente: Autor

La Autoridad de Registro verifica si la información es real, suficiente y adecuada, procede a diligenciar el formulario de solicitud SolicitarCertificado en el software, además puede anexarse una fotografía del usuario –la cual no es obligatoria-.

3.1.4.2 Longitud de las Claves

Las claves generadas en las tarjetas inteligentes para los estudiantes deben tener una longitud mínima de 1024 bits, mientras que las claves emitidas a los operarios deben tener una longitud mínima de 2048 bits.

3.1.4.3 Período de Validez de los Certificados Digitales

El período de validez del Certificado Digital Estudiantil será de 5 años (10 semestres), ya que es el tiempo que duran los diferentes programas en la universidad.

Si un estudiante, en el momento de ingresar a la Universidad, no cuenta con su mayoría de edad, y por lo tanto no tiene número de cédula, se emitirá un certificado provisional que expirará la fecha en que dicho estudiante cumpla la mayoría de edad, tras lo cual, el estudiante deberá solicitar a la Autoridad de Registro la renovación del mismo.

Si un estudiante prolonga su estancia en la universidad más de 5 años, debe solicitar también la renovación de su certificado, tras su expiración.

3.1.4.4 Perfil del Certificado

El perfil de los certificados de los estudiantes se presenta en la Tabla 3-1. Los certificados son emitidos conforme con lo establecido en la especificación ITU X.509 versión 3. La asociación de un certificado con un estudiante se relaciona a partir del ID del certificado con el ID del estudiante en la BD_REPO, es decir, cada certificado poserá un ID distinto al ID del estudiante, es por ello, que en BD_REPO estos dos IDs se relacionan para así hallar el certificado que le corresponde a cada estudiante; la razón esto es porque el estándar X.509 no acepta ID de certificado cuya longitud sea impar, solo deben ser ID con longitud par, y los ID de estudantes pueden no cumplir esta condición, por lo que se debe independizar el ID del estudiante con el ID del certificado en una base de datos.

Tabla 3-1 Perfil de los Certificados Estudiantiles

Certificado x.509 v3 Atributos Extensiones	Nombre del campo	Contenido
Version	Versión	V3
Número de Serie	Serial Number	<Número de serie del certificado en hexadecimal<
Algoritmo de	signatureAlgoritm	SHA512RSA

Certificado x.509 v3 Atributos Extensiones	Nombre del campo	Contenido
firma		
Algoritmo Hash de firma	hashAlgoritm	SHA512
Nombre distintivo del emisor (Issuer)	commonName	CN=Unipamplona CA
	serialNumber	<valor hexadecimal> ej.: 1A
	organizationName	Universidad de Pamplona
	organizationalUnitName	CA-UPA
	stateOrProvinceName	Norte de Santander
	countryName	CO
	emailAddress	informes@unipamplona.edu.co
Validez (desde, hasta)	notBefore	<Fecha y hora de emisión> ej.: 150820150359Z
	notAfter	<Fecha y hora de expiración> ej.: 150820200359Z
Nombre distintivo del suscriptor (Subject DN)	commonName	Nombre1 nombre2 apellido1 apellido2
	serialName	<valor hexadecimal> ej.: 1A
	CountryName	<país de sede>
	stateOrProvinceName	<departamento de sede>
	localityName	<ciudad/municipio de sede >
	organizationName	Universidad de Pamplona
	organizationalUnitName	<programa académico>
Clave pública del suscriptor (Subject Public Key Info)	publicKeyAlgorithm	rsaEncryption
	publicKeyLength	1024 bits
	Clave pública del suscriptor	<valor a designar>
Restricciones básicas	basicConstraint	CA:FALSE
Uso de clave	keyUsage	digitalSignature = 0
		contentCommitment = 0
		keyEncipherment = 0
		dataEncipherment = 0
		keyAgreement = 0
		keyCertSign = 0
		cRLSign = 0
		encipherOnly = 0
		decipherOnly = 0
		nonRepudiation=1
Exponente público	publicExponent	65537 (0x10001)
Acceso a la información	Authority Information Access	OCSP - URI:http://ocsp.tesisjohrmanvides.edu.co:8888
Tipo de	Netscape Cert Type	SSL Client

Certificado x.509 v3 Atributos Extensiones	Nombre del campo	Contenido
Certificado		
Comentario Certificado	Netscape Comment	Certificado Generado por CA-UPA

Fuente: Autor

3.1.4.5 Entrega de la Tarjeta Inteligente

Al momento de la generación de la CSR por parte de la RA, se crea el par de claves del usuario. Este par de claves se crea directamente en la tarjeta inteligente con el formato que trae Criptoki del estándar PKCS#11, permitiendo la seguridad básica de la clave privada, es decir, su no extracción y copia, protegiendo su uso mediante un PIN. El PIN inicial de la tarjeta se solicitará al estudiante una vez este vaya a reclamar su tarjeta inteligente, de esta manera se mantiene la confiedencialidad del PIN.

El certificado digital, que puede ser conocido por todos los entes de la PKI, también estará almacenado en la tarjeta Inteligente, y al igual que la clave pública del usuario, se guarda en un lugar de acceso público de la tarjeta.

3.1.4.6 Revocación de Certificados

Cuando un estudiante se retire de la institución, debe regresar la tarjeta inteligente a la Autoridad de Registro. La Autoridad de Registro deberá enviar las solicitudes de revocación a la Autoridad de Certificación

La Autoridad de Certificación debe revisar constantemente qué estudiantes ya no hacen parte de la institución y revocar sus certificados.

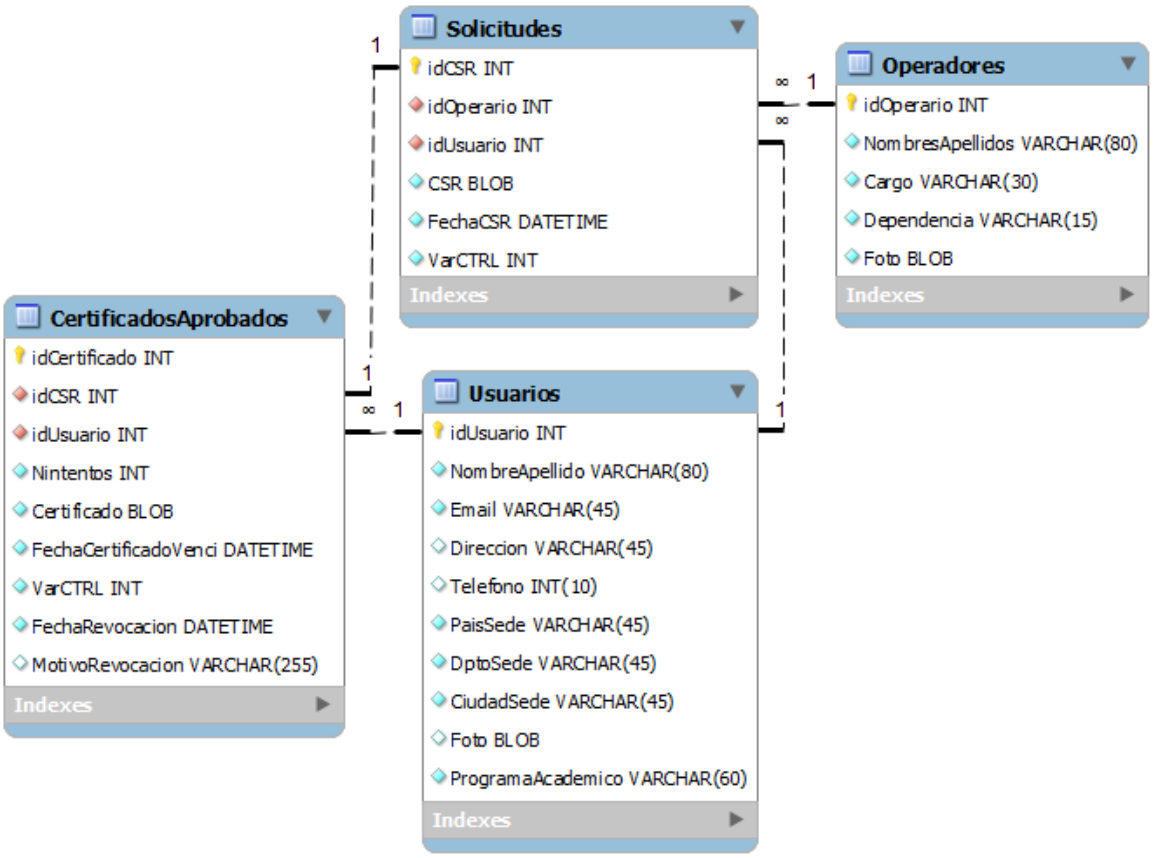
En caso que algún estudiante u operario, crea que su clave privada se vio comprometida, porque su tarjeta inteligente se le perdió, fue manipulada por otra persona ajena a él, o alguien descubrió su PIN, debe reportar esto a la Autoridad de Certificación a través de la Autoridad de Registro, que emitirá una solicitud de revocación y una solicitud de renovación, para que el certificado sea revocado y se emita otro certificado.

3.1.5 Diseño de la Base de Datos de Usuarios

En la Base de Datos de Usuarios se almacenarán los datos de los usuarios registrados en el aplicativo. En la Figura 3-12 se muestra el modelo entidad-relación de la base de datos diseñada. La llave primaria de cada tabla se indica con una llave amarilla. Las tablas que la conforman son:

- **CertificadosAprobados:** La CA tendrá permisos de escritura y actualización sobre esta tabla y la RA solo tendrá permiso de lectura. Esta tabla guarda los datos de los certificados que fueron aprobados por la CA y los asocia con los datos del usuario beneficiario y el operador que emitió la solicitud CSR. Los campos de esta tabla se especifican en la Tabla 3-2.
- **Solicitudes:** La RA y la CA tendrán todos los permisos sobre esta tabla. Almacena las CSRs, relacionando la solicitud directamente con el usuario y el operador que emitió la CSR. La CA, leerá esta tabla para obtener cada CSR generada, comprobar su firma y decidir si emite o no el certificado. Los campos de esta tabla se especifican en la Tabla 3-3.
- **Operadores:** Solo la CA y la RA tendrán acceso a esta tabla, con permisos de solo lectura (una entidad especial creará los operarios, que no está contemplada en este proyecto). Esta tabla contiene los operarios que pueden autenticarse ante el Aplicativo de la Autoridad de Registro para emitir las CSRs y registrar nuevos usuarios. La CA la consulta para hallar más información sobre los operarios que firman las solicitudes CSR. Está tabla es de tipo paramétrica, es decir, contiene información estática. Los campos de esta tabla se especifican en la Tabla 3-4.
- **Usuarios:** Contiene información acerca de los usuarios que se han registrado en la PKI. Sólo la RA y la CA tienen acceso total a esta tabla; la VA, tendrá acceso a sus datos a través de la CA. Los campos de esta tabla se especifican en la Tabla 3-5.

Figura 3-12 Diagrama Entidad/Relación Base de Datos de Usuarios



Fuente: Autor

Tabla 3-2 Datos tabla *CertificadosAprobados* de BD_USU

Nombre de Columna	Tipo	Nombre Descriptivo	Permite Valores nulos	Descripción
idCertificado	INT	ID. del certificado aprobado	NO	Identificador único del certificado –Número de Serie del certificado- (llave primaria)
idCSR	INT	ID. de la solicitud CSR	NO	Identificador de la solicitud CSR con la que fue emitido el certificado (llave foránea).
idUsuario	INT	ID. del usuario	NO	Número de identificación del titular del certificado (llave foránea)
Nintentos	INT	Numero de Intentos	NO	Número de intentos de sesión inválidos restantes, el valor inicial

				son 3 intentos.
Certificado	BLOB	Certificado	NO	Archivo del certificado en formato .PEM
FechaCertificadoVenci	DATETIME	Fecha de expiración del certificado	NO	Fecha en la que termina la validez del certificado
VarCTRL	INT	Variable de control	NO	Variable de Control para establecer el estado actual del certificado: Vigente (1), Revocado (2), porRevocar (3).
FechaRevocacion	DATETIME	Fecha de Revocación	SI	Fecha en la que fue revocado el certificado
MotivoRevocacion	VARCHAR(255)	Motivo de Revocación	SI	Motivo por el cual fue revocado el certificado

Fuente: Autor

Tabla 3-3 Datos tabla *Solicitudes* de BD_USU

Nombre de Columna	Tipo	Nombre Descriptivo	Permite Valores nulos	Descripción
idCSR	INT	ID. CSR	NO	Identificador de la solicitud CSR (llave primaria)
idOperario	INT	ID. Operario	NO	Número de identificación del operario (llave foránea)
idUsuario	INT	ID. Usuario	NO	Número de identificación del titular del CSR
CSR	BLOB	CSR	NO	Fichero CSR en formato PEM
FechaCSR	DATETIME	Fecha CSR	NO	Fecha en la que fue emitida la solicitud CSR
VarCTRL	INT	Variable de Control	NO	Variable de Control para establecer el estado actual de la CSR: No Firmada por CA (1), Firmada (2), Solicitud Duplicada (3), Falló Firma CSR (4), CSR Rechazada por CA (5), CSR Cancelada por RA (6).

Fuente: Autor

Tabla 3-4 Datos tabla *Operadores* de BD_USU

Nombre de Columna	Tipo	Nombre Descriptivo	Permite Valores nulos	Descripción
idOperario	INT	ID. del Operario	NO	Número de Identificación del Operario (llave primaria)
NombresApellidos	VARCHAR(80)	Nombre	NO	Nombre del Operario
Cargo	VARCHAR(30)	Cargo	NO	Cargo que ocupa el operario
Dependencia	VARCHAR(15)	Dependencia	NO	Dependencia del Operario
Foto	BLOB	Foto	NO	Foto del operario en formato JPG

Fuente: Autor

Tabla 3-5 Datos tabla *Usuarios* de BD_USU

Nombre de Columna	Tipo	Nombre Descriptivo	Permite Valores nulos	Descripción
idUsuario	INT	ID. Usuario	NO	Número de Identificación del usuario (llave primaria)
NombreApellido	VARCHAR(80)	Nombre y apellido	NO	Nombres y Apellidos del usuario
Email	VARCHAR(45)	Correo electrónico	SI	Dirección de correo electrónico del usuario
Telefono	INT (10)	Teléfono	SI	Número de celular del usuario
Direccion	VARCHAR(45)	Dirección	SI	Dirección de residencia del usuario
PaisSede	VARCHAR(45)	País de sede	NO	País de la sede donde estudia el usuario
DptoSede	VARCHAR(45)	Departamento de sede	NO	Departamento de la sede donde estudia el usuario
CiudadSede	VARCHAR(45)	Ciudad o municipio de sede	NO	Ciudad o municipio de la sede donde estudia el usuario
ProgramaAcademico	VARCHAR(60)	Programa académico	NO	Programa académico al que pertenece el usuario
Foto	BLOB	Fotografía	NO	Fotografía del usuario en formato JPG

Fuente: Autor

3.1.6 Diseño de la Base de Datos Repositorios

En este caso se ha planteado un repositorio sencillo, el cual almacena los certificados emitidos y revocados por la CA. El servidor OCSP será un puente entre los entes públicos y esta base de datos. La CA será la única entidad que tendrá los permisos necesarios para modificar y borrar la información de esta base de datos; los demás usuarios sólo podrán leer la información. En la Figura 3-13 se muestran los campos de la única tabla que compone esta base de datos y en Tabla 3-6 se describen dichos campos.

Figura 3-13 Diseño Bases de Datos Repositorios

Fuente: Autor

Tabla 3-6 Datos tabla *Respositorios* de BD_REPO

Nombre de Columna	Tipo	Nombre Descriptivo	Permite Valores nulos	Descripción
idCertificado	INT	ID. certificado	NO	Número de serie del certificado (llave primaria)
Certificado	BLOB	Certificado	NO	Archivo del certificado en formato .CRT
Estado	INT	Estado del certificado	NO	Estado en el que se encuentra el certificado: vigente (1), Revocado (2)
FechaVencimiento	DATETIME	Fecha de vencimiento	SI	Fecha de vencimiento del certificado
FechaRevocación	DATETIME	Fecha de Revocación	SI	Fecha de revocación del certificado
NombreTitular	VARCHAR(80)	Nombre de titular	NO	Nombre del titular del certificado

Fuente: Autor

3.1.7 Elección del Software de Desarrollo

Primero se presentan las características del hardware con el que se cuenta (tarjetas y lector). Luego se analizan los proveedores de librerías criptográficas, donde se tiene en cuenta la licencia de operación, el lenguaje de programación en que fue escrito, la calidad de las bibliotecas criptográficas y la compatibilidad con el hardware utilizado. Una vez presentada esta información, se toma una decisión.

3.1.7.1 Características del Hardware

Las tarjetas inteligentes son importantes para este trabajo, pues permiten el almacenamiento y procesamiento de datos. Las tarjetas utilizadas son las ACOS5-32-G de la compañía *Advanced Card Systems* (ACS). Sus características se muestran en la Tabla 3-7.

Tabla 3-7 Características de la Tarjeta Inteligente ACOS-32-G

Característica	ACOS5-32-G	Característica	ACOS5-32-G

Tasa de Baudios	115Kbps	Memoria de Usuario	32 KB
Procesador Criptográfico	SI	Autenticación mutua	Si, por medio de clave de sesión
Protocolo T=0	SI	Protocolo T=1	NO
Estándares	ISO 7816-1/2/3/4/8/9	Arquitecturas PKI soportadas	CSP y PKCS#11
Algoritmos criptográficos	DES / 3DES / AES-128 bits / RSA (hasta 2048 bits)	Rendimiento escritura de certificado	RSA-1024: 605ms por certificado. RSA-2048: 70124ms por certificado

Fuente: Autor

Por otra parte, las características del lector ACR38 de la empresa ACS, se muestran en la Tabla 3-8.

Tabla 3-8 Características del Lector ACR38

Categoría	ACR38	Categoría	ACR38
Puerto de conexión	USB 2.0	Lectura max almacenamiento	256KB
Soporta	ISO 7816, PS/SC	Protección contra circuito eléctrico	SI
Protocolo T=0	SI	Protocolo T=1	SI

Fuente: Autor

Este hardware apenas cumple los requisitos mínimos para la ejecución del proyecto, donde se pretendía emitir claves de 2048 bits para los estudiantes pero debido a los tiempos para operaciones con esta longitud de claves se debió disminuir su longitud a 1024 bits.

3.1.7.2 Elección del Middleware y el Lenguaje de Programación

Los lectores no se comunican directamente con la aplicación, lo hacen a través de un *middleware*, que asiste a la aplicación para interactuar o comunicarse con otras aplicaciones, redes, sistemas operativos o hardware. El uso de un *middleware* en esta tesis simplifica el trabajo del programador, en la compleja tarea de generar conexiones y sincronizaciones con los dispositivos criptográficos. De esta manera, se provee una solución que mejora la calidad del servicio, así como la seguridad de la información intercambiada entre las tarjetas inteligentes y la Aplicación de Registro de Usuarios, por medio del uso de las librerías criptográficas que el *middleware* provee. Cada *middleware* varía dependiendo de sus funciones. Los siguientes *middleware* proporcionan APIs con bibliotecas criptográficas para la interacción de la aplicación con los lectores y tarjetas inteligentes:

- ***OpenSC*:** Ofrece un amplio conjunto de bibliotecas criptográficas para la autenticación, cifrado, descifrado, firmas digitales, etc. Está escrito en lenguaje C, y puede ser ejecutado en cualquier sistema operativo, pero tiene mayor compatibilidad con los SO Windows ya que las bibliotecas más desarrolladas y

completas están bajo el estándar PC/SC (*Personal Computer/Smart Card*)[41]. Su licencia de código abierto LGPL ha permitido que una gran comunidad aporte en el desarrollo de esta API. A pesar de esto no se ha logrado compatibilidad con la tarjetas inteligentes ACOS5. Sus bibliotecas están escritas en C por lo que elegir esta API, implica tener que usar cualquier lenguaje de la familia C (C, C++, C# o D).

- ***NCryptoki*:** A través de .NET *Framework* ofrece muchas bibliotecas criptográficas para autenticación, cifrado, descifrado, firmas digitales, etc. Su desarrollo se basa en el estándar PKCS#11, por lo que proporciona una API para los lenguajes de programación C#, VB.NET, Visual Basic 6, Delphi y Java. Es compatible con la tarjeta inteligente ACOS5, pero su licencia de funcionamiento es *Shareware*[42], lo que no permite su escogencia.

- ***BouncyCastle*:** Cuenta con colecciones de decenas de APIs destinadas a la criptografía, con cientos de bibliotecas de seguridad e interacción con tarjetas inteligentes. Tiene dos líneas de desarrollo, en Java y en C, permitiendo abarcar muchos sistemas operativos y compatibilidades con la gran mayoría de tarjetas inteligentes en el mercado. Su licencia permite el uso, modificación, venta, distribución sin autorización. Además su compatibilidad con la tarjeta ACOS5, la hace un candidato con muchas posibilidades de ser escogido.

- ***Microsoft Cryptography API (CryptoAPI) o (CSP):*** Es la API criptográfica que está incluida en los Sistemas Operativos Windows por medio del estándar CSP. Su licencia es privada e implica que solo los SO Windows la puedan usar. Posee gran popularidad gracias a que sus SO son muy populares en el mundo. Para acceder a sus bibliotecas se debe usar leguaje C, C++ ó C#. Es compatible con la tarjeta ACOS5.

- ***IAIK-JCE:*** Fue el primer proveedor de seguridad criptográfica escrito totalmente en Java. Esta API implementa cientos de bibliotecas tales como funciones de hash, códigos de autenticación de mensajes, criptografía simétrica y asimétrica, cifrado de bloques y de flujo. Su licencia es propietaria, pero IAIK-JCE también se puede obtener libremente con fines académicos, evaluación y desarrollo de código abierto. A diferencia de *BouncyCastle* contiene bibliotecas que controlan el espacio de memoria usado, esto le da un

[41] *PC/SC:* Conjunto de especificaciones para la integración de tarjetas inteligentes en ordenadores personales. En particular se define una API de programación que permite a los desarrolladores trabajar de forma uniforme con lectores de tarjetas de distintos fabricantes que cumplan con la norma PS/SC.

[42] *Shareware:* Modalidad de distribución de software, en la que el usuario puede evaluar de forma gratuita el producto, pero con limitaciones en el tiempo de uso o en algunas de las formas de uso, o con restricciones en las capacidades finales.

plus a esta API en términos de seguridad, además ofrece una gran documentación del producto de la experiencia con usuarios sus comerciales.

BouncyCastle y *IAIK-JCE*, son proveedores de seguridad fascinantes que implementan grandes bibliotecas criptográficas. Por un lado se tiene una API que es libre con código abierto con todo el potencial que una gran comunidad puede aportar, por el otro lado, *IAIK-JCE* con su licencia propietaria, no permite la modificación de su código fuente y solo los desarrolladores de *IAIK-JCE* pueden modificar su producto. A pesar que *BouncyCastle* es muy completo, *IAIK-JCE* se acomoda mejor a los requerimientos de esta tesis. *BouncyCastle* tiene más flexibilidad en algunas áreas como el estándar PKCS#12, mientras que *IAIK-JCE* implementa *XMLsecurity*, y *PKCS11Wrapper*; estas bibliotecas fueron el factor determinante para escoger a *IAIK-JCE* como API proveedora de funciones de seguridad criptográficas. *XMLsecurity* permite la integración con servidores Apache para el cifrado y descifrado de información, haciendo más seguras las conexiones entre los tokens de la API y un servidor Apache; esto puede tener gran ventaja, pues al momento de autenticarnos en una página web, el servidor Apache se intercomunicará directamente con el token (es decir, con la tarjeta inteligente) de forma cifrada evitando que flujos de información crucen por lugares no deseados. *PKCS11Wrapper* es un módulo nativo en Java para el estándar PKCS#11, mientras que en *BouncyCastle*, el módulo que implementa PKCS#11 está escrito en C, y para lograr acceder a él es necesario usar JNI de Java ampliando los espacios usados en la memoria y pudiendo degradar la seguridad de la aplicación.

Al escoger *IAIK-JCE* como proveedor de bibliotecas de seguridad, JAVA debe ser el lenguaje donde se debe desarrollar la Aplicación de Registro de Usuarios, pues la API está destinada exclusivamente a este lenguaje. Además, es un lenguaje muy extendido que podrá extender este proyecto a dispositivos móviles celulares, o cualquier otro equipo que use la máquina virtual de Java.

3.1.7.3 Elección del Gestor de Bases de Datos

El gestor de bases de datos utilizado es MariaDB, debido a su disponibilidad, facilidad de uso (al utilizar la misma sintaxis de MySQL) y características como precisión de los tipos de datos TIME, DATETIME, y TIMESTAMP en microsegundos, columnas dinámicas, que proporcionas columnas virtuales, entre otros.

Para la integración de Java con las bases de datos se usó el JBDC de MySQL, por sus rapidez en operaciones básicas de lectura e inserción (ver Tabla 3-9**¡Error! No se encuentra el origen de la referencia.**), conectividad y seguridad para acceder a las bases de datos en Internet, entre otra gran variedad de funciones que lo hace una de las mejores opciones actualmente.

En la Tabla 3-9**¡Error! No se encuentra el origen de la referencia.** se muestra la omparación de algunas de las funciones y los tiempos de acceso correspondientes a cada API gestora entre Java y MariaBD.

Tabla 3-9 Tiempo de operaciones básicas en JDBC

Manejador de Base de Datos	Tiempo de Lectura (mS)	Tiempo de Inserción (mS)
MySQL_JDBC	367	451
Informix_OBDC	1242	1428
Oracle_OBDC	2080	2261

3.2 Diseño de la Aplicación Web

La aplicación Web se va a encargar de autenticar a los usuarios para que accedan a la base de datos de información académica.

3.2.1 Requisitos Funcionales

I. La aplicación Web debe autenticar a los usuarios:
   a. ¿Quién realiza la operación?: Los usuarios
   b. ¿Qué datos ingresa?: Insertan su tarjeta inteligente e introducen su PIN
   c. ¿Qué hace sistema?: Le envía un desafío cifrado al usuario y verifica que la respuesta a este sea correcta. Si es correcta concede el acceso, de lo contrario se lo deniega.
   d. Extensiones:
      i. La aplicación Web debe obtener el certificado de la tarjeta inteligente, comprobar su firma y verificar que es válido. Luego, debe cifrar el desafío con la clave pública del usuario.
      ii. El navegador del usuario debe acceder a la clave privada guardada en la tarjeta inteligente, descifrar el desafío y enviárselo a la aplicación Web.

II. La aplicación Web debe permitir el acceso de los usuarios a la base de datos de información académica
   a. ¿Quién realiza la operación? La aplicación Web.
   b. ¿Qué datos ingresa? El token que comprueba que el usuario se autenticó correctamente

c. ¿Qué hace el sistema? Permite que el usuario vea la información académica que le corresponde.

3.2.2 Requisitos No Funcionales

- **Desempeño:** El tiempo de respuesta de la aplicación Web deberá ser pequeño, de manera que varios usuarios simultáneamente puedan consultar la base de datos de información académica, sin que haya grandes retardos en recibir la información solicitada.
- **Disponibilidad**: La aplicación Web debe estar disponible todos los días calendario, excepto cuando se dé aviso anticipado a los usuarios, debido a tareas de mantenimiento.
- **Usabilidad**: El aplicativo Web debe ser fácil de usar. Se tratará que sea lo más intuitivo posible. La selección del certificado debe diseñarse de tal manera que el usuario visualice solo la información realmente importante. Igualmente, cuando entre a la Base de Datos de Información Académica, se debe establecer una ruta realmente sencilla, con ayuda de un buen diseño de la interfaz gráfica, para acceder a las notas. Por otro lado, se deben presentar mensajes de error que permitan a los usuarios identificar el tipo de error.
    - **Seguridad:** El aplicativo Web debe evitar el acceso de usuarios no autorizados a la base de datos de información académica, la modificación de la información en esta base de datos y la suplantación de usuarios.

      La información intercambiada entre la aplicación Web y el usuario deberá viajar cifrada para que no sea visible a ningún atacante.

3.2.3 Desarrollo del Diseño

La finalidad de la Aplicación Web a implementar es autenticar y autorizar a los usuarios que van tener acceso a las Bases de Datos de Información Académica (BD_IACA). La autenticación se logra con ayuda de la Aplicación de Registro de Usuarios, pues se hará uso de sus bases de datos para llevarla a cabo. La Aplicación Web permitirá leer el certificado digital del usuario, almacenado en una tarjeta inteligente, y le solicitará el PIN para poder acceder a la clave privada (este PIN se pide para evitar un error de redundancia cíclica[43]). Se realiza autenticación

[43] Este error se refiere a un dispositivo de almacenamiento como una tarjeta inteligente cuando son leídos en cierto sector no coincide con el valor esperado, en otras palabras el archive está dañado, pero en realidad no está dañado, solo que el usuario público de la tarjeta no está autorizado para leer dicho sector y por

fuerte, pues el usuario debe tener la tarjeta y conocer el PIN para poderse autenticar ante el sistema, es decir, el usuario tiene que presentar *algo que tiene y algo que conoce*. Los intercambios de mensajes que se harán entre el Aplicativo Web y el usuario, cuando el inicio de sesión sea exitoso, se muestran en la Figura 3-14.

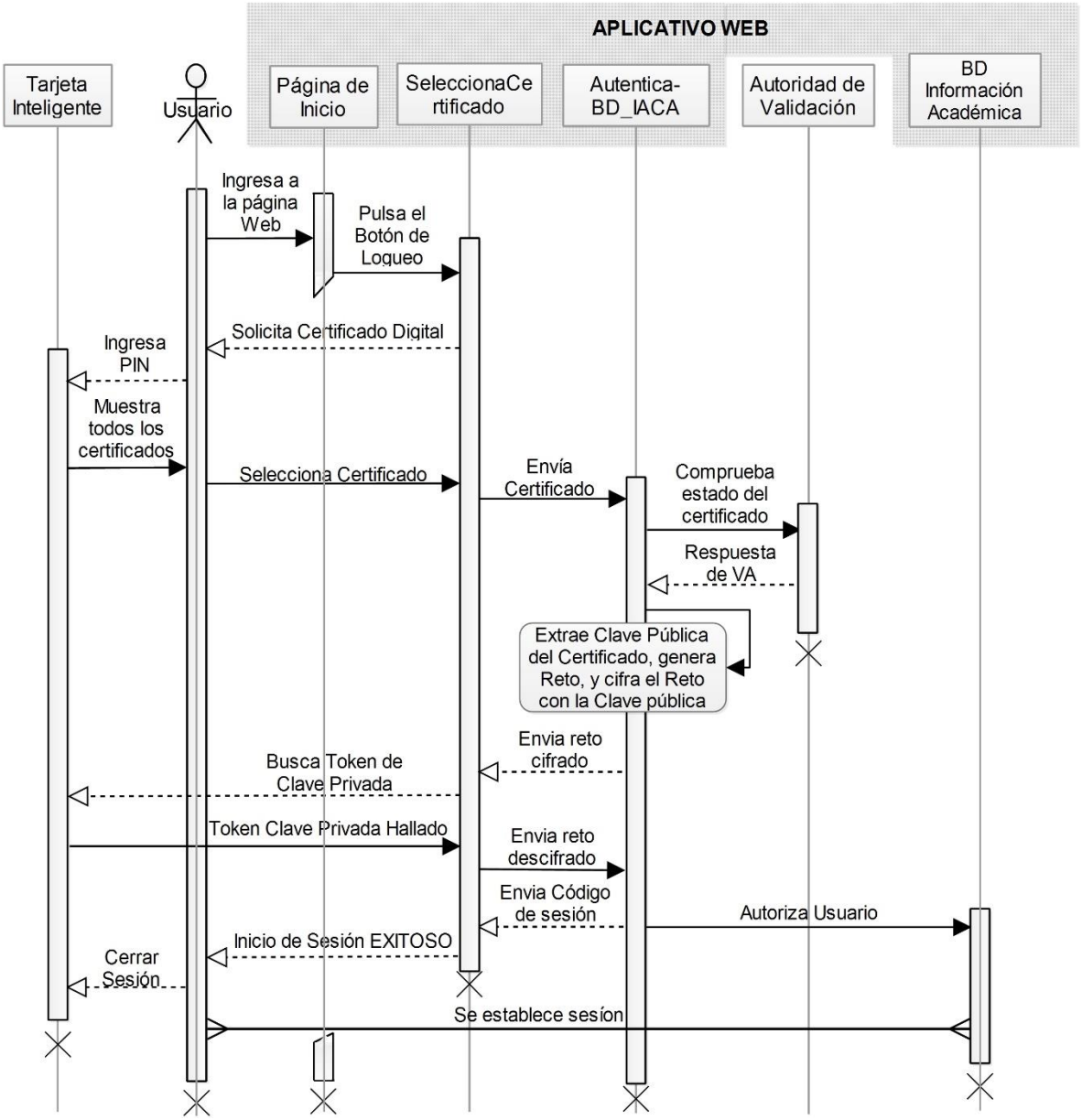
Como se aprecia en la Figura 3-14, el Aplicativo Web estará compuesto por:

- **Página de Inicio:** Página Web que permitirá acceder a SeleccionaCertificado.
- **SeleccionaCertificado:** Será una aplicación que se ejecute en el contexto del navegador (Applet Java), para acceder e interactuar con la tarjeta, el usuario, y el autenticador de la BD_IACA (Autentica-BD_IACA).
- **Autentica-BD_IACA:** Es el autenticador de la base de datos de información académica (BD_IACA). Se tratará de una aplicación Web que autenticará a los usuarios por medio de la comprobación de posesión de la clave privada a través de un reto o desafío (función I). Estará diseñada de tal forma que el usuario no pueda percatarse de su existencia. Se comunicará con el usuario a través de *SeleccionaCertificado*. Las autorizaciones se realizarán por medio de un código de sesión que se almacenará en una *cookie*[44] del navegador (función II).
- **Base de Datos de Información Académica (BD_IACA):** Sitio al que el usuario en cuestión quiere acceder para consultar sus notas. Se asume que los profesores ya han publicado las notas de cada estudiante en esta base de datos.

tanto está escrito de manera diferente que solo el usuario privado puede entender. Recuerde para el usuario privado se accede con el PIN.

[44] C*ookie (o Galleta Informática):* Es una pequeña cantidad de información enviada por un sitio web y almacenada en el navegador del usuario, de manera que el sitio web puede consultar la actividad previa del usuario. Una de las principales funciones es llevar el control de usuarios, caducando estas cuando se fija una fecha de vencimiento, se cierra el navegador o se cierra la sesión.

Figura 3-14 Diagrama de Secuencias de Mensajes Aplicativo Web



Fuente: Autor

3.2.3.1 SeleccionaCertificado

En la Figura 3-15 se aprecia el diseño de esta función de vital importancia para el Aplicativo Web, pues interactúa con los usuarios para que estos seleccionen un certificado digital de la tarjeta inteligente con el cual desean autenticarse. Aunque el proceso de autenticación y autorización lo realiza *Autentica-BD_IACA,* esta necesita datos que solo *SeleccionaCertificado* puede suministrar, como el

certificado digital y las operaciones criptográficas de la tarjeta. Este proceso consta de varios bucles que detectan: si el lector se ha desconectado del PC y si la tarjeta ha sido retirada del lector (en estos casos el proceso de inicio de sesión a la base de datos de información académica se cancela abruptamente).

3.2.3.2 Autentica-BD_IACA

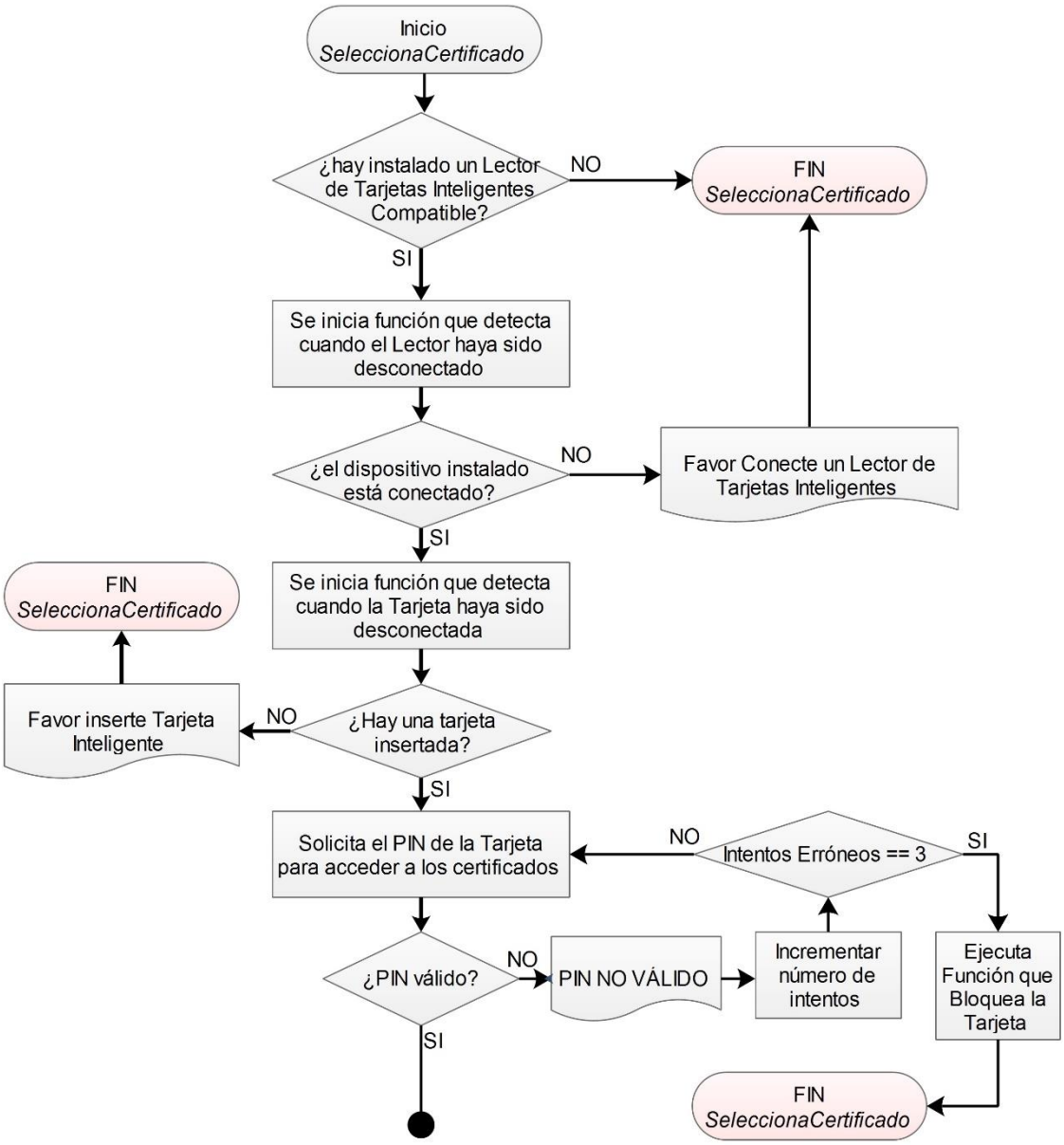
La función que cumplirá esta entidad es autenticar a los usuarios y autorizarlos para que puedan acceder a la Base de Datos de Información Académica (BD_IACA).

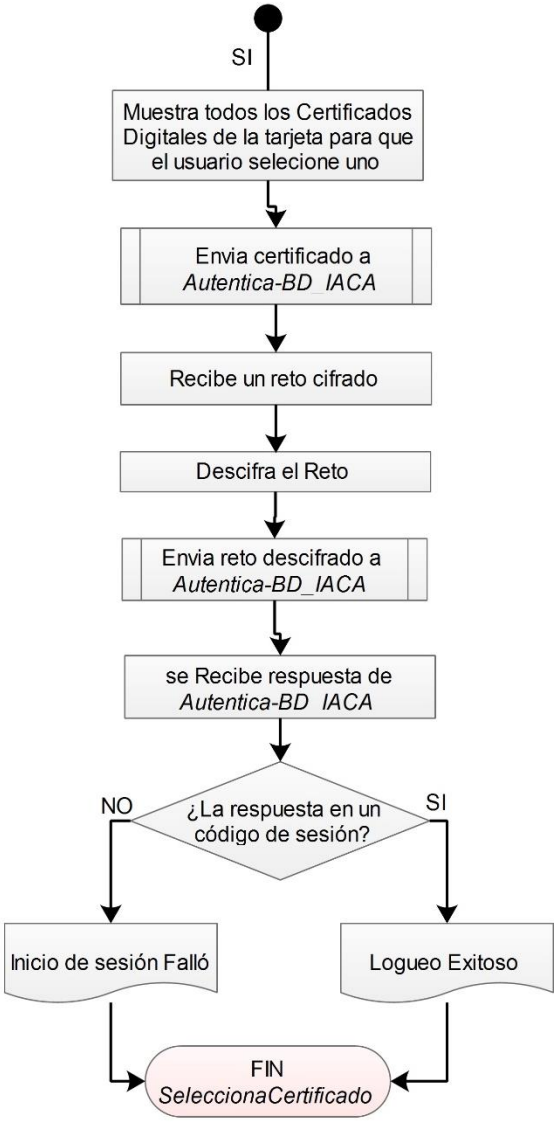
Por si misma, esta función no podrá materializar la autenticación ni mucho menos la autorización, pues se complementa con *SeleccionaCertificado,* ya que todo su flujo de información vendrá de esa entidad, la cual a través de una variable de control definirá qué proceso debe hacer *Autentica-BD_IACA*: en caso de ser 1 comprobará el certificado y generará un reto cifrado, si es 2 comparará el reto descrifrado que ha recibido con el reto original y generará el código de sesión que es un valor alfanumérico de 130 caracteres. Se debe tener en cuenta la seguridad pues violentar este aplicativo supondría un alto riesgo en el acceso a BD_IACA, sin embargo, el diseño que se muestra en la Figura 3-16 no refleja las operaciones que se realizarán para mantener a esta función segura, solo muestra el flujo de datos básico.

3.2.3.3 Base de Datos de Información Académica

Esta base de datos almacenará la información académica de los estudiantes, es decir, sus notas. El acceso a esta base de datos debe ser restringido, por ello *Autentica-BD_IACA* será la única que puede autorizar a los usuarios acceder a esta base de datos. Este acceso también debe implicar el uso de roles, pues un estudiante no puede tener acceso a las notas de otros estudiantes. En la Figura 3-17 se presenta el diagrama entidad-relación de esta base de datos que se compone de 5 tablas relacionadas entre sí por medio de llaves primarias y foráneas. **NOTA:** El diseño e implementación que se realiza de esta base de datos tiene como fin emular una base de datos de información académica real, para demostrar la potencialidad de este proyecto, no porque sea un objetivo a cumplir.

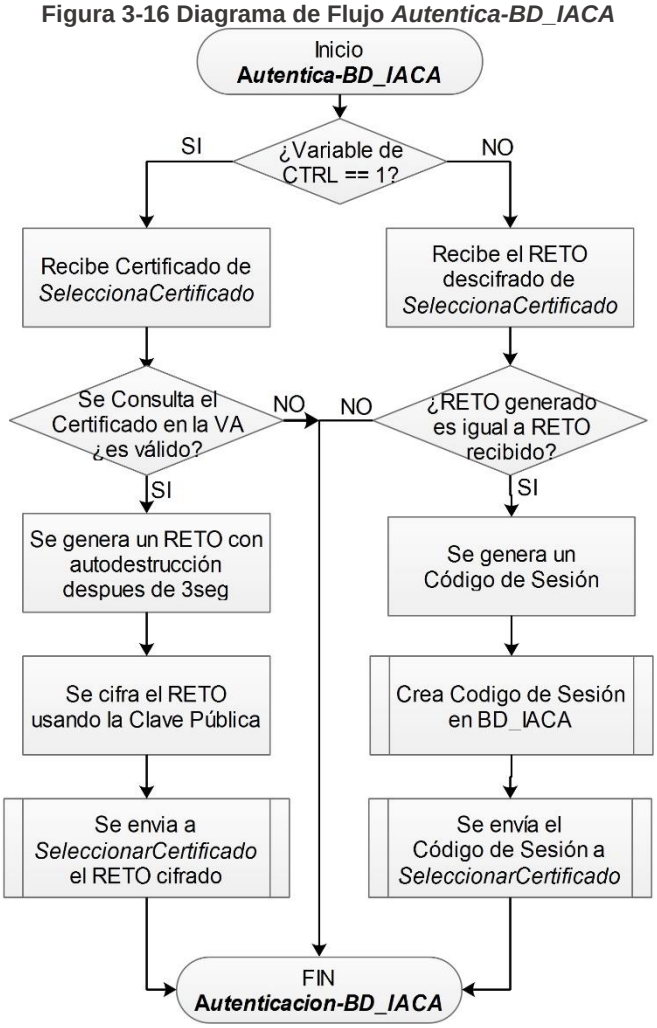
Figura 3-15 Diagrama de Flujo *SeleccionaCertificado*





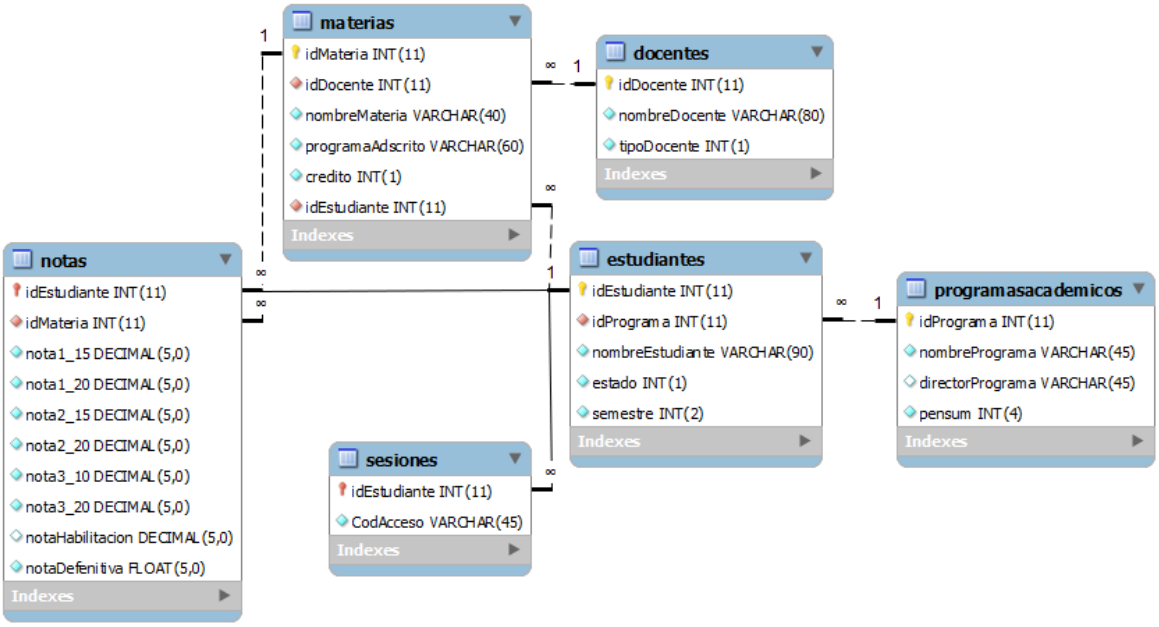
Fuente: Autor

Figura 3-16 Diagrama de Flujo *Autentica-BD_IACA*



Fuente: Autor

Figura 3-17 Modelo Entidad/Relación BD_IACA



Fuente: Autor

Como se observa en la Figura 3-17, las tablas son:

- **ProgramasAcadémicos**: Esta tabla almacena la información acerca del programa académico al que pertenece el usuario. Sus campos se muestran en la Tabla 3-10**¡Error! No se encuentra el origen de la referencia.**.

Tabla 3-10 Datos tabla *ProgramasAcadémicos* en BD_IACA

Nombre de Columna	Tipo	Nombre Descriptivo	Permite Valores nulos	Descripción
idPrograma	INT(11)	ID. programa	NO	Identificador único del programa para la bases de datos (llave primaria)
nombrePrograma	VARCHAR(45)	Nombre de programa	NO	Nombre del programa académico
directorPrograma	VARCHAR(45)	Director del programa	NO	Nombre del director del programa
pensum	INT (4)	Pensum	No	Pensum activo del programa académico

- **Docentes**: Relacionará la información de cada docente. Sus campos se muestran en la Tabla 3-11.

Tabla 3-11 Datos tabla *Docentes* en BD_IACA

Nombre de Columna	Tipo	Nombre Descriptivo	Permite Valores nulos	Descripción
idDocente	INT(11)	ID. docente	NO	Número de identificación del docente
nombreDocente	VARCHAR(80)	Nombres del docente	NO	Nombre completo del docente
tipoDocente	INT(1)	Tipo de docente	NO	Tipo de Vinculación del Docente con el centro de educación superior. Planta (1), Ocasional (2), Hora Cátedra (3).

Fuente: Autor

- **Materias**: Almacena la información de las materias o asignaturas que los docentes imparten y los estudiantes ven. Sus campos se muestran en la Tabla 3-12.

Tabla 3-12 Datos tabla *Materias* en BD_IACA

Nombre de Columna	Tipo	Nombre Descriptivo	Permite Valores nulos	Descripción
idMateria	INT(11)	ID. materia	NO	Identificador único de la materia (llave primaria)
idDocente	INT(11)	ID. docente	NO	Número de identificación del docente (llave foránea)
nombreMateria	VARCHAR(40)	Nombre de materia	NO	Nombre de la materia
programaAdscrito	VARCHAR(60)	Programa adscrito	NO	Nombre del programa al que pertenece la materia
credito	INT(1)	Créditos	NO	Créditos o peso de la materia
idEstudiante	INT(11)	Número de identificacion de estudiante	NO	Número de identificación del estudiante que tiene matriculada la materia (llave foránea)

Fuente: Autor

- **Estudiantes**: Esta tabla contiene información de los estudiantes que consultan su información académica. Sus campos se muestran en la Tabla 3-13.

Tabla 3-13 Datos tabla *Estudiantes* en BD_IACA

Nombre de Columna	Tipo	Nombre Descriptivo	Permite Valores nulos	Descripción
idEstudiante	INT(11)	ID. estudiante	NO	Identificador único del estudiante (llave primaria)
idPrograma	INT(11)	ID. programa	NO	Identificador del programa (llave foránea)
nombreEstudiante	VARCHAR(90)	Nombre del estudiante	NO	Nombre completo del estudiante
estado	INT(1)	Estado del	NO	Estado de estudiante que

Nombre de Columna	Tipo	Nombre Descriptivo	Permite Valores nulos	Descripción
		estudiante		indica su condición: nuevo regular (1) y antiguo activo (2)
semestre	INT(2)	Semestre	NO	Ubicación semestral del estudiante

Fuente: Autor

- **Notas**: Almacena las notas de los estudiantes por materias matriculadas. Sus campos se muestran en la Tabla 3-14.

Tabla 3-14 Datos tabla *Notas* en BD_IACA

Nombre de Columna	Tipo	Rango Válido de Valores	Nombre Descriptivo	Permite Valores nulos	Descripción
idEstudiante	INT(11)		ID. estudiante	NO	Identificación del estudiante (llave primaria y llave foránea)
idMateria	INT(11)		ID. materia	NO	Identificador de la materia (llave foránea)
nota1_15	DECIMAL	0,0 – 5,0	Nota 1 – 15%	SI	Nota quices y talleres del primer corte
nota1_20	DECIMAL	0,0 – 5,0	Nota 2 – 20%	SI	Nota examen parcial primer corte
nota2_15	DECIMAL	0,0 – 5,0	Nota 1 – 15%	SI	Nota quices y talleres del segundo corte
nota2_20	DECIMAL	0,0 – 5,0	Nota 2 – 20%	SI	Nota examen parcial segundo corte
nota3_10	DECIMAL	0,0 – 5,0	Nota 3 – 10%	SI	Nota quices y talleres del tercer corte
nota3_20	DECIMAL	0,0 – 5,0	Nota 3 – 20%	SI	Nota examen parcial tercer corte
notaHabilitacion	DECIMAL	0,0 – 5,0	Nota Habilitación	SI	Nota habilitación en caso que se realice
notaDefinitiva	FLOAT	0,0 – 5,0	Nota Definitiva	NO	Nota definitiva de la materia o parcial total

Fuente: Autor

- **Sesiones**: Esta tabla almacena los códigos de sesión que autenticacion-BD_IACA genera y son guardados en *cookies*. Esta tabla está conectada directamente con estas *cookies*, por lo que al finalizar sesión la *cookie* de entrada a la base de datos es borrada. Los campos que tiene esta tabla se muestran en la Tabla 3-15.

Tabla 3-15 Datos tabla *Notas* en BD_IACA

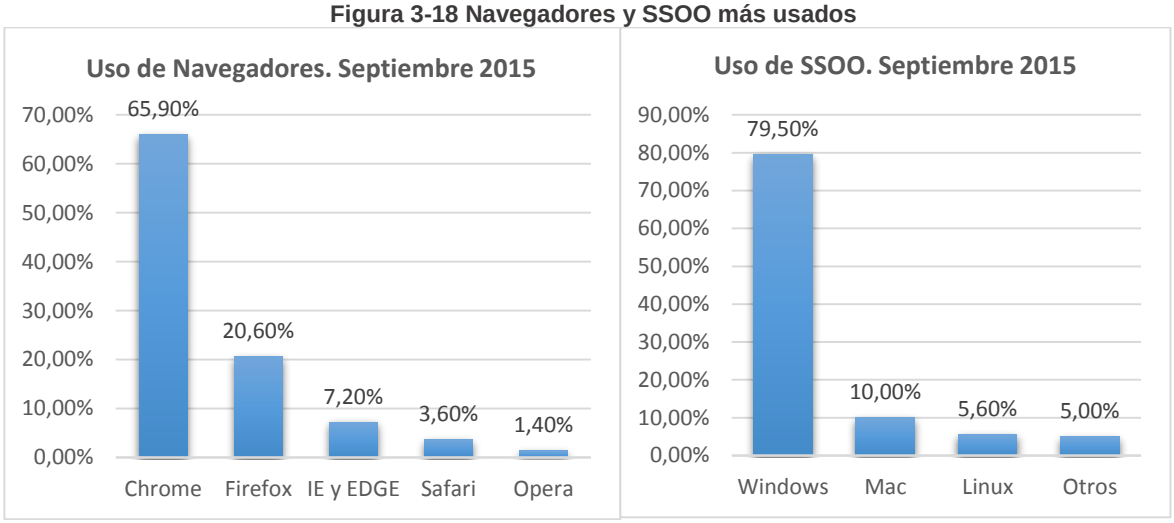
Nombre de Columna	Tipo	Nombre Descriptivo	Permite Valores nulos	Descripción
idEstudiante	INT(11)	ID. estudiante	NO	Identificación del estudiantes a loguearse (llave primaria y llave foránea)
CodAcceso	VARCHAR(45)	Código de acceso	NO	*Hash* del código de acceso a BD-IACA en SHA512

Fuente: Autor

3.2.4 Elección del Software de Desarrollo

Para que una Aplicación Web acceda a lectores y tarjetas inteligentes en el equipo del usuario debe tener acceso a los módulos PKCS11 o CSP (Microsoft *Cryptography API*) (son los más usados, estables y seguros actualmente), pero los lenguajes de programación Web como PHP, ASP.NET, HTML, PERL, etc. no tienen acceso directo a estos, lo deben hacer mediante el navegador de Internet, que debe ser compatible con alguno de los dos módulos. Cada navegador implementa módulos diferentes, mientras que PKCS11 tiene soporte en navegadores como Safari (Mac), Mozilla Firefox (Windows, Linux, Mac), Opera (Windows, Linux, Mac) y Chromium (Linux); PC/SC es usado por Chrome y Edge (antiguo Internet Explorer) sólo en sistemas operativos Microsoft Windows. Es una situación que pone en aprietos a cualquier desarrollador, pues si se quiere implementar una aplicación para la gran mayoría de usuarios debe tener soporte para PKCS11 y CSP, pues si se basa el desarrollo en solo un módulo dejaría por fuera un considerable número de posibles usuarios sin poder usar los servicios a crear. En la Figura 3-18 se observa la predominancia de Google Chrome como navegador y de Microsoft Windows como sistema operativo, cualquier desarrollador sensato al observar estas estadísticas de uso, tendería a diseñar e implementar su producto para CSP, aunque sería consciente que estaría dejando por fuera al 20,6% de usuarios (10% Mac, 5,6 Linux y 5% otros), lo cual, aun es una cantidad considerable. Ante la necesidad de lanzar un Aplicativo que funcione para todos los usuarios, sin importar el sistema operativo o del navegador que usen, desprenderse del soporte de los navegadores sería una opción viable. Todos los sistemas operativos pueden correr máquinas virtuales java y todos los navegadores pueden ejecutar aplicaciones basadas en Java, por ello, la mejor opción para acceder a las tarjetas inteligentes es diseñar e implementar una aplicación Java que pueda ejecutarse en los navegadores, a esta se le conoce como Applet, ofreciendo la posibilidad de usar el *middleware* IAIK-JCE. Por tanto, la Aplicación *SeleccionaCertificado* va a estar implementada como Applet Java desarrollada en IDE NetBeans.

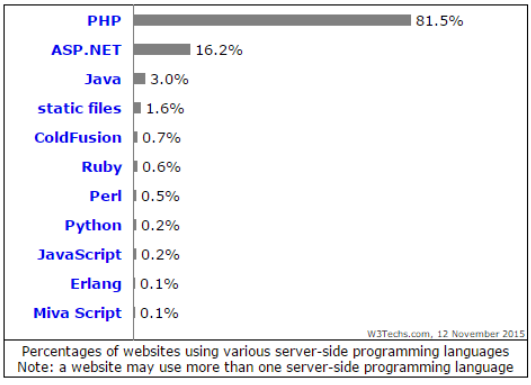
Figura 3-18 Navegadores y SSOO más usados



Fuente: Browser and OS Platform Statistics [71]

Por otro lado, el desarrollo de las páginas Web se realizará en PHP. PHP es un potente lenguaje, y su intérprete bien como módulo del servidor Web o bien como binario CGI (*Common Gateway Interface* o Interfaz de Entrada Común), puede acceder a ficheros, ejecutar comandos o abrir conexiones de red desde el servidor. PHP está diseñado específicamente para ser un lenguaje más seguro para escribir aplicaciones CGI que Perl o C. Partiendo de un correcto ajuste de las opciones de configuración para el tiempo de ejecución y en el tiempo de compilación, y el uso de prácticas de programación apropiadas, puede proporcionar la combinación de libertad y de seguridad que se necesita. En la Figura 3-19 se muestra que PHP es el lenguaje de programación Web más usado en este momento, lo que también justifica su uso en este proyecto.

Figura 3-19 Lenguajes de Programación Web en lado servidor más usados



Fuente: Usage of server-side programming languages for websites [72]

Capítulo 4

IMPLEMENTACIÓN DE LA INFRAESTRUCTURA DE CLAVE PÚBLICA (PKI)

La implementación de la PKI comprende la Aplicación de Registro de Usuarios, y la Aplicación Web, descritas en el Capítulo 3.

4.1 Herramientas Usadas

Las herramientas software utilizadas en este proyecto son:

- Aplicativo de Registro de Usuarios
    - *Java 1.8.0_25 Standard Edition*: Kit de desarrollo de Java
    - *NetBeans 8.0.2*: Entorno de desarrollo de Java.
    - *Provider IAIK*: Proporciona funciones criptográficas y la interfaz con la tarjeta inteligente.
    - *MariaDB*: Sistema de gestión de bases de datos derivado de MySQL con licencia GPL
    - *Openssl 1.0.1*: Paquete robusto de herramientas de administración y bibliotecas relacionadas con la criptografía
- Aplicativo Web:
    - *Java 1.8.0_25 Standard Edition*: Kit de desarrollo de Java
    - *NetBeans 8.0.2*: Entorno de desarrollo de Java.

- *Provider IAIK*: Proporciona funciones criptográficas y la interfaz con la tarjeta inteligente.
- *MariaDB*: Sistema de gestión de bases de datos derivado de MySQL con licencia GPL
- *PHP*: Lenguaje de programación Web
- *Openssl 1.0.1*: Paquete robusto de herramientas de administración y bibliotecas relacionadas con la criptografía.
- *Apache*: Servidor Web con intérprete PHP

En cuanto al hardware:

- Todos los programas mencionados se instalaron en un computador con procesador Pentium Dual Core 2,5GHz, memoria RAM de 4GB y sistema operativo Microsoft Windows 8.1
- Lector de Tarjetas Inteligentes ACR38 y ACR38T con conexión USB2.0
- Tarjetas Inteligentes ACOS5-32 de 32KB.

4.2 Certificados Digitales de las Autoridades

En esta sección se describe primero la generación del par de claves y el certificado de la Autoridad de Certificación. Este certificado es auto-firmado, es decir, que la misma CA emite su propio certificado. Luego, la CA emite los certificados: al servidor OCSP, al servidor Web, donde está albergado el Aplicativo Web, a la Autoridad de Registro y a la Autoridad de Validación. Estos procesos se realizan por medio del software OpenSSL.

Para crear la solicitud CSR auto-firmada de la CA se utiliza el comando:

```
openssl req -newkey rsa:2048 -keyout PathCA/ClavePrivada/private.pem -out PathCA/careq.pem
```

Así, se crea un par de claves de 2048 bits. La salida de este comando son los archivos `private.pem` y `careq.pem`, el primero almacena el par de claves cifrado con una contraseña simétrica y el segundo contiene la solicitud CSR. Ambos archivos están codificados en BASE64.

Para autofirmar la solicitud anterior, usando la clave privada generada, se ejecuta el comando:

```
openssl ca -create_serial -out PathCA/cacert.pem -batch -keyfile PathCA/ClavePrivada/private.pem -selfsign -extensions v3_ca -infiles PathCA/careq.pem
```

Debido a que el par de claves está cifrado con una contraseña simétrica, se debe ingresar esta para poder autofirmar el certificado de la CA con la clave privada de la misma. De esta forma se obtiene el certificado de la CA (`cacert.pem`).

Ahora, se procede a crear una solicitud CSR para el servidor OCSP (`ocspCSR.pem`), junto con su par de claves (`ocspParClaves.pem`), a través del comando:

```
openssl req -newkey rsa:2048 -sha512 -keyout PathOCSP/ocspParClaves.pem -out PathOCSP/ocspCSR.pem -extensions v3_OCSP
```

Este par de claves también se cifra con una contraseña.

Se debe firmar la solicitud (`ocspCSR.pem`) con la clave privada de la Autoridad de Certificación para generar el certificado(`ocspCerti.pem`):

```
openssl ca -in PathOCSP/ocspCSR.pem -out PathOCSP/ocspCerti.pem -extensions v3_OCSP
```

Una vez creados el par de claves y el certificado del servidor OCSP, estos se deben de almacenar en una ubicación segura.

Para crear el par de claves (`miservidor.key`) y el certificado digital del servidor del Aplicativo Web (`webCerti.pem`), se utilizan los siguientes comandos:

```
openssl req -nodes -newkey rsa:2048 -keyout miservidor.key -out webCSR.csr

openssl ca -in PathWEB/webCSR.csr -out PathWEB/webCerti.pem -notext
```

Para permitir la comunicación cifrada entre la Aplicación de Registro de Usuarios y las bases de datos, se genera el par de claves, la solicitud de firma CSR y el certificado digital de la RA (`conexBD_RA.key`, `conexBD_RA.csr` y `conexBD_RA.crt` respectivamente) y de la VA (`conexBD_VA.key`, `conexBD_VA.csr` y `conexBD_VA.crt` respectivamente), a través de los siguientes comandos:

```
openssl req -nodes -newkey rsa:2048 -keyout conexBD_RA.key -out conexBD_RA.csr

openssl ca -in conexBD_RA.csr -out conexBD_RA.crt -notext


openssl req -nodes -newkey rsa:2048 -keyout conexBD_VA.key -out conexBD_VA.csr

openssl ca -in conexBD_VA.csr -out conexBD_VA.crt -notext
```

De esa manera se habrán generado los certificados de cada entidad para cifrar la comunicación con las bases de datos.

Por otro lado, no es seguro almacenar la clave privada de la CA en el PC, aunque esté cifrada, por ello se importa a una tarjeta inteligente y se realiza un método de borrado seguro de estas claves. La importación se realiza por medio de Java, aprovechando las bondades que ofrece el *middleware* de IAIK. Se invoca una clase específica junto con dos parámetros, así:

```
ImportarParClaveCerti("PathCA", "ContraseñaClavePrivada")
```

`PathCA`: Es la ruta donde está creada la CA

`ContraseñaClavePrivada`: Contraseña secreta que se asignó a la clave privada

Una vez el proceso de importación a la tarjeta Inteligente termina, se ejecuta automáticamente la función que borra de forma segura todo rastro de la clave privada, después de 100 pasos, en los que sobrescribe byte a byte el fichero. En cada paso se borra el sobrescrito, y el siguiente paso sobrescribe nuevamente los sectores del disco duro que ocupó el fichero, como se observa en el código presentado a continuación:

```
public static void borradoSeguro(String archivo) throws IOException {
       File file = new File(archivo);
       if (file.exists()) { //evalua si el fichero existe
             RandomAccessFile raf = null;
             for(int i=0;i<100;i++){ //100 pasos
              long tamBytes = file.length(); //longitud en bytes del archivo
              SecureRandom random = new SecureRandom(); //función que crea datos aleatorios
                 System.out.println(random.getAlgorithm()); //algoritmo de aleatoriedad
              raf = new RandomAccessFile(file, "rws"); //abre el archivo en modo RWS que
desordena el archivo
              raf.seek(0); //empieza la sobre escritura desde el primer Byte
              byte[] data = new byte[64];
              int pos = 0;
              while (pos < tamBytes) {
                    random.nextBytes(data); //número aleatorio
                    raf.write(data); //se sobrescribe el fichero con el número aleatorio
                    pos += data.length; //avance de la sobrescritura
              }raf.close(); file.delete(); //borrado del fichero sobrescrito
            }raf.close(); file.delete(); //se borran los sectores utilizados
       }
}
```

La anterior función de borrado seguro, no garantiza que el archivo se halla eliminado totalmente, ya que su almacenamiento en el disco duro está por fuera de la jurisdicción de Java y la estructura de los sistemas operativos consiste en guardar cada parte de un archivo en diferentes clúster (grupo de sectores), y algunas veces los SSOO limitan su acceso en la tablas de particiones. Para implementar un método de borrado seguro eficaz aunque demorado, lo recomendado es sobrescribir todos los sectores que no se estén usando, pero esto conllevaría un tiempo excesivo para este proyecto. Así que al ejecutar esta

función lo que realmente borra es la marca de direcciones (que contiene datos para identificar el número y localidad del sector) pero los sectores de datos se sobrescriben.

4.3 Implementación de la Aplicación de Registro de Usuarios

4.3.1 Autoridad de Registro

4.3.1.1 Inicio de Sesión

El inicio de sesión de la Autoridad de Registro solo permite acceder a operarios autorizados, para evitar que personas inescrupulosas emitan CSRs. En Figura 4-1 se muestra la ventana de inicio de sesión creada.

Figura 4-1 Ventana Inicio de sesión de RA

Fuente: Autor

Cuando el operario pulsa el botón *Aceptar*:

I. Se inicializan los módulos presentes. En caso de no poder inicializar el módulo PKCS#11 se presentará el error: «Dispositivo no conectado».

```
import iaik.pkcs.pkcs11.Module; //se importa la clase Module de la librería iaik.jar
private Module pkcs11Modulo; //se crea un objeto privado de la clase Module
pkcs11Modulo = Module.getInstance("c:\acospkcs11.dll"); //el método getInstance devuelve un objeto tipo Module referenciado con la ubicación de la librería PKCS
pkcs11Modulo.initialize(null); //se inicializa el módulo
```

II. Se establece si la tarjeta está conectada, de lo contrario mostrará el error: «Tarjeta no insertada».

```
Slot[] slots =
pkcs11Modulo.getSlotList(Module.SlotRequirement.TOKEN_PRESENT);
```

III. Se inicia la sesión que solo tiene acceso a los objetos públicos de la tarjeta, y se le solicita al usuario que digite el PIN de la tarjeta para acceder a los objetos privados de la tarjeta inteligente. Si el PIN es incorrecto se presentará el error: «Contraseña no válida».

```
Session session =
token.openSession(Token.SessionType.SERIAL_SESSION,Token.SessionReadWriteBe
havior.RW_SESSION, null, null);
sesion.login(Session.UserType.USER, txtContrase.getPassword());//
txtContrase.getPassword() es un campo de entrada tipo contraseña
```

IV. En caso que el PIN haya sido ingresado correctamente, se comprueba que la tarjeta es de un operario. Para diferenciar las tarjetas de los operarios y de los estudiantes, las primeras tienen grabado un certificado que las identifica como tal, es decir, las tarjetas de los operarios tienen dos certificados, mientras que las de los estudiantes tienen solo uno. Este certificado especial es emitido por una entidad que no se implementa en esta tesis, por tanto es una función emulada.

La encargada de validar la tarjeta es una función que tiene por parámetro de entrada la sesión *User* que fue abierta en el paso III. Esta función se detalla en el Anexo B. En caso de no ser válida, se mostrará el error: «Tarjeta no válida», y si no puede cargar la foto mostrará: «Foto no cargó correctamente».

```
Boolean Resul = validarIntegridadTarjeta(sesion); //la función devuelve un parámetro
booleano (true o false).
```

V. Luego que se haya validado la integridad de la tarjeta, se hará la comprobación en la Autoridad de Validación para verificar el estado del certificado que el operario que está presentando. Solo aquellos certificados validos serán aceptados para continuar con el logueo, luego se generará un reto cifrado con la clave pública del certificado del operario (recordar que la tarjeta de operario tiene dos certificados uno de tarjeta o de operario) para que lo descifre con su clave privada. Los errores que se pueden producir cuando se esté verificando el certificado son: «Certificado revocado», «Certificado desconocido», «Error al comprobar certificado», «No se pudo establecer conexión con el servidor OCSP».

```
Boolean Resul = verificarOCSP(IDObjectosNoRepe.get(0),sesion); //verifica con la
Autoridad de Validación, específicamente con el servidor OCSP. La variable Resul está siendo reutilizada.
String Reto = String.parseString((new Random()).nextLong);
Byte[] retoCifrado = CifrarCPub(Reto, objectTokenClavePublica);
String retoDescifrado = DescifrarReto (retoCifrado, tokenClavePrivada);
If(Reto == retoDescifrado){ … }
```

VI. Una vez comprobado que la tarjeta y el operario son válidos, se prosigue a leer los datos que están almacenados en la tarjeta, tales como: foto, nombre,

apellido, cargo. Estos son parámetros de la sesión abierta en el paso III, ya que se necesita acceder a objetos privados de la tarjeta. Ver Anexo B.

```
leerDatos(sesion); //Lectura de datos en la tarjeta
```

VII. Por último, se realiza la comprobación de conectividad con la Bases de Datos de Usuarios (BD_USU). Si no se logra establecer conexión con BD_USU el aplicativo de Autoridad de Registro no podrá efectuar ninguna operación. En caso no superar este paso, se presentará el error: «No se puede establecer conexión con la base de datos».

```
Boolean Resul = (new conexionBD("BD_USU")).stm //se trata de una clase, cuyo método
stm devuelve un booleano. Nuevamente la variable Resul es reutilizada.
```

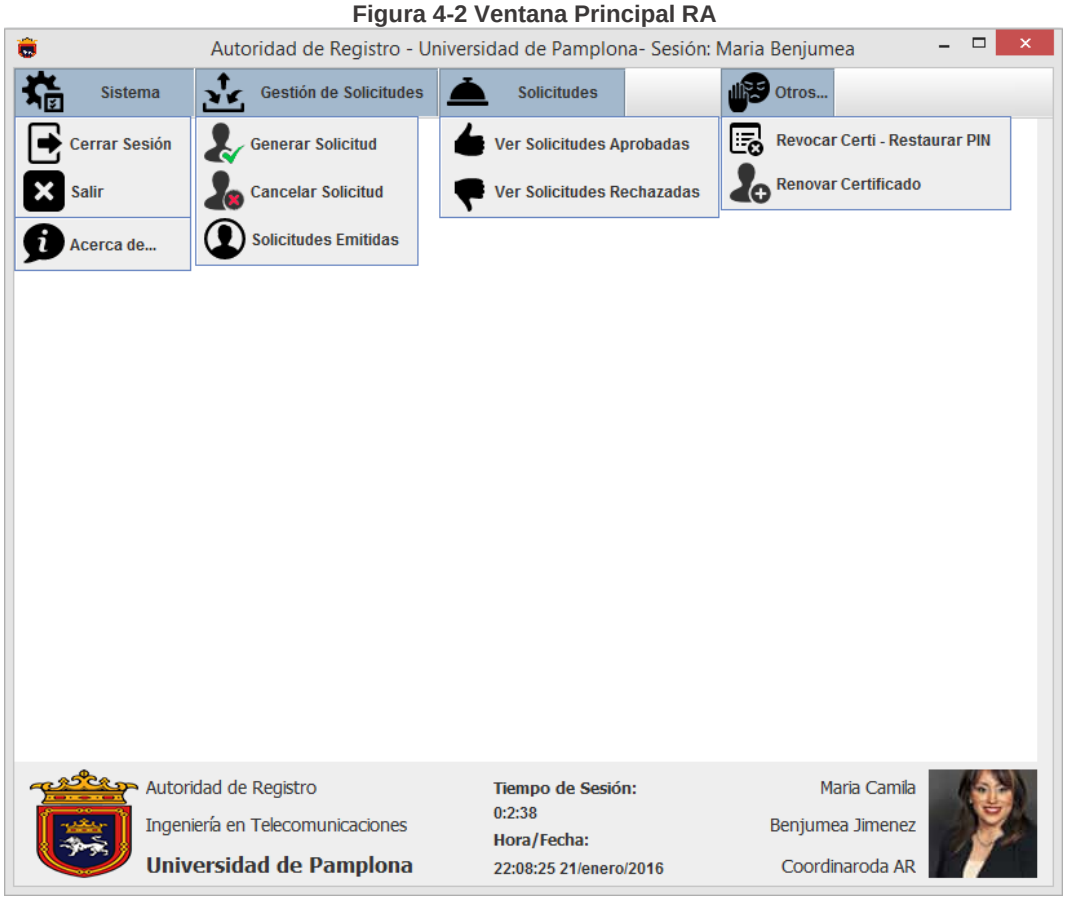
VIII. Todos los pasos anteriores son transparentes al usuario, excepto el ingreso del PIN, lo que le da a esta ventana un alto nivel de usabilidad. Una vez aprobados todos los pasos anteriores, la siguiente sentencia permitirá desplegar la ventana principal. La foto y los datos del operario que fueron obtenidos son enviados a la ventana principal.

```
VentanaPPAL ppal=new VentanaPPAL(FotoOperario,Datos_Usuario);
ppal.setEnabled(true);
ppal.setVisible(true);
this.dispose();
```

4.3.1.2 Ventana Principal

A través de la ventana principal se puede acceder a todas las funciones que realiza la Autoridad de Registro. Consta de un menú, información del operario, y un espacio donde se pueden visualizar los datos de las diferentes funciones, como se muestra en la Figura 4-2.

Figura 4-2 Ventana Principal RA



Fuente: Autor

Además de albergar la ejecución de las funciones de la autoridad de registro, la ventana principal debe detectar cuando la tarjeta inteligente con la cual el operario inició sesión sea retirada del lector, en tal caso, la sesión se cerrará automáticamente, garantizando la seguridad de la Autoridad de Registro, evitando que una persona no autorizada manipule el software. La siguiente sentencia realiza la comprobación de la conexión de la tarjeta inteligente en tiempo semi-real:

```
(new HiloCompruebaTarjeta()).start(); //Clase derivada de la clase Thread que permite la ejecución de procesos en paralelo (multitarea). El proceso paralelo trata una secuencia infinita de preguntas sobre si está conectada la tarjeta inteligente que se usó para iniciar sesión.
```

El menú de la ventana principal cuenta con tres ítems: ***Sistema***, que permite cerrar sesión, salir de la aplicación o ver quien desarrolló la aplicación; ***Gestión de Solicitudes***, que permite generar y cancelar las solicitudes, además de visualizar las solicitudes emitidas; ***Solicitudes***, que permite ver las solicitudes aprobadas y rechazadas; y **Otros**, permite acceder a las opciones donde se pueden emitir solicitudes de revocación de certificados digitales o restaurar el PIN de una tarjeta

bloqueda (Revocar Certi – Restaurar PIN), además de solicitar la renovación de un certificado digital. A continuación se explica cómo se implementaron las diferentes funciones de ***Gestión de Solicitudes***, ***Solicitudes*** y ***Otros***.

4.3.1.2.1 Generar Solicitud

La función *Generar Solicitud* hace parte del menú *Gestión de Solicitudes* y le permite al operario registrar al usuario, crear el par de claves en la tarjeta inteligente, crear la solicitud CSR e importar el certificado a la tarjeta, una vez haya sido generado por la Autoridad de Certificación. La ventana, básicamente, es un formulario con campos de ingreso obligatorios. Este formulario se ejecuta sobre un escritorio virtual de la clase `JDesktopPane` (ver Figura 4-3). Los datos ingresdos son grabados en la Tarjeta Inteligente y la Base de Datos de Usuarios (BD_USU), según corresponda, una vez se presione el botón **Generar CSR**, ubicado en la parte inferior de la ventana.

El proceso que se lleva a cabo, tras presionar dicho botón, es el siguiente:

I. Se verifica si el usuario que hace la solicitud ya tiene un certificado válido. Si es así, su solicitud es rechazada.

```
Boolean Resul = verificarCertificadoID(String.parseLong(NumeroID));
```

II. Se comprueba que además de la tarjeta del operario, haya conectada otra tarjeta inteligente en blanco para generar el par de claves y demás archivos necesarios para autenticar al usuario. En caso de no estar conectada la tarjeta se visualizará una ventana de diálogo con el mensaje «No se puede continuar. Inserte otra tarjeta para generar el CSR».

```
Boolean Resul = compruebaTarjetaUsuarioConectada();
```

Figura 4-3 Ventana Generar Solicitud RA

Fuente: Autor

III. Una vez comprobado que la tarjeta insertada es apta, se genera el par de claves en la tarjeta inteligente. Esta operación la realiza la tarjeta, evitando que la clave privada se pueda extraer o copiar de la misma.

```
import iaik.pkcs.pkcs11.Mechanism; //se importa la clase Mechanism de la librería IAIK.jar
Mechanism MecaGenParClaves = Mechanism.RSA_PKCS_KEY_PAIR_GEN; //se define el mecanismo que usará la tarjeta para la creación del par de claves
import iaik.pkcs.pkcs11.objects.RSAPublicKey; //se importa la clase RSAPublicKey de la librería IAIK.jar. Se trata de una clase que genera plantilla para diseñar la clave pública que generará la tarjeta inteligente
RSAPublicKey PantillaRSAClavePublica = new RSAPublicKey(); //se crea el objeto que definirá la plantilla de la calve pública.
RSAPrivateKey PantillaRSAClavePrivada = new RSAPrivateKey(); //se crea la plantilla de la calve privada
```

En las anteriores líneas de código, la tarjeta no crea el par de claves, pues en ningún momento se estableció su valor; estas son instrucciones de como la tarjeta debe manipular las claves a crear. De igual forma se importa la

fotografía del estudiante y el archivo en texto plano con la información introducida en el formulario.

Una vez creada las instrucciones o pantillas de la claves privadas y publicas, se da la orden a la tarjeta para que cree el par de claves, por medio del comando:

```
KeyPair generatedKeyPair = sesion.generateKeyPair(MecanisnoGenParClaves, PantillaRSAClavePublica, PantillaRSAClavePrivada); //instrucción dirigida a la tarjeta que crea el par de claves en base a las plantillas creadas. Se debe usar una sesión tipo User
```

IV. Con el par de claves creado y la clave privada almacenada en secreto, se crea la solicitud de firma del certificado (CSR). La CSR contiene la clave pública del usuario, información que identifica al usuario (datos que fueron ingresados en el formulario) y debe ser firmada digitalmente por el usuario y por el operario[45], es decir, que la solicitud CSR estará doblemente firmada. Las firmas digitales se agregan a la CSR usando la función de *hash* SHA512. Las siguientes sentencias permiten la creación de la CSR, firmada con la clave privada del usuario y la exportación de la solicitud CSR en un archivo:

```
import iaik.pkcs.pkcs10.CertificateRequest; //se importa la clase CertificateRequest de la librería IAIK.jar
CertificateRequest SolicitudCerticado = new
CertificateRequest(AsuntoClavePublica, Asunto); //se define la información del usuario que será incluida en la CSR. Se trata de un constructor de la clase CertificateRequest que recibe por parámetro AsuntoClavePublica (contiene información sobre la longitud del exponente público y el módulo de la clave pública) y Asunto (contiene la información del usuario en el formato Nombre Distinguido)
SolicitudCerticado.sign(AlgoritmodeFirmaPKCS11,TokenClavePrivada); //se firma digitalmente la solicitud CSR con la clave privada del usuario. El método tiene por entrada el algoritmo que se usará para firmar el cual es SHA512 y el token de la clave privada.
String LineaInicial = "-----BEGIN NEW CERTIFICATE REQUEST-----"; //información de cabecera para adaptar la solicitud CSR a la norma PKCS#10
String LineaFinal =  "-----END NEW CERTIFICATE REQUEST-----"; //información de finalización para adaptar la solicitud CSR a la norma PKCS#10
registrarNuevoUsuario(DatosEstudiante,datosOperario);//se registra el nuevo usuario en la tabla Usuarios de la Base de Datos de Usuarios(BD_USU).
```

Es importante destacar el hecho de que todas las operaciones que requieren el uso de la clave privada se realizan en la tarjeta inteligente y no en el computador. Los métodos de Java encargados de realizar las operaciones delegan al proveedor IAIK sus funciones, ya que es el único que puede tener acceso a la tarjeta inteligente. Por esta razón, siempre es necesario

[45] Al momento de crear la solicitud CSR es indispensable que esté firmada por la clave privada del usuario, ya que así se demuestra la posesión de la clave privada ante la Autoridad de Certificación. Pero en esta implementación la solicitud CSR, adicionalmente, estará firmada digitalmente por el operario, ya que así la Autoridad de Certificación podrá comprobar que realmente la generó un operario autorizado, evitando suplantación de solicitudes CSR.

especificar el nombre del proveedor en la instanciación de un método criptográfico.

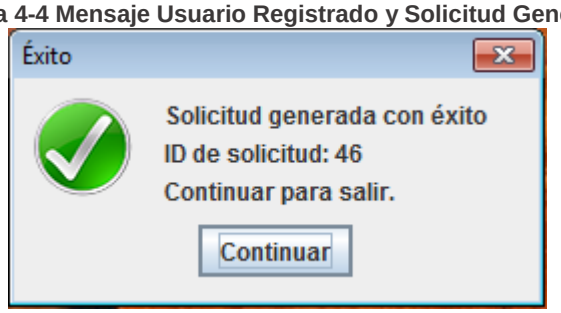
V. Creada la solicitud CSR, se procede a firmarla digitalmente usando el estándar PKCS#7 y la clave privada del operario, por ende, el aplicativo deberá tener la facultad de conmutar de la tarjeta del usuario a la del operario. El proceso de agregado de la firma digital del operario a la solicitud CSR se realiza bajo las siguientes instrucciones (para ver el proceso en forma detallada diríjase al Anexo B).

```
import iaik.pkcs.pkcs11.objects.PrivateKey; //se importa la clase PrivateKey de la librería IAIK.jar
PrivateKey PlantillaRSAClavePrivada = new PrivateKey(); //se crea un plantilla de clave privada para buscar todas las claves privadas en la tarjeta
sesion.findObjectsInit(PlantillaRSAClavePrivada); //se prepara en la tarjeta la búsqueda de las claves privadas
CSRfirmado.writeTo(CSRFirmado_Archivo); //se escribe la información en el archivo
subirCSR(CSR_Archivo,EmitidoA,datosOperario); //se sube la solicitud CSR doblemente firmada, tanto por el usuario como por el operario, a la tabla Solicitudes de la Base de Datos de Usuarios (BD_USU)
```

La solicitud CSR fue enviada a la tabla Solicitudes de BD_USU, que la Autoridad de Certificación consultará y para analizar la solicitud.

Al final se informará al operario que la generación de la solicitud CSR fue exitosa (ver Figura 4-4) o falló. El mensaje de confirmación contiene el ID de la solicitud.

Figura 4-4 Mensaje Usuario Registrado y Solicitud Generada



Fuente: Autor

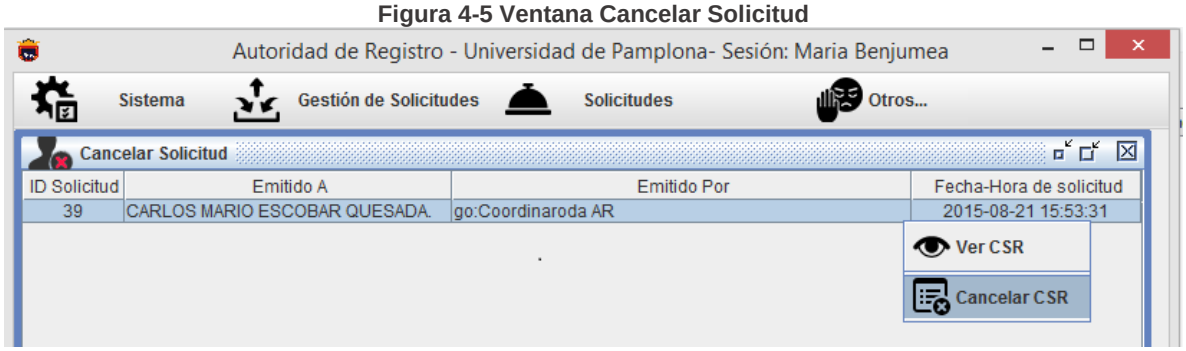
4.3.1.2.2 Cancelar Solicitud

Cancelar Solicitud hace parte del menú *Gestión de Solicitudes* y da la posibilidad al operario de emendar sus errores, en caso que haya cometido algún error en el momento de introducir los datos del usuario para generar la solicitud. Al ingresar a esta opción se muestra una tabla con la información de las solicitudes generadas por el operario y con interacción directa del mouse, se puede ver el contenido de la solicitudes de firma de certificado (CSR) o cancelarla (ver Figura 4-5). La cancelación se podrá realizar siempre y cuando la solicitud aún no haya sido

firmada por la Autoridad de Certificación. Para futuros procedimientos de auditoria, la solicitud no será eliminada de la BD_USU, solo cambiará su estado.

```
int aux = BD_solicitudes.CancelarSolicitud(jTable1.getModel().getValueAt(jTable1.getSelectedRow(), 0).toString());//cambia el estado de la solicitud

((DefaultTableModel)jTable1.getModel()).removeRow(jTable1.getSelectedRow());
//borra la fila de la tabla una vez ha sido actualizada la nueva situación de la solicitud
```

Figura 4-5 Ventana Cancelar Solicitud



Fuente: Autor

4.3.1.2.3 Solicitudes Emitidas

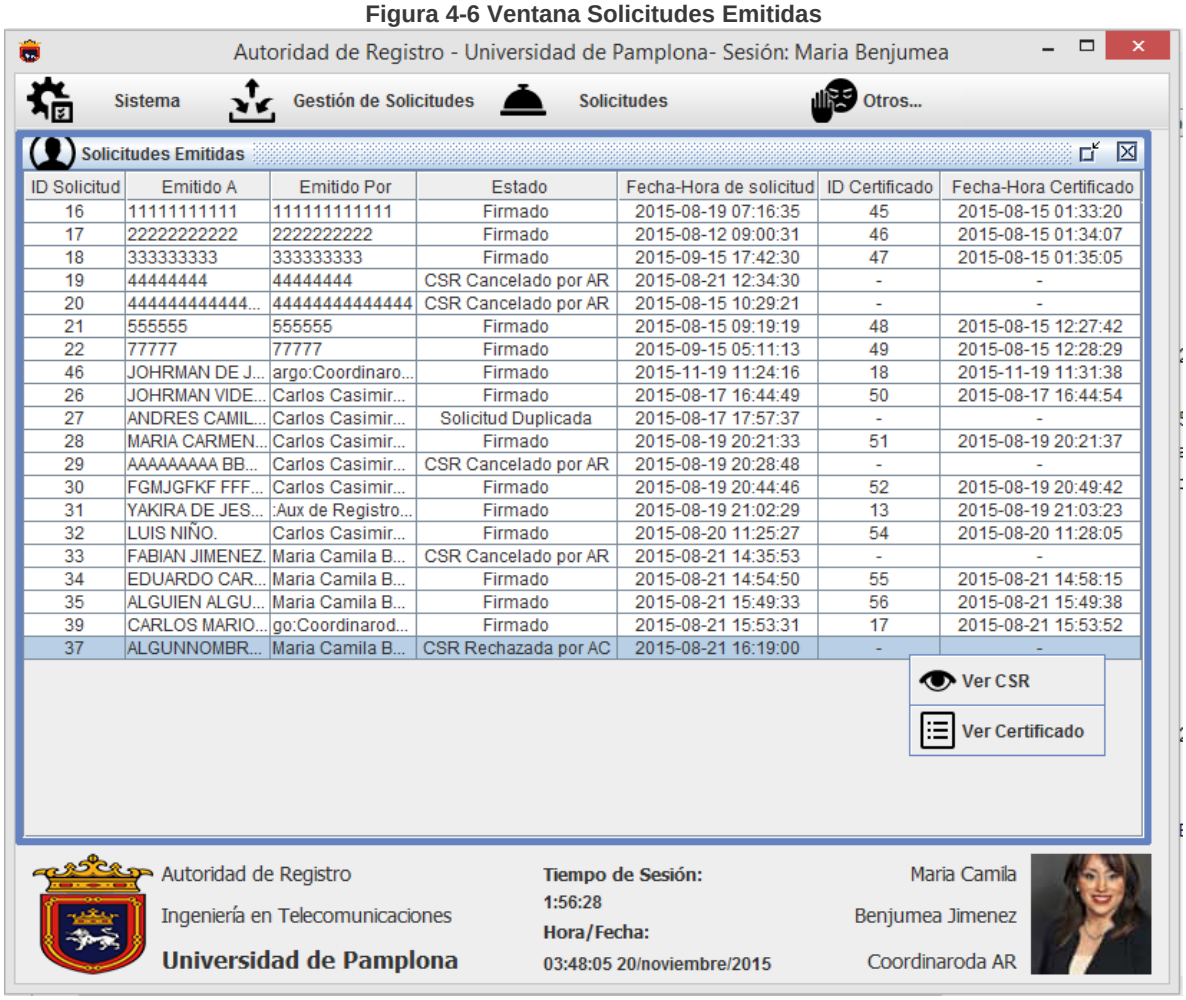
La función *Solicitudes Emitidas* hace parte del menú *Gestión de Solicitudes* y presenta una tabla informativa, que permite al operario conocer el estado de las solicitudes emitidas (ver Figura 4-6). Si el operario pulsa el click derecho sobre un campo de la tabla, se despliega una ventana con las opciones «Ver CSR» y «Ver Certificado». Los posibles estados de una CSR son:

- **No Firmado CA:** Se presenta cuando la solicitud CSR ha sido enviada a la Autoridad de Certificación pero esta no la ha firmado.
- **Firmado:** Denota que la solicitud CSR ya fue aprobada satisfactoriamente por la Autoridad de Certificación.
- **Solicitud Duplicada:** Indica que se ha generado dos o más veces la solicitud CSR para un usuario determinado.
- **Falló Firma CSR:** Error presentado en el momento que la Autoridad de Certificación firmó la solicitud CSR.
- **CSR Rechazada por CA:** La solicitud CSR ha sido rechazada por la Autoridad de Certificación, ya sea por políticas internas o porque la solicitud no pasó la verificación de la firma.
- **CSR Cancelada por RA:** La solicitud que ha sido cancelada por la Autoridad de Registro. No tendrá ninguna validez.

El proceso para rellenar la tabla con el estado de las Solicitudes emitidas, comienza a partir de una consulta total recursiva hecha a la Base de Datos de Usuarios (BD_USU). Las sentencias generales de esta función son:

```
ArrayList<String[]> todas_Solicitudes = BD_USU.listarSolicitudes(); //consulta recursiva que aprovecha el potencial de la implementación de bases de datos relacionales con el fin de extraer toda la información a mostrar en la tabla
Solicitudes_Emitidas csremitidos= new Solicitudes_Emitidas(todas_Solicitudes,BD_solicitudes); //llamado a la función que se encarga de llenar la tabla en base a la consulta hecha.
```

Figura 4-6 Ventana Solicitudes Emitidas



Fuente: Autor

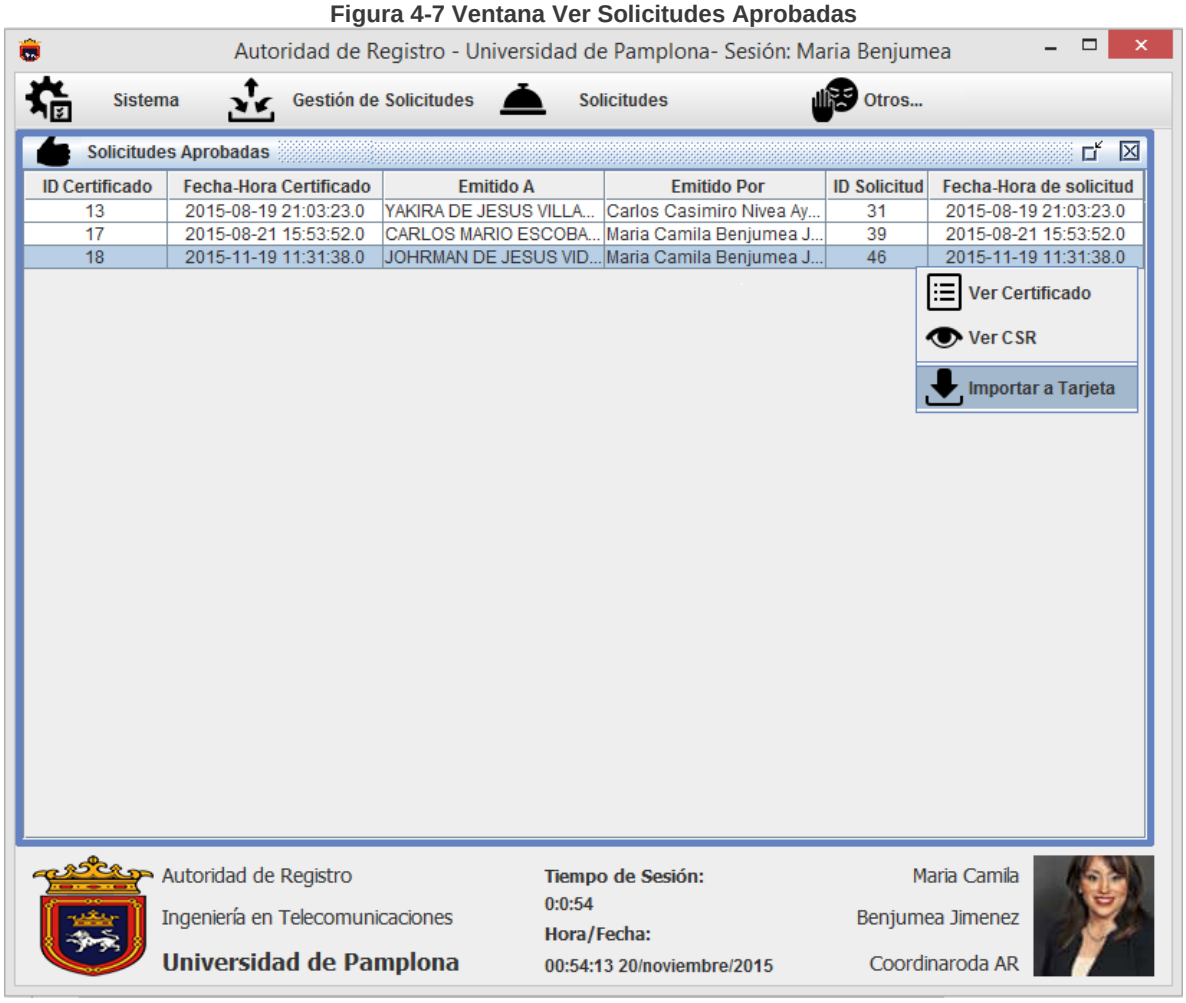
4.3.1.2.4 Solicitudes Aprobadas

Esta función hace parte del menú *Solicitudes* y permite importar los certificados digitales, cuyas solicitudes CSR fueron aprobadas por la Autoridad de

Certificación. También permite visualizar el contenido del certificado aprobado y su solicitud CSR.

La ventana consta de una tabla que se llena conforme las solicitudes se van aprobando. Al oprimir el click derecho del mouse se muestra un submenú con las opciones «Ver Certificado», «Ver CSR» e «Importar a Tarjeta», como se muestra en la Figura 4-7.

Figura 4-7 Ventana Ver Solicitudes Aprobadas



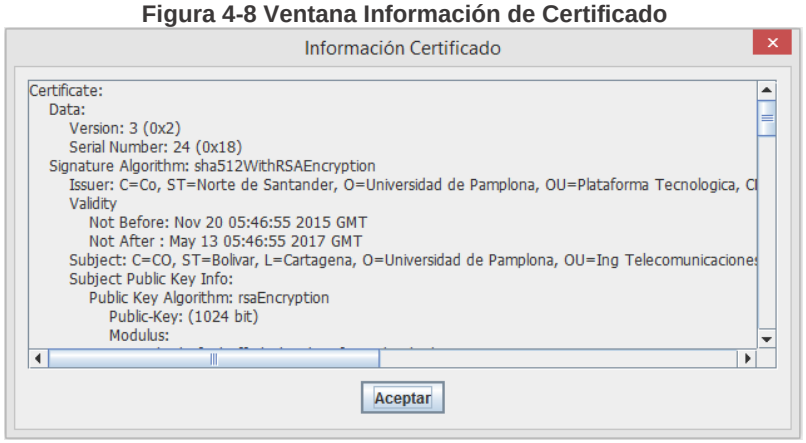
Fuente: Autor

La función de cada submenú es la siguiente:

- **Ver Certificado:** Se encarga de descargar el certificado almacenado en BD_USU, ejecutar comandos `openssl` para extraer su contenido en texto plano con estructura ASN.1 y por último imprimir esta información en un `JOptionPane`.

  La Figura 4-8 muestra la información del certificado en un `JOptionPane`.

Figura 4-8 Ventana Información de Certificado

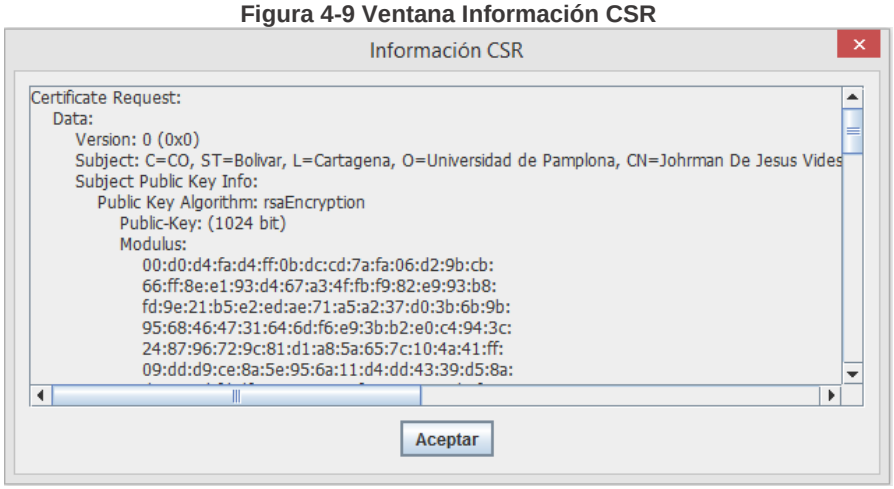


Fuente: Autor

- **Ver CSR:** Es similar a Ver Certificado, pero se descarga la solicitud CSR de la tabla Solicitudes de BD_USU. Las sentencias principales son:

  La Figura 4-9 muestra la información de la CSR que se despliega.

Figura 4-9 Ventana Información CSR



Fuente: Autor

- **Importar Certificado:** Permite guardar el certificado digital en la tarjeta inteligente. El proceso de importación se realiza por medio de las instrucciones:

  I. Inicialmente se debe comprobar que haya otra tarjeta insertada aparte de la del Operario. Esta tarjeta debe contener en su interior el par de claves correspondientes al certificado digital a importar. En caso de no detectar otra tarjeta inteligente el aplicativo arrojará el mensaje de error: «No se puede continuar. Inserte otra tarjeta para Importar el certificado», y si no se hayan las claves correspondientes: «No se puede importar el certificado. Tarjeta no posee la correspondiente clave privada».

```
int idCert =
Integer.parseInt(tablaCSRaprobados.getModel().getValueAt(tablaCSRap
robados.getSelectedRow(), 0).toString()); //se toma el ID del certificado en al tabla
que el usuario ha seleccionado
InputStream  CRTinputStream=BD_USU.importar_CERT(Integer.toString(idCert
)); //se descarga el certificado de la Bases de Datos de Usuarios de la tabla CertificadosAprobados
PublicKey publicKey = x509Certificate.getPublicKey(CRTinputStream); //se
obtiene la clave pública del certificado
sesion.createObject(pkcs11X509PublicKeyCertificate); //se da instrucción a la
tarjeta para que importe la plantilla creada.
```

Como se apreció, para corroborar que el par de claves existan en la tarjeta, se busca una clave privada con el módulo del certificado. En caso de haberla encontrado, se crea una plantilla que lleva todos los valores del certificado que fue emitido por la Autoridad de Certificación, esta plantilla luego es importada a la tarjeta inteligente completando los datos necesarios para poder autenticar al usuario.

II. Cambiar el PIN antes de hacer la entrega de la tarjeta es una buena práctica de seguridad, pues permite que cada tarjeta posea un PIN diferente. El PIN será ingresado por el usuario una vez este reclame su tarjeta inteligente en la Autoridad de Registro, y deberá contener por lo menos 6 digitos numéricos. La siguiente sentencia está basada en el uso de las librerías de IAIK.jar.

```
sesion.setPIN(ultDigitos.toCharArray(),(NumeroID.getText().substring(Num
eroID.getText().length()-6,
NumeroID.getText().length())).toCharArray()); //cambiar el PIN consta de una sola
sentencia. La sesión deber estar abierta en tipo User, y ser usada con el método setPIN, ingresando el PIN
actual y el nuevo PIN. El PIN actual es 12345678 que es el PIN por defecto de todas las tarjetas.
```

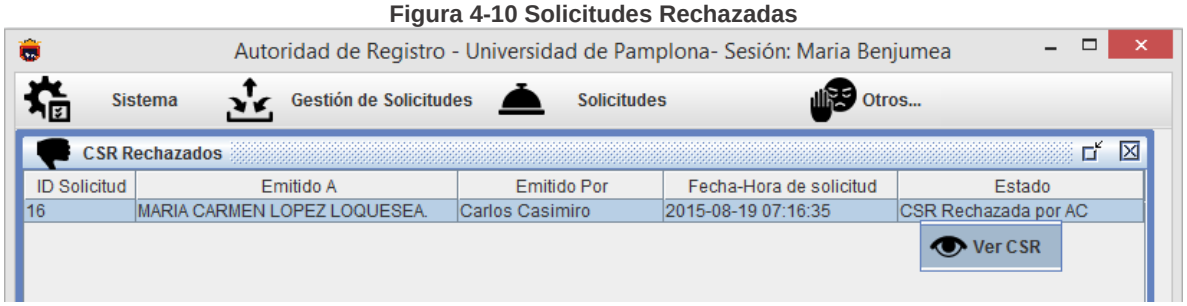
III. Se procede a configurar en la base de datos de usuario el número de intentos válidos para iniciar sesión antes que esta se bloquee

```
bd_usu.ponerNintentos(idUsuario,"3"); //establece que el número de intentos antes de
bloquear la tarjeta es 3 al usuario User
```

4.3.1.2.5 Solicitudes Rechazadas

Esta función hace parte del menú *Solicitudes* y permite listar todas las solicitudes CSR que hayan sido rechazadas por la Autoridad de Certificación (ver **¡Error! No e encuentra el origen de la referencia.**).

Figura 4-10 Solicitudes Rechazadas



Fuente: Autor

El siguiente código muestra como se obtiene la información que se presenta en la ventana principal:

```
String col[] = {"ID Solicitud","Emitido A","Emitido Por","Fecha-Hora de solicitud","Estado"}; //se define un vector tipo String para nombrar las columnas de la tabla
modelo = new DefaultTableModel(col) { }; //se define un modelo de tabla con los nombres y número de columnas
TablaSolRecha.setModel(modelo); //se agrega el modelo a la tabla TablaSolRecha
```

De igual forma que las tablas anteriores, esta también responde al click derecho del mouse, característica que se logra con el comando `TablaSolRecha.addMouseListener(new MouseAdapter(){})`, así se muestra la opción «Ver CSR».

4.3.1.2.6 Revocar Certificado – Restaurar PIN

Esta función permite emitir la solicitud de revocación de certificados digitales a la Autoridad de Certificación quien será la encargada de efectuar la operación final, pues es ella quien puede hacer dicha operación con su par de clave privada (Ver Figura 4-11).

Figura 4-11 Revocar Certificado – Restaurar PIN

Fuente: Autor

Inicialmente se debe realizar la busque ya sea por el nombre, numero de ID del usuario o por el nombre del programa los cuales se visualizarán en la tabla después de haber presionado el botón *Buscar,* se listarán cada uno de los certificados según los criterios ingresado y se podrá escoger tres opciones: *Ver Certificado, Revocar, Restaurar PIN.*

- **Revocar**: Permite emitir la solicitud de revocación a la Autoridad de Certificación.

```
conexionBD bd_usu = new conexionBD("bd_usu"); //conecta a la base de datos
int aux =
bd_usu.revocarCRT(jTable1.getModel().getValueAt(jTable1.getSelected
Row(), 3).toString()); //Cambia el eestado del certificado a 3 en BD_USU
if(aux!=0){ JOptionPane.showMessageDialog(null,"Revocado Con
éxitto","Revocar certificado",JOptionPane.DEFAULT_OPTION);} //Solitud de
revocación enviada
else{ JOptionPane.showMessageDialog(null,"NO se puedo revocar el
certificado","Revocar certificado",JOptionPane.ERROR_MESSAGE);}
//Solitud de revocación enviada
```

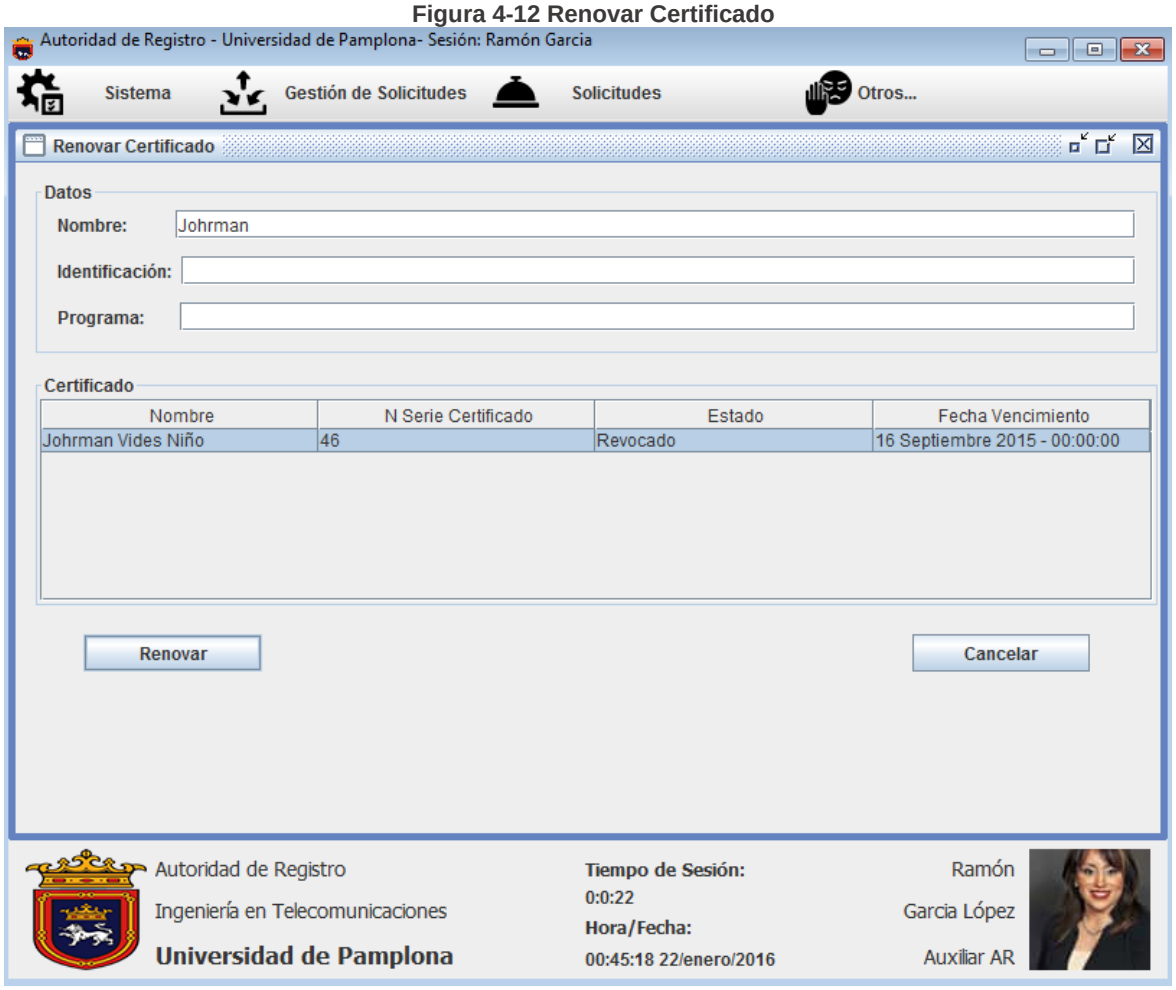
- **Restaurar PIN:** Esta opción restaura el PIN que ha sido bloqueado por exceso de intentos de sesión invalidos. Sólo el usuario puede tramitar este tipo de solicitud ante la Autoridad de Registro. donde primero se comprueba que la

tarjeta inteligente está ingresada, luego pregunta por el nuevo PIN y setea el nuevo de intentos en la BD_USU nuevamente a 3 intentos.

4.3.1.2.7 Renovar Certificado

El proceso de renovación de un certificado digital solo se puede hacer cuando su estado sea Revocado. Se podrá buscar por el nombre, ID de usuario o por el programa académico que pertenece. Una vez se halle un certificado revocado y se pulse el botón renovar (ver Figura 4-12) se formateará la tarjeta inteligente nuevamente dejándola en blanco, para asi crear nuevamente una par de claves junto con una solicitud CSR que nuevamente será enviada a la Autoridad de Certificado con una nueva solicitud.

Figura 4-12 Renovar Certificado



Fuente: Autor

4.3.2 Autoridad de Certificación

Esta autoridad emite y revoca los certificados digitales. El inicio de sesión se hace como se realizó en la Autoridad de Registro.

A continuación se describe el funcionamiento de la Ventana Principal de la CA.

4.3.2.1 Ventana Principal

Como se aprecia en la Figura 4-13, la ventana principal se compone de un menú, una parte central, para visualizar la información de las diferentes opciones, y un botón, tipo *toggle* para el piloto automático.
La revocación de Certificados no se lista, se aprueban automaticamente una vez la Autoridad de Registro haya enviado la solicitud, y las solicitudes de renovación de certificados serán listadas como solicitudes pendientes. Para la revocación automática cada 30 segundos se analiza BD_USU en busca de solicitudes de revocación si se encuentra alguna la Autoridad de Certificación revocará el certificado. Por medio del hilo `(new RevocarCertificado()).start();` se efecturá la operación.

Figura 4-13 Ventana Principal CA

Fuente: Autor

El menú de la ventana principal cuenta con tres ítems: ***Sistema***, que permite salir de la aplicación o ver quien desarrolló la aplicación; ***Certificados***, que permite ver los certificados firmados y revocados; y ***Solicitudes***, que permite ver las CSRs pendientes y rechazadas. A continuación se explica cómo se implementaron las diferentes funciones de ***Certificados*** y ***Solicitudes***, así como el Piloto Automático.

4.3.2.1.1 Certificados Firmados

Esta función hace parte del menú *Certificados* y presenta una tabla con todos los certificados firmados por la CA. Esta tabla ofrece información como: ID del certificado (Número de serie), nombre del propietario del certificado, estado del certificado (válido o revocado), fecha y hora de emisión del certificado. Además, se puede visualizar y revocar un certificado, por medio del submenú desplegado al presionar el click derecho del mouse, que se observa en la **¡Error! No se ncuentra el origen de la referencia.**.

Para visualizar los certificados firmados, se utiliza el siguiente código:

I. Antes de llenar la tabla, el aplicativo analiza cada certificado en busca de los que estén vencidos.

```
write(BD_USU.obtenerCertificado(jTable1.getModel().getValueAt(jTable1.getSe
lectedRow(), 0).toString())); //esta línea está incluida en un ciclo FOR, y permite la descarga y
escritura de un certificado
process= cmd.exec("openssl asn1parse -i -in "\\certificado.pem"); //usando
openssl se extrae la información ASN.1 del certificado
String Aux = process[35].partesa[1].substring(1, 13); //se extrae la fecha final del
certificado
```

La función *RevocarCertificado(Boolean VarCTRL),* se detalla en el Anexo B.

II. Una vez cumplida la verificación de cada certificado, se procede a mostrarlos en la tabla, de forma que solo se visualicen los certificados vigentes.

III. El operario de la Autoridad de Certificación tiene dos opciones en el submenú, «Ver Certificado» que ejecuta la función *MostrarCerti()* y «Revocar Certificado» llamando a la función *RevocarCertificado(true)*.

IV. En cada operación de revocación, ya sea donde se detectó que el certificado no estaba vigente o en la revocación a solicitud del usuario se invocan funciones que actualizan sus estados en las bases de datos BD_USU y/o BD_REPO.

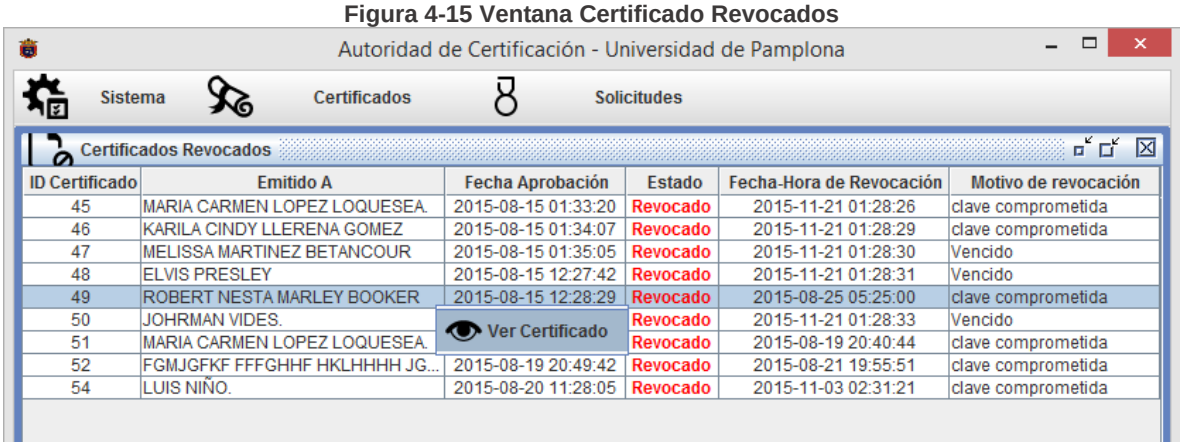
Figura 4-14 Ventana revocar certificados

Fuente: Autor

4.3.2.1.2 Certificados Revocados

Esta función hace parte del menú *Certificados* y lista todos los certificados que han sido revocados por la Autoridad de Certificación. Como se muestra en la Figura 4-15**¡Error! No se encuentra el origen de la referencia.**, el submenú solo tiene la opción «Ver Certificado».

Figura 4-15 Ventana Certificado Revocados

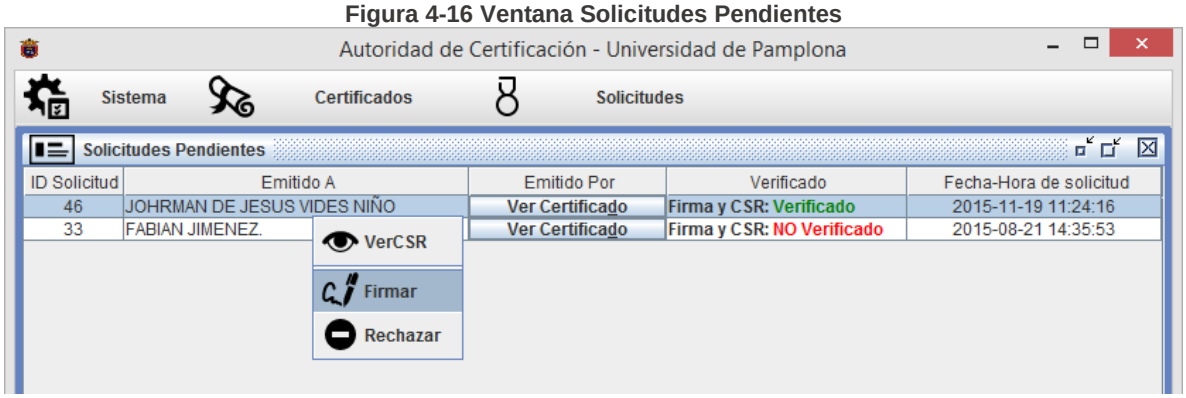


Fuente: Autor

4.3.2.1.3 Solicitudes Pendientes

Esta función hace parte del menú *Solicitudes* y lista en una tabla todas las solicitudes CSR que han sido enviadas por la Autoridad de Registro. Cada una de estas solicitudes puede ser aprobada o rechazada. Como se estipuló en la etapa de diseño, la Autoridad de Certificación solo debe aprobar las solicitudes CSR que hayan sido emitidas por los operarios autorizados de la Autoridad de Registro y que superen la comprobación de firma (ver Figura 4-16).

Figura 4-16 Ventana Solicitudes Pendientes



Fuente: Autor

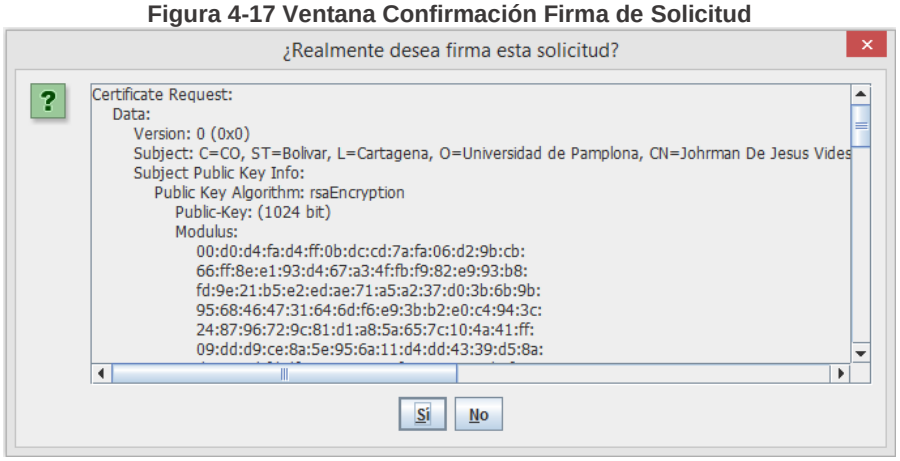
La interacción para decidir sobre la aprobación o rechazo de una solicitud CSR es por medio de un submenú, que permite elegir entre(ver Figura 4-16):

- **Ver CSR:** Muestra la CSR que ha sido enviada, para ello se ejecuta la función *MostrarCSR(),*explicada anteriormente.
- **Firmar:** Permite la aprobación de la CSR, generando y publicando en las bases de datos BD_REPO y BD_USU el certificado digital del usuario. El proceso que se sigue para firmar un certificado es el siguiente:

  I. Se muestra la solicitud CSR que está a punto de ser firmada y se pide confirmación (ver Figura 4-17):

```
int opcion=JOptionPane.showConfirmDialog(null,MostrarCSR(),"¿Realmente
desea firma esta solicitud?",JOptionPane.YES_NO_OPTION);
```

Figura 4-17 Ventana Confirmación Firma de Solicitud



Fuente: Autor

II. En caso de pulsar “SI”, se procede a firmar la solicitud CSR con la que se crea el certificado digital del usuario, el cual debe ser publicado en BD_REPO y BD_USU. El código para aprobar la solicitud es el siguiente:

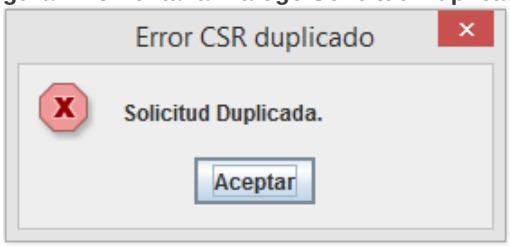
```
InputStream BD_inputStream = BD_REPO.bajarBD(); //se descarga toda la Base de datos de Repositorios para comprobar que el certificado no haya sido emitido con anterioridad
escribeArchivoBDcert(); //se escribe la base de datos descargada en un archivo temporal
FileOutputStream fos = new FileOutputStream("...\\newreq.pem"); //se descarga la solictud CSR
fos.write(certificadosFirmantes.get(tablaSolPen.getSelectedRow()).csr);
//se almacena en un archivo temporal la solictud CSR
```

Para firmar la solicitud se usa Openssl, el cual con la clave privada de la Autoridad de Certificación firma el certificado. Dependiendo del resultado del proceso de firma, se presenta en pantalla un ventana de diálogo mostrando el contenido del certificado firmado, que se sube a BD_USU y a BD_REPO.

```
BD_USU.SOLI_actualizarEstadoCSR(2,tablaSolPen.getModel().getValueAt(tablaSolPen.getSelectedRow(), 0).toString()); //actualiza el estado de la solitud CSR para indicar a las demás entidades que ha sido firmada
BD_REPO. subirBD(BD_Archivo); //se sube el certificado recién aprobado a la Bases de Datos de Repositorios donde el certificado será público
```

Después de todas las comprobaciones que se han hecho, los errores tienen lugar en caso que el usuario ya posea un certificado digital aún válido, es decir, que no ha sido revocado, entonces se muestra el mensaje de la Figura 4-18.

Figura 4-18 Ventana Dialogo Solicitud Duplicada



Fuente: Autor

- **Rechazar:** A través de esta opción se rechaza una solicitud CSR, por tanto no será generado el certificado digital. Solo cambia el estado de la solicitud CSR en BD_USU. Este cambio podrá ser revertido por la misma Autoridad de Certificación en las Solitudes Rechazadas. La siguiente línea actualiza el estado de la CSR.

```
BD_USU.CancelarPeticion(tablaSolPen.getModel().getValueAt(tablaSolPen.getSelectedRow(), 0).toString());
```

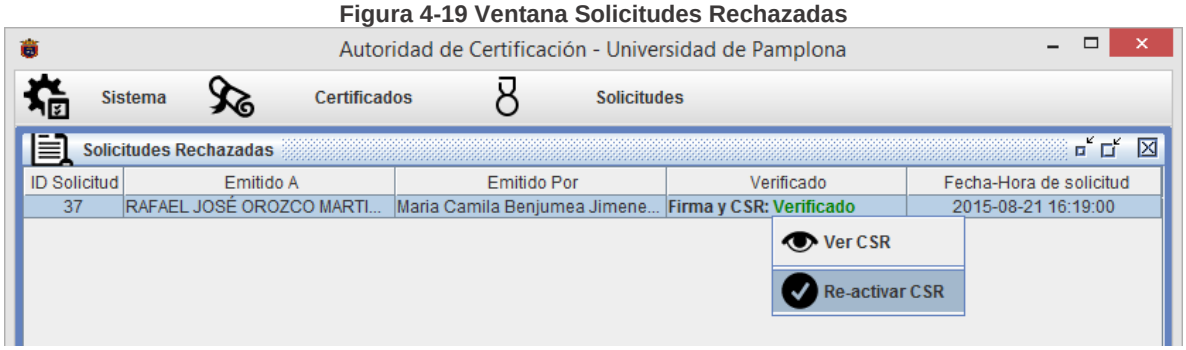
4.3.2.1.4 Solicitudes Rechazadas

Esta función hace parte del menú *Solicitudes* y permite reactivar las solicitudes CSR que fueron rechazadas, para que estén disponibles nuevamente. Un

submenú desplegado mediante el click derecho del mouse muestra las opciones «Ver CSR» y «Re-activar CSR», el cual ejecuta la sentencia:

```
BD_USU.SOLI_actualizarEstadoCSR(1,jTable1.getModel().getValueAt(jTable1.getSelectedRow(), 0).toString());
```

Figura 4-19 Ventana Solicitudes Rechazadas



Fuente: Autor

4.3.2.1.5 Piloto Automático

Uno de los objetivos de esta tesis es obtener un buen rendimiento de la aplicación, aún en caso de presentarse una elevada demanda de los servicios del Aplicativo de Registro de Usuarios, por ello, el Piloto Automático permite aprobar o rechazar las solicitudes CSR de manera automática, para evitar retrasos en la emisón de los certificados.

Esta función consulta, cada cierto tiempo, la tabla Solicitudes de BD_USU buscando nuevas solicitudes CSR. Para su ejecución se debe pulsar el botón tipo *toggle* que se encuentra en la parte inferior de la ventana principal, y pulsarlo de nuevo para detenerlo. La codificación que permite automatizar los procesos de emisión de certificados digital es:

```
if(jPilotoAutomatico.isSelected()){
    jPilotoAutomatico.setText("Desactivar");
    CTRL_FirmaAutomatica=true;
    (new FirmaAutomatica()).start();
}else{
    jPilotoAutomatico.setText("Piloto Automático");
    CTRL_FirmaAutomatica=false;
}
```

El elemento principal del piloto automático es la clase FirmaAutomatica() . Esta clase hereda los métodos tipo *Thread*, que permiten la ejecución de programas en paralelo, dando la potencialidad de ejecutarse junto con otros procesos. Una vez se invoca esta función, a través de un ciclo infinito controlado por la variable CTRL_FirmaAutomatica, se analiza si han llegado nuevas solicitudes CSR, y se verifica la firma del operario de cada una con el servidor OCSP, comprueba que el

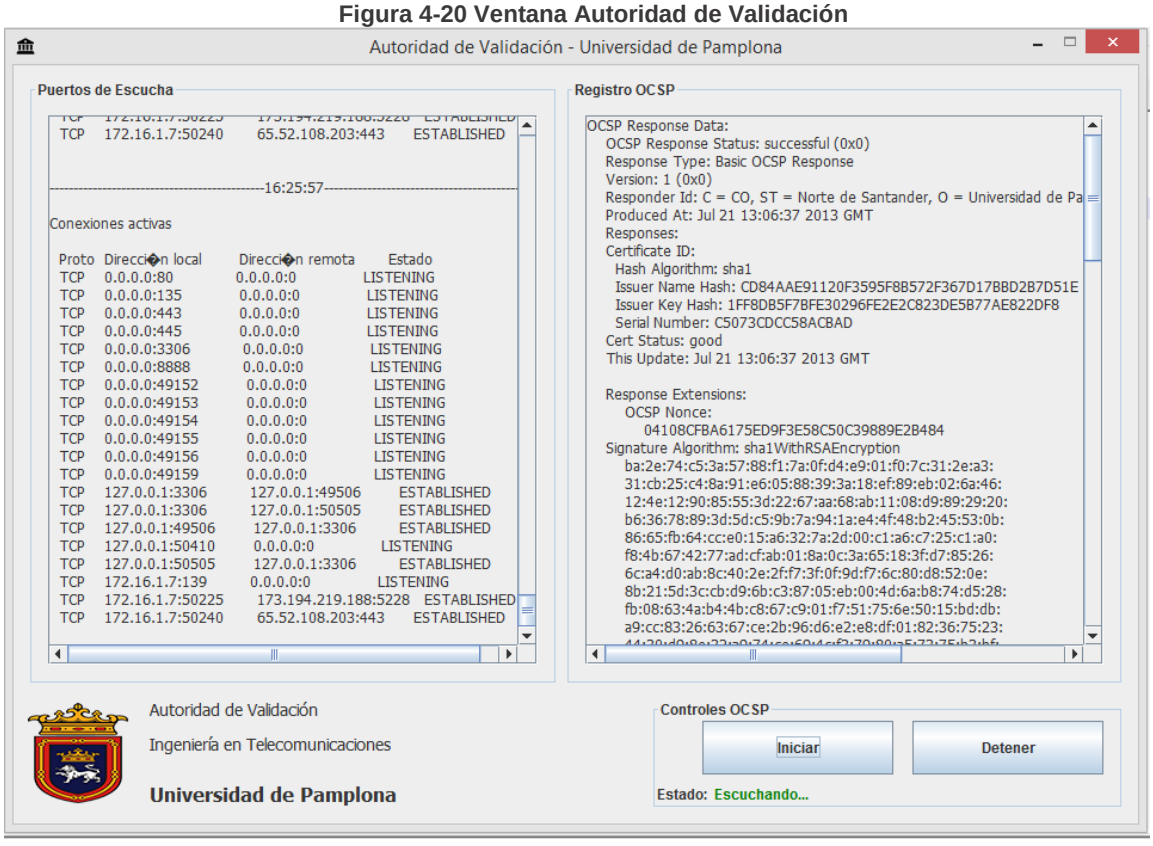
titular de la CSR no tenga ya un certificado válido, también con el apoyo de la Autoridad de Validación, y luego firma y publica el certificado; sin duda una gran ayuda para la persona encargada de manejar la Autoridad de Certificación.

4.3.3 Autoridad de Validación

El servidor OCSP emite el concepto sobre el estado de los certificados a los clientes. Estos clientes pueden ser usuarios que desean saber el estado de su certificado o servidores ante los cuales los usuarios presentan sus certificados para autenticarse. El servidor OCSP consulta la Base de Datos de Repositorios (BD_REPO) para establecer el estado de dichos certificados.

La interfaz gráfica está compuesta de dos jPanelText: uno muestra los puertos que escucha el servidor (parte izquierda de la pantalla), y el otro permite visualizar todas las solicitudes que se han hecho (parte derecha de la pantalla) (ver Figura 4-20). En la parte inferior están los botones Iniciar y Detener que permiten, como su nombre lo indica, colocar en funcionamiento la VA y pararla.

Figura 4-20 Ventana Autoridad de Validación



Fuente: Autor

Parte de la implementación del servidor OCSP se basa en funcionalidades que ofrece OpenSSL. Para que el servidor OCSP tenga en todo momento la

información de la última CRL emitida por la CA, el aplicativo obtiene el ID – o número de serie -del último certificado revocado de la Base de Datos de Repositorios, y luego lo compara con el ID almacenado en la variable idUltimo, si son diferentes procede a cerrar el proceso OpenSSL a través de la función `CortaProceso`(),. y descarga la nueva CRL a un fichero que almacena localmente. Esto proceso se realiza para lograr una mayor rapidez de respuesta, pues cuando se realizan consultas exhaustivas a las bases de datos se abren puertos y procesos en el servidor que se pueden salir de control y ser aprovechados por un inescrupuloso.

```
while(CTRL_OCSP){ //bucle controlado por una variable, se pone a falso cuando se quiera finalizar el servidor
    try {
        BD_repositorios.REPO_obtenerID_ultimo(); //descarga el último ID de BD_REPO
        if( (idUltimo != BD_repositorios.ID_Datos) && !(idUltimo >=
BD_repositorios.ID_Datos) ){ //compara el último ID con el último almacenado
          idUltimo=BD_repositorios.ID_Datos; //cuando se cumpla la condición se actualizará el
último ID
          DescargarBD(); // descarga la BD_REPO para que esté offline en el fichero index.txt
        }
        Thread.sleep(5000); //toma una pausa de 5 segundos para volver a realizar todas las acciones
    } catch (Exception ex) { }
}
```

Debido que el servidor OCSP debe entregar una respuesta sobre el estado de cierto certificado, un potencial atacante puede intersectar esa respuesta y modificarla a su conveniencia, situación que pone en riesgo la confianza en el Aplicativo de Registro de Usuarios. Por ello, el servidor OCSP debe firmar cada respuesta, para garantizar su integridad y autenticidad.

4.3.4 Bases de Datos

Antes de crear una Base de Datos, es necesario crear los roles o usuarios de esta base de datos. MariaDB, al igual que MySQL, maneja roles para controlar el acceso de los usuarios, lo que permite asignarles privilegios para acceder a cualquier objeto de la Base de Datos. El usuario por defecto es «*root*», sin contraseña configurada, y tiene acceso total a cualquier tipo de objeto de la Base de Datos, por ello, es conveniente añadir roles (usuarios o grupos de usuarios) y otorgar privilegios a dichos roles. Además, cada uno de estos usuarios debe presentar un certificado específico para autenticarse, lo que permite el cifrado de la comunicación entre las aplicaciones y las bases de datos.

A continuación se muestran los permisos asociados a cada usuario de BD_USU y BD_REPO sobre sus tablas y su correspondiente código SQL en MariaDB:

- **Bases de Datos de Usuarios (BD_USU):** Consta de los usuarios «AutoridadRegistro» y «AutoridadCertificación»

- El usuario «AutoridadRegistro» tiene los siguientes permisos sobre cada tabla:
  - Tablas *Usuarios* y *Solicitudes*: index (ver la estructura de la tabla), insert (insertar filas y/o datos), select (seleccionar datos), update (actualizar valores).
  - Tablas *CertificadosAprobados* y *Operadores*: select (seleccionar datos).
- El usuario «AutoridadCertificacion» tiene los siguientes permisos sobre cada tabla:
  - Tablas *Usuarios, Solicitudes y Operadores*: select (seleccionar datos).
  - Tabla *CertificadosAprobados*: index (ver la estructura de la tabla), insert (insertar filas y/o datos), select (seleccionar datos), update (actualizar valores).

- **Bases de Datos de Repositorios:** Su única tabla *Repositorios* podrá ser accedida por los siguientes usuarios:
  - El usuario «AutoridadCertificacion» tiene los permisos: index (ver la estructura de la tabla), insert (insertar filas y/o datos), select (seleccionar datos), update (actualizar valores).
  - El usuario «AutoridadValidacion» tiene solo el permiso de select(seleccionar datos).

Una vez creados los roles, se puede llevar a cabo la creación de las bases de datos y sus respectivas tablas.

Para la creación de las tablas se tuvieron en cuenta las siguientes consideraciones:

- La letra inicial de los nombres de las tablas es mayúscula, para facilitar la consulta, inserción y actualización de la información.
- El nombre de los campos de las tablas es específico, para evitar confusiones y facilitar el entendimiento de lo que contiene cada campo.
- Las llaves primarias y foráneas de una tabla empiezan con el término *Id,* para identificarlas fácilmente.
- La longitud de los campos se indicó con base en valores cotidianos, teniendo en cuenta los valores más pequeños y más grandes posibles.

4.3.4.1 Bases de Datos de Usuarios

Para crear la base de datos se utilizó el comando:

```
CREATE DATABASE 'BD_USU' DEFAULT CHARACTER SET utf8 COLLATE utf8_spanish_ci;
```

Para crear las diferentes tablas con sus campos se utilizaron los siguientes comandos:

- Tabla *CertificadosAprobados*

```
CREATE TABLE `BD_USU`.`CertificadosAprobados`(`idCertificado` INT NOT NULL
AUTO_INCREMENT, `Certificado` BLOB NOT NULL, `FechaCertificadoVenci` DATETIME
NOT NULL, `idCSR` INT NOT NULL, `VarCTRL` INT NOT NULL, `FechaRevocacion`
DATETIME NOT NULL, `MotivoRevocacion` VARCHAR(255) NULL, `idUsuario` INT NOT
NULL, PRIMARY KEY (`idCertificadosAprobados`), CONSTRAINT `idUsuario` FOREIGN
KEY (`idUsuario`) REFERENCES `BD_USU`.`Usuarios` (`idUsuario`) ON DELETE NO
ACTION ON UPDATE NO ACTION, CONSTRAINT `idCSR` FOREIGN KEY (`idCSR`)
REFERENCES `BD_USU`.`Solicitudes`(`idCSR`) ON DELETE NO ACTION ON UPDATE NO
ACTION) ENGINE = InnoDB;
```

- Tabla *Solicitudes*

```
CREATE TABLE `bd_usu`.`Solicitudes` ( `idCSR` INT NOT NULL
AUTO_INCREMENT,`idOperario` INT NOT NULL, `idUsuario` INT NOT NULL,
`CSR` BLOB NOT NULL, `FechaCSR` DATETIME NOT NULL,`VarCTRL` INT NOT
NULL, PRIMARY KEY (`idCSR`), UNIQUE INDEX `idCSR_UNIQUE` (`idCSR`
ASC), INDEX `yjkfj_idx` (`idOperario` ASC), INDEX `usuarioCSR_idx`
(`idUsuario` ASC), CONSTRAINT `operarioEmisor` FOREIGN KEY
(`idOperario`) REFERENCES `bd usu`.`Operadores` (`idOperario`) ON
DELETE NO ACTION ON UPDATE NO ACTION, CONSTRAINT `usuarioCSR` FOREIGN
KEY (`idUsuario`) REFERENCES `bd_usu`.`Usuarios` (`idUsuario`) ON
DELETE NO ACTION ON UPDATE NO ACTION) ENGINE = InnoDB;
```

- Tabla *Operadores*

```
CREATE TABLE `BD_USU`.`Operadores`(`idOperario` INT NOT NULL,
`NombresApellidos` VARCHAR(45) NOT NULL, `Cargo` VARCHAR(45) NOT NULL,
`Dependencia` VARCHAR(45) NULL, `Foto` BLOB NOT NULL, PRIMARY KEY
(`IdOperador`))ENGINE = InnoDB;
```

- Tabla *Usuarios*

```
CREATE TABLE `bd_usu`.`Usuarios` ( `idUsuario` INT NOT NULL,
`NombreApellido` VARCHAR(80) NOT NULL, `Email` VARCHAR(45) NOT NULL,
`Direccion` VARCHAR(45) NULL, `Telefono` INT(10) NULL, `PaisSede`
VARCHAR(45) NOT NULL, `DptoSede` VARCHAR(45) NOT NULL, `CiudadSede`
VARCHAR(45) NOT NULL, `Foto` BLOB NULL, `ProgramaAcademico`
VARCHAR(60) NOT NULL, PRIMARY KEY (`idUsuario`)) ENGINE = InnoDB;
```

4.3.4.2 Bases de Datos de Repositorios

Para crear la base de datos se utilizó el comando:

```
CREATE DATABASE 'BD_REPO' DEFAULT CHARACTER SET utf8 COLLATE utf8_spanish_ci;
```

Para crear la tabla *Repositorios* se utilizó el comando:

```
CREATE TABLE `bd_repo`.`Respositorios` ( `idCertificado` INT NOT NULL,
`Certificado` BLOB NOT NULL, `Estado` INT NOT NULL, `FechaVencimiento`
DATETIME NOT NULL, `FechaRevocacion` DATETIME NULL, `NombreTitular`
VARCHAR(80) NOT NULL, PRIMARY KEY (`idCertificado`)) ENGINE = InnoDB;
```

4.4 Implementación de la Aplicación Web

El Aplicativo Web consta de una serie de páginas Web que usan los servicios del Aplicativo de Registro de Usuarios para autenticar a los usuarios y poder acceder a la Base de Datos de Información Académica. A continuación se describe su implementación.

4.4.1 Servidor Web

El software utilizado como plataforma para ofrecer los servicios de comunicación cliente-servidor es Apache, proveído por XAMPP, junto con un intérprete de PHP que logra un verdadero complemento para la ejecución del Aplicativo Web.

Apache no requiere mucha configuración, ya viene listo para operar, pero se debe establecer una comunicación segura por medio del protocolo SSL, para que la información viaje cifrada entre el cliente y el servidor.

Para establecer una sesión SSL/TLS entre el servidor y las páginas Web del Aplicativo se debe configurar el servidor para que acepte conexión por el puerto 443, que es el puerto usado para este tipo de conexiones. La configuración se debe hacer en el archivo `httpd-ssl.conf`, pero antes se debe modificar el tipo de cifrado que usarán las páginas web, ya que es el protocolo TLSv1 el que está configurado por defecto y se considera inseguro en la actualidad.

```
SSLCipherSuite EECDH+AESGCM:EDH+AESGCM:AES128+EECDH:AES128+EDH
SSLProtocol +TLSv1.2
```

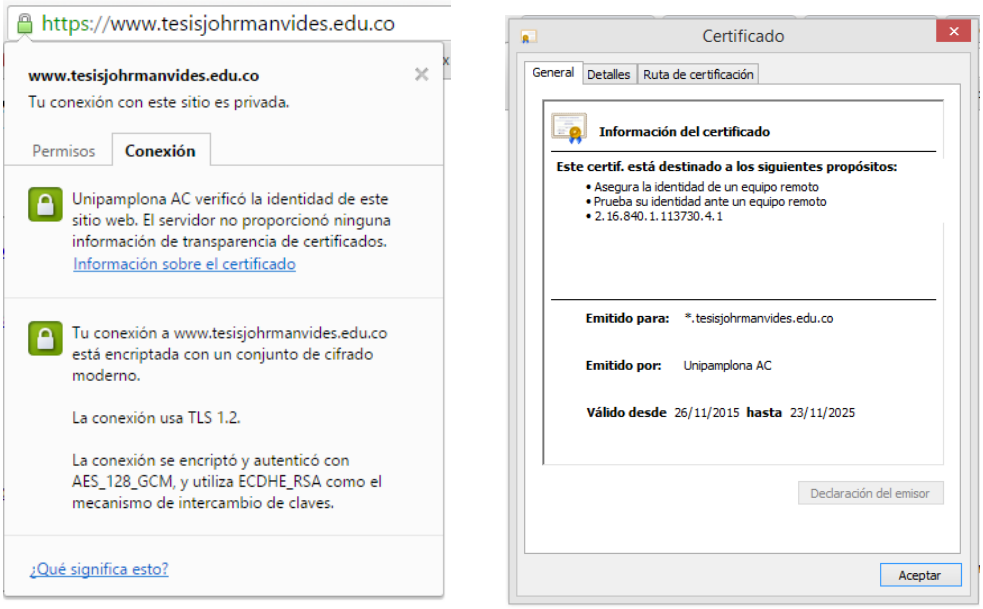
La directiva `SSLCipherSuite` especifica el conjunto de algoritmos de intercambio de claves y cifrado simétrico soportados: EECDH+AESGCM (*Elliptic-Curve Diffie-Hellman* combinado con *Advanced Encryption Standard-Galois Counter Mode*), EDH+AESGCM (*Ephemeral Diffie-Hellman* combinado con *Advanced Encryption Standard-Galois Counter Mode*), AES128+EECDH (*Advanced Encryption Standard* de 128 bits combinado con *Elliptic-Curve Diffie-Hellman*) y AES128+EDH

(*Advanced Encryption Standard* de 128 *bits* combinado con *Ephemeral Diffie-Hellman*). La directiva `SSLProtocol` define que se puede usar solo TLSv1.2, versión de TLS que soportan todos los navegadores. La opciones elegidas para la comunicación se pueden ver en el navegador, como se muestra en la Figura 4-21.

Para crear subdominios con soporte de SSL, se realiza la siguiente configuración:

```
<VirtualHost www.tesisjohrmanvides.edu.co:443>
       DocumentRoot "C:/xampp/htdocs/principal"
       ServerName www.tesisjohrmanvides.edu.co:443
       ServerAdmin admin@example.com
       ErrorLog "C:/xampp/apache/logs/error.log"
       TransferLog "C:/xampp/apache/logs/access.log"
       SSLEngine on
       SSLCertificateFile "conf/ssl.crt/server.crt"
       SSLCertificateKeyFile "conf/ssl.key/server.key"
       SSLCACertificateFile "conf/cassl.crt/cacert.crt"
       <FilesMatch "\.(cgi|shtml|phtml|php)$">
           SSLOptions +StdEnvVars
       </FilesMatch>
       <Directory "C:/xampp/apache/cgi-bin">
           SSLOptions +StdEnvVars
       </Directory>
       BrowserMatch "MSIE [2-5]" \
          nokeepalive ssl-unclean-shutdown \
          downgrade-1.0 force-response-1.0
       CustomLog "C:/xampp/apache/logs/ssl_request.log" \
           "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
</VirtualHost>
```

Figura 4-21 Visor de Certificados Navegador



Fuente: Autor

En la anterior configuración, `VirtualHost` indica el nombre de dominio seguido del puerto por donde se escucharán las solicitudes, `ServerAdmin` indica una dirección de correo electrónico para comunicarse con el administrador de la página (este campo puede ser omitido), `DocumentRoot` indica donde está ubicado el directorio raíz del subdominio, `ServerName` indica la dirección URL del subdominio. Las siguientes líneas de código se utilizan para decirle al servidor donde está su certificado y su clave privada para poder realizar la conexión SSL. , El `commonName` del certificado digital es `CN=*.tesisjohrmanvides.edu.co` lo que permite abarcar los dos subdominios (ver Figura 4-21) y evitar generar un certificado para cada uno. De esta forma se podrá ejecutar la autenticación SSL simple, donde el servidor presenta un certificado al navegador y se inicia un intercambio de mensajes para determinar si en realidad el servidor es quien dice ser y establecer una clave de sesión.

4.4.2 Base de Datos de Información Académica

La implementación de la Base de Datos de Información Académica (BD_IACA) y de sus tablas se realizó a través de MariaDB versión 10.0.19 utilizando su interfaz SQL. La sentencia que permite crear la base de datos es:

```
CREATE DATABASE `BD_IACA` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
```

Creada la base de datos, solo resta incluir las tablas que contendrán información específica de los estudiantes:

- Tabla *ProgramasAcademicos*

```
CREATE TABLE `BD_IACA`.`programasacademicos` ( `idPrograma` INT(11) NOT NULL, `nombrePrograma` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_spanish_ci' NOT NULL, `directorPrograma` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_spanish_ci' NULL DEFAULT NULL, `pensum` INT(4) NOT NULL, PRIMARY KEY (`idPrograma`)) ENGINE = InnoDB;
```

- Tabla *Docentes*

```
CREATE TABLE `BD_IACA`.`docentes` ( `idDocente` INT(11) NOT NULL, `nombreDocente` VARCHAR(80) CHARACTER SET 'utf8' COLLATE 'utf8_spanish_ci' NOT NULL, `tipoDocente` INT(1) NOT NULL, PRIMARY KEY (`idDocente`)) ENGINE = InnoDB;
```

- Tabla *Materias*

```
CREATE TABLE `BD_IACA`.`Materias` ( `idMateria` INT NOT NULL, `idDocente` INT NOT NULL, `codigoMateria` VARCHAR(10) NOT NULL, `programaAdscrito` VARCHAR(60) NOT NULL, PRIMARY KEY (`idMateria`), INDEX `idDocente_idx` (`idDocente` ASC), CONSTRAINT `idDocente`
```

```
FOREIGN KEY (`idDocente`) REFERENCES `BD_IACA`.`Docentes`
(`idDocente`) ON DELETE NO ACTION ON UPDATE NO ACTION) ENGINE =
InnoDB;
```

- Tabla *Estudiantes*

```
CREATE TABLE `BD_IACA`.`estudiantes` ( `idEstudiante` INT(11) NOT
NULL, `idPrograma` INT(11) NOT NULL, `nombreEstudiante` VARCHAR(90)
CHARACTER SET 'utf8' COLLATE 'utf8_spanish_ci' NOT NULL, `estado`
INT(1) NOT NULL, `semestre` INT(2) NOT NULL, PRIMARY KEY
(`idEstudiante`), INDEX `idPrograma idx` (`idPrograma` ASC),
CONSTRAINT `idPrograma` FOREIGN KEY (`idPrograma`) REFERENCES
`BD_IACA`.`programasacademicos` (`idPrograma`) ON DELETE NO ACTION ON
UPDATE NO ACTION) ENGINE = InnoDB;
```

- Tabla *Notas*

```
CREATE TABLE `BD_IACA`.`notas` ( `idEstudiante` INT(11) NOT NULL,
`idMateria` INT(11) NOT NULL, `nota1_15` DECIMAL(5,0) NOT NULL,
`nota1_20` DECIMAL(5,0) NOT NULL, `nota2_15` DECIMAL(5,0) NOT NULL,
`nota2_20` DECIMAL(5,0) NOT NULL, `nota3_10` DECIMAL(5,0) NOT NULL,
`nota3_20` DECIMAL(5,0) NOT NULL, `notaHabilitacion` DECIMAL(5,0) NULL
DEFAULT NULL, `notaDefenitiva` FLOAT(5,0) NOT NULL, PRIMARY KEY
(`idEstudiante`), INDEX `idEstudiante_idx` (`idMateria` ASC),
CONSTRAINT `idnte` FOREIGN KEY (`idMateria`) REFERENCES
`BD_IACA`.`materias` (`idMateria`) ON DELETE NO ACTION ON UPDATE NO
ACTION, CONSTRAINT `idest` FOREIGN KEY (`idEstudiante`) REFERENCES
`BD_IACA`.`estudiantes` (`idEstudiante`) ON DELETE NO ACTION ON UPDATE
NO ACTION) ENGINE = InnoDB;
```

- Tabla *Sesiones*

```
CREATE TABLE `BD_IACA`.`sesiones` ( `idEstudiante` INT(11) NOT NULL,
`CodAcceso` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_spanish_ci'
NOT NULL, PRIMARY KEY (`idEstudiante`), CONSTRAINT `idestSesion`
FOREIGN KEY (`idEstudiante`) REFERENCES `BD_IACA`.`estudiantes`
(`idEstudiante`)) ENGINE = InnoDB;
```

Se crea, además, un usuario con privilegios limitados, que usará un código de sesión, dado por Autentica-BD_IACA, para realizar las consultas. El código SQL usado para crear este usuario y atribuir sus privilegios en cada tabla es:

```
GRANT select ON bd_iaca.Notas TO 'usuarioConsultas@'%' IDENTIFIED BY
'Qv5F2rTm7oglrVy';
GRANT select ON bd_iaca.Estudiantes TO 'usuarioConsultas@'%';
GRANT select ON bd_iaca.ProgramaAcademico TO 'usuarioConsultas@'%';
GRANT select ON bd_iaca.Materias TO 'usuarioConsultas@'%';
GRANT select ON bd_iaca.Docentes TO 'usuarioConsultas@'%';
```

4.4.3 Aplicación Web

El lenguaje de programación escogido es PHP complementado con etiquetas HTML para la interfaz gráfica, JavaScript para lograr niveles de usabilidad más altos y Applets de Java que sirven de puente entre los lenguajes de programación Web y la tarjeta inteligente. Se usó Notepad++ como editor de los lenguajes de programación Web y NetBeans para crear un Applet de Java. Ambos editores son distribuidos con licencias GPL, lo que permite su uso sin restricciones en este proyecto.

4.4.3.1 Página de Inicio

Presenta al usuario una página similar a la de la universidad y tiene insertado en su código el Applet de Java que permite escoger el certificado e interactuar la tarjeta inteligente (*SeleccionaCertificado)*. En la Figura 4-22 se presenta la página de inicio implementada en HTML. El hipervínculo "*Transacciones*" inicia el Applet de Java y el hipervínculo "*Comprobar certificado*" le permite al usuario conocer el estado de su certificado. La inserción del Applet en esta página se hace así:

```
---------------------------------- index.php ----------------------------------
<td align="center">
  <object
    <param name="type" value="application/x-java-applet;version=1.4">
    <param name="code" value="appletSC.class">
    <param name="archive" value="appletSC.jar, lib/iaik.jar, lib/iaik_jce.jar">
    <param name="BotonLeeCert_ax1" value="Selecciona un certificado">
    <param name="varSitio" value="<?php echo
$_SERVER['HTTP_HOST'].":".$_SERVER['SERVER_PORT'];?>">
    <param name="varAutoCerti" value="<?php echo $_SERVER['SSL_SERVER_I_DN_CN'];
?>">
    <param name="varOrga" value="<?php echo $_SERVER['SSL_SERVER_I_DN_O']; ?>">
    <embed
     type="application/x-java-applet;version=1.4"
     pluginspage="http://java.sun.com/products/plugin/index.html#download"
     code="appletSC.class"
     archive="appletSC.jar, lib/iaik.jar, lib/iaik_jce.jar"
     width="150" height="25"
     BotonLeeCert_ax1="Selecciona un certificado"
     varSitio = "<?php echo $_SERVER['HTTP_HOST'].":".$_SERVER['SERVER_PORT'];?>"
     varAutoCerti = "<?php echo $_SERVER['SSL_SERVER_I_DN_CN']; ?>"
     varOrga = "<?php echo $_SERVER['SSL_SERVER_I_DN_O']; ?>" >
     <noembed> No se puede iniciar Applet porque el navegador no tiene Java Instalado
     </noembed>
     </embed>
  </object>
</td>
---------------------------------- index.php ----------------------------------
```

Figura 4-22 Página Web de Inicio

Fuente: Autor

El Applet se incluye en una celda centrada `<td align="center">`. La etiqueta `<object>` se usa para insertar un objeto con los atributos definidos en `<param>` como: tipo de aplicación que soporta el objeto (`type`), código compilado del objeto (`code`), librerías que usará el Applet (`archive`), texto en el hipervínculo (`BotonLeeCert_ax1`), nombre del sitio Web que ejecuta el Applet (`varSitio`), nombre de la autoridad de certificación que firma el certificado del servidor que ejecuta el Applet (`varAutoCerti`), organización a la que pertenece el servidor(`varOrga`). Los últimos tres atributos, se extraen del certificado digital que presenta el servidor, por medio de etiquetas reservadas de PHP.

4.4.3.2 SeleccionaCertificado

Una vez el usuario hace click en "`Transacciones`", se toman cuatro variables del servidor que solicita la autenticación con certificado digital, por ello, el primer paso

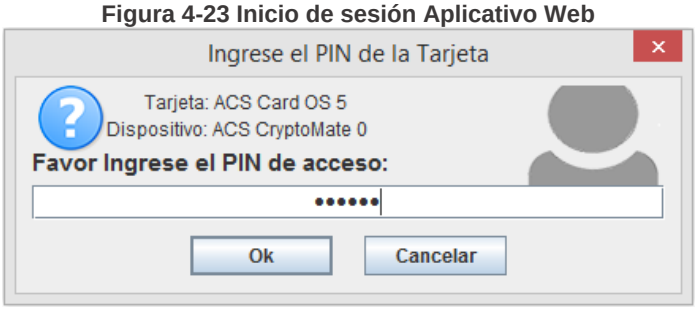
para crear el Applet es obtener estas variables para poder imprimirlas en su interfaz gráfica:

```
---------------------------------- Applet ------------------------------------
BotonLeeCert = new Button(this.getParameter("BotonLeeCert_ax1"));
String[] cabezera = new String[3];
cabezera[0]= this.getParameter("varSitio");
cabezera[1] = this.getParameter("varAutoCerti");
cabezera[2] = this.getParameter("varOrga");
---------------------------------- Applet ------------------------------------
```

Como se observa, estas variables son las mismas del script que ejecuta el Applet en la página de inicio. El vector `cabezera` almacena cada valor para su fácil manejo.

Luego, se busca el controlador de la tarjeta inteligente en el equipo del usuario, por tanto, debe tener instalados los *drivers* de la tarjeta `DireccionLibreria = buscarLiberiaTarjInt(){ … }`. En caso que se haya encontrado una driver instalado, se procede a inicializar el módulo PKCS11 `InicializarModulo(DireccionLibreria)` para poder seleccionar un slot y un token `slots = pkcs11Module.getSlotList(Module.SlotRequirement.TOKEN_PRESENT)`. Seleccionado un token, se inicia la interfaz gráfica para que el usuario ingrese el PIN, que se muestra en la siguiente Figura 4-23.

Figura 4-23 Inicio de sesión Aplicativo Web



Fuente: Autor

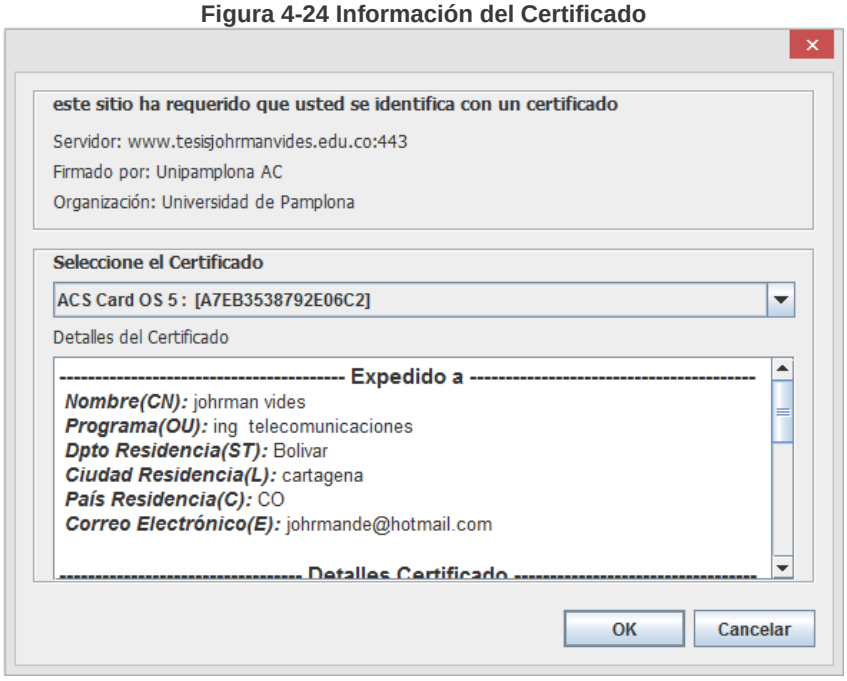
Por otro lado, antes de solicitar el PIN el applet contacta a la BD_USU para obtener cuantos intentos le quedan al usuario, si estos son menores que 3, se entenderá que la tarjeta está bloqueda por exceso de intentos de sesión inválidos, en este caso deberá dirigirse a la Autoridad de Registro donde deberá cambiar PIN para setear los números de intentos después de haberse identificado el usuario ante el operario.

```
---------------------------------- Applet ------------------------------------
```

```
session.login(Session.UserType.USER, pinCode.toCharArray()); //intento de logueo
AuxValidadorContrase=false; AuxControlLogin=false;
AuxValidadorContrase=true; AuxControlLogin=true;} //variables de control
---------------------------------- Applet -------------------------------------
```

En caso que el PIN sea ingresado correctamente, se procede a listar todos los certificados que están en la tarjeta inteligente, por medio de la función `leerCertificados()`, la cual devolverá un *array* con la información de cada certificado digital encontrado para que se pueda seleccionar un certificado. La Figura 4-24 muestra la interfaz gráfica desarrollada para elegir el certificado.

Figura 4-24 Información del Certificado



Fuente: Autor

4.4.3.3 Autentica-BD_IACA

Una vez el usuario ha escogido el certificado con el cual desea autenticarse, el Applet lo envía a Autentica-BD_IACA.php por medio de la función

```
---------------------------------- Applet -------------------------------------
String Reto = intercambiosConPHP(CerticadoPEM,NULL,"10101010"); //enviando certificado.
---------------------------------- Applet -------------------------------------
```

La función `intercambiosConPHP` recibe por parámetros el certificado a enviar, el reto, y una variable de control. Se diseñó de esta forma para que pueda ser

reutilizada en otro proceso y evitar excesos de programación que llenen la memoria RAM.

Mientras tanto, el Applet espera respuesta a través del comando:

```
BufferedReader rd = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
```

Autentica-BD_IACA.php recibe los valores enviados por el Applet, pero antes debe comprobar que la conexión esté cifrada, y que las tres variables existan en la solicitud. Si alguno de estos requerimientos es falso, *Autentica-BD_IACA.php* entenderá que se está intentando acceder ilegalmente al sistema. Esta comprobación se realiza de la siguiente forma(ver Figura 4-25):

```
--------------------------- Autentica-BD_IACA.php -----------------------------
<?php
if( (isset($_POST['VariableDecision'])) && //comprueba que existe esta variable
    (isset($_POST['Certificado'])) && //comprueba que existe esta variable
    (isset($_POST['Reto'])) && //comprueba que existe esta variable
    (isset($_SERVER['SSL_SERVER_I_DN_CN'])) ){//comprueba que la conexión sea cifrada
       $VariableDecision=$_POST['VariableDecision']; //guarda la variable post
       if($VariableDecision==10101010){ //se decide qué proceso iniciar
         echo (AutenticaUNO());
       } elseif ($VariableDecision==21010101){
         echo( AutenticaDOS());
       } else {echo "<center><h2> intento de acceso illegal </center></h2>";}
}
else {echo "<center><h2> intento de acceso illegal </center></h2>";}
?>
--------------------------- Autentica-BD_IACA.php -----------------------------
```

Figura 4-25 Alerta: *Autentica-BD_IACA.php*

Fuente: Autor

Además de proteger contra posibles intrusos, el código anterior permite decidir qué proceso se va a ejecutar. En este caso, ha sido enviada la `VariableDecision` con el valor de `10101010`, lo que iniciará la función de PHP `AutenticaUNO()`, que determina el estado del certificado digital presentado, utilizando los servicios de la Autoridad de Validación. Pero a OpenSSL, para enviar una petición OCSP se le debe incluir el certificado digital de la Autoridad de Certificación, el certificado digital a consultar, y la dirección URL donde se

encuentra el servidor OCSP. El certificado de la Autoridad de Certificación es público y se guarda un copia de este en un lugar que pueda ser accedido por el servidor Web Apache; el certificado digital a consultar se tiene en la variable `Certificado` que ha sido enviada por el Applet, pero OpenSSL necesita un archivo no una variable, por ello se debe escribir este certificado en un fichero e indicarle a OpenSSL su ubicación; la dirección URL del servidor OCSP ya está definida desde el principio. Por tanto, las sentencias que hacen parte de la función `AutenticaUNO()`.

Como se puede observar, esta función se apoya en otras funciones para llevar a cabo su objetivo: `cifrar(...)` y `compruebaFirmaOCSP(...)`; la primera genera el reto cifrado a partir de un texto aleatorio y la clave pública contenida en el certificado del usuario; la segunda verifica la firma digital que contiene la respuesta del servidor OCSP.

La función `cifrar(){...}` genera un cadena aleatoria de 32 caracteres a partir de un conjunto de números, letras minúsculas, letras mayúsculas y caracteres especiales. La seguridad de la función aleatoria, es que no tiene un método fijo para elegir cada carácter, de hecho su manera de funcionar es leyendo sectores vacíos no utilizados de la memoria RAM, luego define un puntero por cada caracter y de acuerdo a la distribución de dichos punteros selecciona el caracter. Este reto es almacenado en una variable temporal por 5 segundos. Después de haber transcurrido ese tiempo, la variable es destruida.

El reto cifrado con la clave pública genera una cadena con algunos caracteres no imprimibles, por ello, se convierte esta cadena a través de la función `convertirHexa(){...}` en su correspondiente valor hexadecimal, para que durante el envió del reto cifrado no se pierdan caracteres.

```
-------------------------- Autentica-BD_IACA.php ------------------------------
<?php
function cifrar($cert){ //cabecera de la función, recibe el certificado
  $pub_key = openssl_pkey_get_public($cert); //extrae la clave pública del certificado y la prepara
   para usarla
  $keyData = openssl_pkey_get_details($pub_key); // devuelve una matriz con los detalles de la
  clave
  $retoClaro =substr(str_shuffle("0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJ
  KLMNOPQRSTUVWXYZ/*-+.!#$%&/?\@"), 0, 32); //devuelve una cadena de 32 caracteres a partir de
  una cadena dada.
  apc_store('retoClaroAux',$retoClaro,5); //almacena por 5 segundos una variable de nombre
  retoClaroAux con el contenido de la variable $retoClaro
  openssl_public_encrypt($retoClaro, $retoCifrado, $keyData['key']); //cifra
  $retoClaro con la clave pública y la guarda en la variable $retoCifrado
  retrun convertirHexa($retoCifrado); //antes de devolver el retoCifrado se convierte a valores
  hexadecimales, pues si se envía sin convertir se pierden caracteres en el camino.
}
?>
```

---------- Autentica-BD_IACA.php ----------

Por otro lado, la función `compruebaFirmaOCSP(...)`, parte del certificado digital del servidor OCSP, con la finalidad de extraer su clave pública y almacenarla en la variable $CPUB. Luego, toma la respuesta del servidor OCSP en la variable `$comprobacion` y obtiene la información en claro que el servidor firmó (el certificado junto con el estado), almacenándola en la variable `$comprobacion2`. En `$comprobacion3` se guarda el resumen cifrado por el servidor OCSP y por último, se invoca a OpenSSL para que verifique el documento (`$comprobacion2` concatenado con `$comprobacion3`). OpenSSL genera una salida en la variable `$Resul`, el cual es un *array* que en la segunda posición almacena el resultado de la verificación. Solo cuando contenga el `"Verified OK"` se entenderá que es válida la firma y devuelve un `true`.

---------- Autentica-BD_IACA.php ----------

```
<?php
function compruebaFirmaOCSP($comprobacion){
    $cert_ocsp= 'C:\xampp\htdocs\tesis\prueba\OCSPcertificado.pem';
    $ClavePub = openssl_pkey_get_public($cert_ocsp);
    $CPUB = openssl_pkey_get_details($ClavePub);
    $comprobacion2 = preg_split('/[\r\n]/', $comprobacion[4]);
    $comprobacion3 = preg_split('/: /', $comprobacion[3]);
    $Resul=shell_exec('openssl dgst -sha1 -verify '.$CPUB['key'].' -signature
'.$comprobacion2+'\n'+$comprobacion3);
    if( strnatcasecmp($Resul[1],'Verified OK') == 0){
        return true;
    }
    return false;
}
?>
```

---------- Autentica-BD_IACA.php ----------

Ahora, el Applet analiza la respuesta de *Autentica-BD_IACA.php* a través de:

---------- Autentica-BD_IACA.php ----------

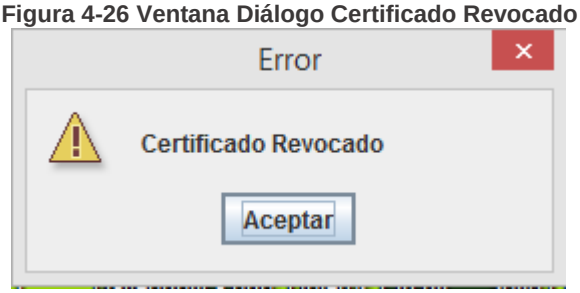
```
// Obteniedo respuesta de Autentica-BD_IACA.php

BufferedReader rd = new BufferedReader(new
InputStreamReader(conn.getInputStream())); //espera la respuesta, hasta que conn tenga algún
valor
String Lineas;
while ((line = rd.readLine()) != null) { //ciclo para leer línea por línea la respuesta
    if(Lineas.length()==512){ return Lineas } //en caso de tener la respuesta una longitud de
    512 caracteres, se considera un reto cifrado
    else if(Lineas.equalsIgnoreCase("6")){...}//usuario autenticado
}
wr.close();
rd.close()
return null;
```

---------- Autentica-BD_IACA.php ----------

Se pueden obtener las siguientes respuestas:

- Si el Applet recibe un 2, el certificado presentado está revocado (ver Figura 4-26).

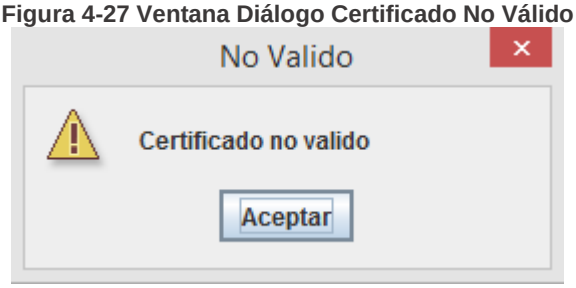
Figura 4-26 Ventana Diálogo Certificado Revocado



Fuente: Autor

- Si el Applet recibe un 3, el certificado presentado no es válido (ver Figura 4-27).

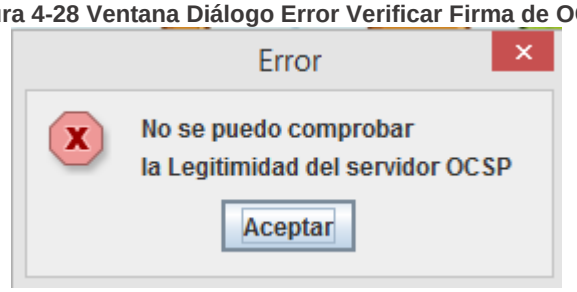
Figura 4-27 Ventana Diálogo Certificado No Válido



Fuente: Autor

- Si el Applet recibe un 4, no se pudo comprobar la legitimidad de la firma digital del servidor OCSP (ver Figura 4-28).

Figura 4-28 Ventana Diálogo Error Verificar Firma de OCSP



Fuente: Autor

- Si el Applet recibe un 5, el servidor OCSP no resolvió la consulta o no se pudo contactar (ver Figura 4-29).

Figura 4-29 Ventana Diálogo OCSP No Resolvió Consulta

Fuente: Autor

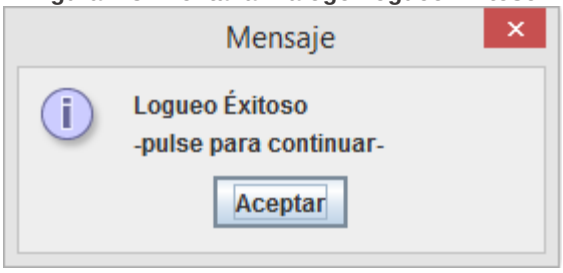
- Si el Applet recibe “intento de acceso ilegal”, la conexión cifrada SSL entre el servidor y la página Web no está habilita (ver Figura 4-30).

Figura 4-30 Ventana Diálogo Conexión No Segura

Fuente: Autor

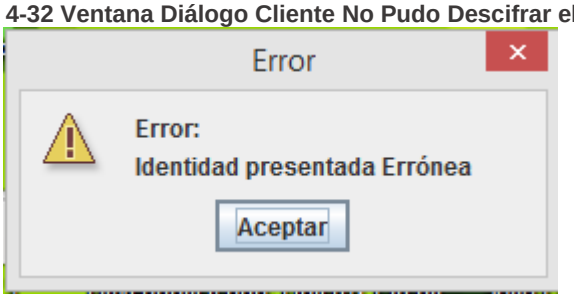
- Si el Applet recibe por respuesta un 6, 7 u 8, el usuario fue autenticado y autorizado, el usuario no pudo descifrar el reto, o el usuario fue autenticado pero no fue autorizado, respectivamente (ver Figura 4-31, Figura 4-32, Figura 4-33).

Figura 4-31 Ventana Diálogo Logueo Exitoso



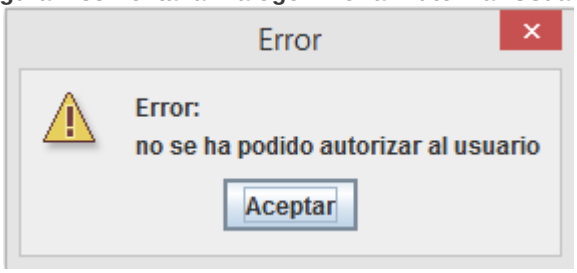
Fuente: Autor

Figura 4-32 Ventana Diálogo Cliente No Pudo Descifrar el Reto



Fuente: Autor

Figura 4-33 Ventana Diálogo Error al Autorizar Usuario



Fuente: Autor

- Si el Applet recibe una cadena de 512 caracteres, se ha recibido un reto cifrado, por lo que continúa con el proceso de autenticación ejecutando la función `DescifrarRetoConClavePrivada(...) {...}`, que descifrar el reto usando la clave privada almacenada en la tarjeta inteligente. La función que recibe el reto en valor hexadecimal debe apoyarse en otra función para convertirlo al formato original, y descifrarlo usando el *token* de la clave privada. Una vez esté descifrado se devuelve a *Autentica-BD_IACA.php* y este compara el reto descifrado con el reto en claro, que tiene almacenado en una variable temporal. El envío del reto descifrado se realiza así:

```
---------------------------------- Applet ----------------------------------
intercambiosConPHP(CerticadoPEM,Reto,"21010101");
---------------------------------- Applet ----------------------------------
```

  Como se observa se llama de nuevo la función `intercambiosConPHP()` pero en esta oportunidad se le envía el certificado, el reto descifrado y una variable de decisión con valor de `21010101`, que indica a *Autentica-BD_IACA.php* que debe comparar el valor almacenado en su variable temporal con el reto descifrado que se le envía.

Autentica-BD_IACA.php debe dirigirse a la función `AutenticaDOS(){...}` donde se comprueba si el reto descifrado, que ha enviado el Applet por medio de POST en la variable `Reto`, es igual a la variable temporal `retoClaroAux`. Si es así, se autoriza al usuario su acceso a la Base de Datos de Información Académica,

devolviéndole un código de acceso. Este código no se devuelve directamente al Applet, al Applet solo se devuelve una respuesta con código 6, como se aprecia en el siguiente código:

```
--------------------------- Autentica-BD_IACA.php -----------------------------
<?php
function AutenticaDOS(){
    if($_POST['Reto']==apc_fetch('retoClaroAux')){
        return Autorizar();
    } else {return "6";}
}
?>
--------------------------- Autentica-BD_IACA.php -----------------------------
```

El código de acceso nombrado anteriormente se escribe en una *cookie*, que será usada por las páginas que el Applet invoque. La función `Autorizar()` genera dicho código de acceso, y se comunica con BD_IACA para informarle que el usuario identificado con id (número de identificación del estudiante) y un código de acceso va a acceder a ella.

Se debe evitar que algún inescrupuloso robe la información de esta *cookie*, por ello, no se almacena la información en claro sino que se almacena su valor de *hash* SHA512 (`$hash_sesion`). El resumen debe contener 5 bits por caracter (`$hash_bit_caracter`), solo se puede acceder a la *cookie* usando la sesión, por tanto, JavaScript no podrá acceder a ella por URL (`$sesion_no_javascript`); solo las conexiones cifradas con SSL podrán acceder a la *cookie* (`$solo_SSL`), solo conexiones http podrán acceder a la *cookie* (`$solo_http`), y la *cookie* debe ser visible para cierto subdominio (`$dominio`).

La generación del código de acceso es por medio de una función que escoge dígitos aleatorios en base a una cadena `$codAcceso =substr(str_shuffle("..."))`. Este código se hace llegar a la Base de Datos de Información Académica después de habérsele calculado el resumen SHA256 (`$codAccesoHASH`), este hash o resumen permitirá relacionar el código de acceso con el número de identificación del estudiante, y es enviado del Applet a *Autentica-BD_IACA.php* por medio de la variable POST `$_POST['Certificado']`. Luego de haber insertado el hash del código de acceso junto con el ID del estudiante en la tabla *Sesiones* de la base de datos, se escribe el valor del código de acceso en la *cookie* `($_SESSION['codAcceso']=$codAcceso)`. Finalmente, si no se genera un error, se devuelve un código al Applet para continuar con el proceso.

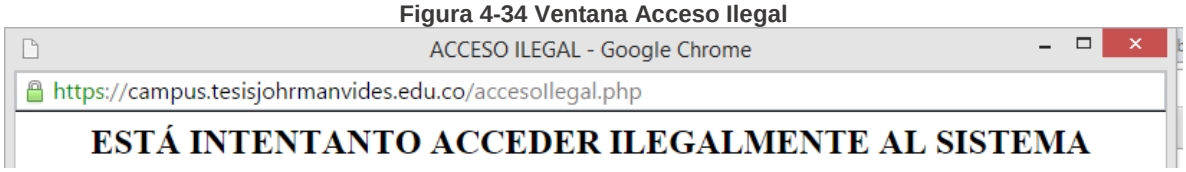
Cuando el Applet recibe el código 6, procede a abrir la página *académico.php,* donde el usuario podrá visualizar su información académica.

Autentica-BD_IACA.php ejecuta una serie de sentencias propias de PHP para acceder al código de acceso y verificar si este es realmente válido:

```
------------------------------ académico.php ---------------------------------
<?php
session_start();
if(!isset($_SESSION['codAcceso'])){
    header("location: accesoIlegal.php");
}
else{
    $conexion = conectaBaseDatos('usuarioConsultas');
    $codAccesoHash=hash('sha256',$_SESSION['codAcceso']);
    $consulta = "SELECT idEstudiante FROM 'bd_iaca'.'sesiones' WHERE
    CodAcceso=$codAccesoHash";
    $sentencia = $conexion->prepare($consulta);
?>
------------------------------ académico.php ---------------------------------
```

Inicialmente se comprueba que exista una *cookie* llamada `'codAcceso'`. Si no existe, se está tratando de acceder ilegalmente al sistema y se redirecciona a la ventana mostrada en la Figura 4-34. También se mostrará el mensaje de acceso ilegal, si no se halla el código de acceso en la tabla *Sesiones* de la base de datos de información académica.

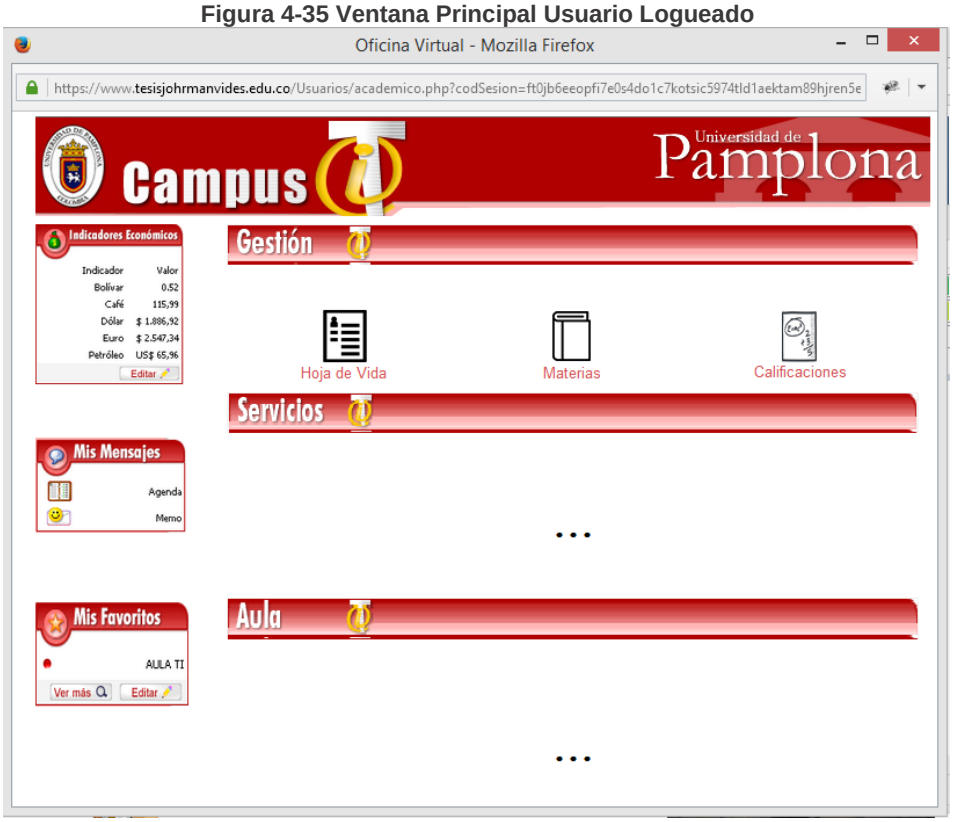
Figura 4-34 Ventana Acceso Ilegal



Fuente: Autor

En caso de superar ambas comprobaciones, se abre la interfaz gráfica mostrada en la Figura 4-35.

Figura 4-35 Ventana Principal Usuario Logueado

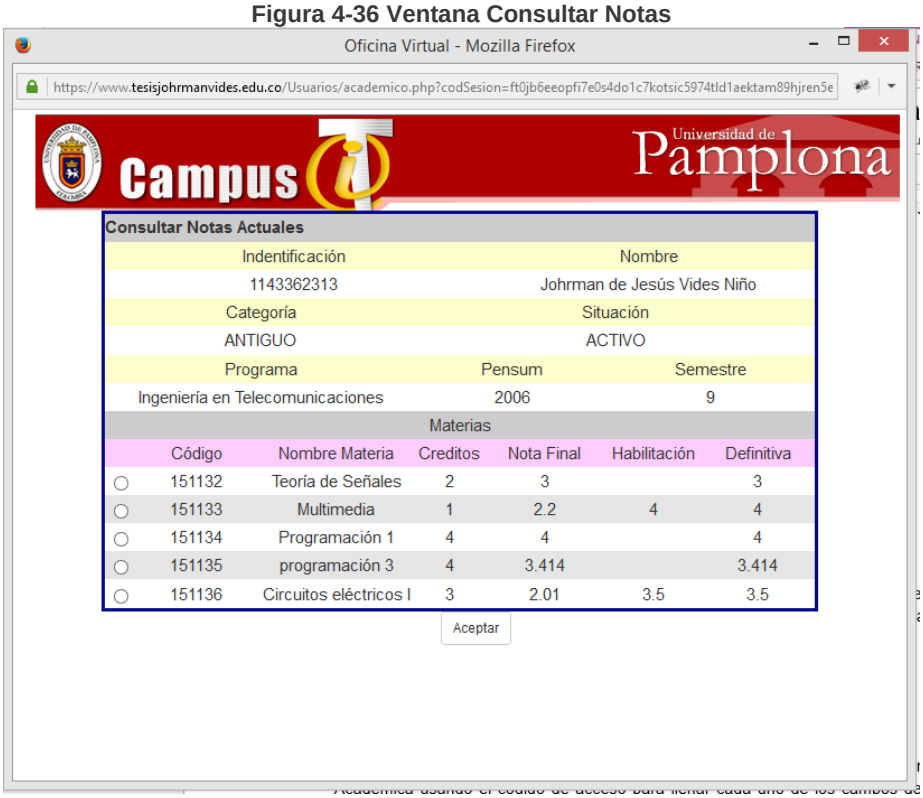


Fuente: Autor

Esta página se organiza por *frames*, los cuales permiten abrir varias páginas en una misma ventana, manteniendo la privacidad de la dirección URL. Cada una de las páginas extrae los datos de los usuarios de la Base de Datos de Información Académica, usando en cada consulta el código de acceso generado, mostrando el nombre y el programa del estudiante en la parte superior de la página.

Cuando el usuario desee conocer sus notas solo debe hacer click en la opción *Calificaciones* para abrir la página mostrada en la Figura 4-36.

Figura 4-36 Ventana Consultar Notas



Fuente: Autor

Para gestionar las cookies y los códigos de acceso en la Base de Datos de Información Académica se usa JavaScript. Por ejemplo, la sesión debe cerrarse cuanto el usuario cierre la ventana donde se muestra su información académica. Cerrar una sesión es equivalente a eliminar la *cookie* y borrar el código de acceso en la tabla *Sesiones* de BD_IACA, para ello en la página *académico.php* se implementa un *script* en JavaScript que detecte cuando se ha cerrado la ventana:

```
-------------------------------- académico.php ----------------------------------
<html>
    <script type="text/javascript">
      function destruirSesion {
        <?php
       $codAccesoHash==hash('sha256',$_SESSION['codAcceso'];
       $consulta = "DELETE idEstudiante, CodAcceso FROM 'bd_iaca'.'sesiones' WHERE
CodAcceso=$codAccesoHash";
       $sentencia = $conexion->prepare($consulta);
       $sentencia->execute();
       $resultado = $sentencia->fetchAll();
       $sentencia->closeCursor();
       session_destroy();
       SessionHandler::gc;
        ?>
       }
    </script>
</html>
-------------------------------- académico.php ----------------------------------
```

Como se aprecia en el `<body>`, el evento `onunload` se activa cuando el usuario abandona la página. Este evento se encarga de llamar al *script* que ejecuta sentencias de PHP para borrar toda la fila de la tabla *Sesiones* de BD_IACA que contenga el código de acceso que se quiere borrar, además de borrar la sesión (`session_destroy()`) eliminando la *cookie* que contiene el código de acceso, y por último, se eliminan todas las sesiones antiguas, si las hay (`SessionHandler::gc`).

Para evitar que los usuarios dejen la sesiones abiertas, se crea un *script* capaz de detectar la inactividad por parte del usuario:

```
--------------------------------- académico.php -----------------------------------
<script type="text/javascript">
  var t;
  window.onload= reiniciarTiempo();
  document.onkeypress= reiniciarTiempo();
  document.onmousemove= reiniciarTiempo();
  function tiempoFuera(){
       destruirSesion();
       alert("El sistema se cierra por 5 minutos de inactividad.");
       window.close();
       location.href='https://www.tesisjohrmanvides.edu.co';
  }
  function reiniciarTiempo(){
       window.clearTimeout(t);
       t=setTimeout(tiempoFuera,300000); //5 minutos de inactividad, tiempo en ms
  }
</script>
--------------------------------- académico.php -----------------------------------
```

Este *script* se ejecuta automáticamente después de la apertura de *académico.php* por el evento `window.onload` y llama a la función `reiniciarTiempo()` para establecer por el tiempo de cerrado (`t`). Usando eventos de Javascript, cuando se detecte un movimiento del mouse (`document.onmousemove`) y cuando se presione una tecla en la página (`document.onkeypress`) el tiempo de cerrado se reinicia. La función `setTimeout()` llama a la función `tiempoFuera()` después de `300000 ms`, la cual ejecuta `destruirSesion()` para eliminar el código de acceso de BD_IACA y la *cookie*, mientras que emite un mensaje (`alert()`) avisando al usuario que su sesión se ha cerrado. Por último, se cierra la ventana actual y se abre la página principal (`location.href`).

Capítulo 5

PRUEBAS REALIZADAS

5.1 Pruebas de la Aplicación de Registro de Usuarios

Esta aplicación debe garantizar la seguridad de los datos que se manejan en ella, ser amigable con el usuario y tener un buen desempeño.

5.1.1 Prueba de Amigabilidad

En esta sección se evalúa la funcionalidad y amigabilidad de las Autoridades de Registro, de Certificación y de Validación. Para ello, se midió el tiempo que tardaron 6 personas en realizar las diferentes funciones de dichas autoridades, tras expllicarles brevemente el funcionamiento de la aplicación. Estas personas se dividieron en dos grupos: el Grupo 1, conformado por personas que usan frecuentemente un computador, y el Grupo 2, conformado por personas que usan con poca frecuencia un computador (< 2 horas/semana). Como se observa en la Tabla 5-1, realizar las funciones de la Autoridad de Registro les tomó mayor tiempo a los usuarios, en promedio 1,19 minutos necesitaron las personas del Grupo 1 para elaborar una solicitud CSR (función que más tiempo tomó), mientras que las personas del Grupo2 en promedio tardaron 4,03 minutos para realizar la misma función. Los tiempos de las pruebas realizadas en las diferentes autoridades, se muestran en la Tabla 5-1.

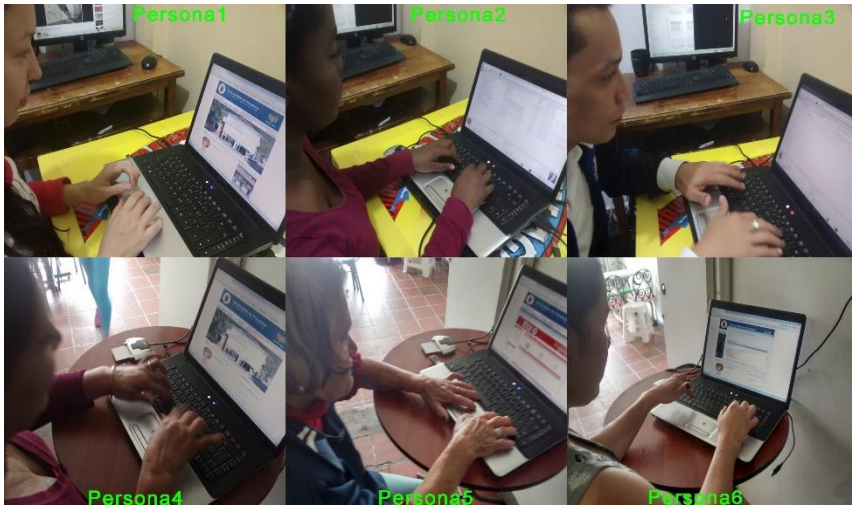
Tabla 5-1 Tiempos de Operaciones Usuario

	Función (en minutos) / **Persona**	Autoridad de Registro					Autoridad de Certificación					Autoridad de Validación	
		Iniciar Sesión	Generar una solicitud	Cancelar una Solicitu	Importar certificado	Ver las Solicitudes rechazados	Revocar un certificado	Ver certificado revocado	Firmar una solicitud CSR	Cancelar una solicitud CSR	Reactivar una CSR cancelada	Activar	Desactivar
GRUPO 1	Persona1	0,41	1,32	0,13	0,21	0,12	0,16	0,1	0,18	0,09	0,14	0,06	0,05
	Persona2	0,44	1,24	0,14	0,17	0,15	0,14	0,13	0,23	0,12	0,18	0,03	0,03
	Persona3	0,3	1,02	0,1	0,18	0,3	0,23	0,4	0,13	0,07	0,15	0,02	0,02
	Promedio	0,38	1,19	0,12	0,19	0,19	0,18	0,21	0,18	0,09	0,16	0,04	0,03
GRUPO 2	Persona4	1,1	4	0,4	1,1	0,71	0,41	0,32	0,31	0,2	0,52	0,17	0,170
	Persona5	0,8	3,7	0,31	0,56	0,42	0,34	0,34	0,33	0,19	0,41	0,24	0,22
	Persona6	0,94	4,4	0,5	1,14	0,58	0,51	0,43	0,42	0,21	0,46	0,22	0,21
	Promedio	0,95	4,03	0,40	0,93	0,57	0,42	0,36	0,35	0,20	0,46	0,21	0,20

Fuente: Autor

Como se observa en la Tabla 5-1, existe bastante diferencia entre los tiempos tomados por las personas del Grupo 2 y los de las personas del Grupo 1. Sin embargo, a medida que fueron avanzado en la prueba, hubo una disminución del tiempo en las funciones cuyas operaciones son similares, pues el usuario va reconociendo el sistema y localizando los menús necesarios. De todas maneras, los tiempos obtenidos no son excesivamente grandes. En la Figura 5-1, se muestran los participantes en esta prueba.

Figura 5-1 Participantes en Prueba de Amigabilidad



Fuente: Autor

5.1.2 Prueba de Seguridad

5.1.2.1 Borrado Seguro de Claves

Como ya se ha mencionado en capítulos anteriores, las claves de los usuarios se generan dentro de la tarjeta inteligente y esta tarjeta está configurada para que la clave privada no pueda ser extraída o copiada. Cuando se necesita realizar alguna operación criptográfica con la clave privada, dicha operación se lleva a cabo en el interior de la tarjeta inteligente, de manera que la clave privada no se ve expuesta a ser capturada por un atacante en ningún momento. Además, gracias a que el par de claves de cada usuario se genera dentro de la tarjeta inteligente, no es necesario borrar dicho par de claves del computador.

La claves que se generaron directamente en el computador fueron las claves de las Autoridades, como se especificó en la sección 4.2, pero estas fueron borradas de manera segura a través de una función, explicada en la misma sección de este documento, después de importarlas en una tarjeta inteligente.

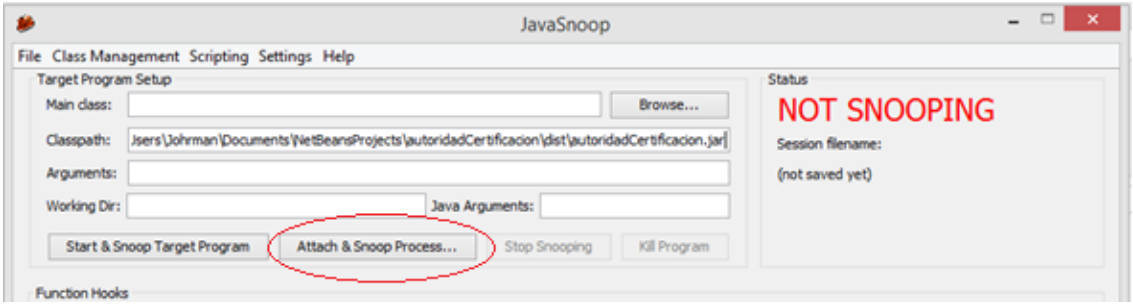
5.1.2.2 Seguridad de Aplicación JAVA

La aplicación está protegida por la función `IAIK.secure-block-device();` que no permite que los métodos a ejecutar sean interceptados por la máquina virtual de Java. Este *bug* es muy peligroso porque se puede abrir y ejecutar un depurador remotamente y cambiar variables del código, pudiéndose agregar usuarios a la base de datos sin permiso de ningún operario.

Para pobar la seguridad del aplicativo se usó JAVASNOOP, una herramienta que permite a los programadores romper la seguridad de las aplicaciones Java.

JAVASNOOP, sin acceder al código fuente original, pone a prueba la seguridad del aplicativo Java, alterando el canal de comunicación entre el aplicativo y la JVM, es decir, puede ver los métodos y sus variables, descompilando el proyecto Java. En la Figura 5-2 se muestra como colocar en funcionamiento JAVASNOOP, agregando primero la dirección del proyecto Java compilado (en este caso la Autoridad de Certificación) y pulsando el botón para atacar a la aplicación Java (*Attach & Snoop proccess)* a fin de encontrar problemas de seguridad. Dedido a que todas las Autoridades (Autoridad de Registro, Autoridad de Certificación, Autoridad deValidación) y el Applet funcionan bajo el proveedor de seguridad IAIK (recuérdese que se agrega con la línea de comando `Security.addProvider(new IAIK());`) ejecutar la prueba con una de ellas es suficiente para demostrar que la función `IAIK.secure-block-device();` protege la ejecución de los compilados JAVA, en este caso se uso el más complejo, la Autoridad de Certificación la cual accede más a espacios de memorias y disco duro que cualquier otra autoridad o aplicación de este proyecto.

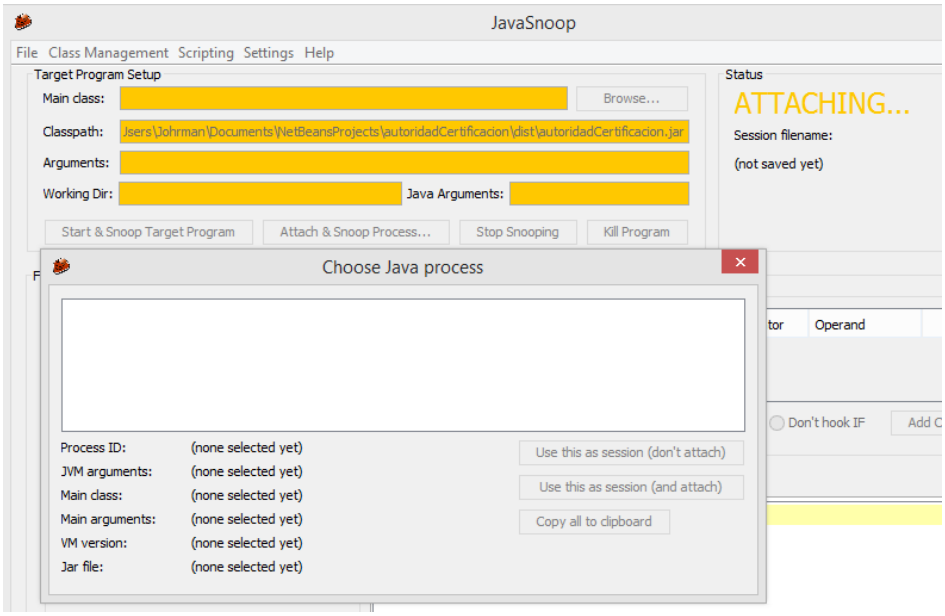
Figura 5-2 Evaluando Seguridad con JAVASNOOP



Fuente: Autor

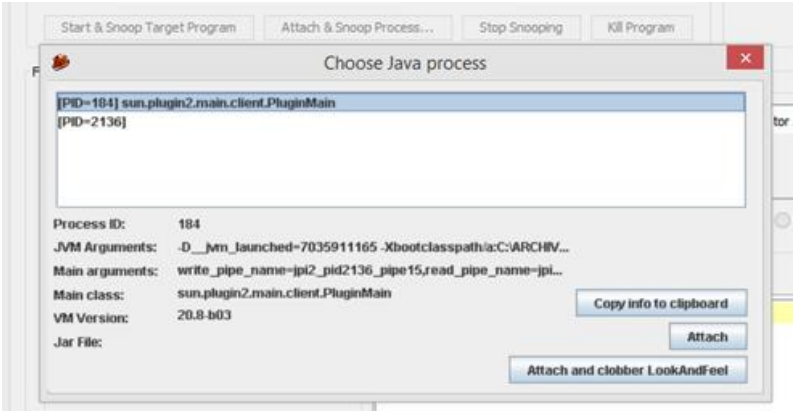
A partir de allí, se ejecuta la ventana mostrada en la Figura 5-3, pero transcurridos más de 30 minutos, el aplicativo no capturó ningún método, por lo que se puede decir que la aplicación es segura, y IAIK está protegiendo el acceso y la comunicación. Para corroborar que efecitvamente la aplicacion funciona, se probó JAVASNOOP con otro aplicativo Java (ver Figura 5-4), deduciéndose el correcto uso de JAVASNOOP y comprobando que la Aplicación desarrollada es segura, pues no divulga ningún dato de sus métodos.

Figura 5-3 Ejecutando JAVASNOOP sobre la Autoridad de Certifición



Fuente: Autor

Figura 5-4 JAVASNOOP capturando datos de métodos en Aplicativo JAVA

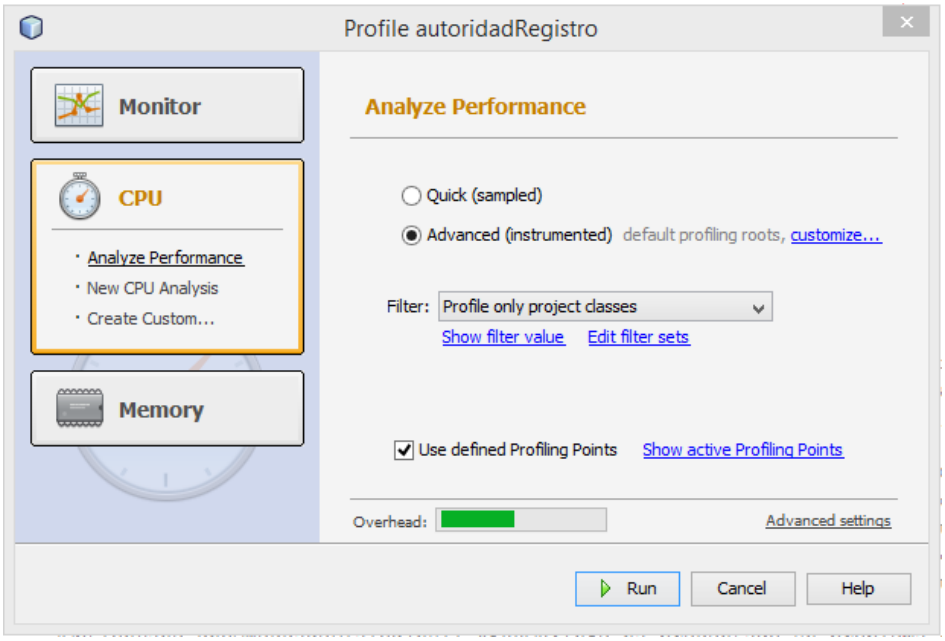


Fuente: Autor

5.1.3 Prueba de Desempeño

En esta sección se analiza el *performance* o desempeño de la aplicación. NetBeans, el IDE utilizado para desarrollar todas las aplicaciones basadas en Java de esta tesis, que tiene incorporado en sus paquetes la herramienta llamada *profiler.* La Figura 5-5 muestra la ventana de inicio de la herramienta, que permite monitorear y analizar el desempeño, y capturar la memoria, entre otras funcionalidades.

Figura 5-5 NetBenas Profiling

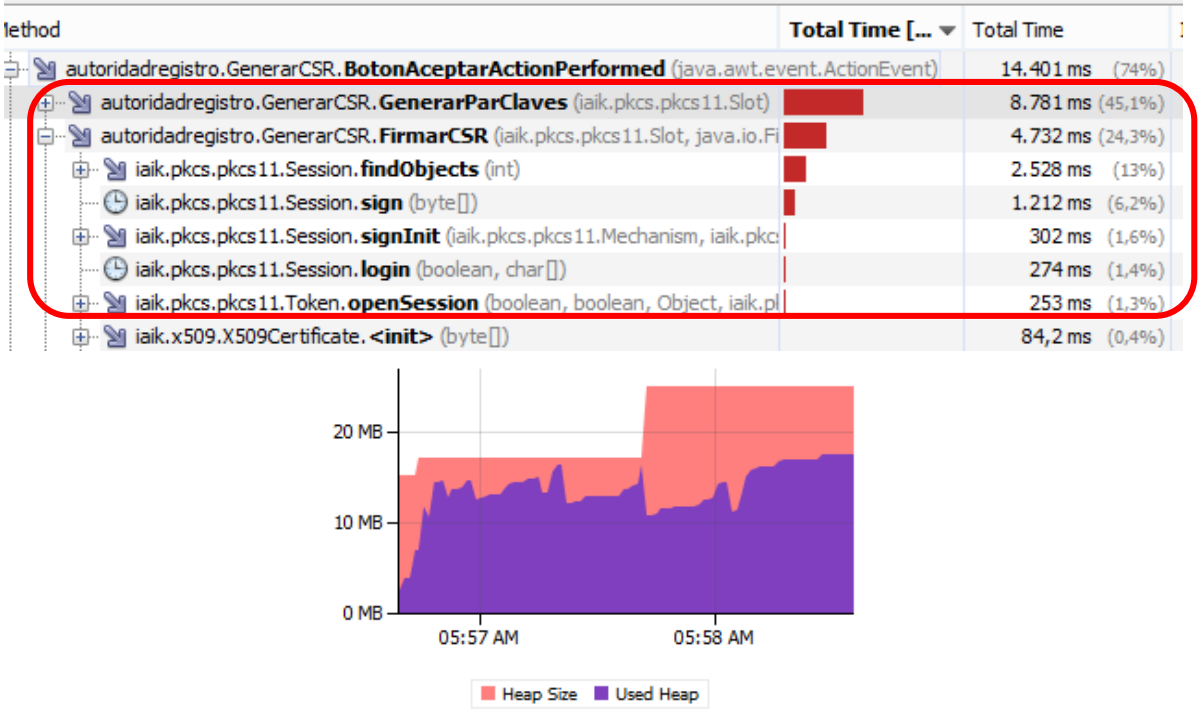


Fuente: Autor

Esta herramienta se conecta a la máquina Virtual de Java (JVM) y hace una traza de las clases que se ejecutan en el proyecto identificando *leaks* (pérdidas de memoria por objetos que no son utilizados), *heap dump* (espacio de memoria que se usa en tiempo de ejecución para almacenar las instancias de clases, objetos y *arrays*), *OutOfMemoryError* (error por perdida de objeto en la memoria) que son errores típicos en aplicaciones Java. La herramienta *profiler* da información de los hilos que se utlizan de la memoria y de la CPU. Se refiere al espacio de memoria asignado por la JVM.

En la Figura 5-6 se muestra el desempeño que tiene la Autoridad de Registro al generar un par de claves de 1024 bits (GenerarParClaves) en la tarjeta inteligente con solo 8781 milisegundos, miestras que la función que debe acceder al certificado del operario para firmar la CSR solo demora en ejecutar 4732 milisegundos un tiempo realmente bajo para este tiempo aplicaciones. En informática, este tiempo es una eternidad, pero a pesar de ello es un buen índice, ya que una vez el operario se loguee satisfactoriamente, ese hilo se finaliza y ningún otro proceso de la ventana principal demora tanto, lo que quiere decir, que el software Autoridad de Registro en general tiene muy poco impacto en el procesador del computador donde está albergado. Además, en la figura referida, se muestra el consumo de memoria de la aplicación Java en el gráfico color morado, mientras que en color Cian el espacio de memoria que consume la Maquina Virtual de Java, es de aclarar que nunca la memoria consumida por la aplicación Java puede superar a la capacidad de la memoria de la Maquina Virtual, y los altibajos que se ven son producto del buen diseño del software pues se libera memoria cada vez que no se usa un proceso, es decir, los objetos que se instancian se eliminan cuando ya no son utilizados.

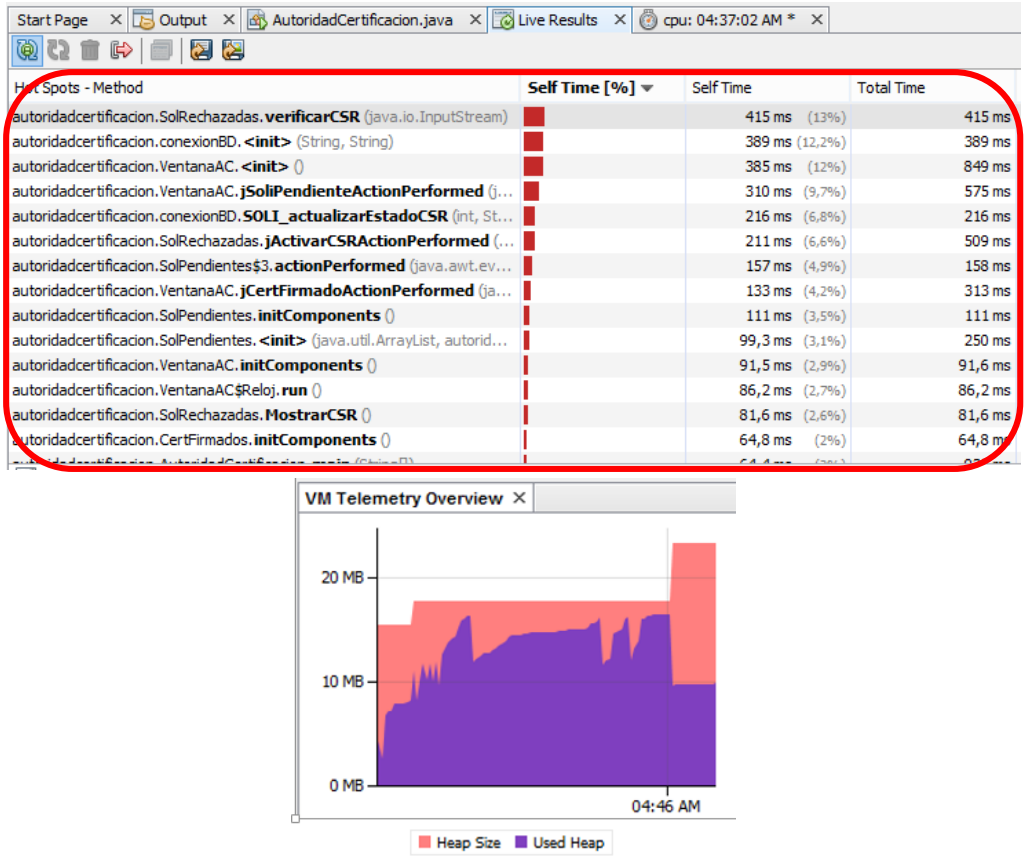
Figura 5-6 Prueba Desempeño RA



Fuente: Autor

De igual manera se analiza la Autoridad de Certificación, que presenta mejor desempeño, pues su consumo de memoria es más bajo, al igual que el uso del procesador, como se aprecia en la Figura 5-7, a pesar que se trató de llevar al nivel de estrés la aplicación (lo que puede comprobarse por el número de veces que han sido llamado los métodos) y esta respondió satisfactoriamente después de cierto tiempo, pues la memoria fue liberada, pues cada objeto no usado instanciado por los métodos es eliminado. Mientras el consumo de memoria RAM del aplicativo java se presenta en la figura en color morado no supera el espacio asignado por la memoria de la Maquina Virtual de Java, lo que precisa la estabilidad del aplicativo pues tendrá sufienciente espacio de memoria para mover dentro de la JVM.

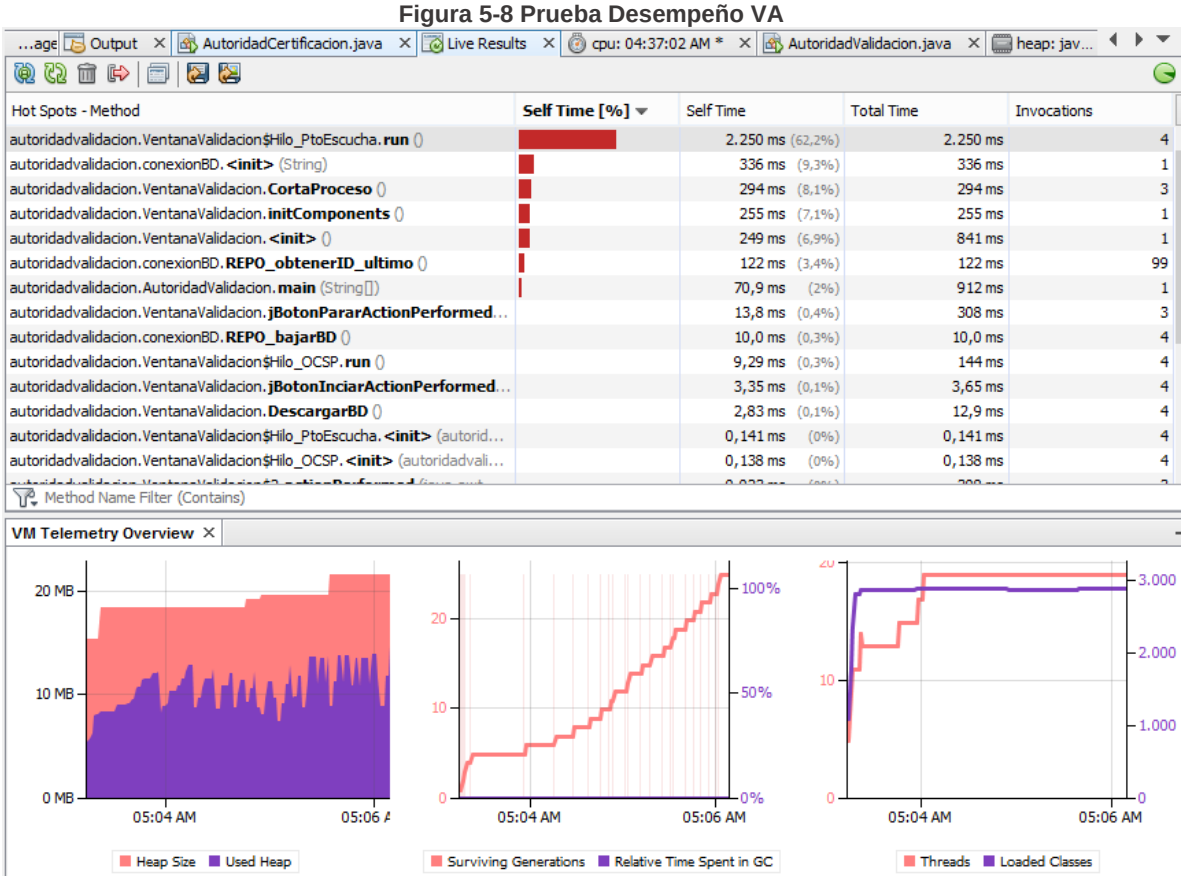
Figura 5-7 Prueba Desempeño CA



Fuente: Autor

La Autoridad de Validación es quizás la que más impacto tiene en el hardware del equipo de cómputo, como se observa en la Figura 5-8. El método que más consume recursos es el `Hilo_PtoEscucha.run()`, que abre el puerto y pregunta cada 2 segundos si hay una nueva solicitud OCSP. Este consumo también se refleja en la gráfica de uso de memoria (parte morada de la gráfica), donde se tuvieron que implementar métodos que borren gradualmente los objetos que el puerto abierto crea, pues en caso contrario dicha gráfica se dispararía pudiendo provocar un colapso de la máquina virtual de Java (parte cian de la gráfica).

Figura 5-8 Prueba Desempeño VA



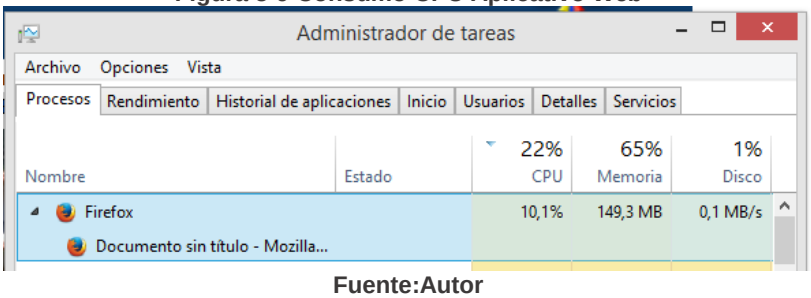
Fuente: Autor

5.2 Pruebas de la Aplicación Web

5.2.1 Prueba de Desempeño

La Aplicación Web es una de las que más consume recursos del computador, como se puede ver en la Figura 5-9, aproximadamente 150MB de memoria RAM.

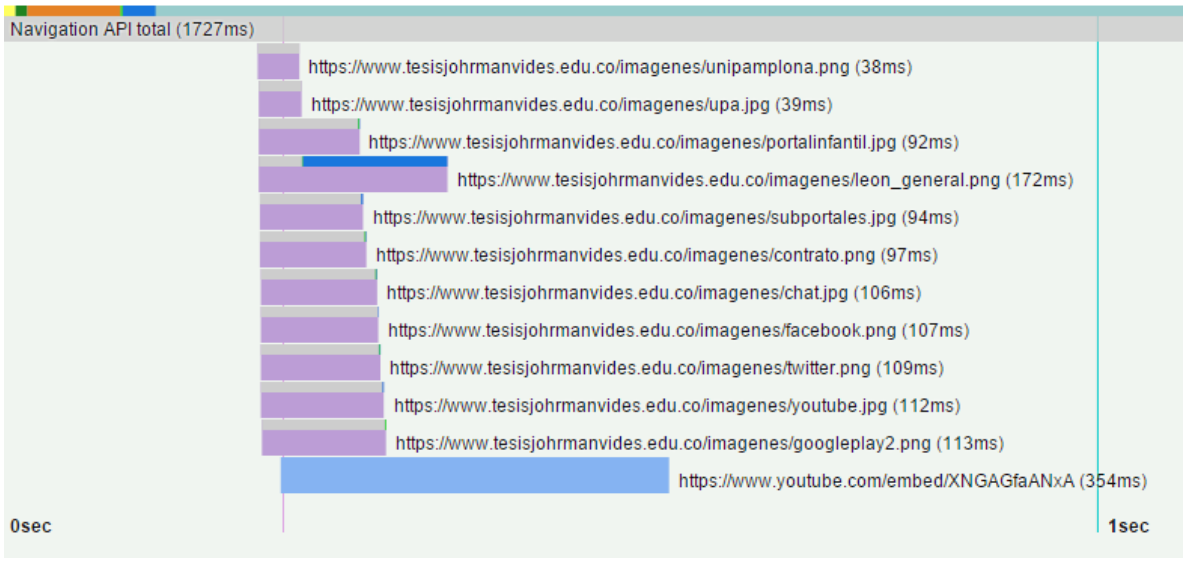
Figura 5-9 Consumo CPU Aplicativo Web



Fuente:Autor

Esto se debe a que este Applet tiene grandes funcionalidades, entre ellas detectar la desconexión de la tarjeta y del lector. En la Figura 5-10 se presenta el tiempo de carga de cada objeto de la página, que es menor a 300ms.

Figura 5-10 Timpo de carga de cada objeto de la pagina

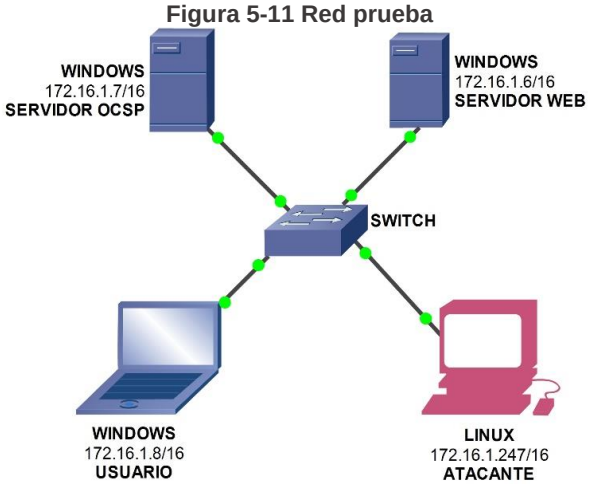


Fuente: Autor

5.2.2 Resistencia a Ataques

La red que se implementó para realizar los diferentes ataques se muestra en la Figura 5-11. Es una red LAN en la cual se encuentran un usuario, un servidor OCSP, un servidor Web, un atacante, y un switch.

Figura 5-11 Red prueba

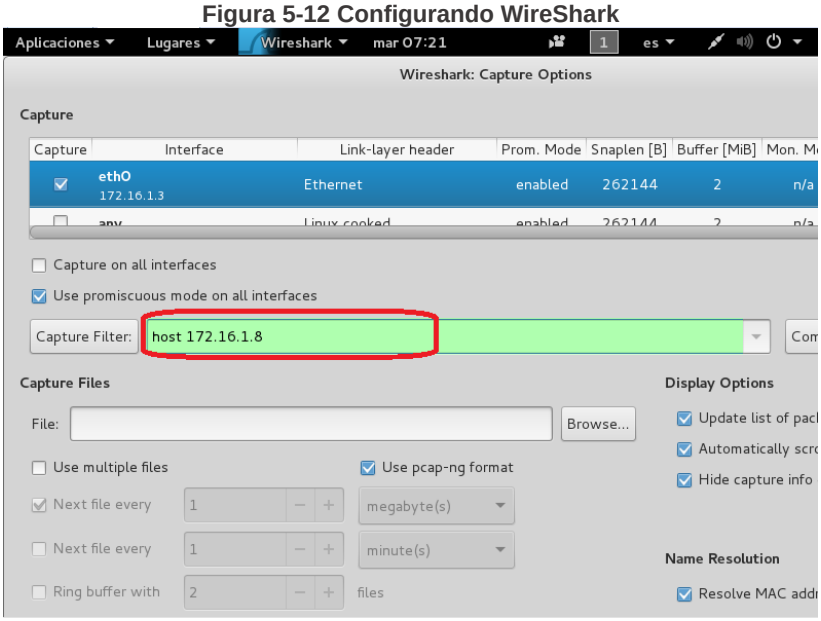


Fuente: Autor

5.2.2.1 Sniffing

A través del software *WireShark,* que provee el sistema operativo Kali Linux, se realizó el monitoreo no autorizado de paquetes entre el cliente y el servidor Web. Este ataque se realizó antes, durante y después de establecer la conexión entre el usuario, que quiere consultar su información académica, y el servidor Web, que lo autentica y autoriza. Para ello, se ingresó la dirección IP de la máquina a atacar, que es la dirección IP del usuario, como se muestra en la Figura 5-12.

Figura 5-12 Configurando WireShark



En la Figura 5-13 se muestra el tráfico capturado. Al hacer click, sobre cada uno de los paquetes se accede a su contenido. Se pueden ver las direcciones de origen y de destino, el protocolo que se está utilizando para el intercambio de los paquetes y la información que estos llevan. El paquete seleccionado, en este caso, es el *client hello*, con el que empieza la negociación del protocolo SSL.

Luego que se establece la sesión entre el cliente y el servidor Web, inicia el intercambio de información, pero esta se encuentra cifrada, como se aprecia en la Figura 5-14. Como el atacante no conoce la clave de sesión, no puede ver la información. Además, en el inicio de sesión se establece otra comunicación SSL entre el servidor Web y la base de datos, como se muestra en la Figura 5-15, para garantizar la confidencialidad de la información.

Figura 5-13 Sniffing Conexión SSL entre Cliente y Servidor

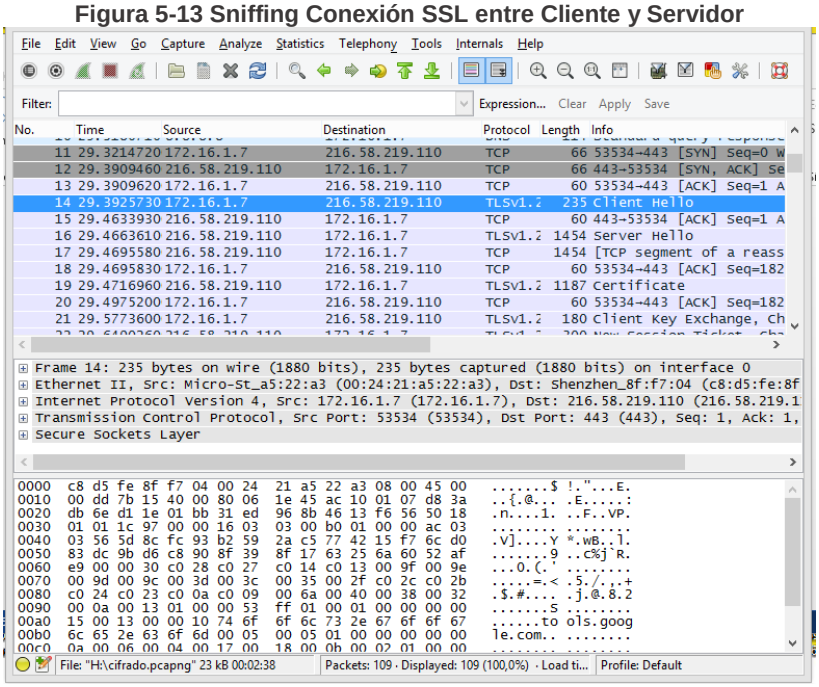


Figura 5-14 Contenido de Paquete Capturado

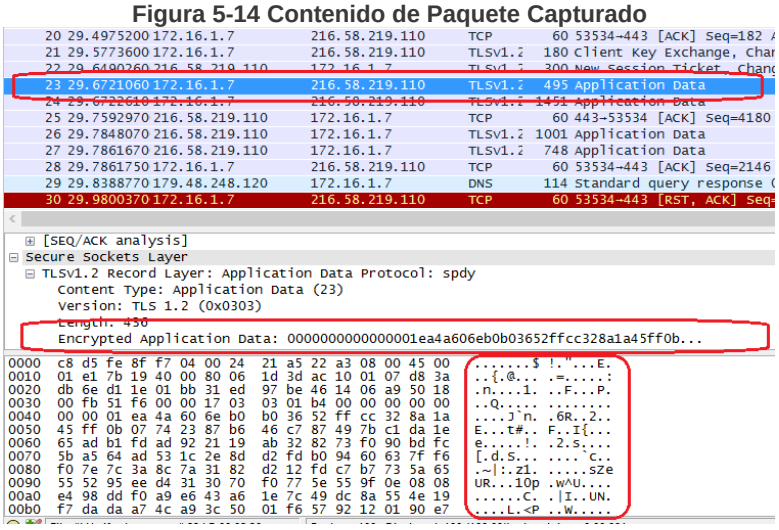
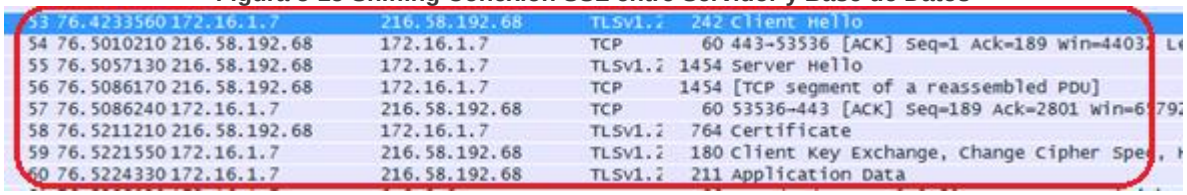
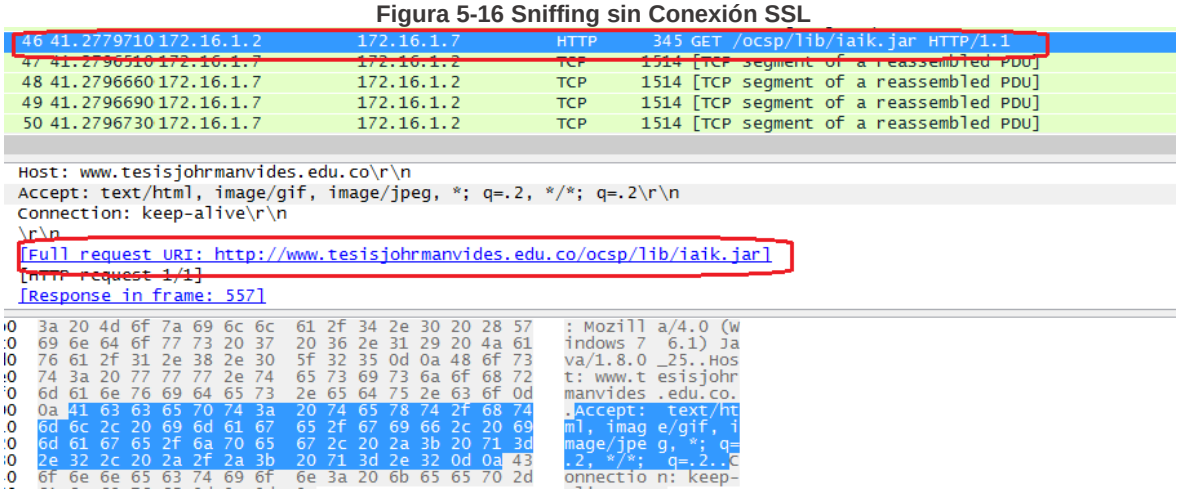


Figura 5-15 Sniffing Conexión SSL entre Servidor y Base de Datos



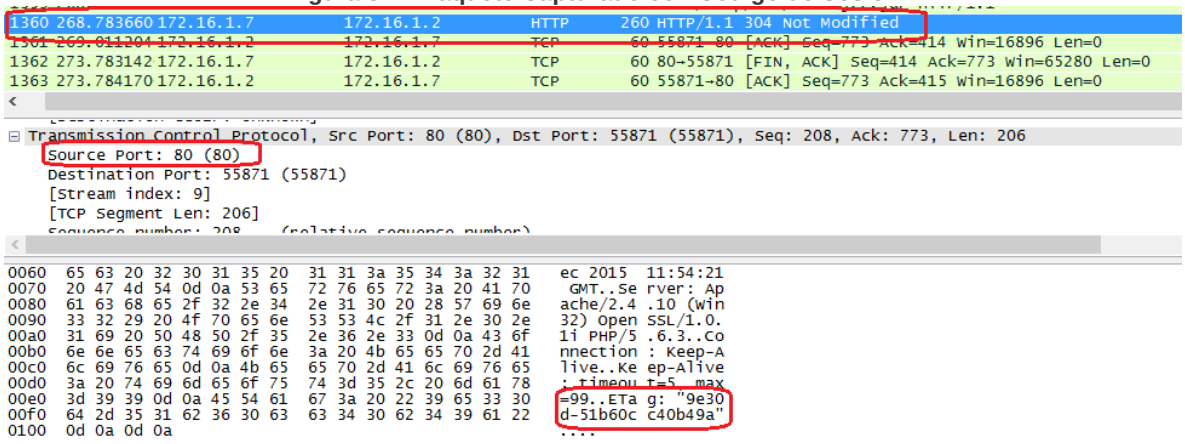
Por otra parte, si la comunicación no estuviera cifrada, se podría ver toda la información, como muestra en la Figura 5-16.

Figura 5-16 Sniffing sin Conexión SSL



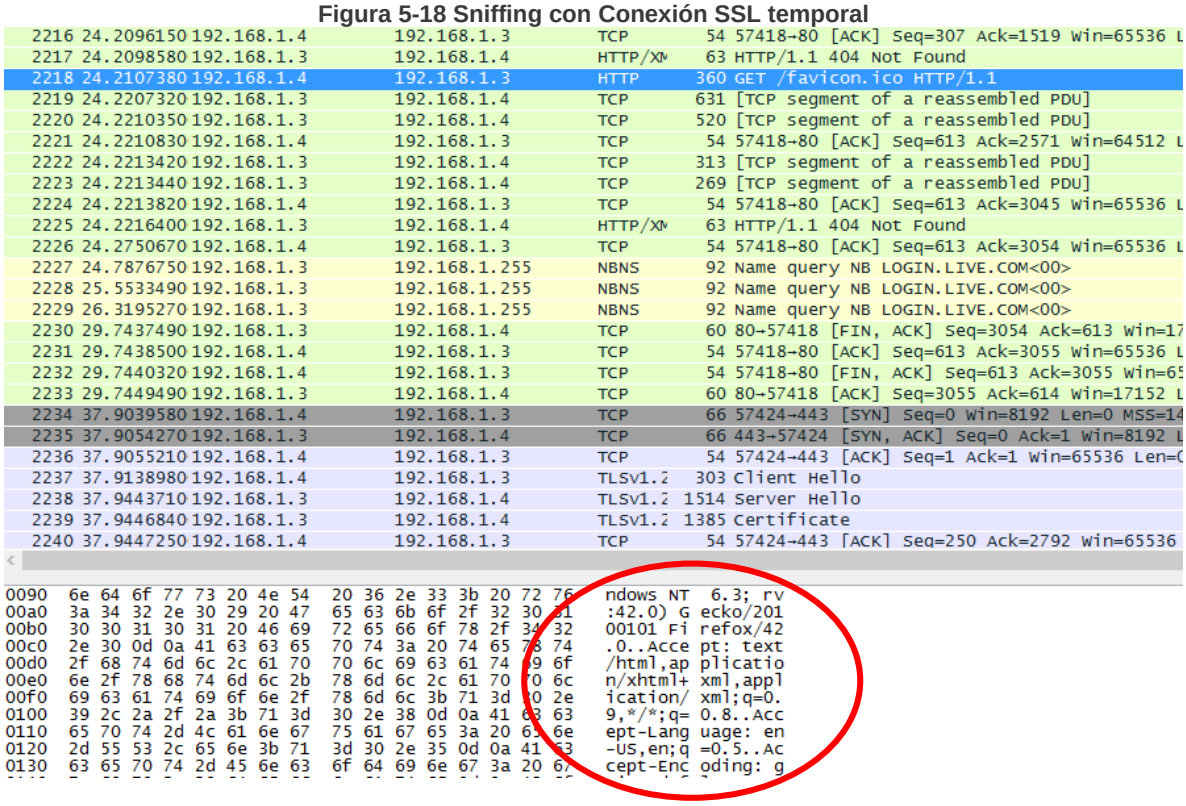
También se podría ver el código de sesión que se usa para acceder a la base de datos de información académica, como se muestra en la Figura 5-17.

Figura 5-17 Paquete Capturado con Código de Sesión



Aunque en esta negociación no se aprecio ningún valor critico como una RETO, se debe a una cifrado que implementa los applet de Java con los servidores Web, como se observa en la Figura 5-18 donde a pesar de no estar cifrada la comunicación cuando se va a intercambiar el reto la comunicación se cifrada justo hasta que este valores pase por la red, se observa que la zona en circulo rojo esta viajando el claro, que correponde a un paquete TCP, pero pocos paquetes mas adelante se cifra la conexión, y se descifra una vez se haya transferido el RETO.

Figura 5-18 Sniffing con Conexión SSL temporal



5.2.2.2 Robo de Identidad

Para realizar una ataque por diccionario o por fuerza bruta se requiere que el sistema tenga una baja entropía en las contraseñas o que no limiten el número de intentos para iniciar sesión, en este caso con no es viable ni teóricamente un ataque diccionario ni de fuerza bruta porque los numero de intentos están limitados, y además el número mínimo de longitud del PIN de 6 cifras, lo que quiere decir que el atacante deberá adivinar en solo tres intentos de un complementado de 1 millón de posibles PINES, una tarea titánicap para cualquier experto.

5.2.2.3 Suplantación de Identidad

La suplantación de identidad se podría realizar fácilmente si la comunicación no estuviera cifrada con SSL, pues solo se tendría que capturar la *cookie* con el código de acceso que el Aplicativo Web genera. Para ello, se haría un ataque *Man in the Middle* (hombre en el medio) para capturar el tráfico y obtener el acceso a la base de datos, usando las herramientas *WireShark* y Cain.

Cain escanea primero todas la IP de la LAN (ver Figura 5-19) y luego lista las MACs de las IPs capturadas (ver Figura 5-20). En este caso, Cain está operando sobre Windows (Linux como maquina anfitrión y Windows SO virtualizado).

Valga de aclarar que una suplantación de identidad, por el momento el autor es capaz de cometerlo sólo robando la cookie, que en otras palabras es la credencia que genera el Aplicativo Web al usuario, pues de otra manera sería muy tedioso o formas muy avanzadas de hacking pues en ningún momento viaja información sensible de la tarjeta inteligente, este es una de las grandes ventajas que tiene esta autenticación, sólo lo sensible que viaja en la red es la cookie la cual puede ser hurtada del equipo de la víctima y así suplantar su identidad ante el equipo remoto, pero esta solo se podría llevar acabo con una comunicación en claro, pues cifra el interlocutor no alcanzaría a entender lo que viaja por la red.

Figura 5-19 Buscando MACs en LAN

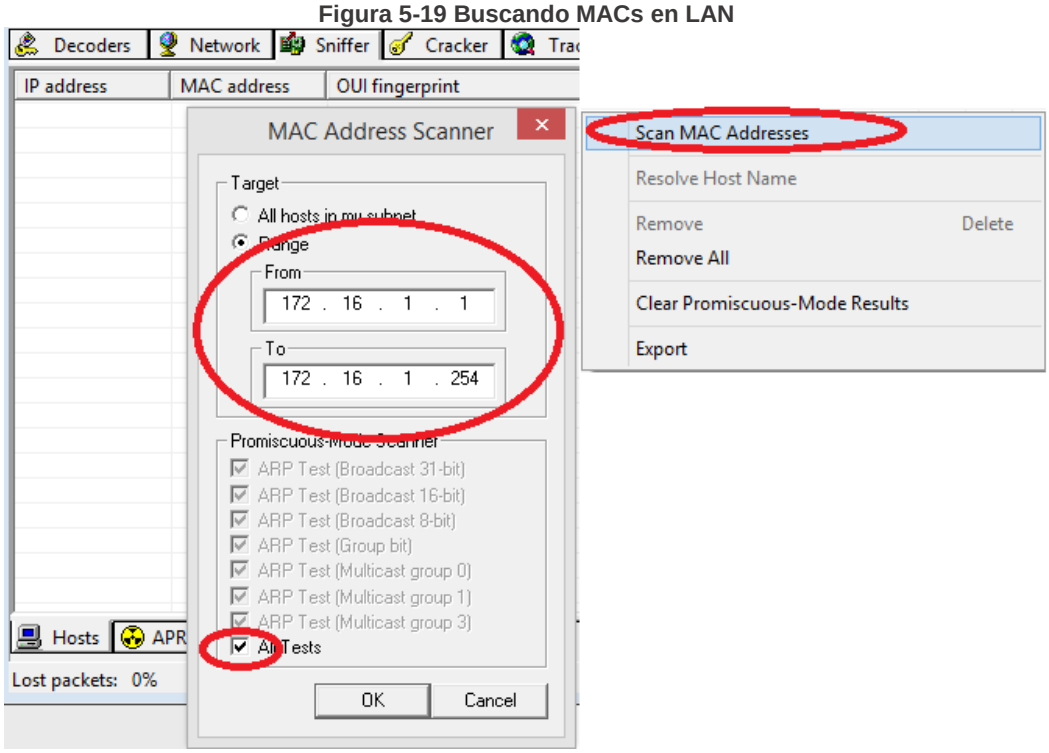


Figura 5-20 Lista de IPs Capturadas

IP address	MAC address	OUI fingerprint
172.16.1.1	C8D5FE8FF704	Shenzhen Zowee Technolog...
172.16.1.2	50E549C723F8	GIGA-BYTE TECHNOLOGY ...
172.16.1.3	000C29ADD4C9	VMware, Inc.
172.16.1.4	089E0133CDEE	QUANTA COMPUTER INC.
172.16.1.5	38B1DB7D2903	
172.16.1.6	00234D79A44D	Hon Hai Precision Ind. Co., ...
172.16.1.16	D8CB8A38CC8F	
172.16.1.18	40786AD8A8BF	
172.16.1.9	289AFA1AEF85	
172.16.1.17	74DE2B133174	Liteon Technology Corporat...
172.16.1.8	50FC9FF75809	Samsung Electronics Co.,Ltd
172.16.1.12	EC55F97540BA	Hon Hai Precision Ind. Co.,L...

Para crear el hombre en el medio, se seleccionan dos IPs: una la del router (si lo hubiese) y otra la de la víctima (ver Figura 5-21). Luego, se activa el MITM (ver Figura 5-22).

Figura 5-21 Agregando IP de Víctima MITM

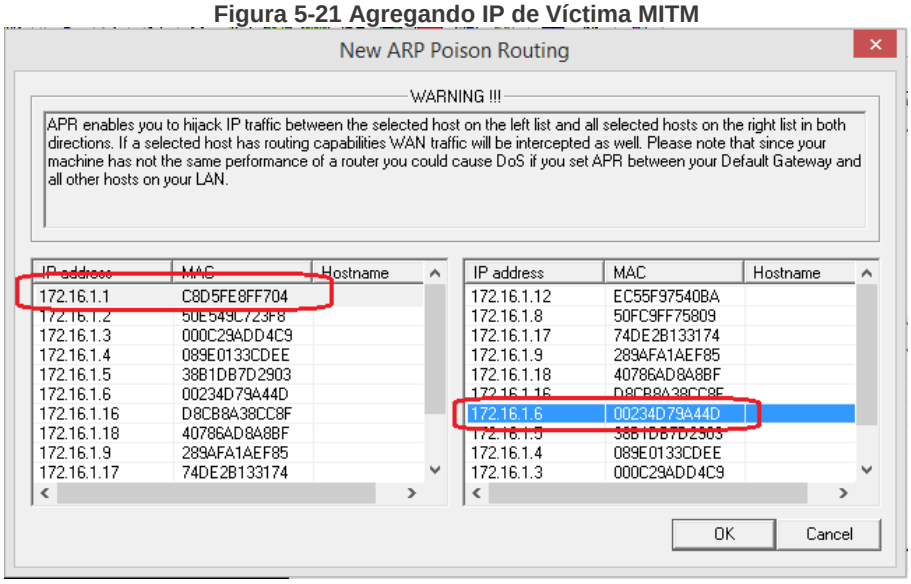
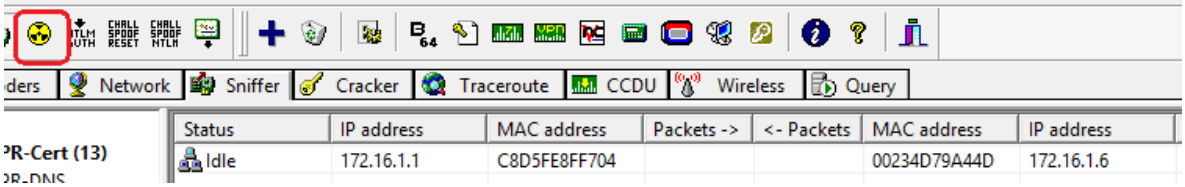
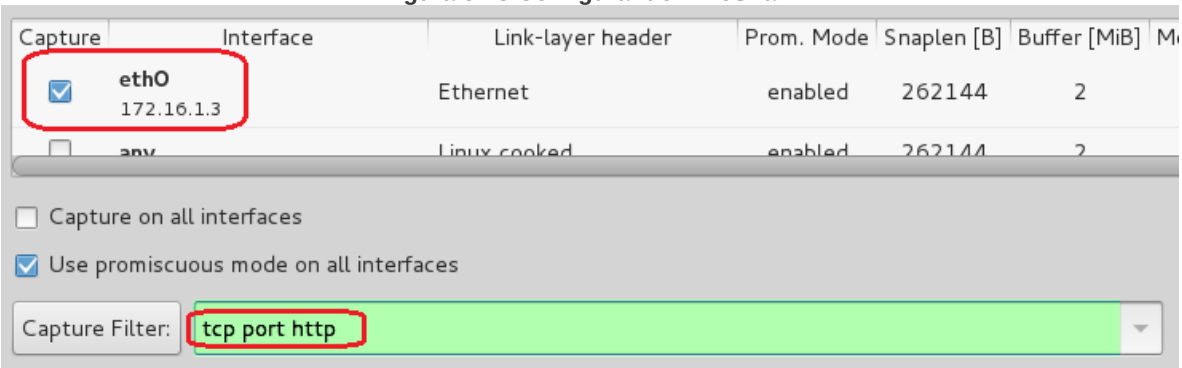


Figura 5-22 Iniciando MITM



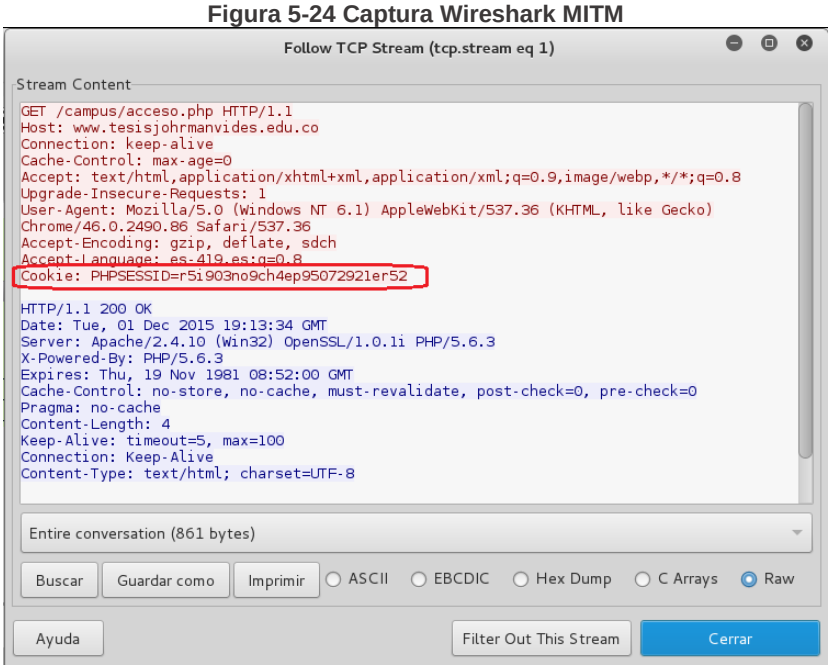
Para continuar con el proceso, se va a WireShark, y se configura para capturar paquetes tcp port http (ver Figura 5-23)

Figura 5-23 Configurando WireShark



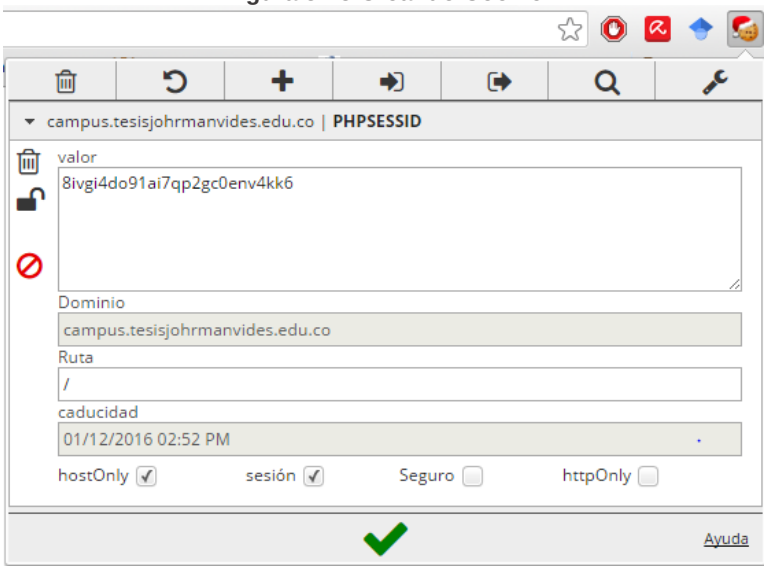
Se capturan los paquetes suficientes para llevar a cabo el robo de sesión y se analiza manualmente cada paquete hasta dar con el indicado (ver Figura 5-24).

Figura 5-24 Captura Wireshark MITM



Este valor de *cookie* no es más que el código de acceso a la base de datos de información académica, el cual puede ser replicado fácilmente por *plugins* de los navegadores, como se observa en la Figura 5-25, donde a través del complemento `editthiscookie` se puede crear una nueva *cookie* en cualquier sitio, robando la sesión del usuario y, por tanto, suplantando su identidad.

Figura 5-25 Creando Cookie



Capítulo 6

ANÁLISIS ECONÓMICO

El costo de este proyecto recae esencialmente en la provisión de los elementos y servicios que se encuentran reflejados en la Tabla 6-1.

Tabla 6-1 Presupuesto del Proyecto

PRESUPUESTO				
ÍTEM	**DESCRIPCIÓN**	**CANTIDAD**	**COSTO UNITARIO**	**COSTO**
1	Computador Escritorio Compaq CQ5115	1	$1’320.000	$1’320.000
2	Computador Portatíl Compaq CQ110	1	$600.000	$600.000
3	ACOS5 Cryptographic Smart Card Software Development Kit (ACOS5 SDK)	1	$USD200 ≈ $640.000	$USD200 ≈ $640.000
4	Papelería	1	$150.000	$150.000
5	Servicio de Internet (meses)	8	$200.000	$160.000
6	Alojamiento (meses)	8	$130.000	$1’040.000
7	Transporte Intermunipal	2	$115.000	$230.000
8	Alimentación (meses)	8	$180.000	$1’440.000
		Total		$5’580.000

Fuente: Autor

Todos los gastos correspondientes a materiales, papelería, servicios, alojamiento, y alimentación fueron cubiertos por la familia del estudiante autor y la directora de esta tesis proporcionó el kit de desarrollo de tarjetas criptográficas ACOS5 SDK.

Capítulo 7

MARCO LEGAL

7.1 Firmas Digitales

La ley colombiana fue una de las primeras de América Latina en dictar un marco legal para firmas digitales y documentos electrónicos(**Ley 521 de 1999**) [73], pero esta ley, que se creó con la intención de reglamentar el comercio electrónico, no tuvo gran despegue, pues el país sufría de un regazo tecnológico. El articulo 7 fue el principio de la validez legal de las firmas digitales, dándoles a estas el valor de una firma autografiada, pero no reglamentó su uso. No fue sino hasta septiembre del año 2000, cuando a través del **Decreto 1747 de 2000** se reglamentaron las entidades de certificación, los certificados y las firmas digitales [74]. En el año de 2001, con la asociación de varias cámaras de comercio del país, se creó Certicamara, la primera autoridad de certificación abierta colombiana. Pero cuando en el país no despegaba aún en asunto del uso de la firma digital, se creó la **Ley 962 de 2005** [75] (conocida popularmente como la ley anti-tramites) donde se da un espaldarazo a las firmas electrónicas, pues se abrió la posibilidad de que los ciudadanos, al momento de realizar trámites en entidades públicas o privadas, se autentiquen mediante un certificado digital, lo que permitió una masificación de las firmas digitales y propició que empresas privadas, con un fuerte músculo financiero, crearan su propias autoridades de certificación, conocidas como autoridades cerradas, prestando un servicio para si mismas como para el público en general, a cambio de dinero.

En el año de 2009, a través de la **Ley 1341** [76], se reestructura el Ministerio de Comunicaciones, pasándose a llamar Ministerio de Tecnologías de la Información y las Comunicaciones. Con esta ley, las firmas digitales mantuvieron su vigencia por medio del numeral 6 del artículo 2, donde bajo el principio de neutralidad tecnológica se impulsa el desarrollo del comercio electrónico, y se promueven enfoques apropiados para el reconocimiento legal de la firmas electrónicas. Ese mismo año, por medio del documento **CONPES 3620 de 2009** [ibíd]**,** se recomendó promover el uso de la firma digital. Con esto, y las anteriores normas creadas, el uso de la firma digital ya contaba con un piso jurídico estable que garantizaba su uso en documentos digitales.

En el año de 2012, se expidió la **Ley 1564 de 2012,** que reglamenta el Código General del Proceso[46]. En el Artículo 224 se redefinió el concepto de documento auténtico, por ello, la base jurídica de las firmas digitales tuvo que ampliarse, y fue expedido el **Decreto 2364 de 2012**, donde se agregan reglamentos a las firmas digitales en materia de confiabilidad de la firma, efectos jurídicos, obligaciones del firmante y criterios para establecer su grado de seguridad, entre otros. [77]

7.2 Delitos Informáticos

La **Ley 1273 de 2009** se conoce la como la ley de delitos informáticos y establece lo que se tipifica como un atentado contra la confidencialidad, la integridad y la disponibilidad de los datos y de los sistemas informáticos, así como las penas dadas a las personas que cometen este tipo de delitos. Estos son:

- **Acceso Abusivo a un Sistema Informático:** Es acceder a todo o parte de un sistema informático, sin autorización o por fuera de lo acordado, y mantenerse dentro del mismo en contra de la voluntad de quien tenga el legítimo derecho a excluirlo.
- **Obstaculización Ilegítima de Sistema Informático o Red de Telecomunicación:** Es impedir u obstaculizar el funcionamiento o el acceso normal a un sistema informático, a los datos informáticos allí contenidos, o a una red de telecomunicaciones.
- **Interceptación de Datos Informáticos:** Es interceptar datos informáticos en su origen, destino o en el interior de un sistema informático, o las emisiones electromagnéticas provenientes de un sistema informático que los trasporte.
- **Daño Informático:** Es destruir, dañar, borrar, deteriorar, alterar o suprimir datos informáticos, o un sistema de tratamiento de información o sus partes o componentes lógicos.
- **Uso de Software Malicioso:** Es producir, traficar, adquirir, distribuir, vender, envíar, introducir o extraer del territorio nacional software malicioso u otros programas de computación de efectos dañinos.
- **Violación de Datos Personales:** Es obtener, compilar, sustraer, ofrecer, vender, intercambiar, envíar, comprar, interceptar, divulgar, modificar o emplear códigos personales, datos personales contenidos en ficheros,

[46] Regula la actividad procesal en los asuntos civiles, comerciales, de familia y agrarios.

archivos, bases de datos o medios semejantes, sin estar facultado para ello, con provecho propio o de un tercero.

- **Suplantación de Sitios Web para Capturar Datos personales:** Es diseñar, desarrollar, traficar, vender, ejecutar, programar o envíar páginas electrónicas, enlaces o ventanas emergentes, con objeto ilícito y sin estar facultado para ello.

Por esta razón, los ataques realizados a las aplicaciones desarrolladas se hicieron a nivel de laboratorio para no incurrir en ningún delito.

7.3 Protección de Datos Personales

La **Ley 1581 de 2012**, dicta las disposiciones generales para la protección de los datos personales registrados en cualquier base de datos que los haga susceptibles de tratamiento por entidades de naturaleza pública o privada, definiendo los principios básicos para su tratamiento, y los deberes de los responsables y encargados del tratamiento.

Los principios para el tratamiento de los datos personales son:

- **Principio de Legalidad:** El tratamiento a que se refiere la ley es una actividad reglada que debe sujetarse a lo establecido en ella y en las demás disposiciones que la desarrollen.
- **Principio de Finalidad:** El tratamiento debe obedecer a una finalidad legítima de acuerdo con la constitución y la ley, la cual debe ser informada al titular.
- **Principio de Libertad:** El tratamiento sólo puede ejercerse con el consentimiento, previo, expreso e informado del titular. Los datos personales no podrán ser obtenidos o divulgados sin previa autorización, o en ausencia de mandato legal o judicial que releve el consentimiento.
- **Principio de Veracidad o Calidad:** La información sujeta a tratamiento debe ser veraz, completa, exacta, actualizada, comprobable y comprensible. Se prohíbe el tratamiento de datos parciales, incompletos, fraccionados o que induzcan a error.
- **Principio de Transparencia:** En el tratamiento debe garantizarse el derecho del titular a obtener del responsable o del encargado del tratamiento, en cualquier momento y sin restricciones, información acerca de la existencia de datos que le conciernan.

- **Principio de Acceso y Circulación Restringida:** El tratamiento se sujeta a los límites que se derivan de la naturaleza de los datos personales, de las disposiciones de la presente ley y la constitución. En este sentido, el tratamiento sólo podrá hacerse por personas autorizadas por el titular y/o por las personas previstas en la presente ley.

  Los datos personales, salvo la información pública, no podrán estar disponibles en Internet u otros medios de divulgación o comunicación masiva, salvo que el acceso sea técnicamente controlable para brindar un conocimiento restringido sólo a los titulares o terceros autorizados conforme a la presente ley.

- **Principio de Seguridad:** La información sujeta a tratamiento por el responsable o encargado del tratamiento, se deberá manejar con las medidas técnicas, humanas y administrativas que sean necesarias para otorgar seguridad a los registros evitando su adulteración, pérdida, consulta, uso o acceso no autorizado o fraudulento.

- **Principio de Confidencialidad:** Todas las personas que intervengan en el tratamiento de datos personales, que no tengan la naturaleza de públicos, están obligadas a garantizar la reserva de la información, inclusive después de finalizada su relación con alguna de las labores que comprende el tratamiento, pudiendo sólo realizar suministro o comunicación de datos personales cuando ello corresponda al desarrollo de las actividades autorizadas en la ley y en los términos de la misma.

Esta ley cobra importancia para el presente proyecto, porque la información académica de los estudiantes hace parte de los datos personales a proteger, por lo que atendiendo a los principios de seguridad y confidencialidad, es relevante que la institución educativa, encargada de ellos, limite el acceso a estos a través de un mecanismo de autenticación fuerte y haga que viajen de manera cifrada, como lo hacen las aplicaciones desarrolladas.

Capítulo 8

PROTECCIÓN E HIGIENE DEL TRABAJO

La interfaz de las aplicaciones de Registro de Usuarios y Web fue desarrollada de manera que fuera lo más amigable y llamativa posible, teniendo en cuenta los colores institucionales, el tamaño adecuado del texto y las gráficas, así como la ubicación apropiada de los menús e información presentada.

El informe del presente trabajo de grado se hizo teniendo en cuenta la norma NTC 1486, que garantiza la estructura y estética del documento. Además, se utilizó el formato de referencias de la IEEE.

Capítulo 9

INFLUENCIA AMBIENTAL DEL TRABAJO

La influencia ambiental de este tipo de proyectos es controversial, pues todos los servidores implementados deben estar en línea, lo que produce un consumo constante de energía eléctrica y, por ende, emisión de gases de efecto invernadero, que por pequeña que sea, no es despreciable, debido a la acumulación que se produce por la operación de otras entidades ejecutando este tipo implementaciones.

Por otro lado, la producción de las tarjetas inteligentes a partir de polietileno (compuesto derivado del petróleo) y de pequeñas cantidades de minerales, como el coltán, hace pensar que este proyecto no es amigable con el ambiente.

A pesar que a corto plazo la implementación de esta tesis puede verse como perjudicial para el medio ambiente, a largo plazo, el uso de una Infraestructura de Clave Pública, que almacena claves en una tarjeta inteligente, ayuda a simplificar los procesos administrativos, ahorrando papel. Además, con el uso de la firma digital se puede evitar la movilización de personas de un lugar a otro, impidiendo el uso de automotores a base de combustibles fósiles que pueden generar muchísimo más gases de efecto invernadero que un par servidores. En resumidas cuentas, todo los que se mueva o se invente sobre la faz de la tierra genera gases nocivos para esta, pero se debe evitar arrojar la menor cantidad posible de estos a la atmósfera, por ello, se debe invertir en productos que a largo plazo su cantidad de emisión contaminable, en promedio, sea mucho menor a los productos de uso habitual.

Capítulo 10

CONCLUSIONES Y RECOMENDACIONES

10.1 Conclusiones

Debido a la limitada capacidad de almacenamiento y procesamiento de las tarjetas inteligentes usadas, se trabajó con claves de 1024 bits para los usuarios, pues las operaciones con claves de 2048 bits se tornan muy lentas, empleando en promedio 3,6 minutos para descifrar un reto.

Por seguridad, la comunicación entre las autoridades y las bases de datos se hace utilizando el protocolo SSL, para que la información viaje cifrada, y se comprueba constantemente que las tarjetas de los operarios estén conectadas al computador, cerrándose automáticamente la sesión en caso de no ser así.

Para demostrar la efectividad y utilidad del Aplicativo de Registro de Usuarios se diseñó una Aplicación Web que consta de: una página de inicio, que simula la página Web de la institución educativa; un applet (SeleccionaCertificado), que sirve de intermediario entre la tarjeta inteligente del usuario y la función de autenticación; un autenticador (Autentica-BD_IACA), que autentica a los usuarios, comprobando la posesión de la clave privada a través de un reto cifrado; y una Base de Datos de Información Académica (BD_IACA), que contiene las notas y demás información académica de los estudiantes.

En la implementación de la Aplicación Web se utilizó: Apache como servidor, PHP y HTML para desarrollar las páginas Web, Java como lenguaje de programación, MariaDB como gestor de bases de datos, y el *middleware* IAIK-JCE, para la comunicación con las tarjetas inteligentes y el uso de sus funciones criptográficas. Además, la comunicación entre el cliente y el servidor Web, y del servidor con la base de datos, se cifran a través de SSL, para darles seguridad. De esta manera, se implementó un mecanismo de autenticación fuerte, que le solicita al usuario algo que conoce (el PIN y su clave privada), y algo que tiene (su tarjeta inteligente).

Las pruebas de amigabilidad de la Aplicación de Registro de Usuarios muestran que aunque los usuarios que no usan frecuentemente computadores necesitan más tiempo para realizar las diferentes funciones de las Autoridades de Registro, Certificación y Validación, todos los usuarios pudieron llevar a cabo exitosamente dichas funciones. En cuanto a la seguridad de este aplicativo, se estableció que al generar el par de claves de los usuarios al interior de las tarjetas inteligentes, no es necesario el borrado seguro de las mismas, y que la clave privada no va a ser extraída en ningún momento de la tarjeta. Además, a través de JAVASNOOP se pudo comprobar la seguridad de las aplicaciones desarrolladas en JAVA, ya que no se divulga ningún dato de sus métodos. Finalmente, las pruebas de desempeño muestran que la función que consume mayor tiempo es el inicio de sesión de la Autoridad de Registro y la que consume más memoria es la Autoridad de Validación, aunque la memoria es liberada gradualmente, al irse eliminando los objetos no usados.

El sniffing realizado a la Aplicación Web, muestra que al estar cifrada la información, gracias al uso de SSL, los atacantes no pueden obtener información sensible del sistema al capturar los paquetes, pues no cuentan con las claves necesarias para decifrarla. Además, el modo en que funciona la aplicación no permite realizar ataques de diccionario ni por fuerza bruta para robar la identidad de los usuarios, ya que la autenticación no se hace usando usuario/contraseña, sino a través de un reto cifrado, que cambia cada vez que el usuario intenta autenticarse y que requiere de la clave privada, y por tanto de la tarjeta inteligente del usuario, para descifrarlo correctamente. Adicionalmente, se limitó el número de intentos de autenticación a tres, por lo que el atacante sólo puede probar 3 códigos de acceso antes de que la cuenta del usuario se bloquee. En cuanto al ataque de suplantación de identidad de hombre en el medio, gracias a que la información viaja cifrada entre el cliente y el servidor y que la información en la *cookie* también se almacena de forma cifrada, no es posible que el atacante obtenga el código de acceso para acceder a la base de datos de información académica.

Los tiempos de ejecucuión de las aplicaciones fueron lo acordes aplicaciones de su tipo, siendo la creación del par de claves un de las mas demoradas pues al momento de realizarse esta operación en la tarjeta es su microprocesador quien las crea y no el pc el cual tiene, por obvias razones, mejores características que las tarjetas inteligentes. Además se evidenció en cada una de las graficas sobre el consumo de memoria, el espacio asignado para la ejecución del programa nunca superó al asignado para la Maquina virtual de Java, lo que da un estabilidad a las aplicacionescreadas.

10.2 Recomendaciones

Se debe tener en cuenta que la seguridad del sistema implementado depende en gran medida del cuidado que tengan los usuarios con sus tarjetas inteligentes y el PIN utilizado para acceder a ellas, pues un atacante que obtenga una tarjeta y su PIN puede suplantar sin problema al usuario. Por ello, se recomienda concientizar a los usuarios sobre su importante papel en la cadena de seguridad del sistema e indicarles el procedimiento a seguir para revocar su certificado, en caso que la tarjeta y/o su clave privada se vean comprometidas.

Debido a que el complemento NPAPI en el navegador Google Chrome está deshabilidato desde septiembre 15 de 2015 y es necesario para ejecutar el Applet de Java se recomienda abrir este software en otro navegador diferente, como Mozila Firefox, NetScape, Safari, Microsoft Edge, o IE.

En cuanto a los aspectos que no fueron abarcados en este trabajo de grado, pero que sería importante considerarlos en trabajos futuros están:

- Debido al tiempo de validez limitado de los certificados digitales, se podría crear un método para renovarlos, que compruebe cierta característica biométrica del usuario, como su huella digital, al insertar su tarjeta inteligente, y genere un certificado con vigencia renovada.
- Es necesario incluir entre las funciones de la Autoridad de Registro un método para borrar las tarjetas inteligentes, ya que este procedimiento se hace actualmente a través de la aplicación proporcionada por el fabricante de las tarjetas.
- Se debe incluir en la Autoridad de Certificación una opción para registrar a los operarios y generar sus claves y certificados, así como darles los permisos de acceso necesarios en las bases de datos, ya que esta funcionalidad no ha sido implementada aún en el sistema.
- Se debe analizar la viabilidad y factibilidad técnica, tecnológica y admistrativa para implementar este sistema en la Universidad de Pamplona, utilizando carnets estudiantiles con chip (tarjetas inteligentes) y lectores de tarjeta, o USBs con capacidades criptográficas, para acreditar la identidad digital de los usuarios.

Capítulo 11

BIBLIOGRAFÍA

[1] B. Neuman y T. Ts'o, «Kerberos: an authentication service for computer networks,» *Communications Magazine, IEEE,* vol. 32, nº 9, pp. 33 - 38, Sept. 1994.

[2] C. Irving A. Williamson, D. R. Pearson, S. L. Aranoff, D. A. Pinkert, D. S. Johanson y M. M. Broadbent, «Digital Trade in the U.S. and Global Economies: Part 1,» Investigación No. 332-531, USITC Publication 4415, 2013.

[3] EMC Coroporation; RSA, «Fraud report: Phishing Kits – the same wolf, just a different sheep's clothing,» Febrero 2013. [En línea]. Available: http://www.emc.com/. [Último acceso: 30 Octubre 2014].

[4] Cano, Saucedo y Prandini, «V encuesta Latinoamérica de seguridad de la información,» de *XIII Jornada Internacional de Seguridad Informática*, Bogotá, 2013.

[5] J. Cano, «XII Encuesta Nacional de Seguridad Informática,» de *XIII Jornada Internacional de Seguridad Informática*, Bogotá, 2012.

[6] A. Kerckhoffs, La cryptographie militaire, vol. IX, Journal des sciences militaire, Feb. 1883, p. 5–83.

[7] G. Álvarez, Dirección, *Lección 3. Sistemas de cifra con clave pública.* [Película]. España: Intypedia, 2010.

[8] Camerfirma S.A., «Tutorial Firma Electrónica,» [En línea]. Available: http://www.camerfirma.com/. [Último acceso: 2015 Abril 22].

[9] A. Kak, «Lecture 12: Public-Key Cryptography and the RSA,» 31 March 2015. [En línea]. Available: https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture12.pdf. [Último acceso: 22 Abril 2015].

[10] G. Álvarez Marañón, «Intypedia,» Diciembre 2010. [En línea]. Available: http://www.criptored.upm.es/intypedia/docs/es/video3/DiapositivasIntypedia003.pdf. [Último acceso: 22 Abril 2015].

[11] SANS Institute Reading Room site, «A Review of the Diffie-Hellman Algorithm,» 5 Noviembre 2001. [En línea]. Available: http://www.sans.org/reading-room/whitepapers/vpns/review-diffie-hellman-algorithm-secure-internet-protocols-751. [Último acceso: 23 Abril 2015].

[12] RSA laboratories, «PKCS#1 v2.2 RSA Cryptography Standard,» 27 Octubre 2012. [En línea]. Available: http://www.emc.com/emc-plus/rsa-labs/pkcs/files/h11300-wp-pkcs-1v2-2-rsa-cryptography-standard.pdf; pp. 58,. [Último acceso: 23 Abril 2015].

[13] A. S. Tanenbaum, «Establecimiento de una clave compartida: El intercambio de clves de Diffie-Hellman,» de *Redes de computadoras*, Pearson Educaction Inc., 2003, pp. 791-792.

[14] RSA laboratories, «PKCS#3: DIFFIE-HELLMAN KEY AGREEMENT STANDARD,» 3 Junio 1991. [En línea]. Available: ftp://ftp.rsasecurity.com/pub/pkcs/ascii/pkcs-3.asc. [Último acceso: 23 Abril 2015].

[15] J. Viega, M. Messier y P. Chandra, «Digital Signature Algorithm,» de *Network Security with OpenSSL: Cryptography for Secure Communications*, O'Reilly Media, Inc., 2002, pp. 225-229.

[16] M. N. Nabi, S. Mahmud y L. Rahman, Implementation and performance analysis of elliptic curve digital signature algorithm, 2007.

[17] Fung, «Digital Signature Cryptography Algorithms,» de *Network security technologies*, CRC Press., 2004, pp. 31-35.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest y C. Stein, «Introduction to Algorithms,» de *Section 31.7: The RSA public-key cryptosystem*, MIT Press and McGraw-Hill, 2001, p. 881–887..

[19] F. D. Escalnate Acosta, K. J. Arcia Arévalo y M. I. Mayo Bautista, «Sistema ADMONPROJECTS: Herramienta de Integración de Diferentes Aplicaciones para la Administración de Proyectos,» de *Avances en Informática y Sistema Computacionales*, Juárez, CONAIS, 2006, p. 115.

[20] J. Ramió Aguirre, «Crypt4you,» Universidad Politecnica de Madrid, 15 Marzo 2012. [En línea]. Available: http://www.criptored.upm.es/crypt4you/temas/RSA/leccion1/leccion01.html. [Último acceso: 27 Abril 2015].

[21] L. Chen y G. Gong, «RSA Encryption,» de *Communication System Security*, Boca Ratón - Florida, CRC press, 2012, pp. 191 - 193.

[22] E. Barker y A. Roginsky, «Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths.,» *NIST Special Publication,* vol. vol. 800, p. 139, 2011.

[23] T. Kleinjung, K. Aoki, J. Franke, A. Lenstra, E. Thomé, P. Gaudry, A. Kruppa, P. Montgomery, J. Bos, D. Osvik, H. Riele, A. Timofeev y P. Zimmermann, «Factorization of a 768-bit RSA modulus,» de *Advances in Cryptology–CRYPTO 2010*, Netherlands, Springer Berlin Heidelberg, 2010, pp. 333-350.

[24] S. Garfinkel, G. Spafford y A. Schwartz, «When is 128 Bigger than 512?,» de *Practical UNIX and Internet Security*, Sebastopol-EEUU, O'Reilly Media, Inc., 2003, p. 183.

[25] I. K. Salah, A. Darwish y S. Oqeili, «Mathematical attacks on RSA cryptosystem,» *Journal of Computer science,* vol. II, nº 08, pp. 665-671, 2006.

[26] IETF Trust, «DomainKeys Identified Mail (DKIM) Signatures,» Mayo 2007. [En línea]. Available: http://www.ietf.org/rfc/rfc4871.txt. [Último acceso: 30 Agosto 2015].

[27] M. WIENER, «Cryptanalysis of short RSA secret exponents,» *IEEE Information Theory Society,* vol. 36, nº 3, pp. 553-558, Mayo 1990.

[28] D. Boneh, «Twenty years of attacks on the RSA cryptosystem,» *Notices of the AMS,* vol. 46, nº 02, pp. 203-213, 199.

[29] D. Davis, «Defective Sign & Encrypt in S/MIME, PKCS# 7, MOSS, PEM, PGP, and XML,» de *USENIX Annual Technical Conference*, 2001.

[30] R. Silverman, An Analysis of Shamir's Factoring Device, vol. III, RSA Security, 1999.

[31] G. Dieter, «Padding,» de *Computer Security*, Chennai, Britsh Library, 2011, p. 269.

[32] B. Schneier, «Cryptanalysis of MD5 and SHA: Time for a New Standard,» *Computerworld,* 2004.

[33] B. A. Forouzan, «Hash Function,» de *Cryptography & Network Security*, McGraw-Hill, Inc., 2007.

[34] R. Rivest, «RFC1321 - The MD5 Message-Digest Algorithm,» Network Working Group , Abril 1992. [En línea]. Available: https://tools.ietf.org/html/rfc1321. [Último

acceso: Septiembre 2015].

[35] G. Leurent, «Message freedom in MD4 and MD5 collisions: Application to APOP,» de *Fast Software Encryption*, Springer Berlin Heidelberg, 2007, pp. 309-328.

[36] Matt_Crypto, One MD5 operation, 2006.

[37] M. Ciampa, «CompTIA Security+ 2008,» de *depth*, Australia, Cengage Learning, 2009.

[38] B. Preneel, «The first 30 years of cryptographic hash functions and the NIST SHA-3 competition,» de *Topics in Cryptology-CT-RSA 2010*, Springer Berlin, Heidelberg, 2010, pp. 1-14.

[39] J.-P. Aumasson, W. Meier, R. Phan y L. Henzen, «the SHA family,» de *The Hash Function BLAKE*, New York, Springer Heidelberg, 2014, pp. 31-42.

[40] X. Wang, Y. L. Yin y H. Yu, «Finding collisions in the full SHA-1,» *Advances in Cryptology–CRYPTO 2005,* vol. Springer Berlin Heidelberg, pp. 17-36, 2005.

[41] T. e. a. Grembowski, «Comparative analysis of the hardware implementations of hash functions SHA-1 and SHA-512,» de *Information Security*, Springer Berlin Heidelberg, 2002, pp. 75-89.

[42] K. Scarfone, M. Souppaya y P. M. Johnson, Guide to Securing Microsoft Windows XP Systems for IT Professionals: A NIST Security Configuration Checklist, 68: NIST Special Publication, 2008.

[43] B. Ballad, T. Ballad y E. Banks, «Access control process,» de *Access control, authentication, and public key infrastructure*, Jones & Bartlett Publishers, 2010, pp. 7-9.

[44] S. A. Brands, «Digital Signature,» de *Rethinking public key infrastructures and digital certificates: building in privacy*, MIT Press, 2000, pp. 77-89.

[45] P. Aguilera López, «Seguridad informática,» de *Redes seguras*, S.A. EDITEX, 2010, p. 162.

[46] A. Carvajal, «PKI y firmas digitales: aplicaciones reales,» *Revista Inventum,* nº 03, 2007.

[47] R. Shirey, «RFC2828 - Internet SecurityGlossary,» IETF Network Working Group.

[48] W. P. W. F. a. D. S. R. Housley, «"RFC3280 - Internet X.509 Public Key,» Internet Engineering Task Force .

[49] I. C. Satizábal Echevarria, Contribución a la validación de certificados en arquitecturas de autenticación y autorización., CATALUNYA, 2007.

[50] C. Adams y S. Lloyd, Understanding PKI: concepts, standards, and deployment considerations, Addison-Wesley Professional, 2003.

[51] S. Kille, «RFC 1779 - A String Representation of Distinguished Names,» 1995.

[52] ITU-T, «Recommendation X.500: Information Technology -Open Systems Interconnection - The Directory - Overview of Concepts, Models, and Services,» International Telecommunication Union, Geneva (Switzerland), 2001.

[53] Microsoft, «How Certificates Work,» 28 Marzo 2003. [En línea]. Available: https://technet.microsoft.com/en-us/library/cc776447(v=ws.10).aspx. [Último acceso: 30 Septiembre 2015].

[54] I. C. Satizábal Echavarría y J. Forné Muñoz, «contribución a la validación de certificados en arquitecturas de autenticación y autorización,» Universitat Politécnica De Catalunya, España(Catalunya), 2007.

[55] Hephaestus Books, Pgp, Including: Pretty Good Privacy, Gnu Privacy Guard, Web of Trust, Hushmail, Key Signing Party, Enigmail, Mean Shortest Distance, Seahorse,

BiblioBazaar, 2011.

[56] A. Das y C. E. Veni Madhavan, «RSA Standars,» de *Public-Key Cryptography: Theory and Practice: Theory and Practice*, bangladesh, Pearson Education India, 2009, pp. 393-406.

[57] B. Möller, T. Duong y K. Kotowicz, «This POODLE bites: exploiting the SSL 3.0 fallback.,» *Google,* 2014.

[58] B. Morton, «Is it SSL, TLS or HTTPS?,» Mayo 2012. [En línea]. Available: http://www.entrust.com/is-it-ssl-tls-or-https/. [Último acceso: 29 Septiembre 2015].

[59] I. Pellejero, F. Andreu y A. Lesta, «Ataques en redes WLAN al detalle,» de *Fundamentos y aplicaciones de seguridad en redes WLAN: de la teoría a la práctica*, Barcelona, España, Marcombo, 2006, pp. 31 - 35.

[60] W. Stallings, «Categorías de ataques,» de *Fundamentos de seguridad en redes: aplicaciones y estándares.*, Pearson Educación, 2004, pp. 7 - 10.

[61] C. Clifton y D. Marks, «Security and privacy implications of data mining,» *ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery,* pp. 15-19, 1996.

[62] M. J. Lucena Lopez, Criptografia y Seguridad en Computadores, Cuarta ed. Espana: Creative Commons, 2006.

[63] D. Philips, smart card reference manual, 1995.

[64] J. Gutierrez y J. G. T. Ayuso, «Tarjetas Inteligentes,» de *Protocolos criptográficos y seguridad en redes*, Cantabria, Universidad de Cantabria, 2003, pp. 35 - 53.

[65] IIT Kanpur, «About Smart Card Operating System,» SmartID, [En línea]. Available: http://www.cse.iitk.ac.in/users/moona/smartcard/index.php?page=smartid/smartos. [Último acceso: 30 Septiembre 2015].

[66] SB-Projects, «ISO 7816 Smart Cards,» SB-Projects, 02 Septiembre 2015. [En línea]. Available: http://www.sbprojects.com/knowledge/footprints/smart/index.php. [Último acceso: 30 Septiembre 2015].

[67] W. Rankl y W. Effing, «transmission portocol,» de *Smart card handbook*, John Wiley & Sons, 2010, pp. 203 - 205.

[68] B. Christianson, «PKCS#11 cryptographic token interface standard,» de *Security Protocols: 11th International Workshop*, Cambridge, UK, Springer Science & Business Media, 2005, pp. 326 - 327.

[69] R. Griffin y V. Fenwick, «PKCS #11 Cryptographic Token Interface Usage Guide Version 2.40,» 16 Noviembre 2014. [En línea]. Available: http://docs.oasis-open.org/pkcs11/pkcs11-ug/v2.40/pkcs11-ug-v2.40.html. [Último acceso: 30 Septiembre 2015].

[70] Graz University of Technology, «IAIK PKCS#11 Wrapper,» 2004. [En línea]. Available: http://javadoc.iaik.tugraz.at/pkcs11_wrapper/current/index.html. [Último acceso: 30 Septiembre 2015].

[71] W3Schools, «Browser and OS Platform Statistics,» 2015. [En línea]. Available: http://www.w3schools.com/browsers/browsers_os.asp . [Último acceso: 13 Noviembre 2015].

[72] W3Techs, «Server-side Languages,» Q-Success, 13 Noviembre 2015. [En línea]. Available: http://w3techs.com/technologies/overview/programming_language/all. [Último acceso: 13 Noviembre 2015].

[73] G. Colombia, «Senado,» 17 Marzo 2000. [En línea]. Available: http://www.senado.gov.co/leyes/1999/ley521.html. [Último acceso: 30 Septiembre 2015].

[74] MinTIC, «Decreto 1747 de 2000,» Bogotá DC, 200.

[75] SenadoCol, «Ley Antitrámites,» Bogotá DC, 2005.

[76] Senado, «Nuevas Tecnologías,» Bogotá DC, 2009.

[77] C. Col, «Decreto 2364 de 2002,» Bogotá DC, 2002.

[78] A. Barth, J. Caballero y D. Song, «Secure Content Sniffing for Web Browsers, or How to Stop Papers from Reviewing Themselves,» *Security and Privacy, 2009 30th IEEE Symposium on,* pp. 360 - 371, 17 May 2009.

[79] A. Chien, B. Calder, S. Elbert y K. Bhatia, «Entropia: architecture and performance of an enterprise,» *Journal of Parallel and Distributed Computing,* vol. 63, nº 5, pp. 597 - 610, Mayo 2003.

[80] M. Hirano, T. Umeda, T. Okuda, E. Kawai y S. Yamaguchi, «T-PIM: Trusted Password Input Method against Data Stealing Malware,» *Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on,* vol. I, nº 10, pp. 429 - 434, 29 April 2009.

[81] Noticias UNO, «Investigan venta de notas y títulos profesionales en Universidad de Pamplona,» 2013. [En línea]. Available: http://noticiasunolaredindependiente.com/. [Último acceso: 29 Octubre 2014].

[82] G. J. Simmons, «A survey of information authentication,» *Proceedings of the IEEE 76.5,* vol. 76, nº 5, pp. 603 - 620, Mayo 1988.

[83] R. Au, M. Looi y P. Ashley, «Cross-domain one-shot authorization using smart cards,» *Proceedings of the 7th ACM conference on Computer and communications security,* pp. 220-227, 2000.

[84] J. H. Ellis, «The history of non-secret encryption,» Taylor & Francis, 1970.

[85] D. o. D. USA, «DTIC,» 24 May 2011. [En línea]. Available: http://www.dtic.mil. [Último acceso: 2015 Abril 22].

[86] Slideplayer, «Caracteristicas de la Criptografia actual,» [En línea]. Available: http://slideplayer.es/slide/102775/. [Último acceso: 14 Abril 2015].

[87] B. Schneie, Applied Cryptography, vol. II, New York: John Wiley & Sons, 1996.

[88] Internet Engineering Task Force (IETF), «More Modular Exponential (MODP) Diffie-Hellman groups,» May 2003. [En línea]. Available: https://www.ietf.org/rfc/rfc3526.txt. [Último acceso: 23 Abril 2015].

[89] Wolfram, «BitLength,» 2007. [En línea]. Available: http://reference.wolfram.com/language/ref/BitLength.html. [Último acceso: 27 Agosto 2015].

[90] E. Weisstein, «Relatively Prime,» Wolfram Research, 25 Agosto 2015. [En línea]. Available: http://mathworld.wolfram.com/RelativelyPrime.html. [Último acceso: 30 Agosto 2015].

[91] E. L. Render, Wiener's Attack on Short Secret Exponents, 2007.

[92] H. Tiwari y K. Asawa, «Cryptographic hash function: an elevated view,» *European Journal of Scientific Research,* vol. XLIII, nº 4, pp. 452-465, 2010.

[93] M. Bellare y T. Kohno, «Hash function balance and its impact on birthday attacks,» de *En Advances in Cryptology-Eurocrypt*, Berlin , Springer Heidelberg, 2004, pp. 401- 418.

[94] H. Dobbertin, «Cryptanalysis of MD4,» de *Fast Software Encryption*, Springer Berlin Heidelberg, 1991, pp. 53-69.

[95] S. Manuel y T. Peyrin, «Collisions on SHA-0 in one hour,» *Computer Science,* vol. 5086, nº 16-35, 2008.

[96] K. SCHMEH, «MD4,» de *Cryptography and public key infrastructure on the Internet*, John Wiley & Sons, 2006, pp. 126-127.

[97] B. Morton, «Entrust,» 08 Octubre 2014. [En línea]. Available: http://www.entrust.com/understanding-sha-1-vulnerabilities-ssl-longer-secure/. [Último acceso: 01 Septiembre 2015].

[98] The Linux Information Project, «linfo,» 29 Enero 2006. [En línea]. Available: http://www.linfo.org/authentication.html. [Último acceso: 30 Septiembre 2015].

[99] S. Young y D. Aitel, «Your Defensive Arsenal,» de *The hacker's handbook: the strategy behind breaking into and defending networks.*, New York, CRC Press, 2003, pp. 122-123.

[100] C.-E. Mascareñas y B. P. Prats, Nueva enciclopedia jurídica, vol. 17, Francisco Seix, 1982.

[101] H. Simon, J. Dongarraa y E. Strohmaier, «top500,» Junio 2015. [En línea]. Available: http://www.top500.org/lists/2015/06/. [Último acceso: 16 Septiembre 2015].

[102] S. A. Schuckers, «Spoofing and anti-spoofing measures.,» *Information Security technical report,* vol. 7, nº 4, pp. 56-62, 2002.

[103] M. Dworkin, Recommendation for block cipher modes of operation: three variants of ciphertext stealing for CBC mode, 2010.

[104] IETF, «RFC2396,» Agosto 1998. [En línea]. Available: https://tools.ietf.org/html/rfc2396. [Último acceso: 30 Septiembre 2015].

[105] Microsoft, «Virtual Private Networking: An Overview,» 04 Septiembre 2001. [En línea]. Available: https://technet.microsoft.com/en-us/library/bb742566.aspx . [Último acceso: 30 Septiembre 2015].

[106] Msspace, «Types of Encryption,» [En línea]. Available: http://books.msspace.net/mirrorbooks/securitytools/ddu/ch09lev1sec1.html. [Último acceso: 30 Septiembre 2015].

[107] UNAD, «Lección 19: Derecho Informático en América Latina: Firma Electrónica, Contrataciones Electrónicas y Comercio Electrónico,» [En línea]. Available: http://datateca.unad.edu.co/contenidos/233005/contenido%20en%20linea%20PELSI_I_2013/leccin_19_derecho_informtico_en_amrica_latina_firma_electrnica_contrataciones_electrnicas_y_comercio_electrnico.html. [Último acceso: 30 Septiembre 2015].

[108] A. JIMENEZ y P. Marquez, «Electronic Commerce in United States and Colombia Free Trade Agreement: Prospective Challenges in Digital Commerce.,» *TLC - Columbia y Los Estados Unidos,* vol. 1, p. 33, 2008.

[109] Gobierno nacional de Colombia, «LEY 527 DE 1999,» Bogotá, 1992.

[110] C. Castilla, Eugenio, M. Fuquen y E. Osorio, Estudio preliminar de la Ley 527 de 1999 y del Decreto 1747 de 2000 a partir de su expedición en Colombia., Diss. , 2012.

[111] Certicamara, «Quienes somos - Historia,» [En línea]. Available: https://web.certicamara.com/quienes-somos/historia/. [Último acceso: 30 Septiembre 2015].

[112] D. R. Patel, «Message Authentication Code (MACs),» de *Information Security: Theory and Practice.*, PHI Learning Pvt. Ltd., 2008, pp. 124-128.

[113] O. Dubuisson, ASN. 1: communication between heterogeneous systems., Morgan Kaufmann, 2001.

[114] U. Hansmann, Smart card application development using Java, Springer Science & Business Media, 2012.

[115] Wikipedia, Mayo 2014. [En línea]. Available:

http://es.wikipedia.org/wiki/Transport_Layer_Security.

[116] S. &. C. G. Ortega Martorell, « PROTOCOLO DE SEGURIDAD SSL,» nº 2, pp. 6-pág., 2010.

[117] G. Urbina, Mayo 2014. [En línea]. Available: http://gerardo-urbinavelasco.blogspot.com/p/certificado-digital.html.

[118] K. G. P. H. W. Hugo Krawczyk, abril 2014. [En línea]. Available: http://eprint.iacr.org/2013/339.pdf.

[119] RAIng, «Semidúplex,» Diccionario Español de Ingeniería (1ra Ed.), Real Academia de Ingeniería de Española, 2014.

[120] D. Everett, «Smart card tutorial,» Reino Unido, [En línea]. Available: http://www.smartcard.co.uk/tutorials/sct-itsc.pdf. [Último acceso: 30 Septiembre 2015].

[121] K. Markantonakis y K. Mayes, «Smart Card Reader APIs,» de *Smart cards, tokens, security and applications*, London, Springer Science & Business Media, 2007, pp. 277 - 282.

[122] Jacquinot Consulting, Inc., «ISO 7816 Part 3: Electronic Signals and Transmission Protocols,» Cardwerk, 09 Agosto 2015. [En línea]. Available: http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-3.aspx. [Último acceso: 30 Septiembre 2015].

[123] H. Guo, «Smart Cards and their Operating Systems,» *Telecommunications Software and Multimedia Laboratory,* vol. 7, p. http://www.scardsoft.com/documents/COS/heng_guo.pdf, 2001.

[124] T. M. Jurgensen y S. B. Guthery, «Interindustry Smart card commands,» de *Smart cards: the developer's toolkit*, Prentice Hall Professional, 2002, pp. 111 - 116.

[125] University of Chicago, «Smart Card Technology and Security,» [En línea]. Available: http://people.cs.uchicago.edu/~dinoj/smartcard/security.html. [Último acceso: 30 Septiembre 2015].

[126] PCSC Workgroup, «PC/SC Workgroup Specifications Overview,» [En línea]. Available: http://www.pcscworkgroup.com/specifications/overview.php. [Último acceso: 30 Septiembre 2015].

[127] Javaworld, «How to write OpenCard card services for Java Card applets,» 1 Octubre 1998. [En línea]. Available: http://www.javaworld.com/article/2076788/embedded-java/how-to-write-opencard-card-services-for-java-card-applets.html. [Último acceso: 30 Septiembre 2015].

[128] J. F. Vélez Serrano, Diseñar y programar, todo es empezar: Una introducción a la programación orientada a objetos usando UML y Java., Universidad Rey Juan Carlos - Escuela Técnica Superior de Ingeniería Informática: Editorial Dykinson, 2010.

[129] A. Kamble, «Java card technology,» 15 Octubre 2014. [En línea]. Available: http://www.slideshare.net/amolkamble16121/java-card-technology-40283281. [Último acceso: 30 Septiembre 2015].

[130] ALEGSA, «Definición de Bytecode,» 12 Junio 2010. [En línea]. Available: http://www.alegsa.com.ar/Dic/bytecode.php. [Último acceso: 30 Septiembre 2015].

[131] ORACLE, «Java Cryptography Architecture (JCA) Reference Guide,» [En línea]. Available: http://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html. [Último acceso: 30 Septiembre 2015].

[132] S. Liang, The Java Native Interface: Programmer's Guide and Specification.,

Addison-Wesley Professional, 1999.

[133] Intermania, Mayo 2014. [En línea]. Available: http://www.internetmania.net/int0/int93.htm.

[134] LSI, Enero 2014. [En línea]. Available: http://www.tic.udc.es/~nino/blog/lsi/documentos/1-categorias-ataques.pdf.

[135] M. J. L. Lopez, Criptografia y Seguridad en Computadores, Cuarta ed., Creative Commons, 2006.

Anexo A

MANUAL DE USUARIO

A.1 Autoridad de Registro

A.1.1 Requisitos Mínimos del Sistema

Sistema Operativo: Microsoft Windows XP SP3
Software requerido: JAVA JRE 7.1.0
Procesador: 1GHz o más rápido
Memoria RAM: 2GB
Disco Duro: 2 GB libres
Conectividad: Tarjeta Ethernet 10Mbps y puerto USB2.0

A.1.2 Instalación

Hardware requerido:

- ✓ Lector de Tarjeta Inteligente ACOS ACR38.
- ✓ Tarjeta Inteligente compatible con estándar PKCS#11.

PASO1: Instalar la versión mas reciente de JAVA JRE

PASO2: Se debe instalar el lector de la Tarjeta inteligente, comprobar por el Adminitrados de Dispositivos de Windows hasta que lo reconozca.

PASO3: Alojar en la ubicación System32 el complemento del estándar PKCS11 para Windows pkcs11wrapper.dll. NOTA: Existen la versión para sustemas de 32bits y 64bits.

PASO4: Ejecutar el paquete autoejecutable AutoridadRegistro.jar en cualquier carpeta de acceso rápido (por ejemplo, el escritorio). NOTA: el Autoejecutable contiene los demás fichero necesario, como openSSL y bibliotecas de Aplicación DLL

A.1.3 Funcionamiento

La Autoridad de Registro permitirá a los operarios registrar a los estudiantes, solicitar los certificados a la Autoridad de Certificación (CSRs), generar las claves de los estudiantes en las tarjetas inteligentes e importar el certificado en las mismas.

A.1.3.1 Inicio de Sesión

Al ejecutar la aplicación, se mostrará la ventana de inicio de sesión de la Figura A-1, que le permitirá a los operarios registrados ingresar a la aplicación. Para ello, debe estar insertada la tarjeta inteligente del operario en el lector de tarjetas, pues la aplicación detecta dicha tarjeta y solicita el PIN de la misma, de lo contrario, mostrará un mensaje de error. Para ingresar a la aplicación, introduzca el PIN y haga click en el botón **Aceptar**. Si el PIN es correcto, se muestra el mensaje de inicio de sesión exitoso de la Figura A- 2.

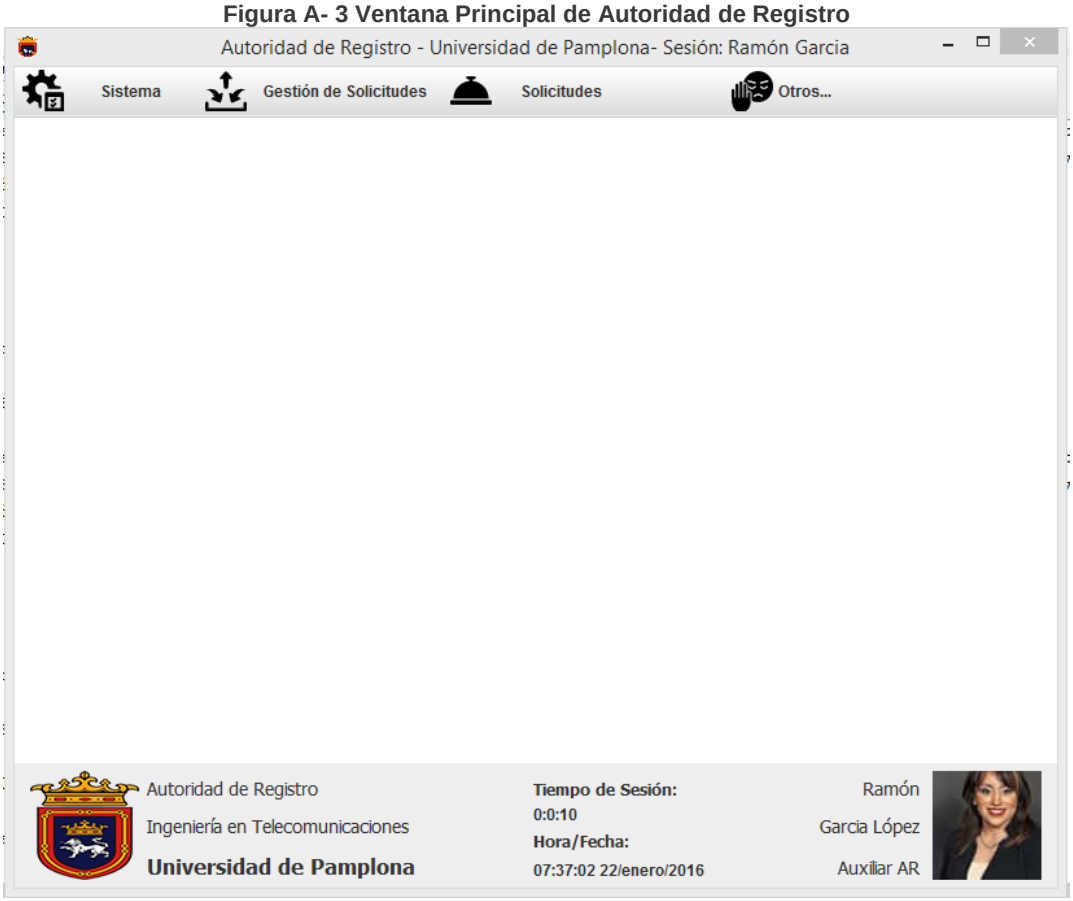
Figura A- 1 Inicio de Sesión

Figura A- 2 Inicio de Sesión Exitoso

A.1.3.2 Menú

Después de que el operario ingresa exitosamente a la aplicación, se despliega la ventana que se muestra en la Figura A- 3. En la parte inferior se encuentran el nombre y la foto del operario, así como la fecha y hora de inicio de sesión, y el tiempo transcurrido desde el inicio de sesión. En la parte superior se muestra un menú, que consta de 3 opciones: **Sistema**, **Gestión de Solicitudes**, **Solicitudes** y **Otros.**

Figura A- 3 Ventana Principal de Autoridad de Registro



Al hacer click sobre la opción **Sistema**, se despliega un sub-menú con tres opciones (ver Figura A- 4): **Cerrar Sesión**, para cerrar la sesión del operario, en caso que se quiera cambiar de usuario; **Salir**, para cerrar la aplicación; y **Acerca de…** , para desplegar una ayuda e información acerca del software

Al hacer click sobre la opción **Gestión de Solicitudes**, se despliega un sub-menú con tres opciones (ver Figura A- 4) : **Generar Solicitud**, para visualizar el formulario que permite registrar a los estudiantes en el sistema; **Cancelar Solicitud,** para cancelar una solicitud de certificado que aún no haya sido aprobada por la Autoridad de Certificación; y **Solicitudes Emitidas,** para ver el estado de las solicitudes de certificados emitidas.

Al hacer click sobre la opción **Solicitudes**, se despliega un sub-menú con dos opciones (ver Figura A- 4): **Ver Solicitudes Aprobadas**, para ver las solicitudes aprobadas por la Autoridad de Certificación e importar los certificados a la tarjeta inteligente; y **Ver Solicitudes Rechazadas,** para ver las solicitudes rechazadas por la Autoridad de Certificación y el motivo de dicho rechazo.

Figura A- 4. Menú y Submenús

- **Generar Solicitud**

Cuando se hace click sobre la opcion **Generar Solicitud**, se despliega en la parte central de la pantalla, el formulario mostrado en la Figura A- 5, que contiene los siguientes campos:

  - **Número de Identificación:** En este campo se debe introducir el número del documento de identidad del estudiante que se va a registrar
  - **Primer Nombre:** En este campo se debe introducir el primer nombre del estudiante
  - **Segundo Nombre:** En este campo se debe introducir el segundo nombre del estudiante
  - **Primer Apellido:** En este campo se debe introducir el primer apellido del estudiante
  - **Segundo Apellido:** En este campo se debe introducir el segundo apellido del estudiante
  - **Correo Electrónico:** En este campo se debe introducir el correo electrónico del estudiante
  - **Teléfono:** En este campo se debe introducir número telefónico del estudiante

- **Dirección:** En este campo se debe introducir la dirección de residencia del estudiante
- **País de Sede:** En este campo se debe introducir el país donde se encuentra la sede en la que estudia el estudiante
- **Departamento de Sede:** En este campo se debe introducir el departamento o estado donde se encuentra la sede en la que estudia el estudiante
- **Ciudad de Sede:** En este campo se debe introducir la ciudad donde se encuentra la sede en la que estudia el estudiante
- **Organización:** En este campo se debe introducir el nombre de la institución educativa donde estudia el estudiante
- **Programa Académico:** En este campo se debe introducir el programa académico al que pertenece el estudiante

Cada casilla del formulario valida los datos a medida que son introducidos, colocándose de color rojo, si la informacion no tiene el tipo ni la longitud adecuada, y de color verde, si es correcta. Los campos con * son campos obligatorios, mientras que los otros pueden omitirse. Sólo se podrá Generar la CSR, si todas las casillas llenadas están de color verde (ver Figura A- 5).

También, se puede cargar una foto del estudiante, haciendo click en el botón **Cargar**. Se desplegará entonces la ventana mostrada en la Figura A- 6, que permite buscar en el computador la foto a cargar. Existe además un cuadro de texto en donde se puede añadir información adicional del estudiante.

Tras llenar todos los campos que se consideren convenientes, se debe insertar en el lector la tarjeta inteligente del estudiante y oprimir el botón **Generar Solicitud**, para registrar los datos del usuario en la base de datos, enviar una solicitud de certificado a la Autoridad de Certificación y generar el par de claves en la tarjeta. Si estas operaciones se realizan con éxito, se mostrará la ventana mostrada en la Figura A- 7, de lo contrario se desplegará un mensaje de error. Si se pulsa el botón **Cancelar**, se volverá a la ventana principal de la aplicación.

- **Cancelar Solicitud**

Cuando se hace click sobre la opción **Cancelar Solicitud**, se despliega una lista con las solicitudes de certificado realizadas por el operario actual (ver Figura A- 8). Pulsando el click derecho del mouse sobre una fila de la tabla, se despliega un menú con las opciones: **Ver CSR**, que permitirá ver el contenido de la solicitud, como se muestra en la Figura A- 9; y **Cancelar Solicitud**, que permitirá cancelar la solicitud, en caso de haberse cometido algún error al introducir los datos del usuario, siempre y cuando dicha solicitud no haya sido aprobada aún por la Autoridad de Certificación. Como se muestra en la Figura A- 10, se pregunta primero si realmente se quiere cancelar la CSR seleccionada. Si es así, el operario debe hacer click en el botón **SI**, tras lo cual recibirá un mensaje de cancelación exitosa (ver Figura A- 11)

Figura A- 5. Formulario de Generar Solicitud

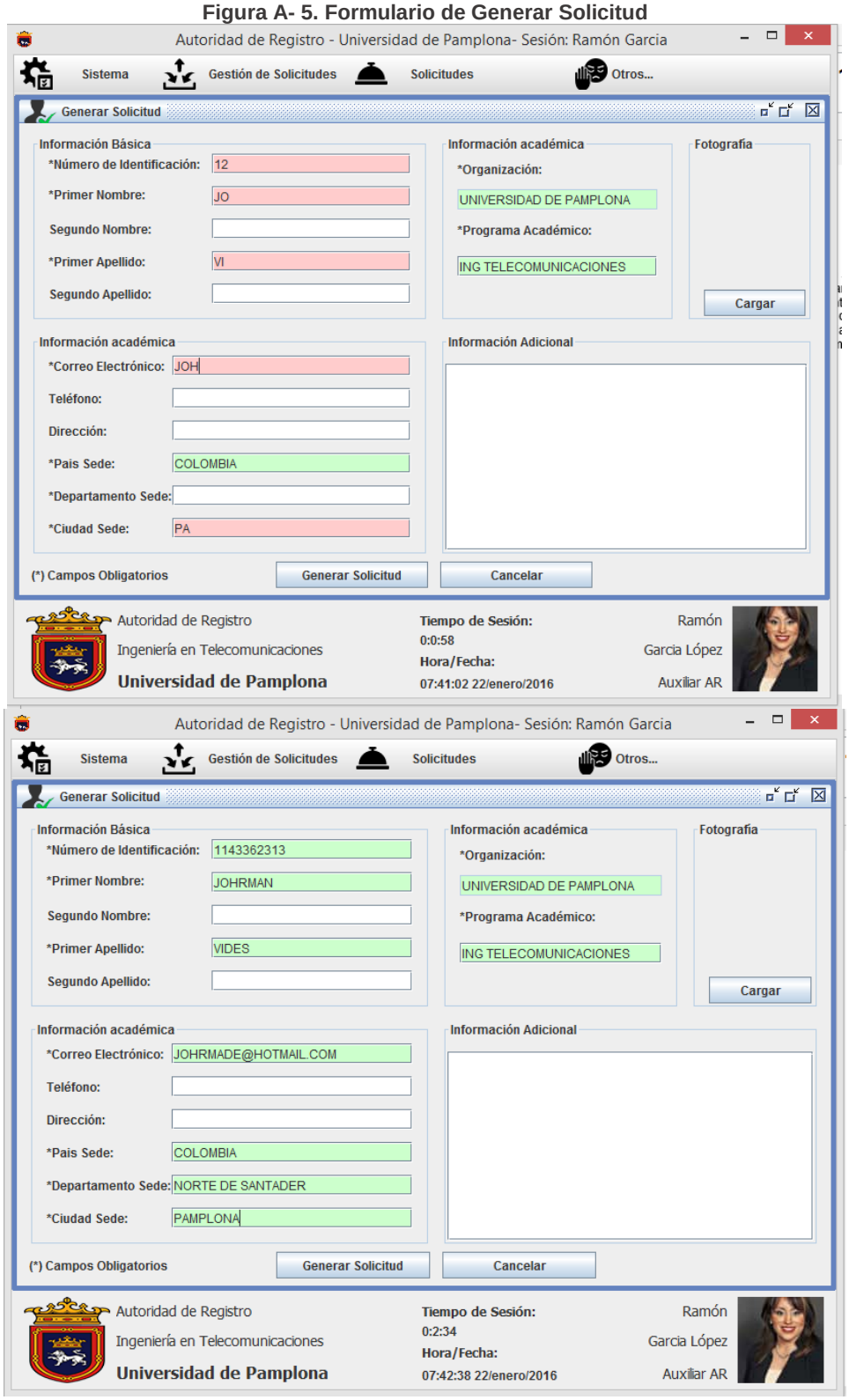


Figura A- 6 Cargar Foto

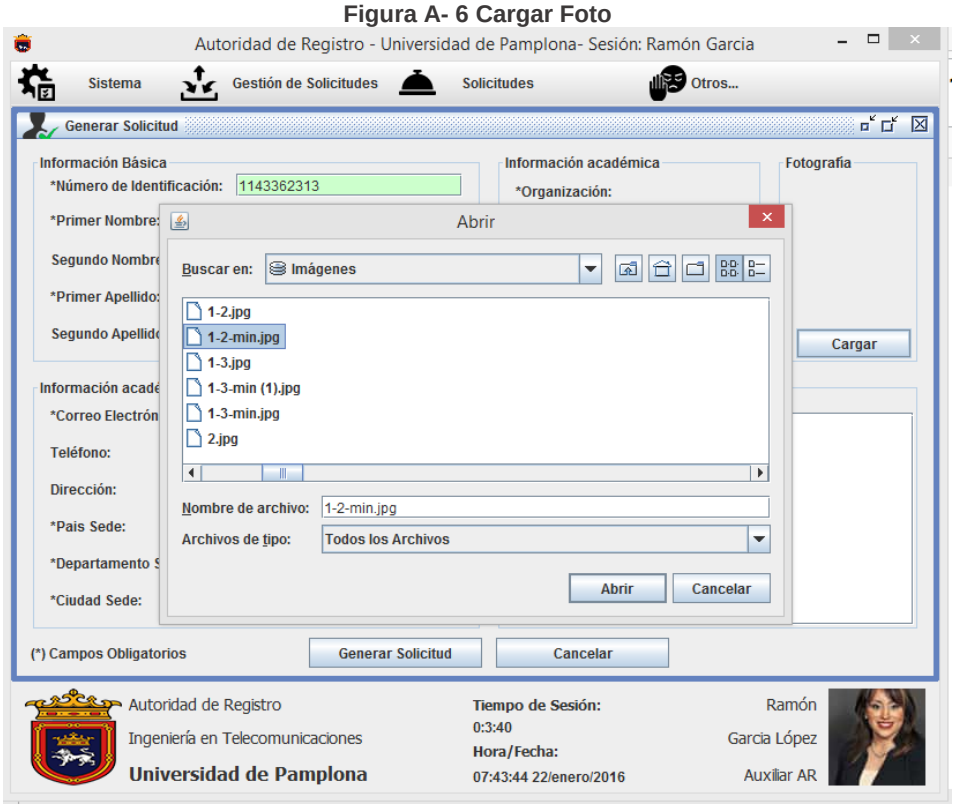


Figura A- 7. Generación Exitosa de Solicitud

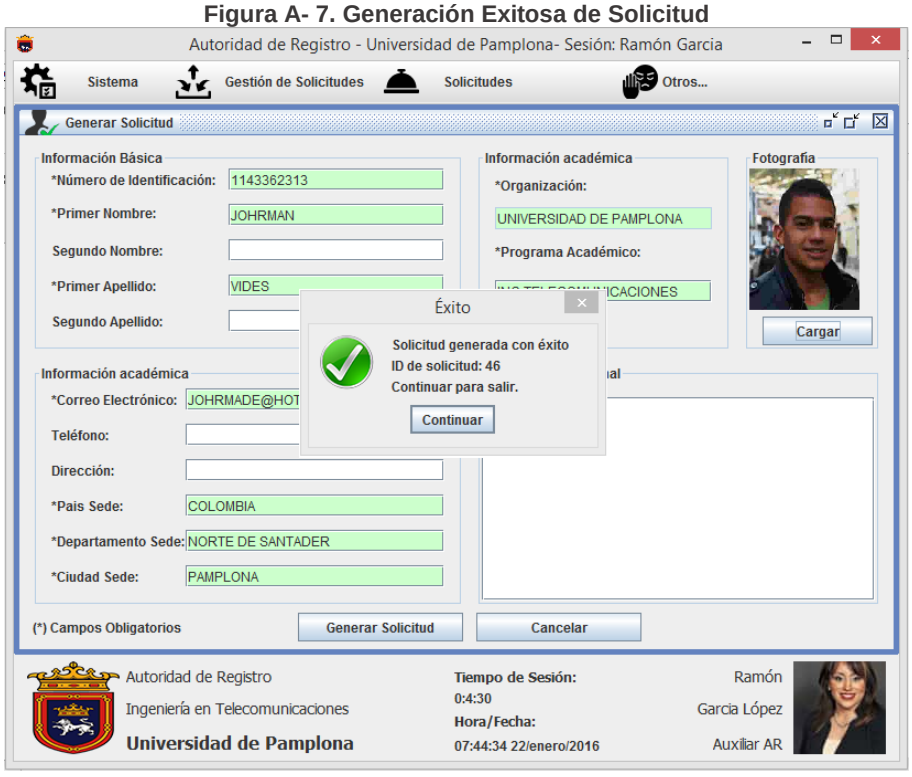


Figura A- 8. Cancelar Solicitud - Submenú

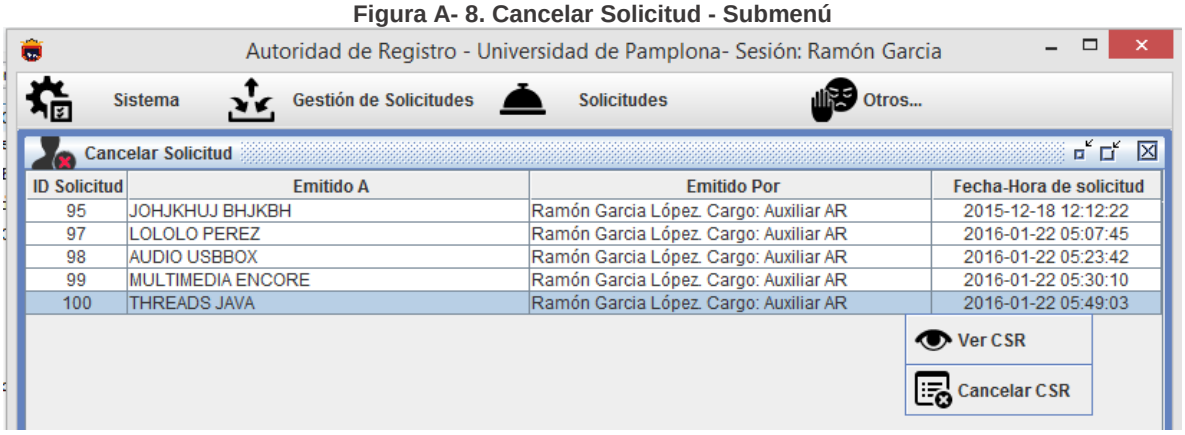


Figura A- 9. Ver CSR

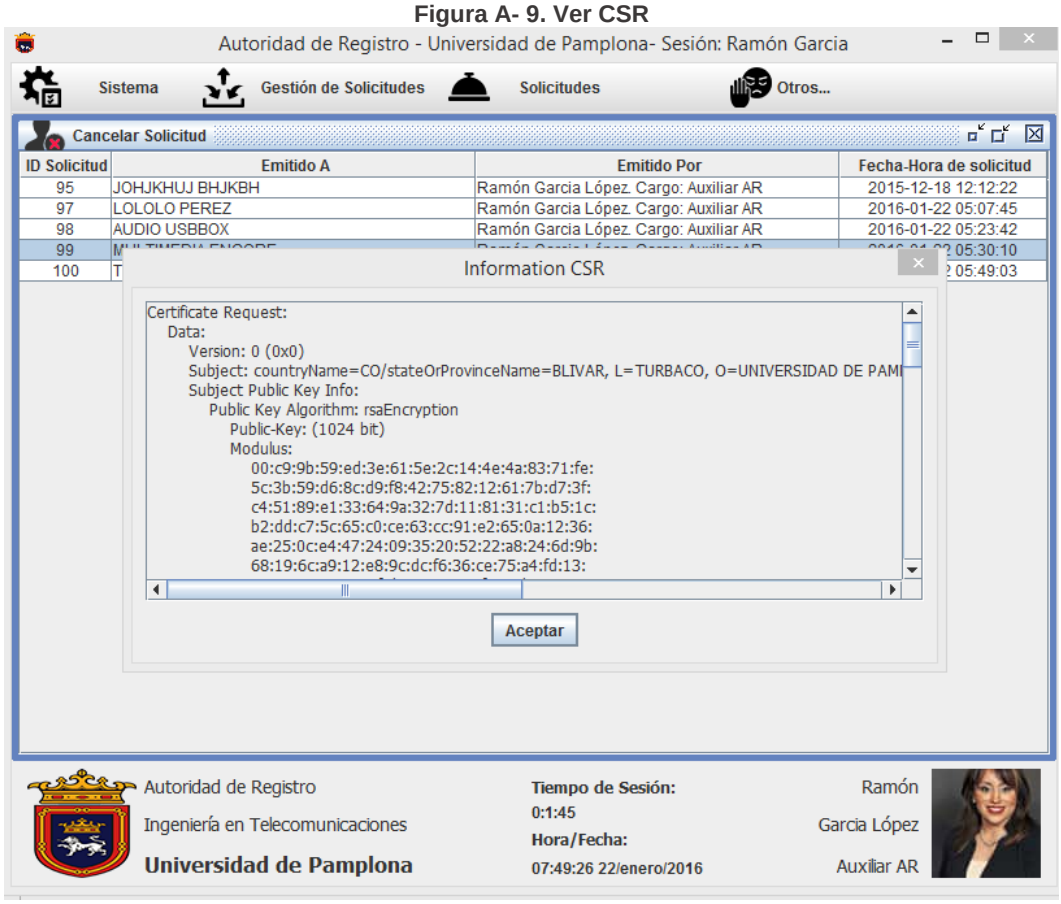


Figura A- 10. Cancelar Solicitud

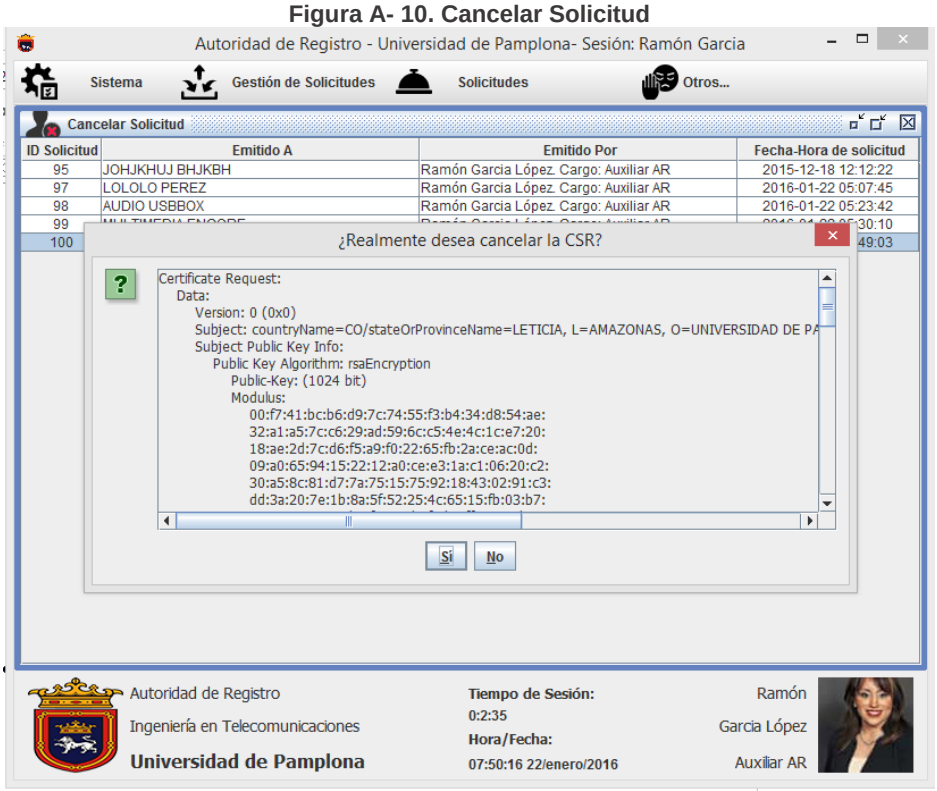
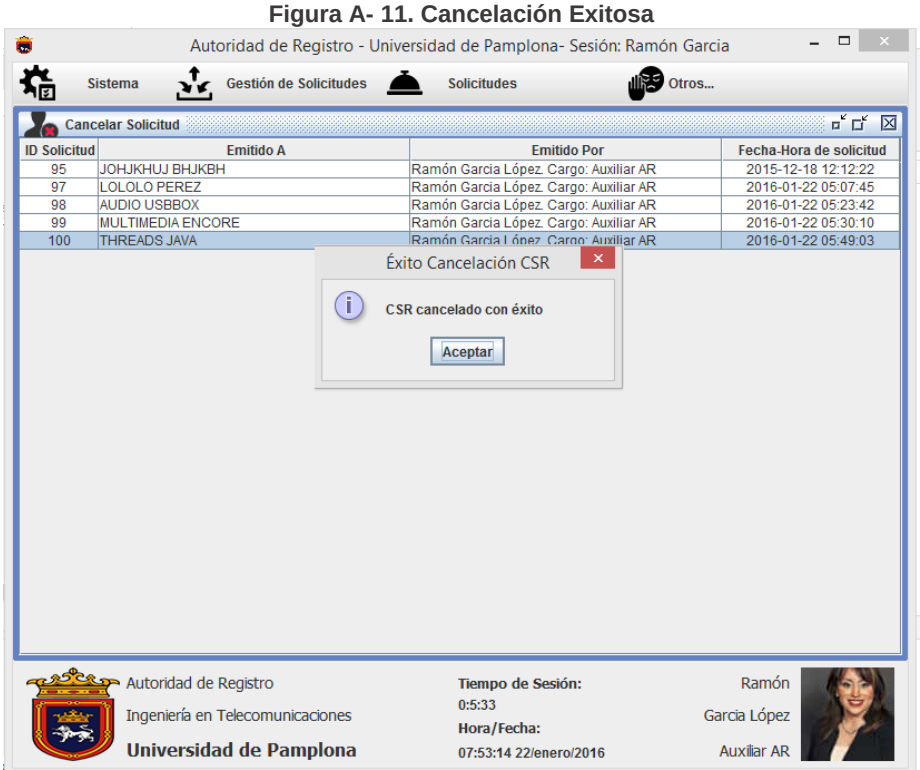


Figura A- 11. Cancelación Exitosa

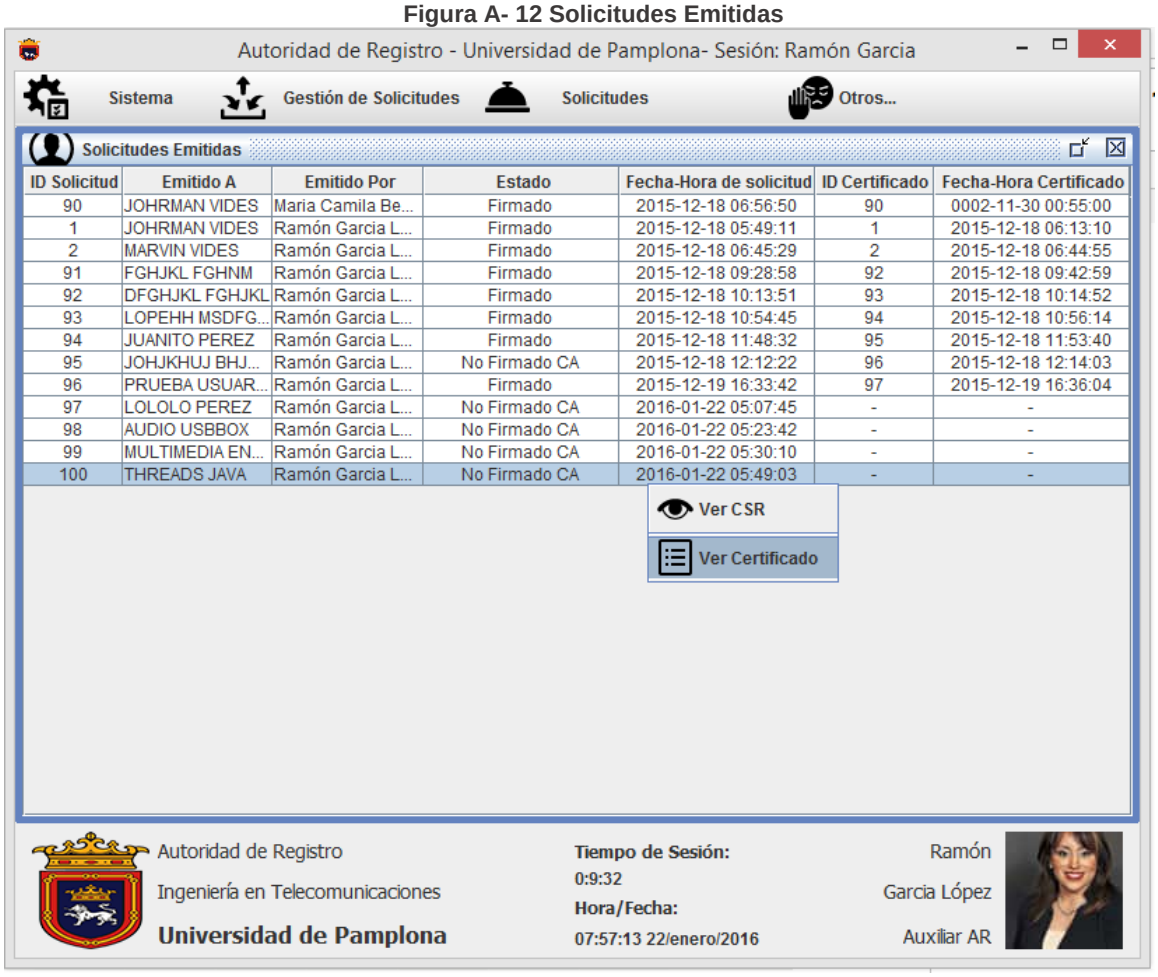


- **Solicitudes Emitidas**

La última opción seleccionable del sub-menú, es Solicitudes Emitidas, opción que desplegara una lista de todos las solicitudes emitides, su estado, la información del operario que lo ha emitido, la fecha y la hora de emisión.

Nuevamente al hacer click derecho (ver Figura A- 12) sobre alguno de estos, se ejecutara un menú contextual, de nuevo podrá verse el certificado (en caso que este firmado), o poder visualizar la información del CSR.

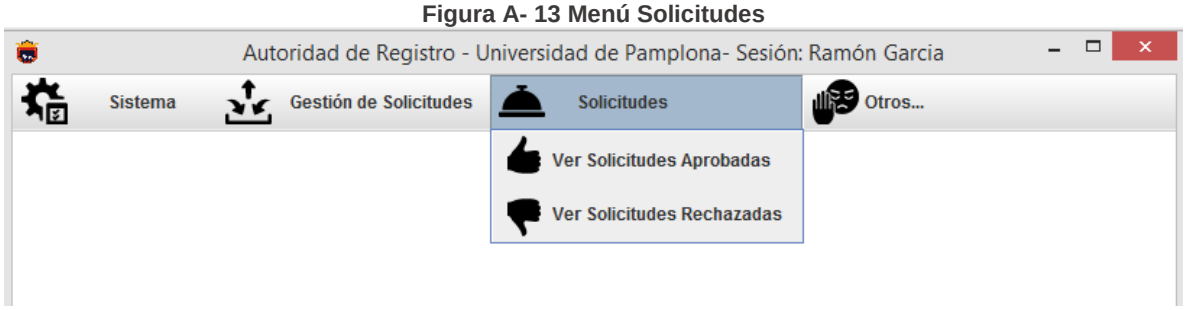
Figura A- 12 Solicitudes Emitidas



- **Solicitudes**

El menú desplegable, Solicitudes; que permite ver las Solicitudes Aprobadas y las rechazadas. Ver Figura A- 13.

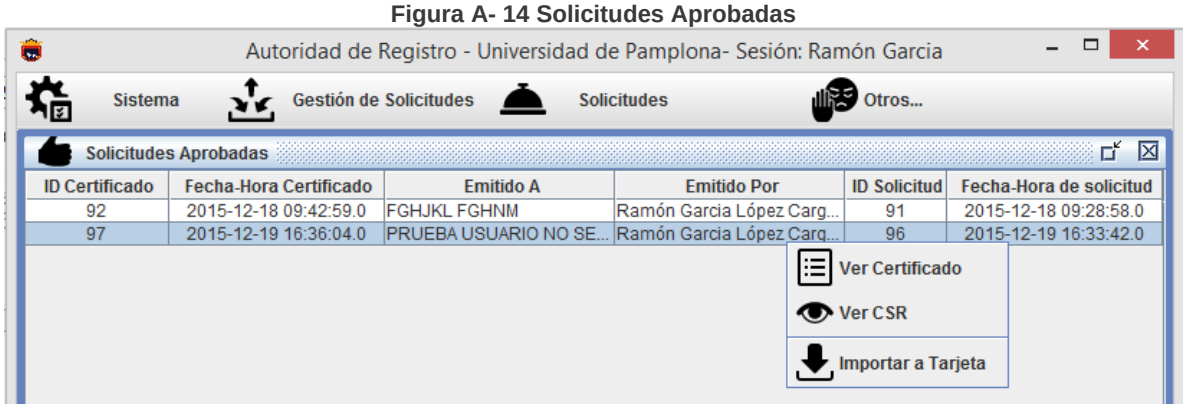
Figura A- 13 Menú Solicitudes



- **Solicitudes Aprobadas**

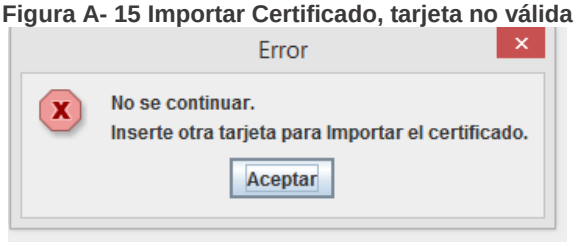
Cuando se ingresa a la primera de estas opciones, se listaran todos las solicitudes que han sido aprobados y al igual que en pasos anteriores, se podrá visualizar un menú contextual clickeando con el botón derecho del ratón sobre cada uno de estos. Esto despliega el menú contextual donde podrá verse nuevamente la solicitud, el certificado (si este está firmado) y una nueva opción que permite importar a una tarjeta inteligente el certificado de la base de datos. Ver Figura A- 14.

Figura A- 14 Solicitudes Aprobadas



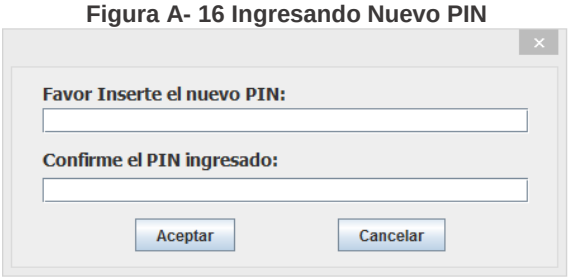
Para importar el certificado a la tarjeta, solo se debe dar clic en *Importar a Tarjeta,* Si no se introduce una tarjeta válida para la importación, el proceso lógicamente fallará, y deberá reanudarlo cuando haya ingresado una tarjeta válida. Ver Figura A- 15.

Figura A- 15 Importar Certificado, tarjeta no válida



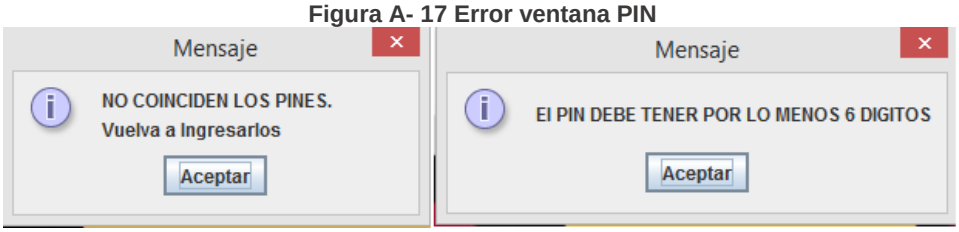
Una vez se importa con éxito el certificado, se despliega una ventana donde podrá configurar el PIN a la tarjeta inteligente. Ver Figura A- 16.

Figura A- 16 Ingresando Nuevo PIN



El PIN a ingresar debe ser iguales y además debe se ser mayor a seis dígitos, de lo contrario se visualizará el error según corresponda. Ver Figura A- 17.

Figura A- 17 Error ventana PIN



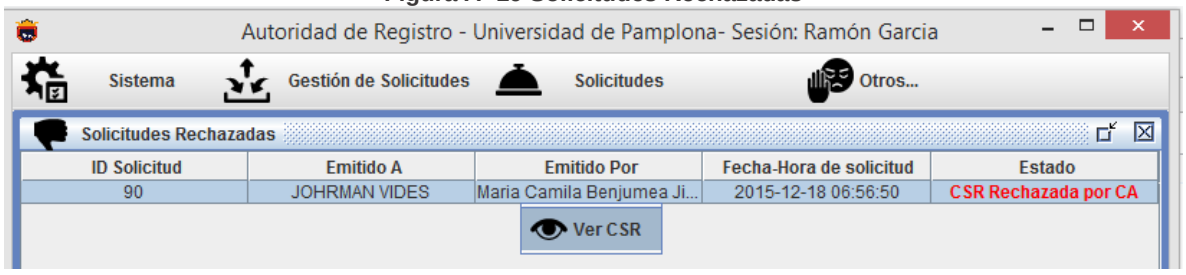
Pero si se cumplen las condiciones, el PIN será establecido a la tarjeta inteligente avisando por medio del mensaje. Ver Figura A- 18.

Figura A- 18 PIN establecido

- **Solicitudes Rechazadas**

Por ultimo cuando seleccionamos la opción Solicitudes rechazadas, del menú solicitudes, aparecerá una lista de los CSR que han sido rechazados por la Autoridad de Certificación, lista de la cual solamente podremos realizar una sola acción, al dar click con el botón derecho del ratón, podremos simplemente seleccionar la opción de ver la información del CSR y nada más. Ver Figura A- 19.

Figura A- 19 Solicitudes Rechazadas



A.2 Autoridad de Certificación

A.2.1 Requisitos Mínimos del Sistema

Sistema Operativo: Microsoft Windows XP SP3
Software requerido: JAVA JRE 7.1.0
Procesador: 1GHz o más rápido
Memoria RAM: 2GB
Disco Duro: 2 GB libres
Conectividad: Tarjeta Ethernet 10Mbps y puerto USB2.0

A.2.2 Instalación

Hardware requerido:

- CryptoMate ACOS5-CTM-R

PASO1: Instalar la versión mas reciente de JAVA JRE

PASO2: Se debe instalar el lector de la Tarjeta inteligente, comprobar por el Adminitrados de Dispositivos de Windows hasta que lo reconozca.

PASO3: Alojar en la ubicación System32 el complemento del estándar PKCS11 para Windows pkcs11wrapper.dll. NOTA: Existen la versión para sustemas de 32bits y 64bits.

PASO4: Ejecutar el paquete autoejecutable AutoridadCertificacion.jar en cualquier carpeta de acceso rápido (por ejemplo, el escritorio). NOTA: el Autoejecutable contiene los demás fichero necesario, como openSSL y bibliotecas de Aplicación DLL necesarias

A.2.3 Funcionamiento

A esta Autoridad, se ingresa sin necesidad de usuario ni contraseña, pero se debe poseer un dispositivo que ha sido grada especialmente para tal propósito. En una CryptoMate que hace las veces de Lector y tarjeta inteligente incorporados en un solo hardware contiene el par de claves de la Autoridad de Certificación con la cual se crearán y se revocarán los certificado, es importante que este dispositivo se almacene muy bien, pues caer en manos ajenas implicaría tener en riesgo la clave privada de la Autoridad de Certificacion.

Al abrir la interfaz, se pueden encontrar con una ventana bastante sobria, pero por debajo se están ejecutando varios procesos entre esos un hilo que revisa constantemente la BD_USU en busca de nuevas solicitudes de revocación de certificados para anularlos en el acto, y varios procesos que detectan la conectividad con la CryptoMate.

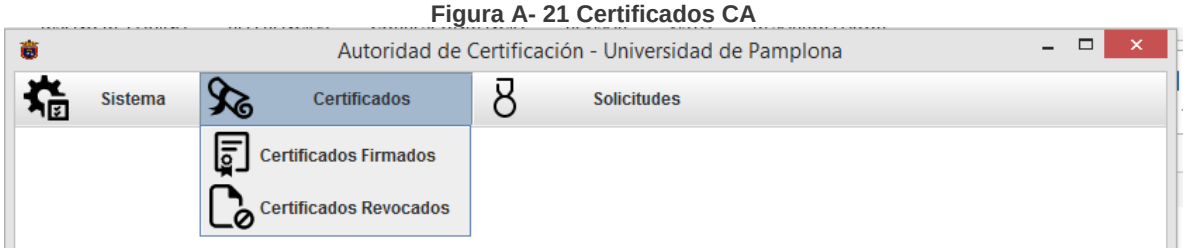
Algunos menús en la parte superior, el primero de estos "**Sistema**". Posee dos opciones, *salir* pasa cerrar la interfaz, y *acerca de*, para información. Ver Figura A-20.

Figura A- 20 Ventana Principal Autoridad de Certificación

- **Certificados**

El menú "Certificados" despliega un par de opciones referentes a los mismos, certificados firmados y certificados revocados. Ver

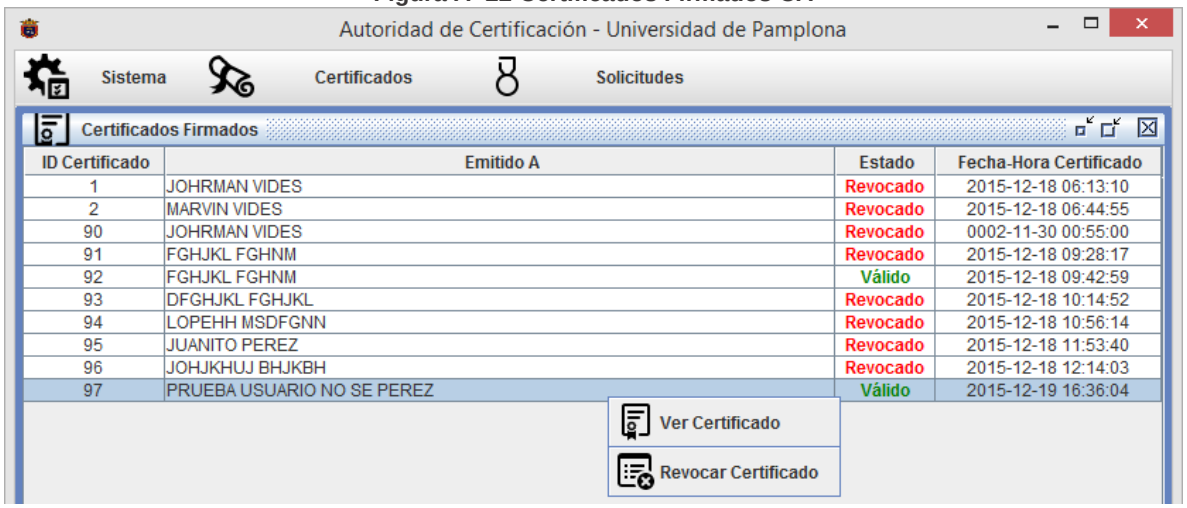
Figura A- 21 Certificados CA



- **Certificados Firmados**

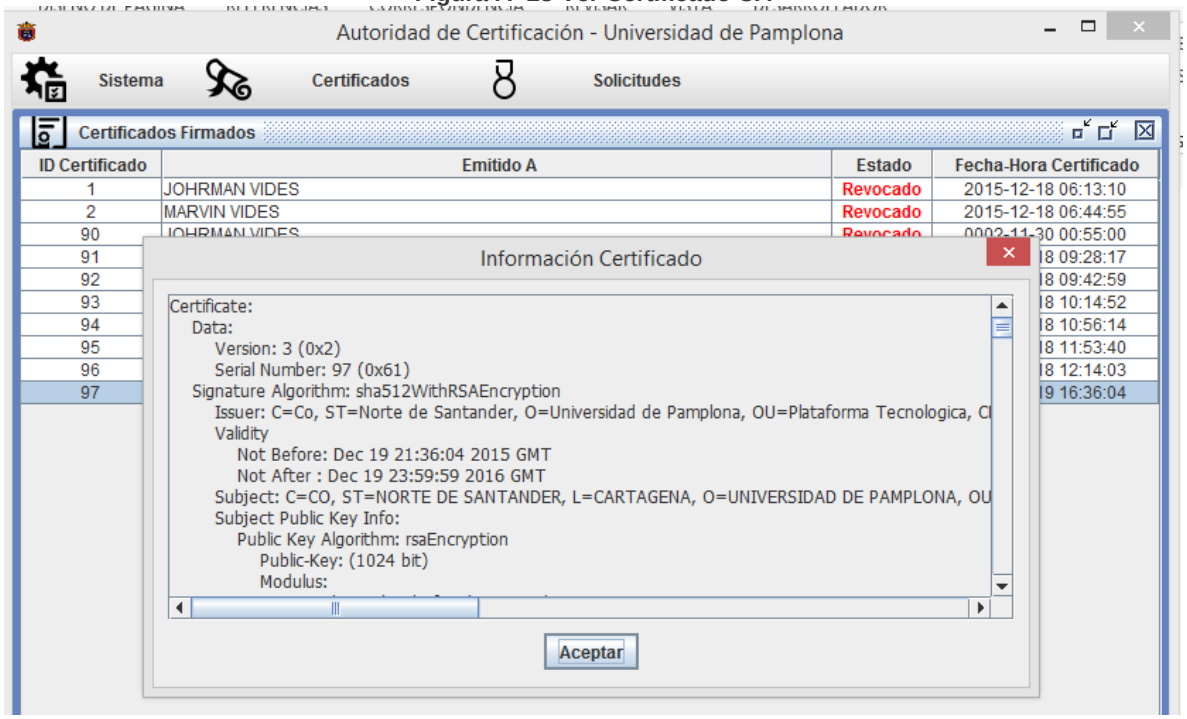
Cuando ingrese a la primera opción (Certificados Firmados), podrá ver un listado de todos los certificados que han sido firmados, al igual que su estatus de valido/revocado y finalmente la fecha y hora de generación del mismo. Ver Figura A- 22

Figura A- 22 Certificados Firmados CA



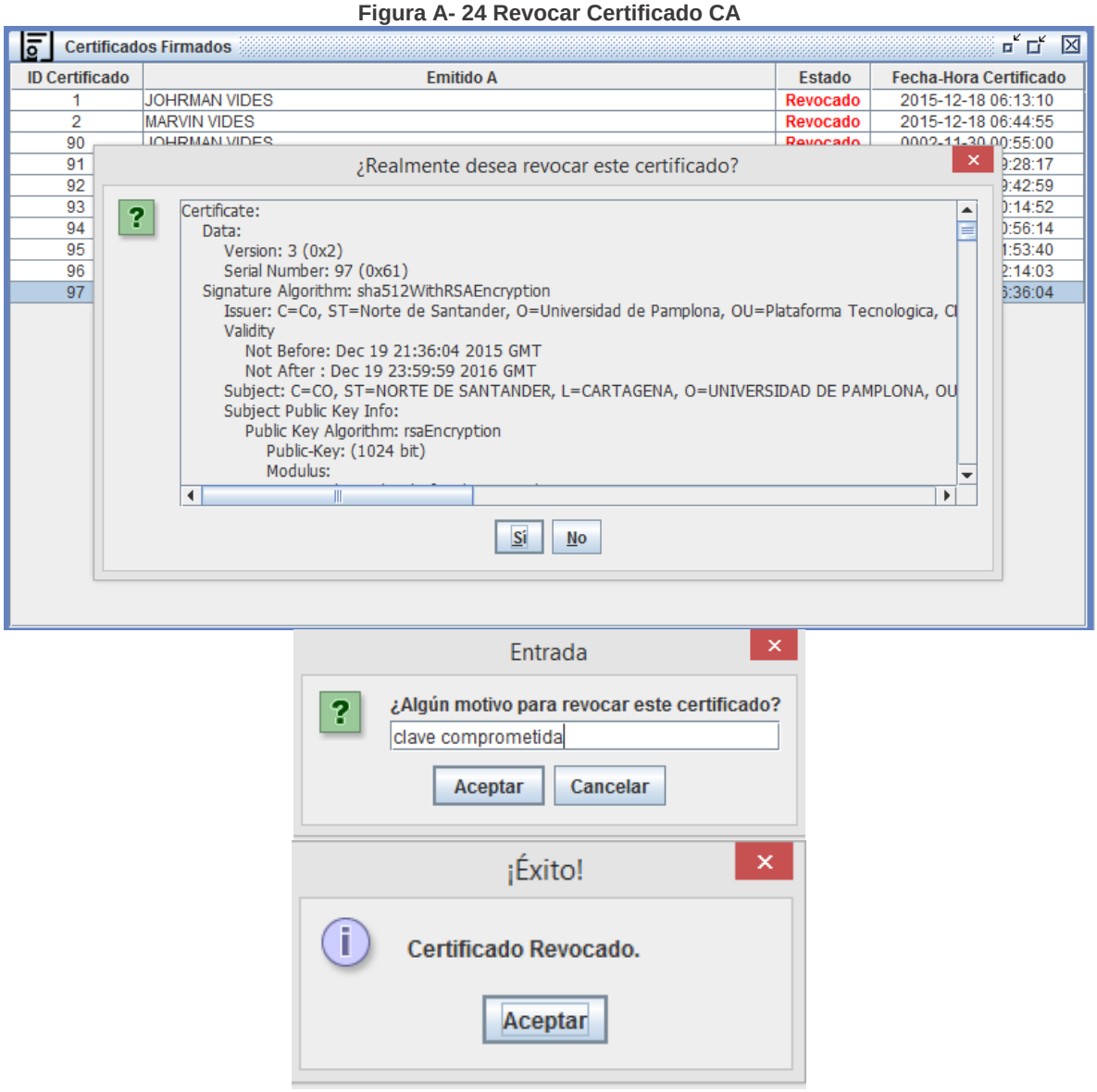
También está la opción de cargar un pequeño menú al clickear con el botón derecho del ratón sobre alguno de ellos, al hacer esto aparecerán dos opciones adicionales, *ver certificado* y *revocar certificado*. Al escoger la primera de estas opciones tendrá acceso a información acerca del certificado

Figura A- 23 Ver Certificado CA



De elegir la segunda podrá revocarlo pero antes deberá escribir en un cuadro de texto la razón por la cual desea realizar el proceso revocatorio de dicho certificado. Ver Figura A- 24.

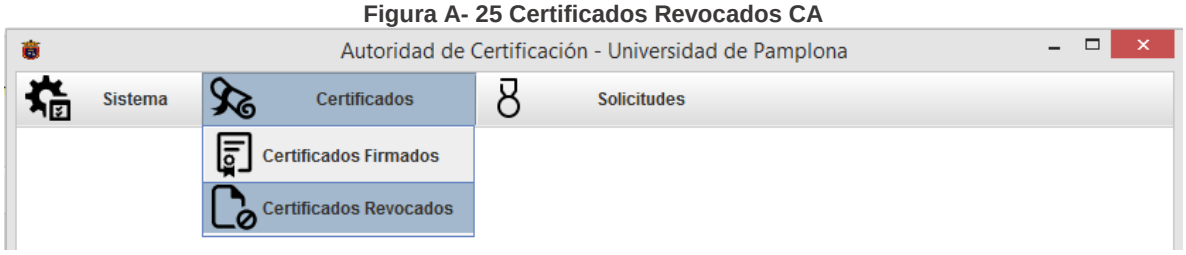
Figura A- 24 Revocar Certificado CA



- **Certificados Revocados**

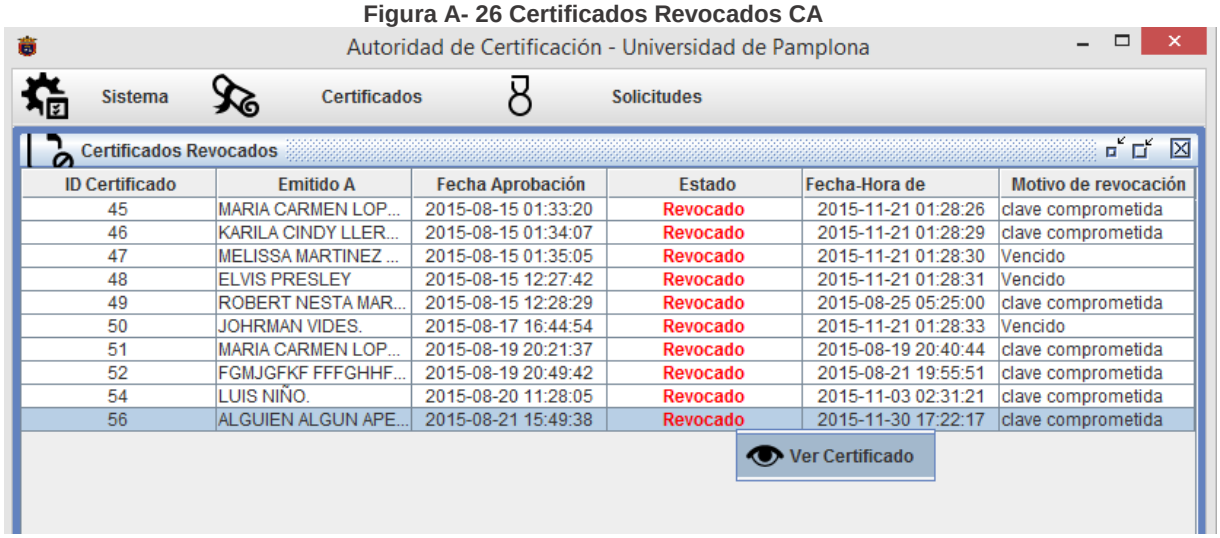
Ahora bien, si selecciona la opción de certificados revocados del menú “Certificados Revocados” (Ver Figura A- 25).

Figura A- 25 Certificados Revocados CA



Se listaran todos aquellos que han sido revocados y cuando oprima click derecho sobre cada uno de estos, solo le arrojara una opción, que es la de *ver certificado.* Ver Figura A- 26*.*

Figura A- 26 Certificados Revocados CA

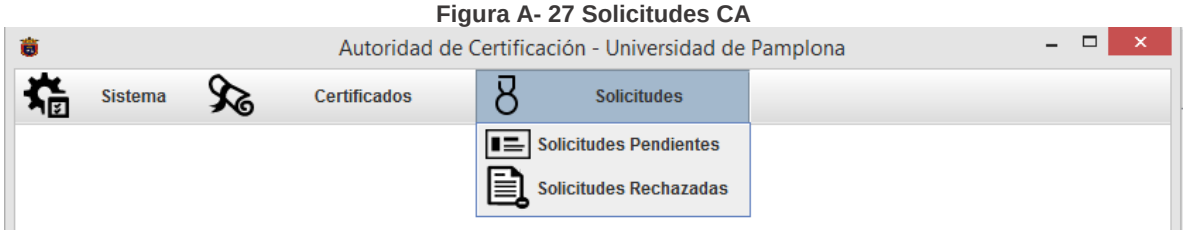


ID Certificado	Emitido A	Fecha Aprobación	Estado	Fecha-Hora de	Motivo de revocación
45	MARIA CARMEN LOP...	2015-08-15 01:33:20	Revocado	2015-11-21 01:28:26	clave comprometida
46	KARILA CINDY LLER...	2015-08-15 01:34:07	Revocado	2015-11-21 01:28:29	clave comprometida
47	MELISSA MARTINEZ ...	2015-08-15 01:35:05	Revocado	2015-11-21 01:28:30	Vencido
48	ELVIS PRESLEY	2015-08-15 12:27:42	Revocado	2015-11-21 01:28:31	Vencido
49	ROBERT NESTA MAR...	2015-08-15 12:28:29	Revocado	2015-08-25 05:25:00	clave comprometida
50	JOHRMAN VIDES.	2015-08-17 16:44:54	Revocado	2015-11-21 01:28:33	Vencido
51	MARIA CARMEN LOP...	2015-08-19 20:21:37	Revocado	2015-08-19 20:40:44	clave comprometida
52	FGMJGFKF FFFGHHF...	2015-08-19 20:49:42	Revocado	2015-08-21 19:55:51	clave comprometida
54	LUIS NIÑO.	2015-08-20 11:28:05	Revocado	2015-11-03 02:31:21	clave comprometida
56	ALGUIEN ALGUN APE...	2015-08-21 15:49:38	Revocado	2015-11-30 17:22:17	clave comprometida

- **Solicitudes**

Ahora al situarse en el menú *solicitudes*, pueden observarse dos opciones; *solicitudes pendientes* y *solicitudes rechazadas*, estas provienen del administrador de registros, para realizar las firmas de las Solicitudes de cada estudiante. Ver Figura A- 27.

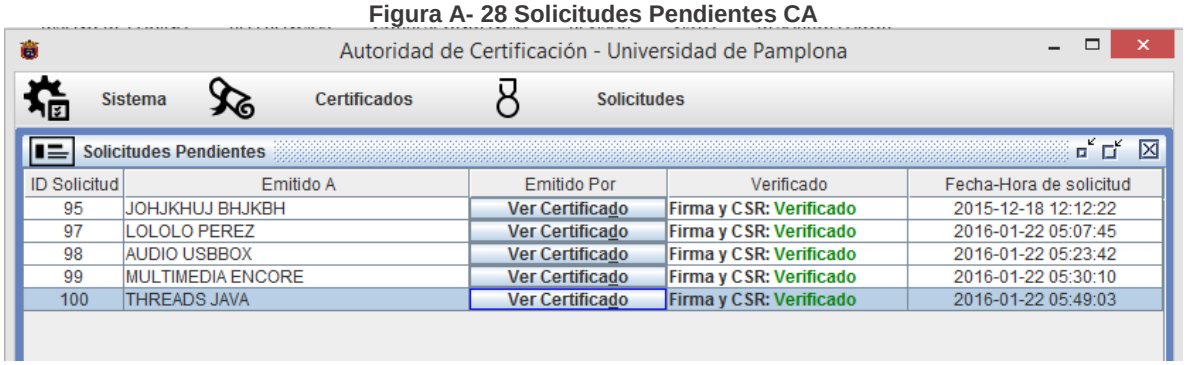
Figura A- 27 Solicitudes CA



- **Solicitudes Pendientes**

Al ingresar a *solicitudes pendientes*, se listan cada una de las solicitudes enviadas por el administrador de registros. Aquí puede verse información como hacia quien es emitido el CSR, quien emite el mismo, el estatus de verificación (si el estado es no verificado querrá decir, que la solicitud CSR recibida no es de confianza) y la fecha/hora, de este. Ver Figura A- 28.

Figura A- 28 Solicitudes Pendientes CA

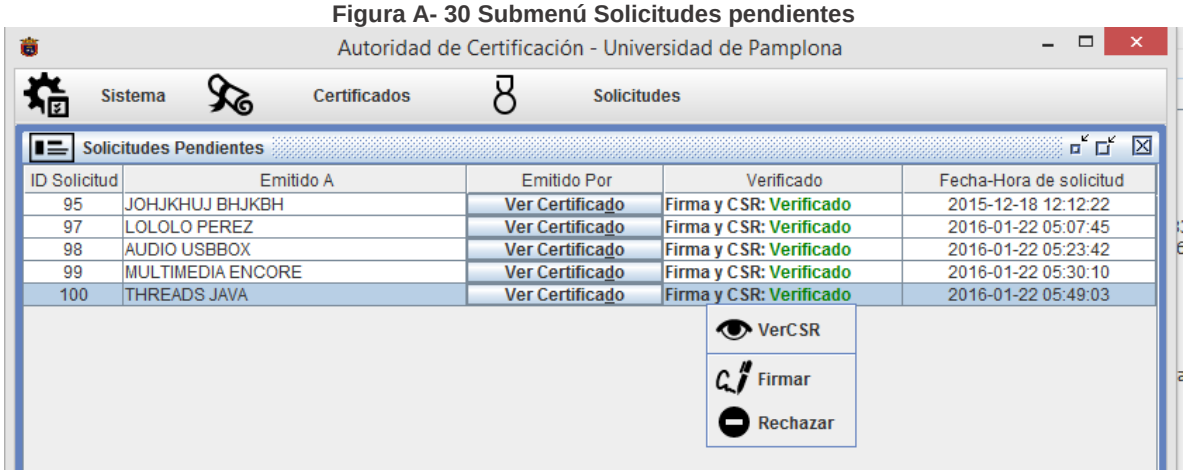


Al seleccionar el botón *ver certificado*, se puede apreciar información detallada acerca del certificado del operario que ha enviado la solicitud. ver Figura A- 29.

Figura A- 29 Ver Certificado Operario

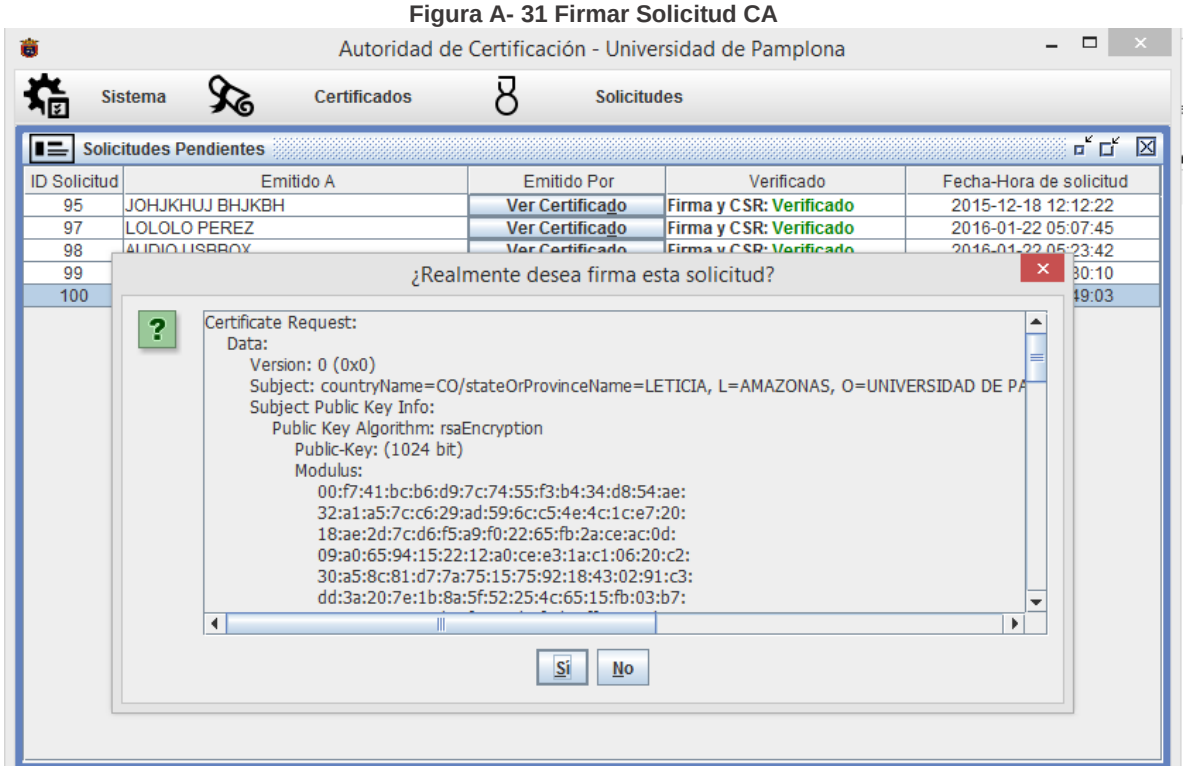
Al igual que en procesos anteriores, se puede desplegar un menú contextual al oprimir click derecho del ratón sobre cada una de las opciones listadas; al hacer esto podrán aparecer tres opciones: *ver CSR, firmar, rechazar.*

Figura A- 30 Submenú Solicitudes pendientes



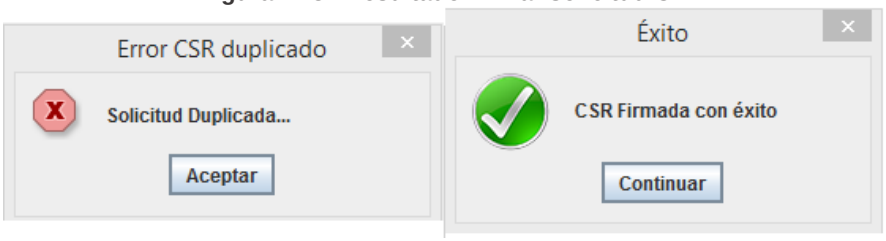
Al seleccionar cada una de ellas, se despliega una ventana con información acerca del CSR, si la opción es *firmar*, se realiza una pregunta de confirmación de firma. Ver Figura A- 31.

Figura A- 31 Firmar Solicitud CA



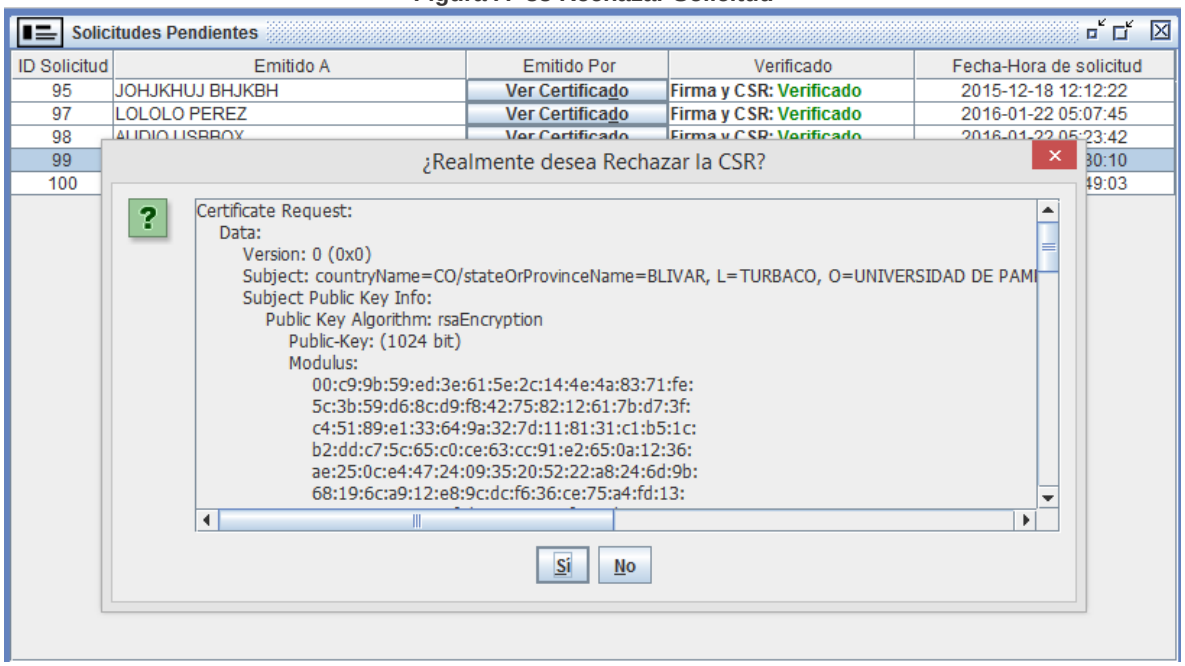
Sí la CSR no está duplicada se firmará correctamente y automáticamente publicará el certificado en BD_REPO y en BD_USU, para ello, se emitairá un mensja dando el resultado correspondiente. Ver Figura A- 32.

Figura A- 32 Resultado Firmar solicitud CA



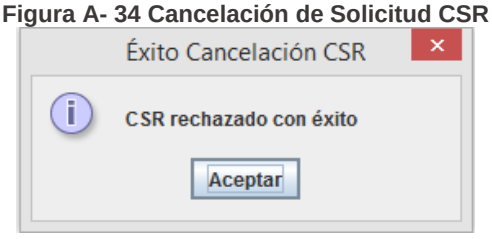
De otro modo, si se selecciona la opción *rechazar* también emitirá el mensaje de confirmación. Ver Figura A- 33.

Figura A- 33 Rechazar Solicitud



En caso de dar con el botón *SI* la solicitud CSR se rechazará. Ver Figura A- 34. Cabe resaltar que solo se podrán firmar, aquellos CSR que ya hayan sido verificados, es decir, que el estatus de verificación deberá aparecer escrito en letras de color verde. La opción de rechazar, solo se utilizaría en el caso de que la información del estudiante no sea válida.

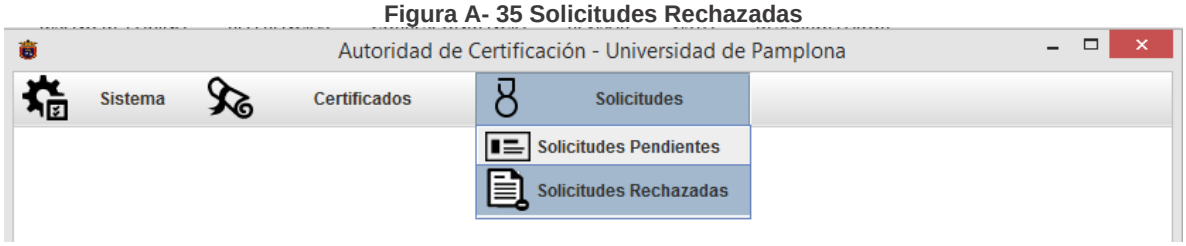
Figura A- 34 Cancelación de Solicitud CSR



- **Solicitudes Rechazadas**

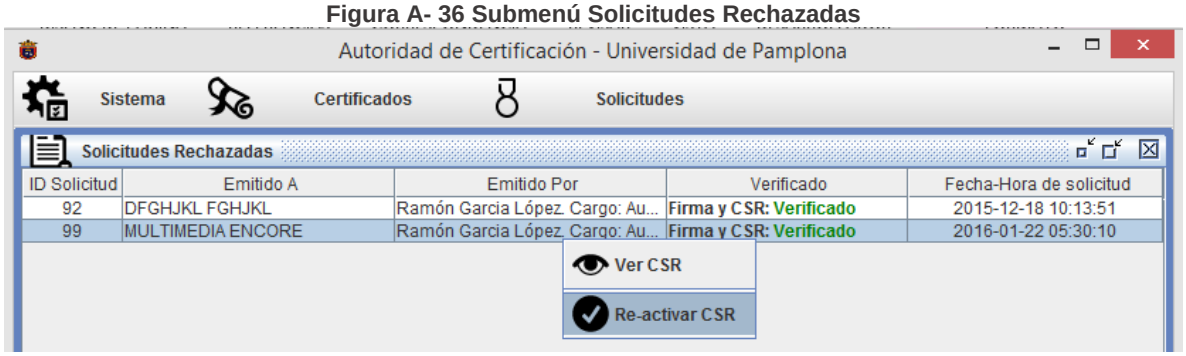
Ahora, finalmente en la opción del sub-menú, *solicitudes rechazadas*. Ver Figura A- 35

Figura A- 35 Solicitudes Rechazadas



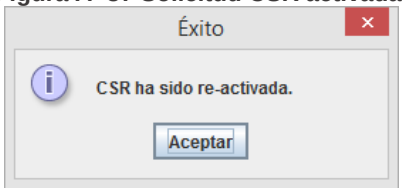
Se observa el listado de todas aquellas que han sido rechazadas previamente. Como se ha venido desarrollando con anterioridad se puede acceder a un menú contextual al hacer click con el botón derecho del ratón sobre los elementos de la lista, esto permitirá nuevamente seleccionar algunas opciones: *ver CSR* o *reactivar CSR* (para devolverla al menú anterior de solicitudes pendientes). Ver Figura A- 36.

Figura A- 36 Submenú Solicitudes Rechazadas



Donde através de una cuadro de dialogo se confirmará si realmente desea activar la solicitud CSR, lo caul se avisirá con el siguiente mensaje. Ver Figura A- 37.

Figura A- 37 Solicitud CSR activada.



- **Piloto Automático**

Una característica interesante de esta interfaz de Autoridad de Certificación, es el hecho que cuenta con un botón que permite activar o desactivar un piloto automático, es decir, va a permitir desarrollar la secuencia de operaciones aquí descritas en el orden de llegada de las solicitudes por parte de la Autoridad de Registro, validando la información de los estudiantes y rechazando/firmando los certificados, según sea el caso, de esta manera se agiliza el procedimiento para la gestión de los certificados digitales en el Aplicativo de registro. La figura muestra cuando el Piloto Automático está activo y la Figura muestra cuando cuando esta desactivado.

Figura A- 38 Piloto Automático Activo

Figura A- 39 Piloto Automático desactivado

A.3 Autoridad de Validación

A.3.1 Requisitos Mínimos del Sistema

Sistema Operativo: Microsoft Windows XP SP3
Software requerido: JAVA JRE 7.1.0
Procesador: 1GHz o más rápido
Memoria RAM: 2GB
Disco Duro: 2 GB libres
Conectividad: Tarjeta Ethernet 10Mbps

A.3.2 Instalación

PASO1: Instalar la versión mas reciente de JAVA JRE**.**

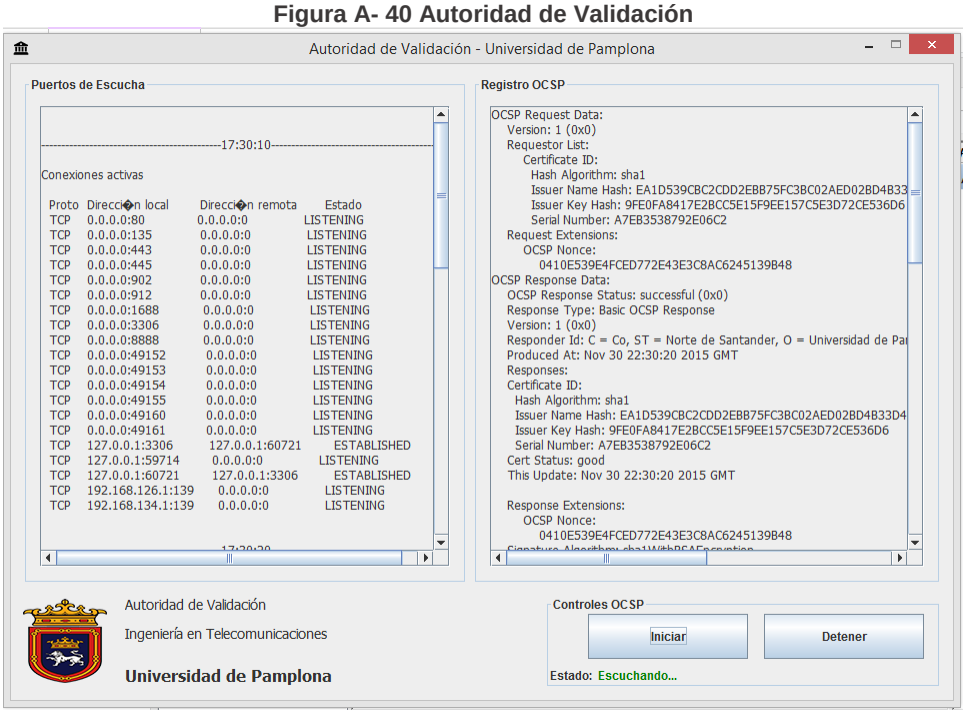
PASO4: Ejecutar el paquete autoejecutable AutoridadValidacion.jar en cualquier carpeta de acceso rápido (por ejemplo, el escritorio). NOTA: el Autoejecutable contiene los demás fichero necesario, como openSSL y bibliotecas de Aplicación DLL.

A.3.3 Funcionamiento

En este capítulo realizamos el proceso de validación con una interfaz bastante sencilla, donde se observa la activación de los puertos de interés y donde se listan los puertos activos del servidor

Al presionar el botón iniciar, se listan los puertos activos del servidor en la parte izquierda de la ventana, y a la derecha todos los certificados que han sido validados. Ver Figura A- 40. Esta interfaz solo cuenta con dos botones, iniciar la validación y detener la validación, por ello es importante que en todo momento este encendida para que pueda responder a la las solicitudes OCSP. Esta aplicación esta tipificado con un OCSP *responder.*

Figura A- 40 Autoridad de Validación



A.4 Aplicación Web

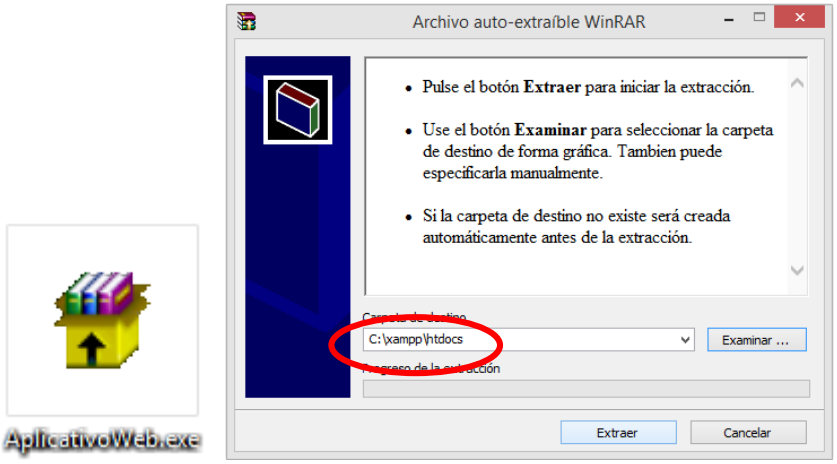
A.4.1 Requisitos Mínimos del Sistema

Sistema Operativo: Microsoft Windows XP SP3
Software requerido: Servidor Web Apache e interprete PHP
Procesador: 2GHz o más rápido
Memoria RAM: 2GB
Disco Duro: 4 GB libres
Conectividad: Tarjeta Ethernet 100Mbps

A.4.2 Instalación

Dedido que este aplicativo está compuesto por muchos archivos, se ha preparado un autoextraíble de WinRar, el cual tiene todo los archivos necesario para la instalación del aplicativo web, sólo el administrador de indicar la ruta donde este alojado el servidor Web.

Figura A- 41 Extraible para la Instalación Aplicativo Web

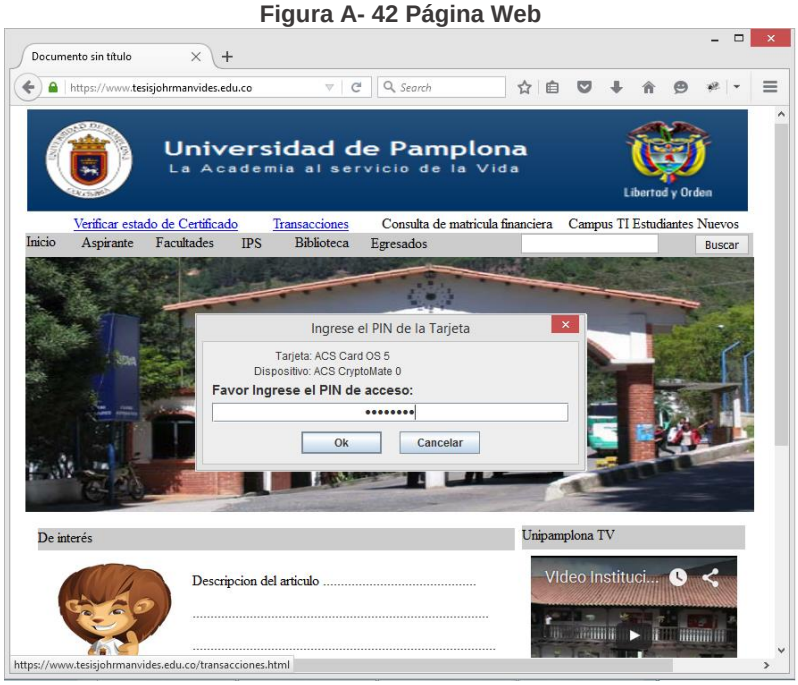


A.4.3 Funcionamiento

En este capítulo se presenta el portal web de toda la aplicación, para ingresar a este se debe digitar la dirección URL https://www.tesisjohrmanvides.edu.co, esta página web tiene dos enlaces de interés, que son los que realmente tienen gran relevancia en este proyecto, "**transacciones**" y "**Verificar estado de certificado**". En este último se recibe el certificado del estudiante asignado a su tarjeta y verificarlo con la Autoridad de Validación. Cuando se desee ingresar en

transacciones, inicia la aplicación de acceso de usuarios (estudiantes), en la cual se solicita el código PIN que ha sido cargado en la tarjeta inteligente. Ver Figura A-42.

Figura A- 42 Página Web



Una vez que se ha ingresado el PIN de manera correcta, se despliega una ventana en la que se le exige la selección de un certificado digital, y de esta manera poder terminar la fase de logueo de usuario.

Figura A- 43 Selección Certificado

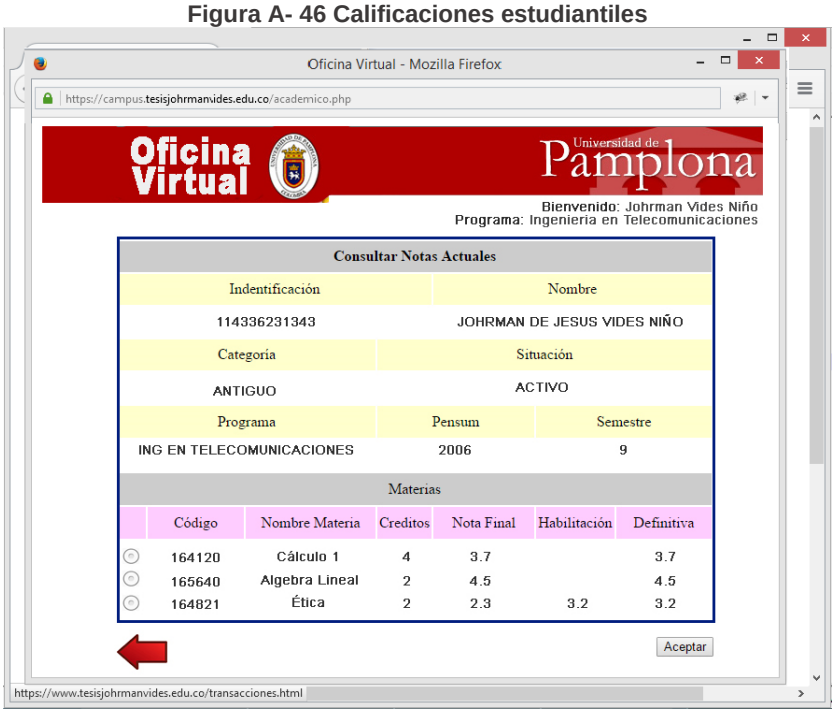
Realizada la selección, en la sub-ventana detalles del certificado, se mostrara al usuario la información general del CSR, a quien fue expedido, quien lo emite entre otros. Si ese certificado es correcto entonces al pulsar el botón “OK” saldrá un nuevo mensaje, informando que el logueo ha sido exitoso. Ver Figura A- 44.

Figura A- 44 Logueo exitoso

Una vez logueados se accede de manera automática al vortal, que no es más que otra página web, con una interfaz más amena, que cuenta con algunos indicadores en su parte izquierda, y tres opciones principales en el espacio de Gestión: Hoja de vida, Materias, Calificaciones. (Ver Figura A- 45) Al ingresar a cada una de estas, el estudiante podrá tener acceso a su información académica (Ver Figura A- 46) habiendo usado una método de autentición fuerte sin temor a que su información confidencial pueda ser capturada en el camino por que esta cifrada la comunicación.

Figura A- 45 Oficina Virtual

Figura A- 46 Calificaciones estudiantiles



Anexo B.

B CÓDIGO FUENTE

B.1 Aplicación de Registro de Usuarios

B.1.1 Autoridad de Registro

- AutoridadRegistro.java

```
package autoridadregistro;
import java.applet.AppletContext;
import java.io.IOException;
import java.net.URL;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
public class AutoridadRegistro {
    public static void main(String[] args) throws IOException {
        IniciarSesion objeto = new IniciarSesion();
        objeto.setLocationRelativeTo(null);
        objeto.setTitle("Autoridad de Registro - Universidad de Pamplona");
        objeto.setVisible(true);
    }
}
```

- IniciarSesion.java

```
package autoridadregistro;
import iaik.pkcs.pkcs11.Module;
import iaik.pkcs.pkcs11.Session;
import iaik.pkcs.pkcs11.Slot;
import iaik.pkcs.pkcs11.Token;
import iaik.pkcs.pkcs11.TokenException;
import iaik.pkcs.pkcs11.TokenInfo;
import iaik.utils.Util;
import iaik.pkcs.pkcs11.objects.ByteArrayAttribute;
import iaik.pkcs.pkcs11.objects.Data;
import iaik.pkcs.pkcs11.wrapper.PKCS11Exception;
import iaik.pkcs.pkcs11.objects.Object;
import iaik.pkcs.pkcs11.objects.X509PublicKeyCertificate;
import java.awt.Color;
import java.awt.event.*;
import java.awt.image.BufferedImage;
import java.io.BufferedWriter;
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.net.URI;
import java.nio.file.Paths;
import java.security.Security;
import java.security.cert.CertPath;
import java.security.cert.CertPathValidator;
import java.security.cert.CertPathValidatorException;
import java.security.cert.CertStore;
import java.security.cert.CertStoreParameters;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.CollectionCertStoreParameters;
import java.security.cert.PKIXCertPathValidatorResult;
import java.security.cert.PKIXParameters;
import java.security.cert.TrustAnchor;
import java.security.cert.X509Certificate;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Arrays;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Set;
import java.util.Vector;
import java.util.concurrent.TimeUnit;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.*;

public class IniciarSesion extends javax.swing.JFrame {
Module pkcs11Modulo;
Slot[] slot General;
int CTRLgeneral=1;
int auxCTRL;
Statement sta;
Icon FotoUsuario;
String Datos Usuario;
String DireccionLibreria;

    private String buscarLiberiaTarjInt(){
        String lineSeparator = System.getProperty("path.separator");
        String libraryPath = System.getProperty("java.library.path");
        for (String dir : libraryPath.split(lineSeparator)) {
            File f = new File(dir + "/" + "acospkcs11.dll");
             if (f.exists()){
                return f.getAbsolutePath();
            }
        }
        msjEstado.setForeground(Color.RED);
        msjEstado.setText("NO se encuentra un dispositivo compatible instalado.");
        return null;
    }
```

```
    private class Hilo extends Thread{
        String DireccionLibreria;
        private Hilo(int a, String c){auxCTRL=a;DireccionLibreria=c;}
        public void run(){
             Boolean CtrlInicioModulo=true, CtrlPrograma=true;
            while(auxCTRL==1){
                if(CTRLgeneral==1){
                    if(CtrlPrograma){
                    try {
                        if(CtrlInicioModulo){
                            try{
                                pkcs11Modulo =
Module.getInstance(DireccionLibreria);
                                pkcs11Modulo.initialize(null);
                                CtrlInicioModulo=false;
                            }catch(TokenException ex){
                                msjEstado.setForeground(Color.RED);
                                msjEstado.setText("Dispositivo no conectado.");
                                des_habilitarCamposInicioSesion(0);
                                CtrlInicioModulo=true;
                            }
                        }
                        if(!CtrlInicioModulo){
                            Slot[] slots =
pkcs11Modulo.getSlotList(Module.SlotRequirement.TOKEN_PRESENT);
                            if(slots.length!=0){
                                CtrlPrograma=false;
                                CtrlInicioModulo=false;
                                msjEstado.setForeground(Color.BLACK);
                                msjEstado.setText("Tarjeta Conectada");
                                des_habilitarCamposInicioSesion(1);
                                slot_General=slots;
                            } else{
                                msjEstado.setForeground(Color.RED);
                                msjEstado.setText("Tarjeta no insertada.");
                                des_habilitarCamposInicioSesion(0);
                                CtrlInicioModulo=true;
                                pkcs11Modulo.finalize();
                                System.gc();
                            }
                        }
                        Thread.sleep(4500);
                    }  catch (InterruptedException ex) {
                        JOptionPane.showMessageDialog(null, ex.getMessage(),"Error",
JOptionPane.ERROR_MESSAGE);
                    } catch (TokenException ex) {
                         JOptionPane.showMessageDialog(null,
ex.getMessage(),"Error", JOptionPane.ERROR_MESSAGE);
                    }
                    catch (Throwable ex) {
                        JOptionPane.showMessageDialog(null, ex.getMessage(),"Error",
JOptionPane.ERROR_MESSAGE);
                    }
```

```
                }else{
                        try {
                            Thread.sleep(2500);
                            Slot[] slots1 =
pkcs11Modulo.getSlotList(Module.SlotRequirement.TOKEN_PRESENT);
                            if(slots1.length==0){
                                CtrlPrograma=true;
                            }
                        } catch (TokenException ex) {
                            JOptionPane.showMessageDialog(null,
ex.getMessage(),"Error", JOptionPane.ERROR_MESSAGE);
                        } catch (InterruptedException ex) {

Logger.getLogger(IniciarSesion.class.getName()).log(Level.SEVERE, null, ex);
                        }
                    }
                }
            }
        }
    }

    private void des_habilitarCamposInicioSesion(int i) {
            if(i==0){
                txtContrase.setEnabled(false);
                botonAceptar.setEnabled(false);
                txtContrase.setText("");
                fotoUsuario.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/imagenes/usuariodesconocido.png")));
            }else if(i==1){
                txtContrase.requestFocus();
                txtContrase.setEnabled(true);
                botonAceptar.setEnabled(true);
            }else if(i==2){
                CTRLgeneral=2;
                txtContrase.setEnabled(false);
                botonAceptar.setEnabled(false);
                botonCancelar.setEnabled(false);
            }else if(i==3){
                CTRLgeneral=1;
                txtContrase.setText("");
                txtContrase.requestFocus();
                txtContrase.setEnabled(true);
                botonAceptar.setEnabled(true);
                botonCancelar.setEnabled(true);
            }
        }

    public IniciarSesion() throws IOException {
        initComponents();
        msjEstado.setHorizontalAlignment(SwingConstants.CENTER);
        msjEstado.setVerticalAlignment(SwingConstants.CENTER);
        DireccionLibreria = buscarLiberiaTarjInt();
        if(DireccionLibreria!=null){
```

```
            System.out.println("1");
            new Hilo(1,DireccionLibreria).start();
        }
        setLocation(400, 100);
        setIconImage(new ImageIcon(getClass().getResource
("/imagenes/icono.png")).getImage());
        this.setResizable(false);
    }

    public void InicioSesion(){
    des_habilitarCamposInicioSesion(2);
    msjEstado.setForeground(Color.BLACK); msjEstado.setText("Comprobando
Contraseña...");
    SwingWorker sw = new SwingWorker() {
        @Override
        protected Object doInBackground() throws Exception {
            barraProgreso.setIndeterminate(true);
            Thread.sleep(5000); // Here you should establish connection
            return null;
        }

        @Override
        public void done(){
            barraProgreso.setIndeterminate(false);
            barraProgreso.setValue(100); // 100%
        }
    };
    sw.execute();
    Session sesion=null;
    boolean ctrlInicioSesion=false;
    try {
        Slot slotSeleccionado = slot_General[0];
        Token token = slotSeleccionado.getToken();
        sesion = token.openSession(Token.SessionType.SERIAL_SESSION,
Token.SessionReadWriteBehavior.RW_SESSION, null, null);
        sesion.login(Session.UserType.USER, txtContrase.getPassword());
        ctrlInicioSesion=true;
    } catch (TokenException ex) {
        msjEstado.setForeground(Color.RED);
        msjEstado.setText("Contraseña no Válida");
        ctrlInicioSesion=false;
        des_habilitarCamposInicioSesion(3);
    }
    if(ctrlInicioSesion){
        msjEstado.setForeground(Color.BLACK); msjEstado.setText("Validando
Tarjeta...");
        Vector<String> IDObjectosNoRepe=validarIntegridadTarjeta(sesion);
        if(IDObjectosNoRepe.size()!=1){
            msjEstado.setForeground(Color.RED); msjEstado.setText("Tarjeta no
Válida");
            des_habilitarCamposInicioSesion(3);
            try {
                des_habilitarCamposInicioSesion(3);
```

```
                sesion.closeSession();
                sesion=null;
                } catch (Exception ex) {
Logger.getLogger(IniciarSesion.class.getName()).log(Level.SEVERE, null, ex);
                }
        }
        else{
            msjEstado.setForeground(Color.BLACK); msjEstado.setText("Leyendo
Datos...");
            leerDatos(sesion);
            msjEstado.setForeground(Color.BLACK); msjEstado.setText("Verificando
Certificados...");
            if(verificarOCSP(IDObjectosNoRepe.get(0),sesion)!=1){
                try {
                des_habilitarCamposInicioSesion(3);
                sesion.closeSession();
                sesion=null;
                } catch (Exception ex) {

Logger.getLogger(IniciarSesion.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
            else{
                msjEstado.setForeground(Color.BLACK); msjEstado.setText("Conectando
a la Base de Datos...");
                sta = (new conexionBD("autoridad_certificacion_bd")).stm;
                if(sta==null){
                    msjEstado.setForeground(Color.RED);
                    msjEstado.setText("No se puede establecer conexión la base de
datos.");
                    des_habilitarCamposInicioSesion(3);
                    try {
                    des habilitarCamposInicioSesion(3);
                    sesion.closeSession();
                    sesion=null;
                    } catch (Exception ex) {
Logger.getLogger(IniciarSesion.class.getName()).log(Level.SEVERE, null, ex);
                    }
                }
                else{
                try {
                    sta.close();
                    msjEstado.setForeground(Color.BLACK); msjEstado.setText("Inicio
de sesión completado con éxito");
                    auxCTRL=2;
                    sesion.closeSession();
                    sesion=null;
                    pkcs11Modulo.finalize();
                    pkcs11Modulo=null;
                    System.gc();
                    JDialog d = (new JOptionPane("Inicio de sesión completado\nPulse
para continuar")).createDialog((JFrame)null, "Inicio Sesión");
                    d.setLocation(543,460);
```

```
                    d.setVisible(true);
                    VentanaPPAL ppal=new
VentanaPPAL(FotoUsuario,Datos_Usuario,DireccionLibreria,txtContrase.getPassword());
                    ppal.setEnabled(true);
                    ppal.setVisible(true);
                    this.dispose();
                } catch (SQLException ex) {
Logger.getLogger(IniciarSesion.class.getName()).log(Level.SEVERE, null, ex);
                } catch (TokenException ex) {
Logger.getLogger(IniciarSesion.class.getName()).log(Level.SEVERE, null, ex);
                } catch (Throwable ex) {
Logger.getLogger(IniciarSesion.class.getName()).log(Level.SEVERE, null, ex);
                }
                }
            }
        }
    }
    }

    private byte[] hexadecimal_A_ByteArray(String s) {
    int len = s.length();
    byte[] data = new byte[len / 2];
    for (int i = 0; i < len; i += 2) {
        data[i / 2] = (byte) ((Character.digit(s.charAt(i), 16) << 4)
                             + Character.digit(s.charAt(i+1), 16));
    }
    return data;
}

    private int verificarOCSP(String IDObjeto, Session sesion){
        String ctrl_Resul="1";
        try {
        X509PublicKeyCertificate certificadoPlantilla = new
X509PublicKeyCertificate();
certificadoPlantilla.getId().setByteArrayValue(hexadecimal_A_ByteArray(IDObjeto.subs
tring(0, 8)));
        sesion.findObjectsInit(certificadoPlantilla);
        Object[] Certificado = sesion.findObjects(1);
        sesion.findObjectsFinal();
        byte[] CertificadoPEM = ((X509PublicKeyCertificate)
Certificado[0]).getValue().getByteArrayValue();
        String Certicado = new String(Util.Base64Encode(CertificadoPEM));
        Certicado="-----BEGIN CERTIFICATE-----\n"+Certicado+"\n-----END CERTIFICATE-
----";
        File CertificadoArchivo = new File("thetextfile.txt");
        FileWriter fw = new FileWriter(CertificadoArchivo);
        fw.write(Certicado);
        fw.close();
        CertificadoArchivo.deleteOnExit();
        CertPath cp = null;
        Vector certs = new Vector();
        URI servidorOCSP = null;
        certs.add(Archivo_A_Certificado(CertificadoArchivo.getAbsolutePath()));
```

```
        servidorOCSP = new URI("http://ocsp.tesisjohrmanvides.edu.co:8888");
        CertificateFactory cf = CertificateFactory.getInstance("X509");
        cp = (CertPath)cf.generateCertPath(certs);
        X509Certificate certRaizCA =
Archivo_A_Certificado((Paths.get("")).toAbsolutePath().toString()+"/src/certificados
/cacert.pem");
        TrustAnchor ta = new TrustAnchor(certRaizCA, null);
        Set RutaDeVerificacion = new HashSet();
        RutaDeVerificacion.add(ta);
        Set datosCert = new HashSet();
        X509Certificate certOCSP =
Archivo_A_Certificado((Paths.get("")).toAbsolutePath().toString()+"/src/certificados
/ocsp.pem");
        datosCert.add(certOCSP);
       CertStoreParameters parameCertifi = new
CollectionCertStoreParameters(datosCert);
       CertStore storeCertifiCA = CertStore.getInstance("Collection", parameCertifi);
        PKIXParameters params = null;
       params = new PKIXParameters(RutaDeVerificacion);
       params.addCertStore(storeCertifiCA);
        Security.setProperty("ocsp.enable", "true");
        if (servidorOCSP != null) {
            Security.setProperty("ocsp.responderURL",
"http://ocsp.tesisjohrmanvides.edu.co:8888");
            Security.setProperty("ocsp.responderCertSubjectName",
            certOCSP.getSubjectX500Principal().getName());
       }
        CertPathValidator consultaOCSP = CertPathValidator.getInstance("PKIX");
        PKIXCertPathValidatorResult consultaOCSP_Resul  =
(PKIXCertPathValidatorResult) consultaOCSP.validate(cp, params);
       X509Certificate certifi_Resp_OCSP =
(X509Certificate)consultaOCSP_Resul.getTrustAnchor().getTrustedCert();
        if (certifi Resp OCSP != null) {
            ctrl_Resul=null;
        }
       } catch (CertPathValidatorException e) {
            ctrl_Resul = e.toString();
            msjEstado.setForeground(Color.RED);
            if(ctrl_Resul.contains("Certificate has been revoked")){
                msjEstado.setText("Certificado Revocado.");
            } else if(ctrl_Resul.contains("status is unknown")){
                msjEstado.setText("Certificado Desconocido");
            } else if(ctrl Resul.contains("status due to network error")){
                msjEstado.setText("No se pudo establecer conexión con el servidor
OCSP");
            }else{msjEstado.setText("Error al comprobar certificado.");}
            return 0;
       } catch(Exception e) {
            JOptionPane.showMessageDialog(null, e.getMessage(),"Error",
JOptionPane.ERROR_MESSAGE);
            return 0;
        }
        return 1;
```

```
    }

    private X509Certificate Archivo_A_Certificado(String path) {
        X509Certificate cert = null;
        try {
            File certFile = new File(path);
            if (!certFile.canRead()){throw new IOException(); }
            FileInputStream fis = new FileInputStream(path);
            CertificateFactory cf = CertificateFactory.getInstance("X509");
            cert = (X509Certificate)cf.generateCertificate(fis);

        } catch(IOException | CertificateException e) {
            msjEstado.setForeground(Color.RED);
            msjEstado.setText("Error en la lectura de certificados.");
        }
        return cert;
    }

    private Vector<String> validarIntegridadTarjeta(Session sesion){
    try {
        sesion.findObjectsInit(null);
        Object[] objects1 = sesion.findObjects(1);
        String[] IDObjetos = new String[15];
        int i;
        int contIDObjetos = 0;
        int PosCert =0;
        while (objects1.length > 0) {
            Object object = objects1[0];
            String aux = object.toString();
            String[] parts = aux.split("\n");
            if (parts[0].contains("Private"))
{IDObjetos[contIDObjetos]=parts[6].substring(6); contIDObjetos++;}
            if (parts[0].contains("Public")) {IDObjetos[contIDObjetos] =
parts[6].substring(6); contIDObjetos++;}
            if (parts[0].contains("Data")) {IDObjetos[contIDObjetos] =
parts[6].substring(24); contIDObjetos++;}
            if (parts[0].contains("Certificate")) {
                if(parts[8].contains("hex")){IDObjetos[contIDObjetos]=
parts[8].substring(12); PosCert=8; contIDObjetos++;} else {IDObjetos[contIDObjetos]=
parts[12].substring(12); PosCert = 12; contIDObjetos++;}
            }
            objects1 = sesion.findObjects(1);
        }
        contIDObjetos = 0;
        int cont;
        Vector<String> IDnoRepe = new Vector<String>();
        while (IDObjetos[contIDObjetos]!=null){
            cont = 0;
            for (i=contIDObjetos; IDObjetos[i]!=null ; i ++ ){
                if (IDObjetos[i].equalsIgnoreCase(IDObjetos[contIDObjetos])){
cont++; }
            }
            if(cont==5){IDnoRepe.add(IDObjetos[contIDObjetos]);}
```

```
            contIDObjetos++;
        }
        return IDnoRepe;
    } catch (TokenException ex) { }
    return null;
    }

    private void leerDatos(Session sesion){
    try {
        Data fotoObjectoEstructura = new Data();
        fotoObjectoEstructura.getLabel().setCharArrayValue("foto".toCharArray());

fotoObjectoEstructura.getApplication().setCharArrayValue("foto".toCharArray());
        sesion.findObjectsInit(fotoObjectoEstructura);
        Object[] objectos = sesion.findObjects(1);
        Data fotoObjeto = (Data) objectos[0];
        byte[] fotoByte = fotoObjeto.getValue().getByteArrayValue();
        FotoUsuario = new ImageIcon(arrayByte_A_Imagen(fotoByte));
        fotoUsuario.setIcon(FotoUsuario);
        //Leyendo Datos Usuario
        fotoObjectoEstructura = new Data();

fotoObjectoEstructura.getLabel().setCharArrayValue("infoAdicional".toCharArray());

fotoObjectoEstructura.getApplication().setCharArrayValue("infoAdicional".toCharArray
());
        sesion.findObjectsInit(fotoObjectoEstructura);
        objectos = sesion.findObjects(1);
        fotoObjeto = (Data) objectos[0];
        fotoByte = fotoObjeto.getValue().getByteArrayValue();
        Datos_Usuario = new String(fotoByte);
    } catch (TokenException ex) {
        msjEstado.setForeground(Color.RED);
        msjEstado.setText("Foto no cargó correctamente");
    }
    }

    public static BufferedImage  arrayByte_A_Imagen(byte[] bytes){
        BufferedImage imagen=null;
        try {
            InputStream inputStream = new ByteArrayInputStream(bytes);
            imagen = ImageIO.read(inputStream);
        } catch (IOException ex) { }
        return imagen;
}

    private void botonAceptarActionPerformed(java.awt.event.ActionEvent evt) {
    InicioSesion();
    }

    private void botonCancelarActionPerformed(java.awt.event.ActionEvent evt) {
System.exit(0);
    }
```

```
    private void txtContraseKeyReleased(java.awt.event.KeyEvent evt) {
        char car=(char) evt.getKeyCode();
        if(car==KeyEvent.VK_ENTER){
            InicioSesion();
        }
        if(car==KeyEvent.VK_DELETE || car==KeyEvent.VK_BACK_SPACE){
            txtContrase.setText("");
        }
    }
    private javax.swing.JProgressBar barraProgreso;
    private javax.swing.JButton botonAceptar;
    private javax.swing.JButton botonCancelar;
    private javax.swing.JLabel campoContrasena;
    private javax.swing.JLabel fotoUsuario;
    private javax.swing.JLabel msjEstado;
    private javax.swing.JPasswordField txtContrase;
}
```

- VentanaPPAL.java

```
package autoridadregistro;
import iaik.pkcs.pkcs11.Module;
import iaik.pkcs.pkcs11.Session;
import iaik.pkcs.pkcs11.Slot;
import iaik.pkcs.pkcs11.TokenException;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.beans.PropertyVetoException;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
/**
 *
 * @author Johrman
 */
public class VentanaPPAL extends javax.swing.JFrame {
Boolean ctrlWhileHiloCompruebaTarjeta=true;
String DireccionLibreria=null;
Module pkcs11Modulo=null;
char[] claveOperario = null;
Boolean ctrlPrograma=true;
String[] datosOperario;
conexionBD BD_solicitudes, BD_certi_apro;
    public VentanaPPAL(Icon FotoUsuario, String DatosUsuario, String Dir, char[]
cla) {
```

```
    try {
        initComponents();
        BD_solicitudes = new conexionBD("autoridad_certificacion_bd");
        BD_certi_apro=  new conexionBD("autoridad_certificacion_bd");
        DireccionLibreria=Dir;
        pkcs11Modulo = Module.getInstance(DireccionLibreria);
        pkcs11Modulo.initialize(null);
        claveOperario = cla;
        jfoto.setIcon(FotoUsuario);
        String[] partes = DatosUsuario.split("\n");
        JOptionPane.showMessageDialog(null,"Datos
Usuario:\n"+Arrays.toString(partes));
        datosOperario = new String[3];
        datosOperario[0]=partes[0].substring(8);
        datosOperario[1]=partes[1].substring(11);
        datosOperario[2]=partes[2].substring(7);
        partes=null;
        jnombre.setText(datosOperario[0]);
        japellido.setText(datosOperario[1]);
        jcargo.setText(datosOperario[2]);
        this.setTitle("Autoridad de Registro - Universidad de Pamplona- Sesión:
"+(datosOperario[0].split(" "))[0]+" "+(datosOperario[1].split(" "))[0]);
        this.setIconImage(new ImageIcon(getClass().getResource
("/imagenes/icono.png")).getImage());
        (new Reloj()).start();
        (new HiloCompruebaTarjeta()).start();
        this.setLocationRelativeTo(null);
    } catch (IOException ex) {
        Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null, ex);
    } catch (TokenException ex) {
        Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null, ex);
    }
    }

    private void CerrarSesion() {
        try {
            ctrlWhileHiloCompruebaTarjeta=false;
            pkcs11Modulo.finalize();
            System.gc();
            IniciarSesion InicioSesion = new IniciarSesion();
            InicioSesion.setEnabled(true);
            InicioSesion.setVisible(true);
            this.dispose();
        } catch (Throwable ex) {
            Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

    private class HiloCompruebaTarjeta extends Thread{

        public void run(){
            while(ctrlWhileHiloCompruebaTarjeta){
```

```
                try {
                    if(ctrlPrograma){
                            Slot[] slots =
pkcs11Modulo.getSlotList(Module.SlotRequirement.TOKEN_PRESENT);
                            if(slots.length==0){
                                ctrlPrograma=false;
                                CerrarSesion();
                            }
                    }
                    Thread.sleep(8000);
                } catch (Exception ex) {
                    Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE,
null, ex);
                } catch (Throwable ex) {
                    Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE,
null, ex);
                }
            }
        }
    }

    private class Reloj extends Thread{
        Date horaFechaInicio = new Date();
        public void run(){
            int Min=0, Hora=0, Seg=-1;
            while(true){
                try {
                    Date horaFecha_Actual = new Date();
                    SimpleDateFormat f = new SimpleDateFormat("HH:mm:ss
dd/MMMM/yyyy");
                    horaFecha.setText(f.format(horaFecha_Actual));
                    Seg++;
                    if(Seg==60){Seg=0;Min++;if(Min==60){Min=0;Hora++;}}
                    tSesion.setText(Hora+":"+Min+":"+Seg);
                    Thread.sleep(1000);
                } catch (InterruptedException ex) { }
            }
        }
    }

    private void jAcercadeActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jCerrarSesionActionPerformed(java.awt.event.ActionEvent evt) {
    CerrarSesion();
    }

    private void jSalirActionPerformed(java.awt.event.ActionEvent evt) {
        int opcion=JOptionPane.showConfirmDialog(null,"Realmente desea
Salir","Confirmar",JOptionPane.YES_NO_OPTION);
        if(opcion==JOptionPane.YES_NO_OPTION){System.exit(0);}
    }
```

```
    private void jcsrEmitidosActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        ArrayList<String[]> todas_Solicitudes = BD_solicitudes.listarSolicitudes();
        todas_Solicitudes = BD_certi_apro.listarSolicitudes(todas_Solicitudes);
        if(todas_Solicitudes.size()==0){JOptionPane.showMessageDialog(null,"No se
encuentra CSR disponibles.");}
        else{
            espacioTrabajo.removeAll();
            espacioTrabajo.repaint();
            Solicitudes_Emitidas csremitidos= new
Solicitudes_Emitidas(todas_Solicitudes,BD_solicitudes);
            espacioTrabajo.add(csremitidos);
            csremitidos.setEnabled(true);
            csremitidos.setVisible(true);
            csremitidos.setMaximum(true);
        }
    } catch (PropertyVetoException ex) {
        Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null, ex);
    }
    }

    private void jGenerarSolicitudActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        espacioTrabajo.removeAll();
        espacioTrabajo.repaint();

        GenerarSolicitud gencsr= new GenerarSolicitud(this);
        espacioTrabajo.add(gencsr);
        gencsr.setEnabled(true);
        gencsr.setVisible(true);
        gencsr.setMaximum(true);
    } catch (PropertyVetoException ex) {
        Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null, ex);
    }
    }

    private void jCancelarSolicitudActionPerformed(java.awt.event.ActionEvent evt) {

        ArrayList<String[]> SolicitudesPendientes =
BD_solicitudes.SolicitudesPendientes();
        if(SolicitudesPendientes.size()==0){JOptionPane.showMessageDialog(null,"No
se encuentra nuevas solictudes");}
           else{
            try {
                espacioTrabajo.removeAll();
                espacioTrabajo.repaint();
                CancelarSolicitud CancelarSolicitud = new
CancelarSolicitud(SolicitudesPendientes,BD_solicitudes);
                espacioTrabajo.add(CancelarSolicitud);
                CancelarSolicitud.setEnabled(true);
                CancelarSolicitud.setVisible(true);
                CancelarSolicitud.setMaximum(true);
```

```
            } catch (PropertyVetoException ex) {
                Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE,
null, ex);
            }
           }
    }

    private void jCSRaprobadosActionPerformed(java.awt.event.ActionEvent evt) {
        ArrayList<conexionBD.campos> camposCerti = BD_certi_apro.obtener_CERT();
        if(camposCerti.size()==0){JOptionPane.showMessageDialog(null,"No se
encuentra solicitudes aprobadas");}
        else{
            camposCerti = BD_certi_apro.obtenerInfoCSR(camposCerti);
            espacioTrabajo.removeAll();
            espacioTrabajo.repaint();
            CSR_Aprobados csraprobados = new
CSR_Aprobados(camposCerti,BD_certi_apro,this);
            espacioTrabajo.add(csraprobados);
            csraprobados.setEnabled(true);
            csraprobados.setVisible(true);
        try {
            csraprobados.setMaximum(true);
        } catch (PropertyVetoException ex) {
            Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null,
ex);
        }
        }
    }

    private void jCSRrechazadosActionPerformed(java.awt.event.ActionEvent evt) {

           try {
        ArrayList<String[]> todas Solicitudes = BD solicitudes.listarSolicitudes();
        todas_Solicitudes = BD_certi_apro.listarSolicitudes(todas_Solicitudes);
        if(todas_Solicitudes.size()==0){JOptionPane.showMessageDialog(null,"No se
encuentra CSR disponibles.");}
        else{
            espacioTrabajo.removeAll();
            espacioTrabajo.repaint();
            CSR_Rechazados csrrechazados = new
CSR_Rechazados(todas_Solicitudes,BD_solicitudes);
            espacioTrabajo.add(csrrechazados);
            csrrechazados.setEnabled(true);
            csrrechazados.setVisible(true);
            csrrechazados.setMaximum(true);
        }
    } catch (PropertyVetoException ex) {
        Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null, ex);
    }
    }
    private javax.swing.JLabel IconoUP;
    private javax.swing.JLabel campoAutoridad;
    private javax.swing.JLabel campoHF;
```

```
    private javax.swing.JLabel campoIng;
    private javax.swing.JLabel campoTsesion;
    private javax.swing.JLabel campoUnivers;
    private javax.swing.JDesktopPane espacioTrabajo;
    private javax.swing.JLabel horaFecha;
    private javax.swing.JMenuItem jAcercade;
    private javax.swing.JMenuItem jCSRaprobados;
    private javax.swing.JMenuItem jCSRrechazados;
    private javax.swing.JMenuItem jCancelarSolicitud;
    private javax.swing.JMenuItem jCerrarSesion;
    private javax.swing.JMenuItem jGenerarSolicitud;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jSalir;
    private javax.swing.JPopupMenu.Separator jSeparator1;
    private javax.swing.JLabel japellido;
    private javax.swing.JLabel jcargo;
    private javax.swing.JMenuItem jcsrEmitidos;
    private javax.swing.JLabel jfoto;
    private javax.swing.JLabel jnombre;
    private javax.swing.JMenu menuGestionCSR;
    private javax.swing.JMenu menuSistema;
    private javax.swing.JMenu menuSolictudes;
    private javax.swing.JLabel tSesion;
}
```

- Solicitudes_Aprobadas.java

```
package autoridadregistro;
import iaik.pkcs.pkcs11.Module;
import iaik.pkcs.pkcs11.Session;
import iaik.pkcs.pkcs11.Slot;
import iaik.pkcs.pkcs11.TokenException;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.beans.PropertyVetoException;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
/** * * @author Johrman */
public class VentanaPPAL extends javax.swing.JFrame {
Boolean ctrlWhileHiloCompruebaTarjeta=true;
String DireccionLibreria=null;
Module pkcs11Modulo=null;
char[] claveOperario = null;
Boolean ctrlPrograma=true;
String[] datosOperario;
```

```
conexionBD BD_solicitudes, BD_certi_apro;

public VentanaPPAL(Icon FotoUsuario, String DatosUsuario, String Dir, char[] cla) {
try {
initComponents();
BD_solicitudes = new conexionBD("autoridad_certificacion_bd");
BD_certi_apro=new conexionBD("autoridad_certificacion_bd");
DireccionLibreria=Dir;
pkcs11Modulo = Module.getInstance(DireccionLibreria);
pkcs11Modulo.initialize(null);
claveOperario = cla;
jfoto.setIcon(FotoUsuario);
String[] partes = DatosUsuario.split("\n");
JOptionPane.showMessageDialog(null,"Datos Usuario:\n"+Arrays.toString(partes));
datosOperario = new String[3];
datosOperario[0]=partes[0].substring(8);
datosOperario[1]=partes[1].substring(11);
datosOperario[2]=partes[2].substring(7);
partes=null;
jnombre.setText(datosOperario[0]);
japellido.setText(datosOperario[1]);
jcargo.setText(datosOperario[2]);
this.setTitle("Autoridad de Registro - Universidad de Pamplona- Sesión:
"+(datosOperario[0].split(" "))[0]+" "+(datosOperario[1].split(" "))[0]);
this.setIconImage(new ImageIcon(getClass().getResource
("/imagenes/icono.png")).getImage());
(new Reloj()).start();
(new HiloCompruebaTarjeta()).start();
this.setLocationRelativeTo(null);
} catch (IOException ex) {
Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null, ex);
} catch (TokenException ex) {
Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null, ex);
}
}
private void CerrarSesion() {
try {
ctrlWhileHiloCompruebaTarjeta=false;
pkcs11Modulo.finalize();
System.gc();
IniciarSesion InicioSesion = new IniciarSesion();
InicioSesion.setEnabled(true);
InicioSesion.setVisible(true);
this.dispose();
} catch (Throwable ex) {
Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null, ex);
}
}
private class HiloCompruebaTarjeta extends Thread{
public void run(){
while(ctrlWhileHiloCompruebaTarjeta){
try {
if(ctrlPrograma){
```

```
Slot[] slots = pkcs11Modulo.getSlotList(Module.SlotRequirement.TOKEN_PRESENT);
if(slots.length==0){
ctrlPrograma=false;
CerrarSesion();
}
}
Thread.sleep(8000);
} catch (Exception ex) {
Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null, ex);
} catch (Throwable ex) {
Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null, ex);
}
}
}
}
private class Reloj extends Thread{
Date horaFechaInicio = new Date();
public void run(){
int Min=0, Hora=0, Seg=-1;
while(true){
try {
Date horaFecha_Actual = new Date();
SimpleDateFormat f = new SimpleDateFormat("HH:mm:ss dd/MMMM/yyyy");
horaFecha.setText(f.format(horaFecha_Actual));
Seg++;
if(Seg==60){Seg=0;Min++;if(Min==60){Min=0;Hora++;}}
tSesion.setText(Hora+":"+Min+":"+Seg);
Thread.sleep(1000);
} catch (InterruptedException ex) { }
}
}
}
private void jAcercadeActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
}

private void jCerrarSesionActionPerformed(java.awt.event.ActionEvent evt) {
CerrarSesion();
}

private void jSalirActionPerformed(java.awt.event.ActionEvent evt) {
int opcion=JOptionPane.showConfirmDialog(null,"Realmente desea
Salir","Confirmar",JOptionPane.YES NO OPTION);
if(opcion==JOptionPane.YES_NO_OPTION){System.exit(0);}
}

private void jcsrEmitidosActionPerformed(java.awt.event.ActionEvent evt) {
try {
ArrayList<String[]> todas_Solicitudes = BD_solicitudes.listarSolicitudes();
todas_Solicitudes = BD_certi_apro.listarSolicitudes(todas_Solicitudes);
if(todas_Solicitudes.size()==0){JOptionPane.showMessageDialog(null,"No se encuentra
CSR disponibles.");}
else{
```

```
espacioTrabajo.removeAll();
espacioTrabajo.repaint();
Solicitudes_Emitidas csremitidos= new
Solicitudes_Emitidas(todas_Solicitudes,BD_solicitudes);
espacioTrabajo.add(csremitidos);
csremitidos.setEnabled(true);
csremitidos.setVisible(true);
csremitidos.setMaximum(true);
}
} catch (PropertyVetoException ex) {
Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null, ex);
}
}

private void jGenerarSolicitudActionPerformed(java.awt.event.ActionEvent evt) {
try {
espacioTrabajo.removeAll();
espacioTrabajo.repaint();

GenerarSolicitud gencsr= new GenerarSolicitud(this);
espacioTrabajo.add(gencsr);
gencsr.setEnabled(true);
gencsr.setVisible(true);
gencsr.setMaximum(true);
} catch (PropertyVetoException ex) {
Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null, ex);
}
}

private void jCancelarSolicitudActionPerformed(java.awt.event.ActionEvent evt) {

ArrayList<String[]> SolicitudesPendientes = BD_solicitudes.SolicitudesPendientes();
if(SolicitudesPendientes.size()==0){JOptionPane.showMessageDialog(null,"No se
encuentra nuevas solictudes");}
 else{
try {
espacioTrabajo.removeAll();
espacioTrabajo.repaint();
CancelarSolicitud CancelarSolicitud = new
CancelarSolicitud(SolicitudesPendientes,BD_solicitudes);
espacioTrabajo.add(CancelarSolicitud);
CancelarSolicitud.setEnabled(true);
CancelarSolicitud.setVisible(true);
CancelarSolicitud.setMaximum(true);
} catch (PropertyVetoException ex) {
Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null, ex);
}
 }
}

private void jCSRaprobadosActionPerformed(java.awt.event.ActionEvent evt) {
ArrayList<conexionBD.campos> camposCerti = BD_certi_apro.obtener_CERT();
```

```
if(camposCerti.size()==0){JOptionPane.showMessageDialog(null,"No se encuentra
solicitudes aprobadas");}
else{
camposCerti = BD_certi_apro.obtenerInfoCSR(camposCerti);
espacioTrabajo.removeAll();
espacioTrabajo.repaint();
CSR_Aprobados csraprobados = new CSR_Aprobados(camposCerti,BD_certi_apro,this);
espacioTrabajo.add(csraprobados);
csraprobados.setEnabled(true);
csraprobados.setVisible(true);
try {
csraprobados.setMaximum(true);
} catch (PropertyVetoException ex) {
Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null, ex);
}
}
}

private void jCSRrechazadosActionPerformed(java.awt.event.ActionEvent evt) {

 try {
ArrayList<String[]> todas_Solicitudes = BD_solicitudes.listarSolicitudes();
todas_Solicitudes = BD_certi_apro.listarSolicitudes(todas_Solicitudes);
if(todas_Solicitudes.size()==0){JOptionPane.showMessageDialog(null,"No se encuentra
CSR disponibles.");}
else{
espacioTrabajo.removeAll();
espacioTrabajo.repaint();
CSR_Rechazados csrrechazados = new CSR_Rechazados(todas_Solicitudes,BD_solicitudes);
espacioTrabajo.add(csrrechazados);
csrrechazados.setEnabled(true);
csrrechazados.setVisible(true);
csrrechazados.setMaximum(true);
}
} catch (PropertyVetoException ex) {
Logger.getLogger(VentanaPPAL.class.getName()).log(Level.SEVERE, null, ex);
}
}
private javax.swing.JLabel IconoUP;
private javax.swing.JLabel campoAutoridad;
private javax.swing.JLabel campoHF;
private javax.swing.JLabel campoIng;
private javax.swing.JLabel campoTsesion;
private javax.swing.JLabel campoUnivers;
private javax.swing.JDesktopPane espacioTrabajo;
private javax.swing.JLabel horaFecha;
private javax.swing.JMenuItem jAcercade;
private javax.swing.JMenuItem jCSRaprobados;
private javax.swing.JMenuItem jCSRrechazados;
private javax.swing.JMenuItem jCancelarSolicitud;
private javax.swing.JMenuItem jCerrarSesion;
private javax.swing.JMenuItem jGenerarSolicitud;
private javax.swing.JMenuBar jMenuBar1;
```

```
private javax.swing.JMenuItem jSalir;
private javax.swing.JPopupMenu.Separator jSeparator1;
private javax.swing.JLabel japellido;
private javax.swing.JLabel jcargo;
private javax.swing.JMenuItem jcsrEmitidos;
private javax.swing.JLabel jfoto;
private javax.swing.JLabel jnombre;
private javax.swing.JMenu menuGestionCSR;
private javax.swing.JMenu menuSistema;
private javax.swing.JMenu menuSolictudes;
private javax.swing.JLabel tSesion;
}
```

- Solicitudes_Emitidas.java

```
package autoridadregistro;
import java.awt.Color;
import java.awt.Point;
import java.awt.event.KeyEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.ListSelectionModel;
import javax.swing.SwingUtilities;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
/*** * @author Johrman */
public class Solicitudes_Emitidas extends javax.swing.JInternalFrame {
conexionBD BD_solicitudes;
 ArrayList<String[]> todas_Solicitudes;
 private DefaultTableModel modelo;
public Solicitudes_Emitidas(ArrayList<String[]> t_Sol,conexionBD BD_sol) {
initComponents();
todas_Solicitudes=t_Sol;
BD_solicitudes=BD_sol;
String data[][] ={};
String col[] = {"ID Solicitud","Emitido A","Emitido Por","Estado","Fecha-Hora de
solicitud","ID Certificado","Fecha-Hora Certificado"};
modelo = new DefaultTableModel(data,col) {
@Override
public boolean isCellEditable(int data, int col) {
return false;
} };
jTable1.setModel(modelo);
DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
centerRenderer.setHorizontalAlignment( JLabel.CENTER );
jTable1.getColumnModel().getColumn(0).setCellRenderer( centerRenderer );
```

```
jTable1.getColumnModel().getColumn(0).setMaxWidth(71);
jTable1.getColumnModel().getColumn(0).setMinWidth(70);
jTable1.getColumnModel().getColumn(3).setMinWidth(140);
jTable1.getColumnModel().getColumn(3).setMaxWidth(143);
jTable1.getColumnModel().getColumn(3).setCellRenderer( centerRenderer );
jTable1.getColumnModel().getColumn(4).setMaxWidth(143);
jTable1.getColumnModel().getColumn(4).setMinWidth(140);
jTable1.getColumnModel().getColumn(4).setCellRenderer( centerRenderer );
jTable1.getColumnModel().getColumn(5).setMaxWidth(85);
jTable1.getColumnModel().getColumn(5).setMinWidth(86);
jTable1.getColumnModel().getColumn(5).setCellRenderer( centerRenderer );
jTable1.getColumnModel().getColumn(6).setMaxWidth(143);
jTable1.getColumnModel().getColumn(6).setMinWidth(140);
jTable1.getColumnModel().getColumn(6).setCellRenderer( centerRenderer );
int cont = 0;
while(cont<todas_Solicitudes.size()){
modelo.insertRow(cont, new Object[]{});
modelo.setValueAt(todas_Solicitudes.get(cont)[0], cont, 0);
modelo.setValueAt(todas_Solicitudes.get(cont)[1], cont, 1);
if(todas_Solicitudes.get(cont)[2].length()>36){
modelo.setValueAt(todas_Solicitudes.get(cont)[2].substring(36), cont, 2);
}else{
modelo.setValueAt(todas_Solicitudes.get(cont)[2], cont, 2);
}
switch (Integer.parseInt(todas_Solicitudes.get(cont)[3])) {
case 1:modelo.setValueAt("No Firmado AC", cont, 3); break;
case 2:modelo.setValueAt("Firmado", cont, 3); break;
case 3:modelo.setValueAt("Solicitud Duplicada", cont, 3); break;
case 4:modelo.setValueAt("Falló Firma CSR", cont, 3); break;
case 5:modelo.setValueAt("CSR Rechazada por AC", cont, 3); break;
case 6:modelo.setValueAt("CSR Cancelado por AR", cont, 3); break;
default: modelo.setValueAt("NS/NR", cont, 3); break;
}
modelo.setValueAt(todas_Solicitudes.get(cont)[4], cont, 4);
modelo.setValueAt(todas_Solicitudes.get(cont)[5], cont, 5);
modelo.setValueAt(todas_Solicitudes.get(cont)[6], cont, 6);
cont++;
}
jTable1.addMouseListener(new MouseAdapter() {
public void mousePressed(MouseEvent e) {
if ( SwingUtilities.isRightMouseButton(e)) {
Point p = e.getPoint();
int rowNumber = jTable1.rowAtPoint( p );
ListSelectionModel modelo = jTable1.getSelectionModel();
modelo.setSelectionInterval( rowNumber, rowNumber );
}
if (e.getClickCount()==2){
JOptionPane.showMessageDialog(null,MostrarCSR(),"Información
CSR",JOptionPane.PLAIN_MESSAGE);
}
}
}
);
```

```
}
private void jVerCSRActionPerformed(java.awt.event.ActionEvent evt) {
JOptionPane.showMessageDialog(null,MostrarCSR(),"Information
CSR",JOptionPane.PLAIN_MESSAGE);
}
private void jVerCertiActionPerformed(java.awt.event.ActionEvent evt) {
if((jTable1.getModel().getValueAt(jTable1.getSelectedRow(),
5).toString()).contains("-") ||
!(jTable1.getModel().getValueAt(jTable1.getSelectedRow(),
3).toString()).contains("Firmado"))){
JOptionPane.showMessageDialog(null,"La Solicitud no se puede realizar.","Error al
mostrar Certificado",JOptionPane.ERROR_MESSAGE);
}
else{
try {
File miDir = new File (".");
FileOutputStream fos = new
FileOutputStream(miDir.getCanonicalPath()+"\\newreq.pem");
fos.write(BD_solicitudes.obt_CERT(jTable1.getModel().getValueAt(jTable1.getSelectedR
ow(), 5).toString()));
fos.close();
Runtime cmd = Runtime.getRuntime();
Process process= cmd.exec("openssl x509 -in "+ miDir.getCanonicalPath()
+"\\newreq.pem -text");
BufferedReader read = new BufferedReader(new
InputStreamReader(process.getInputStream()));
String line, Salida="";
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}
Salida=Salida+"\n";
Salida=Salida+"-----------------------------------------------------------------
Estructura ASN.1-----------------------------------------------------------------\n";
process= cmd.exec("openssl asn1parse -i -in
"+miDir.getCanonicalPath()+"\\newreq.pem");
read = new BufferedReader(new InputStreamReader(process.getInputStream()));
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}
jPanel1 = new javax.swing.JPanel();
jScrollPane1 = new javax.swing.JScrollPane();
jTextArea1 = new javax.swing.JTextArea();
jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jScrollPane1.setViewportView(jTextArea1);
javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup()
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 599,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(0, 0, Short.MAX_VALUE))
```

```
);
jPanel1Layout.setVerticalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 231,
javax.swing.GroupLayout.PREFERRED_SIZE)
);
jTextArea1.setBackground(new Color(238,238,238));
jTextArea1.setEditable(false);
jTextArea1.setText(Salida);
jTextArea1.setCaretPosition(0);
jTextArea1.setFont(new java.awt.Font("Tahoma", 0, 12));
jScrollPane1.isWheelScrollingEnabled();
jTextArea1.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
JOptionPane.showMessageDialog(null,jPanel1,"Información
Certificado",JOptionPane.PLAIN_MESSAGE);

} catch (Exception ex) {
JOptionPane.showMessageDialog(null,ex.getMessage(),"Error Information
Certificado",JOptionPane.ERROR_MESSAGE);
}
}

}
private JPanel MostrarCSR(){
try {
File miDir = new File (".");
FileOutputStream fos = new
FileOutputStream(miDir.getCanonicalPath()+"\\newreq.pem");
fos.write(BD_solicitudes.obtener_CSR(jTable1.getModel().getValueAt(jTable1.getSelect
edRow(), 0).toString()));
fos.close();
Runtime cmd = Runtime.getRuntime();
Process process= cmd.exec("openssl req -in
"+miDir.getCanonicalPath()+"\\newreq.pem"+" -noout -text");
BufferedReader read = new BufferedReader(new
InputStreamReader(process.getInputStream()));
String line, Salida="";
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}
Salida=Salida+"\n";
Salida=Salida+"----------------------------------------------------------------
Estructura ASN.1----------------------------------------------------------------\n";
process= cmd.exec("openssl asn1parse -i -in
"+miDir.getCanonicalPath()+"\\newreq.pem");
read = new BufferedReader(new InputStreamReader(process.getInputStream()));
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}

jPanel1 = new javax.swing.JPanel();
jScrollPane1 = new javax.swing.JScrollPane();
jTextArea1 = new javax.swing.JTextArea();
```

```
jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jScrollPane1.setViewportView(jTextArea1);
javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup()
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 599,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(0, 0, Short.MAX_VALUE))
);
jPanel1Layout.setVerticalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 231,
javax.swing.GroupLayout.PREFERRED_SIZE)
);
jTextArea1.setBackground(new Color(238,238,238));
jTextArea1.setEditable(false);
jTextArea1.setText(Salida);
jTextArea1.setCaretPosition(0);
jTextArea1.setFont(new java.awt.Font("Tahoma", 0, 12));
jScrollPane1.isWheelScrollingEnabled();
jTextArea1.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
} catch (Exception ex) {
JOptionPane.showMessageDialog(null,ex.getMessage(),"Error Information
CSR",JOptionPane.ERROR_MESSAGE);
}
return jPanel1;
}
private javax.swing.JPanel jPanel1;
private javax.swing.JTextArea jTextArea1;
// Variables declaration - do not modify
private javax.swing.JPopupMenu jPopupMenu1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JTable jTable1;
private javax.swing.JMenuItem jVerCSR;
private javax.swing.JMenuItem jVerCerti;
}
```

- CSR_Rechazados.java

```
package autoridadregistro;
import java.awt.Color;
import java.awt.Point;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import javax.swing.JOptionPane;
```

```
import javax.swing.JPanel;
import javax.swing.ListSelectionModel;
import javax.swing.SwingUtilities;
import javax.swing.table.DefaultTableModel;
/**** @author Johrman */
public class CSR_Rechazados extends javax.swing.JInternalFrame {
ArrayList<String[]> todas_Solicitudes;
conexionBD BD_solicitudes;
private DefaultTableModel modelo;
public CSR_Rechazados(ArrayList<String[]> t_Sol,conexionBD BD_sol) {
initComponents();
todas_Solicitudes=t_Sol;
BD_solicitudes=BD_sol;
jTable1.setComponentPopupMenu(jPopupMenu1);
jScrollPane1.setViewportView(jTable1);
String data[][] ={};
String col[] = {"ID Solicitud","Emitido A","Emitido Por","Fecha-Hora de
solicitud","Estado"};
modelo = new DefaultTableModel(data,col) {
@Override
public boolean isCellEditable(int data, int col) {
return false;
} };
jTable1.setModel(modelo);
int cont = 0, c1=0;
while(cont<todas_Solicitudes.size()){
if(Integer.parseInt(todas_Solicitudes.get(cont)[3]) == 5){
modelo.insertRow(c1, new Object[]{});
modelo.setValueAt("CSR Rechazada por AC", c1, 4);
modelo.setValueAt(todas_Solicitudes.get(c1)[0], c1, 0);
modelo.setValueAt(todas_Solicitudes.get(c1)[1], c1, 1);
if(todas_Solicitudes.get(c1)[2].length()>36){
modelo.setValueAt(todas_Solicitudes.get(c1)[2].substring(36), c1, 2);
}else{
modelo.setValueAt(todas_Solicitudes.get(c1)[2], c1, 2);
}
modelo.setValueAt(todas_Solicitudes.get(c1)[4], c1, 3);
c1++;
}
cont++;
}
 jTable1.addMouseListener(new MouseAdapter() {
public void mousePressed(MouseEvent e) {
if ( SwingUtilities.isRightMouseButton(e)) {
Point p = e.getPoint();
int rowNumber = jTable1.rowAtPoint( p );
ListSelectionModel modelo = jTable1.getSelectionModel();
modelo.setSelectionInterval( rowNumber, rowNumber );
}
if (e.getClickCount()==2){
JOptionPane.showMessageDialog(null,MostrarCSR(),"Información
CSR",JOptionPane.PLAIN_MESSAGE);
}
```

```
}
} );
}
private JPanel MostrarCSR(){
try {
File miDir = new File (".");
FileOutputStream fos = new
FileOutputStream(miDir.getCanonicalPath()+"\\newreq.pem");
fos.write(BD_solicitudes.obtener_CSR(jTable1.getModel().getValueAt(jTable1.getSelect
edRow(), 0).toString()));
fos.close();
Runtime cmd = Runtime.getRuntime();
Process process= cmd.exec("openssl req -in
"+miDir.getCanonicalPath()+"\\newreq.pem"+" -noout -text");
BufferedReader read = new BufferedReader(new
InputStreamReader(process.getInputStream()));
String line, Salida="";
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}
Salida=Salida+"\n";
Salida=Salida+"---------------------------------------------------------------
Estructura ASN.1----------------------------------------------------------\n";
process= cmd.exec("openssl asn1parse -i -in
"+miDir.getCanonicalPath()+"\\newreq.pem");
read = new BufferedReader(new InputStreamReader(process.getInputStream()));
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}
jPanel1 = new javax.swing.JPanel();
jScrollPane1 = new javax.swing.JScrollPane();
jTextArea1 = new javax.swing.JTextArea();
jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jScrollPane1.setViewportView(jTextArea1);
javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup()
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 599,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(0, 0, Short.MAX_VALUE))
);
jPanel1Layout.setVerticalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 231,
javax.swing.GroupLayout.PREFERRED_SIZE)
);
jTextArea1.setBackground(new Color(238,238,238));
jTextArea1.setEditable(false);
jTextArea1.setText(Salida);
jTextArea1.setCaretPosition(0);
```

```
jTextArea1.setFont(new java.awt.Font("Tahoma", 0, 12));
jScrollPane1.isWheelScrollingEnabled();
jTextArea1.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
} catch (Exception ex) {
JOptionPane.showMessageDialog(null,ex.getMessage(),"Error Information
CSR",JOptionPane.ERROR_MESSAGE);}
return jPanel1;
}
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
JOptionPane.showMessageDialog(null,MostrarCSR(),"Información
CSR",JOptionPane.PLAIN_MESSAGE);
}
private javax.swing.JPanel jPanel1;
private javax.swing.JTextArea jTextArea1;
private javax.swing.JMenuItem jMenuItem1;
private javax.swing.JPopupMenu jPopupMenu1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
}
```

- CancelarSolicitud.java

```
package autoridadregistro;
import java.awt.Color;
import java.awt.Point;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.util.ArrayList;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.ListSelectionModel;
import javax.swing.SwingUtilities;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
/** * * @author Johrman */
public class CancelarSolicitud extends javax.swing.JInternalFrame {
ArrayList<String[]> SolicitudesPendientes;
conexionBD BD_solicitudes;
private DefaultTableModel modelo;
    public CancelarSolicitud( ArrayList<String[]> Sol,conexionBD BD_sol) {
        initComponents();
        SolicitudesPendientes=Sol;
        BD_solicitudes=BD_sol;
        String data[][] ={};
        String col[] = {"ID Solicitud","Emitido A","Emitido Por","Fecha-Hora de
solicitud"};
        modelo = new DefaultTableModel(data,col) {
```

```
            boolean[] canEdit = new boolean [] {
                false, false, false, false
            };
            @Override
            public boolean isCellEditable(int data, int col) {
                return canEdit [col];
            } };
        jTable1.setModel(modelo);
        DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
        centerRenderer.setHorizontalAlignment( JLabel.CENTER );
        jTable1.getColumnModel().getColumn(0).setCellRenderer( centerRenderer );
        jTable1.getColumnModel().getColumn(0).setMaxWidth(71);
        jTable1.getColumnModel().getColumn(0).setMinWidth(70);
        jTable1.getColumnModel().getColumn(2).setMinWidth(320);
        jTable1.getColumnModel().getColumn(2).setMaxWidth(321);
        jTable1.getColumnModel().getColumn(3).setMaxWidth(163);
        jTable1.getColumnModel().getColumn(3).setMinWidth(160);
        jTable1.getColumnModel().getColumn(3).setCellRenderer( centerRenderer );
        int cont = 0;
        while(cont<SolicitudesPendientes.size()){
            modelo.insertRow(cont, new Object[]{});
            modelo.setValueAt(SolicitudesPendientes.get(cont)[0], cont, 0);
            modelo.setValueAt(SolicitudesPendientes.get(cont)[1], cont, 1);
            if(SolicitudesPendientes.get(cont)[2].length()>36){
                modelo.setValueAt(SolicitudesPendientes.get(cont)[2].substring(36),
cont, 2);
            }else{
                modelo.setValueAt(SolicitudesPendientes.get(cont)[2], cont, 2);
            }
            modelo.setValueAt(SolicitudesPendientes.get(cont)[3], cont, 3);
            cont++;
        }
        jTable1.addMouseListener(new MouseAdapter() {
            public void mousePressed(MouseEvent e) {
                if ( SwingUtilities.isRightMouseButton(e)) {
                    Point p = e.getPoint();
                    int rowNumber = jTable1.rowAtPoint( p );
                    ListSelectionModel modelo = jTable1.getSelectionModel();
                    modelo.setSelectionInterval( rowNumber, rowNumber );
                }
                if (e.getClickCount()==2){
                    JOptionPane.showMessageDialog(null,MostrarCSR(),"Information
CSR",JOptionPane.PLAIN_MESSAGE);
                }
            }
        }
        );
    }
    private void jVerCSRActionPerformed(java.awt.event.ActionEvent evt) {
        JOptionPane.showMessageDialog(null,MostrarCSR(),"Information
CSR",JOptionPane.PLAIN_MESSAGE);
    }
```

```
    private void jCancelarActionPerformed(java.awt.event.ActionEvent evt) {
        int opcion=JOptionPane.showConfirmDialog(null,MostrarCSR(),"¿Realmente desea
cancelar la CSR?",JOptionPane.YES_NO_OPTION);
        if(opcion==JOptionPane.YES_NO_OPTION){
            int aux =
BD_solicitudes.CancelarSolicitud(jTable1.getModel().getValueAt(jTable1.getSelectedRo
w(), 0).toString());

((DefaultTableModel)jTable1.getModel()).removeRow(jTable1.getSelectedRow());
            if(aux==1){
                JOptionPane.showMessageDialog(null,"CSR cancelado con éxito","Éxito
Cancelación CSR",JOptionPane.INFORMATION_MESSAGE);
            }
            else{
                JOptionPane.showMessageDialog(null,"No ha sido posible cancelar el
CSR.\nCSR ha cambiado de estado.","Error Cancelación
CSR",JOptionPane.ERROR_MESSAGE);
            }
        }
    }
    private JPanel MostrarCSR(){
        try {
        File miDir = new File (".");
        FileOutputStream fos = new
FileOutputStream(miDir.getCanonicalPath()+"\\newreq.pem");

fos.write(BD_solicitudes.obtener_CSR(jTable1.getModel().getValueAt(jTable1.getSelect
edRow(), 0).toString()));
        fos.close();
        Runtime cmd = Runtime.getRuntime();
        Process process= cmd.exec("openssl req -in
"+miDir.getCanonicalPath()+"\\newreq.pem"+" -noout -text");
        BufferedReader read = new BufferedReader(new
InputStreamReader(process.getInputStream()));
        String line, Salida="";
        while((line = read.readLine()) != null){
            Salida = Salida+line+"\n";
        }
        Salida=Salida+"\n";
        Salida=Salida+"-----------------------------------------------------------------
Estructura ASN.1-----------------------------------------------------------\n";
        process= cmd.exec("openssl asn1parse -i -in
"+miDir.getCanonicalPath()+"\\newreq.pem");
        read = new BufferedReader(new InputStreamReader(process.getInputStream()));
        while((line = read.readLine()) != null){
            Salida = Salida+line+"\n";
        }
        jPanel1 = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jTextArea1.setColumns(20);
        jTextArea1.setRows(5);
        jScrollPane1.setViewportView(jTextArea1);
```

```
        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
599, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 0, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 231,
javax.swing.GroupLayout.PREFERRED_SIZE)
        );
        jTextArea1.setBackground(new Color(238,238,238));
        jTextArea1.setEditable(false);
        jTextArea1.setText(Salida);
        jTextArea1.setCaretPosition(0);
        jTextArea1.setFont(new java.awt.Font("Tahoma", 0, 12));
        jScrollPane1.isWheelScrollingEnabled();
        jTextArea1.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(null,ex.getMessage(),"Error Information
CSR",JOptionPane.ERROR_MESSAGE);
    }
    return jPanel1;
    }
    private javax.swing.JPanel jPanel1;
    private javax.swing.JTextArea jTextArea1;
    private javax.swing.JMenuItem jCancelar;
    private javax.swing.JPopupMenu jPopupMenu1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JTable jTable1;
    private javax.swing.JMenuItem jVerCSR;
}
```

- GenerarSolicitud.java

```
package autoridadregistro;
import iaik.asn1.ASN;
import iaik.asn1.ASN1Object;
import iaik.asn1.CodingException;
import iaik.asn1.DerCoder;
import iaik.asn1.OCTET_STRING;
import iaik.asn1.ObjectID;
import iaik.asn1.structures.AlgorithmID;
import iaik.asn1.structures.Attribute;
import iaik.asn1.structures.ChoiceOfTime;
import iaik.asn1.structures.Name;
import iaik.pkcs.pkcs10.CertificateRequest;
```

```
import iaik.pkcs.pkcs11.Mechanism;
import iaik.pkcs.pkcs11.Module;
import iaik.pkcs.pkcs11.Session;
import iaik.pkcs.pkcs11.Slot;
import iaik.pkcs.pkcs11.Token;
import iaik.pkcs.pkcs11.TokenException;
import iaik.pkcs.pkcs11.objects.Data;
import iaik.pkcs.pkcs11.objects.Key;
import iaik.pkcs.pkcs11.objects.KeyPair;
import iaik.pkcs.pkcs11.objects.PrivateKey;
import iaik.pkcs.pkcs11.objects.RSAPrivateKey;
import iaik.pkcs.pkcs11.objects.RSAPublicKey;
import iaik.pkcs.pkcs11.objects.X509PublicKeyCertificate;
import iaik.pkcs.pkcs7.DigestInfo;
import iaik.pkcs.pkcs7.IssuerAndSerialNumber;
import iaik.pkcs.pkcs7.SignedData;
import iaik.pkcs.pkcs7.SignerInfo;
import iaik.utils.PemOutputStream;
import iaik.x509.X509Certificate;
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.math.BigInteger;
import java.nio.file.Files;
import java.security.InvalidKeyException;
import java.security.InvalidParameterException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.Signature;
import java.security.SignatureException;
import java.security.cert.CertificateException;
import java.security.spec.RSAPublicKeySpec;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Iterator;
import java.util.List;
import java.util.Random;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.scene.paint.Color;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;
```

```
/** * * @author Johrman */
public class GenerarSolicitud extends javax.swing.JInternalFrame {
VentanaPPAL VentanaPrincipal;
File DirFotoUsuario = null;
String EmitidoA;
Module moduloPkcs11;
public GenerarSolicitud(VentanaPPAL VPPAL) {
initComponents();
VentanaPrincipal = VPPAL;
Organizacion.setText("UNIVERSIDAD DE PAMPLONA");
Organizacion.setBackground(new java.awt.Color(204, 255, 204));
Organizacion.setEditable(false);
}

private void PriApellidoKeyTyped(java.awt.event.KeyEvent evt) {
evt.setKeyChar(Character.toUpperCase(evt.getKeyChar()));
char car=evt.getKeyChar();
if((car<'A' || car>'Z')&&(car<' '||car>' ') && car!='Ñ') {
evt.consume();
}
int lon = PriApellido.getText().length();
if(lon<18){
if(((lon>=0 && lon<3) || (lon>16))){
PriApellido.setBackground(new java.awt.Color(255, 204, 204));
}
else{PriApellido.setBackground(new java.awt.Color(204, 255, 204));}} else
{evt.consume();}
}

private void NumeroIDKeyTyped(java.awt.event.KeyEvent evt) {
evt.setKeyChar(Character.toUpperCase(evt.getKeyChar()));
char car=evt.getKeyChar();
if((car<'0' || car>'9')&&(car<' '||car>' ')) {
evt.consume();
}
int lon = NumeroID.getText().length();
if(lon<18){
if(((lon>=0 && lon<5) || (lon>16))){
NumeroID.setBackground(new java.awt.Color(255, 204, 204));
}
else{NumeroID.setBackground(new java.awt.Color(204, 255, 204));}} else
{evt.consume();}
}

private void PriNombreKeyTyped(java.awt.event.KeyEvent evt) {
evt.setKeyChar(Character.toUpperCase(evt.getKeyChar()));
char car=evt.getKeyChar();
if((car<'A' || car>'Z')&&(car<' '||car>' ') && car!='Ñ') {
evt.consume();
}
int lon = PriNombre.getText().length();
if(lon<18){
if(((lon>=0 && lon<3) || (lon>16))){
```

```
PriNombre.setBackground(new java.awt.Color(255, 204, 204));
}
else{PriNombre.setBackground(new java.awt.Color(204, 255, 204));}} else
{evt.consume();}
}
private void SegNombreKeyTyped(java.awt.event.KeyEvent evt) {
evt.setKeyChar(Character.toUpperCase(evt.getKeyChar()));
char car=evt.getKeyChar();
if((car<'A' || car>'Z')&&(car<' '||car>' ') && car!='Ñ') {
evt.consume();
}
int lon = SegNombre.getText().length();
if(lon<18){
if(((lon>=1 && lon<3) || (lon>16))){
SegNombre.setBackground(new java.awt.Color(255, 204, 204));
}else if (lon==0){SegNombre.setBackground(new java.awt.Color(255, 255, 255));}
else{SegNombre.setBackground(new java.awt.Color(204, 255, 204));}} else
{evt.consume();}
}
private void EmailKeyTyped(java.awt.event.KeyEvent evt) {
evt.setKeyChar(Character.toUpperCase(evt.getKeyChar()));
char car=evt.getKeyChar();
if((car<'A' || car>'Z')&&(car<' '||car>' ') && !("123456789-
._@".contains(Character.toString(car))) ) {
evt.consume();
}
int lon = Email.getText().length();
if(lon<30){
if(((lon>=0 && lon<3) || (lon>28))){
Email.setBackground(new java.awt.Color(255, 204, 204));
}
else{Email.setBackground(new java.awt.Color(204, 255, 204));}} else {evt.consume();}
}
private void SegApellidoKeyTyped(java.awt.event.KeyEvent evt) {
evt.setKeyChar(Character.toUpperCase(evt.getKeyChar()));
char car=evt.getKeyChar();
if((car<'A' || car>'Z')&&(car<' '||car>' ') && car!='Ñ') {
evt.consume();
}
int lon = SegApellido.getText().length();
if(lon<18){
if(((lon>=1 && lon<3) || (lon>16))){
SegApellido.setBackground(new java.awt.Color(255, 204, 204));
}else if (lon==0){SegApellido.setBackground(new java.awt.Color(255, 255, 255));}
else{SegApellido.setBackground(new java.awt.Color(204, 255, 204));}} else
{evt.consume();}
}
private void TelefonoKeyTyped(java.awt.event.KeyEvent evt) {
evt.setKeyChar(Character.toUpperCase(evt.getKeyChar()));
char car=evt.getKeyChar();
if((car<'0' || car>'9')&&(car<' '||car>' ')) {
evt.consume();
}
```

```
int lon = Telefono.getText().length();
if(lon<18){
if(((lon>=1 && lon<5) || (lon>16))){
Telefono.setBackground(new java.awt.Color(255, 204, 204));
}else if (lon==0){Telefono.setBackground(new java.awt.Color(255, 255, 255));}
else{Telefono.setBackground(new java.awt.Color(204, 255, 204));}} else
{evt.consume();}
}
private void DirKeyTyped(java.awt.event.KeyEvent evt) {
evt.setKeyChar(Character.toUpperCase(evt.getKeyChar()));
char car=evt.getKeyChar();
if((car<'A' || car>'Z')&&(car<' '||car>' ') && !("123456789-
._@".contains(Character.toString(car))) ) {
evt.consume();
}
int lon = Dir.getText().length();
if(lon<30){
if(((lon>=1 && lon<3) || (lon>28))){
Dir.setBackground(new java.awt.Color(255, 204, 204));
}else if (lon==0){Dir.setBackground(new java.awt.Color(255, 255, 255));}
else{Dir.setBackground(new java.awt.Color(204, 255, 204));}} else {evt.consume();}
}
private void PaisResidenciaKeyTyped(java.awt.event.KeyEvent evt) {
 evt.setKeyChar(Character.toUpperCase(evt.getKeyChar()));
char car=evt.getKeyChar();
if((car<'A' || car>'Z')&&(car<' '||car>' ') && car!='Ñ') {
evt.consume();
}
int lon = PaisResidencia.getText().length();
if(lon<18){
if(((lon>=0 && lon<1) || (lon>16))){
PaisResidencia.setBackground(new java.awt.Color(255, 204, 204));
}
else{PaisResidencia.setBackground(new java.awt.Color(204, 255, 204));}} else
{evt.consume();}
}
private void DptoResidenciaKeyTyped(java.awt.event.KeyEvent evt) {
 evt.setKeyChar(Character.toUpperCase(evt.getKeyChar()));
char car=evt.getKeyChar();
if((car<'A' || car>'Z')&&(car<' '||car>' ') && car!='Ñ') {
evt.consume();
}
int lon = DptoResidencia.getText().length();
if(lon<18){
if(((lon>=0 && lon<3) || (lon>16))){
DptoResidencia.setBackground(new java.awt.Color(255, 204, 204));
}
else{DptoResidencia.setBackground(new java.awt.Color(204, 255, 204));}} else
{evt.consume();}
}
private void CiudadResidenciaKeyTyped(java.awt.event.KeyEvent evt) {
 evt.setKeyChar(Character.toUpperCase(evt.getKeyChar()));
char car=evt.getKeyChar();
```

```
if((car<'A' || car>'Z')&&(car<' '||car>' ') && car!='Ñ') {
evt.consume();
}
int lon = CiudadResidencia.getText().length();
if(lon<18){
if(((lon>=0 && lon<3) || (lon>16))){
CiudadResidencia.setBackground(new java.awt.Color(255, 204, 204));
}
else{CiudadResidencia.setBackground(new java.awt.Color(204, 255, 204));}} else
{evt.consume();}
}
private void ProAcademicoKeyTyped(java.awt.event.KeyEvent evt) {
evt.setKeyChar(Character.toUpperCase(evt.getKeyChar()));
char car=evt.getKeyChar();
if((car<'A' || car>'Z')&&(car<' '||car>' ') && car!='Ñ') {
evt.consume();
}
int lon = ProAcademico.getText().length();
if(lon<37){
if(((lon>=0 && lon<3) || (lon>35) )){
ProAcademico.setBackground(new java.awt.Color(255, 204, 204));
}
else{ProAcademico.setBackground(new java.awt.Color(204, 255, 204));}} else
{evt.consume();}
}

private void BotonCargarFotoActionPerformed(java.awt.event.ActionEvent evt) {

FileFilter filtro = new FileNameExtensionFilter( "Archivos de Imágenes",
ImageIO.getReaderFileSuffixes());
JFileChooser dig = new JFileChooser();
dig.addChoosableFileFilter(filtro);
if(dig.showOpenDialog(null) == JFileChooser.APPROVE_OPTION){
try {
DirFotoUsuario = dig.getSelectedFile();
fotoUsuario.getGraphics().drawImage(ImageIO.read(DirFotoUsuario),0,0, 100, 130,
fotoUsuario);
} catch (IOException ex) {
DirFotoUsuario=null;
Logger.getLogger(GenerarSolicitud.class.getName()).log(Level.SEVERE, null, ex);
}
}
}
private void BotonAceptarActionPerformed(java.awt.event.ActionEvent evt) {
 try {
 Slot[] slots =
VentanaPrincipal.pkcs11Modulo.getSlotList(Module.SlotRequirement.TOKEN_PRESENT);
 if(slots.length!=2){JOptionPane.showMessageDialog(null,"No se continuar.\nInserte
otra tarjeta para generar el CSR.","Error", JOptionPane.ERROR_MESSAGE);}
 else{
if(ValidarIngreso()==0){JOptionPane.showMessageDialog(null,"Problemas en algunos de
los campos ingresados\n!Favor revise los campos¡","Error",
JOptionPane.ERROR_MESSAGE);}
```

```
else{
VentanaPrincipal.ctrlPrograma=false;
VentanaPrincipal.pkcs11Modulo.finalize();
moduloPkcs11 = Module.getInstance(VentanaPrincipal.DireccionLibreria);
moduloPkcs11.initialize(null);
slots = moduloPkcs11.getSlotList(Module.SlotRequirement.TOKEN_PRESENT);
File CSR_Archivo = GenerarParClaves(slots[1]);
if(CSR_Archivo !=null){
if(FirmarCSR(slots[0],CSR_Archivo)!=0){
//ConfigurarPIN(sesion);
int id = EnviarCSR(CSR_Archivo);
if(id!=1){
final ImageIcon icon = new
ImageIcon(ImageIO.read(IniciarSesion.class.getResource("../imagenes/ok.png")));
JOptionPane.showOptionDialog(null,"CSR generado con éxito\nID de solicitud:
"+"12"+"\nContinuar para salir.","Éxito",JOptionPane.OK_OPTION,
JOptionPane.PLAIN_MESSAGE,icon,new java.lang.Object[]{"Continuar"},"Continuar");
SetearCampos();
}
}
}
moduloPkcs11.finalize();
VentanaPrincipal.pkcs11Modulo =
Module.getInstance(VentanaPrincipal.DireccionLibreria);
VentanaPrincipal.pkcs11Modulo.initialize(null);
VentanaPrincipal.ctrlPrograma=true;
}
 }
 } catch (TokenException ex) {
 JOptionPane.showMessageDialog(null,"Error al iniciar la tarjeta del
usuario\n"+ex.getMessage(),"Error", JOptionPane.ERROR_MESSAGE);
 } catch (Throwable ex) {
Logger.getLogger(GenerarSolicitud.class.getName()).log(Level.SEVERE, null, ex);
}
}
private int ValidarIngreso() {
if( (CiudadResidencia.getText().length()>=4) &&
(Dir.getText().length()>=4 || Dir.getText().length()==0) &&
(DptoResidencia.getText().length()>=4) &&
(Email.getText().length()>=4) &&
(NumeroID.getText().length()>=6) &&
(PaisResidencia.getText().length()>=2) &&
(PriApellido.getText().length()>=4) &&
(PriNombre.getText().length()>=4) &&
(ProAcademico.getText().length()>=4) &&
(SegApellido.getText().length()>=4 || SegApellido.getText().length()==0) &&
(SegNombre.getText().length()>=4 || SegNombre.getText().length()==0) &&
(Telefono.getText().length()>=6 || Telefono.getText().length()==0) )
{ return 1;}
return 0;
}
private void ConfigurarPIN(Session sesion) {
 try {
```

```
 sesion.setPIN("12345678".toCharArray(),
(NumeroID.getText().substring(NumeroID.getText().length()-4,
NumeroID.getText().length())).toCharArray());
 } catch (TokenException ex) {
 JOptionPane.showMessageDialog(null,"Error al cambiar contraseña","Error",
JOptionPane.ERROR_MESSAGE);
 }
}
private File GenerarParClaves(Slot slotSeleccionado) {
File CSR_Archivo=null;
 try {
Token token = slotSeleccionado.getToken();
Session sesion = token.openSession(Token.SessionType.SERIAL_SESSION,
Token.SessionReadWriteBehavior.RW_SESSION, null, null);
sesion.login(Session.UserType.USER,"12345678".toCharArray());

 String Nom;
 if(SegNombre.getText().length()>3){Nom=PriNombre.getText() +"
"+SegNombre.getText().substring(0, 1)+". ";}
 else{Nom=PriNombre.getText()+" ";}
 if(SegApellido.getText().length()>3){Nom=Nom+PriApellido.getText()+"
"+SegApellido.getText().substring(0, 1);}
 else{Nom=Nom+PriApellido.getText()+".";}
 Nom=Nom.toLowerCase();
 char[] Nombre = Nom.toCharArray();
 Nombre[0] = Character.toUpperCase(Nombre[0]);
 if(SegApellido.getText().length()>3){Nombre[Nombre.length-1] =
Character.toUpperCase(Nombre[Nombre.length-1]);}
for (int i = 0; i < Nom.length()- 2; i++){
 if (Nombre[i] == ' '){
Nombre[i + 1] = Character.toUpperCase(Nombre[i + 1]);
 }
}
Nom=new String(Nombre);
Nom=Nom+". Certificado por: Unipamplona AC";
Nombre=Nom.toCharArray();
 Mechanism MecanisnoGenParClaves = Mechanism.RSA_PKCS_KEY_PAIR_GEN;
 RSAPublicKey PantillaRSAClavePublica = new RSAPublicKey();
 RSAPrivateKey PantillaRSAClavePrivada = new RSAPrivateKey();
 PantillaRSAClavePublica.getObjectClass().setPresent(true);
 PantillaRSAClavePublica.getToken().setBooleanValue(Boolean.TRUE);
 PantillaRSAClavePublica.getPrivate().setBooleanValue(Boolean.FALSE);
 PantillaRSAClavePublica.getLabel().setCharArrayValue(Nombre);
 PantillaRSAClavePublica.getModifiable().setBooleanValue(Boolean.FALSE);
 PantillaRSAClavePublica.getKeyType().setPresent(true);
 byte[] id = new byte[4];
 new Random().nextBytes(id);
 PantillaRSAClavePublica.getId().setByteArrayValue(id);
 PantillaRSAClavePublica.getLocal().setBooleanValue(Boolean.TRUE);
 PantillaRSAClavePublica.getKeyGenMechanism().setMechanism(MecanisnoGenParClaves);
 PantillaRSAClavePublica.getEncrypt().setBooleanValue(Boolean.TRUE);
 PantillaRSAClavePublica.getVerify().setBooleanValue(Boolean.TRUE);
 PantillaRSAClavePublica.getVerifyRecover().setBooleanValue(Boolean.TRUE);
```

```
 PantillaRSAClavePublica.getWrap().setBooleanValue(Boolean.TRUE);
 byte[] mod = new byte[256];
 new Random().nextBytes(mod);
 PantillaRSAClavePublica.getModulus().setByteArrayValue(mod);
 PantillaRSAClavePublica.getModulusBits().setLongValue(new Long(2048));
 byte[] ExponentePublico = { 0x01,0x00,0x01 }; // 2^16 + 1
 PantillaRSAClavePublica.getPublicExponent().setByteArrayValue(ExponentePublico);
 PantillaRSAClavePrivada.getObjectClass().setPresent(true);
 PantillaRSAClavePrivada.getToken().setBooleanValue(Boolean.TRUE);
 PantillaRSAClavePrivada.getPrivate().setBooleanValue(Boolean.TRUE);
 PantillaRSAClavePrivada.getModifiable().setBooleanValue(Boolean.TRUE);
 PantillaRSAClavePrivada.getKeyType().setPresent(true);
 PantillaRSAClavePrivada.getLabel().setCharArrayValue(Nombre);
 PantillaRSAClavePrivada.getId().setByteArrayValue(id);
 PantillaRSAClavePrivada.getLocal().setBooleanValue(Boolean.TRUE);
 PantillaRSAClavePrivada.getKeyGenMechanism().setMechanism(MecanisnoGenParClaves);
 PantillaRSAClavePrivada.getSensitive().setBooleanValue(Boolean.TRUE);
 PantillaRSAClavePrivada.getDecrypt().setBooleanValue(Boolean.TRUE);
 PantillaRSAClavePrivada.getSign().setBooleanValue(Boolean.TRUE);
 PantillaRSAClavePrivada.getSignRecover().setBooleanValue(Boolean.TRUE);
 PantillaRSAClavePrivada.getUnwrap().setBooleanValue(Boolean.TRUE);
 PantillaRSAClavePrivada.getExtractable().setBooleanValue(Boolean.FALSE);
 PantillaRSAClavePrivada.getAlwaysSensitive().setBooleanValue(Boolean.TRUE);
 PantillaRSAClavePrivada.getNeverExtractable().setBooleanValue(Boolean.TRUE);
 PantillaRSAClavePrivada.getModulus().setByteArrayValue(mod);
 PantillaRSAClavePrivada.getPublicExponent().setByteArrayValue(ExponentePublico);
KeyPair generatedKeyPair = sesion.generateKeyPair(MecanisnoGenParClaves,
PantillaRSAClavePublica, PantillaRSAClavePrivada);
RSAPublicKey RSAClavePublica = (RSAPublicKey) generatedKeyPair.getPublicKey();
RSAPrivateKey RSAClavePrivada = (RSAPrivateKey) generatedKeyPair.getPrivateKey();
if(DirFotoUsuario!=null){
InputStream dataInputStream = new FileInputStream(DirFotoUsuario);
ByteArrayOutputStream bufferStream = new ByteArrayOutputStream(256);
byte[] buffer = new byte[4096];
int bytesRead;
while ((bytesRead = dataInputStream.read(buffer)) >= 0) {
bufferStream.write(buffer, 0, bytesRead);
}
dataInputStream.close();
byte[] data = bufferStream.toByteArray();
Data PlantillaObjeto = new Data();
PlantillaObjeto.getLabel().setCharArrayValue("foto".toCharArray());
PlantillaObjeto.getValue().setByteArrayValue(data);
PlantillaObjeto.getToken().setBooleanValue(Boolean.TRUE);
PlantillaObjeto.getModifiable().setBooleanValue(Boolean.FALSE);
PlantillaObjeto.getApplication().setCharArrayValue("foto".toCharArray());
PlantillaObjeto.getObjectID().setByteArrayValue(id);
sesion.createObject(PlantillaObjeto);
}

Name Asunto = new Name();
Asunto.addRDN(ObjectID.country, PaisResidencia.getText().substring(0, 2));
Asunto.addRDN(ObjectID.stateOrProvince, DptoResidencia.getText());
```

```
Asunto.addRDN(ObjectID.locality, CiudadResidencia.getText());
Asunto.addRDN(ObjectID.organization ,Organizacion.getText());
Asunto.addRDN(ObjectID.organizationalUnit ,ProAcademico.getText());
Nom=PriNombre.getText()+" ";
if(SegNombre.getText().length()>3){Nom=Nom+SegNombre.getText()+" ";}
if(SegApellido.getText().length()>3){Nom=Nom+PriApellido.getText()+"
"+SegApellido.getText()+".";}
else{Nom=Nom+PriApellido.getText()+".";}
EmitidoA=Nom;
Asunto.addRDN(ObjectID.commonName , Nom);
Asunto.addRDN(ObjectID.emailAddress ,Email.getText());
BigInteger ModuloBitsTamaño = new BigInteger(1,
RSAClavePublica.getModulus().getByteArrayValue());
BigInteger ExpPublico = new BigInteger(1,
RSAClavePublica.getPublicExponent().getByteArrayValue());
RSAPublicKeySpec RSAClavePublicaDetalles = new RSAPublicKeySpec(ModuloBitsTamaño,
ExpPublico);
iaik.security.rsa.RSAPublicKey AsuntoClavePublica = new
iaik.security.rsa.RSAPublicKey(RSAClavePublicaDetalles);
CertificateRequest SolicitudCerticado = new CertificateRequest(AsuntoClavePublica,
Asunto);
Signature TipodeFirma = new PKCS11SignatureEngine("SHA512withRSA", sesion,
Mechanism.RSA_PKCS, AlgorithmID.sha512);
AlgorithmIDAdapter AlgoritmodeFirmaPKCS11 = new
AlgorithmIDAdapter(AlgorithmID.sha512WithRSAEncryption);
AlgoritmodeFirmaPKCS11.setSignatureInstance(TipodeFirma);
java.security.PrivateKey TokenClavePrivada = new
TokenKeyRSAPrivate(RSAClavePrivada);
SolicitudCerticado.sign(AlgoritmodeFirmaPKCS11 , TokenClavePrivada);
sesion.closeSession();
sesion=null;
String LineaInicial = "-----BEGIN NEW CERTIFICATE REQUEST-----";
String LineaFinal ="-----END NEW CERTIFICATE REQUEST-----";
CSR_Archivo = new File("newreqSING.pem");
OutputStream CSR = new PemOutputStream(new FileOutputStream(CSR_Archivo),
LineaInicial, LineaFinal);
SolicitudCerticado.writeTo(CSR);
CSR.flush();
CSR.close();
 } catch (Exception ex) {
 CSR_Archivo = null;
JOptionPane.showMessageDialog(null,"Error al crear el
CSR\n"+ex.getMessage(),"Error", JOptionPane.ERROR_MESSAGE);
 }
 return CSR_Archivo;
}
private int FirmarCSR(Slot slotSeleccionado, File CSR_Archivo){
try {
VentanaPrincipal.ctrlPrograma=false;
Token token = slotSeleccionado.getToken();
Session sesion = token.openSession(Token.SessionType.SERIAL_SESSION,
Token.SessionReadWriteBehavior.RW_SESSION, null, null);
sesion.login(Session.UserType.USER,VentanaPrincipal.claveOperario);
```

```
PrivateKey PlantillaRSAClavePrivada = new PrivateKey();
sesion.findObjectsInit(PlantillaRSAClavePrivada);
Object[] ClavePrivadaObjectos = sesion.findObjects(1);
PrivateKey ClavePrivada = (PrivateKey) ClavePrivadaObjectos[0];
sesion.findObjectsFinal();
List Certificados = new Vector();
X509PublicKeyCertificate PlantillaCertificado = new X509PublicKeyCertificate();
PlantillaCertificado.getId().setByteArrayValue(ClavePrivada.getId().getByteArrayValu
e());
sesion.findObjectsInit(PlantillaCertificado);
Object[] CertificadosObjectos;
while ((CertificadosObjectos = sesion.findObjects(1)).length > 0) {
Certificados.add(CertificadosObjectos[0]);
}
sesion.findObjectsFinal();
X509PublicKeyCertificate CertificadoFirma = null;
Iterator tokenCertificatesIterator = Certificados.iterator();
while (tokenCertificatesIterator.hasNext()) {
CertificadoFirma = (X509PublicKeyCertificate) tokenCertificatesIterator.next();
}
X509Certificate Firma =new
X509Certificate(CertificadoFirma.getValue().getByteArrayValue());
InputStream CSRclaro = new FileInputStream(CSR_Archivo.getAbsolutePath());
MessageDigest TipodeFirma = MessageDigest.getInstance("SHA-512");
ByteArrayOutputStream bu = new ByteArrayOutputStream();
byte[] datosBu = new byte[1024];
int bytesRead;
while ((bytesRead = CSRclaro.read(datosBu)) >= 0) {
TipodeFirma.update(datosBu, 0, bytesRead);
bu.write(datosBu, 0, bytesRead);
}
byte[] Hash = TipodeFirma.digest();
bu.close();
SignedData CSRfirmado = new SignedData(bu.toByteArray(), SignedData.IMPLICIT);
CSRfirmado.setCertificates(new X509Certificate[] { Firma });
SignerInfo InfoFirma = new SignerInfo(new IssuerAndSerialNumber(Firma),
AlgorithmID.sha512, null);
iaik.asn1.structures.Attribute[] CamposAutenticacion = {
new Attribute(ObjectID.contentType, new ASN1Object[] {ObjectID.pkcs7_data}),
new Attribute(ObjectID.signingTime, new ASN1Object[] {new
ChoiceOfTime().toASN1Object()}),
new Attribute(ObjectID.messageDigest, new ASN1Object[] {new OCTET_STRING(Hash)})
};
InfoFirma.setAuthenticatedAttributes(CamposAutenticacion);
byte[] aFirmar = DerCoder.encode(ASN.createSetOf(CamposAutenticacion, true));
byte[] hashFirmado = TipodeFirma.digest(aFirmar);
DigestInfo InfoTipoFirma = new DigestInfo(AlgorithmID.sha512, hashFirmado);
byte[] CSRcifrado = InfoTipoFirma.toByteArray();
sesion.signInit(Mechanism.RSA_PKCS, ClavePrivada);
byte[] CSRcifradoByte = sesion.sign(CSRcifrado);
InfoFirma.setEncryptedDigest(CSRcifradoByte);
CSRfirmado.addSignerInfo(InfoFirma);
OutputStream CSRFirmado_Archivo = new FileOutputStream("newreqSING.pem");
```

```
CSRfirmado.writeTo(CSRFirmado_Archivo);
CSRFirmado_Archivo.flush();
CSRFirmado_Archivo.close();
sesion.closeSession();
sesion=null;
} catch (Exception ex) {
JOptionPane.showMessageDialog(null,ex.getMessage(),"Error Firma de Autenticidad
CSR", JOptionPane.ERROR_MESSAGE);
return 0;
}
return 1;
}
private int EnviarCSR(File CSR_Archivo){
if(VentanaPrincipal.BD_solicitudes.stm==null){JOptionPane.showMessageDialog(null,"Er
ror al conectar a la base de datos" ,"Error", JOptionPane.ERROR_MESSAGE);}
else
if(VentanaPrincipal.BD_solicitudes.subirCSR(CSR_Archivo,EmitidoA,VentanaPrincipal.da
tosOperario)){
return VentanaPrincipal.BD_solicitudes.obtenerID_CSR();
}
return -1;
}
private void SetearCampos(){
CiudadResidencia.setText("");
 Dir.setText("");
 DptoResidencia.setText("");
 Email.setText("");
InformacionAdicional.setText("");
 NumeroID.setText("");
 PaisResidencia.setText("");
 PriApellido.setText("");
 PriNombre.setText("");
 ProAcademico.setText("");
 SegApellido.setText("");
 SegNombre.setText("");
 Telefono.setText("");
 DirFotoUsuario = null;
 fotoUsuario.setText("");
 CiudadResidencia.setBackground(java.awt.Color.WHITE);
 Dir.setBackground(java.awt.Color.WHITE);
 DptoResidencia.setBackground(java.awt.Color.WHITE);
 Email.setBackground(java.awt.Color.WHITE);
InformacionAdicional.setBackground(java.awt.Color.WHITE);
 NumeroID.setBackground(java.awt.Color.WHITE);
 PaisResidencia.setBackground(java.awt.Color.WHITE);
 PriApellido.setBackground(java.awt.Color.WHITE);
 PriNombre.setBackground(java.awt.Color.WHITE);
 ProAcademico.setBackground(java.awt.Color.WHITE);
 SegApellido.setBackground(java.awt.Color.WHITE);
 SegNombre.setBackground(java.awt.Color.WHITE);
 Telefono.setBackground(java.awt.Color.WHITE);
 fotoUsuario.setBackground(java.awt.Color.WHITE);
}
```

```
private javax.swing.JButton BotonAceptar;
private javax.swing.JButton BotonCancelar;
private javax.swing.JButton BotonCargarFoto;
private javax.swing.JTextField CiudadResidencia;
private javax.swing.JTextField Dir;
private javax.swing.JTextField DptoResidencia;
private javax.swing.JTextField Email;
private javax.swing.JTextArea InformacionAdicional;
private javax.swing.JTextField NumeroID;
private javax.swing.JTextField Organizacion;
private javax.swing.JTextField PaisResidencia;
private javax.swing.JTextField PriApellido;
private javax.swing.JTextField PriNombre;
private javax.swing.JTextField ProAcademico;
private javax.swing.JTextField SegApellido;
private javax.swing.JTextField SegNombre;
private javax.swing.JTextField Telefono;
private javax.swing.JLabel fotoUsuario;
private javax.swing.JLabel jCiudadResidencia;
private javax.swing.JLabel jDir;
private javax.swing.JLabel jDptoResidencia;
private javax.swing.JLabel jEmail;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jNumeroID;
private javax.swing.JLabel jOrganizacion;
private javax.swing.JLabel jPaisResidencia;
private javax.swing.JPanel jPanelFotografia;
private javax.swing.JPanel jPanelInforAdicional;
private javax.swing.JPanel jPanelInformacionAcademica;
private javax.swing.JPanel jPanelInformacionBasica;
private javax.swing.JPanel jPanelInformacionContacto;
private javax.swing.JLabel jPriApellido;
private javax.swing.JLabel jPriNombre;
private javax.swing.JLabel jProAcademico;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JLabel jSegApellido;
private javax.swing.JLabel jSegNombre;
private javax.swing.JLabel jTelefono;
public class TokenKeyRSAPrivate implements java.security.PrivateKey {
protected PrivateKey tokenPrivateKey_;
public TokenKeyRSAPrivate(PrivateKey tokenPrivateKey) {
tokenPrivateKey_ = tokenPrivateKey;
}
public String getAlgorithm() {
return null ;
}
public String getFormat() {
return null ;
}
public byte[] getEncoded() {
return null ;
}
public PrivateKey getTokenPrivateKey() {
```

```
return tokenPrivateKey_ ;
}
}
public class PKCS11SignatureEngine extends Signature {
protected Session session_;
protected Mechanism signatureMechanism_;
protected Key signatureKey_;
protected AlgorithmID hashAlgorithm_;
protected MessageDigest digestEngine_;
public PKCS11SignatureEngine(String algorithmName, Session session, Mechanism
signatureMechanism, AlgorithmID hashAlgorithm) throws NoSuchAlgorithmException{
super(algorithmName);
session_ = session;
signatureMechanism_ = signatureMechanism;
hashAlgorithm_ = hashAlgorithm;
digestEngine_ = hashAlgorithm_.getMessageDigestInstance();
}
protected boolean engineVerify(byte[] signatureValue) throws SignatureException {
throw new UnsupportedOperationException();
}
protected java.lang.Object engineGetParameter(String name) throws
InvalidParameterException {
throw new UnsupportedOperationException();
}
protected void engineSetParameter(String param, java.lang.Object value) throws
InvalidParameterException{
throw new UnsupportedOperationException();
}
protected void engineInitSign(java.security.PrivateKey privateKey) throws
InvalidKeyException{
if (!(privateKey instanceof TokenKeyRSAPrivate)) {
throw new InvalidKeyException("Private key must be of instance
InvalidKeyException");
}
signatureKey_ = ((TokenKeyRSAPrivate) privateKey).getTokenPrivateKey();
}
protected byte[] engineSign() throws SignatureException {
byte[] hashToBeSigned = digestEngine .digest();
DigestInfo digestInfoEngine = new DigestInfo(AlgorithmID.sha512, hashToBeSigned);
byte[] toBeEncrypted = digestInfoEngine.toByteArray();
byte[] signatureValue = null;
try {
session .signInit(signatureMechanism , signatureKey_);
signatureValue = session_.sign(toBeEncrypted);
} catch (TokenException ex) { throw new SignatureException(ex.toString()); }
return signatureValue ;
}
protected void engineInitVerify(java.security.PublicKey publicKey) throws
InvalidKeyException {
throw new UnsupportedOperationException();
}
protected void engineUpdate(byte dataByte) throws SignatureException{
digestEngine_.update(dataByte);
```

```
}
protected void engineUpdate(byte[] data, int offset, int length) throws
SignatureException{
digestEngine_.update(data, offset, length);
}
}

public class AlgorithmIDAdapter extends AlgorithmID {
protected AlgorithmID delegate_;
protected Signature signatureEngine_;
public AlgorithmIDAdapter(AlgorithmID delegate) {
super(delegate.getAlgorithm());
delegate_ = delegate;
}
public void setSignatureInstance(Signature signatureEngine) {
signatureEngine_ = signatureEngine;
}
public Signature getSignatureInstance() throws NoSuchAlgorithmException{
return (signatureEngine_ != null) ? signatureEngine_ : super.getSignatureInstance();
}
public Signature getSignatureInstance(String providerName) throws
NoSuchAlgorithmException {
return (providerName == null) ? getSignatureInstance() :
super.getSignatureInstance(providerName);
}
}
}
```

- conexionBD.java

```
package autoridadregistro;
import iaik.pkcs.PKCSParsingException;
import iaik.pkcs.pkcs7.SignedDataStream;
import iaik.pkcs.pkcs7.SignerInfo;
import iaik.x509.X509Certificate;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.logging.Level;
import java.util.logging.Logger;
/** * * @author Johrman */
public class conexionBD {
Connection co;
Statement stm = null;
public conexionBD(String BD){
try {
 Class.forName("com.mysql.jdbc.Driver");
```

```
 co =
DriverManager.getConnection("jdbc:mysql://localhost/"+BD+"?user=root2&password=");
 if(co!=null){stm = co.createStatement();}
} catch(Exception ex){stm = null;
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
}
public boolean subirCSR(File CSR_Archivo,String Emitido_A,String[] Emitido_Por){
String insert = "insert into solicitudes(CSR,FechaCSR,EmitidoA,EmitidoPor,VarCTRL)
values(?,?,?,?,?)";
FileInputStream fis = null;
PreparedStatement ps = null;
try {
co.setAutoCommit(false);
fis = new FileInputStream(CSR_Archivo);
ps = co.prepareStatement(insert);
ps.setBinaryStream(1,fis,(int)CSR_Archivo.length());
ps.setString(2,(new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")).format(new
java.util.Date()));
ps.setString(3, Emitido_A);
ps.setString(4, "Unipamplona RA. Nombre y Apellidos: "+Emitido_Por[0]+"
"+Emitido_Por[1]+". Cargo:"+Emitido_Por[2]);
ps.setInt(5, 1);
ps.executeUpdate();
co.commit();
return true;
} catch (Exception ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}finally{
try {
ps.close();
fis.close();
} catch (Exception ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
}
return false;
}
public ArrayList<String[]> SolicitudesPendientes(){
ArrayList<String[]> solicitudesPend = new ArrayList();
String[] campos;
try {
ResultSet rs = stm.executeQuery("SELECT * FROM solicitudes");
while (rs.next())
{
campos = new String[4];
if(rs.getInt("VarCTRL")==1){
campos[0]=rs.getObject("idCSR").toString();
campos[1] = rs.getObject("EmitidoA").toString();
campos[2]= rs.getObject("EmitidoPor").toString();
campos[3] = ((new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss")).format(rs.getTimestamp("FechaCSR"))).toUpperCase();
solicitudesPend.add(campos);
```

```
}
}
rs.close();
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
return solicitudesPend;
}
public ArrayList<String[]> listarSolicitudes(){
ArrayList<String[]> solicitudesPend = new ArrayList();
String[] campos;
try {
ResultSet rs = stm.executeQuery("SELECT * FROM solicitudes");
while (rs.next())
{
campos = new String[7];
campos[0]=rs.getObject("idCSR").toString();
campos[1] = rs.getObject("EmitidoA").toString();
campos[2]= rs.getObject("EmitidoPor").toString();
campos[3]=rs.getObject("VarCTRL").toString();
campos[4] = ((new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss")).format(rs.getTimestamp("FechaCSR"))).toUpperCase();
campos[5] = "-";
campos[6] = "-";
solicitudesPend.add(campos);

}
rs.close();
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
return solicitudesPend;
}
public ArrayList<String[]> listarSolicitudes(ArrayList<String[]> sol){
String[] campos;
try {
ResultSet rs = stm.executeQuery("SELECT * FROM certificadosaprobados");
while (rs.next())
{
for (int i =0; i<sol.size(); i++){
if(sol.get(i)[0].contentEquals(rs.getObject("idCSR").toString())){
campos = new String[7];
campos[0]=sol.get(i)[0];
campos[1] = sol.get(i)[1];
campos[2]= sol.get(i)[2];
campos[3]=sol.get(i)[3];
campos[4]=sol.get(i)[4];
campos[5]=rs.getObject("IDCertiAprobado").toString();
campos[6]=((new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss")).format(rs.getTimestamp("FechaCertificado"))).toUpperCase();
sol.set(i, campos);
}
}
```

```
}
rs.close();
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
return sol;
}

public int obtenerID_CSR(){
String id = "-1";
try {
ResultSet rs = stm.executeQuery("SELECT idCSR FROM solicitudes ORDER BY idCSR DESC
LIMIT 1");
while (rs.next()){
id = rs.getObject("idCSR").toString();
}
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
return -1;
}
return Integer.parseInt(id);
}
 public byte[] obtener_CSR(String idCSR){
byte[] CSR = null;
try {
ResultSet rs = stm.executeQuery("SELECT `CSR` FROM `solicitudes` WHERE
`idCSR`="+idCSR);
while (rs.next()){
Blob blob = rs.getBlob("CSR");
InputStream CSRinputStream = new ByteArrayInputStream(blob.getBytes(1,
(int)blob.length()));
SignedDataStream signedData = new SignedDataStream(CSRinputStream);
InputStream contentStream = signedData.getInputStream();
byte[] buffer = new byte[1024];
int bytesRead;
ByteArrayOutputStream output = new ByteArrayOutputStream();
while ((bytesRead = contentStream.read(buffer)) > 0) {
output.write(buffer, 0, bytesRead);
}
CSR=output.toByteArray();
}
} catch (Exception ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
CSR=null;
}
return CSR;
}
public byte[] obt_CERT(String idCSR){
byte[] CSR = null;
try {
ResultSet rs = stm.executeQuery("SELECT `Certificado` FROM `certificadosaprobados`
WHERE `IDCertiAprobado`="+idCSR);
while (rs.next()){
```

```
Blob blob = rs.getBlob("Certificado");
InputStream CSRinputStream = new ByteArrayInputStream(blob.getBytes(1,
(int)blob.length()));
ByteArrayOutputStream bos = new ByteArrayOutputStream();
int next = CSRinputStream.read();
while (next > -1) {
bos.write(next);
next = CSRinputStream.read();
}
bos.flush();
CSR=bos.toByteArray();
}
} catch (Exception ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
CSR=null;
}
return CSR;
}
public InputStream importar_CERT(String idCSR){
InputStream CSRinputStream = null;
try {
ResultSet rs = stm.executeQuery("SELECT `Certificado` FROM `certificadosaprobados`
WHERE `IDCertiAprobado`="+idCSR);
while (rs.next()){
Blob blob = rs.getBlob("Certificado");
CSRinputStream = new ByteArrayInputStream(blob.getBytes(1, (int)blob.length()));
}
} catch (Exception ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
CSRinputStream=null;
}
return CSRinputStream;
}
public int cambiarEstadoCertificado(String idCSR, int Estado){
int n=0;
try {
String query = "UPDATE `certificadosaprobados` SET `VarCRTL`=? WHERE
`IDCertiAprobado`=?;";
PreparedStatement preparedStmt = co.prepareStatement(query);
preparedStmt.setInt(1, Estado);
preparedStmt.setInt(2, Integer.parseInt(idCSR));
n = preparedStmt.executeUpdate();
} catch (SQLException ex) {
n=0;
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
return n;
}
 public int CancelarSolicitud(String idCSR){
int n=0;
try {
String query = "UPDATE `solicitudes` SET `VarCTRL`=6 WHERE `idCSR`=? and
`VarCTRL`=1;";
```

```
PreparedStatement preparedStmt = co.prepareStatement(query);
preparedStmt.setInt(1, Integer.parseInt(idCSR));
n = preparedStmt.executeUpdate();
} catch (SQLException ex) {
n=0;
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
return n;
}
public ArrayList<campos>obtener_CERT(){
ArrayList<campos> camposCerti = new ArrayList();
try {
ResultSet rs = stm.executeQuery("SELECT * FROM `certificadosaprobados` WHERE
`VarCRTL`=1");
while (rs.next()){
campos c = new campos();
c.idCerti=new Long( rs.getObject("IDCertiAprobado").toString());
Blob blob = rs.getBlob("Certificado");
c.Certificado = new ByteArrayInputStream(blob.getBytes(1, (int)blob.length()));
c.FechaCertificado = rs.getObject("FechaCertificado").toString();
c.idCSR=new Long( rs.getObject("idCSR").toString());
c.FechaCSR=rs.getObject("FechaCSR").toString();
camposCerti.add(c);
}
} catch (Exception ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
return camposCerti;
}
public ArrayList<campos>obtenerInfoCSR( ArrayList<campos>camposCerti){
try {
ResultSet rs = stm.executeQuery("SELECT `idCSR`, `EmitidoA`, `EmitidoPor` FROM
`solicitudes` WHERE `VarCTRL`=2");
while (rs.next()){
for (int i =0; i<camposCerti.size(); i++){
if(camposCerti.get(i).idCSR == new Long(rs.getObject("idCSR").toString())){
campos c = new campos();
c.idCerti = camposCerti.get(i).idCerti;
c.Certificado = camposCerti.get(i).Certificado;
c.FechaCertificado = camposCerti.get(i).FechaCertificado;
c.idCSR = camposCerti.get(i).idCSR;
c.FechaCSR = camposCerti.get(i).FechaCSR;
c.Emitido A = rs.getObject("EmitidoA").toString();
c.Emitido_por = rs.getObject("EmitidoPor").toString();
camposCerti.set(i, c);
}
}
}
} catch (Exception ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
return camposCerti;
}
```

```
public class campos {
long idCerti;
InputStreamCertificado;
String FechaCertificado;
String Emitido_A;
String Emitido_por;
long idCSR;
String FechaCSR;
}
}
```

B.1.2 Autoridad de Certificación

- VentanaAC.java

```
package autoridadcertificacion;
import iaik.pkcs.pkcs11.Module;
import iaik.pkcs.pkcs11.Slot;
import iaik.pkcs.pkcs11.TokenException;
import iaik.pkcs.pkcs7.SignedDataStream;
import iaik.pkcs.pkcs7.SignerInfo;
import iaik.x509.X509Certificate;
import java.awt.Color;
import java.beans.PropertyVetoException;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
public class VentanaAC extends javax.swing.JFrame {
conexionBD BD_certi_apro = new conexionBD("autoridad_certificacion_bd",
"certificadosaprobados");
conexionBD BD_solicitudes = new
conexionBD("autoridad_certificacion_bd","solicitudes");
conexionBD BD_repositorios = new
conexionBD("repositorios_certificados","repositorios");
boolean CTRL_FirmaAutomatica = false;
```

```
public VentanaAC() {
JOptionPane.showMessageDialog(null,"Solicitud Duplicada.","Error CSR
duplicado",JOptionPane.ERROR_MESSAGE);
initComponents();

this.setIconImage(new ImageIcon(getClass().getResource
("/imagenes/icono2.png")).getImage());
(new Reloj()).start();
}
private class Reloj extends Thread{
Date horaFechaInicio = new Date();
public void run(){
int Min=0, Hora=0, Seg=-1;
while(true){
try {
Date horaFecha_Actual = new Date();
SimpleDateFormat f = new SimpleDateFormat("HH:mm:ss dd/MMMM/yyyy");
horaFecha.setText(f.format(horaFecha_Actual).toUpperCase());
Seg++;
if(Seg==60){Seg=0;Min++;if(Min==60){Min=0;Hora++;}}
tSesion.setText(Hora+":"+Min+":"+Seg);
Thread.sleep(1000);
} catch (InterruptedException ex) { }
}
}
}
private void jSoliPendienteActionPerformed(java.awt.event.ActionEvent evt) {
try {
 ArrayList<conexionBD.campos> Solicitudes =
BD_solicitudes.obtenerSolictudesNuevas();
 if(Solicitudes.size()==0){JOptionPane.showMessageDialog(null,"No se encuentra
nuevas solictudes");}
 else{
 CTRL_FirmaAutomatica=false;
 if(jPilotoAutomatico.isSelected()){
jPilotoAutomatico.setText("Piloto Automático");
jPilotoAutomatico.setSelected(false);
}
 escritorioVirtual.removeAll();
escritorioVirtual.repaint();
SolPendientes gencsr= new
SolPendientes(Solicitudes,BD_solicitudes,BD_certi_apro,BD_repositorios);
escritorioVirtual.add(gencsr);
gencsr.setEnabled(true);
gencsr.setVisible(true);
gencsr.setMaximum(true);
 }
} catch (PropertyVetoException ex) {
Logger.getLogger(VentanaAC.class.getName()).log(Level.SEVERE, null, ex);
}
}
private void jPilotoAutomaticoActionPerformed(java.awt.event.ActionEvent evt) {
if(jPilotoAutomatico.isSelected()){
```

```
jPilotoAutomatico.setText("Desactivar");
CTRL_FirmaAutomatica=true;
(new FirmaAutomatica()).start();
}else{
jPilotoAutomatico.setText("Piloto Automático");
CTRL_FirmaAutomatica=false;
}
}
private void jSoliRechazadasActionPerformed(java.awt.event.ActionEvent evt) {
try {
 ArrayList<conexionBD.campos> Solicitudes =
BD_solicitudes.obtenerSolictudesRechazadas();
 if(Solicitudes.size()==0){JOptionPane.showMessageDialog(null,"No se encuentra
solictudes Rechazadas.");}
 else{
 CTRL_FirmaAutomatica=false;
 if(jPilotoAutomatico.isSelected()){
jPilotoAutomatico.setText("Piloto Automático");
jPilotoAutomatico.setSelected(false);
}
 escritorioVirtual.removeAll();
escritorioVirtual.repaint();
SolRechazadas gencsr= new SolRechazadas(Solicitudes,BD_solicitudes);
escritorioVirtual.add(gencsr);
gencsr.setEnabled(true);
gencsr.setVisible(true);
gencsr.setMaximum(true);
 }
} catch (PropertyVetoException ex) {
Logger.getLogger(VentanaAC.class.getName()).log(Level.SEVERE, null, ex);
}
}
private void jCertFirmadoActionPerformed(java.awt.event.ActionEvent evt) {
try {
 ArrayList<conexionBD.campos> Certificados = BD_solicitudes.obtenerCertificados();
 if(Certificados.size()==0){JOptionPane.showMessageDialog(null,"No se encuentran
certificados");}
 else{
 Certificados = BD_solicitudes.obtenerEmitidoA_CERT(Certificados);
 CTRL_FirmaAutomatica=false;
 if(jPilotoAutomatico.isSelected()){
jPilotoAutomatico.setText("Piloto Automático");
jPilotoAutomatico.setSelected(false);
}
 escritorioVirtual.removeAll();
escritorioVirtual.repaint();
CertFirmados gencsr= new CertFirmados(Certificados,BD_solicitudes,BD_repositorios);
escritorioVirtual.add(gencsr);
gencsr.setEnabled(true);
gencsr.setVisible(true);
gencsr.setMaximum(true);
 }
} catch (PropertyVetoException ex) {
```

```
Logger.getLogger(VentanaAC.class.getName()).log(Level.SEVERE, null, ex);
}
}

private void jCertRevocadoActionPerformed(java.awt.event.ActionEvent evt) {
try {
 ArrayList<conexionBD.campos> Certificados =
BD_solicitudes.obtenerCertificadosRevocados();
 if(Certificados.size()==0){JOptionPane.showMessageDialog(null,"No se encuentran
certificados revocados");}
 else{
 Certificados = BD_solicitudes.obtenerEmitidoA_CERT(Certificados);
 CTRL_FirmaAutomatica=false;
 if(jPilotoAutomatico.isSelected()){
jPilotoAutomatico.setText("Piloto Automático");
jPilotoAutomatico.setSelected(false);
}
 escritorioVirtual.removeAll();
escritorioVirtual.repaint();
CertRevocados gencsr= new CertRevocados(Certificados,BD_solicitudes);
escritorioVirtual.add(gencsr);
gencsr.setEnabled(true);
gencsr.setVisible(true);
gencsr.setMaximum(true);
 }
} catch (PropertyVetoException ex) {
Logger.getLogger(VentanaAC.class.getName()).log(Level.SEVERE, null, ex);
}
}
private class FirmaAutomatica extends Thread{
public void run(){
boolean auxBD_UPLOAD = false, auxBD_Archivo = true;;
File BD Archivo = null, fichero=null, fichero1=null ;
while(CTRL_FirmaAutomatica){
ArrayList<conexionBD.campos> Solicitudes =
BD_solicitudes.obtenerSolictudesNuevasAutomaticas();
if(Solicitudes.size()>=1){
try {
boolean estaVerificado;
byte[] csr = null;
Runtime cmd;

for (int i=0;i<Solicitudes.size();i++){
try{
SignedDataStream signedData = new SignedDataStream(Solicitudes.get(i).CSR);
InputStream contentStream = signedData.getInputStream();
byte[] bufer = new byte[1024];
int bytesRead;
ByteArrayOutputStream output = new ByteArrayOutputStream();
while ((bytesRead = contentStream.read(bufer)) > 0) {
output.write(bufer, 0, bytesRead);
}
csr=output.toByteArray();
```

```
SignerInfo[] signerInfos = signedData.getSignerInfos();
X509Certificate signerCertificate = signedData.verify(0);

estaVerificado=true;
} catch (Exception ex) {
estaVerificado=false;
}
if(estaVerificado){
File miDir = new File (".");

if(auxBD_Archivo){
InputStream BD_inputStream = BD_repositorios.REPO_bajarBD();
byte[] buffer = new byte[BD_inputStream.available()];
BD_inputStream.read(buffer);
BD_Archivo = new File(miDir.getCanonicalPath() +"\\CA\\index.txt");
BD_Archivo.deleteOnExit();
OutputStream BD_Escritura = new FileOutputStream(BD_Archivo);
BD_Escritura.write(buffer);
BD_Escritura.close();
Thread.sleep(200);
auxBD_Archivo=false;

fichero = new File(miDir.getCanonicalPath()+"\\newcert.pem");
fichero.deleteOnExit();
fichero1 = new File(miDir.getCanonicalPath()+"\\newreq.pem");
fichero1.deleteOnExit();
}

FileOutputStream fos = new
FileOutputStream(miDir.getCanonicalPath()+"\\newreq.pem");
fos.write(csr);
fos.close();
Thread.sleep(100);

cmd = Runtime.getRuntime();
Date f= new Date();
int j = Integer.parseInt(new SimpleDateFormat("yyMMdd").format(f));
j=j+10000;
Process process= cmd.exec("openssl ca -config C:\\OpenSSL-Win32\\bin\\openssl.cfg -
key 12345678 -enddate "+j+"235959Z -notext -batch -policy policy_anything -infiles
newreq.pem");
BufferedReader read = new BufferedReader(new
InputStreamReader(process.getInputStream()));
String line, Salida="";
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}
Thread.sleep(10);
if(Salida.length()>100){
auxBD_UPLOAD = true;

BufferedWriter writer = null;
```

```
writer = new BufferedWriter( new FileWriter(miDir.getCanonicalPath()
+"\\newcert.pem"));
writer.write(Salida);
writer.close();
Thread.sleep(60);
BD_solicitudes.SOLI_actualizarEstadoCSR(2,
String.valueOf(Solicitudes.get(i).idCSR));
BD_certi_apro.CERTI_subirCertificado(fichero,String.valueOf(Solicitudes.get(i).idCSR
),Solicitudes.get(i).Fecha);
}
else{
BD_solicitudes.SOLI_actualizarEstadoCSR(3,
String.valueOf(Solicitudes.get(i).idCSR));
}
}
else{
BD_solicitudes.SOLI_actualizarEstadoCSR(4,
String.valueOf(Solicitudes.get(i).idCSR));
}
}
Thread.sleep(100);
if(auxBD_UPLOAD){
BD_repositorios.REPO_subirBD(BD_Archivo);auxBD_UPLOAD=false;}
} catch (IOException ex) {
Logger.getLogger(SolPendientes.class.getName()).log(Level.SEVERE, null, ex);
} catch (InterruptedException ex) {
Logger.getLogger(VentanaAC.class.getName()).log(Level.SEVERE, null, ex);
}
}
try{
Thread.sleep(5000);
} catch (InterruptedException ex) {
Logger.getLogger(VentanaAC.class.getName()).log(Level.SEVERE, null, ex);
}
}
}
}
private javax.swing.JLabel campoAutoCert;
private javax.swing.JLabel campoHoraFecha;
private javax.swing.JLabel campoIngTele;
private javax.swing.JLabel campoTSesion;
private javax.swing.JLabel campoUniversidad;
private javax.swing.JDesktopPane escritorioVirtual;
private javax.swing.JLabel horaFecha;
private javax.swing.JMenuItem jAcercade;
private javax.swing.JMenuItem jCertFirmado;
private javax.swing.JMenuItem jCertRevocado;
private javax.swing.JMenu jCertificado;
private javax.swing.JToggleButton jPilotoAutomatico;
private javax.swing.JMenuItem jSalir;
private javax.swing.JPopupMenu.Separator jSeparator1;
private javax.swing.JMenu jSistema;
private javax.swing.JMenuItem jSoliPendiente;
```

```
private javax.swing.JMenuItem jSoliRechazadas;
private javax.swing.JMenu jSolicitudes;
private javax.swing.JLabel logoUPA;
private javax.swing.JMenuBar menuPPAL;
private javax.swing.JLabel tSesion;
}
```

- SolRechazadas.java

```
package autoridadcertificacion;
import iaik.pkcs.pkcs7.SignedDataStream;
import iaik.pkcs.pkcs7.SignerInfo;
import iaik.x509.X509Certificate;
import java.awt.Color;
import java.awt.Point;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.BufferedReader;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.ListSelectionModel;
import javax.swing.SwingUtilities;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;

/** * * @author Johrman */
public class SolRechazadas extends javax.swing.JInternalFrame {
ArrayList<sol> certificadosFirmantes =new ArrayList();
conexionBD BD_solicitudes;
private DefaultTableModel modelo;
public SolRechazadas(ArrayList<conexionBD.campos> Solicitudes, conexionBD BD) {
initComponents();
BD_solicitudes=BD;
String data[][] ={};
String col[] = {"ID Solicitud","Emitido A","Emitido Por","Verificado","Fecha-Hora de
solicitud"};
modelo = new DefaultTableModel(data,col) {
@Override
public boolean isCellEditable(int data, int col) {
return false;
} };
jTable1.setModel(modelo);
DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
centerRenderer.setHorizontalAlignment( JLabel.CENTER );
jTable1.getColumnModel().getColumn(0).setCellRenderer( centerRenderer );
jTable1.getColumnModel().getColumn(0).setMaxWidth(71);
jTable1.getColumnModel().getColumn(0).setMinWidth(70);
```

```
jTable1.getColumnModel().getColumn(4).setMaxWidth(183);
jTable1.getColumnModel().getColumn(4).setMinWidth(180);
jTable1.getColumnModel().getColumn(4).setCellRenderer( centerRenderer );
jTable1.getColumnModel().getColumn(3).setMaxWidth(173);
jTable1.getColumnModel().getColumn(3).setMinWidth(170);
int cont = 0;
sol soli = new sol();
while(cont<Solicitudes.size()){
modelo.insertRow(cont, new Object[]{});
modelo.setValueAt(Solicitudes.get(cont).idCSR, cont, 0);
modelo.setValueAt(Solicitudes.get(cont).Emitido_A, cont, 1);
if(Solicitudes.get(cont).Emitido_por.length()>36){
modelo.setValueAt(Solicitudes.get(cont).Emitido_por.substring(36), cont, 2);
}else{ modelo.setValueAt(Solicitudes.get(cont).Emitido_por, cont, 2); }
soli = verificarCSR(Solicitudes.get(cont).CSR);
certificadosFirmantes.add(soli);
if(certificadosFirmantes.get(cont).estaVerificado){
modelo.setValueAt("<html><strong>Firma y CSR:</strong> <b><font size=\"3\"
color=\"green\">Verificado</font></b></html>", cont, 3);
} else{
modelo.setValueAt("<html><strong>Firma y CSR:</strong> <b><font size=\"3\"
color=\"red\">NO Verificado</font></b></html>", cont, 3);
}
modelo.setValueAt(Solicitudes.get(cont).Fecha, cont, 4);
cont++;
}
jTable1.addMouseListener(new MouseAdapter() {
public void mousePressed(MouseEvent e) {
if ( SwingUtilities.isRightMouseButton(e)) {
Point p = e.getPoint();
int rowNumber = jTable1.rowAtPoint( p );
ListSelectionModel modelo = jTable1.getSelectionModel();
modelo.setSelectionInterval( rowNumber, rowNumber );
}
if (e.getClickCount()==2){
JOptionPane.showMessageDialog(null,MostrarCSR(),"Información
CSR",JOptionPane.PLAIN_MESSAGE);
}
}
}
);
}
private JPanel MostrarCSR(){
try {
File miDir = new File (".");
FileOutputStream fos = new
FileOutputStream(miDir.getCanonicalPath()+"\\newreq.pem");
fos.write(certificadosFirmantes.get(jTable1.getSelectedRow()).csr);
fos.close();
Runtime cmd = Runtime.getRuntime();
Process process= cmd.exec("openssl req -in
"+miDir.getCanonicalPath()+"\\newreq.pem"+" -noout -text");
```

```
BufferedReader read = new BufferedReader(new
InputStreamReader(process.getInputStream()));
String line, Salida="";
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}
Salida=Salida+"\n";
Salida=Salida+"-----------------------------------------------------------------
Estructura ASN.1----------------------------------------------------------\n";
process= cmd.exec("openssl asn1parse -i -in
"+miDir.getCanonicalPath()+"\\newreq.pem");
read = new BufferedReader(new InputStreamReader(process.getInputStream()));
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}

jPanel1 = new javax.swing.JPanel();
jScrollPane1 = new javax.swing.JScrollPane();
jTextArea1 = new javax.swing.JTextArea();
jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jScrollPane1.setViewportView(jTextArea1);
javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup()
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 599,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(0, 0, Short.MAX_VALUE))
);
jPanel1Layout.setVerticalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 231,
javax.swing.GroupLayout.PREFERRED_SIZE)
);
jTextArea1.setBackground(new Color(238,238,238));
jTextArea1.setEditable(false);
jTextArea1.setText(Salida);
jTextArea1.setCaretPosition(0);
jTextArea1.setFont(new java.awt.Font("Tahoma", 0, 12));
jScrollPane1.isWheelScrollingEnabled();
jTextArea1.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
} catch (Exception ex) {
JOptionPane.showMessageDialog(null,ex.getMessage(),"Error Information
CSR",JOptionPane.ERROR_MESSAGE);
}
return jPanel1;
}
private sol verificarCSR(InputStream dataInput){
sol soli = new sol();
try {
SignedDataStream signedData = new SignedDataStream(dataInput);
```

```
InputStream contentStream = signedData.getInputStream();
byte[] buffer = new byte[1024];
int bytesRead;
ByteArrayOutputStream output = new ByteArrayOutputStream();
while ((bytesRead = contentStream.read(buffer)) > 0) {
output.write(buffer, 0, bytesRead);
}
soli.csr=output.toByteArray();
SignerInfo[] signerInfos = signedData.getSignerInfos();
X509Certificate signerCertificate = signedData.verify(0);
soli.estaVerificado=true;
} catch (Exception ex) {
soli.estaVerificado=false;
}
return soli;
}

private class sol{
byte[] csr;
boolean estaVerificado;
}
private void jVerCSRActionPerformed(java.awt.event.ActionEvent evt) {
JOptionPane.showMessageDialog(null,MostrarCSR(),"Información
CSR",JOptionPane.PLAIN_MESSAGE);
}
private void jActivarCSRActionPerformed(java.awt.event.ActionEvent evt) {
int opcion=JOptionPane.showConfirmDialog(null,MostrarCSR(),"¿Realmente desea Activar
esta solicitud?",JOptionPane.YES_NO_OPTION);
if(opcion==JOptionPane.YES_NO_OPTION){
if(!certificadosFirmantes.get(jTable1.getSelectedRow()).estaVerificado){
BD_solicitudes.SOLI_actualizarEstadoCSR(4,jTable1.getModel().getValueAt(jTable1.getS
electedRow(), 0).toString());
JOptionPane.showMessageDialog(null,"No se puede proseguir.\nError en la firma o hash
de la CSR","Error",JOptionPane.ERROR_MESSAGE);
}
else{
BD_solicitudes.SOLI_actualizarEstadoCSR(1,jTable1.getModel().getValueAt(jTable1.getS
electedRow(), 0).toString());
((DefaultTableModel)jTable1.getModel()).removeRow(jTable1.getSelectedRow());
JOptionPane.showMessageDialog(null,"CSR ha sido re-
activada.","Éxito",JOptionPane.INFORMATION_MESSAGE);
}
}
}
private javax.swing.JPanel jPanel1;
private javax.swing.JTextArea jTextArea1;
private javax.swing.JMenuItem jActivarCSR;
private javax.swing.JPopupMenu jPopupMenu1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JTable jTable1;
private javax.swing.JMenuItem jVerCSR;
}
```

- SolPendientes.java

```
package autoridadcertificacion;
import iaik.pkcs.PKCSParsingException;
import java.io.InputStream;
import java.util.ArrayList;
import iaik.pkcs.pkcs7.SignedDataStream;
import iaik.pkcs.pkcs7.SignerInfo;
import iaik.utils.PemOutputStream;
import iaik.x509.X509Certificate;
import java.awt.Color;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.KeyEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.nio.charset.Charset;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.security.SignatureException;
import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Date;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.AbstractAction;
import javax.swing.Action;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.ListSelectionModel;
import javax.swing.SwingUtilities;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;

/** * * @author Johrman */
public class SolPendientes extends javax.swing.JInternalFrame {
ArrayList<sol> certificadosFirmantes =new ArrayList();
conexionBD BD_solicitudes, BD_certi_apro, BD_repositorios;
```

```
private DefaultTableModel modelo;
public SolPendientes(ArrayList<conexionBD.campos> Solicitudes, conexionBD
BD_sol,conexionBD BD_apro,conexionBD BD_repo) {
initComponents();
BD_solicitudes = BD_sol;
BD_certi_apro=BD_apro;
BD_repositorios=BD_repo;
String data[][] ={};
String col[] = {"ID Solicitud","Emitido A","Emitido Por","Verificado","Fecha-Hora de
solicitud"};
modelo = new DefaultTableModel(data,col) {
@Override
public boolean isCellEditable(int data, int col) {
return col==2;
} };
tablaSolPen.setModel(modelo);
DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
centerRenderer.setHorizontalAlignment( JLabel.CENTER );
tablaSolPen.getColumnModel().getColumn(0).setCellRenderer( centerRenderer );
tablaSolPen.getColumnModel().getColumn(0).setMaxWidth(71);
tablaSolPen.getColumnModel().getColumn(0).setMinWidth(70);
tablaSolPen.getColumnModel().getColumn(2).setMinWidth(140);
tablaSolPen.getColumnModel().getColumn(2).setMaxWidth(150);
tablaSolPen.getColumnModel().getColumn(4).setMaxWidth(183);
tablaSolPen.getColumnModel().getColumn(4).setMinWidth(180);
tablaSolPen.getColumnModel().getColumn(4).setCellRenderer( centerRenderer );
tablaSolPen.getColumnModel().getColumn(3).setMaxWidth(173);
tablaSolPen.getColumnModel().getColumn(3).setMinWidth(170);
int cont = 0;
sol soli = new sol();
while(cont<Solicitudes.size()){
modelo.insertRow(cont, new Object[]{});
modelo.setValueAt(Solicitudes.get(cont).idCSR, cont, 0);
modelo.setValueAt(Solicitudes.get(cont).Emitido_A, cont, 1);
soli = verificarCSR(Solicitudes.get(cont).CSR);
certificadosFirmantes.add(soli);
modelo.setValueAt("Ver Certificado", cont, 2);
if(certificadosFirmantes.get(cont).estaVerificado){
modelo.setValueAt("<html><strong>Firma y CSR:</strong> <b><font size=\"3\"
color=\"green\">Verificado</font></b></html>", cont, 3);
} else{
modelo.setValueAt("<html><strong>Firma y CSR:</strong> <b><font size=\"3\"
color=\"red\">NO Verificado</font></b></html>", cont, 3);
}
modelo.setValueAt(Solicitudes.get(cont).Fecha, cont, 4);
ButtonColumn buttonColumn = new ButtonColumn(tablaSolPen, delete, 2);
buttonColumn.setMnemonic(KeyEvent.VK_D);
cont++;
}
tablaSolPen.addMouseListener(new MouseAdapter() {
public void mousePressed(MouseEvent e) {
if ( SwingUtilities.isRightMouseButton(e)) {
Point p = e.getPoint();
```

```
int rowNumber = tablaSolPen.rowAtPoint( p );
ListSelectionModel modelo = tablaSolPen.getSelectionModel();
modelo.setSelectionInterval( rowNumber, rowNumber );
}
if (e.getClickCount()==2){
JOptionPane.showMessageDialog(null,MostrarCSR(),"Información
CSR",JOptionPane.PLAIN_MESSAGE);
}
}
}
);
}
Action delete = new AbstractAction(){
@Override
public void actionPerformed(ActionEvent e){
int modelRow = Integer.valueOf( e.getActionCommand() );
JOptionPane.showMessageDialog(null,certificadosFirmantes.get(modelRow).CertificadoFi
rmante,"Certificado de Firma", JOptionPane.INFORMATION_MESSAGE);
}
};
private sol verificarCSR(InputStream dataInput){
sol soli = new sol();
try {
SignedDataStream signedData = new SignedDataStream(dataInput);
InputStream contentStream = signedData.getInputStream();
byte[] buffer = new byte[1024];
int bytesRead;
ByteArrayOutputStream output = new ByteArrayOutputStream();
while ((bytesRead = contentStream.read(buffer)) > 0) {
output.write(buffer, 0, bytesRead);
}
soli.csr=output.toByteArray();
SignerInfo[] signerInfos = signedData.getSignerInfos();
X509Certificate signerCertificate = signedData.verify(0);
soli.CertificadoFirmante=signerCertificate.toString();
soli.estaVerificado=true;
} catch (Exception ex) {
soli.estaVerificado=false;
}
return soli;
}
private void FirmarCSR() {
try {
File miDir = new File (".");
File fichero = new File(miDir.getCanonicalPath()+"\\newcert.pem");
fichero.deleteOnExit();
File fichero1 = new File(miDir.getCanonicalPath()+"\\newreq.pem");
fichero1.deleteOnExit();
int opcion=JOptionPane.showConfirmDialog(null,MostrarCSR(),"¿Realmente desea firma
esta solicitud?",JOptionPane.YES_NO_OPTION);
if(opcion==JOptionPane.YES_NO_OPTION){
if(!certificadosFirmantes.get(tablaSolPen.getSelectedRow()).estaVerificado){
```

```
BD_solicitudes.SOLI_actualizarEstadoCSR(4,tablaSolPen.getModel().getValueAt(tablaSol
Pen.getSelectedRow(), 0).toString());
JOptionPane.showMessageDialog(null,"No se puede proseguir.\nError en la firma o hash
de la CSR","Error",JOptionPane.ERROR_MESSAGE);
}
else{
FileOutputStream fos = null;
try {
InputStream BD_inputStream = BD_repositorios.REPO_bajarBD();
byte[] buffer = new byte[BD_inputStream.available()];
BD_inputStream.read(buffer);
File BD_Archivo = new File(miDir.getCanonicalPath() +"\\CA\\index.txt");
BD_Archivo.deleteOnExit();
OutputStream BD_Escritura = new FileOutputStream(BD_Archivo);
BD_Escritura.write(buffer);
BD_Escritura.close();
fos = new FileOutputStream(miDir.getCanonicalPath()+"\\newreq.pem");
fos.write(certificadosFirmantes.get(tablaSolPen.getSelectedRow()).csr);
fos.close();
Runtime cmd = Runtime.getRuntime();
Date f= new Date();
int j = Integer.parseInt(new SimpleDateFormat("yyMMdd").format(f));
j=j+10000;
Process process= cmd.exec("openssl ca -config C:\\OpenSSL-Win32\\bin\\openssl.cfg -
key 12345678 -enddate "+j+"235959Z -notext -batch -policy policy_anything -infiles
newreq.pem");
BufferedReader read = new BufferedReader(new
InputStreamReader(process.getInputStream()));
String line, Salida="";
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}
if(Salida.length()>100){
BufferedWriter writer = null;
writer = new BufferedWriter( new FileWriter(miDir.getCanonicalPath()
+"\\newcert.pem"));
writer.write(Salida);
writer.close();
cmd = Runtime.getRuntime();
process= cmd.exec("openssl x509 -in "+ miDir.getCanonicalPath() +"\\newcert.pem -
text");
read = new BufferedReader(new InputStreamReader(process.getInputStream()));
Salida="";
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}
Salida=Salida+"\n";
Salida=Salida+"-----------------------------------------------------------------
Estructura ASN.1-----------------------------------------------------------\n";
process= cmd.exec("openssl asn1parse -i -in
"+miDir.getCanonicalPath()+"\\newcert.pem");
read = new BufferedReader(new InputStreamReader(process.getInputStream()));
while((line = read.readLine()) != null){
```

```
Salida = Salida+line+"\n";
}
jPanel1 = new javax.swing.JPanel();
jScrollPane1 = new javax.swing.JScrollPane();
jTextArea1 = new javax.swing.JTextArea();
jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jScrollPane1.setViewportView(jTextArea1);
javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup()
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 599,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(0, 0, Short.MAX_VALUE))
);
jPanel1Layout.setVerticalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 231,
javax.swing.GroupLayout.PREFERRED_SIZE)
);
jTextArea1.setBackground(new Color(238,238,238));
jTextArea1.setEditable(false);
jTextArea1.setText(Salida);
jTextArea1.setCaretPosition(0);
jTextArea1.setFont(new java.awt.Font("Tahoma", 0, 12));
jScrollPane1.isWheelScrollingEnabled();
jTextArea1.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
BD_solicitudes.SOLI_actualizarEstadoCSR(2,tablaSolPen.getModel().getValueAt(tablaSol
Pen.getSelectedRow(), 0).toString());
BD_certi_apro.CERTI_subirCertificado(fichero,tablaSolPen.getModel().getValueAt(tabla
SolPen.getSelectedRow(),
0).toString(),tablaSolPen.getModel().getValueAt(tablaSolPen.getSelectedRow(),
4).toString());
BD_repositorios.REPO_subirBD(BD_Archivo);
((DefaultTableModel)tablaSolPen.getModel()).removeRow(tablaSolPen.getSelectedRow());
JOptionPane.showMessageDialog(null,jPanel1,"¡Éxito, CSR
firmado!",JOptionPane.OK_OPTION);
} else{
JOptionPane.showMessageDialog(null,"Solicitud Duplicada.","Error CSR
duplicado",JOptionPane.ERROR_MESSAGE);
BD solicitudes.SOLI actualizarEstadoCSR(3,tablaSolPen.getModel().getValueAt(tablaSol
Pen.getSelectedRow(), 0).toString());
((DefaultTableModel)tablaSolPen.getModel()).removeRow(tablaSolPen.getSelectedRow());
 }
} catch (IOException ex) {
JOptionPane.showMessageDialog(null,"Solicitud Duplicada...","Error CSR
duplicado",JOptionPane.ERROR_MESSAGE);
}
}
}
} catch (IOException ex) {
```

```
Logger.getLogger(SolPendientes.class.getName()).log(Level.SEVERE, null, ex);
}
}
private class sol{
byte[] csr;
boolean estaVerificado;
String CertificadoFirmante;
}
private void FirmarActionPerformed(java.awt.event.ActionEvent evt) {
FirmarCSR();
}
private void VerCSRActionPerformed(java.awt.event.ActionEvent evt) {
JOptionPane.showMessageDialog(null,MostrarCSR(),"Information
CSR",JOptionPane.PLAIN_MESSAGE);
}
private void RechazarActionPerformed(java.awt.event.ActionEvent evt) {
int opcion=JOptionPane.showConfirmDialog(null,MostrarCSR(),"¿Realmente desea
Rechazar la CSR?",JOptionPane.YES_NO_OPTION);
if(opcion==JOptionPane.YES_NO_OPTION){
int aux =
BD_solicitudes.CancelarPeticion(tablaSolPen.getModel().getValueAt(tablaSolPen.getSel
ectedRow(), 0).toString());
((DefaultTableModel)tablaSolPen.getModel()).removeRow(tablaSolPen.getSelectedRow());
if(aux==1){
 JOptionPane.showMessageDialog(null,"CSR rechazado con éxito","Éxito Cancelación
CSR",JOptionPane.INFORMATION_MESSAGE);
}
else{
 JOptionPane.showMessageDialog(null,"No ha sido posible rechazar el CSR.\nCSR ha
cambiado de estado.","Error Cancelación CSR",JOptionPane.ERROR_MESSAGE);
}
}
}
private JPanel MostrarCSR(){
try {
File miDir = new File (".");
FileOutputStream fos = new
FileOutputStream(miDir.getCanonicalPath()+"\\newreq.pem");
fos.write(certificadosFirmantes.get(tablaSolPen.getSelectedRow()).csr);
fos.close();
Runtime cmd = Runtime.getRuntime();
Process process= cmd.exec("openssl req -in
"+miDir.getCanonicalPath()+"\\newreq.pem"+" -noout -text");
BufferedReader read = new BufferedReader(new
InputStreamReader(process.getInputStream()));
String line, Salida="";
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}
Salida=Salida+"\n";
Salida=Salida+"-----------------------------------------------------------------
Estructura ASN.1-----------------------------------------------------------\n";
```

```
process= cmd.exec("openssl asn1parse -i -in
"+miDir.getCanonicalPath()+"\\newreq.pem");
read = new BufferedReader(new InputStreamReader(process.getInputStream()));
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}
jPanel1 = new javax.swing.JPanel();
jScrollPane1 = new javax.swing.JScrollPane();
jTextArea1 = new javax.swing.JTextArea();
jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jScrollPane1.setViewportView(jTextArea1);
javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup()
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 599,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(0, 0, Short.MAX_VALUE))
);
jPanel1Layout.setVerticalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 231,
javax.swing.GroupLayout.PREFERRED_SIZE)
);
jTextArea1.setBackground(new Color(238,238,238));
jTextArea1.setEditable(false);
jTextArea1.setText(Salida);
jTextArea1.setCaretPosition(0);
jTextArea1.setFont(new java.awt.Font("Tahoma", 0, 12));
jScrollPane1.isWheelScrollingEnabled();
jTextArea1.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
} catch (Exception ex) {
JOptionPane.showMessageDialog(null,ex.getMessage(),"Error Information
CSR",JOptionPane.ERROR_MESSAGE);
}
return jPanel1;
}
private javax.swing.JPanel jPanel1;
private javax.swing.JTextArea jTextArea1;
private javax.swing.JMenuItem Firmar;
private javax.swing.JMenuItem Rechazar;
private javax.swing.JMenuItem VerCSR;
private javax.swing.JPopupMenu jMenuContextual;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JPopupMenu.Separator jSeparator1;
private javax.swing.JTable tablaSolPen;
}
```

- CertRevocados.java

```
package autoridadcertificacion;
import java.awt.Color;
```

```
import java.awt.Point;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.BufferedReader;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.ListSelectionModel;
import javax.swing.SwingUtilities;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
public class CertRevocados extends javax.swing.JInternalFrame {
conexionBD BD_solicitudes;
private DefaultTableModel modelo;
public CertRevocados(ArrayList<conexionBD.campos> Solicitudes, conexionBD BD) {
initComponents();
BD_solicitudes=BD;
String data[][] ={};
String col[] = {"<html><b>ID Certificado</b></html>","<html><b>Emitido
A</b></html>","<html><b>Fecha
Aprobación</b></html>","<html><b>Estado</b></html>","<html><b>Fecha-Hora de
Revocación</b></html>","<html><b>Motivo de revocación</b></html>"};
modelo = new DefaultTableModel(data,col) {
@Override
public boolean isCellEditable(int data, int col) {
return false;
} };
jTable1.setModel(modelo);
DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
centerRenderer.setHorizontalAlignment( JLabel.CENTER );
jTable1.getColumnModel().getColumn(0).setCellRenderer( centerRenderer );
jTable1.getColumnModel().getColumn(2).setCellRenderer( centerRenderer );
jTable1.getColumnModel().getColumn(3).setCellRenderer( centerRenderer );
jTable1.getColumnModel().getColumn(4).setCellRenderer( centerRenderer );
int cont = 0;
while(cont<Solicitudes.size()){
modelo.insertRow(cont, new Object[]{});
modelo.setValueAt(Solicitudes.get(cont).idCer, cont, 0);
modelo.setValueAt(Solicitudes.get(cont).Emitido_A, cont, 1);
 modelo.setValueAt(Solicitudes.get(cont).Fecha, cont, 2);
switch (Solicitudes.get(cont).EstadoCerti) {
case 1:modelo.setValueAt("<html><b><font size=\"3\"
color=\"green\">Válido</font></b></html>", cont, 3); break;
case 2:modelo.setValueAt("<html><b><font size=\"3\"
color=\"green\">Válido</font></b></html>", cont, 3); break;
case 3:modelo.setValueAt("<html><b><font size=\"3\"
color=\"red\">Revocado</font></b></html>", cont, 3); break;
```

```
default: modelo.setValueAt("NS/NR", cont, 3); break;
}
modelo.setValueAt(Solicitudes.get(cont).FechaRevocacion, cont, 4);
modelo.setValueAt(Solicitudes.get(cont).MotivoRevocacion, cont, 5);
cont++;
}
jTable1.addMouseListener(new MouseAdapter() {
public void mousePressed(MouseEvent e) {
if ( SwingUtilities.isRightMouseButton(e)) {
Point p = e.getPoint();
int rowNumber = jTable1.rowAtPoint( p );
ListSelectionModel modelo = jTable1.getSelectionModel();
modelo.setSelectionInterval( rowNumber, rowNumber );
}
if (e.getClickCount()==2){MostrarCerti(); }
}
}
);
}
private void MostrarCerti(){
 try {
File miDir = new File (".");
FileOutputStream fos = new
FileOutputStream(miDir.getCanonicalPath()+"\\newreq.pem");
fos.write(BD_solicitudes.obtenerCertificado(jTable1.getModel().getValueAt(jTable1.ge
tSelectedRow(), 0).toString()));
fos.close();
Runtime cmd = Runtime.getRuntime();
Process process= cmd.exec("openssl x509 -in "+ miDir.getCanonicalPath()
+"\\newreq.pem -text");
BufferedReader read = new BufferedReader(new
InputStreamReader(process.getInputStream()));
String line, Salida="";
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}
Salida=Salida+"\n";
Salida=Salida+"-----------------------------------------------------------------
Estructura ASN.1-----------------------------------------------------------------\n";
process= cmd.exec("openssl asn1parse -i -in
"+miDir.getCanonicalPath()+"\\newreq.pem");
read = new BufferedReader(new InputStreamReader(process.getInputStream()));
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}
jPanel1 = new javax.swing.JPanel();
jScrollPane1 = new javax.swing.JScrollPane();
jTextArea1 = new javax.swing.JTextArea();
jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jScrollPane1.setViewportView(jTextArea1);
javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
```

```
jPanel1Layout.setHorizontalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup()
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 599,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(0, 0, Short.MAX_VALUE))
);
jPanel1Layout.setVerticalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 231,
javax.swing.GroupLayout.PREFERRED_SIZE)
);
jTextArea1.setBackground(new Color(238,238,238));
jTextArea1.setEditable(false);
jTextArea1.setText(Salida);
jTextArea1.setCaretPosition(0);
jTextArea1.setFont(new java.awt.Font("Tahoma", 0, 12));
jScrollPane1.isWheelScrollingEnabled();
jTextArea1.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
JOptionPane.showMessageDialog(null,jPanel1,"Información
Certificado",JOptionPane.PLAIN_MESSAGE);
} catch (Exception ex) {
JOptionPane.showMessageDialog(null,ex.getMessage(),"Error Information
Certificado",JOptionPane.ERROR_MESSAGE);
}
}
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
MostrarCerti();
}
private javax.swing.JPanel jPanel1;
private javax.swing.JTextArea jTextArea1;
private javax.swing.JMenuItem jMenuItem1;
private javax.swing.JPopupMenu jPopupMenu1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
}
```

- CertFirmados.java

```
package autoridadcertificacion;
import java.awt.Color;
import java.awt.Point;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
```

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.ListSelectionModel;
import javax.swing.SwingUtilities;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
/** * * @author Johrman */
public class CertFirmados extends javax.swing.JInternalFrame {
conexionBD BD_solicitudes, BD_repositorios;
ArrayList<InputStream> CRTinputStream = new ArrayList();;
private DefaultTableModel modelo;
ArrayList<conexionBD.campos> Certi;
public CertFirmados(ArrayList<conexionBD.campos> Certificados, conexionBD BD,
conexionBD BDrepo) {
initComponents();
BD_solicitudes = BD;
BD_repositorios = BDrepo;
Certi = Certificados;
try {

File miDir = new File (".");
InputStream BD_inputStream = BD_repositorios.REPO_bajarBD();
byte[] buffer = new byte[BD_inputStream.available()];
BD_inputStream.read(buffer);
File BD_Archivo = new File(miDir.getCanonicalPath() +"\\CA\\index.txt");
BD_Archivo.deleteOnExit();
OutputStream BD_Escritura = new FileOutputStream(BD_Archivo);
BD_Escritura.write(buffer);
BD_Escritura.close();
} catch (IOException ex) {
Logger.getLogger(CertFirmados.class.getName()).log(Level.SEVERE, null, ex);
}
String data[][] ={};
String col[] = {"<html><b>ID Certificado</b></html>","<html><b>Emitido
A</b></html>","<html><b>Estado</b></html>","<html><b>Fecha-Hora
Certificado</b></html>"};
modelo = new DefaultTableModel(data,col) {
@Override
public boolean isCellEditable(int data, int col) {
return false;
} };
jTable1.setModel(modelo);
DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
centerRenderer.setHorizontalAlignment( JLabel.CENTER );
jTable1.getColumnModel().getColumn(0).setCellRenderer( centerRenderer );
jTable1.getColumnModel().getColumn(0).setMaxWidth(101);
jTable1.getColumnModel().getColumn(0).setMinWidth(100);
```

```
jTable1.getColumnModel().getColumn(2).setCellRenderer( centerRenderer );
jTable1.getColumnModel().getColumn(2).setMaxWidth(71);
jTable1.getColumnModel().getColumn(2).setMinWidth(70);
jTable1.getColumnModel().getColumn(3).setCellRenderer( centerRenderer );
jTable1.getColumnModel().getColumn(3).setMaxWidth(161);
jTable1.getColumnModel().getColumn(3).setMinWidth(160);
int cont = 0;
while(cont<Certificados.size()){
modelo.insertRow(cont, new Object[]{});
modelo.setValueAt(Certificados.get(cont).idCer, cont, 0);
modelo.setValueAt(Certificados.get(cont).Emitido_A, cont, 1);
switch (Certificados.get(cont).EstadoCerti) {
case 1:modelo.setValueAt("<html><b><font size=\"3\"
color=\"green\">Válido</font></b></html>", cont, 2); break;
case 2:modelo.setValueAt("<html><b><font size=\"3\"
color=\"green\">Válido</font></b></html>", cont, 2); break;
case 3:modelo.setValueAt("<html><b><font size=\"3\"
color=\"red\">Revocado</font></b></html>", cont, 2); break;
default: modelo.setValueAt("NS/NR", cont, 2); break;
}
CRTinputStream.add(Certificados.get(cont).CSR);
modelo.setValueAt(Certificados.get(cont).Fecha, cont, 3);
cont++;
}
jTable1.addMouseListener(new MouseAdapter() {
public void mousePressed(MouseEvent e) {
if ( SwingUtilities.isRightMouseButton(e)) {
Point p = e.getPoint();
int rowNumber = jTable1.rowAtPoint( p );
ListSelectionModel modelo = jTable1.getSelectionModel();
modelo.setSelectionInterval( rowNumber, rowNumber );
}
if (e.getClickCount()==2){ MostrarCerti(0); }
}
}
);
}
private JPanel MostrarCerti(int Opc){
if((jTable1.getModel().getValueAt(jTable1.getSelectedRow(),
2).toString()).contains("Revocado") ||
((jTable1.getModel().getValueAt(jTable1.getSelectedRow(),
2).toString()).contains("NS/NR"))){
JOptionPane.showMessageDialog(null,"La Solicitud no se puede realizar.","Error al
mostrar Certificado",JOptionPane.ERROR_MESSAGE);
}
else{
try {
File miDir = new File (".");
FileOutputStream fos = new
FileOutputStream(miDir.getCanonicalPath()+"\\newreq.pem");
fos.write(BD_solicitudes.obtenerCertificado(jTable1.getModel().getValueAt(jTable1.ge
tSelectedRow(), 0).toString()));
fos.close();
```

```
Runtime cmd = Runtime.getRuntime();
Process process= cmd.exec("openssl x509 -in "+ miDir.getCanonicalPath()
+"\\newreq.pem -text");
BufferedReader read = new BufferedReader(new
InputStreamReader(process.getInputStream()));
String line, Salida="";
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
}
Salida=Salida+"\n";
Salida=Salida+"-----------------------------------------------------------------
Estructura ASN.1-----------------------------------------------------------\n";
process= cmd.exec("openssl asn1parse -i -in
"+miDir.getCanonicalPath()+"\\newreq.pem");
read = new BufferedReader(new InputStreamReader(process.getInputStream()));
String Aux="";
while((line = read.readLine()) != null){
Salida = Salida+line+"\n";
Aux=Aux+line+"\n";
}
String[] partes = Aux.split("\n");
String[] partesa=partes[35].split(" ");
Aux = partesa[1].substring(1, 13);
partes=null; partesa=null;
Date FechaPasada =(new SimpleDateFormat("yyMMddHHmmss")).parse(Aux);
Date FechaActual = new Date();
if(FechaActual.before(FechaPasada)){
jPanel1 = new javax.swing.JPanel();
jScrollPane1 = new javax.swing.JScrollPane();
jTextArea1 = new javax.swing.JTextArea();
jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jScrollPane1.setViewportView(jTextArea1);
javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup()
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 599,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(0, 0, Short.MAX_VALUE))
);
jPanel1Layout.setVerticalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 231,
javax.swing.GroupLayout.PREFERRED_SIZE)
);
jTextArea1.setBackground(new Color(238,238,238));
jTextArea1.setEditable(false);
jTextArea1.setText(Salida);
jTextArea1.setCaretPosition(0);
jTextArea1.setFont(new java.awt.Font("Tahoma", 0, 12));
jScrollPane1.isWheelScrollingEnabled();
```

```
jTextArea1.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
if(Opc==0){
JOptionPane.showMessageDialog(null,jPanel1,"Información
Certificado",JOptionPane.PLAIN_MESSAGE);
}
}else{
RevocarCertificado(false);
}
} catch (Exception ex) {
JOptionPane.showMessageDialog(null,ex.getMessage(),"Error Information
Certificado",JOptionPane.ERROR_MESSAGE);
}
}
return jPanel1;
}
private void jVerCRTActionPerformed(java.awt.event.ActionEvent evt) {
MostrarCerti(0);
}
private void jRevocarActionPerformed(java.awt.event.ActionEvent evt) {
RevocarCertificado(true);
}
private void RevocarCertificado(boolean Opc){
if(Opc){
if((jTable1.getModel().getValueAt(jTable1.getSelectedRow(),
2).toString()).contains("Revocado") ||
((jTable1.getModel().getValueAt(jTable1.getSelectedRow(),
2).toString()).contains("NS/NR"))){
JOptionPane.showMessageDialog(null,"No se puede proseguir.\nError en la firma o hash
de la CSR","Error",JOptionPane.ERROR_MESSAGE);
}
else{
int opcion=JOptionPane.showConfirmDialog(null,MostrarCerti(1),"¿Realmente desea
revocar este certificado?",JOptionPane.YES_NO_OPTION);
if(opcion==JOptionPane.YES_NO_OPTION){
FileOutputStream fos = null;
File miDir = new File (".");
try {
Runtime cmd = Runtime.getRuntime();
Process process= cmd.exec("openssl ca -key 12345678 -revoke "+
miDir.getCanonicalPath() +"\\newreq.pem");
String Motivo = JOptionPane.showInputDialog("¿Algún motivo para revocar este
certificado?", JOptionPane.QUESTION_MESSAGE);
JOptionPane.showMessageDialog(null,"Certificado
Revocado.","¡Éxito!",JOptionPane.INFORMATION_MESSAGE);
BD_solicitudes.SOLI_actualizarEstadoCRT(3,jTable1.getModel().getValueAt(jTable1.getS
electedRow(), 0).toString(),(new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")).format(new
java.util.Date()),Motivo);
modelo.setValueAt("<html><b><font size=\"3\"
color=\"red\">Revocado</font></b></html>", jTable1.getSelectedRow(), 2);
File BD_Archivo = new File(miDir.getCanonicalPath() +"\\CA\\index.txt");
BD_repositorios.REPO_subirBD(BD_Archivo);
}catch(Exception ex){
System.out.println("error:\n"+ex.getMessage());
```

```
}
}
}
}else{
try {
File miDir = new File (".");
Runtime cmd = Runtime.getRuntime();
cmd.exec("openssl ca -key 12345678 -revoke "+ miDir.getCanonicalPath()
+"\\newreq.pem");
BD_solicitudes.SOLI_actualizarEstadoCRT(3,jTable1.getModel().getValueAt(jTable1.getS
electedRow(), 0).toString(),(new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")).format(new
java.util.Date()),"Certificado vencido");
modelo.setValueAt("<html><b><font size=\"3\"
color=\"red\">Revocado</font></b></html>", jTable1.getSelectedRow(), 2);
File BD_Archivo = new File(miDir.getCanonicalPath()+"\\CA\\index.txt");
BD_repositorios.REPO_subirBD(BD_Archivo);
} catch (IOException ex) {
Logger.getLogger(CertFirmados.class.getName()).log(Level.SEVERE, null, ex);
}
}
}
private javax.swing.JPanel jPanel1;
private javax.swing.JTextArea jTextArea1;
private javax.swing.JPopupMenu jPopupMenu1;
private javax.swing.JMenuItem jRevocar;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JTable jTable1;
private javax.swing.JMenuItem jVerCRT;
}
```

- conexionBD.java

```
package autoridadcertificacion;
import com.mysql.jdbc.jdbc2.optional.MysqlDataSource;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
/** * * @author Johrman */
public class conexionBD {
Connection co;
Statement stm = null;
String Tabla=null;
public conexionBD(String BD, String Ta){
Tabla = Ta;
 MysqlDataSource mysqlDS = null;
```

```
try {
 Class.forName("com.mysql.jdbc.Driver");
 co =
DriverManager.getConnection("jdbc:mysql://localhost/"+BD+"?user=root1&password=");
if(co!=null){stm = co.createStatement();}
} catch(Exception ex){stm = null;
System.out.println(ex.getMessage());
 System.out.println( mysqlDS.getLogWriter());
JOptionPane.showMessageDialog(null,"No se pudo establecer\nconexión con las bases de
datos."+BD,"Error conexión Bases de Datos",JOptionPane.ERROR_MESSAGE);
System.exit(1);
}
}
       public int obtenerTabla(){
String id = "-1";
try {
ResultSet rs = stm.executeQuery("SELECT * FROM solicitudes ORDER BY idCSR DESC");
while (rs.next()){
id = rs.getObject("idCSR").toString();
}
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
return -1;
}
return Integer.parseInt(id);
}
       public ArrayList<campos> obtenerSolictudesNuevas(){
ArrayList<campos> solicitudes = new ArrayList();
try {
ResultSet rs = stm.executeQuery("SELECT * FROM solicitudes");
while (rs.next())
{
//if(rs.getInt("VarCTRL")>=1 && rs.getInt("VarCTRL")<=4 && rs.getInt("VarCTRL")!=2){
if(rs.getInt("VarCTRL")==1){
campos c = new campos();
Blob blob = rs.getBlob("CSR");
c.idCSR = new Long( rs.getObject("idCSR").toString());
c.CSR = new ByteArrayInputStream(blob.getBytes(1, (int)blob.length()));
c.Emitido_A = rs.getObject("EmitidoA").toString();
c.Emitido_por = rs.getObject("EmitidoPor").toString();
rs.getTimestamp("FechaCSR");
c.Fecha = ((new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss")).format(rs.getTimestamp("FechaCSR"))).toUpperCase();
solicitudes.add(c);
}
}
rs.close();
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
return solicitudes;
}
 public ArrayList<campos> obtenerSolictudesRechazadas(){
```

```
ArrayList<campos> solicitudes = new ArrayList();
try {
ResultSet rs = stm.executeQuery("SELECT * FROM solicitudes");
while (rs.next())
{
//if(rs.getInt("VarCTRL")>=1 && rs.getInt("VarCTRL")<=4 && rs.getInt("VarCTRL")!=2){
if(rs.getInt("VarCTRL")==5){
campos c = new campos();
Blob blob = rs.getBlob("CSR");
c.idCSR = new Long( rs.getObject("idCSR").toString());
c.CSR = new ByteArrayInputStream(blob.getBytes(1, (int)blob.length()));
c.Emitido_A = rs.getObject("EmitidoA").toString();
c.Emitido_por = rs.getObject("EmitidoPor").toString();
rs.getTimestamp("FechaCSR");
c.Fecha = ((new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss")).format(rs.getTimestamp("FechaCSR"))).toUpperCase();
solicitudes.add(c);
}
}
rs.close();
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
return solicitudes;
}
public ArrayList<campos> obtenerSolictudesNuevasAutomaticas(){
ArrayList<campos> solicitudes = new ArrayList();
try {
ResultSet rs = stm.executeQuery("SELECT * FROM solicitudes");
while (rs.next())
{
if(rs.getInt("VarCTRL")==1){
campos c = new campos();
Blob blob = rs.getBlob("CSR");
c.idCSR = new Long( rs.getObject("idCSR").toString());
c.CSR = new ByteArrayInputStream(blob.getBytes(1, (int)blob.length()));
c.Emitido_A = rs.getObject("EmitidoA").toString();
c.Emitido por = rs.getObject("EmitidoPor").toString();
rs.getTimestamp("FechaCSR");
c.Fecha = ((new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss")).format(rs.getTimestamp("FechaCSR"))).toUpperCase();
solicitudes.add(c);
}
}
rs.close();
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
return solicitudes;
}
public boolean SOLI_actualizarEstadoCSR(int VarControl, String idCSR){
try {
```

```
PreparedStatement updateEXP = co.prepareStatement("UPDATE `solicitudes` SET
`VarCTRL`="+VarControl+" WHERE `idCSR`="+idCSR);
updateEXP.executeUpdate();
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
return false;
}
return true;
}
public boolean SOLI_actualizarEstadoCRT(int VarControl, String idCSR,String
Fecha,String Motivo){
try {
PreparedStatement updateEXP = co.prepareStatement("UPDATE `certificadosaprobados`
SET `VarCRTL`="+VarControl+", `FechaRevocacion`='"+Fecha+"',
`MovitoRevocacion`='"+Motivo+"' WHERE `IDCertiAprobado`="+idCSR);
updateEXP.executeUpdate();
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
return false;
}
return true;
}
public boolean CERTI_subirCertificado(File CERT_Archivo,String idCSR,String
fechaCSR){
String insert = "insert into
certificadosaprobados(Certificado,FechaCertificado,idCSR,FechaCSR,VarCRTL)
values(?,?,?,?,?)";
FileInputStream fis = null;
PreparedStatement ps = null;
try {
co.setAutoCommit(false);
fis = new FileInputStream(CERT_Archivo);
ps = co.prepareStatement(insert);
ps.setBinaryStream(1,fis,(int)CERT_Archivo.length());
ps.setString(2,(new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")).format(new
java.util.Date()));ps.setString(3, idCSR);
ps.setString(4, fechaCSR);
ps.setInt(5, 1);
ps.executeUpdate();
co.commit();

} catch (Exception ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
return false;
}finally{
try {
ps.close();
fis.close();
} catch (Exception ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
}
return true;
```

```
}
public boolean REPO_subirBD(File BD_Archivo){
String insert = "insert into repositorios(Datos,FechaUpload) values(?,?)";
FileInputStream fis = null;
PreparedStatement ps = null;
try {
co.setAutoCommit(false);
fis = new FileInputStream(BD_Archivo);
ps = co.prepareStatement(insert);
ps.setBinaryStream(1,fis,(int)BD_Archivo.length());
ps.setString(2,(new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")).format(new
java.util.Date()));
ps.executeUpdate();
co.commit();
} catch (Exception ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
return false;
}finally{
try {
ps.close();
fis.close();
} catch (Exception ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
}
return true;
}
public int CancelarPeticion(String idCSR){
 int n=0;
try {
String query = "UPDATE `solicitudes` SET `VarCTRL`=5 WHERE `idCSR`=? and
`VarCTRL`=1;";
PreparedStatement preparedStmt = co.prepareStatement(query);
preparedStmt.setInt(1, Integer.parseInt(idCSR));
n = preparedStmt.executeUpdate();
} catch (SQLException ex) {
n=0;
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
return n;
}
public InputStream REPO_bajarBD(){
InputStream BDindex = null;
try {
ResultSet rs = stm.executeQuery("SELECT Datos FROM repositorios ORDER BY `ID` DESC
LIMIT 1");
while (rs.next()){
Blob blob = rs.getBlob("Datos");
BDindex = new ByteArrayInputStream(blob.getBytes(1, (int)blob.length()));
}
rs.close();
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
```

```
BDindex = null;
}
return BDindex;
}
public ArrayList<campos> obtenerCertificados(){
ArrayList<campos> solicitudes = new ArrayList();
try {
ResultSet rs = stm.executeQuery("SELECT * FROM certificadosaprobados");
while (rs.next())
{
campos c = new campos();
Blob blob = rs.getBlob("Certificado");
c.idCer = new Long( rs.getObject("IDCertiAprobado").toString());
c.CSR = new ByteArrayInputStream(blob.getBytes(1, (int)blob.length()));
c.Fecha = ((new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss")).format(rs.getTimestamp("FechaCertificado"))).toUpperCase();
c.idCSR = new Long( rs.getObject("idCSR").toString());
c.EstadoCerti = rs.getInt("VarCRTL");
c.Emitido_A = "";
solicitudes.add(c);
}
rs.close();
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
return solicitudes;
}
public byte[] obtenerCertificado(String idCert){
byte[] CSR = null;
try {
ResultSet rs = stm.executeQuery("SELECT `Certificado` FROM `certificadosaprobados`
WHERE `IDCertiAprobado`="+idCert);
while (rs.next()){
Blob blob = rs.getBlob("Certificado");
InputStream CSRinputStream = new ByteArrayInputStream(blob.getBytes(1,
(int)blob.length()));
byte[] buffer = new byte[1024];
int bytesRead;
ByteArrayOutputStream output = new ByteArrayOutputStream();
while ((bytesRead = CSRinputStream.read(buffer)) > 0) {
output.write(buffer, 0, bytesRead);
}
CSR=output.toByteArray();
}
} catch (Exception ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
CSR=null;
}
return CSR;
}
public ArrayList<campos> obtenerCertificadosRevocados(){
ArrayList<campos> solicitudes = new ArrayList();
try {
```

```
ResultSet rs = stm.executeQuery("SELECT * FROM certificadosaprobados WHERE
VarCRTL=3");
while (rs.next())
{
campos c = new campos();
Blob blob = rs.getBlob("Certificado");
c.idCer = new Long( rs.getObject("IDCertiAprobado").toString());
c.CSR = new ByteArrayInputStream(blob.getBytes(1, (int)blob.length()));
c.Fecha = ((new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss")).format(rs.getTimestamp("FechaCertificado"))).toUpperCase();
c.idCSR = new Long( rs.getObject("idCSR").toString());
c.EstadoCerti = rs.getInt("VarCRTL");
c.Emitido_A = "";
c.FechaRevocacion = ((new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss")).format(rs.getTimestamp("FechaRevocacion"))).toUpperCase();
c.MotivoRevocacion = rs.getString("MovitoRevocacion");
solicitudes.add(c);
}
rs.close();
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
return solicitudes;
}

public ArrayList<campos> obtenerEmitidoA_CERT(ArrayList<campos> Certificados){
try {
ResultSet rs = stm.executeQuery("SELECT idCSR,EmitidoA FROM solicitudes");
while (rs.next())
{
for (int i =0; i<Certificados.size(); i++){
if(Certificados.get(i).idCSR == rs.getInt("idCSR")){
campos c = new campos();
c.idCer = Certificados.get(i).idCer;
c.CSR = Certificados.get(i).CSR;
c.Fecha = Certificados.get(i).Fecha;
c.idCSR = Certificados.get(i).idCSR;
c.EstadoCerti = Certificados.get(i).EstadoCerti;
c.Emitido_A = rs.getString("EmitidoA");
if( Certificados.get(i).FechaRevocacion!=null){
c.FechaRevocacion = Certificados.get(i).FechaRevocacion;
c.MotivoRevocacion = Certificados.get(i).MotivoRevocacion;
}
Certificados.set(i, c);
}
}
}
rs.close();
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
}
return Certificados;
}
```

```
public class campos {
long idCSR;
long idCer;
InputStreamCSR;
String Emitido_A;
String Emitido_por;
String Fecha;
String FechaRevocacion;
String MotivoRevocacion;
int EstadoCerti;
}
}
```

- AutoridadCertificacion.java

```
package autoridadcertificacion;
public class AutoridadCertificacion {
public static void main(String[] args) {
VentanaAC objeto = new VentanaAC();
conexionBD BD_certi_apro = new conexionBD("autoridad_certificacion_bd",
"certificadosaprobados");
objeto.setLocationRelativeTo(null);
objeto.setTitle("Autoridad de Certificación - Universidad de Pamplona");
objeto.setVisible(true);
}
}
```

B.1.3 Autoridad de Validación

- conexionBD.java

```
package autoridadvalidacion;
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
public class conexionBD {
Connection co;
Statement stm = null;
int ID_Datos;
public conexionBD(String BD){
try {
 Class.forName("com.mysql.jdbc.Driver");
 co =
DriverManager.getConnection("jdbc:mysql://localhost/"+BD+"?user=root3&pas
sword=");
 if(co!=null){stm = co.createStatement();}
```

```
} catch(Exception ex){stm = null;
JOptionPane.showMessageDialog(null,"No se pudo establecer\nconexión con
las bases de datos.","Error conexión Bases de
Datos",JOptionPane.ERROR_MESSAGE);
System.exit(1);
}
}
public InputStream REPO_bajarBD(){
InputStream BDindex = null;
try {
ResultSet rs = stm.executeQuery("SELECT Datos FROM repositorios ORDER BY
`ID` DESC LIMIT 1");
while (rs.next()){
Blob blob = rs.getBlob("Datos");
BDindex = new ByteArrayInputStream(blob.getBytes(1, (int)blob.length()));
}
rs.close();
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
BDindex = null;
}
return BDindex;
}
public void REPO_obtenerID_ultimo(){
try {
ResultSet rs = stm.executeQuery("SELECT ID FROM repositorios ORDER BY
`ID` DESC LIMIT 1");
while (rs.next()){
ID_Datos = rs.getInt("ID");
}
rs.close();
} catch (SQLException ex) {
Logger.getLogger(conexionBD.class.getName()).log(Level.SEVERE, null, ex);
ID_Datos = 0;
}
}
}
```

- VentanaValidacion.java

```
package autoridadvalidacion;
import java.awt.Color;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.text.SimpleDateFormat;
```

```
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
/** * * @author Johrman */
public class VentanaValidacion extends javax.swing.JFrame {
boolean CTRL_PtoEscucha = false, CTRL_Registro = false;
boolean CTRL_OCSP = false, CTRL_IniciaOCSP = false;
int idUltimo = 0;
conexionBD BD_repositorios = new conexionBD("repositorios_certificados");
public VentanaValidacion() {
try {
initComponents();
BD_repositorios.ID_Datos=0;
this.setIconImage(new ImageIcon(getClass().getResource
("/img/icono1.png")).getImage());
jPtoEscucha.setBackground(new Color(238,238,238));
jPtoEscucha.setEditable(false);
jPtoEscucha.setFont(new java.awt.Font("Tahoma", 0, 12));
jScrollPane1.isWheelScrollingEnabled();
jScrollPane2.isWheelScrollingEnabled();
jRegistroOCSP.setBackground(new Color(238,238,238));
jRegistroOCSP.setEditable(false);
jRegistroOCSP.setFont(new java.awt.Font("Tahoma", 0, 12));
jEstado.setText("<html><b><font size=\"3\"
color=\"red\">Detenido</font></b></html>");
File Dir = new File (".");
File BD_Archivo = new File(Dir.getCanonicalPath()+"\\log.txt");
BD_Archivo.deleteOnExit();
} catch (IOException ex) {
Logger.getLogger(VentanaValidacion.class.getName()).log(Level.SEVERE,
null, ex);
}

}
private void jBotonInciarActionPerformed(java.awt.event.ActionEvent evt)
{
BD_repositorios.ID_Datos=0;
idUltimo = 0;
CTRL_OCSP = true;
CTRL_PtoEscucha = true;
(new Hilo_OCSP()).start();
(new Hilo_PtoEscucha()).start();
}

private void jBotonPararActionPerformed(java.awt.event.ActionEvent evt) {
jEstado.setText("<html><b><font size=\"3\"
color=\"red\">Detenido</font></b></html>");
CTRL_PtoEscucha = false;
CTRL_OCSP = false;
CortaProceso();
jPtoEscucha.setText("");
}
```

```
 private class Hilo_OCSP extends Thread{
 boolean CTRL =false;
public void run(){
while(CTRL_OCSP){
try {
BD_repositorios.REPO_obtenerID_ultimo();
if( (idUltimo != BD_repositorios.ID_Datos) && !(idUltimo >=
BD_repositorios.ID_Datos) ){
idUltimo=BD_repositorios.ID_Datos;
DescargarBD();
if(CTRL){CortaProceso(); }
File Dir = new File(".");
Runtime.getRuntime().exec("openssl ocsp -index
"+Dir.getCanonicalPath()+"\\ocsp\\index.txt -port *:8888 -rsigner
"+Dir.getCanonicalPath()+"\\ocsp\\ocsp.tesisjohrmanvides.edu.co.crt -rkey
"+Dir.getCanonicalPath()+"\\ocsp\\ocsp.tesisjohrmanvides.edu.co.key -CA
"+Dir.getCanonicalPath()+"\\ocsp\\cacert.pem -text -out
"+Dir.getCanonicalPath()+"\\log.txt");
CTRL=true;
}
Thread.sleep(5000);
} catch (Exception ex) {
Logger.getLogger(VentanaValidacion.class.getName()).log(Level.SEVERE,
null, ex);
}
}
}
}
private void CortaProceso(){
try {
      Process p = Runtime.getRuntime().exec("taskkill /IM openssl.exe /F
/T");
BufferedReader in = new BufferedReader(
new InputStreamReader(p.getInputStream(),Charset.forName("UTF-8")));
String line = null, Salida = "", hora;
while ((line = in.readLine()) != null) {
Salida = Salida + line +"\n";
}
hora = (new SimpleDateFormat("HH:mm:ss")).format(new java.util.Date());
jPtoEscucha.setText();
jPtoEscucha.setCaretPosition(0);

} catch (IOException ex) {
Logger.getLogger(VentanaValidacion.class.getName()).log(Level.SEVERE,
null, ex);
}
}
      private void DescargarBD(){
try {
File Dir = new File (".");
InputStream BD_inputStream = BD_repositorios.REPO_bajarBD();
byte[] buffer = new byte[BD_inputStream.available()];
BD_inputStream.read(buffer);
```

```
File BD_Archivo = new File(Dir.getCanonicalPath()+"\\ocsp\\index.txt");
BD_Archivo.deleteOnExit();
OutputStream BD_Escritura = new FileOutputStream(BD_Archivo);
BD_Escritura.write(buffer);
BD_Escritura.close();
} catch (IOException ex) {
Logger.getLogger(VentanaValidacion.class.getName()).log(Level.SEVERE,
null, ex); }
}
      private class Hilo_PtoEscucha extends Thread{
File Dir = new File (".");
public void run(){
while(CTRL_PtoEscucha){
try {
Process p = Runtime.getRuntime().exec("netstat -a -n -p tcp");
BufferedReader in = new BufferedReader(
new InputStreamReader(p.getInputStream(),Charset.forName("UTF-8")));
String line = null, Salida = "", hora;
while ((line = in.readLine()) != null) {

Salida = Salida + line +"\n";
}
hora = (new SimpleDateFormat("HH:mm:ss")).format(new java.util.Date());
jPtoEscucha.setText(jPtoEscucha.getText()+"\n\n"+"----------------------
----------------------"+hora+"----------------------------------------
--"+"\n"+Salida);
File Archivo_log = new File(Dir.getCanonicalPath()+"\\log.txt");
Archivo_log.deleteOnExit();
byte[] encoded =
Files.readAllBytes(Paths.get(Dir.getCanonicalPath()+"\\log.txt"));
String Registro = new String(encoded, Charset.defaultCharset());
jRegistroOCSP.setText("");
jRegistroOCSP.setText(Registro);
jRegistroOCSP.setCaretPosition(0);
jEstado.setText("<html><b><font size=\"3\"
color=\"green\">Escuchando...</font></b></html>");
Thread.sleep(10000);
} catch (Exception e) {
e.printStackTrace();
}
}
}
}
private javax.swing.JButton jBotonInciar;
private javax.swing.JButton jBotonParar;
private javax.swing.JLabel jEstado;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
```

```
private javax.swing.JPanel jPanel3;
private javax.swing.JTextArea jPtoEscucha;
private javax.swing.JTextArea jRegistroOCSP;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
}
```

- AutoridadValidacion.java

```
package autoridadvalidacion;
public class AutoridadValidacion {
public static void main(String[] args) {
VentanaValidacion objeto = new VentanaValidacion();
objeto.setLocationRelativeTo(null);
objeto.setTitle("Autoridad de Validación - Universidad de Pamplona");
objeto.setVisible(true);
}
}
```

B.2 Aplicación de Web

- appletSC.java

```
import iaik.pkcs.pkcs11.TokenException;
import iaik.pkcs.pkcs11.wrapper.PKCS11Exception;
import java.applet.Applet;
import javax.swing.JButton;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import javax.swing.JOptionPane;
import netscape.javascript.JSException;
import netscape.javascript.*;
import iaik.asn1.structures.Name;
import java.io.ByteArrayInputStream;
import java.security.cert.CertificateFactory;
import iaik.asn1.ObjectID;
import iaik.pkcs.pkcs11.Mechanism;
import iaik.pkcs.pkcs11.Module;
import iaik.pkcs.pkcs11.Session;
import iaik.pkcs.pkcs11.Slot;
import iaik.pkcs.pkcs11.Token;
import iaik.pkcs.pkcs11.TokenInfo;
import iaik.pkcs.pkcs11.objects.Object;
import iaik.pkcs.pkcs11.objects.PrivateKey;
import iaik.pkcs.pkcs11.objects.X509PublicKeyCertificate;
import iaik.security.provider.IAIK;
import iaik.utils.Util;
import iaik.x509.X509Certificate;
import java.awt.Color;
```

```
import java.awt.Font;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.InputStream;
import java.net.URLEncoder;
import java.security.Security;
import java.util.Collection;
import java.util.Hashtable;
import java.util.Random;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JDialog;
import javax.swing.JLabel;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.URL;
import java.net.URLConnection;
import java.net.URLEncoder;
import javax.swing.BorderFactory;
import javax.swing.border.Border;
import javax.swing.border.CompoundBorder;
import javax.swing.border.EmptyBorder;
import javax.swing.border.LineBorder;
public class appletSC extends Applet {
private static final String SIGN_BUTTON_CAPTION_PARAM = "BotonLeeCert_ax1";
private static final String SITIO_CAPTION_PARAM = "varSitio";
private static final String AUTO_CERTI_CAPTION_PARAM = "varAutoCerti";
private static final String ORGANIZA_CAPTION_PARAM = "varOrga";
private JButton BotonLeeCert;
private Module pkcs11Module;
int auxHilo=0;
VentanaCerti obtenerCertificado;
VentanaPIN pkcs11Dialog;
String[] cabezera = new String[3];
private boolean PriVez = true,AuxControlLogin=true;
public void init() {
String BotonLeeCert_ax1 = this.getParameter(SIGN_BUTTON_CAPTION_PARAM);
cabezera[0]= this.getParameter(SITIO_CAPTION_PARAM);
cabezera[1] = this.getParameter(AUTO_CERTI_CAPTION_PARAM);
cabezera[2] = this.getParameter(ORGANIZA_CAPTION_PARAM);
BotonLeeCert = new JButton("<HTML><font size=\"4\"
color=\"BLACK\">"+BotonLeeCert_ax1+"</font></HTML>");
BotonLeeCert.setLocation(0, 0);
Dimension appletSize = this.getSize();
BotonLeeCert.setSize(appletSize);
BotonLeeCert.setOpaque(false);
BotonLeeCert.setContentAreaFilled(false);
BotonLeeCert.setBorderPainted(false);
Border emptyBorder = BorderFactory.createEmptyBorder();
BotonLeeCert.setBorder(emptyBorder);
BotonLeeCert.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
```

```
Font fuenteOrginal = BotonLeeCert.getFont();
BotonLeeCert.addMouseListener(new MouseAdapter() {
public void mouseEntered(MouseEvent evt){
Font buttonFont=new
Font(BotonLeeCert.getFont().getName(),Font.ROMAN_BASELINE,BotonLeeCert.getFont().get
Size());
BotonLeeCert.setFont(buttonFont);
BotonLeeCert.setText("<HTML><font size=\"4\"
color=\"BLUE\"><U>"+BotonLeeCert_ax1+"</U></font></HTML>");
}
public void mouseExited(MouseEvent evt){
BotonLeeCert.setFont(fuenteOrginal);
BotonLeeCert.setText("<HTML><font size=\"4\"
color=\"BLACK\">"+BotonLeeCert_ax1+"</font></HTML>");
}
});
BotonLeeCert.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
leerCertificado();
}
});
this.setLayout(null);
this.add(BotonLeeCert);
}
private String buscarLiberiaTarjInt(){
String lineSeparator = System.getProperty("path.separator");
String libraryPath = System.getProperty("java.library.path");
for (String dir : libraryPath.split(lineSeparator)) {
File f = new File(dir + "/" + "acospkcs11.dll");
 if (f.exists()){
return f.getAbsolutePath();
}
}
return null;
}
private boolean InicializarModulo(String DireccionLibreria) throws TokenException {
boolean Aux=false;
 try {
 pkcs11Module = Module.getInstance(DireccionLibreria);
 pkcs11Module.initialize(null);
 Aux=false;
 } catch (IOException ex) {
JOptionPane.showMessageDialog(null, "Error:\nNo se puede inicar el dispositivo",
"Error Grave", JOptionPane.ERROR_MESSAGE);
Aux=true;
 }catch (PKCS11Exception es) {
JOptionPane.showMessageDialog(null, "Dispositivo no detectado:\nFavor conecte un
tarjetero", "Error Grave", JOptionPane.ERROR_MESSAGE);
Aux=true;
 }
 return Aux;
 }
private void leerCertificado() {
```

```
String DireccionLibreria = null;
try {
if (PriVez){
DireccionLibreria = buscarLiberiaTarjInt();
if(DireccionLibreria==null){
JOptionPane.showMessageDialog(null, "Error:\nNo se halló lector compatible", "Error
Grave", JOptionPane.ERROR_MESSAGE);
}else {PriVez=InicializarModulo(DireccionLibreria);}
}
if (!PriVez){
if(auxHilo==0){new Hilo(1,DireccionLibreria).start(); }
Slot[] slots = pkcs11Module.getSlotList(Module.SlotRequirement.TOKEN_PRESENT);
if (slots.length == 0) {
JOptionPane.showMessageDialog(null, "Error:\nNo se halló Tarjeta Insertada", "Error
Grave", JOptionPane.ERROR_MESSAGE);
}else{
Slot selectedSlot = slots[0];
Token token = selectedSlot.getToken();
TokenInfo tokenInfo = token.getTokenInfo();
Session session = token.openSession(Token.SessionType.SERIAL_SESSION,
Token.SessionReadWriteBehavior.RW_SESSION, null, null);
pkcs11Dialog = new
VentanaPIN(tokenInfo.getLabel(),selectedSlot.getSlotInfo().getSlotDescription());
boolean AuxSeleccion, AuxValidadorContrase=true;
String oldButtonLabel = BotonLeeCert.getText();
BotonLeeCert.setText("Esperando...");
BotonLeeCert.setEnabled(false);
if (AuxControlLogin){
while(AuxValidadorContrase){
try { AuxSeleccion = pkcs11Dialog.run(); } finally {pkcs11Dialog.dispose(); }
if (AuxSeleccion) {
try {
String pinCode = pkcs11Dialog.obtenerPIN();
if (tokenInfo.isLoginRequired()) {
try{
session.login(Session.UserType.USER, pinCode.toCharArray());
AuxValidadorContrase=false; AuxControlLogin=false;
}catch(PKCS11Exception ex){
JOptionPane.showMessageDialog(null, "Contraseña no Valida", "Contraseña erronea",
JOptionPane.WARNING_MESSAGE);
AuxValidadorContrase=true; AuxControlLogin=true;}
}
} catch (Exception e) { }
} else { AuxValidadorContrase=false; }
}
}
if(AuxControlLogin){BotonLeeCert.setText(oldButtonLabel);
BotonLeeCert.setEnabled(true);}
else{
BotonLeeCert.setText("Leyendo Tarjeta...");
int i;
session.findObjectsInit(null);
Object[] objects1 = session.findObjects(1);
```

```
String[] IDObjetos = new String[100];
int contIDObjetos = 0;
int PosCert =0;
while (objects1.length > 0) {
Object object = objects1[0];
String aux = object.toString();
String[] parts = aux.split("\n");
if (parts[0].contains("Private")) {IDObjetos[contIDObjetos]=parts[6].substring(6);
contIDObjetos++;}
if (parts[0].contains("Public")) {IDObjetos[contIDObjetos] = parts[6].substring(6);
contIDObjetos++;}
if (parts[0].contains("Certificate")) {
if(parts[8].contains("hex")){IDObjetos[contIDObjetos]= parts[8].substring(12);
PosCert=8; contIDObjetos++;} else {IDObjetos[contIDObjetos]=
parts[12].substring(12); PosCert = 12; contIDObjetos++;}

}
objects1 = session.findObjects(1);
}
BotonLeeCert.setText("Leyendo Certificados...");
contIDObjetos = 0;
int cont;
Vector<String> IDnoRepe = new Vector<String>();
while (IDObjetos[contIDObjetos]!=null){
cont = 0;
for (i=contIDObjetos; IDObjetos[i]!=null ; i ++ ){
if (IDObjetos[i].equalsIgnoreCase(IDObjetos[contIDObjetos])){ cont++; }
}
if(cont==3){IDnoRepe.add(IDObjetos[contIDObjetos]);}
contIDObjetos++;
}
int NumeroCertificados = 0;
contIDObjetos=0;
Long[] HandleObjCert = new Long[20];
Long[] HandleObjPriv = new Long[20];
String[] NombreObjCert = new String[20];
session.findObjectsInit(null);
objects1 = session.findObjects(1);
Security.addProvider(new IAIK());
CertificateFactory certificateFactory = CertificateFactory.getInstance("X.509",
"IAIK");
X509Certificate[] currentCertificate = new iaik.x509.X509Certificate[20];
Hashtable objectHandleToObject = new Hashtable(10);
while (objects1.length > 0) {
Object object = objects1[0];
objectHandleToObject.put(new Long(object.getObjectHandle()), object);
String aux = object.toString();
String[] parts = aux.split("\n");
if(parts[0].contains("Private")){HandleObjPriv[contIDObjetos]=object.getObjectHandle
();contIDObjetos++;}
if (parts[0].contains("Certificate")) {
for (i=0;i<IDnoRepe.size();i ++ ){
if(IDnoRepe.get(i).equalsIgnoreCase(parts[PosCert].substring(12))){
```

```
HandleObjCert[NumeroCertificados] = object.getObjectHandle();
if(parts[4].substring(9).contains("<NULL_PTR>")) {NombreObjCert[NumeroCertificados]
= "";} else {NombreObjCert[NumeroCertificados] = parts[4].substring(9);}
byte[] encodedCertificate1 = ((X509PublicKeyCertificate)
object).getValue().getByteArrayValue();
InputStream fileInputStream = new ByteArrayInputStream(encodedCertificate1);
Collection certificateChain =
certificateFactory.generateCertificates(fileInputStream);
iaik.x509.X509Certificate x509Certificate = (iaik.x509.X509Certificate)
certificateChain.iterator().next();
currentCertificate[NumeroCertificados] = x509Certificate; NumeroCertificados++;
break;
}
}
}
objects1 = session.findObjects(1);
}
session.findObjectsFinal();
BotonLeeCert.setText("Esperando....");
AuxSeleccion=false;
obtenerCertificado = new
VentanaCerti(tokenInfo.getLabel(),NumeroCertificados,HandleObjCert,NombreObjCert,cur
rentCertificate,cabezera);
try { AuxSeleccion = obtenerCertificado.run(); } finally
{obtenerCertificado.dispose(); }
if (!AuxSeleccion) {
BotonLeeCert.setText(oldButtonLabel); BotonLeeCert.setEnabled(true);
}else{
Long certificadoSeleccionadoHandle = obtenerCertificado.obtenerCertificado();
Object certifiSeleccionado = (Object)
objectHandleToObject.get(certificadoSeleccionadoHandle);
Long handlePrivaCertiSeleccionado =
HandleObjPriv[obtenerCertificado.obtenerNumeroCertificado()];
Object CprivadaSeleccionada = (Object)
objectHandleToObject.get(handlePrivaCertiSeleccionado);
PrivateKey correspondienteCPriv=(PrivateKey) CprivadaSeleccionada;
if (certifiSeleccionado instanceof X509PublicKeyCertificate) {
try {
byte[] encodedCertificate = ((X509PublicKeyCertificate)
certifiSeleccionado).getValue().getByteArrayValue();
byte[] signature = Util.Base64Encode(encodedCertificate);
String Certicate = new String(signature);
Certicate="-----BEGIN CERTIFICATE-----\n"+Certicate+"\n-----END CERTIFICATE-----";
String Reto = intercambiosConPHP(Certicate,"10","10101010"); //Enviando
certificado...
if(Reto.length()==512){
Reto=DescifrarRetoConClavePrivada(correspondienteCPriv,Reto,session);
JOptionPane.showMessageDialog(null,"RETOOO\n"+Reto);
String Var =intercambiosConPHP(Certicate,Reto,"21010101");
if(Var.equalsIgnoreCase("6")){
Var=intercambiosConPHP("12345678",Reto,"31010101");
}
}
```

```
BotonLeeCert.setText(oldButtonLabel); BotonLeeCert.setEnabled(true);
} catch (Exception ex) {
System.out.println("Could not decode this X509PublicKeyCertificate. Exception is: "
+ ex.toString());
}
}
}
}
}
}
} catch (SecurityException se) {
se.printStackTrace();
JOptionPane.showMessageDialog(null,"No se puede acceder al lector.\n"
 + "Esta aplicación debe ejecutarse sin restricciones\n"
 + "Por favor! Acepte este Applicación como de confianza, cuando se navegador lo
solicite");
} catch (JSException jse) {
jse.printStackTrace();
JOptionPane.showMessageDialog(null,"No se pudo accder a varibales PHP.\n"
+ "Por favor! contacte al administrador.");
} catch (Exception e) {
e.printStackTrace();
JOptionPane.showMessageDialog(null, "Error inesperado: "+e.getMessage());
}
}
private String DescifrarRetoConClavePrivada(PrivateKey correspondienteCPriv, String
Reto, Session session){
byte[] retoByte=cadenaHexa_A_arregloByte(Reto);
try {
session.decryptInit(Mechanism.RSA_PKCS, correspondienteCPriv);
} catch (TokenException ex) {
JOptionPane.showMessageDialog(null,ex.getMessage(), "Error1",
JOptionPane.ERROR_MESSAGE);
}
byte[] retoDescifrado = null;
try {
retoDescifrado = session.decrypt(retoByte);
} catch (TokenException ex) {
JOptionPane.showMessageDialog(null,ex.getMessage(), "Error2",
JOptionPane.ERROR_MESSAGE);
}
return (new String(retoDescifrado));
}
privatebyte[] cadenaHexa_A_arregloByte(String s) {
int len = s.length();
byte[] data = new byte[len / 2];
for (int i = 0; i < len; i += 2) {
data[i/2] = (byte) ((Character.digit(s.charAt(i), 16) << 4) +
Character.digit(s.charAt(i+1), 16));
}
return data;
}
private void redirigir(){
```

```
JSObject window = JSObject.getWindow(this);
window.eval("ejecutarVentana()");
}
private String intercambiosConPHP(String CerticadoPEM, String Reto, String
Decision){
try {
String data =URLEncoder.encode("Certificado", "UTF-8") + "=" +
URLEncoder.encode(CerticadoPEM, "UTF-8")
 + "&" + URLEncoder.encode("VariableDecision", "UTF-8") + "=" +
URLEncoder.encode(Decision, "UTF-8")
 + "&" + URLEncoder.encode("Reto", "UTF-8") + "=" + URLEncoder.encode(Reto, "UTF-
8");

URL url = new URL("https://www.tesisjohrmanvides.edu.co/autentica-bd_iaca.php");
URLConnection conn = url.openConnection();
conn.setDoOutput(true);
OutputStreamWriter wr = new OutputStreamWriter(conn.getOutputStream());
wr.write(data);
wr.flush();
BufferedReader rd = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
String line;
while ((line = rd.readLine()) != null) {
JOptionPane.showMessageDialog(null,"LIne00000000 \n"+line.length());
if(line.length()==512){JOptionPane.showMessageDialog(null,"LIneeee \n"+line); return
line;}
else if(line.equalsIgnoreCase("6")){JOptionPane.showMessageDialog(null,"Logueo
Éxitoso\n-pulse para continuar-"); return line;}
else if(line.equalsIgnoreCase("10")){pkcs11Module.finalize(); redirigir();}
else if(line.equalsIgnoreCase("2")){pkcs11Module.finalize();
JOptionPane.showMessageDialog(null,"Certificado Revocado", "Error",
JOptionPane.WARNING_MESSAGE);}
else if(line.equalsIgnoreCase("7")){pkcs11Module.finalize();
JOptionPane.showMessageDialog(null,"Error:\nIdentidad presentada Errónea", "Error",
JOptionPane.WARNING_MESSAGE);}
else if(line.equalsIgnoreCase("3")){pkcs11Module.finalize();
JOptionPane.showMessageDialog(null,"Certificado no valido", "No Valido",
JOptionPane.WARNING MESSAGE);}
else if(line.equalsIgnoreCase("5")){pkcs11Module.finalize();
JOptionPane.showMessageDialog(null,"El Servidor OCSP\nno resolvió la consulta",
"Error", JOptionPane.ERROR_MESSAGE);}
else if(line.equalsIgnoreCase("4")){pkcs11Module.finalize();
JOptionPane.showMessageDialog(null,"no se pudo comprobar\nla legitimidad de la
repuesta OCSP", "Error", JOptionPane.ERROR_MESSAGE);}
else if(line.equalsIgnoreCase("9")){pkcs11Module.finalize();
JOptionPane.showMessageDialog(null,"Error al Iniciar sesión.\nNo se puede conectar a
la BD", "Error", JOptionPane.ERROR_MESSAGE);}
else if(line.equalsIgnoreCase("8")){pkcs11Module.finalize();
JOptionPane.showMessageDialog(null,"Error al Iniciar sesión.\nUsuario NO EXISTE",
"Error", JOptionPane.ERROR_MESSAGE);}
else if(line.contains("ilegal")){pkcs11Module.finalize();
JOptionPane.showMessageDialog(null,"ERROR:\nSe ha presentado un problema con\nel
certificado del servidor", "Error", JOptionPane.ERROR_MESSAGE);}
```

```
break;
}
wr.close();
rd.close();
} catch (Exception e) {
} catch (Throwable ex) {
Logger.getLogger(appletSC.class.getName()).log(Level.SEVERE, null, ex);
}
return null;
}
private class Hilo extends Thread{
int b; String c;
private Hilo(int a, String DireccionLibreria){b=a;c=DireccionLibreria;}
public void run(){
while(b==1){
try {
if(!PriVez){Slot[] slots =
pkcs11Module.getSlotList(Module.SlotRequirement.TOKEN_PRESENT);
if(slots.length==0){b=0;pkcs11Module.finalize(); PriVez=true; AuxControlLogin=true;
pkcs11Dialog.dispose();
if(BotonLeeCert.getLabel().contentEquals("Esperando...."))){obtenerCertificado.dispos
e();}; JOptionPane.showMessageDialog(null,"tarjta desconectada"); }

}
Thread.sleep(1000);
} catch (TokenException ex) {
Logger.getLogger(appletSC.class.getName()).log(Level.SEVERE, null, ex);
} catch (InterruptedException ex) {
Logger.getLogger(appletSC.class.getName()).log(Level.SEVERE, null, ex);
} catch (Throwable ex) {
Logger.getLogger(appletSC.class.getName()).log(Level.SEVERE, null, ex);
}
}
}
}
}
```

- autentica-bd_iaca.php

```
<?php
if( (isset($_POST['VariableDecision'])) && (isset($_POST['Certificado']))
&& (isset($_POST['Reto'])) && (isset($_SERVER['SSL_SERVER_I_DN_CN'])) ){
      $VariableDecision=$_POST['VariableDecision'];
      if($VariableDecision==10101010){
            echo (AutenticaUNO());
      } elseif ($VariableDecision==21010101){
            echo( AutenticaDOS());
      } elseif ($VariableDecision==31010101 &&
apc_fetch('j')==$_POST['Reto']){
            echo (Autorizar());
      }
       else {echo "<center><h2> intento de acceso ilegal
</h2></center>";}
```

```
}
else {echo "<center><h2> intento de acceso ilegal </h2></center>";}
?>
<?php
function AutenticaUNO(){

      $CertiUsuario=$_POST['Certificado'];
      $cert_raiz = 'C:\xampp\htdocs\tesis\certs\emisor.pem';
      $URL_COMPROBACION = '_____';
      $emisor = 'C:\xampp\htdocs\tesis\certs\emisor.pem';
      $a = rand(1000,99999);
      $cliente=$a.'cliente.pem';
      file_put_contents($cliente, $CertiUsuario);
      $comprobacion = shell_exec('openssl ocsp -CAfile '.$cert_raiz.' -
issuer '.$emisor.' -cert '.$cliente.' -url '.$URL_COMPROBACION);
      unlink($cliente);
      if($comprobacion!=null){
            $comprobacion2 = preg_split('/[\r\n]/', $comprobacion);
            $comprobacion3 = preg_split('/: /', $comprobacion2[0]);
             $resultado = $comprobacion3[1];
                  switch ($resultado){
                        case 'good':
                              $reto=cifrar($CertiUsuario);
                              return $reto;
                              break;
                        case 'revoked':
                              return "2";
                              break;
                        default:
                           return "3";
                  }
      }
      else {return "5";}
}

function cifrar($cert){
      $pub_key = openssl_pkey_get_public($cert);
      $keyData = openssl_pkey_get_details($pub_key);
$textoClaro
=substr(str_shuffle("0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOP
QRSTUVWXYZ/*-+.!#$%&/?\@"), 0, 32);
      apc_store('j',$textoClaro,100);
      openssl_public_encrypt($textoClaro, $crypttext, $keyData['key']);
      //file_put_contents('johr1', $crypttext);
      return convertirHexa($crypttext);
}
function convertirHexa($data){
    $hexi   = '';
    $dump   = '';
    $len    = strlen($data);
    for ($i = $j = 0; $i < $len; $i++){
  $hexi .= sprintf("%02x", ord($data[$i]));
  if (++$j === 16 || $i === $len - 1) {
```

```
$dump .= sprintf("%s",$hexi);
$hexi  = '';
$j     = 0;
  }
    }
      return $dump;
}
?>
<?php
function AutenticaDOS(){
      if($_POST['Reto']==apc_fetch('j')){
            return "6";
      } else {return "7";}
}
?>
<?php
function Autorizar(){
      require('sesion.class.php');
      $session = new session();
      $session->inicio_sesion('Usuario');
      $_SESSION['codAcceso'] = '123';
      return "10";
}

function conectaBaseDatos(){
      try{
            $servidor = "localhost";
            $puerto = "3306";
            $basedatos = "bd_iaca";
            $usuario = "root";
            $contrasena = "";
            $conexion = new
PDO("mysql:host=$servidor;port=$puerto;dbname=$basedatos",
$usuario, $contrasena, array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES
utf8"));
$conexion->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
return $conexion;
}
catch (PDOException $e){ return "8"; }
}

?>
```