DESARROLLO DE SOFTWARE PARA FACTURACIÓN ELECTRÓNICA BAJO LA RESOLUCIÓN No. 042 DEL 2020 DE LA DIRECCIÓN DE IMPUESTOS Y ADUANAS NACIONALES

AUTOR

DARWIN MANUEL MERCADO CERPA

PROGRAMA DE INGENIERÍA EN TELECOMUNICACIONES
DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA, ELÉCTRICA, SISTEMAS Y TELECOMUNICACIONES
FACULTAD DE INGENIERÍAS Y ARQUITECTURAS

UNIVERSIDAD DE PAMPLONA

PAMPLONA, 10 DE MAYO DEL 2021

DESARROLLO DE SOFTWARE PARA FACTURACIÓN ELECTRÓNICA BAJO LA RESOLUCIÓN No. 042 DEL 2020 DE LA DIRECCIÓN DE IMPUESTOS Y ADUANAS NACIONALES

DARWIN MANUEL MERCADO CERPA

Trabajo de grado presentado como requisito para optar al título de INGENIERO EN TELECOMUNICACIONES

Director: JOHRMAN DE JESÚS VIDES NIÑO,
MsC. Seguridad Informática

PROGRAMA DE INGENIERÍA EN TELECOMUNICACIONES
DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA, ELÉCTRICA, SISTEMAS Y TELECOMUNICACIONES
FACULTAD DE INGENIERÍAS Y ARQUITECTURAS
UNIVERSIDAD DE PAMPLONA
PAMPLONA, 10 DE MAYO DEL 2021

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

PROGRAMA DE INGENIERÍA EN TELECOMUNICACIONES
TRABAJO PRESENTADO PARA OPTAR POR EL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

TEMA:

DESARROLLO DE SOFTWARE PARA FACTURACIÓN ELECTRÓNICA BAJO LA RESOLUCIÓN No. 042 DEL 2020 DE LA DIRECCIÓN DE IMPUESTOS Y ADUANAS NACIONALES

FECHA DE INICIO DEL TRABAJO:
FECHA DE TERMINACIÓN DEL TRABAJO:

NOMBRES Y FIRMAS DE AUTORIZACIÓN PARA LA SUSTENTACIÓN:

_____

DARWIN MANUEL MERCADO CERPA
AUTOR

_____	_____
JOHRMAN DE JESÚS VIDES NIÑO	**HERNANDO JOSÉ VELANDIA**
DIRECTOR	**DIRECTOR DE PROGRAMA**

JURADO CALIFICADOR:

_____	_____
GERMAN PORTILLA	**KELLY TORRES**
ING ELETRÓNICO	**ING TELECOMUNICACIONES**

RESUMEN

El gobierno de Colombia reglamentó el uso de la factura electrónica de manera obligatoria para empresas e incluso personas que cumplen ciertas características tributarias, esta tecnología se implementó con el objetivo de simplificar los procesos tributarios de los contribuyentes y tener un mayor control fiscal, este proceso es dirigido por la dirección de impuestos y aduanas nacionales (DIAN). El desarrollo de este proyecto se realiza con el objetivo de que las personas o pequeñas empresas que están obligadas a facturar electrónicamente puedan aplicar estas técnicas e implementar su propio sistema de facturación electrónica, se detallan aspectos del formato UBL-2.1 para generar las facturas electrónicas de ventas, se describen procesos para generar la firma digital XAdES y autenticación por WS-SECURITY por medio del protocolo SOAP los cuales son los procesos esenciales para lograr emitir facturas electrónicas ante la DIAN.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

CAPÍTULO I: INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

Con la expedición del decreto número 2242 del 2015, se inició la adopción de una nueva tecnología en el territorio colombiano la cual traerá beneficios tanto para el país como para empresas y personas que hacen uso de esta tecnología, esta

consiste en la implementación de un nuevo mecanismo de facturación que resulta ser la evolución de factura convencional, esta tendrá la misma validez fiscal que una factura de papel. Esto es un avance hacia la modernización de Colombia y se lleva acabo con el fin de simplificar el proceso de reporte de pago de los impuestos de todas las empresas contribuyentes en Colombia y generar un mayor recaudo de impuestos. Esta tecnología ya es tendencia en diferentes países como México, Perú, Chile, Ecuador, entre otros.

En el artículo 615 del estatuto tributario se dispone; Todas las personas o entidades que tenga la calidad comerciante, ejerzan profesiones liberales o presten servicios inherentes a éstas, o enajenen bienes o productos de la actividad agrícola o ganadera, deberán expedir factura electrónica, y deberán conservar copia de la misma por cada una de las operaciones de comercio que realicen, independientemente de su calidad de contribuyentes o no contribuyentes. [1]

Decreto 358 del 2020 artículo 1.6.1.4.2 sujetos obligados a expedir factura de venta o documento equivalente. [2]

Se encuentran obligados a expedir factura de venta o documento equivalente por todas y cada una de operaciones realicen, siguientes sujetos:

- Quienes sean responsables del impuesto sobre las ventas IVA.
- Quienes sean responsables del impuesto nacional al consumo.
- Todas las personas o entidades que tengan la calidad comerciante, ejerzan profesiones liberales o presten servicios inherentes a éstas, enajenen bienes o productos de la actividad agrícola o ganadera, deberán expedir factura electrónica, y deberán conservar copia de la misma por cada una de las operaciones de comercio que realicen, independientemente de su calidad de contribuyentes o no contribuyentes previstos en los artículos 616-2 inciso 4 del parágrafo 2 y parágrafo 3 del artículo 437 y 512-13 del estatuto tributario y en artículo 1.6.1.4.3. del decreto 358 del 2020.
- Los comerciantes importadores o prestadores de servicios o en las ventas a consumidores finales

- Los tipógrafos y litógrafos que no sean responsables del impuesto sobre las ventas IVA, de acuerdo al parágrafo 3 del artículo 437 del estatuto tributario, por el servicio prestado de conformidad con lo previsto en el artículo 618 del estatuto tributario.
- Los contribuyentes inscritos en el impuesto unificado bajo el régimen simple de tributación simple.

Artículo 652-1 del estatuto tributario, sanción por no facturar electrónicamente; Artículo modificado por el artículo 56 de la Ley 6 de 1992. Quien esté obligado a expedir facturas y no lo haga, podrán ser objeto de sanción de clausura o cierre del establecimiento de comercio, oficina o consultorio, o sitio donde se realice la actividad, profesión u oficio de conformidad con lo dispuesto en los artículos 657 y 658 del estatuto tributario. [1]

Los sujetos mencionados anteriormente se encuentran obligados a expedir factura de venta como se indica en el decreto 358 del 2020 y quienes no realicen esta actividad estando obligado puede incurrir en sanciones y multas.

En Colombia la DIAN presenta 3 modos de facturar electrónicamente:

- **Software gratuito de la Dian**
  Este es un software gratuito que la DIAN a puesto a disposición de los colombianos para que puedan emitir sus facturas electrónicas, solo requiere de un dispositivo que tenga acceso a internet.
- **Proveedor tecnológico**
  El contribuyente puede contratar los servicios de un proveedor tecnológico habilitado por la DIAN.
- **Software propio**
  Pueden desarrollar su propio software de facturación, se requiere conocimientos técnicos de desarrollo. Para tal fin la DIAN dispone de una "Caja de herramientas" (paquete de documentación y ejemplos de facturas),

con esta información se puede implementar un desarrollo propio del software.

Aunque podamos optar por la opción más fácil y utilizar el software gratuito este puede presentar la grabe limitante por su usabilidad y limitaciones de tiempo para generar grandes cantidades de facturas dificultando este proceso.

El segundo método que podemos utilizar es por medio de un proveedor tecnológico, los proveedores tecnológicos prestan sus servicios a personas o entidades a cambio de un pago mensual que puede variar según la cantidad de facturas y usuarios, el principal problema que se puede tener al momento de escoger un proveedor tecnológico es que la entidad o persona, se convertirá dependiente de una empresa tercera para la gestión y seguridad de sus datos y la de sus clientes. Igualmente se expone al riesgo de que la información privada de su negocio sea filtrada por causas ajenas o problemas del proveedor tecnológico, además de incurrir en gastos mensuales para realizar este proceso.

Por tercer método tenemos el desarrollo propio que puede ser la mejor opción a optar por entidades o personas debido a que este software puede ser diseñado justamente a las necesidades de su negocio mejorando la usabilidad, dependencia de su negocio, otro gran beneficio es que no contaran con limitación en la cantidad de facturas que pueden emitir.

1.2 JUSTIFICACIÓN

El desarrollo de este proyecto surge de la necesidad que tienen las personas o entidades obligadas a facturar electrónicamente, para no incurrir en sanciones y multas ante el estado. Y la necesidad de contar un sistema que permite evitar los problemas presentes en el modo de facturación gratuito y por proveedor tecnológico, mejorando la usabilidad (con el objetivo de tener un sistema fácil de usar y fácil de adaptarse a las necesidades de los negocios y personas), gestión de datos (ser su propio administrador de la información de su negocio y la de sus

clientes, para no convertirse en una entidad dependiente), capacidad para generar facturas (con el objetivo de no tener límites para emitir facturas)

Con el desarrollo de este proyecto se busca facilitar una guía, con el objetivo de que las personas o empresas puedan implementar su propio sistema de facturación electrónica, disminuyendo las dificultades y gastos asociados durante el desarrollo de este y hacer cumplimiento del decreto 358 del 2020 y no incurrir en sanciones.

1.3 DELIMITACIÓN

1.3.1 OBJETIVO GENERAL

Desarrollar un software para emisión de facturas electrónicas por medio de la resolución No. 000042 05 de mayo del 2020

1.3.2 OBJETIVOS ESPECÍFICOS

1) Fundamentar los aspectos técnicos aplicables de la normativa para la emisión de facturas electrónicas de ventas.
2) Diseñar un software que permita la emisión de facturas electrónicas de ventas a través de Firma XAdES.
3) Vincular el software diseñado con los servicios web de la DIAN mediante el uso del protocolo SOAP permitiendo la emisión y registro de facturas electrónicas.
4) Validar el software de facturación electrónica a través del set pruebas para habilitación definidos por la DIAN ante sus servicios web.

1.4 ACOTACIONES

En el anexo técnico de factura electrónica Versión 1.7.-2020 descrito por la DIAN la facturación electrónica abarca 5 tipos de documentos los cuales son:

- Invoice (Factura)
- CreditNote (Nota Crédito)

- DebitNote (Nota Débito)
- ApplicationResponse (Registro de Evento)
- AttachedDocument (Contenedor Electrónico)

Sin embargo, Este proyecto se centrara en la factura electrónica de venta (Invoice); A pesar que el software a desarrollar tenga las características finales para ser apto de lanzar a producción, en este trabajo de grado no se abordaran emisiones de facturas de venta con software en entorno de producción, solo serán emitidas dentro del set de prueba debido a que las facturas quedan registradas en los sistemas contables de la DIAN con ventas de productos no existentes por tanto implicaría afecciones tributarias. además, el autor del trabajo quedaría obligado a emitir facturas. y esto se debe a que si una persona no está obligada a facturar electrónicamente y opta por facturar electrónicamente de manera voluntaria este tendrá que asumir todas las obligaciones de una persona obligada a facturar.

las acotaciones respecto a tecnologías y procesos son las siguientes:

- Este proyecto se desarrollará en entorno web con el lenguaje de programación PHP del lado del backend y Vuejs del lado del frontend.
- Para la gestión de los datos se utilizará el gestor de bases de datos postgreSQL.
- El proceso de validación se llevará a cabo con los resultados o respuestas obtenidas por el sistema de validación de la DIAN, el cual indica el cumplimiento satisfactorio de los requisitos de la resolución No. 000042 05 de mayo del 2020 [3] para emitir facturas de ventas (invoice) o el rechazo por el no cumplimiento.

CAPÍTULO II: MARCO TEÓRICO

Antes de poder emitir facturas electrónicas de ventas se deben tener conocimiento en diferentes aspectos fundamentales como:

- **XML:** Este metalenguaje permite crear la estructura de las facturas electrónicas.
- **Canonical XML:** Permite crear transformaciones necesarias para aplicar la firma digital.
- **Formatos estándares de comercio electrónico:** Conocer los formatos estándares definidos por oasis para el comercio electrónico.
- **Funciones Hash:** Funciones criptográficas que permiten calcular la huella digital de archivos.
- **Criptografía:** Conocer los métodos de cifrados y sus aplicaciones.
- **Formatos estándares para firmas digitales:** Distinguir entre los diferentes formatos estándares para firmas digitales.
- **Protocolos de transmisión:** Conocer el formato estándar de comunicación necesario para la emisión de las facturas.

por lo que en capitulo se abordaran los aspectos necesarios para emitir facturas electrónicas.

2.1 XML (EXTENSIBLE MARKUP LANGUAGE)

XML del acrónimo lenguaje extensible de etiquetado, es una sintaxis universal para la descripción y estructuración de datos en un documento. XML fue planteado por WC3 (World Wide Web Consortium) en el año 1996 y en 1998 lanza su primera versión “XML 1.0”, [4] está basada en el estándar SGML (Standard Generalized Markup Language, ISO 8879), que data del año 1986. Éste a su vez en GML (Generalized Markup Language), creado por IBM en 1969.

2.1.1 Conceptos generales XML

XML se propone como un formato estándar para la transmisión de datos estructurados entre diferentes equipos, este formato debe cumplir principalmente 2 criterios, primero este debe estar bien formado.

Un documento XML se dice que está bien formado si se cumple con las siguientes normas:

- Los documentos deben seguir una estructura jerárquica de elementos. Una etiqueta debe estar correctamente incluida en otra, es decir de haber un correcto anidamiento.
- Los documentos XML requieren de un elemento raíz o padre, del que todos los demás sean parte, es decir, la jerarquía de elementos sólo puede tener un elemento inicial. Como se observa en la ilustración 2.
- Los elementos no nulos o vacíos tienen siempre una etiqueta inicial y una etiqueta final de cierre.
- Los elementos vacíos deben referirse en su notación abreviada, ejemplo <elemento/> agregando “/” justo antes del cierre de la etiqueta.
- Los valores de atributos siempre deben estar encerrados entre comillas simples o dobles.
- XML es sensible a mayúsculas y minúsculas. Es decir, se debe hacer referencia a este tal y como se declaró.

Como segundo criterio debe ser válido, en general, un documento válido es un documento cuyo contenido obedece a las restricciones expresadas en un esquema particular como puede ser un DTD o un XSD [5].

2.1.1.1 Estructura de un documento XML

Un documento XML la mayoría de las veces puede estar conformado por 2 partes que se conocen como document prolog donde se especifica información correspondiente al XML y document Elements donde se inserta todos los datos del documento [6], como se muestra en la siguiente ilustración.

	<?xml version="1.0" encoding="UTF-8" standalone="no"?>	Document prolog
	<Materia> <Alumnos> <Maestros/> </Alumnos> </Materia>	Document elements

Ilustración 1 estructura XML

En el document prolog se agregan atributos como:

- **Versión:** Este atributo indica que versión de XML se utiliza.
- **Encoding**: Especifica la codificación del documento. Por defecto se usa la codificación UTF-8 aunque existen otras, como UTF-16, ISO-10646-UCS-2, ISO-10646-UCS-4, ISO-8859-1 en ISO-8859-9, ISO-2022-JP, Shift_JIS, EUC-JP.
- **Standalone**: Indica si el documento va acompañado de un DTD, (yes) indica que el procesador de XML debe usar la DTD para la validación y (no) si no es necesario validar un DTD, Su presencia no es necesaria, ya que más tarde se indicará el DTD en caso de ser necesario.

Para la declaración del document prolog se deben tener en cuenta criterios como:

- La declaración XML es sensible a mayúsculas y minúsculas y esta debe empezar en minúscula.

- La declaración del prolog debe ser especificada siempre en la primera declaración del documento XML.

Es importante distinguir entre la estructura lógica y física de un documento XML, se puede definir la estructura lógica como la relación o agrupación entre distintos datos como se muestra en la ilustración 2 donde los elementos se distribuyen de manera jerárquica, la estructura física se define como la apariencia del documento, incluyendo la ubicación de los elementos y la tipografía empleada.

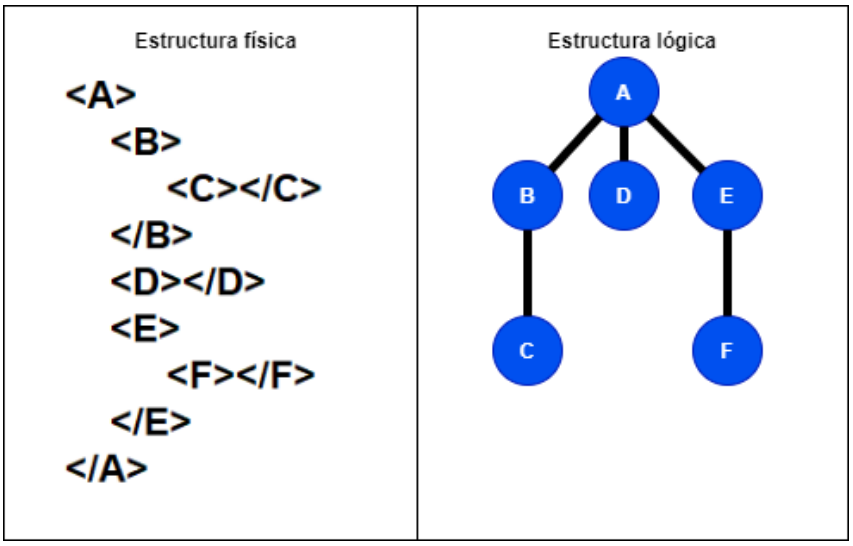


Ilustración 2 estructura física y lógica

No se debe confundir un metalenguaje como XML con un lenguaje de programación, los lenguajes y metalenguajes de marcado no son capaces de realizar operaciones aritméticas, estos lenguajes son esencialmente utilizados para usos como:[6]

- Simplificar la creación de documentos HTML (HyperText Markup Language) para los sitios web de gran tamaño, incluso es usado para definir la interfaz gráfica de aplicaciones en Android.
- Puede ser usado para intercambiar información entre las organizaciones y sistemas.

- Se puede usar para la descarga y recarga de información en bases de datos.
- Se puede usar para almacenar y organizar información
- Diferentes tipos de documentos pueden ser expresado como un documento XML.

2.1.1.2 XML Namespace (espacios de nombre)

Namespace es una recomendación desarrollada por W3C [7] que surge de la necesidad de solucionar la colisión de nombres, este problema se presenta cuando existen etiquetas con mismo nombre. Aunque puedan tener diferente significado, como se muestra en la ilustración 3. Los namespace o espacio de nombres son un mecanismo para evitar conflictos de nombres al diferenciar elementos o atributos dentro de un documento XML que pueden tener nombres idénticos, pero definiciones diferentes, los espacios de nombres son identificados por URI (Uniform Resource Identifiers); La declaración de un espacio de nombre se define por medio de la siguiente expresión **xmlns:prefix = " URI "** donde prefix es el nombre de dicho espacio.

```
<carta>
   <destinatario/>
   <direccion/>
</carta>

<carta>
   <menu>
      <carne/>
   </menu>
</carta>
```

Ilustración 3 ejemplo conflicto de nombre

La declaración del espacio de nombre se identifica por medio de una URI única para cada namespace declarado, haciendo referencia en cada elemento en el cual pertenece como se muestra en la ilustración 4. Normalmente una URI (Uniform Resource Identifiers) Es una cadena de texto que Identifica un recurso de manera única e inequívoca,[8] tiene la apariencia de una URL (Uniform resource locator) la cual es utilizada para ubicar recursos en internet, habitualmente se usan URL de páginas de internet, pero no quiere decir que tenga que ser una URL de algo existente, el estándar no indica nada sobre que tenga que existir la URL indicada. W3C decidió utilizar las URL

como namespace porque contienen los nombres de dominio que son únicos en Internet.[8]

```
<restaurante xmlns:ms="URI"  xmlns:mn="URI">

        <ms:carta>
            <ms:destinatario/>
            <ms:direccion/>
        </ms:carta>

        <mn:carta>
              <mn:menu>
                  <mn:carne/>
              </mn:menu>
        </mn:carta>

<restaurante>
```

Ilustración 4 ejemplo de Namespace

2.1.1.3 DTD (Document type definition)

Como se mencionó anterior mente un documento XML debe cumplir los criterios de estar bien formado y de ser un documento válido por medio de un DTD o XSD. Un DTD o definición de tipo de documento cumple la función de validar un documento XML haciendo cumplimiento de unas reglas de estructura. Reglas como la existencia de elementos, numero de ocurrencia, atributos, valores y principalmente la jerarquía del documento.

2.1.1.4 XSD (XML Schema Definition)

XSD es otra herramienta como DTD que permite validar documentos XML, XSD es un conjunto de componentes como definiciones de tipo y declaraciones de elementos. La principal diferencia entre ellos, es que DTD se usa para definir la estructura, mientras que XSD se usa para definir la estructura y el contenido del documento.[5] El propósito de un XSD es definir las reglas de construcción de un documento XML, indica que elementos y atributos pueden aparecer en un documento, el número de veces de ocurrencias, el orden de los elementos secundarios, los tipos de datos y los valores predeterminados para elementos y atributos.

Las principales ventajas de usar XSD son:

- **Tipado fuerte:**

  Se conoce como tipado fuerte a los lenguajes que no permiten la violación de los tipos de datos. En un DTD, las posibilidades de limitar el contenido de los elementos y atributos son muy limitado.

- **Tipos de Datos:**

  Permite declarar tipos de datos, como suele ocurrir en POO donde podemos crear tipos de datos de nuestras clases. el usuario puede definir sus propios tipos de datos a partir de los existentes en el XSD.

- **Cantidad de ocurrencia:**

  Permite controlar el número de ocurrencias de los elementos con mayor precisión.

2.1.1.5 DOM (Document Object Model)

Se encuentra también asociado a la recomendación del W3C DOM “Document Object Model”,[9] El Modelo de objetos de documento (DOM) es una API de programación para documentos HTML y XML. Define la estructura lógica de los documentos y la forma en que se accede y se manipula un documento. Un objetivo importante del DOM es proporcionar una interfaz de programación estándar que se pueda utilizar en distintos entornos y lenguajes de programación. Gracias a esta API los programadores pueden crear, modificar, eliminar y navegar por la estructura de documentos XML sin importar el lenguaje de programación.

2.1.2 Canonical XML

El metalenguaje XML es flexible en aspectos de sintaxis lo que presenta problemas para garantizar la integridad de los datos en un documento XML; Canonical XML surge de la necesidad de mantener la integridad de la información del documento sin tener en cuenta los cambios sintácticos presentes en el. Gracias a esto un documento puede ser lógicamente similar pero físicamente diferente y aun así mantener la integridad de la información del documento.

Canonical es la recomendación de la W3C[10] también conocida comúnmente como C14N, esta permite comparar pares de documentos para determinar su equivalencia mediante transformaciones, estas eliminan las diferencias no significativas entre los documentos[11]. Esta recomendación indica las transformaciones que permite convertir cualquier documento XML al formato canónico y compararlo byte a byte, esto es muy útil al aplicar firmas digitales, debido a que la firma debe ser validada ante los mismos bits con los que se aplicó la firma. Esto quiere decir que antes de aplicar una firma digital se debe primero calcular la forma canónica de los elementos a firmar para posterior mente aplicar la firma. Esto también es una de sus ventajas permite aplicar canonización a elementos y no al documento como un todo.

Canonical realiza las principales transformaciones[11]:

- Normaliza los saltos de línea a

- Cada valor de los atributos se normaliza.
- Se reemplazan las referencias a entidades analizadas y de caracteres.
- Los elementos de tipo cdata se reemplazan con su contenido de caracteres.
- Se elimina el document prolog
- Los elementos nulos o vacíos se convierten en pares de etiquetas de inicio y fin, ejemplo <element/> quedara como <element></element>
- Se deben normalizar los espacios en blanco fuera del elemento del documento y dentro de las etiquetas de inicio y finalización.
- Se reservan todos los espacios en blanco en el contenido de los caracteres.
- Los valores del atributo se establecen entre comillas dobles
- Los caracteres especiales dentro de los valores de los atributos y el contenido de los caracteres se reemplazan por referencias de caracteres.
- Se omiten las declaraciones de espacio de nombres.
- Se agregar los atributos adicionados por defecto.
- Se impone el orden lexicográfico de las declaraciones y atributos del espacio de nombres de cada elemento.

La canonicalización XML está diseñada para ser usada en aplicaciones que requieren comprobar la integridad de la información de un documento o elementos.[11] Esto se hace comparando la forma canónica del documento original con la forma canónica del documento después de aplicarle este método. Por ejemplo, una firma digital sobre la forma canónica de un documento o elementos XML, permite que los cálculos de resumen no tengan en cuenta los cambios en la representación física del documento original, durante el proceso de firma el resumen se calcula sobre la forma canónica del documento. Luego, para comprobar la firma se deberá calcular el resumen de la forma canónica del documento, donde ambos deberán coincidir para garantizar la integridad de la información presente el documento.

2.2 UBL-2.1 (UNIVERSAL BUSINESS LANGUAGE)

Desde la aprobación como recomendación del W3C en 1998, XML se ha adoptado en varias industrias como marco para la definición de intercambiado de mensajes en el comercio electrónico[12]. El uso de XML ha llevado al desarrollo de múltiples versiones para diferentes tipos de industrias, documentos básicos como facturas, órdenes de compra, avisos de envío entre otros.

No contar con formatos específicos en la industria tienen desventajas significativas como: [13]

- Desarrollar y mantener las diferentes versiones de documentos comerciales.
- Establecer y mantener múltiples adaptadores que permita las relaciones comerciales a través de los límites del dominio.
- Tener múltiples formatos XML dificulta la integración de los mensajes comerciales XML con los sistemas administrativos.
- Tener múltiples formatos XML hace que las herramientas sean más costosas y dificulta encontrar trabajadores capacitados.

UBL (Universal Business Language) fue desarrollado para ayudar a resolver estos problemas mediante la definición de un formato estándar para los documentos comerciales que se puede restringir o ampliar para cumplir con los requisitos de industrias específicas. Esencialmente UBL proporciona lo siguiente: [13]

- Un grupo de objetos comerciales estructurados, su semántica asociada expresada como componentes de datos reutilizables y documentos comerciales comunes.
- Facilita una biblioteca de esquemas XML para componentes de datos reutilizables como artículo, pago, dirección, entre otros elementos comunes en los documentos comerciales cotidianos.
- Un grupo de esquemas XML para documentos comerciales comunes como Factura, Pedido, y Aviso de envío que se crean a partir de los componentes de la biblioteca UBL.

Tener una base estándar para esquemas comerciales XML ofrece las siguientes ventajas:[13]

- Menor costo de integración entre empresas y dentro de ellas, mediante la reutilización de una estructura de datos comunes.
- Menores costos y tiempo de desarrollo del software comercial, porque el software deberá solo procesar un numero de etiquetas dado.
- Rápido aprendizaje, los usuarios solo necesitan dominar una sola biblioteca.
- Menor costo de entrada, lo que facilita una adopción más rápida por parte de pequeñas y medianas empresas (PYME).
- Capacitación estandarizada, lo que facilita encontrar empleados calificados.
- grupos de integradores de sistemas disponible universalmente.
- Herramientas de datos estandarizadas de entrada y salida.
- Permite tener un objetivo estándar para software comercial de bajo costo y listo para usar.

UBL está diseñado para proporcionar una sintaxis estandarizada y entendida universalmente para documentos comerciales legalmente vinculantes y para operar

dentro de un marco comercial estándar como la ISO 15000[14], UBL está disponible gratuitamente para todos sin gravámenes legales o tarifas de licencia.

Los XSD de UBL son las únicas representaciones normativas de los tipos de documentos y componentes de la biblioteca de UBL para fines de validación y conformidad de documentos. [13]

En UBL se pueden encontrar diferentes tipos de documentos como se muestra en la siguiente tabla, cabe aclarar que los documentos descritos en la tabla no son todos los documentos que posee UBL-2.1, la información descrita fue tomada de la documentación OASIS Standard donde podrá acceder a la información completa sobre UBL-2.1.[13]

Attached Document	Este es un contenedor UBL que permite empaquetar un documento de cualquier tipo con el documento UBL que lo hace referencia.
Credit Note	Este documento es utilizado para especificar los créditos adeudados al Deudor por parte del Acreedor.
Debit Note	este documento es utilizado para especificar las deudas contraídas por el Deudor.
Invoice	Este es un documento utilizado para solicitar el pago.
Application Response	Este es un documento para indicar la respuesta de una transacción. Esta puede ser una respuesta comercial, respuesta técnica, enviada automáticamente por una aplicación.

Tabla 1 Documentos UBL 2.1

La biblioteca de UBL y los esquemas de documentos están destinados a respaldar los intercambios de información empresarial y comercial por lo que es sumamente importante garantizar la seguridad de la información, así como debe ser validado un documento ante su esquema correspondiente es necesario firmar electrónicamente un documento UBL.[13] Este puede ser el caso al crear licitaciones o facturas. Por ejemplo, en algunos países la ley exige la firma digital de facturas electrónicas.

UBL tiene una estructura de datos conocida como Signature que permite insertar firmas electrónicas y una serie de elementos para usar dichas firmas en un documento, con UBL se pueden utilizar estándares de firma digital como XMLDSIG (XML Signature).[13] lo que permite la utilización de XAdES, firmas electrónicas avanzadas XML (ETSI TS 101 903).

2.3 CODIFICACIÓN BASE64

En los inicios de internet surgió la necesidad de definir un lenguaje por el cual las computadoras pudieras comunicarse. Los fabricantes de computadoras utilizaban diferentes métodos para comunicar sus equipos, pero al final no podían comunicarse equipos de diferentes fabricantes. En mayo de 1961, un ingeniero de IBM, Bob Bemer, envió una propuesta al American National Standards Institute (ANSI) para desarrollar un código único para la comunicación informática llamado ASCII (Código Estándar Estadounidense para el Intercambio de Información), en aquel entonces cuando trabajaban con hardware de 7 bits solo podían representar 2^7 es decir 128 caracteres. ASCII paso por muchas modificaciones hasta tener el código ASCII extendido el cual trabaja con 8 bits. debido a la expansión de internet y los cambios que tenían las computadoras se presentaban problemas de compatibilidad. para dar soluciona esto se ideo un modo de codificación llamado BASE el cual está definido en el RFC 4648 [15] donde se definen 3 tipos de codificación BASE16, BASE32, BASE64.

Base64 aún sigue siendo ampliamente utilizado, se usa más comúnmente para codificar datos binarios como archivos multimedia. para insertarlos en páginas web. Además, Base64 se utiliza para codificar datos que pueden no ser compatibles o dañados durante la transferencia, o el almacenamiento. Estas son algunas de las aplicaciones del algoritmo:

- Adjunte de archivos al para envió por medio del protocolo SMTP
- Incrustar imágenes en HTML a través de URI de datos
- Conservar bytes sin procesar de funciones criptográficas
- Salida de datos binarios como XML o JSON en las respuestas de la API
- Transferencia de datos por internet

Ahora veamos como es el proceso de codificación y decodificación en BASE64. Para codificar primero que todo asumamos el dato que queremos codificar para este ejemplo consideremos “Pamplona”

- **Paso 1**
  Primero debemos convertir cada carácter en el código correspondiente según la tabla ASCII como se muestra a continuación.
  P = **80**, a = **97**, m = **109**, p = **112**, l = **108**, o = **111**, n = **110**, a = **97**
- **Paso 2**
  Ahora debemos convertir cada código a binario (base 2), es importante representarlo en 8 bits
  80 = **01010000**, 97 = **01100001,** 109 = **01101101,** 112 = **01110000,**
  108 = **01101100,** 111 = **01101111,** 110 = **01101110,** 97 = **01100001**
- **Paso 3**
  Ahora debemos unir todos estos valores para formar una trama, se deben formar en el mismo orden del dato original
  010100000110000101101101011100000110110001101111011011100110001

- **Paso 4**
  Tomamos las tramas del paso 3 y dividimos en grupos 6 bits
  [010100], [000110], [000101], [101101], [011100], [000110], [110001], [101111], [011011], [100110], [0001]

  Si observamos vemos que nos sobraron 4 bits, para estos casos debemos agregar tantos ceros a la derecha como sea necesario para completar el grupo de 6 bits.
  [010100], [000110], [000101], [101101], [011100], [000110], [110001], [101111], [011011], [100110], [000100]
  para este caso vemos que fueron necesarios 2 bits este número es importante para representar el resultado al final.

- **Paso 5**

  Ahora debemos convertir los valores obtenidos en decimal

010100 = **20**, 000110 = **6**, 000101 = **5**, 101101 = **45**, 011100 = **28**, 000110 = **6**, 110001 = **49**, 101111 = **47**, 011011 = **27**, 100110 = **38**, 000100 = **4**

Seguido tomamos los valores decimales que acabamos de calcular y lo representamos por su equivalente como se muestra en la tabla

Valor	Carácter	Valor	Carácter	Valor	Carácter	Valor	Carácter
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Ilustración 5 tabla base64

20 = **U,** 6 = **G,** 5 = **F,** 45 = **t,** 28 = **c,** 6 = **G,** 49 = **x,** 47 = **v,** 27 = **b,** 38 = **m,** 4 = **E**

Ahora formamos la salida uniendo cada carácter obtenido, obtendremos el siguiente resultado **UGFtcGxvbmE** ahora para finalizar demos tener en cuenta la cantidad de ceros que se agregó en paso 4 los cuales fueron 2 dígitos 0, por cada par de cero, es decir por cada 2 ceros debemos agregar un símbolo = al resultado, este proceso solo se lleva a cabo cuando se adicionan ceros.

RESULTADO: **UGFtcGxvbmE=**

Hemos logrado codificar el dato “Pamplona” que obtuvimos como resultado **UGFtcGxvbmE=** ahora podemos proceder a decodificar el cual es aplicar el proceso inverso como se muestra a continuación:

- **Paso 1**
  Para decodificar debemos separar el resultado en caracteres Excluyendo los símbolos =

  U, G, F, t, c, G, x, v, b, m, E
  Haciendo uso de la tabla base64 procedemos a hallar los valores de cada carácter **U, G, F, t, c, G, x, v, b, m, E,**

  20, 6, 5, 45, 28, 6, 49, 47, 27, 38, 4

- **Paso 2**
  Una vez tenemos el valor decimal de cada carácter debemos convertirlos a binario, importante convertir a binario de 6 dígitos.

  20 = **010100**, 6 = **000110**, 5 = **000101**, 45 = **101101**, 28 = **011100**, 6 = **000110**, 49 = **110001**, 47 = **101111**, 27 = **011011**, 38 = **100110**, 4 = **000100**

- **Paso 3**
  Creamos una trama de bits con todos los valores anteriores

  010100000110000101101101011100000110110001101111011011100110000100

- **Paso 4**

  Ahora separamos en grupos de 8 bits, si te diste cuenta para pasar de ascii a base64 estuvimos que agrupar en grupos de 6 bits y si haces 2^6 te darás cuenta que da como resultado 64 de ahí su nombre, y si haces 2^8 veras que da 256 que son los caracteres presentes en, tabla ascii

  [01010000], [01100001], [01101101], [01110000], [01101100], [01101111], [01101110], [01100001], [00]

Vemos que obtenemos los ceros que se agregaron en la codificación estos deben ser ignorados para este proceso

- **Paso 5**

  Convertimos cada valor binario obtenido del paso anterior

  01010000 = **80,** 01100001 = **97,** 01101101 = **109,** 01110000 = **112,** 01101100 = **108,** 01101111 = **111,** 01101110 = **110,** 01100001 = **97**

Por ultimo buscamos el carácter correspondiente en tabla ascii

P = **80**, a = **97**, m = **109**, p = **112**, l = **108**, o = **111**, n = **110**, a = **97**

RESULTADO = **Pamplona**

2.4 FUNCIONES HASH

Las funciones hash son algoritmos que permiten calcular el resumen de datos y archivos, este resumen se denomina huella digital la cual permite identificar datos, archivos, de manera única.

Un hash consisten en una función computable que se obtiene a partir de un mensaje M de tamaño variable y da como salida una representación de tamaño fijo del propio mensaje, llamada H(M) [16], como se muestra en la siguiente ilustración.

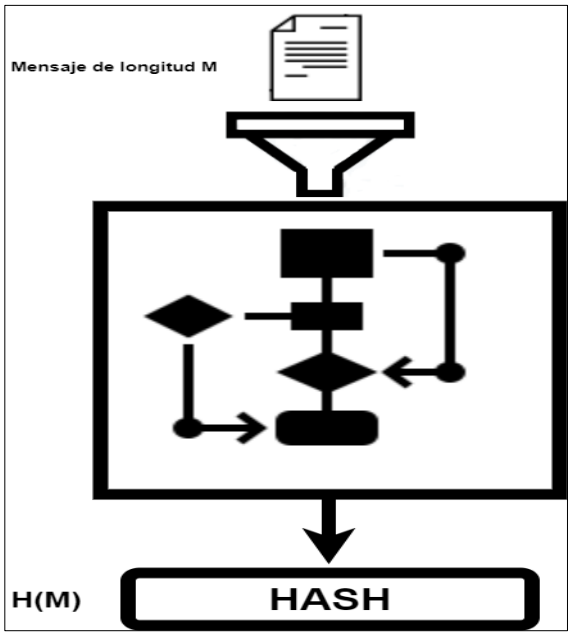


Ilustración 6 Funcion hash

El resultado se conoce como su resumen o hash. Este valor es la representación compacta del mensaje de Entrada.

El termino hash proviene del significado de los verbos del inglés cortar y mezclar, ya que las funciones hash cortan y mezclan la entrada para obtener la salida. Una función hash H debe cumplir que para cualquier resumen H(M) es difícil encontrar el mensaje M que produzca dicho resumen. También se conocen como funciones unidireccionales ya que solo se puede avanzar en un sentido. La funcionalidad de hash radica en que para un

mensaje en específico solo existirá un hash único. Si un solo bit de la entrada cambia se obtendrá un hash totalmente distinto.

Una función hash debe cumplir los siguientes criterios para considerarse segura:[16]

- **Unidireccionalidad:** son funciones fáciles de calcular, pero imposible computacionalmente revertir. Es decir que a partir del resumen H(M) es imposibles hallar el mensaje M.
- **Avalancha:** para una entrada se generar un hash único, si a esta misma entrada se le cambie por lo menos un bit el resumen debería cambiar por los menos la mitad de los bits.[16]
- **Resistencia a colisiones:** una colisión se da cuando existen 2 mensajes M y M' que dan como resultado el mismo hash es decir H(M) = H(M'), como existen infinitos mensajes se puede concluir que hay infinitas posibilidades que 2 mensajes colisionen.[16] Uno de los requisitos de las funciones hash es que sea lo menos probables que existan 2 mensajes con el mismo hash.

Entre las funciones hash más utilizadas encontramos[16]:

- **Message Digest(MD):**
  MD o resumen del mensaje es una familia de, en la familia MD tenemos algoritmos como md2, md4, md5, md6 [17] estos producen una resumen de 128 bits Actualmente no es recomendado usar esta familia debido al número de colisiones que estos presenta
- **SHA**
  Esta es otra familia de algoritmos hash que se dividen en 2 grupos
  - **SHA1**
    Este algoritmo fue diseñado por la Agencia de Seguridad Nacional (NSA) y publicadas por NIST(National Institute of Standards and Technology) en el año del 1995, este fue inspirado en MD4, [16]SHA1 produce a su salida un resumen de 160 bits, este algoritmo fue roto recientemente por google por lo que ya no se recomienda su uso.
  - **SHA2**
    Este surgió como una mejora de SHA1, fue estandarizado por el NIST en el 2001, Esta familia de algoritmos está conformada por SHA-224, SHA-256, SHA-384 Y SHA-256 los cuales producen un resumen de 224,256,384,512 bits, esta familia de funciones hash son consideradas

seguras actualmente y tienen gran popularidad en el ámbito de seguridad.

2.5 CRIPTOGRAFÍA

La palabra criptografía proviene de cripto (ocultar) – grafia (escritura), esta surge de la necesidad de ocultar información ante observadores no autorizados. Se define la criptografía como " la ciencia que estudian las técnicas empleadas para la transmisión o almacenamiento de información de forma segura , de modo que no pueda ser leída o modificada por usuarios que no estén autorizados a hacerlo"[18] así como existe la ciencia de criptografía existe su contraparte la cual es el criptoanálisis el cual se define como "ciencia que se encarga de analizar los sistemas criptográficos para intentar buscar la forma de acceder a la información original que se está ocultando, sin ser usuarios autorizados para hacerlo"[18] este se encarga de buscar vulnerabilidades de los criptosistemas a partir de datos cifrado, si este alcanza su objetivo se dice que el algoritmo o el criptosistema está roto, un criptosistema se conoce como el conjunto de pasos o procedimientos lógicos, que permiten cifrar información.

Dentro de la criptografía tenemos 2 grupos que se conocen como criptografía simétrica y asimétrica:

- **Cifrado simétrico:**
  El cifrado simétrico hace uso de una única clave que se utiliza para cifrar y descifrar. La clave que se utiliza debe permanecer en secreto y solo debe ser conocida por quien deba cifrar y descifrar el mensaje, de aquí surge otro gran problema que se denomina "problema de distribución de claves" y esto se da porque el receptor también debe conocer la clave y esta debe ser enviada por medio de un canal inseguro. Esto es solucionado gracias a una de los usos del cifrado asimétrico.

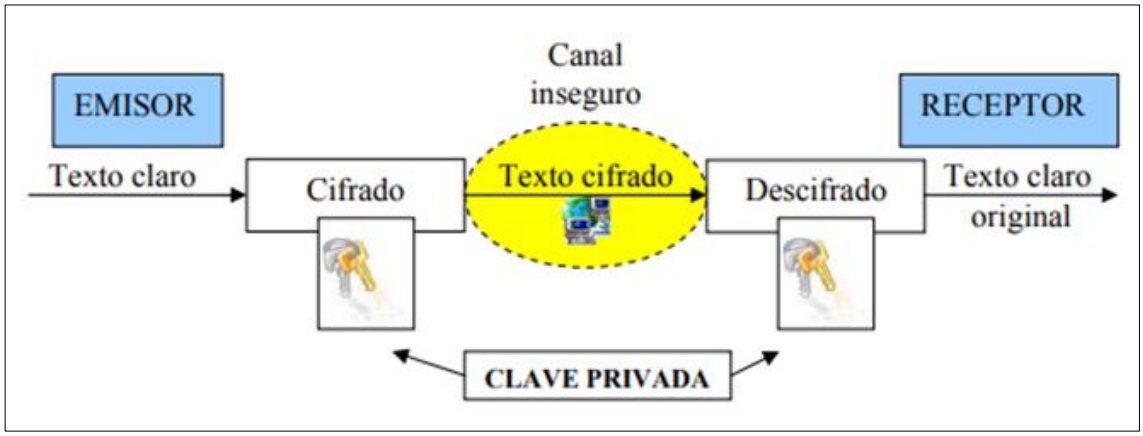


Ilustración 7 cifrado simétrico; tomado de [19]

- **Cifrado asimétrico:**

  El cifrado asimétrico hace uso de un par de claves, una para cifrar y otra para descifrar, tener 2 claves tiene ventajas frente al cifrado simétrico una de ellas es la distribución de claves.

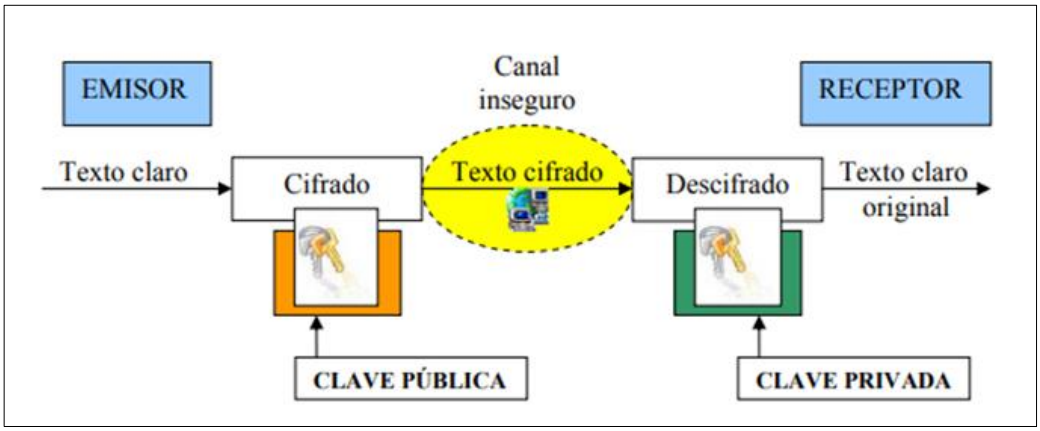


Ilustración 8 cifrado asimétrico; tomado de [20]

CIFRADO ASIMETRICO

Como se mencionó anteriormente el cifrado asimétrico o de clave pública hace uso de un par de clave denominas clave pública y privada, la clave pública puede ser compartida o subida en repositorios y ser de acceso libre pero la clave privada debe permanecer en secreto.[16] Esto permite funcionalidades extras frente al cifrado simétrico que sería muy difícil o casi imposible de lograr con el cifrado simétrico. Este cifrado permite realizar 3 operación básicas las cuales son:

- **Cifrado:**
  Al igual que el cifrado simétrico este permite cifrar información con el objetivo de ocultar la información ante observadores no autorizados, la gran diferencia radica en que se utilizaran claves distintas para cifrar y descifrar, para lograr cifrar información esto debe llevarse a cabo con la clave pública, y esto se debe a que solo el destinatario es quien conoce la clave privada y solo el podrá descifrar la información.

- **Firma:**
  La firma se realiza con el objetivo de garantizar la autenticidad de datos o información, este proceso se realiza de manera inversa al proceso de cifrado. Para llevar a cabo el proceso de firma se utiliza la clave privada y la clave pública se utiliza para comprobar la firma. Como la clave privada permanece en secreto solo podrá firmar información quien tienen acceso a esta. La firma puede ser comprobado por cual quiera que conozca la clave pública, la cual puede ser de acceso libre. Si la información es capturada podrán tener acceso a esta y comprobar su firma. es importante aclarar que la firma garantiza autenticidad mas no confidencialidad si se quiere confidencialidad primero se debe firmar y luego cifrar.
- **Distribución de claves:**

  Como se mencionó anteriormente el cifrado simétrico hace uso de una única clave la cual debe ser distribuida al receptor para que este pueda descifrar la información, esto se facilita gracias al cifrado asimétrico, la clave que se quiere compartir debe ser cifrada con la clave pública del receptor con el objetivo de solo el receptor pueda descifrar dicho mensaje.

Aunque la criptografía asimétrica tiene ciertas ventajas frente a criptografía simétrica, esta también tiene sus desventajas, una de ellas es que suele ser muy lento en comparación a la simétrica.[16] por esto es usualmente usada para distribución de claves, por ejemplo, en el proceso de firma no se suele firmar todo el documento y esto se debe a que entre más grande sean los datos más tardado será este proceso, por este motivo se suele calcular el hash del documento y es a este al que se le aplica la firma, realizando este proceso obtendremos dos indicadores primero el hash nos garantizara la integridad de la información y la firma garantizara la autenticidad o autoría. En caso de que estos sean modificados y firmados nuevamente por una persona no autorizada, podremos comprobar la integridad de la información por medio del hash donde este debe coincidir con el hash calculado originalmente.

Algoritmo de Cifrado Asimétrico RSA

Este algoritmo fue propuesto en el año 1978 por Ronald **R**ivest, Adi **S**hamir y Leonard **A**dleman, de ahí su nombre RSA, este algoritmo estuvo patentado hasta el año 2000, por lo cual se debía pagar por su uso, el funcionamiento de este algoritmo reside en el problema que existe de factorizar números primos muy grandes. En la actualidad suele ser computacionalmente imposible por lo que es considerado seguro. Su seguridad se ve amenazada por el desarrollo de las computadoras cuánticas las cuales indican tener la capacidad de hacer estas operaciones con un menor costo computacional hasta que esto ocurra se podrá usar tranquilamente RSA.

2.6 FIRMA DIGITAL

El comercio electrónico ha surgido como el nuevo modo de adquirir bienes, productos y servicios por lo que la confianza en este modo de hacer negocios es fundamental para el éxito y el desarrollo del comercio electrónico. Por lo tanto, es esencial que las empresas que utilizan este medio electrónico para hacer negocios cuenten con mecanismos de seguridad adecuados para proteger sus transacciones. Por este motivo han surgido estándares como XML Encryption [20] y XML-Signature [21] definidos por la W3C los cuales especifican los procesos para firmar y cifrar documentos XML con el objetivo de garantizar la integridad y la autenticidad.

Firmar documentos de manera electrónica se ha vuelto esencial en operaciones comerciales y de autoría, por ejemplo, en caso de crear licitaciones, facturas o cualquier documento electrónico que requiera garantizar la autoría de quien los emite, incluso en algunos países la ley exige la firma digital de facturas electrónicas.

Las firmas digitales permiten garantizar aspectos como:[13]

- **Integridad**: garantiza que el documento no ha sido modificado desde que fue firmado.
- **Autenticidad**: se certifica la identidad de la parte que crea la firma del documento.
- **No repudio**: el firmante del documento no puede negar su participación en la creación o aprobación.
- **Anterioridad**: se permite asociar un sello de tiempo a la firma, la cual permite conocer el tiempo de firma del documento.

2.6.1 XMLDSIG

El estándar XMLDSIG establece un marco general para la firma digital de documentos XML, define las reglas y la sintaxis de procesamiento de firmas XML para proporcionar integridad y autenticación de mensajes.[21]

Dentro de los tipos de firmas XMLDSIG se pueden encontrar las siguientes:[21]

- **Enveloped**: La firma se aplica al contenido que es externo al elemento Signature y se puede identificar mediante un URI o una transformación.
- **Enveloping**: La firma se aplica al contenido que se encuentra dentro del elemento Object de la propia firma.
- **Detached**: La firma se aplica al contenido XML que contiene al elemento Signature. para implementaciones de firmas envueltas se debe tener cuidado de no incluir la firma en el cálculo del valor de la firma.

Los mismos tipos de firmas están admitidos por XAdES, que es otro estándar de firma XML que extiende y amplía a XMLDSIG con la inclusión datos útiles para el proceso de validación como sello de tiempo, listas de revocación de certificados. proporcionando así una mayor integridad a la firma haciendo que sea válida durante largos periodos de tiempo.

2.6.1.1 Estructura de XMLDSIG

Las firmas digitales XML están representadas por el elemento Signature que tiene la siguiente estructura

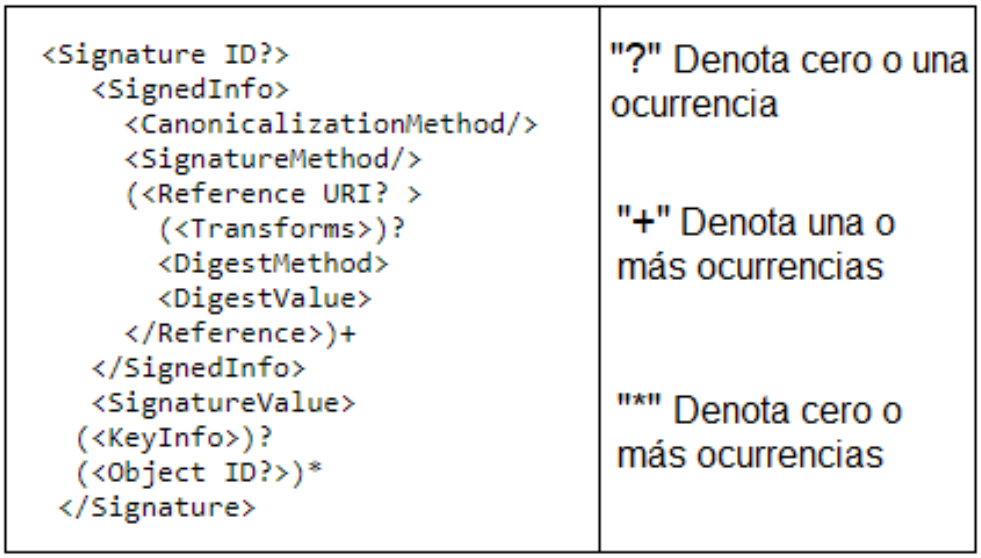


Ilustración 9 Estructura XMLDSIG

Como se muestra en la ilustración anterior la estructura XMLDSIG está compuesta por:

- **Signature:** Este es el elemento padre que contiene información de lo que se está firmando, la propia firma, y la claves para comprobar la firmar.
- **SignedInfo:** Este elemento contendrá la información sobre la firma y los resúmenes de los elementos con información importante.
- **SignatureMethod:** Este elemento especifica qué tipo de algoritmo de firma se utilizará.
- **CanonicalizationMethod:** Especifica el método de canonicalización para la estandarización del documento.
- **References:** Contiene la URI que identifica los elementos que se le deben calcular el resumen.
- **Transform:** Este contendrá una lista en la que cada uno de sus elementos indica un paso realizado en el proceso de cálculo del resumen.
- **DigestMethod:** Especifica la función de resumen utilizada.

- **DigestValue:** Este elemento contendrá el valor de resumen codificado en Base64.
- **Keyinfo:** Este elemento identifica al firmante. En este elemento se suele incluir token o certificados que sirven para validar la firma del documento.
- **Object:** Este elemento se utiliza para contener cualquier tipo de dato por lo general importante para la firma, como sellos de tiempo y, en el caso de una firma de tipo enveloping, para contener los datos que se firman.

2.6.2 XADES (XML Advanced Electronic Signatures)

El estándar XAdES se encuentra defino en el ETSI TS 101 903 V1.4.2 (2010-12)[22] y extiende de XMLDSIG [21] lo que permite su uso con firmas electrónicas avanzadas y calificadas como se especifica en la Directiva europea [1999/93 **/** EC] [23].

Un beneficio importante de XAdES es que permite agregar información y sellos de tiempo que extienden la validez de una firma más allá del vencimiento o revocación de los certificados electrónicos involucrados en la firma o la obsolescencia de las claves y algoritmos criptográficos subyacentes. XAdES contiene varios módulos que permiten varios niveles de seguridad para garantizar aspectos como autenticidad y no repudio.

La firma digital XAdES proporciona autenticación básica y protección de integridad. Sin embargo, sin la adición de un sello de tiempo, la firma electrónica no protege contra la amenaza de que el firmante niega posteriormente haber creado la firma electrónica, es decir no garantiza el no repudio. La marca de tiempo XAdES-T debe crearse en el momento en que se creó la firma para proporcionar protección contra el repudio.

XAdES se compone de los siguientes perfiles donde cada una está basada en la anterior, el firmante está obligado a utilizar una de los siguientes formatos XAdES-BES, XAdES-EPES, XAdES-T, XAdES-C como indica ETSI TS 101 903 V1.4.2 (2010-12) [22] y los formatos extendidos XAdES-X , XAdES-X-L, XAdES-A no es obligatorio su implementación de acurdo a la recomendación.

XAdES-BES:

XAdES-BES se construirá sobre un XMLDSIG incorporando propiedades que se incluirán, algunas, entre los datos firmados (dentro del elemento SignedProperties y otras que no deben ser firmadas (dentro del elemento UnsignedProperties [22]. Como este está basado en XMLDSIG se hace obligatorio incluir y proteger el certificado del firmante dentro de la firma utilizando una de las dos formas siguientes:

- ya sea incorporando la propiedad firmada SigningCertificate
- Incorporando el certificado dentro de KeyInfo y firmando después el elemento

Ilustración 10 estructura XAdES-BES; tomado de [22]

- **XAdES-EPES:**

  Este contienen el formato XMLDSIG + XAdES-BES + la inclusion del elemento SignaturePolicyIdentifier dentro de SignedSignatureProperties La propiedad indica que debe utilizarse una política de firmas para la validación de firmas. puede identificar explícitamente el política de firmas. Es posible que la política exigida requiera otras propiedades.

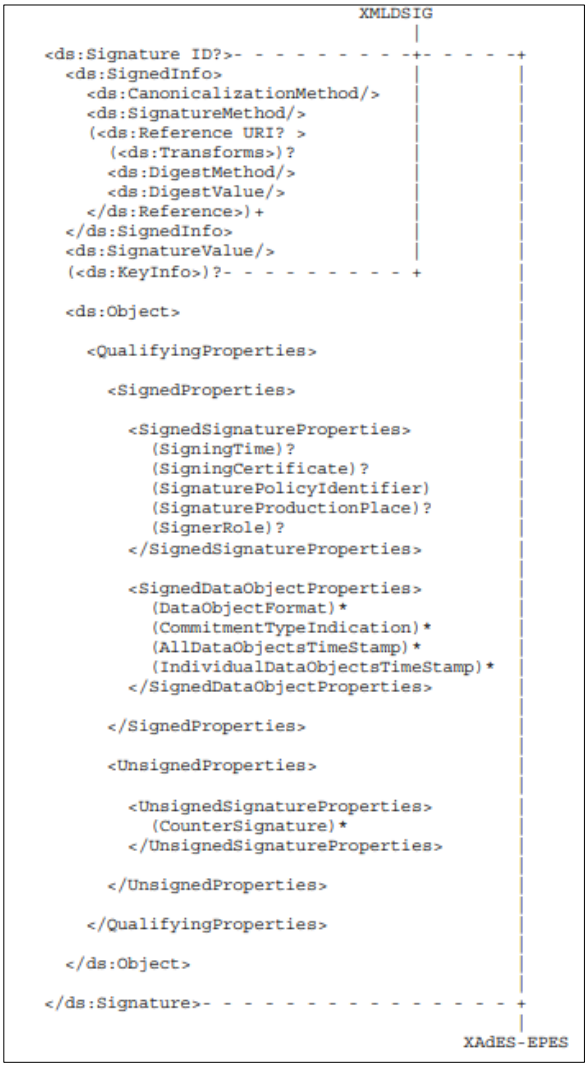


Ilustración 11 Estructura XAdES-EPES; tomado de [22]

- **XAdES-T(timestamp):**

  Este formato agrega una marca de tiempo para garantizar el no repudio y la prueba de anterioridad. Este sobre permite conocer la validez de una firma en caso de que el certificado del firmante sea posteriormente revocado.

  Se debe agregar el elemento SignatureTimeStamp dentro de UnsignedSignatureProperties.

Ilustración 12 XAdES-T; tomado de [22]

- **XAdES-C (complete)**:

  Este añade a XAdES-T referencias al conjunto completo de autoridades de certificación que han sido utilizadas para validar la firma electrónica, así como referencias a los datos de revocación de los certificados. Añade CompleteCertificateRefs, CompleteRevocationsRefs y opcionalmente AttributeCertificateRefs, AtributeRevocationRefs.

Ilustración 13 XAdES-C; tomado de[22]

2.7 SOAP (SIMPLE OBJECT ACCESS PROTOCOL)

SOAP proporciona un mecanismo simple y liviano para intercambiar información estructurada entre diferentes entornos utilizando XML. SOAP no define ninguna semántica de aplicación, como un modelo de programación o semántica específica de implementación, más bien define un mecanismo simple para expresar la semántica de la aplicación al proporcionar un modelo de empaquetado modular y mecanismos de codificación para codificar datos dentro de los módulos.

SOAP consta de tres partes[24]:

- La construcción del sobre SOAP define un marco general para expresar lo que hay en un mensaje quién debe ocuparse de ello y si es opcional u obligatorio.
- Las reglas de codificación SOAP definen un mecanismo de serialización que se puede utilizar para intercambiar instancias de tipos de datos definidos por la aplicación.
- La representación SOAP RPC (Remote Procedure Call) define una convención que se puede utilizar para representar llamadas y respuestas a procedimientos remotos.

2.7.1 Enveloped soap

Un mensaje SOAP es un documento XML que consta de un sobre SOAP obligatorio, un encabezado opcional y un cuerpo obligatorio, un mensaje SOAP contiene lo siguiente[24]:

- El sobre es el elemento superior del documento XML que representa el mensaje.
- El Header es un mecanismo genérico para agregar funciones a un mensaje SOAP de manera descentralizada sin acuerdo previo entre las partes comunicantes. SOAP define algunos atributos que se pueden utilizar para indicar quién debe ocuparse de una característica y si es opcional u obligatoria.
- El Body es un elemento obligatorio que está destinado a contener la información dirigida al destinatario final del mensaje.

Este debe cumplir los siguientes criterios:

1. **Enveloped (Sobre)**

   ✓ Este es el elemento padre del mensaje soap.
   ✓ El elemento debe estar presente en un mensaje SOAP
   ✓ El elemento puede contener declaraciones de espacio de nombres, así como atributos adicionales.

2. **Header (Encabezado)**

- ✓ El elemento puede estar presente en un mensaje SOAP, este debe ser indicado antes del elemento Body.
- ✓ El elemento puede contener un conjunto de entradas de encabezado, cada una de las cuales es un elemento hijo inmediato del elemento de encabezado SOAP.

3. Body (Cuerpo)

- ✓ este elemento debe existir una vez, y debe ir justo después del elemento header si este es indicado.
- ✓ El elemento puede contener un conjunto de entradas de Body, cada una de las cuales es un elemento secundario inmediato del elemento SOAP Body.

A continuación, se muestra un ejemplo básico de un Envelope SOAP, donde este debe contener toda la estructura definida por la aplicación dentro del elemento Body.

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
       <m:GetLastTradePrice xmlns:m="Some-URI">
           <symbol>DIS</symbol>
       </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

NameSpace

Body

Ilustración 14 ejemplo SOAP

2.7.2 WSDL (Web Service Description Language)

La estructura del Enveloped SOAP contendrá la información o el payload dentro del elemento Body pero esta debe seguir una estructurada según la definida en el web service. [22] wsdl se puede considerar como el contrato que deben seguir 2 aplicación para la transmisión de mensajes SOAP. El wsdl Se utiliza para describir la funcionalidad de un servicio web basado en SOAP.

Los archivos WSDL definen aspectos como:

- El número de ocurrencias de elementos o atributos.
- Los elementos y atributos opcionales y obligatorios.
- La estructura del mensaje si es necesario.
- Los métodos permitidos por el web service.

2.7.3 ws-security

WS-SECURITY es un protocolo gestionado por oasis que proporciona seguridad para garantizar la integridad, confidencialidad y autenticidad de mensajes. Los mecanismos de WS-Security se pueden utilizar para dar cabida a una amplia variedad de tecnologías de cifrado y modelos de seguridad.

WS-Security es un estándar a nivel de mensajes que tiene su base en la protección de mensajes SOAP a través de la firma digital XML, la confidencialidad a través del cifrado XML. La especificación de servicios web define los recursos para proteger la integridad y la confidencialidad de un mensaje.

En la siguiente ilustración se muestra la estructura ws-security para autenticación por medio de certificado.

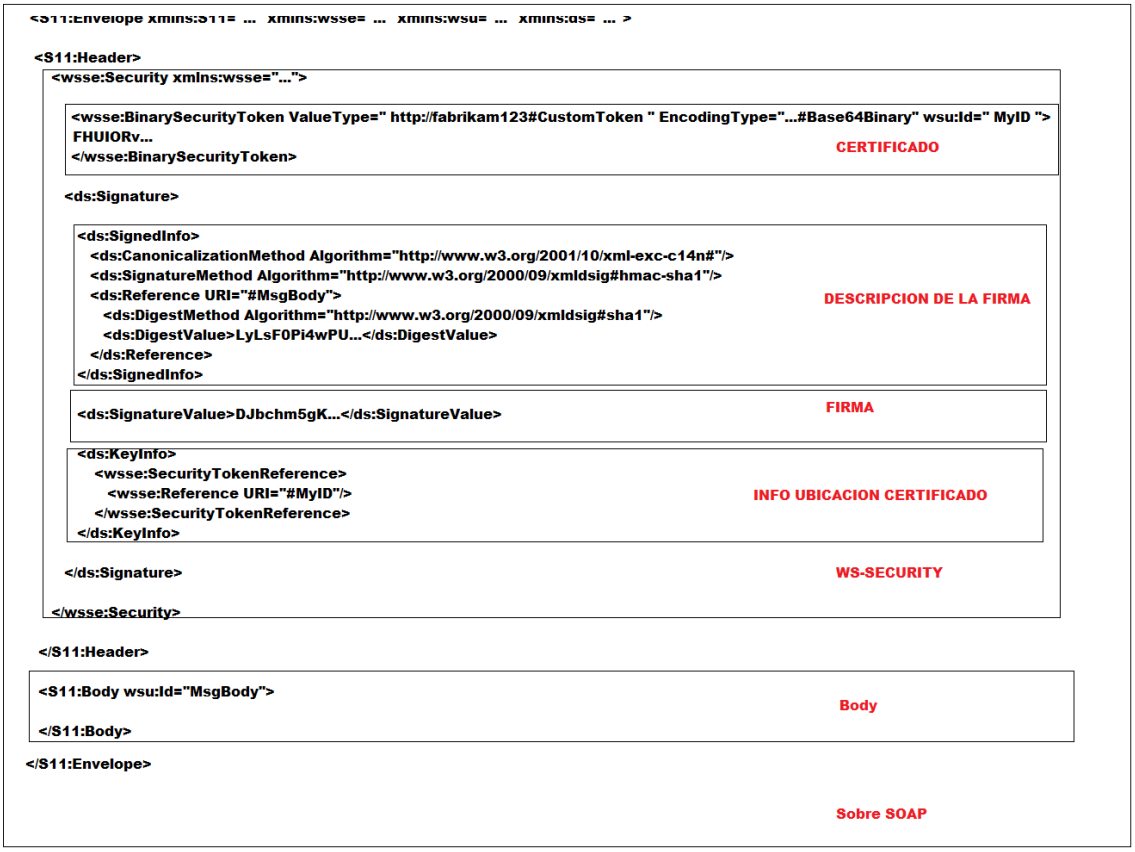


Ilustración 15 sobre soap con ws-security

BinarySecurityToken: Contendrá el certificado con el que se aplicó la firma

SignedInfo: describen lo que se está firmando y el tipo de canonicalización que se está utilizando.

SignatureValue: especifica el valor de firma de la forma canonicalizada de los datos que se están firmando como se define en la especificación de XML signature.

KeyInfo: proporcionan información, parcial o completa, sobre dónde encontrar el certificado de seguridad asociado con esta firma.

2.8 ESTADO DEL ARTE

2.8.1 Trabajos realizados a nivel regional

titulo: Diseño de un software contable con la inclusión de facturación electrónica para pequeñas y medianas empresas.

autor: GALVIS ARÉVALO HERNANDO

institución: Universidad de pamplona.

año: 2018

Descripción:

> Este proyecto se desarrolló en el municipio de Ocaña norte de Santander, plantearon el desarrollo de una aplicación en la nube para firmas de facturas electrónicas centrándose principalmente en el marco legal, contemplando estudios cuantitativos acerca del estado de las medianas y microempresas en Ocaña, Indagan en los aspectos que se deben tener en cuenta para el desarrollo de un sistema que se adapte a las necesidades de las empresas.[25]

2.8.2 Trabajos realizados a nivel nacional

título: Lenguaje de dominio específico para generar facturas electrónicas de acuerdo a los requerimientos técnicos de la DIAN – invoiceQL.

autor: edwar Alonso Rojas Blanco

institución: Universidad nacional de Colombia.

año: 2020

Descripción:

> En este trabajo realizan la implementación de facturación electrónica mediante el lenguaje de dominio específico llamado InvoiceQL el cual permite generar facturas electrónicas mediante sentencias.

Para desarrollar InvoiceQL utilizaron una metodología basada en desarrollo de software con el paradigma MDDF (Desarrollo de funcionalidades dirigido por modelos) mediante el framework EMF con el IDE de desarrollo eclipse, con el cual pueden crear facturas haciendo uso de instrucciones, este genera igualmente código fuente en lenguaje de programación Python el cual también genera facturas electrónicas. En este proyecto realizan un sistema de facturación mediante DSL con el objetivo de aislar la lógica empresarias y la lógica para la facturación, con el objetivo de minimizar los conflictos entre ambos sistemas. Se presenta una idea que puede facilitar y evitar problemas al unificar la generación de facturas a sistemas de facturación utilizados por las empresas.[26]

2.8.3 Trabajos realizados a nivel internacional

título: análisis, diseño e implementación de facturación electrónica para la optimización de los procesos tributarios en la empresa corporación agrolatina.a.c.

autor: chipana luna katherine xiomara, camacho medina janet brigyth

institución: Universidad Nacional San Luis Gonzaga de Ica

país: perú

año: 2018

Descripción:

Como debe de esperarse las normativas para cada país deben cambiar debido a que en Colombia está regida por la DIAN (dirección de impuesto y aduanas nacionales) y en Perú es SUNAT (superintendencia nacional de administración tributaria) por lo que los procesos pueden diferir entre países. en este proyecto se dedican a desarrollar un sistema de facturación electrónica, en este se da una visión más general del funcionamiento completo de un sistema de facturación convencional. por medio de diagrama de flujos de cómo cada parte de se comunica con las demás [27].

CAPÍTULO III: METODOLOGÍA

En este capítulo se abordará el proceso que se llevó acabo para la emisión de facturas electrónicas de venta; En el siguiente diagrama se muestran los procesos que se deben seguir para generar y distribuir facturas electrónicas.

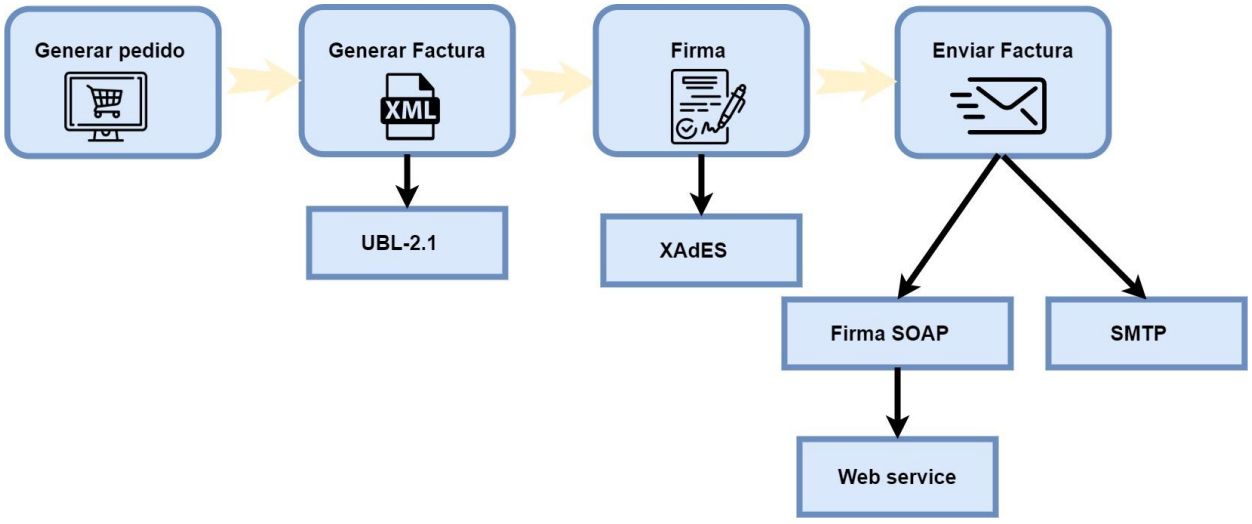


Ilustración 16 procesos necesarios para generar facturas

3.1 ASPECTOS TÉCNICOS SEGÚN NORMATIVA

En la normativa correspondiente a la facturación electrónica “Anexo técnico o resolución 000042” se especifican las tecnologías requeridas para implementar la facturación electrónica en Colombia.

3.1.1 Formato para factura electrónica

Los formatos utilizados por la DIAN para la facturación electrónica son formatos estándares para el comercio electrónico UBL (Universal Business Language). Estos formatos son gestionados por la organización oasis, en la que se encuentran un total de 65 formatos en su versión 2.1, entre ellos 5 formatos seleccionados por la DIAN para implementar la facturación electrónica en Colombia, dichos formatos se listan a continuación:

1. **Attached Document:** Un contenedor UBL que permite empaquetar un documento de cualquier tipo con el documento UBL que lo hace referencia.
2. **Credit Note:** Un documento utilizado para especificar los créditos adeudados al Deudor por parte del Acreedor.
3. **Debit Note**: Un documento utilizado para especificar las deudas contraídas por el Deudor.
4. **Invoice**: Un documento utilizado para solicitar el pago.
5. **Application Response:** Un documento para indicar la respuesta de la aplicación a una transacción.

3.1.2 Políticas de firma

El objetivo de la Política de firma define las principales características técnicas para la firma digital, la cual garantizara la seguridad, autenticidad y confiabilidad de todas las operaciones comerciales en la que se involucra la facturación electrónica.

3.1.2.1 Certificado digital

Dentro de las políticas de firmas requeridas por la DIAN es necesario la utilización de un certificado digital expedido por una entidad avalada por la ONAC (Organismo Nacional de Acreditación de Colombia), esto se debe a que debe existir una entidad que pueda garantizar la identidad de la persona o entidad a la que pertenece el certificado digital, cualquier documento electrónico firmado que no cumpla con esta condición, será invalido y no tendrá los efectos fiscales establecidos en el artículo 616-1 del estatuto tributario[1].

3.1.2.2 Formato para firma digital

Se debe utilizar el estándar XMLDSig enveloped con formato XAdES-EPES según la especificación técnica ETSI TS 101 903S.

3.1.2.3 Algoritmos de Firma

El algoritmo para el cálculo de la firma digital de la factura electrónica puede ser cualquiera de los definidos en la especificación XMLDSIG [21] que actualmente son:

- ✓ Cifrado RSA con resumen SHA256
- ✓ Cifrado RSA con resumen SHA384
- ✓ Cifrado RSA con resumen SHA512

3.1.2.4 Algoritmo de canonicalización XML

El algoritmo requerido para normalizar los documentos XML para calcular el resumen o la firma debe ser canonical exclusivo como se recomienda en XMLDSIG.

3.1.3 Mecanismos de comunicación

La comunicación con el web service de la DIAN se ejecuta en la web por lo tanto este usa el protocolo HTTP el cual es el estándar para la web, también existe su versión segura HTTPS la cual usa un mecanismo de cifrado para proporcionar confidencialidad a los datos que viajan entre diferentes sistemas. La DIAN cuenta con el protocolo TLS versión 1.2 el cual provee seguridad a todos los datos que viajan desde los distintos sistemas de facturación de los contribuyentes hacia el web service de la DIAN.

3.1.3.1 Protocolo de comunicación

El web service de la DIAN está basado en el protocolo SOAP por lo que él envió de las facturas debe realizarse con el formato XML el cual es el formato utilizado para compartir mensajes entre servicios web basados en el protocolo SOAP.

3.1.3.1.1 Método de identificación

La DIAN requiere identificar la persona o entidad que envía el mensaje SOAP por lo que requiere un modelo compatible con el protocolo SOAP para tal fin. El modelo de comunicación requerido sigue el estándar de servicios web definido por el WS-Security 1.0 Oasis, con autenticación X.509 Certificate Token Profile 1.1

3.2 GENERAR PEDIDO

Para generar facturas electrónicamente se debe contar con un aplicativo que permita interactuar con el sistema, las empresas que contaban con un sistema de facturación convencional se vieron obligados a crear o modificar sus sistemas para agregar la funcionalidad de factura electrónica. Este proyecto se centró principalmente en el proceso de generar, firmar y enviar la factura, estos suelen ser los procesos esenciales debido a que la interfaz del sistema puede ser ajustada a las necesidades de las empresas o negocios.

3.2.1 Arquitectura del sistema

La arquitectura implementada para el desarrollo de este proyecto es la que se presenta en la siguiente ilustración.

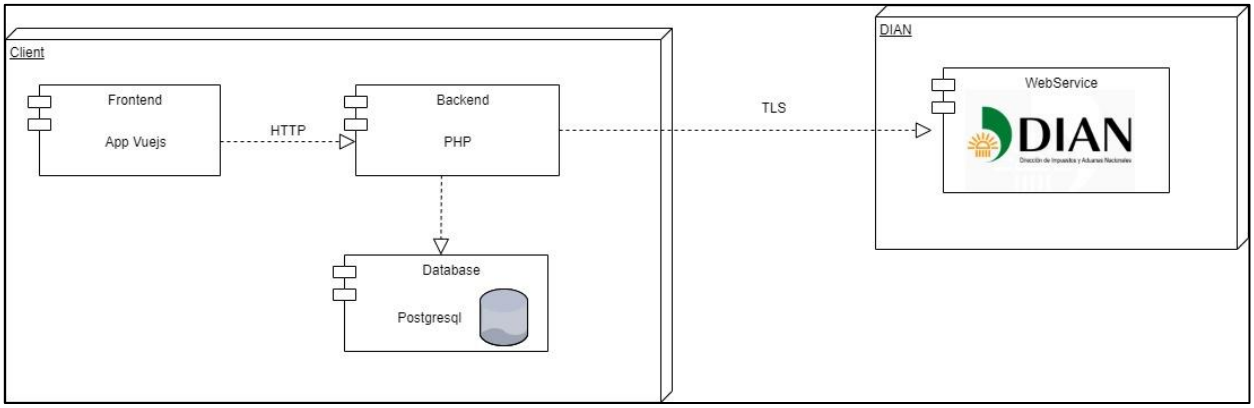


Ilustración 17 Arquitectura del sistema

Este sistema será planteado en bloques por separados, con el objetivo de tener un menor acoplamiento entre la lógica del sistema y la interfaz de usuario. Esto permite diseñar el sistema en distintos lenguajes de programación.

3.2.2 Diseño de la base de datos

Para el diseño de la base de datos se tomaron aspectos como la gestión de usuarios, gestión de clientes y gestión de productos, que son los principales involucrados en una operación de comercio.

Antes de desarrollar la base de datos se debe crear el modelo entidad relación el cual indica cómo se relacionarán todas las entidades entre sí, se decidió permitir que múltiples usuarios puedan ingresar al sistema y facturar de manera independiente almacenando la información requerida para en envió de las facturas. Como se muestra en la siguiente ilustración.

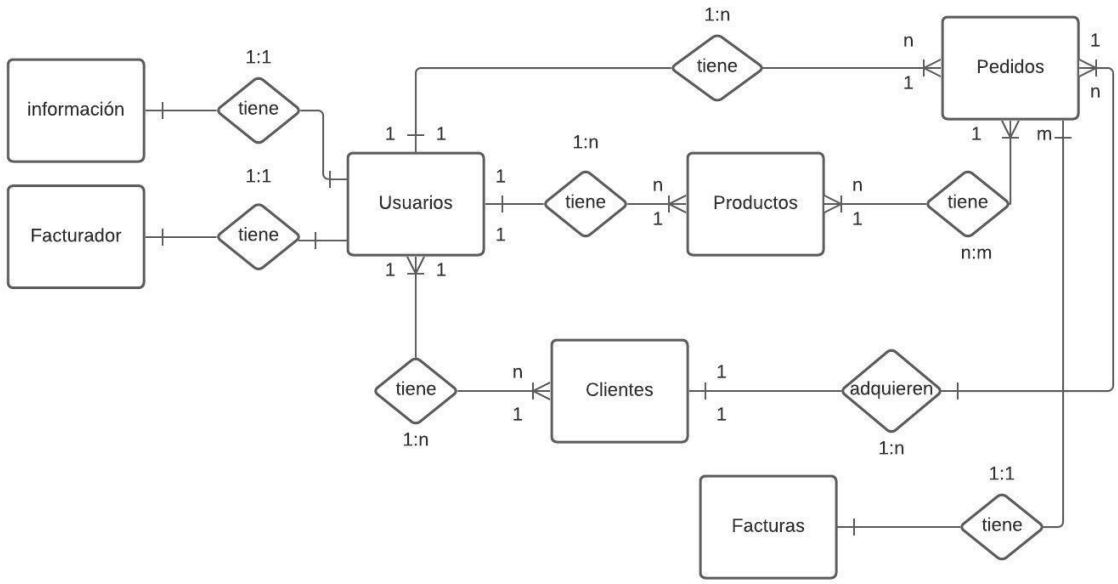


Ilustración 18 modelo entidad relación

Una vez tenemos definido en modelo entidad relación podemos continuar con el diagrama entidad relación el cual presenta los datos involucrados por cada una de las entidades, como se muestra a continuación.

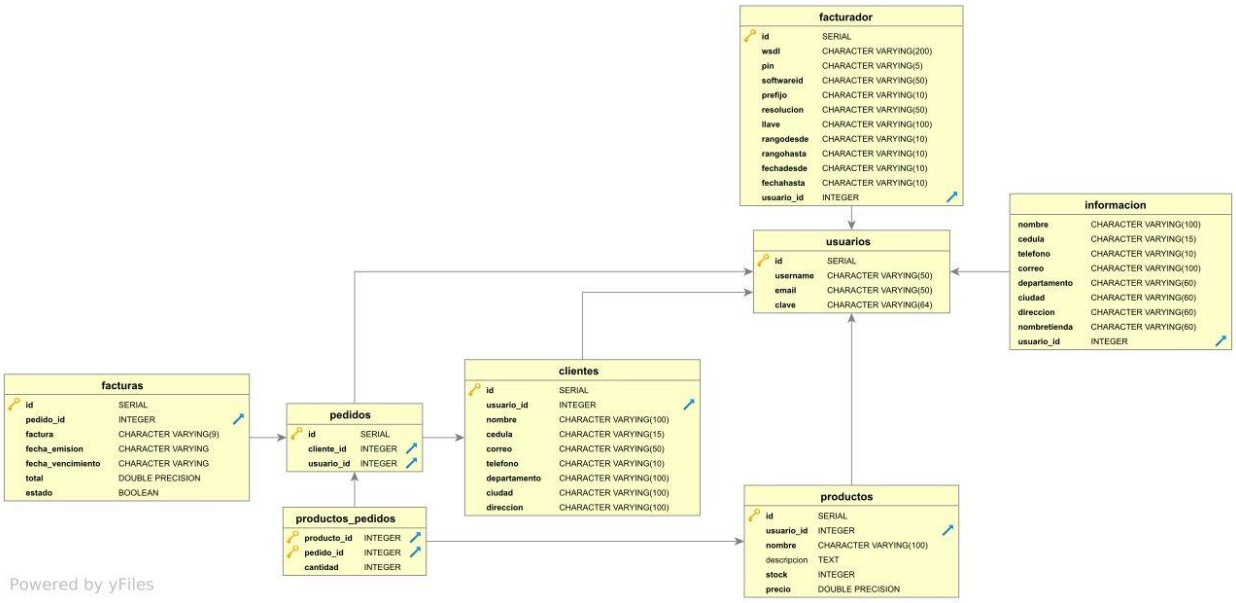


Ilustración 19 Diagrama entidad relación

En las siguientes tablas se presentan las características principales y descripción de las entidades empleadas.

Nombre	Tipo dato	NULO	Descripción
Id	SERIAL	NO	Identificador único
Email	VARCHAR	NO	Correo electrónico, requerido para inicio de sesión
Username	VARCHAR	NO	Nombre de usuario, solo para representación de perfil
Clave	VARCHAR	NO	Password para inicio de sesión

Tabla 2 usuarios

Nombre	Tipo dato	NULO	Descripción
Id	SERIAL	NO	Identificador único
Wsdl	VARCHAR	NO	WSDL del servicio web de la dian
Pin	VARCHAR	NO	Pin del software
SoftwareID	VARCHAR	NO	Identificador del software
Prefijo	VARCHAR	NO	Prefijo de facturación
Resolucion	VARCHAR	NO	Resolución de facturación
Llave	VARCHAR	NO	Llave técnica
RangoDesde	VARCHAR	NO	Inicio de rango de numeración
RangoHasta	VARCHAR	NO	fin de rango de numeración
FechaDesde	VARCHAR	NO	Rango fecha inicio
FechaHasta	VARCHAR	NO	Rango fecha fin
Usuario_id	VARCHAR	NO	Id del usuario facturador

Tabla 3 información

Nombre	Tipo dato	NULO	Descripción
Id	SERIAL	NO	Identificador único

Nombre	VARCHAR	NO	Nombre registrado en el RUT del facturador
Cedula	VARCHAR	NO	Cedula registrada en el RUT del facturador
Teléfono	VARCHAR	NO	Teléfono de contacto
Correo	VARCHAR	NO	Correo de contacto
Departamento	VARCHAR	NO	Departamento de residencia
Ciudad	VARCHAR	NO	Ciudad de residencia
Dirección	VARCHAR	NO	Dirección de residencia
NombreTienda	VARCHAR	NO	Nombre de empresa
Usuario_id	INTEGER	NO	Identificador usuario

Tabla 4 facturador

Nombre	**Tipo dato**	**NULO**	**Descripción**
Id	SERIAL	NO	Identificador único
usuario_id	INTEGER	NO	Identifica a que al cliente y al vendedor
Nombre	VARCHAR	NO	Nombres del cliente
Cedula	VARCHAR	NO	Cedula de ciudadanía, será requerida para generar factura
Correo	VARCHAR	NO	Correo electrónico de usuario

Teléfono	VARCHAR	NO	Teléfono de cliente
Departamento	VARCHAR	NO	Departamento de residencia
Ciudad	VARCHAR	NO	Ciudad de residencia
Dirección	VARCHAR	NO	Dirección de residencia

Tabla 5 clientes

Nombre	**Tipo dato**	**NULO**	**Descripción**
Id	SERIAL	NO	Identificador único
usuario_id	INTEGER	NO	Relación productos usuarios
Nombre	VARCHAR	NO	Nombre del producto
Descripción	TEXT	SI	Descripción del producto
Stock	INTEGER	NO	Cantidad disponible del producto
Precio	FLOAT	NO	Precio del producto

Tabla 6 Productos

Nombre	**Tipo dato**	**NULO**	**Descripción**
Id	SERIAL	NO	Identificador único
cliente_id	INTEGER	NO	Identificador de cliente
usuario_id	VARCHAR	NO	Identificador de usuario

Tabla 7 Pedidos

Nombre	Tipo dato	NULO	Descripción
Id	SERIAL	NO	Identificador único
Producto_id	INTEGER	NO	Identificador de producto
Pedido_id	INTEGER	NO	Identificador de pedido
Cantidad	INTEGER	NO	Cantidad de productos

Tabla 8 productos_pedidos

3.2.2.1 Sentencias SQL

A continuación, se muestras las sentencias SQL, implementada con el gestor de bases de datos PostgreSQL.

```
CREATE TABLE usuarios (
  id SERIAL PRIMARY KEY,
  UserName VARCHAR(50) NOT NULL,
  Email VARCHAR(50) UNIQUE NOT NULL,
  Clave VARCHAR(64) NOT NULL
);
CREATE TABLE clientes (
  id SERIAL PRIMARY KEY,
  usuario_id int not null,
  Nombre VARCHAR(100) NOT NULL,
  Cedula VARCHAR(15) UNIQUE NOT NULL,
  Correo VARCHAR(50) UNIQUE NOT NULL,
  Telefono VARCHAR(10) UNIQUE NOT NULL,
  Departamento VARCHAR(100) NOT NULL,
  Ciudad VARCHAR(100) NOT NULL,
  Direccion VARCHAR(100) NOT NULL,
  CONSTRAINT FK1 FOREIGN KEY(usuario_id) references usuarios(id)
);

CREATE TABLE productos (
  id SERIAL PRIMARY KEY,
  usuario_id INTEGER NOT NULL,
  Nombre VARCHAR(100) NOT NULL,
```

```
  Descripcion TEXT NULL,
  Stock INTEGER NOT NULL,
  Precio FLOAT NOT NULL,
  CONSTRAINT FK1 FOREIGN KEY(usuario_id) REFERENCES usuarios(id)
);


CREATE TABLE pedidos (
  id SERIAL PRIMARY KEY,
  cliente_id INTEGER  NOT NULL,
  usuario_id INTEGER  NOT NULL,
 CONSTRAINT FK2 FOREIGN KEY(cliente_id) REFERENCES clientes(id),
 CONSTRAINT FK3 FOREIGN KEY(usuario_id) REFERENCES usuarios(id)
);
CREATE TABLE productos_pedidos (
  Producto_id INTEGER  NOT NULL,
  Pedido_id INTEGER NOT NULL,
  cantidad INTEGER NOT NULL,
  PRIMARY KEY(Producto_id, Pedido_id),
  CONSTRAINT FK1 FOREIGN KEY(Producto_id) REFERENCES productos(id),
  CONSTRAINT FK2 FOREIGN KEY(Pedido_id) REFERENCES pedidos(id)
);


CREATE TABLE facturador(
    id SERIAL PRIMARY KEY,
    Wsdl varchar(200) NOT NULL,
    Pin varchar(5) NOT NULL,
    SoftwareID varchar(50) NOT NULL,
    Prefijo varchar(10) NOT NULL,
    Resolucion varchar(50) NOT NULL,
    llave varchar(100) NOT NULL,
    RangoDesde varchar(10) NOT NULL,
    RangoHasta varchar(10) NOT NULL,
    FechaDesde varchar(10) NOT NULL,
    FechaHasta varchar(10) NOT NULL,
    Usuario_id int  NOT NULL,
    CONSTRAINT FK1 FOREIGN KEY(Usuario_id) REFERENCES usuarios(id)
);
```

```
CREATE TABLE informacion(
   Nombre varchar(100) NOT NULL,
   Cedula varchar(15) NOT NULL,
   Telefono varchar(10) NOT NULL,
   Correo varchar(100)  NOT NULL,
   Departamento varchar(60) NOT NULL,
   Ciudad varchar(60) NOT NULL,
   Direccion varchar(60) NOT NULL,
   NombreTienda varchar(60) NOT NULL,
   Usuario_id int NOT NULL,
   CONSTRAINT FK1 FOREIGN KEY(Usuario_id) REFERENCES usuarios(id)
);


CREATE TABLE facturas (
  id SERIAL PRIMARY KEY,
  Pedido_id int NOT NULL,
  Factura VARCHAR(9) NOT NULL,
  Fecha_emision varchar NOT NULL,
  Fecha_vencimiento varchar NOT NULL,
  Total FLOAT NOT NULL,
  TrackId varchar NULL,
  Estado character  DEFAULT '0' NOT NULL,
  CONSTRAINT FK1 FOREIGN KEY(Pedido_id) REFERENCES pedidos(id)
);
```

3.2.2.2 Conexión php-postgresql

Para realizar la conexión al gestor de bases de datos postgres desde el lenguaje de programación php se debe utilizarse la librería pg, esta librería cuenta con el método pg_connect que nos permitirá conectarnos; esta función recibe como parámetros el usuario, contraseña, puerto, la ubicación del servicio y el nombre de base de datos.

```
<?php
class Database{
   protected $user="postgres";
   protected $password="";
   protected $port ="5432";
   protected $host="127.0.0.1";
   protected $database="Facturas";
```

```
    public function __construct(){}

    public function Connect(){
        try {
            $conection = pg_connect("host=".$this->host." port=".$this->port." dbname=".$this-
>database." user=".$this->user." password=".$this->password);
        } catch (Exception  $E) {
            //throw $th;
            echo "error de conexion: ".$E;
        }
        return $conection;
    }
}
?>
```

3.2.3 Diseño del aplicativo Web

El diseño de la interfaz de usuario se empleó con el Framework Vuejs, principalmente por su facilidad de implementar aplicaciones tipo spa(Single-Page Applications).

VueJS es un framework progresivo que permite construir interfaces de usuario de una sola página o spa La librería centra su funcionalidad en la capa de visualización, y es fácil de utilizar e integrar con otras librerías o proyectos existentes.

para llevar la construcción de aplicaciones spa se deben tener conocimientos en el lenguaje de marcado HTML, hoja de estilos en cascada (CSS) y principalmente conocimientos en el lenguaje de programación JavaScript, resulta no ser conveniente abarcar estos temas para no dejar de lado los objetivos principales de este proyecto.

A continuación, se presenta el wireframe o boceto de la aplicación que se desarrollara para la interfaz de usuario.

La vista de registro usuarios permitirá a los usuarios registrarse en sistema.

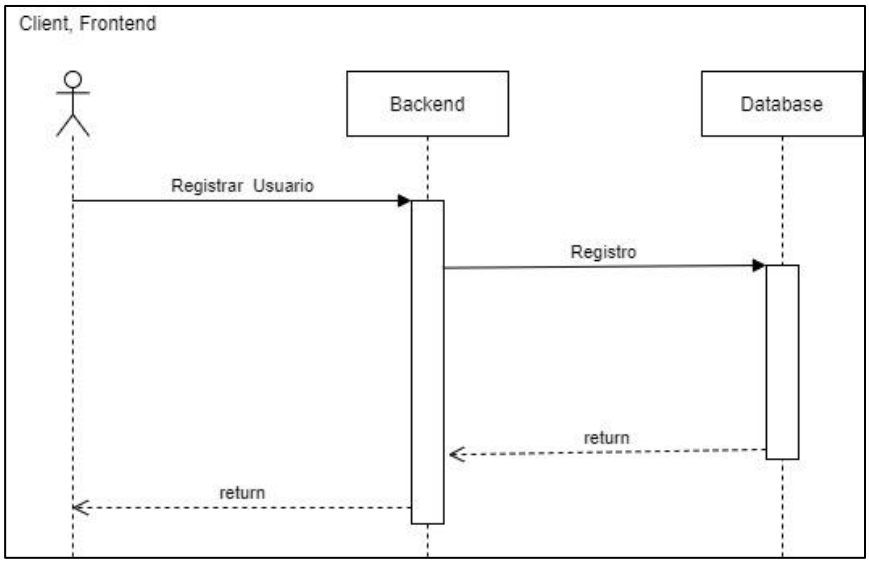


Ilustración 20 secuencia UML registro de usuario

Ilustración 21 wireframe registro

La vista login permite a los usuarios registrados ingresar al dashboard el aplicativo, esta valida la existencia del registro del usuario en la base de datos.

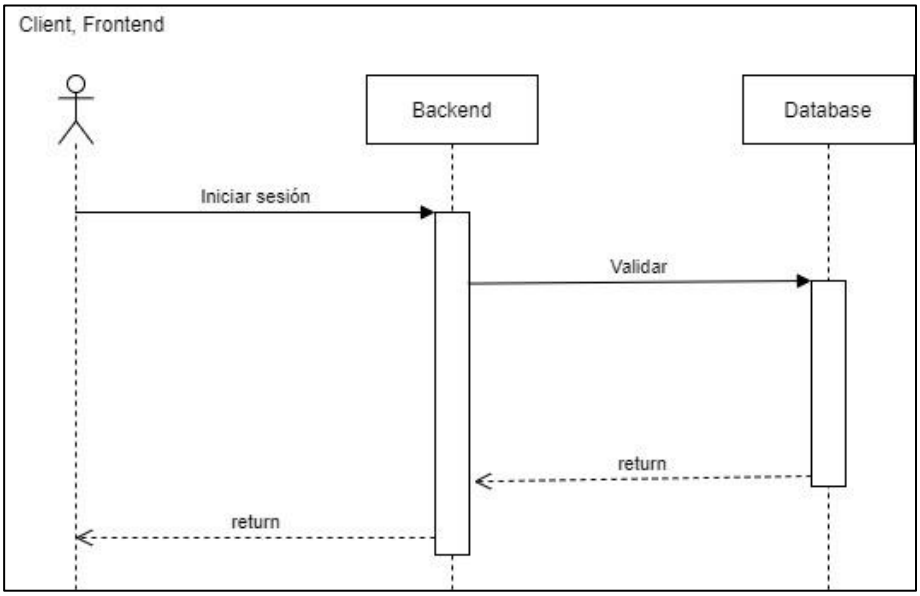


Ilustración 22 Secuencia UML login

Ilustración 23 Wireframe login

La vista dashboard permite visualizar información como la cantidad de productos, clientes y facturas registradas.

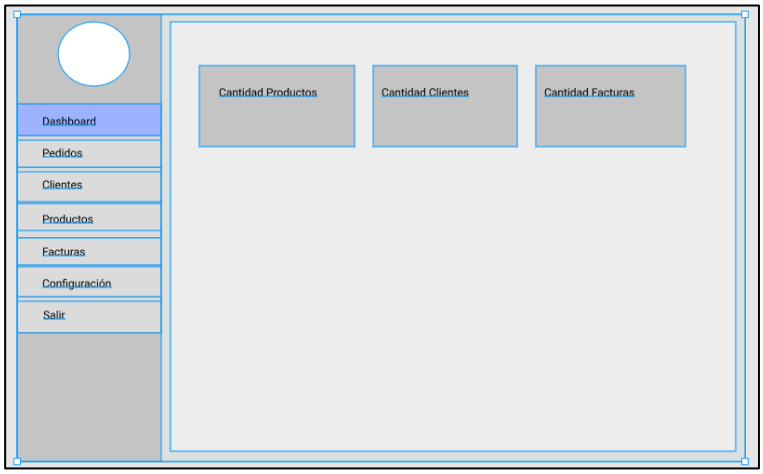


Ilustración 24 wireframe Dasboard

La vista pedidos permite generar facturas tanto con el método síncrono como asíncrono y básicamente todos los procesos esenciales de los sistemas de facturación como la generación y proceso de distribución.

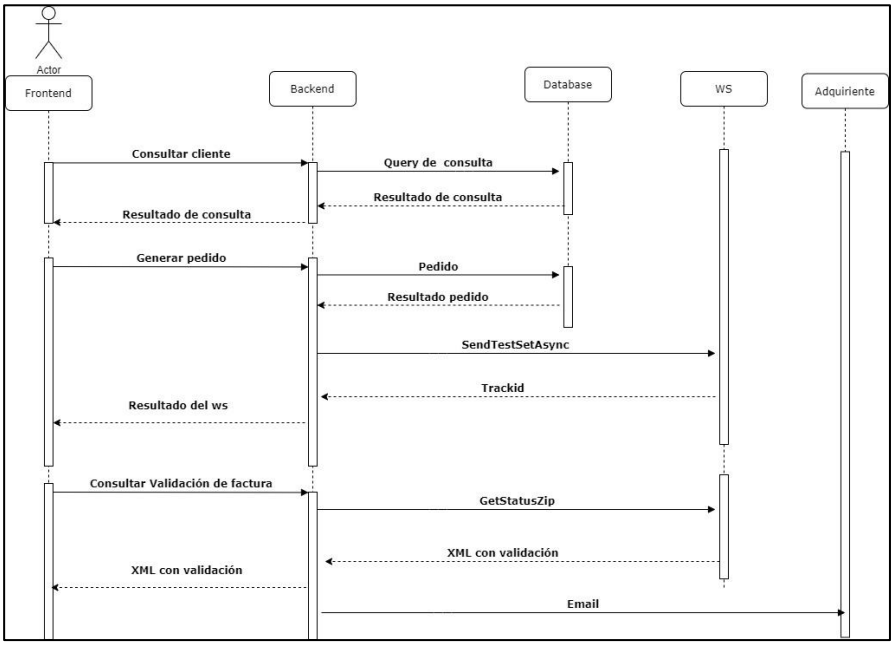


Ilustración 25 Secuencia UML envió de factura método asíncrono

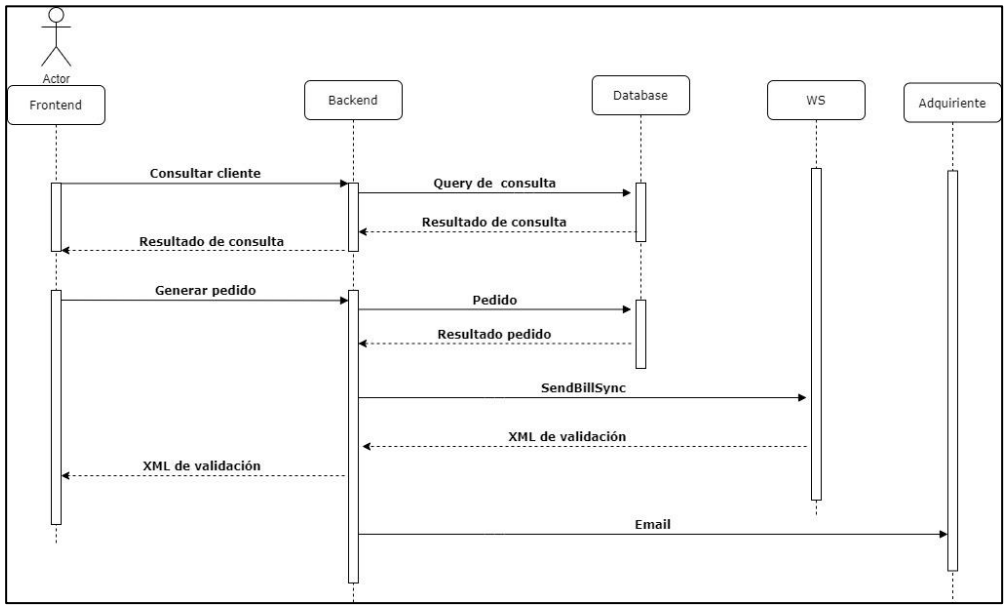


Ilustración 26 Secuencia UML envió factura método síncrono

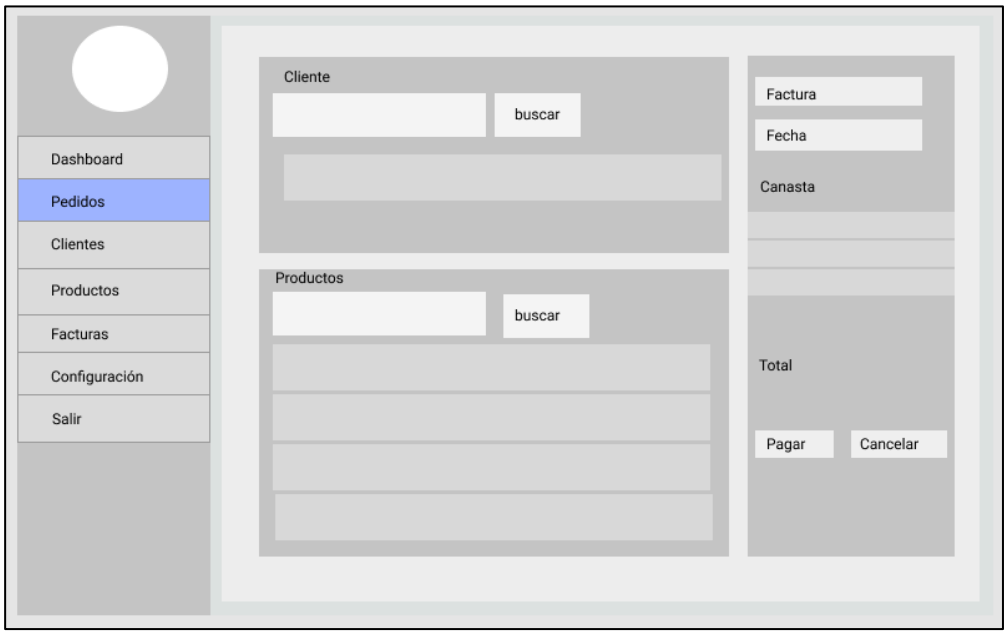


Ilustración 27 Wireframe pedidos

La vista cliente y productos cumplen la función de consultar y registrar los productos en la base de datos.

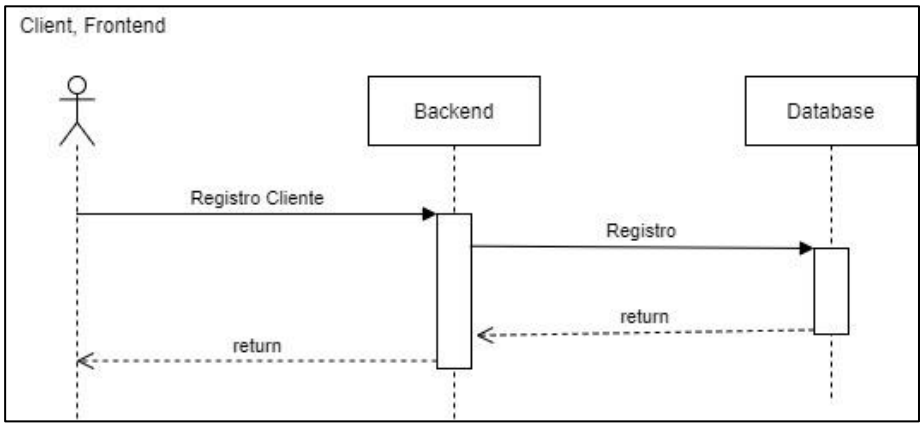


Ilustración 28 Secuencia UML registro cliente

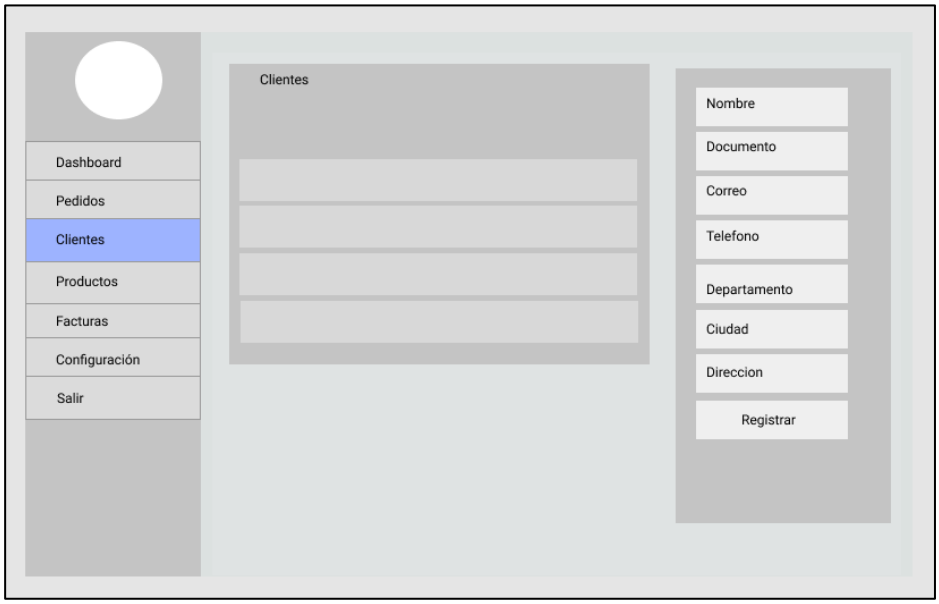


Ilustración 29 wireframe Clientes

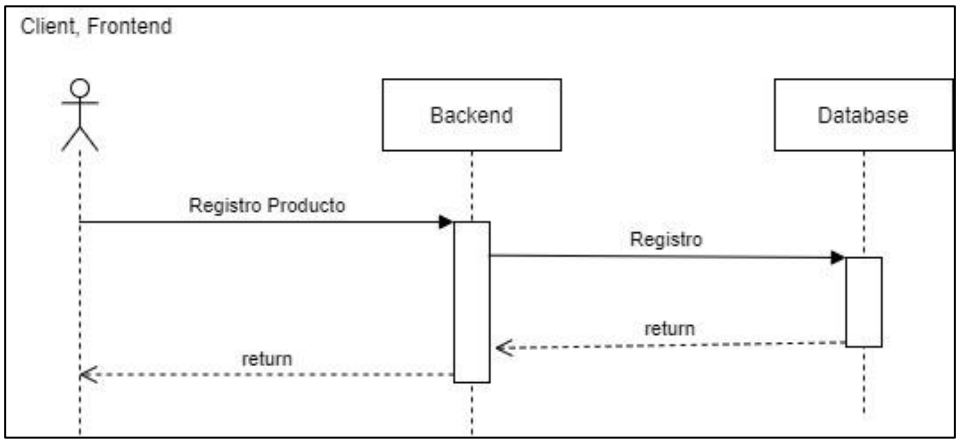


Ilustración 30 Secuencia UML productos

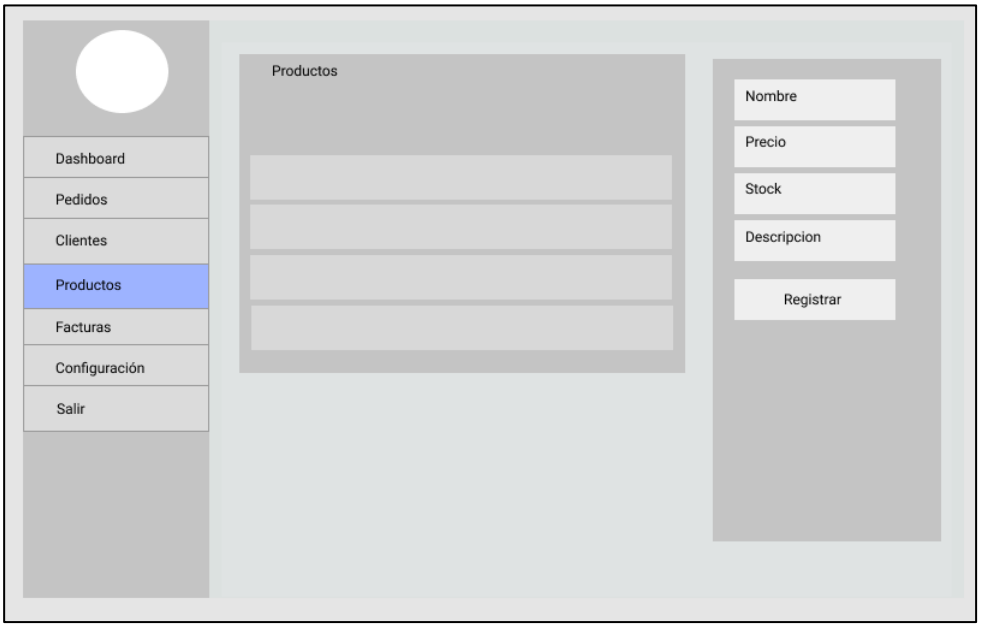


Ilustración 31 wireframe productos

La vista facturas permite vizualizar las facturas generadas y su estado de validacion.

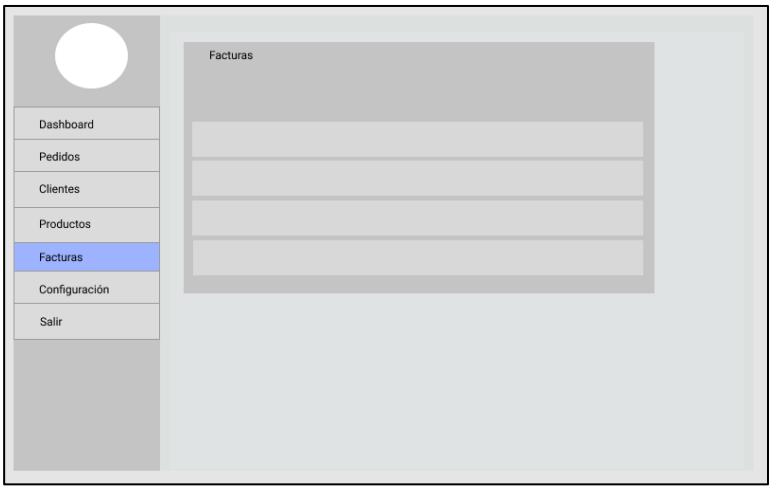


Ilustración 32 wireframe facturas

La vista configuración permitirá registrar toda la información necesaria para generar y distribuir la factura electrónica al web service de la DIAN.

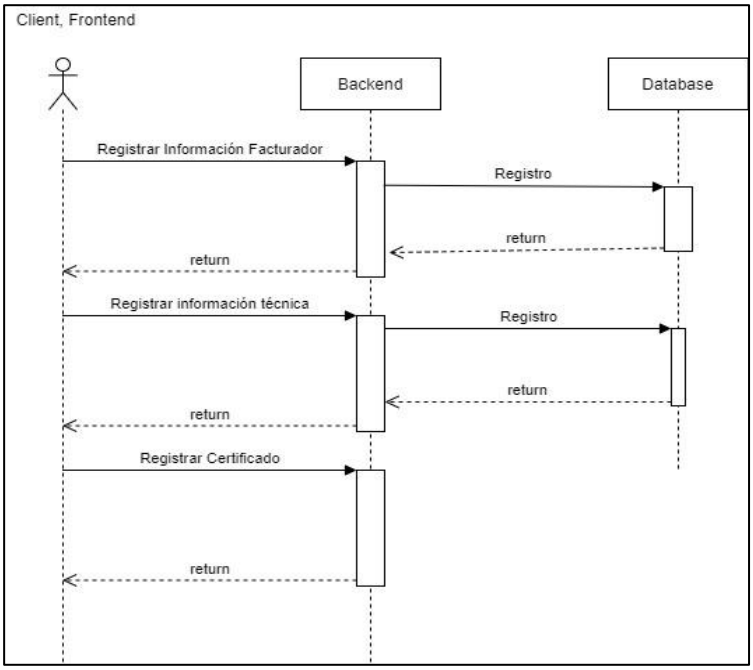


Ilustración 33 Secuencia UML configuración

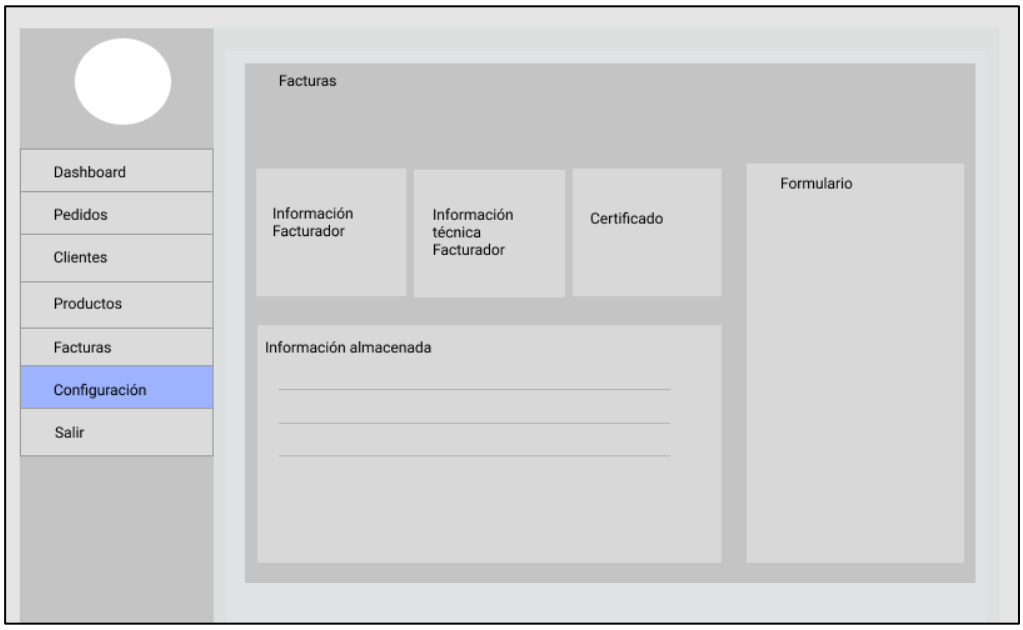


Ilustración 34 wireframe configuración

3.2.4 Despliegue del sistema

Un hosting es empleado para desplegar aplicación web con el objetivo de que puedan ser accedidas desde internet.

3.2.4.1 AWS (Amazon Web Services)

AWS es un conjunto de servicios de computación en la nube que permite mantener una infraestructura sin necesidad de adquirir equipos de hardware.

3.2.4.1.1 EC2 (Amazon Elastic Compute Cloud)

Entre los servicios de AWS se encuentra EC2, este proporciona una capacidad informática escalable en la nube, el uso de EC2 elimina la necesidad de invertir en hardware, lo que facilita desarrollar e implementar aplicaciones más rápidamente. EC2 permite iniciar tantos servidores virtuales como sea necesario y permite seleccionar entre diferentes sistemas operativo y capacidades de computo.

3.2.4.1.1.1 proceso de despliegue del sistema en EC2

Para desplegar un proyecto en una instancia EC2 se debe contar con una cuenta en aws es necesario contar con una tarjeta de crédito para el registro, este proceso requiero 1

dólar para la activación de la cuenta. Una vez registrado se procede a buscar el servicio EC2.

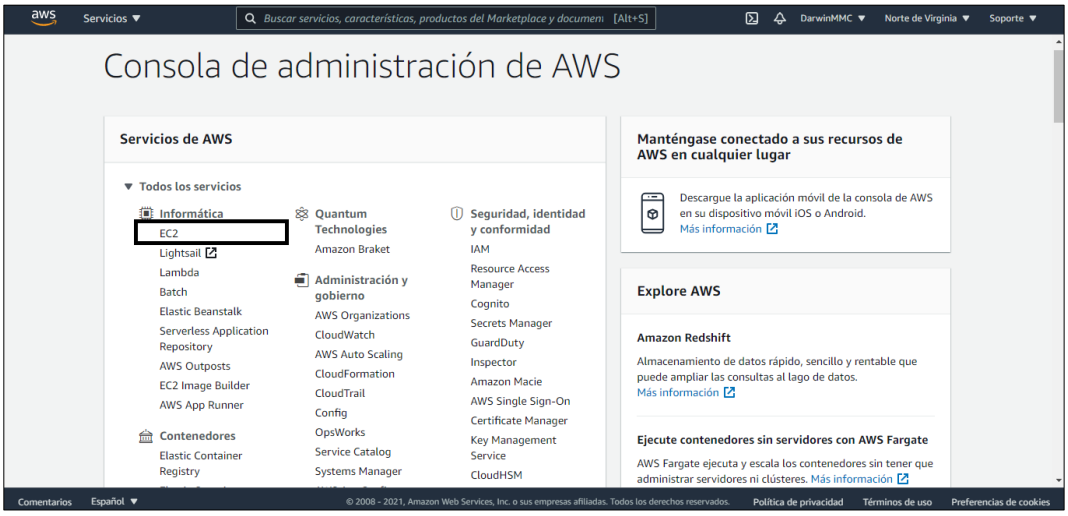


Ilustración 35 AWS EC2

Se debe ingresar a EC2 esta opción dirigirá al panel de administración EC2 y dar clic en lanzar instancia.

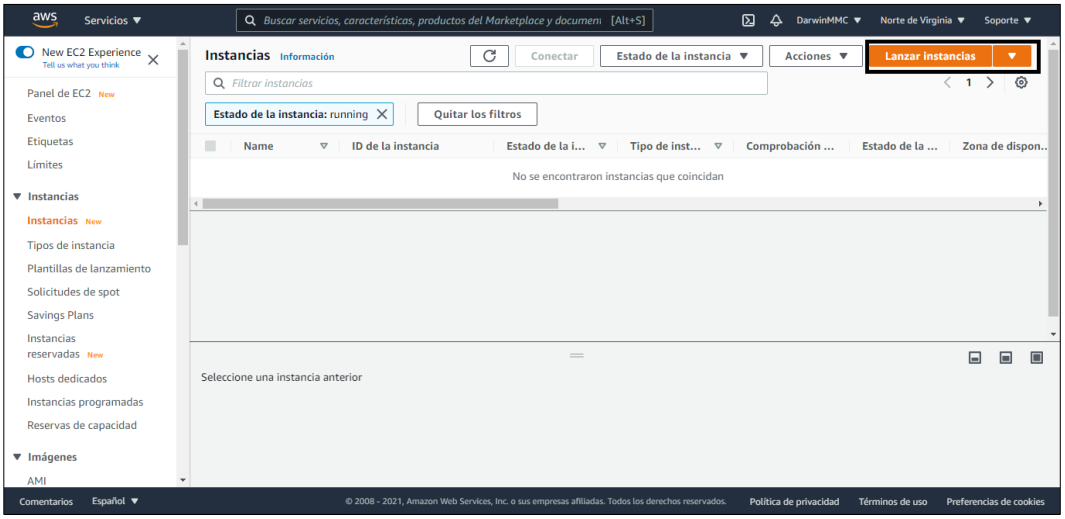


Ilustración 36 aws EC2

Como primer paso para lanzar una instancia se debe seleccionar una AMI (Amazon Machine Images), para este caso se utilizará Ubuntu server 20.04.LTS.

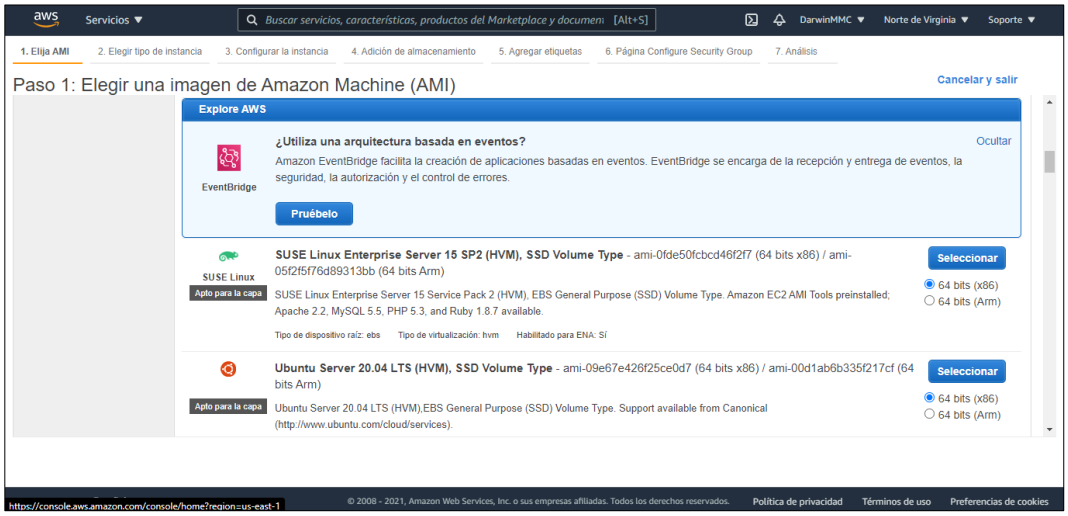


Ilustración 37 aws AMI

Seguido se debe seleccionar el tipo de instancia, en aws se cuenta múltiples familias que varían entre capacidad de cómputo y almacenamiento, aws proporciona una capa gratuita por tiempo limitado, esta capa permite utilizar instancias de tipo micro de manera gratuita.

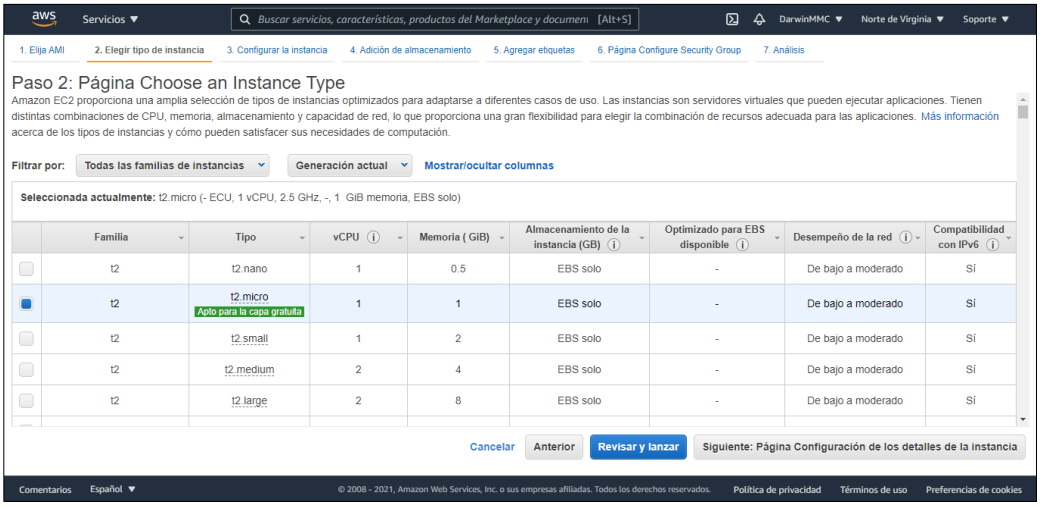


Ilustración 38 aws tipo de instancia

Seguido se debe configurar el grupo de seguridad este actúa como firewall de la instancia, se deben habilitar el puerto 22 correspondiente al puerto por defecto de SSH

y el puerto 80 de HTTP, en el caso de las aplicaciones vuejs estas deben ser ejecutadas en un puerto, por lo que debe ser configurado en este grupo de seguridad.

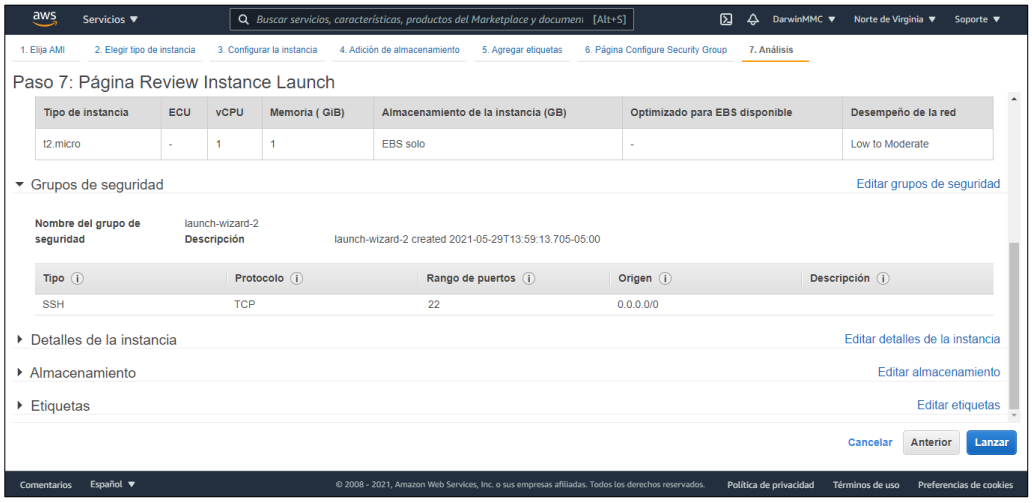


Ilustración 39 AWS EC2 grupo de seguridad

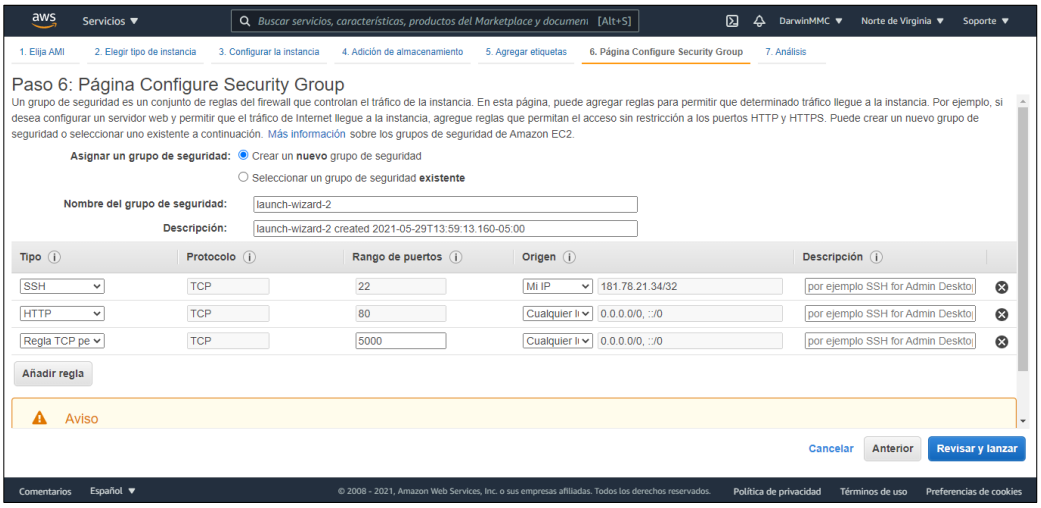


Ilustración 40 AWS EC2 grupo de seguridad

Una vez configurado los aspectos anteriores se procede a lanzar instancia

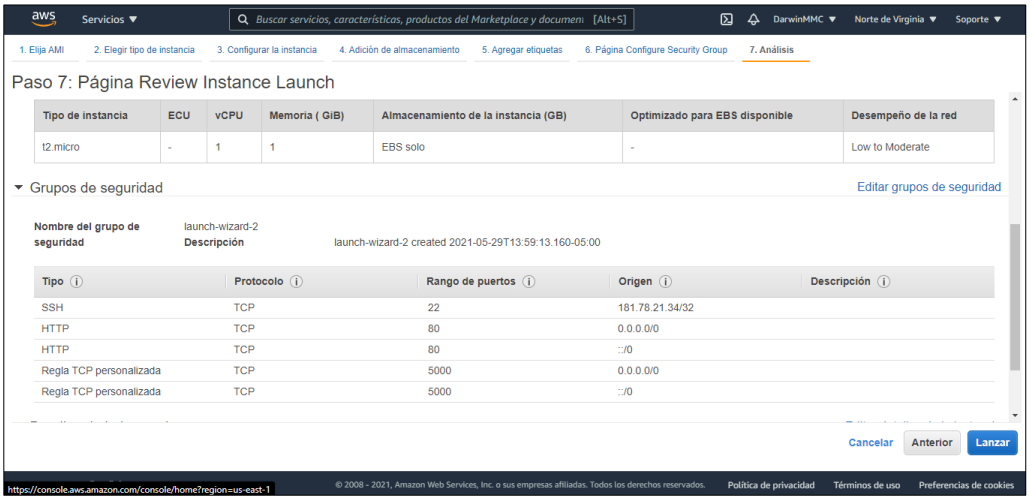


Ilustración 41 aws revisar instancia

Antes de poder lanzar una instancia se debe generar un par de claves, estas permitirán ingresar a la instancia. Esta clave debe permanecer en secreto debido a que por defecto SSH está configurado para solo iniciar sesión por medio de par de clave por lo que es el único modo de iniciar sesión.

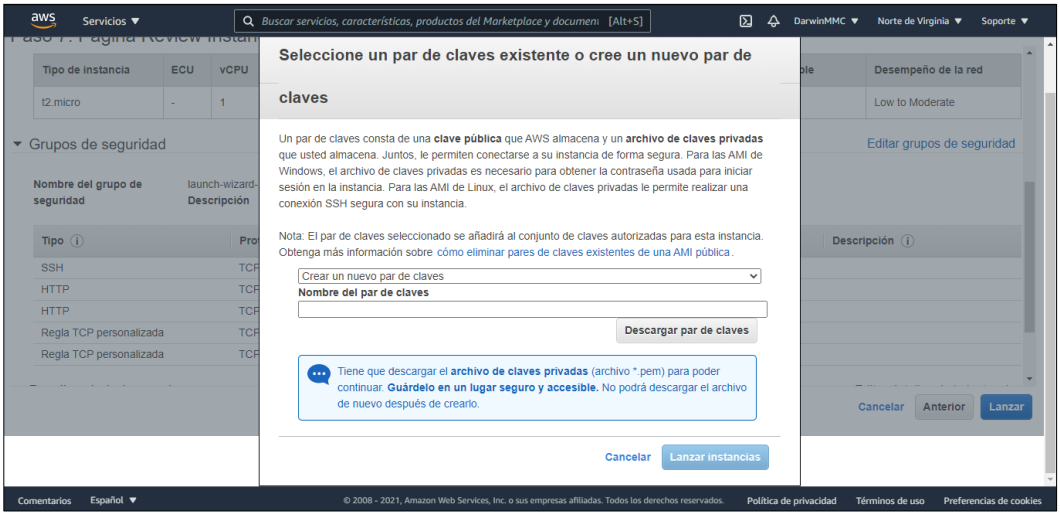


Ilustración 42 AWS KEY-PAR

Si este proceso es realizado desde un equipo Linux se deben configurar los permisos a la clave antes de poder ingresar a la instancia por medio de SSH. El permiso requerido es el siguiente: chmod 400 key-aws.pem

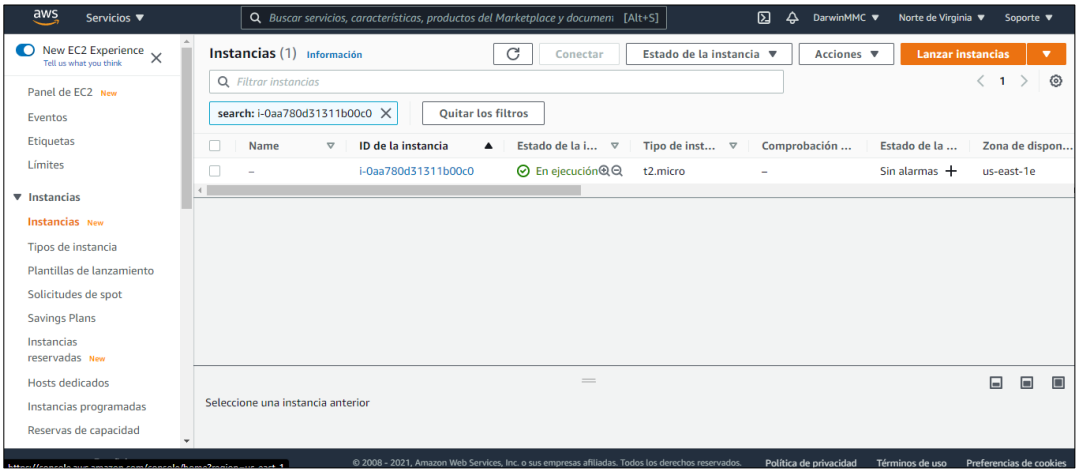


Ilustración 43 AWS instancias

Una vez la instancia se encuentre en ejecución se procede a buscar la ip publica con la que se podrá ingresar.

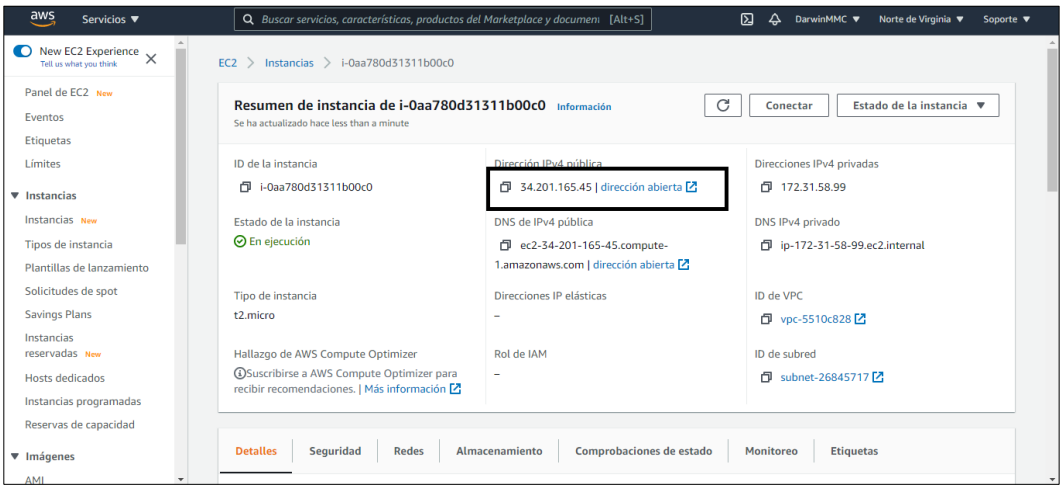


Ilustración 44 EC2 public address

el comando para acceder a la instancia por SSH es el siguiente:

```
darwin@darwin-X555LAB:~/Descargas$ ssh -i key-aws.pem ubuntu@34.201.165.45
```

Ilustración 45 conexión ssh

Donde –i especifica la key-par seguido de ubuntu el cual es el usuario por defecto para esta instancia y la ip publica de la instancia.

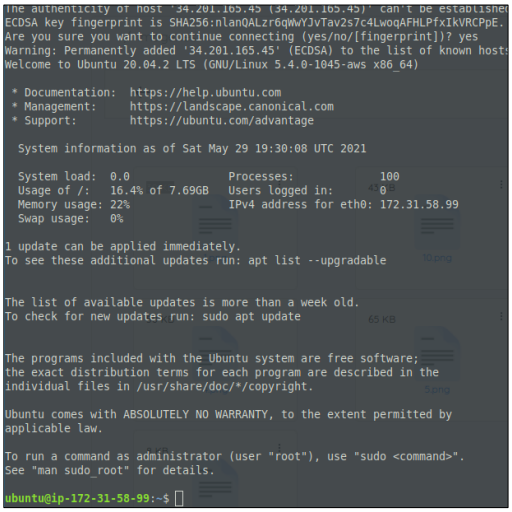


Ilustración 46 Ubuntu server consola

Una vez se tiene acceso a la instancia se procede a realizar los siguientes pasos:

1. **Configurar contraseña del sistema**
   por defecto no se tiene una contraseña asociada por lo que es necesario configurar una, esto se realiza con el siguiente comando.
   sudo passwd root
2. **Actualizar el sistema**
   se debe actualizar el sistema con el objetivo de que este cuente con los últimos repositorios.
   sudo apt-get update

3. **Configurar SSH**
   Dentro del archivo de configuración SSH que se encuentra en **/etc/ssh/sshd_config** se pueden modificar parámetros importantes como el puerto el cual por defecto es el 22, pero principalmente se modificara la configuración para el inicio de sesión por medio de contraseña. Esto se realiza

con el objetivo de poder utilizar SCP, esta es una herramienta asociada a SSH que permite la transferencia de archivos de manera segura.

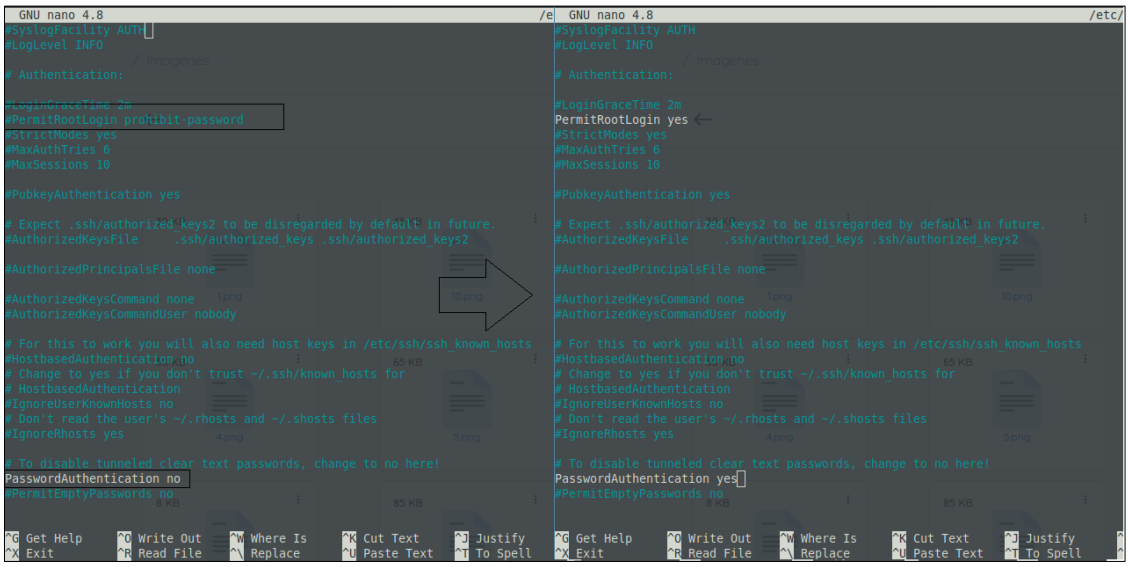


Ilustración 47 configuración SSH

Los parámetros a configurar son los mostrados en la ilustración anterior, para que SSH tome los parámetros configurados se debe reiniciar el servicio con el comando **systemctl restart ssh,** con los parámetros anteriormente configurados se puede acceder a la a instancia por medio de usuario y contraseña ejemplo ssh root@34.201.165.45

4. **Instalar paquetes**

   Para la ejecución de este proyecto se requieren los siguientes paquetes, los cuales se deben instalar con la ejecución del siguiente comando.

   sudo apt-get install apache2 php7.2 php7.2-dom php7.2-pgsql libapache2-mod-php7.2 php7.2-zip php7.2-mbstring php7.2-curl nodejs npm -y

5. **Subir el sistema a la instancia**

   Habitualmente se utilizan servidores FTP para subir proyectos a un hosting pero por motivo de que no se cuenta con servidor FTP se hace uso de la herramienta SCP, ejemplo scp **–r directorio/ usuario@ip:**ruta este comando transferirá el directorio a la cuenta de usuario del sistema especificado por medio de la ip y este será dirigido a la ruta especificada.

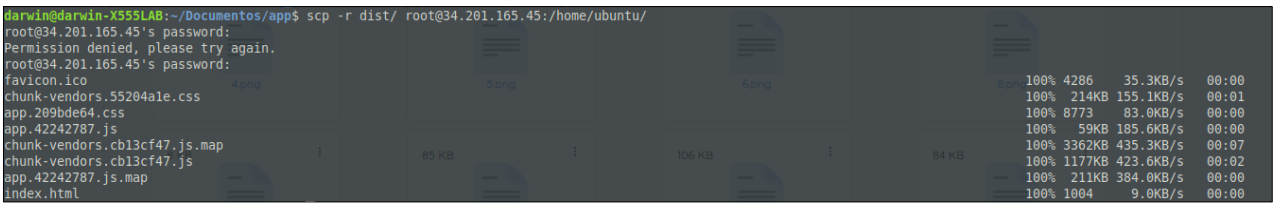


Ilustración 48 Transferencia SCP

Una vez configurado el servidor se procede a ejecutar la aplicación, para ejecutar aplicaciones vuejs se debe hacer uso del paquete serve que debe ser instalado con el comando **sudo npm install serve,** una vez instalado se debe ubicar la carpeta del proyecto y ejecutar **serve –s directorio/** esto iniciará la aplicación y podrá ser accedida desde internet por medio de la ip publica y el puerto especificado.

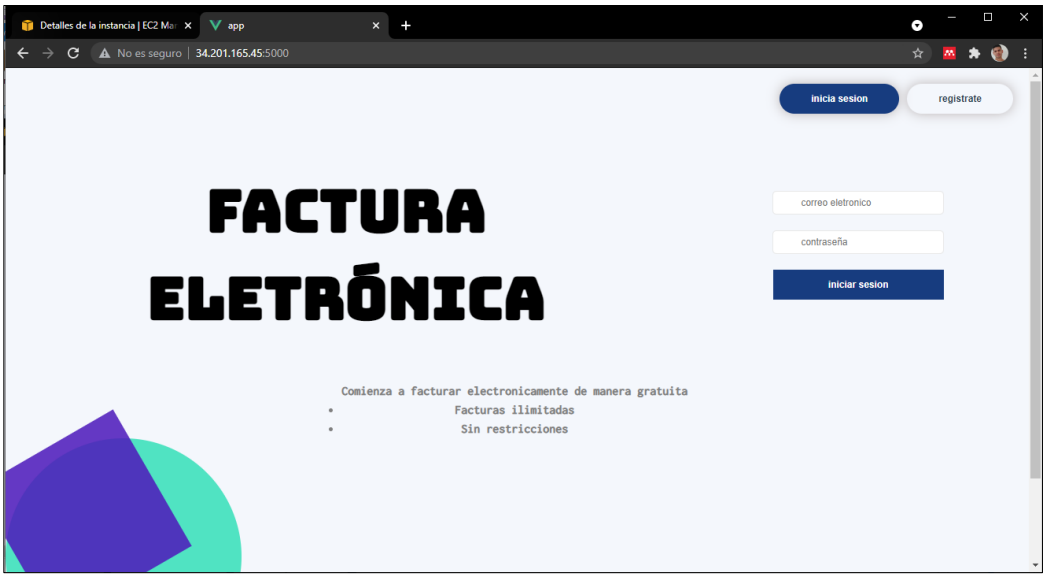


Ilustración 49 aplicación web desde internet

Una vez finalizado el proceso de despliegue de la aplicación web que conforma el frontend del sistema se debe subir el backed a la misma instancia haciendo uso del comando SCP como se mostró anteriormente. A diferencia del frontend el backend está construido con el lenguaje de programación PHP por lo que se necesita un servicio apache que permita ejecutar código PHP, el servicio previamente instalado es apache2

este contiene su directorio de proyectos en /var/www/ donde debe ser agregado el directorio del proyecto.

El sistema requiere conexión a la base de datos por lo que se debe obtener un servicio de base de datos postgres. En aws los gestores de bases de datos son conocidos como los servicios RDS(Relational Database Service), el proceso necesario para crear una instancia RDS es el siguiente:

1. Dirigirse al dashboard del servicio RDS, dar click en DB instance

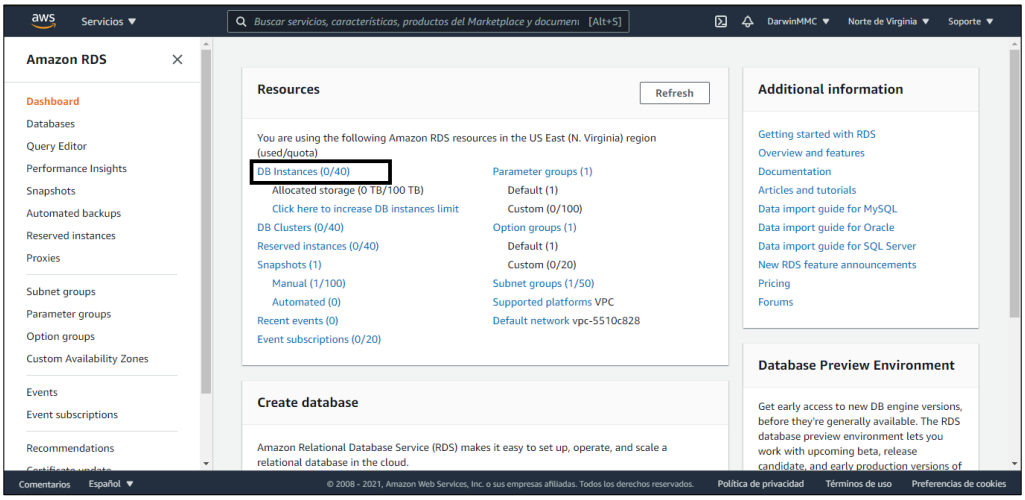


Ilustración 50 RDS

2. Click en créate database

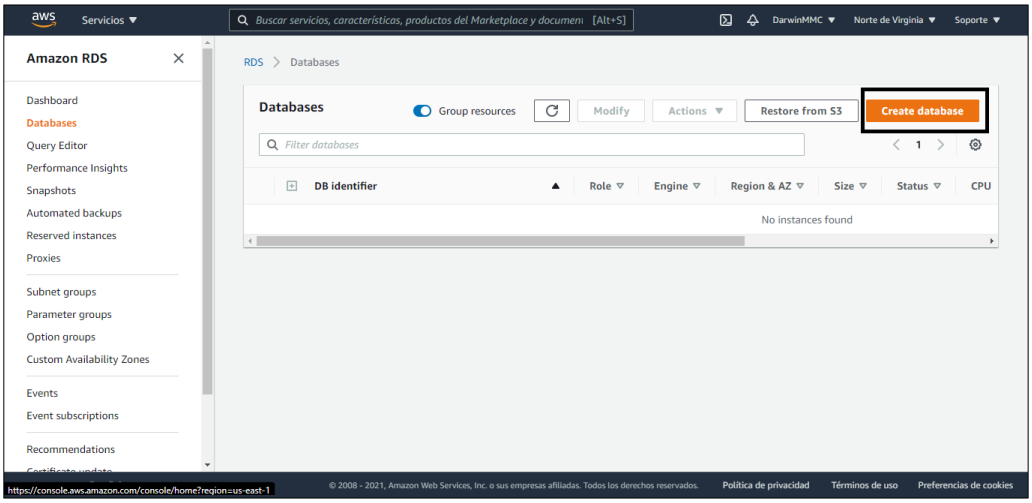


Ilustración 51 RDS

3. Los parámetros de configuración de la instancia deben ser los siguientes

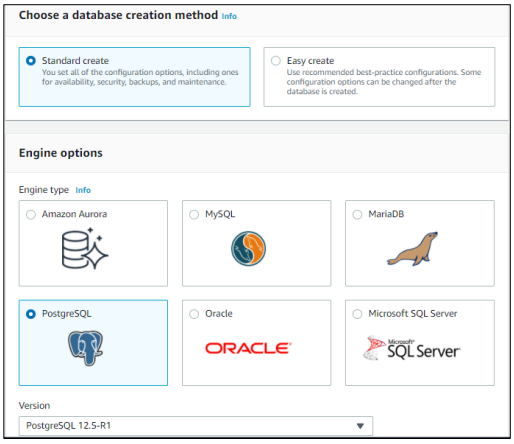


Ilustración 52 configuración RDS 1

Configurar el tipo de destino de la base de datos y establecer el identificador de la base de datos

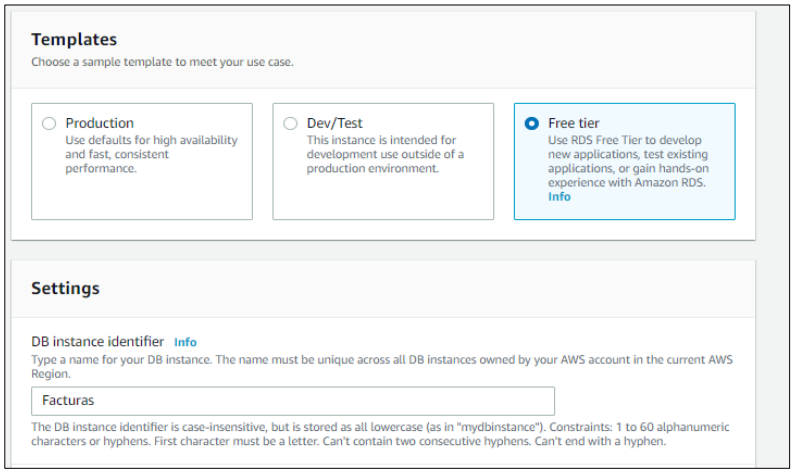


Ilustración 53 configuración RDS 2

Usuario y contraseña para inicio de sesión

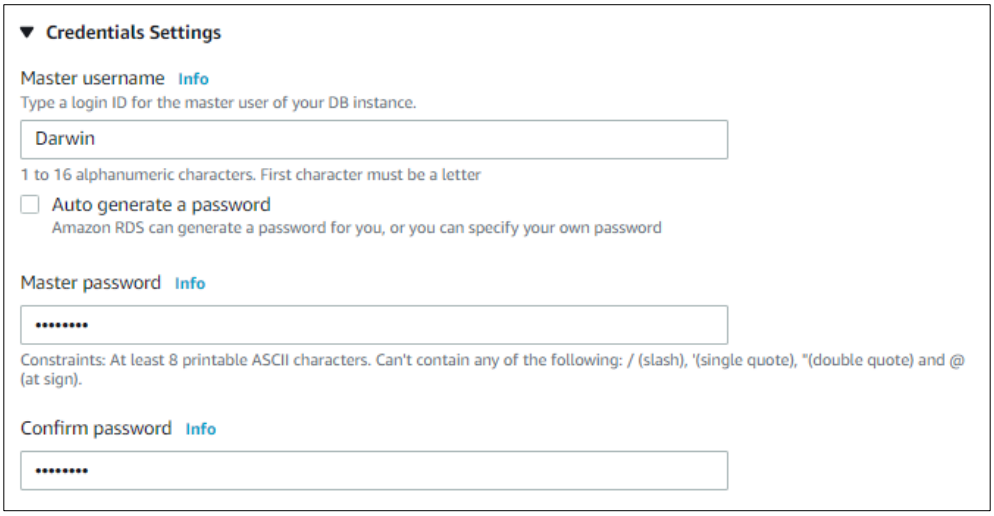


Ilustración 54 configuración RDS 3

En la opción de conectividad se debe seleccionar de acceso público con el objetivo de poder conectarse a la base de datos por fuera de la VPC de Amazon.

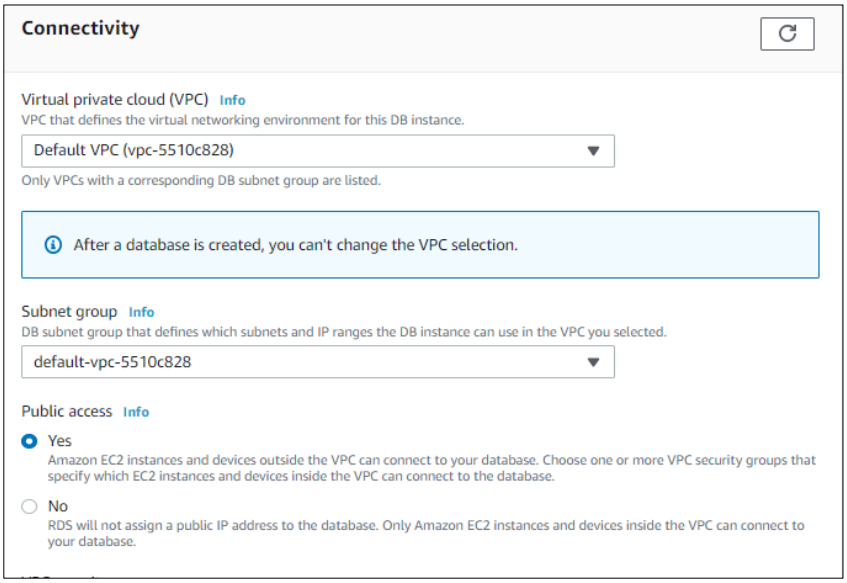


Ilustración 55 configuración RDS 4

Una vez creada la instancia se agregar a la lista

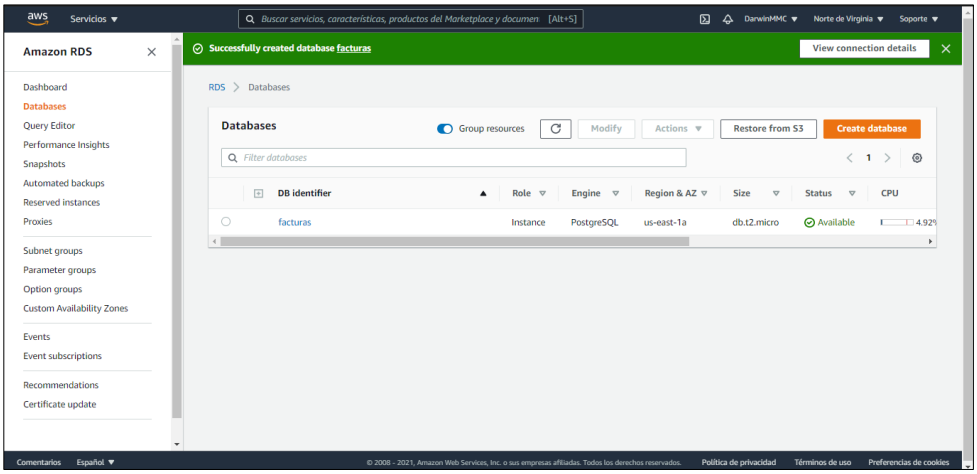


Ilustración 56 instancias RDS

En la información de la instancia se debe localizar el EndPoint, este será punto que nos permitirá tener acceso a la base de datos.

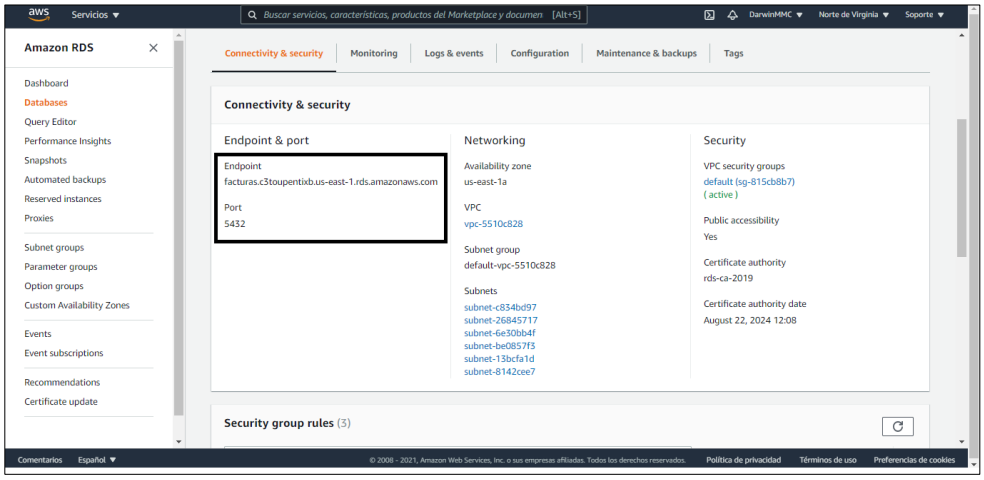


Ilustración 57 datos de RDS

para el ingreso a la base de datos se debe hacer uso de una aplicación cliente que permita gestionar bases de datos postgresql, como lo es pgadmin este permite crear una conexión a servidor de bases de datos de aws RDS. La configuración pgadmin es la siguiente:

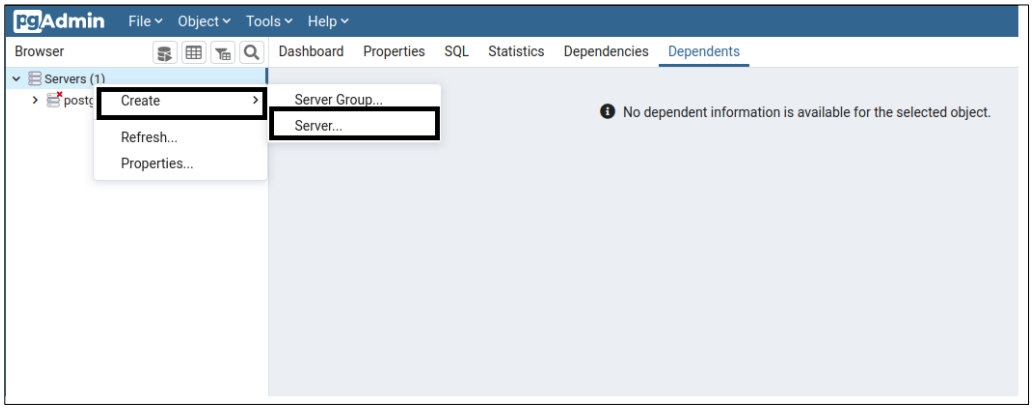


Ilustración 58 pgadmin agregar servidor

Para acceder al servicio se debe especificar el EndPoint, puerto, usuario y clave configurada en el proceso de crear la base de datos.

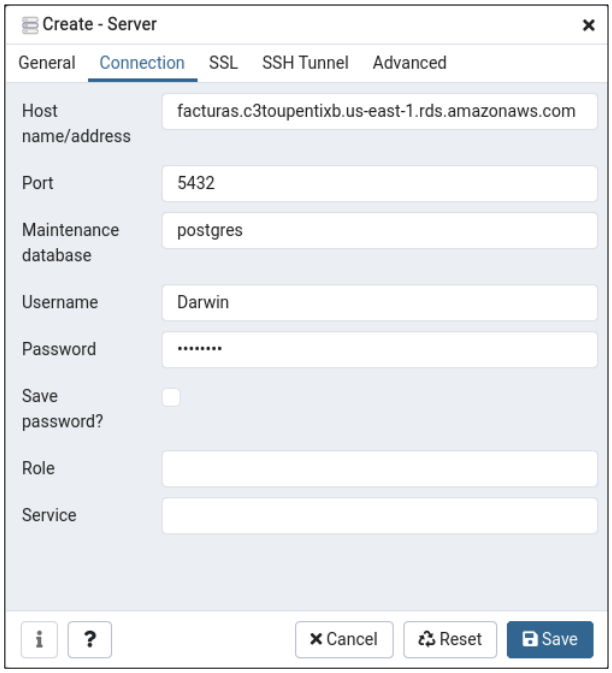


Ilustración 59 pgadmin configuración

una vez conectado al servicio se podrá gestionar la base de datos

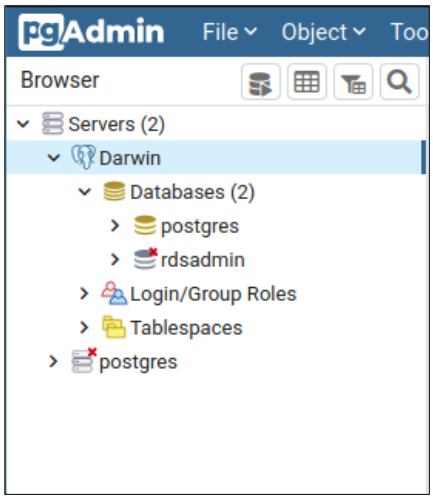


Ilustración 60 acceso a RDS desde pgadmin

Es importante cambiar el punto de acceso en la conexión de pg en PHP de la dirección Loopback 127.0.0.1 por el EndPoint para que el sistema tenga conexión con la base de datos.

3.3 GENERAR FACTURA

El proceso para generar facturas electrónicas de venta se deberá llevar a cabo con el metalenguaje de marcado XML con el formato UBL-2.1 defino por la DIAN en su Resolución No. 000042 del 2020 [3] en el que especifica los formatos implementados los cuales son **Invoice, CreditNote, DebitNote, ApplicationResponse y AttachedDocument.**

3.3.1 Estructura del formato UBL-2.1 Invoice

El formato a seguir se encuentra descrito en anexo técnico, este especifica los aspectos a tener en cuenta para la construcción de la factura. Se describen los elementos y atributos y valores definidos por la DIAN por lo que se aconseja seguir las indicaciones descrita en esta resolución.

Los espacios de nombres requeridos para la construcción de formato Invoice son los siguientes.

```
xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2"
xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:ext="urn:oasis:names:specification:ubl:schema:xsd:CommonExtensionComponents-2"
xmlns:sts="http://www.dian.gov.co/contratos/facturaelectronica/v1/Structures"
xmlns:xades="http://uri.etsi.org/01903/v1.3.2#"
xmlns:xades141="http://uri.etsi.org/01903/v1.4.1#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2  http://docs.oasis-open.org/ubl/os-UBL-2.1/xsd/maindoc/UBL-Invoice-2.1.xsd"
```

Ilustración 61 namespace

La factura XML debe tener el elemento raíz invoice donde especificaran los elementos hijos que deberá contener, cabe aclarar que existen elementos opcionales que pueden

ser adicionados según las necesidades del facturador electrónico. La estructura básica de la factura invoice es la mostrada en la siguiente ilustración.

Ilustración 62 Estructura de factura de venta XML

Para llevar a cabo la construcción de la factura XML se deben tener en cuenta los valores y atributos que define la DIAN en su anexo técnico, por lo que no serán especificados en este documento, se describirá los procesos de cálculo para valores que requiere la factura y la descripción de los principales elementos, para una visión completa y detallada se aconseja que se dirija a anexo técnico.

- **UBLExtensions:**

Este elemento contendrá información técnica con respecto al facturador y la firma digital del documento, la cual se calculará en la siguiente sección.

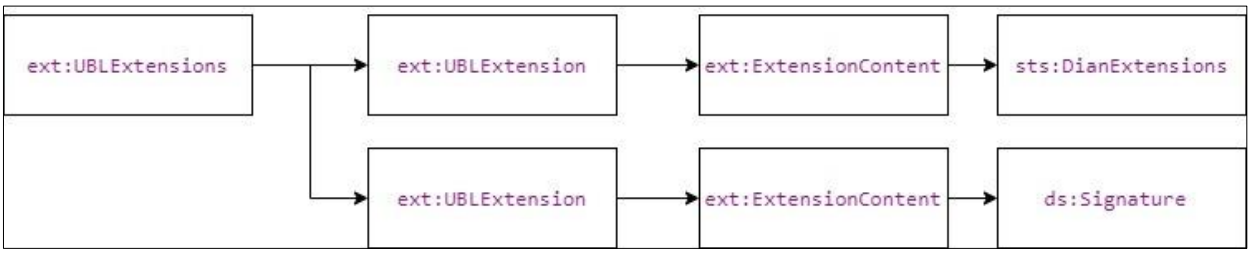


Ilustración 63 estructura UBLExtensions

Este elemento contiene 2 elementos hijos, el primero contendrá la información técnica del facturador electrónico, y el segundo hijo es el elemento que contendrá la Firma electrónica avanzada XML.

- **UBLVersionID:**

  Este elemento especifica la versión del formato UBL utilizado, para este caso se debe especificar UBL-2.1
- **CustomizationID:**

  Este elemento indica el tipo de operación

6.1.5.1. Documento Invoice – Factura electrónica

Código	Significado
01	Combustibles
02	Emisor es Autorretenedor
03	Excluidos y Exentos
04	Exportación
05	Genérica
06	Genérica con pago anticipado
07	Genérica con periodo de facturacion
08	Consorcio
09	Servicios AIU
10	Estándar *
11	Mandatos bienes
12	Mandatos Servicios

Ilustración 64 tipo de operación; tomado de[3]

- **ProfileID**

  Versión del Formato: Indicar versión del documento "DIAN 2.1"

- **ProfileExecutionID**
  Este código describe el ambiente donde será procesada la validación previa de este documento electrónico

Código	Ambiente de Destino
1	Producción
2	Pruebas

Ilustración 65 ambiente destino; tomado de[3]

- **ID**

  Corresponde al Número de factura, Incluye prefijo + consecutivo de factura autorizados por la DIAN. es decir, contendrá el prefijo de facturación + el número correspondiente de la factura según el rango de numeración asignado por la DIAN, por ejemplo, SEPT900000001.

- **UUID**

  Denominado CUFE Código Único de Facturación Electrónica. Elemento que verifica la integridad de la información recibida, este debe ser calculado de la siguiente manera:

Composición del CUFE = SHA-384(NumFac + FecFac + HorFac + ValFac + CodImp1 + ValImp1 + CodImp2 + ValImp2 + CodImp3 + ValImp3 + ValTot + NitOFE + NumAdq + ClTec + TipoAmbie)

Ilustración 66 calculo cufe; tomado de[3]

NumFac	Número de factura.(prefijo concatenado con el número de la factura)
FecFac	Fecha de factura.
HorFac	Hora de la factura incluyendo GMT.
ValFac	Valor de la Factura sin Impuestos, con punto decimal, con decimales truncados a dos (2) dígitos, sin separadores de miles, ni símbolo pesos.
CodImp1	01 Este valor es fijo.
ValImp1	Valor impuesto 01 - IVA, con punto decimal, con decimales truncados a dos (2) dígitos, sin separadores de miles, ni símbolo pesos.. Si no esta referenciado el impuesto 01 – IVA este valor se representa con 0.00
CodImp2	04 Este valor es fijo.
ValImp2	Valor impuesto 04 - Impuesto Nacional al Consumo, con punto decimal, con decimales truncados a dos (2) dígitos, sin separadores de miles, ni símbolo pesos. Si no esta referenciado el impuesto 04- INC este valor se representa con 0.00
CodImp3	03 Este valor es fijo.
ValImp3	Valor impuesto 03 - ICA, con punto decimal, con decimales truncados a dos (2) dígitos, sin separadores de miles, ni símbolo pesos. Si no esta referenciado el impuesto 03 - ICA este valor se representa con 0.00
ValTot	Valor Total, con punto decimal, con decimales truncados a dos (2) dígitos, sin separadores de miles, ni símbolo pesos.
NitFE	NIT del Facturador Electrónico sin puntos ni guiones, sin digito de verificación.
NumAdq	Número de identificación del adquirente sin puntos ni guiones, sin digito de verificación.
ClTec	Clave técnica del rango de facturación.
TipoAmbiente	Número de identificación del ambiente utilizado por el contribuyente para emitir la factura validar el numeral 6.1.1.

Ilustración 67 valores para cálculo de cufe; tomado de[3]

- **IssueDate**
  Fecha de emisión de la factura debe estar en formato GMT.
- **IssueTime**
  Hora de emisión de la factura en formato GMT.
- **DueDate**
  Fecha de vencimiento de la factura.
- **InvoiceTypeCode**
  Corresponde un valor de la siguiente tabla

Código	*Significado*	*Uso*
01	Factura electrónica de Venta	Tipos de factura
02	Factura electrónica de venta - exportación	
03	Documento electrónico de transmisión – tipo 03	
04	Factura electrónica de Venta - tipo 04	
91	Nota Crédito	Exclusivo en referencias a documentos (elementos *DocumentReference*)
92	Nota Débito	

Ilustración 68 tipo de factura; tomado de [3]

- **Note**
  Este elemento contiene información adicional a la factura.
- **DocumentCurrencyCode**
  Divisa aplicable a toda la factura para Colombia debe ser COP.
- **LineCountNumeric**
  Número o cantidad de elementos InvoiceLine de la factura “numero de productos”.
- **InvoicePeriod**
  Intervalo de fechas a las que hace referencia la factura “periodo de facturación”.
- **OrderReference**
  Grupo de elementos para información que describen una exclusiva orden para esta factura.
- **BillingReferences:**
  Este elemento se debe diligenciar únicamente cuando la factura se origina a partir de la corrección o ajuste de una Nota Débito o Nota Credito.
- **DespatchDocumentReference:**
  Se utiliza cuando se necesita referenciar uno o más documentos de despacho asociado a la factura realizada.
- **ReceiptDocumentReference:**
  Se utiliza cuando se necesita referenciar uno o más documentos de recepción asociado a la factura realizada.
- **AccountingSupplierParty**
  Grupo con información del facturador electrónico.
- **AccountingCustomerParty**
  Grupo con información del Adquirente.
- **TaxRepresentativeParty**
  Grupo con información de la Persona autorizada para descargar documentos.
- **Delivery**
  Grupo con información para entrega de bienes.
- **DeliveryTerms**
  Grupo con información relacionada con la entrega.
- **PaymentMeans**
  Grupo con información relacionada con el pago de la factura.
- **PrePaidPayment**
  Grupo con información relacionada con un anticipo.
- **AllowanceCharge**

Grupo con información relacionadas con un cargo o un descuento.

- **TaxTotal**
  Grupo con información de valores totales relacionadas con un tributo.
- **WithholdingTaxTotal**
  Grupo con información de valores totales relacionas con los tributos retenidos.
- **LegalMonetaryTotal**
  Grupo con información relacionada con los valores totales aplicables a la factura.
- **InvoiceLine**
  Grupo con información relacionada con una línea de factura, información de cada producto.

La construcción de la factura XML en el lenguaje de programación PHP se puede realizar utilizando el api DOM, para PHP se conoce como DOMDocument. Esta permite crear el documento haciendo uso de las herramientas que esta clase posee, pero al tratarse de un documento medianamente largo se recomienda hacer la estructura XML como una plantilla de tipo de datos string como se muestra a continuación.

```
<?php

$xml = '<Invoice
  xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2"
  xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
  xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:ext="urn:oasis:names:specification:ubl:schema:xsd:CommonExtensionComponents-2"
  xmlns:sts="http://www.dian.gov.co/contratos/facturaelectronica/v1/Structures"
  xmlns:xades="http://uri.etsi.org/01903/v1.3.2#"
  xmlns:xades141="http://uri.etsi.org/01903/v1.4.1#"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2
  http://docs.oasis-open.org/ubl/os-UBL-2.1/xsd/maindoc/UBL-Invoice-2.1.xsd">
  <ext:UBLExtensions>
      <ext:UBLExtension/>
      <ext:UBLExtension/>
    </ext:UBLExtensions>
  <cbc:UBLVersionID/>
  <cbc:CustomizationID/>
  <cbc:ProfileID/>
```

```
<cbc:ProfileExecutionID/>
<cbc:ID/>
<cbc:UUID/>
<cbc:InvoiceTypeCode/>
<cbc:DocumentCurrencyCode/>
<cbc:LineCountNumeric/>
<cac:AccountingSupplierParty/>
<cac:AccountingCustomerParty/>
<cac:PaymentMeans/>
<cac:LegalMonetaryTotal/>
<cac:InvoiceLine/>
</Invoice>';

$dom = new DOMDocument("1.0","UTF-8");
$dom->loadXML($xml);

?>
```

Esto permitirá elaborar la factura de una manera más simplificada, lo importante es seguir la estructura jerárquica de los elementos que requiere la factura, DOM permite cargar un string de xml a un objeto DOM por medio de loadXML lo que facilitara el proceso para calcular el XML canonico antes de calcular el resumen de la factura.

3.3.2 Digito de verificación

En algunos elementos donde se especifican los documentos de identidad es necesario incluir un dígito de verificación. El dígito de verificación es un número que se asigna a las personas naturales o jurídicas que se registran en el Rut, este digito se utiliza para distinguir si es un contribuyente o no, este le permite a la DIAN evitar problema de duplicidad, fraude o suplantación. El algoritmo para el cálculo del digito de verificación se encuentra descrito en la orden administrativa 4 del año 1989 expedido por la DIAN [28] donde se explica de manera detalla este proceso.

A continuación, se muestra la función implementada para el cálculo del digito de verificación.

```php
<?php
function getDV(int $nit){
  $cedula = str_split($nit);
  $length_cedula = count($cedula);
  $multiplos = array(71,67,59,53,47,43,41,37,29,23,19,17,13,7,3);
  $posicion = count($multiplos) - $length_cedula;
  $sum = 0;

  for ($i=0; $i<$length_cedula; $i++) {
    $sum += $cedula[$i] * $multiplos[$i+$posicion];
  }
  $mod = $sum%11;

  if ($mod > 1)
    return (11 - $mod);

  return $mod;
 }
?>
```

El algoritmo de cálculo de digito de verificación es una sumatoria que multiplica los numero primos de manera ascendente por cada uno de los dígitos del documento de identidad, a esta sumatoria se le debe calcular el módulo 11, para cualquier número mayor a 1 se deberá restar el modulo al número 11.

3.4 FIRMA XML

La política de firma se implementa con el objetivo de garantizar la autenticidad y confiabilidad de todos los documentos que soportan las transacciones de factura electrónica en Colombia. La Firma está indicada y referenciada para todos los documentos electrónicos que componen el conjunto de documentos del comercio electrónico, denominado Facturación Electrónica establecida por el Gobierno Nacional a cargo de la DIAN. Para todos los documentos que componen la facturación electrónica la firma se hará mediante la inclusión de una etiqueta Signature dentro segundo elemento UBLExtesion del formato estándar de intercambio XML.

3.4.1 Actores de la Firma

Los actores involucrados en la facturación electrónica son:

- **Facturador Electrónico**
  Persona natural o jurídica que está obligada o quiere facturar de manera voluntaria como dictan las condiciones establecidas en la normatividad vigente[2].
- **Adquirente**
  En términos de facturación electrónica es el receptor de la factura electrónica.
- **Proveedor Tecnológico**
  El proveedor tecnológico puede ser el firmante autorizado por el facturador electrónico a actuar en su nombre.

3.4.2 Formato de Firma

Para llevar a cabo el proceso de firma se debe utilizar el estándar XMLDSig enveloped con formato XAdES-EPES según la especificación técnica ETSI TS 101 903[22]. El formato XAdES de firma digital avanzada adoptado por la DIAN [3] debe incluir los elementos **X509Data**, **Object**, **SignaturePolicyIdentifier** y **SignerRole**. Se admiten como válidos los algoritmos de generación de hash, codificación en base64, firma, normalización y transformación definidos en el estándar XMLDSig.

3.4.3 Algoritmos de firma

El algoritmo de firma usado sobre el elemento SignedInfo para la firma digital de la factura electrónica puede ser cualquiera de los definidos en la especificación XMLDSIG [21] que actualmente son:

- ✓ Cifrado RSA con resumen SHA256
- ✓ Cifrado RSA con resumen SHA384
- ✓ Cifrado RSA con resumen SHA512

3.4.4 Algoritmo Canonical XML

El algoritmo de canonicalización usado para la firma digital de la factura electrónica es “Canonical XML” [10] Para esto se debe usar el valor “http:/www.w3.org/TR/2001/REC-xml-c14n-20010315” dentro del elemento CanonicalizationMethod.

3.4.5 Certificado Digital

El certificado exigido por la DIAN para la realización de la firma digital debe ser un certificado proveniente de una entidad avalada por la ONAC (Organismo Nacional de Acreditación de Colombia), para la elaboración de este proyecto se adquirió un certificado con la entidad GSE [29] por valides de un año, la cual al momento de la ejecución de este proyecto es una entidad avalada por la ONAC, esta entidad genera un certificado digital que contiene una clave privada y una clave pública.

3.4.6 Proceso de firma

3.4.6.1 Estructura del formato XAdES-EPES

La estructura del formato XdES-EPES utilizada bajo los criterios previsto en la resolución 000042 [3] es la siguiente.

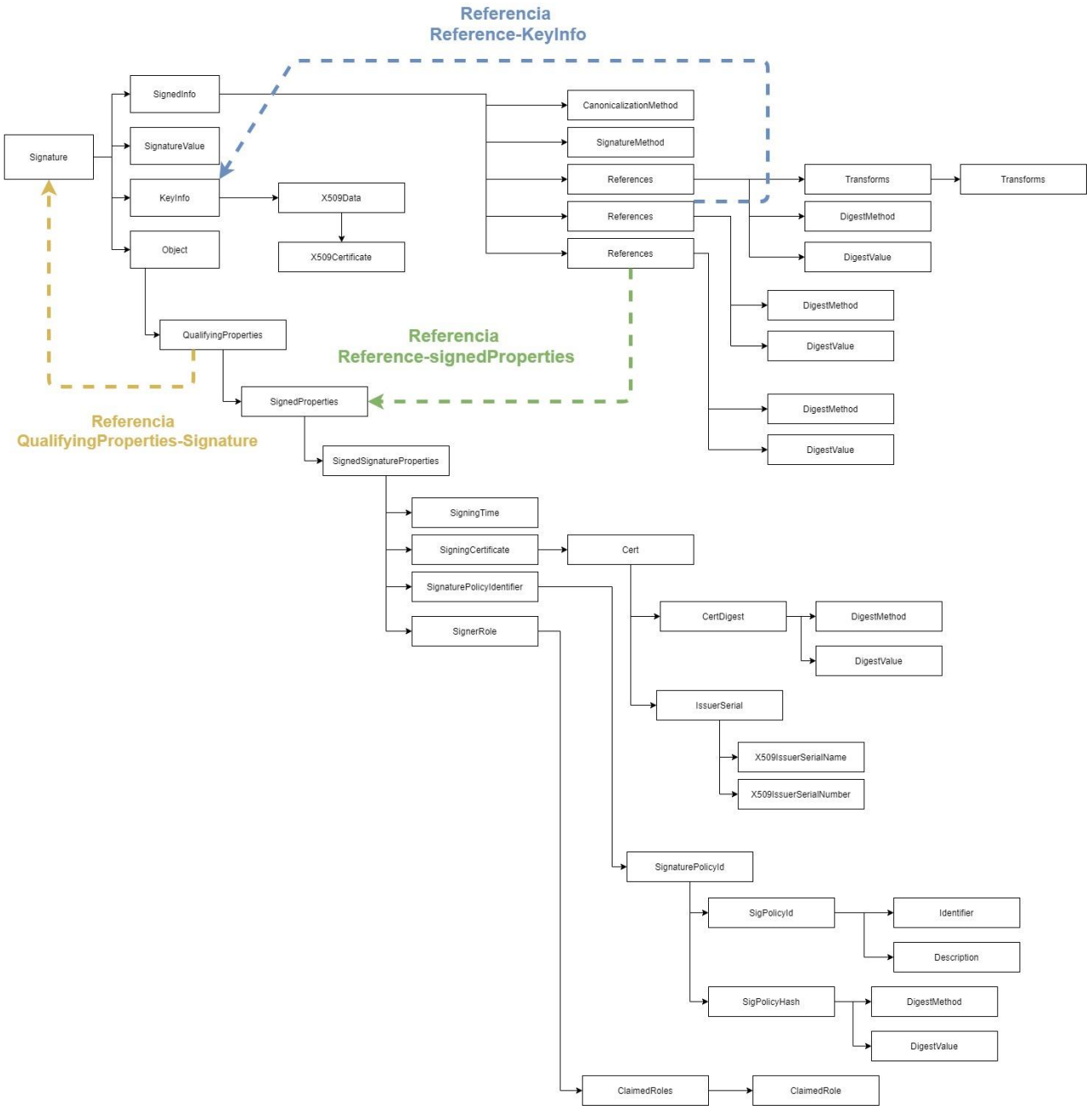


Ilustración 69 estructura XAdES-EPES

Descripción de los elementos:

- **SignedInfo**

  Este elemento incluye el algoritmo de canonicalización, algoritmo de firma y una o más referencias, la firma digital debe ser aplicada a este elemento y debe ser incluida en el elemento SignatureValue**.**

➢ **CanonicalizationMethod**

Este elemento contendrá la especificación del método canonical XML mediante el atributo Algorithm con valor "https://www.*w3*.org/TR/2001/REC-xml-c14n-20010315"

➢ **SignatureMethod**

Este elemento contendrá la especificación del algoritmo de firma, para este proyecto se seleccionó RSA con SHA256, este debe ser especificado mediante el atributo Algorithm con valor "https://www.w3.org/2001/04/xmldsig-more#rsa-sha256"

➢ **Reference**

El primer elemento Reference contendrá el hash de la factura XML canonicalizada con el algoritmo C14N, este debe contener un atributo ID con identificador único y un atributo URI con valor vacío como se indica en el RFC 2396[30], este documento describe un "superconjunto" de operaciones que se pueden aplicar al atributo URI, el cual indica que si el componente de la ruta URI está vacío entonces hará referencia al documento actual.

❖ **Transforms**
Los formatos de firma XMLDSIG y XAdES permiten la configuración de transformaciones XML personalizada. Las transformaciones se aplican a un documento XML antes de firmarlo, de tal forma que no se firma el XML original, sino el transformado. Para este elemento se utilizará la transformación por defecto la cual debe ser anunciada por medio del atributo Algorithm con valor http://www.w3.org/2000/09/xmldsig#enveloped-signature

Ilustración 70 transformaciones permitidas; tomado de [21]

- **DigestMethod**
  Este atributo indicara el algoritmo de resumen que se le aplicara a la factura Invoice el cual debe ser especificado por medio del atributo Algorithm con valor “http://www.w3.org/2001/04/xmlenc#sha256”
- **DigestValue**
  En este elemento se insertará el resumen calculado de la factura mediante el algoritmo SHA256.

- **References**
  El segundo Reference contendrá el resumen del elemento KeyInfo el cual debe ser incluida en el elemento DigestValue, este elemento debe hacer referencia al ID del elemento Keyinfo como se muestra en la ilustración 43, igualmente debe contener la especificación del algoritmo utilizado dentro del elemento DigestMethod.
- **References**
  El tercer Reference contendrá el Resumen del elemento SignedProperties el cual debe ser incluida en el elemento DigestValue, este elemento debe hacer referencia al ID del elemento SignedProperties como se muestra en la ilustración 43, igualmente debe contener la especificación del algoritmo utilizado dentro del elemento DigestMethod.

- **SignatureValue**
  En este elemento debe incluirse la firma del elemento SignedInfo codificada en base64.
- **KeyInfo**
  Este elemento permite al destinatario obtener la clave pública necesaria para validar la firma. Este debe ser incluido específicamente dentro del elemento X509Certificate.
- **Object**
  Este es un elemento opcional que puede aparecer una o más veces.
  - **SigningTime**
    Este elemento contendrá el sello de tiempo el cual debe ser incluido en formato GMT.
  - **SigningCertificate**
    Este elemento contendrá información importante acerca del certificado con el que se aplicó la firma.
  - **SignaturePolicyIdentifier**
    Este elemento contendrá información definida por la DIAN se especificará más adelante durante el proceso de firma.
  - **SignerRole**
    Este elemento especifica el ROL del facturador, igualmente este elemento es especificado por la DIAN.

A continuación, se muestra el formato XAdES-EPES implementado

```
<ds:Signature Id="identificador">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
      <ds:Reference Id="identificador-ref0" URI="">
       <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
              </ds:Transforms>
              <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
              <ds:DigestValue/>
      </ds:Reference>
      <ds:Reference URI="#identificador_keyinfo">
              <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
               <ds:DigestValue/>
               </ds:Reference>
```

```
        <ds:Reference Type="http://uri.etsi.org/01903#SignedProperties"
        URI="#identificador_SignedProperties">
          <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
          <ds:DigestValue/>
         </ds:Reference>
</ds:SignedInfo>

<ds:SignatureValue/>

<ds:KeyInfo Id="identificador_keyinfo">
        <ds:X509Data>
          <ds:X509Certificate/>
        </ds:X509Data>
</ds:KeyInfo>

<ds:Object>
        <xades:QualifyingProperties Target="#identificador">
           <xades:SignedProperties Id="identificador_SignedProperties">
            <xades:SignedSignatureProperties>
              <xades:SigningTime/>
               <xades:SigningCertificate>
                <xades:Cert>
                 <xades:CertDigest>
                  <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
                  <ds:DigestValue/>
                 </xades:CertDigest>
                 <xades:IssuerSerial>
                   <ds:X509IssuerName/>
                   <ds:X509SerialNumber/>
                  </xades:IssuerSerial>
                </xades:Cert>
               </xades:SigningCertificate>

                <xades:SignaturePolicyIdentifier>
                  <xades:SignaturePolicyId>
                    <xades:SigPolicyId>
                      <xades:Identifier/>
                       <xades:Description/>
                         </xades:SigPolicyId>
                         <xades:SigPolicyHash>
                           <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
                            <ds:DigestValue/>
                         </xades:SigPolicyHash>
                    </xades:SignaturePolicyId>
                  </xades:SignaturePolicyIdentifier>
                  <xades:SignerRole>
                     <xades:ClaimedRoles>
                        <xades:ClaimedRole/>
                     </xades:ClaimedRoles>
                  </xades:SignerRole>
               </xades:SignedSignatureProperties>
            </xades:SignedProperties>
          </xades:QualifyingProperties>
        </ds:Object>
        </ds:Signature>
```

3.4.6.2 Firma

Antes de proceder con el firmado del documento se debe tener en cuenta los espacios de nombres (Ilustración 61 namespace) utilizados en la factura, estos deben ser anexado al componente que se le aplica el cálculo de resumen o de firma, pero no debe ser incluido en el documento final. Para realizar el proceso de firma debemos seguir los siguientes pasos.

1. **Generar identificadores únicos**

   Se deben generar identificadores para los elementos Signature, Reference, KeyInfo, SignedProperties estos identificadores por lo generar son generados aleatoriamente (UUID), estos identificadores deben ser agregados mediante el uso de el atributo Id como se muestra a continuación.

```
<ds:Signature Id="identificador-signature">
   <ds:SignedInfo>
      <ds:CanonicalizationMethod/>
      <ds:SignatureMethod/>
      <ds:Reference Id="identificador-ref0">
         <ds:Transforms>
            <ds:Transform/>
         </ds:Transforms>
         <ds:DigestMethod/>
         <ds:DigestValue/>
      </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue/>
   <ds:KeyInfo Id="identificador-keyinfo">
      <ds:X509Data>
         <ds:X509Certificate/>
      </ds:X509Data>
   </ds:KeyInfo>
   <ds:Object>
      <xades:QualifyingProperties>
            <xades:SignedProperties Id="identificador-SignedProperties">
            </xades:SignedProperties>
   </xades:QualifyingProperties>
</ds:Object>
</ds:Signature>
```

2. SignedInfo

- Incluir el atributo `Algorithm=”http://www.w3.org/TR/2001/REC-xml-c14n-20010315”` en el elemento `Signature:SignedInfo:CanonicalizationMethod`.
- Incluir el atributo `Algorithm=”http://www.w3.org/2001/04/xmldsig-more#rsa-sha256”` en el elemento `Signature:SignedInfo:SignatureMethod`

3. Reference 1

- Incluir el atributo `URI=""` en `Signature:SinedInfo:Reference`
- Incluir el atributo `Algorithm=”http://www.w3.org/2000/09/xmldsig#enveloped-signature”` en el elemento `Signature:SignedInfo:Reference:Transforms:Transform`
- Incluir el atributo `Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"` en el elemento `Signature:SignedInfo:References:DigestMethod`
- Insertar en `Signature:SinedInfo:Reference:DigestValue` el cálculo de resumen SHA-256 codificado en base64 de la factura canonicalizada con C14N.

```
<ds:Reference Id="identificador-ref0" URI="">
  <ds:Transforms>
    <ds:Transform Algorithm=”http://www.w3.org/2000/09/xmldsig#enveloped-signature”/>
  </ds:Transforms>
  <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
   <ds:DigestValue> akcOQ5qEh4dkMwt0d5BoXRR8Bo4vdy9DBZtfF5O0SsA=</ds:DigestValue>
 </ds:Reference>
```

4. KeyInfo

- Incluir la clave pública en `Signature:KeyInfo:X509Data:X509Certificate`

```
<ds:KeyInfo Id="identificador-keyinfo">
  <ds:X509Data>
    <ds:X509Certificate>clave pública</ds:X509Certificate>
  </ds:X509Data>
</ds:KeyInfo>
```

5. Reference 2

- Incluir `URI="#identificador-keyinfo"` en el elemento `Signature:SignedInfo:Reference`
- Incluir el atributo `Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"` en el elemento `Signature:SignedInfo:Reference:DigestMethod`
- Incluir el cálculo de resumen SHA-256 codificado en base64 del elemento KeyInfo y canonizado con C14N.

```
<ds:Reference URI="#identificador-keyinfo”>
  <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
  <ds:DigestValue>troRYR2fcmJLV6gYibVM6XlArbddSACZJP44=</ds:DigestValue>
</ds:Reference>
```

6. SignedProperties

   Para la construcción del elemento SignedPropierties se debe seguir la siguiente estructura.

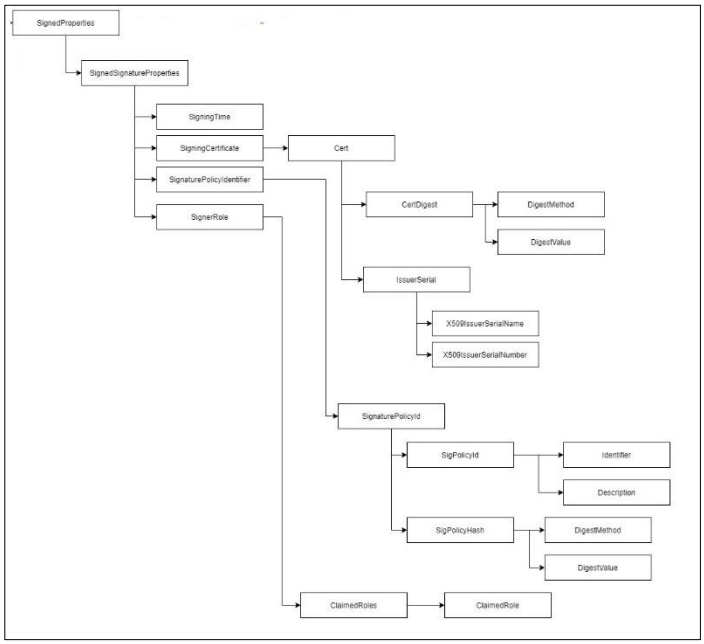


Ilustración 71 esquema SignedPropertis

```
<xades:SignedProperties Id="identificador-SignedProperties">
        <xades:SignedSignatureProperties>
```

```
<xades:SigningTime/>
<xades:SigningCertificate>
        <xades:Cert>
                <xades:CertDigest>
                        <ds:DigestMethod/>
                        <ds:DigestValue/>
                </xades:CertDigest>
                <xades:IssuerSerial>
                        <ds:X509IssuerName/>
                        <ds:X509SerialNumber/>
                </xades:IssuerSerial>
        </xades:Cert>
</xades:SigningCertificate>
<xades:SignaturePolicyIdentifier>
        <xades:SignaturePolicyId>
                <xades:SigPolicyId>
                        <xades:Identifier/>
                </xades:SigPolicyId>
                <xades:SigPolicyHash>
                        <ds:DigestMethod/>
                        <ds:DigestValue/>
                </xades:SigPolicyHash>
        </xades:SignaturePolicyId>
</xades:SignaturePolicyIdentifier>
<xades:SignerRole>
        <xades:ClaimedRoles>
                <xades:ClaimedRole/>
        </xades:ClaimedRoles>
</xades:SignerRole>
</xades:SignedSignatureProperties>
</xades:SignedProperties>
```

- Incluir fecha y hora en formato GMT dentro del elemento `Signature:Object:QualifyingProperties:SignedProperties:SignedSignatureProperties:SigningTime` por ejemplo 2021-06-21T19:09:35-05:00

- Incluir el atributo `Algorithm="http://www.w3.org/2001/04/xmlenc#sha256”` en `Signature:Object:QualifyingProperties:SignedProperties:SignedSignatureProperties:SigningCertificate:Cert:CertDigest:DigestMethod`

- Incluir el resumen SHA-256 de la clave pública en el elemento `Signature:Object:QualifyingProperties:SignedProperties:SignedSignatureProperties:SigningCertificate:Cert:CertDigest:DigestValue`

- Incluir el nombre del emisor (Issuere) de la clave pública en el elemento `Signature:Object:QualifyingProperties:SignedProperties:SignedSignatureProperties:SigningCertificate:Cert:IssuerSerial:X509IssuerName`, cabe aclarar que estos datos están incluidos dentro del certificado con extensión p12.
- Incluir el número de serie (SerialNumber) en el elemento `Signature:Object:QualifyingProperties:SignedProperties:SignedSignatureProperties:SigningCertificate:Cert:IssuerSerial:X509SerialNumber`
- Incluir la cadena `"https://facturaelectronica.dian.gov.co/politicadefirma/v1/politicadefirmav2.pdf"` en el elemento `Signature:Object:QualifyingProperties:SignedProperties:SignedSignatureProperties:SignaturePolicyIdentifier:SignaturePolicyId:SigPolicyId:Identifier`
- Incluir el atributo `Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"` en el elemento `Signature:Object:QualifyingProperties:SignedProperties:SignedSignatureProperties:SignaturePolicyIdentifier:SignaturePolicyId:SigPolicyHash:DigestMethod`
- Incluir el cálculo de resumen SHA-256 codificado en base64 del elemento `Signature:Object:QualifyingProperties:SignedProperties:SignedSignatureProperties:SignaturePolicyIdentifier:SignaturePolicyId:SigPolicyId:Identifier` en el elemento `Signature:Object:QualifyingProperties:SignedProperties:SignedSignatureProperties:SignaturePolicyIdentifier:SignaturePolicyId:SigPolicyHash:DigestValue`
- El elemento `Signature:Object:QualifyingProperties:SignedProperties:SignedSignatureProperties:SignerRole:ClaimedRoles:ClaimedRole` contiene uno de los siguientes valores [3]:
  - ✓ “supplier” cuando la firma de la factura la realiza el Obligado a Facturar.
  - ✓ “third party” cuando la firma la realiza un Proveedor Tecnológico.

```
<xades:SignedProperties Id="identificador-SignedProperties">
 <xades:SignedSignatureProperties>
```

```
<xades:SigningTime>2021-06-21T19:09:35-05:00</xades:SigningTime>
 <xades:SigningCertificate>
  <xades:Cert>
        <xades:CertDigest>
          <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
<ds:DigestValue>nem6KXhqlV0A0FK5o+MwJZ3Y1aHgmL1hDs/RMJu7HYw=</ds:DigestValue>
        </xades:CertDigest>
        <xades:IssuerSerial>
          <ds:X509IssuerName>C=CO,L=Bogota D.C.,O=Andes SCD.,OU=Division de certificacion entidad
final,CN=CA ANDES SCD S.A. Clase
II,1.2.840.113549.1.9.1=#1614696e666f40616e6465737363642e636f6d2e636f</ds:X509IssuerName>
          <ds:X509SerialNumber>7785324499979575522</ds:X509SerialNumber>
        </xades:IssuerSerial>
  </xades:Cert>
 </xades:SigningCertificate>
 <xades:SignaturePolicyIdentifier>
  <xades:SignaturePolicyId>
        <xades:SigPolicyId>
        <xades:Identifier>https://facturaelectronica.dian.gov.co/politicadefirma/v1/politicadefirmav2.pdf</xades:Ident
        ifier>
        </xades:SigPolicyId>
        <xades:SigPolicyHash>
          <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
    <ds:DigestValue>dMoMvtcG5aIzgYo0tIsSQeVJBDnUnfSOfBpxXrmor0Y=</ds:DigestValue>
        </xades:SigPolicyHash>
        </xades:SignaturePolicyId>
        </xades:SignaturePolicyIdentifier>
        <xades:SignerRole>
                <xades:ClaimedRoles>
                        <xades:ClaimedRole>supplier</xades:ClaimedRole>
                </xades:ClaimedRoles>
        </xades:SignerRole>
</xades:SignedSignatureProperties>
</xades:SignedProperties>
```

7. Reference 3
   - Incluir los atributos `Type="http://uri.etsi.org/01903#SignedProperties" URI="#identificador-SignedProperties”` en el elemento `Signature:SignedInfo:Reference`
   - Incluir el atributo `Algorithm=”http://www.w3.org/2001/04/xmlenc#sha256”` en el elemento `Signature:SignedInfo:Reference:DigestMethod`
   - Incluir el cálculo de resumen SHA-256 codificado en base64 del elemento `SignedProperties` y canonizado con C14N en el elemento `Signature:SignedInfo:Reference:DigestValue`

8. SignatureValue
   una vez calculado los resúmenes del elemento SignedInfo deberíamos tener la siguiente estructura.

```
<ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
  <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
<ds:Reference Id="identificador-ref0" URI="">
  <ds:Transforms>
 <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
 </ds:Transforms>
 <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
       <ds:DigestValue>akcOQ5qEh4dkMwt0d5BoXRR8Bo4vdy9DBZtfF0SsA=</ds:DigestValue>
</ds:Reference>
<ds:Reference URI="#identificador-keyinfo”
  <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
  <ds:DigestValue>troRYR2fcmJLV6gYibVM6XlArbddSCkjYkACP47/4=</ds:DigestValue>
</ds:Reference>
<ds:Reference Type="http://uri.etsi.org/01903#SignedProperties" URI="# identificador-SignedProperties">
<ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
<ds:DigestValue>hpIsyD/08hVUc1exnfEyhGyKX5s3pUPbpMKmPhkPPqU=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
```

- Incluir la firma digital calculada con RSA Y SHA-256 codificada en base64 del elemento `SignedInfo` Canonicalizado con C14N en el elemento `Signature:SignatureValue`, importante recordar que en el elemento raíz del cual se calcularon los resúmenes y firma se deben incluir los espacios de nombres, pero estos no deben existir para la construcción del documento, esto se hace con el objetivo de que al momento de comprobar la firma no ocurran errores de espacios de nombres, un ejemplo de este procedimiento es el siguiente.

```
<ds:SignedInfo
xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2"
xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:ext="urn:oasis:names:specification:ubl:schema:xsd:CommonExtensionComponents-2"
xmlns:sts="http://www.dian.gov.co/contratos/facturaelectronica/v1/Structures"
xmlns:xades="http://uri.etsi.org/01903/v1.3.2#" xmlns:xades141="http://uri.etsi.org/01903/v1.4.1#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
  <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
<ds:Reference Id="identificador-ref0" URI="">
  <ds:Transforms>
 <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
 </ds:Transforms>
 <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
```

```
        <ds:DigestValue>akcOQ5qEh4dkMwt0d5BoXRR8Bo4vdy9DBZtfF0SsA=</ds:DigestValue>
    </ds:Reference>
    <ds:Reference URI="#identificador-keyinfo”
      <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
      <ds:DigestValue>troRYR2fcmJLV6gYibVM6XlArbddSCkjYkACP47/4=</ds:DigestValue>
    </ds:Reference>
    <ds:Reference Type="http://uri.etsi.org/01903#SignedProperties" URI="# identificador-SignedProperties">
    <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
    <ds:DigestValue>hpIsyD/08hVUc1exnfEyhGyKX5s3pUPbpMKmPhkPPqU=</ds:DigestValue>
    </ds:Reference>
    </ds:SignedInfo>
```

Se incluyen los espacios de nombres en el elemento raíz y se calcula la firma o el resumen y se elimina los nombres de espacios para su inclusión en el documento.

Una vez finalizados todos los pasos anteriores debemos formar la estructura final de la firma, se debe agregar el elemento `SignedProperties` dentro de `Signature:Object:QualifyingProperties` y agregar el atributo `Target="#identificador-signature"` en el elemento `Signature:Object:QualifyingProperties` como se muestra a continuación.

```
<ds:Signature ID="identificador-signature">
 <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
    <ds:Reference Id="identificador-ref0" URI="">
     <ds:Transforms>
      <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
     </ds:Transforms>
     <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
      <ds:DigestValue>akcOQ5qEh4dkMwt0d5BoXRR8Bo4vdy9DBZtfF0SsA=</ds:DigestValue>
    </ds:Reference>
    <ds:Reference URI="#identificador-keyinfo”>
      <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
      <ds:DigestValue>troRYR2fcmJLV6gYibVM6XlArbddSCkjYkACP47/4=</ds:DigestValue>
     </ds:Reference>
     <ds:Reference Type="http://uri.etsi.org/01903#SignedProperties" URI="# identificador-SignedProperties">
     <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
       <ds:DigestValue>hpIsyD/08hVUc1exnfEyhGyKX5s3pUPbpMKmPhkPPqU=</ds:DigestValue>
     </ds:Reference>
    </ds:SignedInfo>

<ds:SignatureValue>q4HWeb47oLdDM4D3YiYDOSXE4YfSHkQKxUfSYiEiPuP2XWvD7ELZTC4ENFv6krg
DAXczmi0W7OMiLIVvuFz0ohPUc4KNlUEzqSBHVi6sC34sCqoxuRzOmMEoCB9Tr4VICxU1Ue9XhgP7o
6X4f8KFAQWW1MKP59aAnFMfYl8lXpaS0kpUwuui3wdCZsGycsl1prEWiwzpaukEUOXyTo7o
RBOuNsDIUhP24Fv1alRFnX6/9zEOpRTs4rEQKN3IQnibF757LE/nnkutElZHTXaSV637gpHjXoUN
5JrUwTNOXvmFS98N6DczCQfeNuDIozYwtFVlMw==</SignatureValue>
            <ds:KeyInfo Id="identificador-keyinfo">
           <ds:X509Data>
         <ds:X509Certificate>MIIIODCCBiCgAwIBAgIIbAsHYmJtoOIwDQYJKoZIhvcNAQELBQAwgbQxIzAhBgkq
```

hkiG9w0BCQEWGluZm9AYW5kZXNzY2QuY29tLmNvMSMwIQYDVQQDExpDQSBBTkRFUyBTQ0QgU
y5BLiBDbGFzZSBJSTEwMC4GA1UECxMnRGl2aXNpb24gZGUgY2VydGlmaWNhY2lvbiBlbnRpZGFkIG
ZpbmFsMRMwEQYDVQQKEwpBbmRlcyBTQ0QuMRQwEgYDVQQHEwtCb2dvdGEgRC5DLjELMAkGA
1UEBhMCQ08wHhcNMTcwOTE2MTM0ODE5WhcNMjAwOTE1MTM0ODE5WjCCARQxHTAbBgNVBAk
TFENhbGxlIEZhbHNhIE5vIDEyIDM0MTgwNgYJKoZIhvcNAQkBFilwZXJzb25hX2p1cmlkaWNhX3BydWV
iYXMxQGFuZGVzc2NkLmNvbS5jbzEsMCoGA1UEAxMjVXN1YXJpbyBkZSBQcnVlYmFzIFBlcnNvbmEg
SnVyaWRpY2ExETAPBgNVBAUTCDExMTExMTExMRkwFwYDVQQMExBQZXJzb25hIEp1cmlkaWNhM
SgwJgYDVQQLEx9DZXJ0aWZpY2FkbyBkZSBQZXJzb25hIEp1cmlkaWNhMQ8wDQYDVQQHEwZCb2d
vdGExFTATBgNVBAgTDEN1bmRpbmFtYXJjYTELMAkGA1UEBhMCQ08wggEiMA0GCSqGSIb3DQEBA
QUAA4IBDwAwggEKAoIBAQC0Dn8toZ2CXun+63zwYecJ7vNmEmS+YouH985xDek7ImeE9lMBHXE1M
5KDo7iT/tUrcFwKj717PeVL52NtB6WU4+KBt+nrK+R+OSTpTno5EvpzfIoS9pLI74hHc017rY0wqjl0lw+8m
7fyLfi/JO7AtX/dthS+MKHIcZ1STPlkcHqmbQO6nhhr/CGl+tKkCMrgfEFIm1kv3bdWqk3qHrnFJ6s2GoVNZ
VCTZW/mOzPCNnnUW12LDd/Kd+MjN6aWbP0D/IJbB42Npqv8+/oIwgCrbt0sS1bysUgdT4im9bBhb00M
WVmNRBBe3pH5knzkBid0T7TZsPCyiMBstiLT3yfpAgMBAAGjggLpMIIC5TAMBgNVHRMBAf8EAjAAMB
8GA1UdIwQYMBaAFKhLtPQLp7Zb1KAohRCdBBMzxKf3MDcGCCsGAQUFBwEBBCswKTAnBggrBgEF
BQcwAYYbaHR0cDovL29jc3AuYW5kZXNzY2QuY29tLmNvMIIB4wYDVR0gBIIB2jCCAdYwggHSBg0rBg
EEAYH0SAECCQIFMIIBvzBBggrBgEFBQcCARY1aHR0cDovL3d3dy5hbmRlc3NjZC5jb20uY28vZG9jcy
9EUENfQW5kZXNTQ0RfVjIuNS5wZGYwggF4BggrBgEFBQcCAjCCAWoeggFmAEwAYQAgAHUAdABp
AGwAaQB6AGEAYwBpAPMAbgAgAGQAZQAgAGUAcwB0AGUAIABjAGUAcgB0AGkAZgBpAGMAYQB
kAG8AIABlAHMAdADhACAAcwB1AGoAZQB0AGEAIABhACAAbABhAHMAIABQAG8AbADtAHQAaQBj
AGEAcwAgAGQAZQAgAEMAZQByAHQAaQBmAGkAYwBhAGQAbwAgAGQAZQAgAFAAZQByAHMAb
wBuAGEAIABKAHUAcgDtAGQAaQBjAGEAIAAoAFAAQwApACAAeQAgAEQAZQBjAGwAYQByAGEAY
wBpAPMAbgAgAGQAZQAgAFAAcgDhAGMAdABpAGMAYQBzACAAZABlACAAQwBlAHIAdABpAGYAa
QBjAGEAYwBpAPMAbgAgACgARABQAEMAKQAgAGUAcwB0AGEAYgBsAGUAYwBpAGQAYQBzACA
AcABvAHIAIABBAG4AZABlAHMAIABTAEMARDAdBgNVHSUEFjAUBggrBgEFBQcDAgYIKwYBBQUHA
wQwRgYDVR0fBD8wPTA7oDmgN4Y1aHR0cDovL3d3dy5hbmRlc3NjZC5jb20uY28vaW5jbHVkZXMvZ2
V0Q2VydC5waHA/Y3JsPTEwHQYDVR0OBBYEFL9BXJHmFVE5c5Ai8B1bVBWqXsj7MA4GA1UdDwEB
/wQEAwIE8DANBgkqhkiG9w0BAQsFAAOCAgEAb/pa7yerHOu1futRt8QTUVcxCAtK9Q00u7p4a5hp2fVz
VrhVQIT7Ey0kcpMbZVPgU9X2mTHGfPdbR0hYJGEKAxiRKsmAwmtSQgWh5smEwFxG0TD1chmeq6y0
GcY0lkNA1DpHRhSK368vZlO1p2a6S13Y1j3tLFLqf5TLHzRgl15cfauVinEHGKU/cMkjLwxNyG1KG/FhCe
CCmawATXWLgQn4PGgvKcNrz+y0cwldDXLGKqriw9dce2Zerc7OCG4/XGjJ2PyZOJK9j1VYIG4pnmoirV
mZbKwWaP4/TzLs6LKaJ4b66xLxH3hUtoXCzYQ5ehYyrLVwCwTmKcm4alrEht3FVWiWXA/2tj4HZiFoG+I
1OHKmgkNv7SwHS7z9tFEFRaD3W3aD7vwHEVsq2jTeYInE0+7r2/xYFZ9biLBrryl+q22zM5W/EJq6EJP
Q6SM/eLqkpzqMEF5OdcJ5kIOxLbrIdOh0+grU2IrmHXr7cWNP6MScSL7KSxhjPJ20F6eqkO1Z/LAxqNslB
IKkYS24VxPbXu0pBXQvu+zAwD4SvQntIG45y/67h884I/tzYOEJi7f6/NFAEuV+lokw/1MoVsEgFESASI9s
N0DfUniabyrZ3nX+LG3UFL1VDtDPWrLTNKtb4wkKwGVwqtAdGFcE+/r/1WG0eQ64xCq0NLutCxg=</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
<ds:Object>
<xades:QualifyingProperties Target="#identificador-signature">
<xades:SignedProperties Id="identificador-SignedProperties">
<xades:SignedSignatureProperties>
<xades:SigningTime>2021-05-03T10:41:41-05:00</xades:SigningTime>
<xades:SigningCertificate>
<xades:Cert>
<xades:CertDigest>
<ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
<ds:DigestValue>lQpmujwABy611YrtuPBma5oi2ugLpIvH7GQuEsb6OWM=</ds:DigestValue>
</xades:CertDigest>
<xades:IssuerSerial>
<ds:X509IssuerName>C=CO, L=Bogota D.C., O=GSE, OU=PKI, CN=Autoridad Subordinada 01 GSE, emailAddress=info@gse.com.co</ds:X509IssuerName>
<ds:X509SerialNumber>460132437665470756863835</ds:X509SerialNumber>

```
      </xades:IssuerSerial>
     </xades:Cert>
</xades:SigningCertificate>
  <xades:SignaturePolicyIdentifier>
   <xades:SignaturePolicyId>
    <xades:SigPolicyId>
     <xades:Identifier>https://facturaelectronica.dian.gov.co/politicadefirma/v2/politicadefirmav2.pdf</xades:Identifier>
     <xades:Description>Política de firma para facturas electrónicas de la República de
Colombia</xades:Description>
    </xades:SigPolicyId>
    <xades:SigPolicyHash>
     <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
     <ds:DigestValue>dMoMvtcG5aIzgYo0tIsSQeVJBDnUnfSOfBpxXrmor0Y=</ds:DigestValue>
    </xades:SigPolicyHash>
  </xades:SignaturePolicyId>
 </xades:SignaturePolicyIdentifier>
<xades:SignerRole>
  <xades:ClaimedRoles>
   <xades:ClaimedRole>supplier</xades:ClaimedRole>
  </xades:ClaimedRoles>
</xades:SignerRole>
      </xades:SignedSignatureProperties>
   </xades:SignedProperties>
 </xades:QualifyingProperties>
</ds:Object>
</ds:Signature>
```

Este elemento Signature debe ser incluido en la factura XML dentro del segundo elemento `UBLExtensions:UBLExtension:ExtensionContent` y con esto tendremos nuestra factura firmada con el formato XAdES-EPES, una vez completado este proceso resulta sumamente importante contar con una herramientas que nos permita validar la estructura del formato. Afortunadamente ETSI suministra una herramienta llamada ETSI Signature Conformance Checker[31] la cual nos permite validar este formato (valida el formato mas no la firma).

3.4.6.2.1 Código Firma digital

El proceso anterior fue realizado con la siguiente clase implementada en el lenguaje de programación PHP

```
<?php
class Firma{
   protected $certDigest;
   protected $publicKey;
   protected $privateKey;
   protected $certSerialNumber;
```

```
    protected $certIssuer;

    protected $id = [
        "SIGNATURE" => "SIGNATURE-",
        "REFERENCE" => "REFERENCE-",
        "KEYINFO"   => "KEYINFO-",
        "PROPERTIES"=> "PROPERTIES-"
    ];
    protected $xmlns = 'xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-
2" xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-
2" xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-
2" xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:ext="urn:oasis:names:specification:ubl:schema:xsd:Comm
onExtensionComponents-2"
xmlns:sts="http://www.dian.gov.co/contratos/facturaelectronica/v1/Structures"
xmlns:xades="http://uri.etsi.org/01903/v1.3.2#" xmlns:xades141="http://uri.etsi.org/01903/v1.4.1#" xmlns:xsi="http://w
ww.w3.org/2001/XMLSchema-instance"';

const EXC_C14N = 'http://www.w3.org/TR/2001/REC-xml-c14n-20010315';
const RSA_SHA256 = 'http://www.w3.org/2001/04/xmldsig-more#rsa-sha256';
const TRANSFORM = 'http://www.w3.org/2000/09/xmldsig#enveloped-signature';
const SHA256 = 'http://www.w3.org/2001/04/xmlenc#sha256';
const TYPEPROPERTIES = 'http://uri.etsi.org/01903#SignedProperties';
const IDENTIFIER='https://facturaelectronica.dian.gov.co/politicadefirma/v2/politicadefirmav2.pdf';
const DESCRIPTION='Política de firma para facturas electrónicas de la República de Colombia';
const XMLDSIG = 'http://www.w3.org/2000/09/xmldsig#';

function __construct($PathCert,$PasswordCert){
  $this->get_certificado($PathCert,$PasswordCert);
  $this->setUUID();
}
public function firmar($xml){
   $dom_xml = new DOMDocument('1.0','UTF-8');
   $dom_xml->loadXML($xml);
   $hash_xml = base64_encode(hash("sha256",$dom_xml->C14N(),true));
   //crear elemento KeyInfo
  $KeyInfo = '<ds:KeyInfo Id="'.$this->id['KEYINFO'].'">'.
            '<ds:X509Data>'.
              '<ds:X509Certificate>'.$this->publicKey.'</ds:X509Certificate>'.
            '</ds:X509Data>'.
          '</ds:KeyInfo>';
```

```
//agregar namespace
$KeyInfo_xmlns = str_replace('<ds:KeyInfo', '<ds:KeyInfo '.$this->xmlns, $KeyInfo);
//hash keyinfo
$hash_KeyInfo = base64_encode(hash("sha256",$KeyInfo_xmlns,true));

//crear elemento signedpropierties
$SignedProperties = '<xades:SignedProperties Id="'.$this->id['PROPERTIES'].'">'.
  '<xades:SignedSignatureProperties>'.
    '<xades:SigningTime>'.(date('Y-m-d\Th:i:s')."-05:00").'</xades:SigningTime>' .
      '<xades:SigningCertificate>'.
        '<xades:Cert>'.
          '<xades:CertDigest>'.
            '<ds:DigestMethod Algorithm="'.self::SHA256.'"></ds:DigestMethod>'.
            '<ds:DigestValue>'.$this->certDigest.'</ds:DigestValue>'.
          '</xades:CertDigest>'.
          '<xades:IssuerSerial>' .
            '<ds:X509IssuerName>'.$this->certIssuer.'</ds:X509IssuerName>'.
            '<ds:X509SerialNumber>'.$this->certSerialNumber.'</ds:X509SerialNumber>' .
          '</xades:IssuerSerial>'.
        '</xades:Cert>'.
      '</xades:SigningCertificate>' .
      '<xades:SignaturePolicyIdentifier>'.
        '<xades:SignaturePolicyId>' .
          '<xades:SigPolicyId>'.
            '<xades:Identifier>'.self::IDENTIFIER.'</xades:Identifier>'.
            '<xades:Description>'.self::DESCRIPTION.'</xades:Description>'.
          '</xades:SigPolicyId>'.
          '<xades:SigPolicyHash>' .
            '<ds:DigestMethod Algorithm="'.self::SHA256.'"></ds:DigestMethod>'.
            '<ds:DigestValue>dMoMvtcG5aIzgYo0tIsSQeVJBDnUnfSOfBpxXrmor0Y=</ds:DigestValue>'.
          '</xades:SigPolicyHash>'.
        '</xades:SignaturePolicyId>' .
      '</xades:SignaturePolicyIdentifier>'.
      '<xades:SignerRole>' .
      '<xades:ClaimedRoles>' .
        '<xades:ClaimedRole>supplier</xades:ClaimedRole>' .
      '</xades:ClaimedRoles>' .
      '</xades:SignerRole>' .
  '</xades:SignedSignatureProperties>'.
  '</xades:SignedProperties>';
```

```
//agregar namespace
$SignedProperties_xmlns = str_replace('<xades:SignedProperties', '<xades:SignedProperties '.$this->xmlns, $SignedProperties);
//hash SignedProperties
$hash_SignedProperties = base64_encode(hash("sha256",$SignedProperties_xmlns,true));

  //elemento SignedInfo
$SignedInfo = '<ds:SignedInfo>'.
‘<ds:CanonicalizationMethod Algorithm="'.self::EXC_C14N.'">'.'</ds:CanonicalizationMethod>'.
        '<ds:SignatureMethod Algorithm="'.self::RSA_SHA256.'">'.'</ds:SignatureMethod>'.
            '<ds:Reference Id="'.$this->id['REFERENCE'].'" URI="">'.
              '<ds:Transforms>'.
                '<ds:Transform Algorithm="'.self::TRANSFORM.'"></ds:Transform>'.
              '</ds:Transforms>'.
              '<ds:DigestMethod Algorithm="'.self::SHA256.'"></ds:DigestMethod>'.
              '<ds:DigestValue>'.$hash_xml.'</ds:DigestValue>'.
            '</ds:Reference>'.
            '<ds:Reference URI="#'.$this->id['KEYINFO'].'">'.
              '<ds:DigestMethod Algorithm="'.self::SHA256.'"></ds:DigestMethod>'.
              '<ds:DigestValue>'.$hash_KeyInfo.'</ds:DigestValue>'.
            '</ds:Reference>'.
      '<ds:Reference Type="'.self::TYPEPROPERTIES.'" URI="#'.$this->id['PROPERTIES'].'">'.
              '<ds:DigestMethod Algorithm="'.self::SHA256.'"></ds:DigestMethod>'.
              '<ds:DigestValue>'.$hash_SignedProperties.'</ds:DigestValue>'.
            '</ds:Reference>'.
          '</ds:SignedInfo>';

  //agregar namespace
  $signedinfo_xmlns = str_replace('<ds:SignedInfo', '<ds:SignedInfo '.$this->xmlns, $SignedInfo);
  //firma
  openssl_sign($signedinfo_xmlns, $signatureResult, $this->privateKey,"SHA256");
  $signatureResult = base64_encode($signatureResult);

  $signatue = '<ds:Signature xmlns:ds="'.self::XMLDSIG.'" Id="'.$this->id['SIGNATURE'].'">'.
            $SignedInfo.
            '<ds:SignatureValue>'.$signatureResult.'</ds:SignatureValue>'.
            $KeyInfo.
            '<ds:Object>'.
```

```
                    '<xades:QualifyingProperties Target="#'.$this->id['SIGNATURE'].'">'.
                        $SignedProperties.
                    '</xades:QualifyingProperties>'.
                '</ds:Object>'.
            '</ds:Signature>';
        return $signatue;
    }
    public function setUUID(){
        foreach ($this->id as $key => $value) {
            $this->id[$key] = mb_strtoupper("{$value}".sha1(uniqid()));
        }
    }
    public function get_certificado($PathCertP12,$PasswordCert){
        $contenido_certificado = file_get_contents($PathCertP12);
        openssl_pkcs12_read($contenido_certificado, $cert, $PasswordCert);
        $this->privateKey = $cert["pkey"];
        $this->publicKey = $cert["cert"];
        $certData = openssl_x509_parse($this->publicKey);
        $this->certDigest = base64_encode(openssl_x509_fingerprint($this->publicKey, "sha256", true));
        $this->certSerialNumber = $certData['serialNumber'];
        $this->certIssuer = $this->getIssuer($certData['issuer']);
        $this->publicKey = str_replace(["\r", "\n", '-----BEGIN CERTIFICATE-----', '-----END CERTIFICATE-----'], '', $this-
>publicKey);

    }
    public function getIssuer($certData){
        $Issuer = array();
        foreach ($certData as $item => $value){
            $Issuer[] = $item.'='.$value;
        }
        $Issuer = implode(', ',array_reverse($Issuer));
        return $Issuer;
    }
}
?>
```

Para instanciar la clase anterior se debe hacer de la siguiente manera

```
$firma = new Firma("ruta del certificado p12","clave del certificado");
```

```
$signature = $firma->firmar("Factura Invoice xml en string");
```

La variable $signature contendrá toda la estructura de la firma la cual debe ser anexada en la factura en el segundo elemento `Invoice:UBLExtensions:UBLExtension:ExtensionContent`

3.5 ENVIAR XML

Él envió de la factura de venta se realiza con el objetivo de validar los documentos ante los servicios web de la DIAN, esto debe ser realizado con UBL 2.1 como lenguaje para el intercambio de información de los documentos electrónicos y el protocolo de c5municación SOAP.

3.5.1 Estándar de comunicación

El medio comunicación es Internet con la utilización del protocolo TLS versión 1.2, con autentificación mutua través de certificados digitales. El modelo de comunicación sigue el estándar de servicios web definido por el WS-Security 1.0 [32] Oasis, con autenticación X.509 Certificate Token Profile 1.1, El intercambio de mensajes entre los Servicios Web de la DIAN y el sistema del contribuyente o del Proveedor Tecnológico será realizado mediante el estándar SOAP versión 1.2, con intercambio de mensajes XML. La llamada de cada uno de los servicios web es realizada con el envío de un mensaje XML a través del campo body[3].

3.5.2 Estándar de mensajes de los servicios de La DIAN

La estructura del mensaje SOAP para la comunicación con el web services de la DIAN es la siguiente.

```
<soap:Envelope xmlns:soap="http:/www.w3.org/2003/05/soap-envelope" xmlns:wcf="http:/wcf.dian.colombia">
   <soap:Header xmlns:wsa="http:/www.w3.org/2005/08/addressing">
        ------ Área de inclusión de la autenticación por medio de Certificado Digital
   </soap:Header>
   <soap:Body>
        ------ Área de Dato: La información en el área de datos es un documento XML que debe atender al formato definido para cada WS
   </soap:Body>
</soap:Envelope>
```

Ilustración 72 estándar de mensaje; tomado de[3]

El elemento para la autenticación debe ser incluida en el elemento header y los XML correspondiente al mensaje debe ser incluido en el elemento body. Esto se realiza con el objetivo de identificar y autenticar al contribuyente por medio del certificado digitales.

3.5.3 Modelo conceptual de comunicación

La DIAN dispone de un sistema de validación previa, por medio de un web services con diferentes métodos, cada servicio se encuentra respaldado por un método. El modelo de comunicación e interoperabilidad siempre iniciará en el sistema del contribuyente (obligado a facturar), por medio del consumo dicho método.

El web service de la DIAN Presenta 2 médelos de validación como se muestra en la siguiente ilustración.

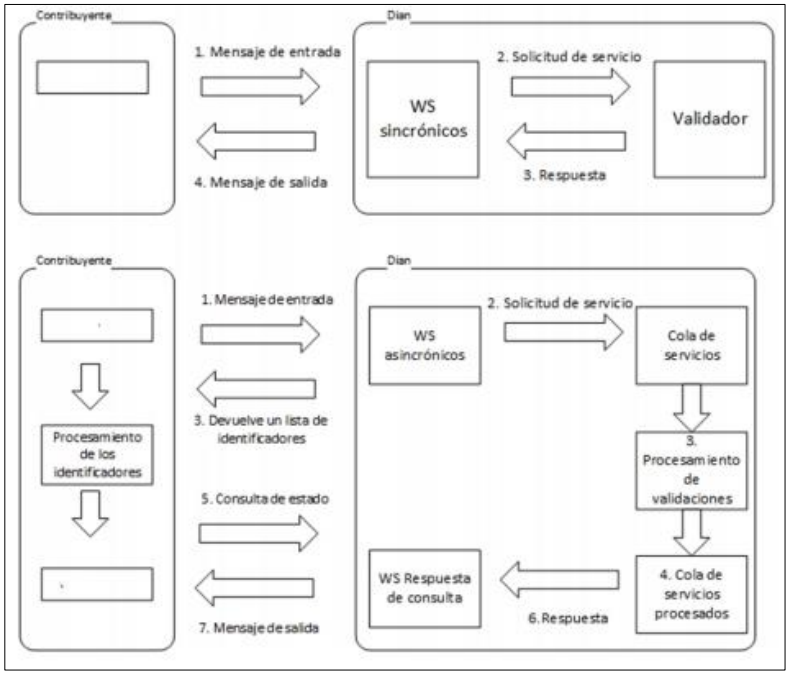


Ilustración 73 modelos de validación web service; tomado de [3]

El modelo síncrono permite obtener el estado de validación en la misma petición en la que fue enviada la factura, es decir se obtendrá el resultado de la validación de manera inmediata. El segundo modelo presentado es el modelo asíncrono este permite enviar

la factura en una petición o Request y esta devolverá un trackId que permitirá consultar el estado de validación por medio de otro método en otra conexión.

3.5.4 Métodos de los servicios web de La DIAN

Cada servicio se encuentra respaldado por un método web específico, tanto métodos síncronos como asíncronos como observo anteriormente donde podemos encontrar los siguientes métodos.

- **GetExchangeEmails:**
  Función: Consultar el correo electrónico suministrada por el adquiriente registrado en el procedimiento de habilitación como facturador electrónico.
  Proceso**:** síncrono.
- **GetNumberingRange:**
  Función: Consulta de Rangos de Numeración registrado en DIAN entregando la información relacionada con estos rangos.
  Proceso**:** Sincrónico
- **GetStatus:**
  Función: Recibir una consulta para obtener el estado del documento en el proceso de validación y devolver la respuesta del estado del documento.
  Proceso: Sincrónico
- **GetStatusZip:**
  Función: Recibir una consulta para obtener el estado de todos los documentos asociados a un ZIP en el proceso de validación y devolver la respuesta del estado de cada uno de los documentos.
  Proceso: Sincrónico
- **GetXmlByDocumentKey:**
  Función: Descarga de XML.
  Proceso: Sincrónico
- **SendBillAsync**:
  Función: Recibir un ZIP con varios documentos UBL, el servicio puede recibir un ZIP con uno o más (Máximo 50) documentos electrónicos e instrumentos electrónicos firmados digitalmente.
  Proceso: Asincrónico
- **SendBillSync:**
  Función: Recibir un ZIP con un UBL, Este servicio atiende la funcionalidad de enviar a la DIAN los documentos, de forma tal que la plataforma DIAN reciba y

valide los documentos UBL (factura electrónica, nota de crédito y nota de débito) de forma síncrona y de respuesta de la validación para su uso y expedición.
Proceso: Sincrónico

- **SendTestSetAsync:**
  Función: Recibir un ZIP con UBL para pruebas de Habilitación.
  Proceso: Asincrónico.
- **SendEventUpdateStatus**:
  Función: Este servicio atiende la funcionalidad de recepción y registro de los eventos de los documentos tributarios, ante la DIAN.
  Proceso: Sincrónico

3.5.4.1 SendTestSetAsync

Este método atiende la funcionalidad de enviar a la DIAN los documentos, de tal forma que la plataforma DIAN reciba y valide los documentos de factura electrónica, nota de crédito y nota de débito para efectos de obtener un TrackId que permite consumir el método GetStatusZIP con el cual se obtendrá el resultado de la validación de estos documentos en pruebas de habilitación. Este método se utiliza únicamente en el ambiente de producción en habilitación, el cual debe ser utilizado para superar los requisitos (cantidad mininas de documentos correctos) para habilitarse como facturador electrónico.

3.5.4.1.1 Petición SendTestSetAsync

La estructura que se debe emplear para consumir este método es la siguiente.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:wcf="http://wcf.dian.colombia">
  <soap:Header/>
  <soap:Body>
    <wcf:SendTestSetAsync>
      <!--Optional:-->
      <wcf:fileName>invoice-2.zip</wcf:fileName>
      <!--Optional:-->
      <wcf:contentFile>cid:3571097601175</wcf:contentFile>
      <!--Optional:-->
      <wcf:testSetId>4de36cb4-9973-4ea4-a156-34e909aa24dc</wcf:testSetId>
    </wcf:SendTestSetAsync>
  </soap:Body>
</soap:Envelope>
```

Ilustración 74 estructura del mensaje SOAP SendTestSetAsync; tomado de [3]

Donde

- **fileName:** Corresponde al nombre del ZIP.
- **contentFile:** Corresponde al arreglo de byte codificados en base64 que representa al ZIP adjunto y que contiene los UBL a validar.
- **testSetId:** Corresponde al valor TestID que se genera al configurar un modo de operación para el proceso de habitación.

3.5.4.1.2 Respuesta de petición SendTestSetAsync

El mensaje de respuesta a este método es la siguiente

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://www.w3.org/2005/08/addressing" xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://wcf.dian.colombia/IWcfDianCustomerServices/SendTestSetAsyncResponse</a:Action>
    <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <u:Timestamp u:Id="_0">
        <u:Created>2020-02-03T22:44:33.925Z</u:Created>
        <u:Expires>2020-02-03T22:49:33.925Z</u:Expires>
      </u:Timestamp>
    </o:Security>
  </s:Header>
  <s:Body>
    <SendTestSetAsyncResponse xmlns="http://wcf.dian.colombia">
      <SendTestSetAsyncResult xmlns:b="http://schemas.datacontract.org/2004/07/UploadDocumentResponse" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <b:ErrorMessageList i:nil="true" xmlns:c="http://schemas.datacontract.org/2004/07/XmlParamsResponseTrackId"/>
        <b:ZipKey>358f9538-1f80-4ed5-a3f6-aaa1ef36bebd</b:ZipKey>
      </SendTestSetAsyncResult>
    </SendTestSetAsyncResponse>
  </s:Body>
</s:Envelope>
```

Ilustración 75 estructura de respuesta SendTestSetAsync}, tomado de [3]

Donde

- **zipKey:** Corresponde al número generado una vez los documentos pasan a la cola de validación. Este TrackId o ZipKey, es el que se ocupara en el método GetStatusZip para obtener el resultado de la validación.

3.5.4.2 SendBillSync

Este método permite enviar a la DIAN los documentos, de tal forma que la plataforma DIAN valide los documentos de factura electrónica, nota de crédito y nota de débito de forma síncrona y de respuesta de la validación para su uso y expedición en la misma conexión. El servicio puede recibir un ZIP con un solo documento electrónico firmado digitalmente, en formato UBL y construido según el esquema detallado en la resolución 0000042.[3]

3.5.4.2.1 Petición SendBillSync

La estructura que se debe emplear para consumir este método es la siguiente

```
<soap:Body>
  <wcf:SendBillAsync>
    <wcf:fileName>Test</wcf:fileName>
    <wcf:contentFile>cid:179956799470</wcf:contentFile>
  </wcf:SendBillAsync>
</soap:Body>
            </soap:Envelope>
```

Ilustración 76 estructura SendBillSync; tomado de [3]

Donde

- **fileName:** Corresponde al nombre del ZIP.
- **contentFile:** Corresponde al arreglo de byte codificados en base64 que representa al ZIP adjunto y que contiene el UBL a validar.

3.5.4.2.2 Respuesta de petición SendBillSync

La estructura del mensaje de respuesta es la siguiente.

```
<s:Envelope xmlns:s="http:/www.w3.org/2003/05/soap-envelope" xmlns:a="http:/www.w3.org/2005/08/addressing" xmlns:u="http:/docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <s:Header>
    <a:Action s:mustUnderstand="1">http:/wcf.dian.colombia/IWcfDianCustomerServices/SendBillSyncResponse</a:Action>
    <o:Security s:mustUnderstand="1" xmlns:o="http:/docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <u:Timestamp u:Id="_0">
        <u:Created>2019-02-19T21:33:41.785Z</u:Created>
        <u:Expires>2019-02-19T21:38:41.785Z</u:Expires>
      </u:Timestamp>
    </o:Security>
  </s:Header>
  <s:Body>
    <SendBillSyncResponse xmlns="http:/wcf.dian.colombia">
      <SendBillSyncResult xmlns:i="http:/www.w3.org/2001/XMLSchema-instance">
        <b:ErrorMessage xmlns:c="http:/schemas.microsoft.com/2003/10/Serialization/Arrays">
          <c:string>Regla: AC38b  Documento fue enviado para el ambiente errado (producción o pruebas)</c:string>
          <c:string>Regla: ZB01 Fallo en el Schema XML del archivo - The XmlSchemaSet on the document is either null or has no schemas in
t. Provide schema information before calling Validate. -</c:string>
          <c:string>Regla: AA08d Número de factura debe estar contenido en el rango de numeración otorgado</c:string>
          <c:string>Regla: AA09 Valor del CUFE no está calculado correctamente.</c:string>
        </b:ErrorMessage>
        <b:IsValid>false</b:IsValid>
        <b:StatusCode>99</b:StatusCode>
        <b:StatusDescription>Validación contiene errores en campos mandatorios.</b:StatusDescription>
        <b:StatusMessage i:nil="true"/>
        <b:XmlBase64Bytes>xmlbase64</b:XmlBase64Bytes>
        <b:XmlBytes i:nil="true"/>
        <b:xmlDocumentKey>A08f2283e5dd6c1878e6ea9ec3a695a9431c924e1086607f6ae7123d081af7b88</b:xmlDocumentKey>
        <b:xmlFileName>invoice-1-firmado-SHA256</b:xmlFileName>
      </SendBillSyncResult>
    </SendBillSyncResponse>
  </s:Body>
</s:Envelope>
```

Ilustración 77 estructura de respuesta SendBillSync; tomado de [3]

Donde

- **ErrorMessage:** Entrega una descripción con cada una de las validaciones fallidas o con obsevaciones
- **IsValid:** Si es válida informa true y sino es válida informa false
- **StatusCode:** La codificación del estado de procesamiento es la siguiente
  00 = indica que la factura fue procesada correctamente
  66 = NSU no encontrado
  90 = TrackId no encontrado
  99 = las validaciones de la factura contienen errores en campos mandatorios
- **StatusMessage:** Entrega una descripción del error de cada una de las validaciones iniciales. Si no hay errores no entrega descripción
- **XmlBase64Bytes:** Entrega el UBL correspondiente a la petición en forma estructurada en base64.
- **xmlDocumentKey:** Este elemento corresponde al TrackId o CUFE del documento procesado
- **xmlFileName:** Este elemento corresponde al nombre del archivo UBL procesado.

3.5.4.3 GetStatusZip

Este método atiende la funcionalidad de consultar el estado de todos los documentos enviados en un ZIP, por los métodos SendBillAsync o SendTestSetAsync. Este método estará disponible en los ambientes de producción en habilitación y producción en operación.

3.5.4.3.1 Petición GetStatusZip

La estructura que se debe emplear para consumir este método es la siguiente

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:wcf="http://wcf.dian.colombia">
  <soap:Header/>
  <soap:Body>
    <wcf:GetStatusZip>
      <!--Optional:-->
      <wcf:trackId>358f9538-1f80-4ed5-a3f6-aaa1ef36bebd</wcf:trackId>
    </wcf:GetStatusZip>
  </soap:Body>
</soap:Envelope>
```

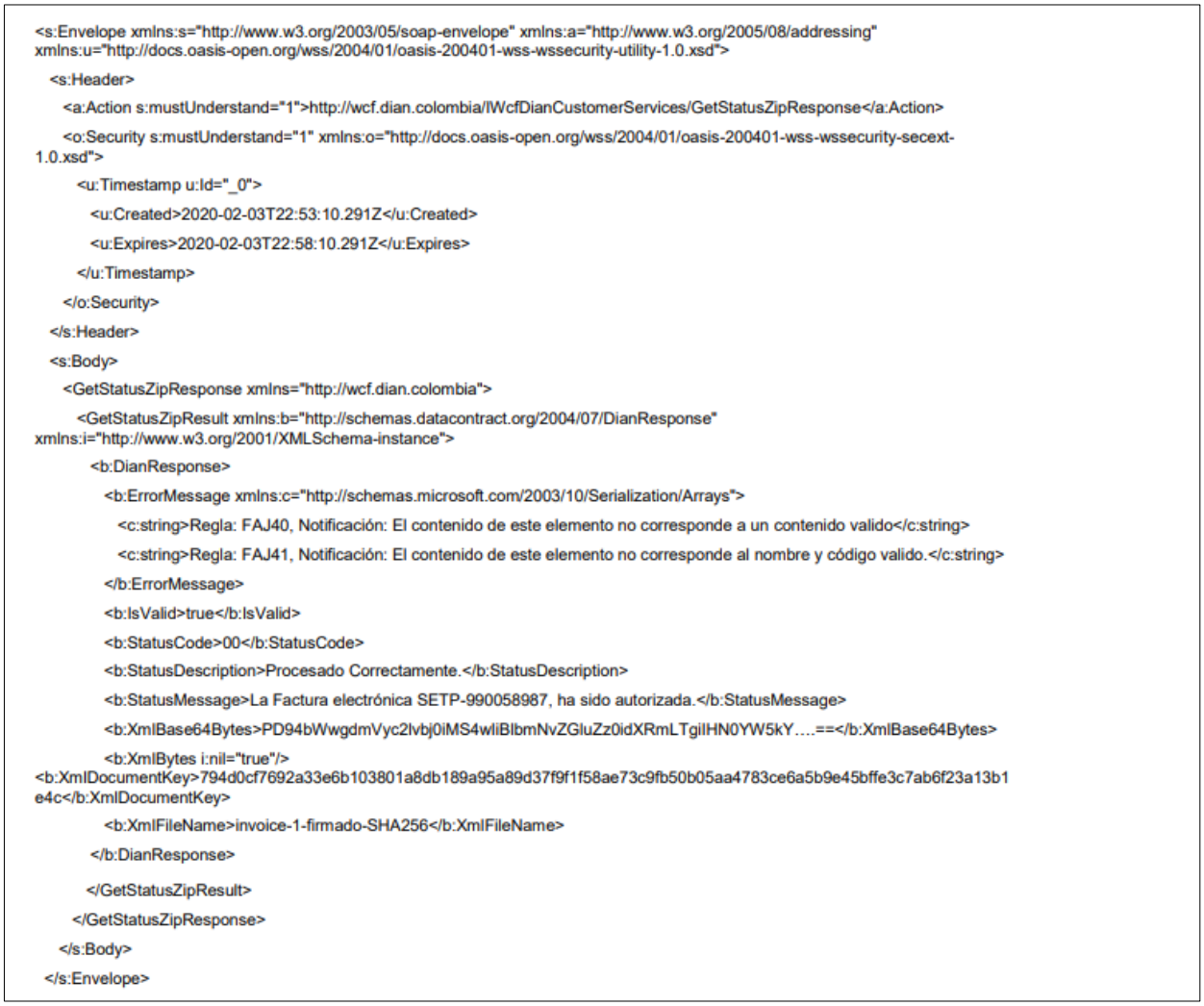
Ilustración 78 estructura mensaje GetStatusZip; tomado de [3]

Donde

- **trackId:** Corresponde al valor del trackId de respuesta que entrega los métodos SendBillAsync y SendTestSetAsync

3.5.4.3.2 Respuesta de petición GetStatusZip

La estructura del mensaje de respuesta es la siguiente



```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://www.w3.org/2005/08/addressing"
xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://wcf.dian.colombia/IWcfDianCustomerServices/GetStatusZipResponse</a:Action>
    <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd">
      <u:Timestamp u:Id="_0">
        <u:Created>2020-02-03T22:53:10.291Z</u:Created>
        <u:Expires>2020-02-03T22:58:10.291Z</u:Expires>
      </u:Timestamp>
    </o:Security>
  </s:Header>
  <s:Body>
    <GetStatusZipResponse xmlns="http://wcf.dian.colombia">
      <GetStatusZipResult xmlns:b="http://schemas.datacontract.org/2004/07/DianResponse"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <b:DianResponse>
          <b:ErrorMessage xmlns:c="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
            <c:string>Regla: FAJ40, Notificación: El contenido de este elemento no corresponde a un contenido valido</c:string>
            <c:string>Regla: FAJ41, Notificación: El contenido de este elemento no corresponde al nombre y código valido.</c:string>
          </b:ErrorMessage>
          <b:IsValid>true</b:IsValid>
          <b:StatusCode>00</b:StatusCode>
          <b:StatusDescription>Procesado Correctamente.</b:StatusDescription>
          <b:StatusMessage>La Factura electrónica SETP-990058987, ha sido autorizada.</b:StatusMessage>
          <b:XmlBase64Bytes>PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmLTgiIHN0YW5kY....==</b:XmlBase64Bytes>
          <b:XmlBytes i:nil="true"/>
<b:XmlDocumentKey>794d0cf7692a33e6b103801a8db189a95a89d37f9f1f58ae73c9fb50b05aa4783ce6a5b9e45bffe3c7ab6f23a13b1
e4c</b:XmlDocumentKey>
          <b:XmlFileName>invoice-1-firmado-SHA256</b:XmlFileName>
        </b:DianResponse>
      </GetStatusZipResult>
    </GetStatusZipResponse>
  </s:Body>
</s:Envelope>
```

Ilustración 79 estructura de respuesta GetStatusZip; tomado de [3]

Donde

- **ErrorMessage:** Entrega una descripción con cada una de las validaciones fallidas o con obsevaciones.
- **IsValid:** Si es válida informa true y sino es válida informa false.
- **StatusCode:** La codificación del estado de procesamiento es la siguiente.
  00 = indica si la factura fue procesada correctamente
  66 = NSU no encontrado

90 = TrackId no encontrado
99 = validaciones contienen errores en campos mandatorios

- **StatusMessage:** Entrega una descripción del error de cada una de las validaciones iniciales. Si no hay errores no entrega descripción.
- **XmlBase64Bytes:** Entrega el UBL correspondiente a la petición en forma estructurada en base64.
- **xmlDocumentKey:** Este elemento corresponde al TrackId o CUFE del documento procesado.
- **xmlFileName:** Este elemento corresponde al nombre del archivo UBL procesado.

3.5.5 Estructura del sobre SOAP

El proceso de envió de la factura electrónica de venta al web services de la DIAN requieren autenticación por medio de certificado por lo que es necesario aplicar una firma digital en el sobre SOAP, la estructura del mensaje SOAP con ws-security[33] y ws-addressing [33] se muestra a continuación.

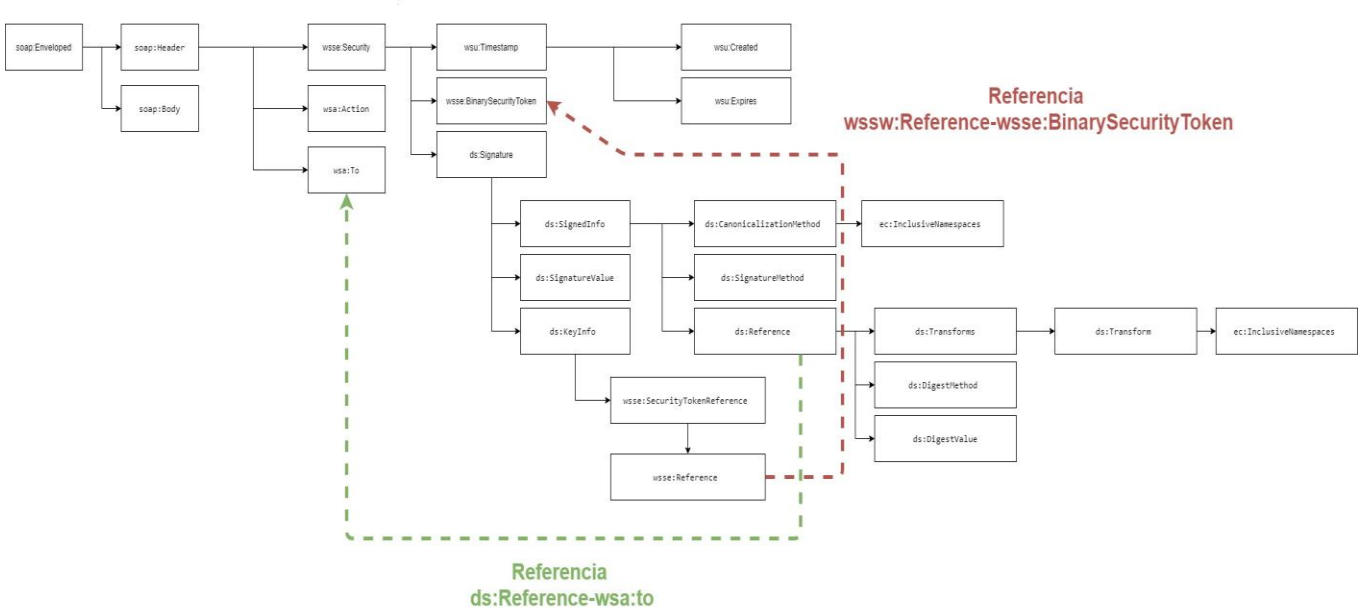


Ilustración 80 estructura SOAP ws-security

Donde

- **soap:Enveloped:** Es el sobre del mensaje soap.
- **soap:header:** Este contendrá el esquema de autenticación.
- **soap:body:** Contendrá el payload del mensaje.
- **ws:security:** Esta es la estructura correspondiente a ws-security el cual contendrá el certificado y la firma.
- **ws:action:** Este contendrá el nombre del método al que se dirige la petición.
- **ws:to:** Corresponde a la URL del web service.
- **ws:Timestamp:** Contiene el tiempo de vigencia del token de seguridad.
- **wsse:BinarySecureToken:** Contendrá la clave pública o también llamado certificado público.
- **SignedInfo:** Este elemento cumple la misma función del firmado de la factura con XAdES como se vio anteriormente, este contendrá los algoritmos utilizados y resumen del elemento a firmar.
- **SignatureValue:** Contendrá la firma digital.
- **KeyInfo:** Este elemento indicara la ubicación del certificado digital.

3.5.6 Construcción del sobre SOAP

Para la construcción del sobre SOAP se debe seguir el esquema de la Ilustración 80 estructura SOAP ws-security y los siguientes pasos

- Crear la estructura generar del sobre SOAP con sus espacios de nombres

```
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:wcf="http://wcf.dian.colombia">
        <soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
        </soap:Header>
        <soap:Body>

        </soap:Body>
</soap:Envelope>
```

- Incluir el elemento `wsse:Security` dentro `soap:header` y los siguientes espacios de nombres

```
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
```

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:wcf="http://wcf.dian.colombia">
<soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
<wsse:Security
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

</wsse:Security>

</soap:Header>
<soap:Body></soap:Body>
</soap:Envelope>
```

- Incluir el elemento `Timestamp` dentro de `enveloped:header:security`, este elemento contendrá el tiempo de expiración del token. created es la fecha y hora en que se creó el sobre y expires indica cuando espirara el token.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:wcf="http://wcf.dian.colombia">
<soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
<wsse:Security
        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

        <wsu:Timestamp>
                <wsu:Created>2021-05-26T21:39:12Z</wsu:Created>
                <wsu:Expires>2021-05-27T14:19:12Z</wsu:Expires>
        </wsu:Timestamp>

</wsse:Security>
</soap:Header>
<soap:Body></soap:Body>
</soap:Envelope>
```

- Incluir el elemento `BinarySecurityToken` dentro de `enveloped:header:security` este contendrá el certificado o clave pública, debemos especificar el tipo de codificación que usa el certificado con el atributo `EncodingType` y tipo de certificado con el atributo `ValueType` y un identificador único (UUID) por medio del atributo Id el cual nos permitirá referirnos a este elemento desde otro.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:wcf="http://wcf.dian.colombia">
<soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
<wsse:Security
        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
```

```
<wsu:Timestamp>
        <wsu:Created>2021-05-26T21:39:12Z</wsu:Created>
        <wsu:Expires>2021-05-27T14:19:12Z</wsu:Expires>
</wsu:Timestamp>

<wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
wsu:Id="BIN-F83C505F28692390FDD8B7F5D404E03C31B2A806"></wsse:BinarySecurityToken>

</wsse:Security>
</soap:Header>
<soap:Body></soap:Body>
</soap:Envelope>
```

- Incluir el elemento `Signature` dentro de `enveloped:header:security` este elemento contendrá información relacionado a la firma y algoritmos utilizados y debe contener el espacio de nombre. `xmlns:ds="http://www.w3.org/2000/09/xmldsig#"`

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:wcf="http://wcf.dian.colombia">
<soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
<wsse:Security
        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <wsu:Timestamp>
                <wsu:Created>2021-05-26T21:39:12Z</wsu:Created>
                <wsu:Expires>2021-05-27T14:19:12Z</wsu:Expires>
        </wsu:Timestamp>

<wsse:BinarySecurityToken
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
wsu:Id="BIN-F83C505F28692390FDD8B7F5D404E03C31B2A806">
</wsse:BinarySecurityToken>

        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

        </ds:Signature>

</wsse:Security>

</soap:Header>

<soap:Body></soap:Body>
</soap:Envelope>
```

- Incluir el elemento `wsa:Action` dentro de `enveloped:header` este elemento contendrá el método al método al cual va dirigida la solicitud.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-
envelope"xmlns:wcf="http://wcf.dian.colombia">
<soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
   <wsse:Security
                xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd"
                xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
                <wsu:Timestamp>
                        <wsu:Created>2021-05-26T21:39:12Z</wsu:Created>
                        <wsu:Expires>2021-05-27T14:19:12Z</wsu:Expires>
                </wsu:Timestamp>

                <wsse:BinarySecurityToken
                EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis 200401-wss-soap-message-
security-1.0#Base64Binary"
                ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3"
                wsu:Id="BIN-F83C505F28692390FDD8B7F5D404E03C31B2A806">
</wsse:BinarySecurityToken>

   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"></ds:Signature>

   </wsse:Security>
<wsa:Action>http://wcf.dian.colombia/IWcfDianCustomerServices/GetStatusZip</wsa:Action>
</soap:Header>

   <soap:Body></soap:Body>
</soap:Envelope>
```

- Incluir el elemento `wsa:To` dentro de `enveloped:header` este elemento contendrá el método la ruta del servico web al que se dirige la solicitud, y los siguientes espacios de nombres

  `xmlns:soap="http://www.w3.org/2003/05/soap-envelope"`
  `xmlns:wcf="http://wcf.dian.colombia"`
  `xmlns:wsa="http://www.w3.org/2005/08/addressing"`
  `xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"`
  y un atributo único (UUID)

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-
envelope"xmlns:wcf="http://wcf.dian.colombia">
<soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
   <wsse:Security
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
```

```
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
            <wsu:Timestamp>
                        <wsu:Created>2021-05-26T21:39:12Z</wsu:Created>
                        <wsu:Expires>2021-05-27T14:19:12Z</wsu:Expires>
            </wsu:Timestamp>

            <wsse:BinarySecurityToken
                        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis 200401-wss-soap-
message-security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
wsu:Id="BIN-F83C505F28692390FDD8B7F5D404E03C31B2A806">
</wsse:BinarySecurityToken>

   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"></ds:Signature>

   </wsse:Security>
<wsa:Action>http://wcf.dian.colombia/IWcfDianCustomerServices/GetStatusZip</wsa:Action>
<wsa:To xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:wcf="http://wcf.dian.colombia"
xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="ID-
D171739FFC14FF17739B7FA149258C6B3F7FFBC3" >https://vpfe-
hab.dian.gov.co/WcfDianCustomerServices.svc</wsa:To>
</soap:Header>

   <soap:Body></soap:Body>
</soap:Envelope>
```

- Incluir los elementos `SignedInfo, SignatureValue y KeyInfo` dentro del elemento `enveloped:header:signature`
- Incluir el elemento CanonicalizationMethod dentro de `enveloped:header:signature:signedinfo` y el atributo `Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"` el cual indica el algoritmo de canonicalización
- Incluir el elemento InclusiveNamespaces dentro del elemento creado en el paso anterior junto con los siguientes atributos `xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="wsa soap wcf"`
- Incluir el elemento SignatureMethod dentro de `enveloped:header:signature:signedinfo` con el atributo `Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"` el cual indica el algoritmo utilizado para la firma.
- Incluir el elemento Reference dentro de `enveloped:header:signature:signedinfo` este elemento cumple la misma función que en xades, la cual es contener el valor del cálculo de hash y los

métodos implicados, este elemento debe contener un atributo URI que haga referencia al elemento `enveloped:header:To` el cual se le debe aplicar el cálculo de resumen SHA-256 y ser anexado en el elemento `enveloped:header:signature:signedinfo:Reference:DigesValue`

- Incluir el elemento `SignatureValue` dentro de `enveloped:header:signature:signedinfo` este elemento contendrá el valor de la firma con RSA Y SHA-256 codificado en base 64 y canonizado con C14N
- Incluir el elemento `SecurityTokenReference` dentro de `enveloped:header:signature:signedinfo:Keyinfo`
- Incluir el elemento Reference dentro de `enveloped:header:signature:signedinfo:Keyinfo:SecurityTokenReference`, este elemento debe hacer referencia al elemento `BinarySecurityToken` y contener el elemento `ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"`

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:wcf="http://wcf.dian.colombia">
 <soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
   <wsu:Timestamp>
    <wsu:Created>2021-05-26T21:39:12Z</wsu:Created>
    <wsu:Expires>2021-05-27T14:19:12Z</wsu:Expires>
   </wsu:Timestamp>

  <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" wsu:Id="BIN-F83C505F28692390FDD8B7F5D404E03C31B2A806"></wsse:BinarySecurityToken>

   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
     <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
   <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="wsa soap wcf"/>
     </ds:CanonicalizationMethod>
     <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
     <ds:Reference URI="#ID-D171739FFC14FF17739B7FA149258C6B3F7FFBC3">
   <ds:Transforms>
    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
```

```
        <ec:InclusiveNamespaces PrefixList="soap wcf" xmlns:ec="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
      </ds:Transform>
     </ds:Transforms>
     <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
     <ds:DigestValue></ds:DigestValue>
      </ds:Reference>
     </ds:SignedInfo>
     <ds:SignatureValue></ds:SignatureValue>

    <ds:KeyInfo>
     <wsse:SecurityTokenReference>
      <wsse:Reference URI="#BIN-F83C505F28692390FDD8B7F5D404E03C31B2A806"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
     </wsse:SecurityTokenReference>
    </ds:KeyInfo>
   </ds:Signature>
    </wsse:Security>
    <wsa:Action>http://wcf.dian.colombia/IWcfDianCustomerServices/GetStatusZip</wsa:Action>
    <wsa:To xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:wcf="http://wcf.dian.colombia"
xmlns:wsa="http://www.w3.org/2005/08/addressing"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="ID-D171739FFC14FF17739B7FA149258C6B3F7FFBC3" >https://vpfe-
hab.dian.gov.co/WcfDianCustomerServices.svc</wsa:To>


</soap:Header>
   <soap:Body></soap:Body>
</soap:Envelope>
```

3.5.7 Código Firma digital SOAP

El proceso anterior fue realizado con la siguiente clase implementada en el lenguaje de programación PHP.

```php
<?php
class EnvelopeSoap{
    protected $signedinfo;
    protected $privateKey;
    protected $XML_SOAP;
    protected $soap;
    protected $to;
    protected $id = [
        'BinarySecurityToken' => 'BIN',
        'ID' => 'ID',
    ];

    const ADDRESSING = 'http://www.w3.org/2005/08/addressing';
```

```
    const SOAP_ENVELOPE = 'http://www.w3.org/2003/05/soap-envelope';
    const DIAN = 'http://wcf.dian.colombia';
    const XMLDSIG = 'http://www.w3.org/2000/09/xmldsig#';
    const WS_WSSECURITY = 'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd';
    const WS_WSSECURITY_UTILITY = 'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd';
    const EXC_C14N = 'http://www.w3.org/2001/10/xml-exc-c14n#';
    const RSA_SHA256 = 'http://www.w3.org/2001/04/xmldsig-more#rsa-sha256';
    const SHA256 = 'http://www.w3.org/2001/04/xmlenc#sha256';
    const X509V3 = 'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3';
    const BASE64BINARY = 'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary';

public function __construct($pathCertificate = null, $password = null, $xmlString = null,$wsaAction_method = null){
        $certificado = $this->get_certificado($pathCertificate,$password);
        $this->setUUID();
        $this->soap = $this->enveloped_soap($xmlString,$wsaAction_method,$certificado);
    }

    private function setUUID()
    {
        foreach ($this->id as $key => $value) {
            $this->id[$key] = mb_strtoupper("{$value}-".sha1(uniqid()));
        }
    }

    private function get_certificado($pathCertificate,$password)
    {
        $certs = null;
        openssl_pkcs12_read(file_get_contents($pathCertificate), $certs, $password);
        openssl_x509_export($certs['cert'], $stringCert);
        $stringCert = str_replace(["\r", "\n", '-----BEGIN CERTIFICATE-----', '-----END CERTIFICATE-----'], '', $stringCert);
        $this->privateKey = $certs["pkey"];

        return $stringCert;
    }

    public function Digest_to(){
```

```
        $this->to = '<wsa:To xmlns:soap="'.self::SOAP_ENVELOPE.'" xmlns:wcf="'.self::DIAN.'" xmlns:wsa="'.self::ADDRESSING.'" xmlns:wsu="'.self::WS_WSSECURITY_UTILITY.'" wsu:Id="'.$this->id['ID'].'" >https://vpfe-hab.dian.gov.co/WcfDianCustomerServices.svc</wsa:To>';
        $dom = new DOMDocument();
        $dom->loadXML($this->to);
        return base64_encode(hash('sha256',$dom->C14N(),true));
    }

    public function signature(){
        $hash = $this->Digest_to();

        $signedinfo_SIG =  '<ds:SignedInfo xmlns:ds="'.self::XMLDSIG.'" xmlns:wsa="'.self::ADDRESSING.'" xmlns:soap="'.self::SOAP_ENVELOPE.'" xmlns:wcf="'.self::DIAN.'">'.
                        '<ds:CanonicalizationMethod Algorithm="'.self::EXC_C14N.'">'.
                            '<ec:InclusiveNamespaces xmlns:ec="'.self::EXC_C14N.'" PrefixList="wsa soap wcf"/>'.
                        '</ds:CanonicalizationMethod>'.
                        '<ds:SignatureMethod Algorithm="'.self::RSA_SHA256.'"/>'.
                        '<ds:Reference URI="#'.$this->id['ID'].'">'.
                            '<ds:Transforms>'.
                                '<ds:Transform Algorithm="'.self::EXC_C14N.'">'.
                                    '<ec:InclusiveNamespaces PrefixList="soap wcf" xmlns:ec="'.self::EXC_C14N.'"/>'.
                                '</ds:Transform>'.
                            '</ds:Transforms>'.
                            '<ds:DigestMethod Algorithm="'.self::SHA256.'"/>'.
                            '<ds:DigestValue>'.$hash.'</ds:DigestValue>'.
                        '</ds:Reference>'.
                    '</ds:SignedInfo>';

        $DOM = new DOMDocument();
        $DOM->loadXML($signedinfo_SIG);

        openssl_sign($DOM->C14N(), $signatureResult, $this->privateKey, "SHA256");
        $signatureValue = base64_encode($signatureResult);

        $this->signedinfo = str_replace('<ds:SignedInfo xmlns:ds="'.self::XMLDSIG.'" xmlns:wsa="'.self::ADDRESSING.'" xmlns:soap="'.self::SOAP_ENVELOPE.'" xmlns:wcf="'.self::DIAN.'">',"<ds:SignedInfo>",$signedinfo_SIG);

        return $signatureValue;
```

```
    }


    public function enveloped_soap($xmlString,$method,$certificado){

        $Time = time();
        $firma = $this->signature();
$this->XML_SOAP = '<soap:Envelope xmlns:soap="'.self::SOAP_ENVELOPE.'" xmlns:wcf="'.self::DIAN
.'">'.
          '<soap:Header xmlns:wsa="'.self::ADDRESSING.'">'.
'<wsse:Security xmlns:wsse="'.self::WS_WSSECURITY.'" xmlns:wsu="'.self::WS_WSSECURITY_UTILITY.'">'.
                '<wsu:Timestamp>'.
                '<wsu:Created>'.gmdate("Y-m-d\TH:i:s\Z", $Time).'</wsu:Created>'.
                '<wsu:Expires>'.gmdate("Y-m-d\TH:i:s\Z", $Time +  60000).'</wsu:Expires>'.
                '</wsu:Timestamp>'.
'<wsse:BinarySecurityToken EncodingType="'.self::BASE64BINARY.'" ValueType="'.self::X509V3.'" wsu:Id="'.$this-
>id['BinarySecurityToken'].'">'.$certificado.'</wsse:BinarySecurityToken>'.
               '<ds:Signature xmlns:ds="'.self::XMLDSIG.'">'.
                   $this->signedinfo.
                  <ds:SignatureValue>'.$firma.'</ds:SignatureValue>'
                      '<ds:KeyInfo>'.
                    '<wsse:SecurityTokenReference>'.
'<wsse:Reference URI="#'.$this->id['BinarySecurityToken'].'" ValueType="'.self::X509V3.'"/>'.
                  '</wsse:SecurityTokenReference>'.
                              '</ds:KeyInfo>'.
                            '</ds:Signature>'.
                          '</wsse:Security>'.

    '<wsa:Action>http://wcf.dian.colombia/IWcfDianCustomerServices/'.$method.'</wsa:Action>'.
                          $this->to.
                          '</soap:Header>'.
                             $xmlString.
                          '</soap:Envelope>'
                        ;
        return $this->XML_SOAP;
    }
    public function getSoap(){
        return $this->soap;
    }
}
```

```
?>
```

Para instanciar la clase anterior se debe realizar de la siguiente manera

```
$enveloped_soap = new EnvelopeSoap("ruta del certificado p12","clave del certificado","xml del ele
mento body","tipo de metodo del web service a utilizar");

$soap = $enveloped_soap->get_soap();
```

La variable $soap contendrá el formato del sobre SOAP para su envió al web service.

3.6 DISTRIBUCIÓN DE LA FACTURA ELECTRÓNICA

La distribución de factura electrónica consta de 3 pasos, él envió de la factura al web service de la DIAN, la consulta del estado de validación y la expedición de las facturas al adquiriente, el medio de comunicación con la DIAN es por medio del protocolo HTTP directa mente al web service con la URL especifica en la información técnica del facturador. El medio de comunicación con el adquirente es el correo electrónico por lo que se debe utilizar el protocolo SMTP para él envió de las facturas por medio de mensajes de correo electrónico.

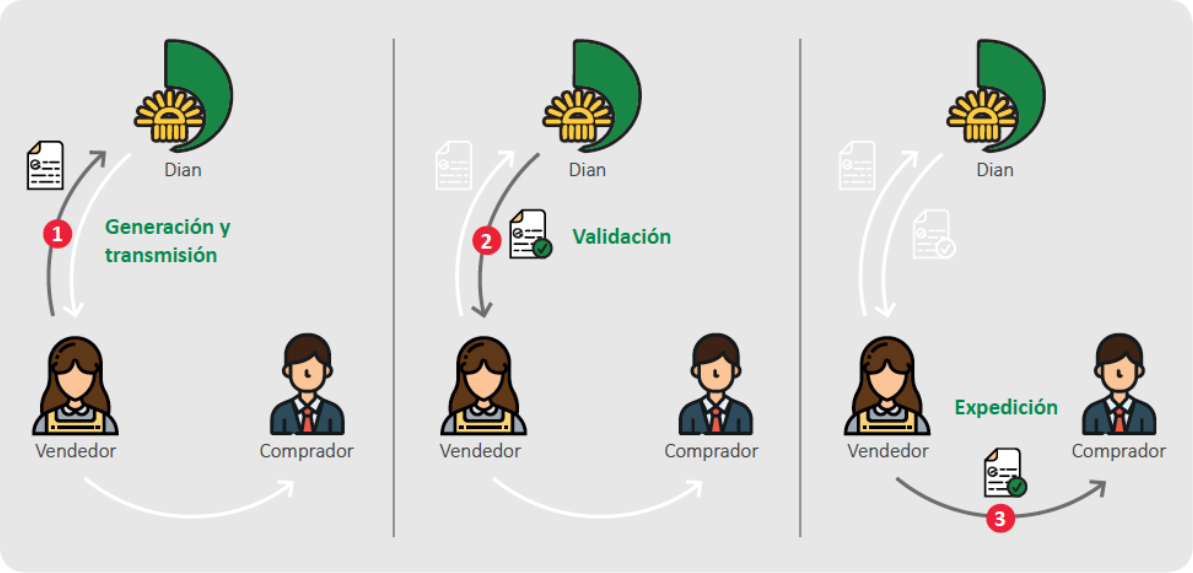


Ilustración 81 proceso de distribución; tomado de [34]

3.6.1 Curl

El sobre SOAP generado anteriormente debe ser enviado al web service por lo que es necesario usar una herramienta que nos permita transferir recursos por internet.

Curl es una herramienta de trasferencia de archivos por medio de protocolos de internet, está disponible tanto como en línea de comandos como en librerías [35]. permite usar métodos del protocolo HTTP como get, post, put, delete entre otros lo que facilitara la transferencia del sobre SOAP.

php cuenta con la librería de curl por lo que permite transferir archivos de manera directa, a continuación, se muestra como se configuro los comando de curl para trasferir el sobre SOAP.

```
$url = $wsdl; // URL del web service
$curl = curl_init($url); // crear instancia curl
curl_setopt($curl, CURLOPT_URL, $url);  // agregar la url destino a curl
curl_setopt($curl, CURLOPT_POST, true); //seleccionar el método post
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true); //habilita la transferencia de respuesta
//cabeceras
$headers = array(
  "Content-type: application/soap+xml",
  "Accept: application/xml",
);

curl_setopt($curl, CURLOPT_HTTPHEADER, $headers); //agregar cabeceras
$data= file_get_contents("request.xml"); //obtener contenido del sobre SOAP
curl_setopt($curl, CURLOPT_POSTFIELDS, $data); //agregar el sobre SOAP a curl
curl_setopt($curl, CURLOPT_SSL_VERIFYHOST, false); //deshabilitar comprobación de host
curl_setopt($curl, CURLOPT_SSL_VERIFYPEER, false); //deshabilitar la opción de comprobar certificados

$resp = curl_exec($curl); //enviar mensaje
curl_close($curl); // cerrar conexión

echo $resp; // respuesta del web service
```

La respuesta contiene un XML con todos los datos correspondientes, como se mencionó anteriormente en los métodos disponibles del servicio web de la DIAN.

3.6.2 SMTP (Simple Mail Transfer Protocol)

La emisión de facturas electrónicas de venta requiere que sea enviado el formato XML a la DIAN y la factura en PDF y XML al adquirente, en el proceso anterior se envió la factura XML al web service de la DIAN mediante la herramienta curl pero es necesario hacer uso del protocolo SMTP para enviar la factura al adquiérete por medio de correo electrónico.

Existen librerías que permiten hacer uso del protocolo SMTP lo que facilita este proceso, en el lenguaje de programación php se tiene la librería PHPMailer.

Antes de poder enviar correos electrónicos desde php se necesita habilitar la cuenta de correo electrónico y activar la opción de acceso a aplicaciones poco seguras, esto con el objetivo de que se permita acceder a la cuenta de correo.

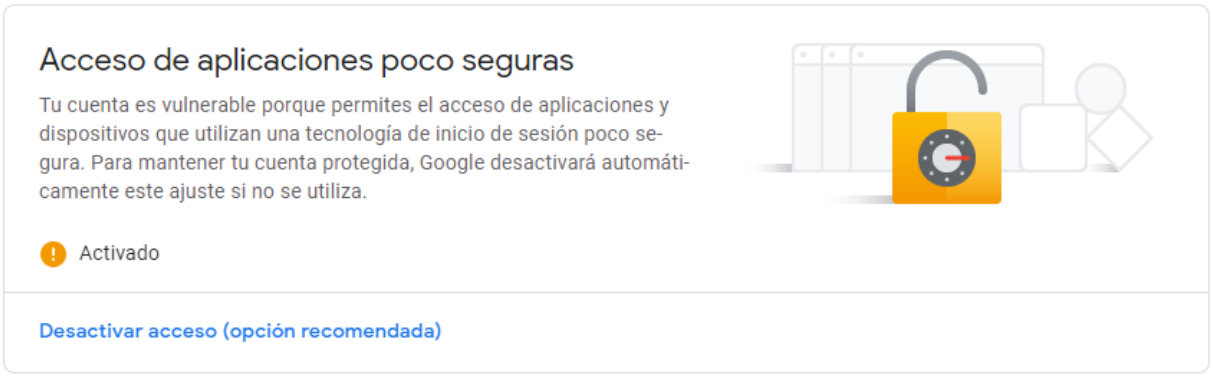


Ilustración 82 gmail configuración

Es necesario conocer el puerto TCP del servidor de correo electrónico para Gmail se tiene que es el puerto 587, este varía de acuerdo al servidor SMTP. A continuación, se muestra el uso de PHPMailer para el envío de correos.

```
//se debe incluir la clase de PHPmailer de la libreria

    $mail = new PHPMailer(true);
    $mail->SMTPDebug = 0; // Habilitar la salida de depuración detallada
```

```
$mail->isSMTP();//Enviar usando SMTP
$mail->Host= 'smtp.gmail.com';// Configura el servidor SMTP para enviar
$mail->SMTPAuth = true;// Habilitar la autenticación SMTP
$mail->Username ='proyectogradodarwin@gmail.com';// username SMTP
$mail->Password   = 'password';  //SMTP password
$mail->SMTPSecure=PHPMailer::ENCRYPTION_STARTTLS;// Habilita cifrado TLS
$mail->Port= 587;//puerto TCP del servidor de correo
$mail->setFrom('darwin.cerpa1@gmail.com','Darwin Manuel Mercado Cerpa');
$mail->addAddress('correo destino','destinatario');//agregar el destino
    //agregar documentos
$mail->addAttachment('Facturacion/factura.pdf');
$mail->addAttachment('Facturacion/invoice.xml');
//Content
$mail->isHTML(true);// establecer HTML en el correo
$mail->Subject = 'Factura Electronica. Emisor Darwin Mercado Cerpa';
$mail->Body = '<h1>Factura electrónica</h1>
            <p>Estimado:</p>
            Darwin Manuel Mercado Cerpa hace entrega de factura';
$mail->send(); // enviar correo
```

3.7 COSTOS DE IMPLEMENTACIÓN

Los costos de implementación de un sistema de facturación electrónica pueden variar según los criterios necesarios para el desarrollo tanto como el lenguaje y el entorno donde se ejecutará. Para un entorno web se debe tener en cuenta si es necesario subirlo a internet o si puede ser ejecutado de manera local para el cual no se asumen los valores de hosting y base de datos.

Materiales y servicios	Duración	Cantidad	Precio COP
Certificado de firma digital (GSE)	1 año	1	205,000.00
Hosting web (AWS ec2-a1.medium)	1 mes	1	67,987.00

Base de datos (AWS RDS postgres-db.t3.small)	1 mes	1	95,981.00

Tabla 9 Costos de implementación

El certificado digital puede ser adquirido por un periodo mayor a 1 año por lo generar entre más largo es el periodo de expiración, ofrecen mejores precios por ejemplo con la entidad GSE el mismo certificado con una duración de 2 años tiene un precio de 247.520.

Los precios de instancias EC2 y RDS varían según la capacidad de cómputo y almacenamiento.

CAPÍTULO IV: RESULTADOS

En este capítulo se recopilarán los resultados y evidencias que dan cumplimiento a los objetivos propuestos para el desarrollo de este proyecto.

4.1 APLICACIÓN WEB

La aplicación web ayuda navegar entre las diferentes opciones que puede presentar un sistema de facturación, este permitirá el registro de múltiples facturadores de manera independiente, la primera vista que se encontrará al entrar al aplicativo será la página de inicio la cual permitirá registrarse e iniciar sesión.

Ilustración 83 aplicación web Registro de usuario

Para realizar el registro se debe ingresar los datos solicitados y no estar registrado anteriormente, una ves se complete el proceso de registro se podra iniciar sesion por medio del correo electronico registrado y contraseña.

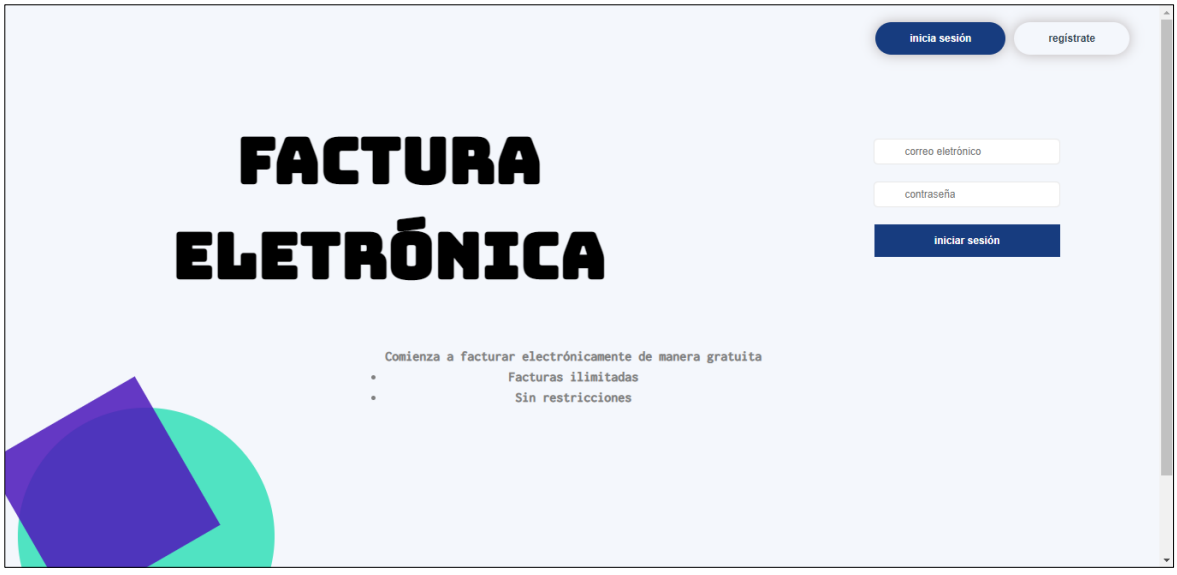


Ilustración 84 aplicación web iniciar sesión

Una vez ingresado al sistema nos dirigirá a la primera opción que nos indicará la cantidad de clientes, productos y facturas registradas.

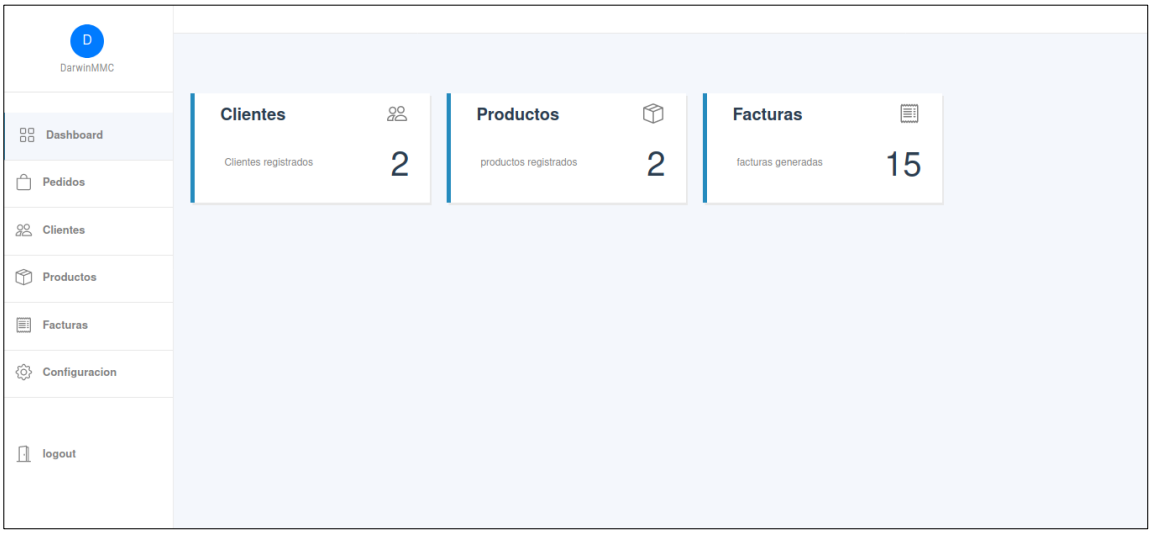


Ilustración 85 aplicación web dashboard

Como segunda opción se tiene pedidos esta nos permitirá generar pedidos y la factura correspondiente, para este proceso se debe contar con clientes y productos previamente registrados, para la selección del cliente se debe buscar con número de documento de identidad en el campo cedula y seleccionar los productos involucrados y la cantidad solicitada, una vez seleccionado los productos y el cliente se visualizara información acerca del pedido en la panel derecho donde se podrá hacer él envió de la factura o limpiar los datos.

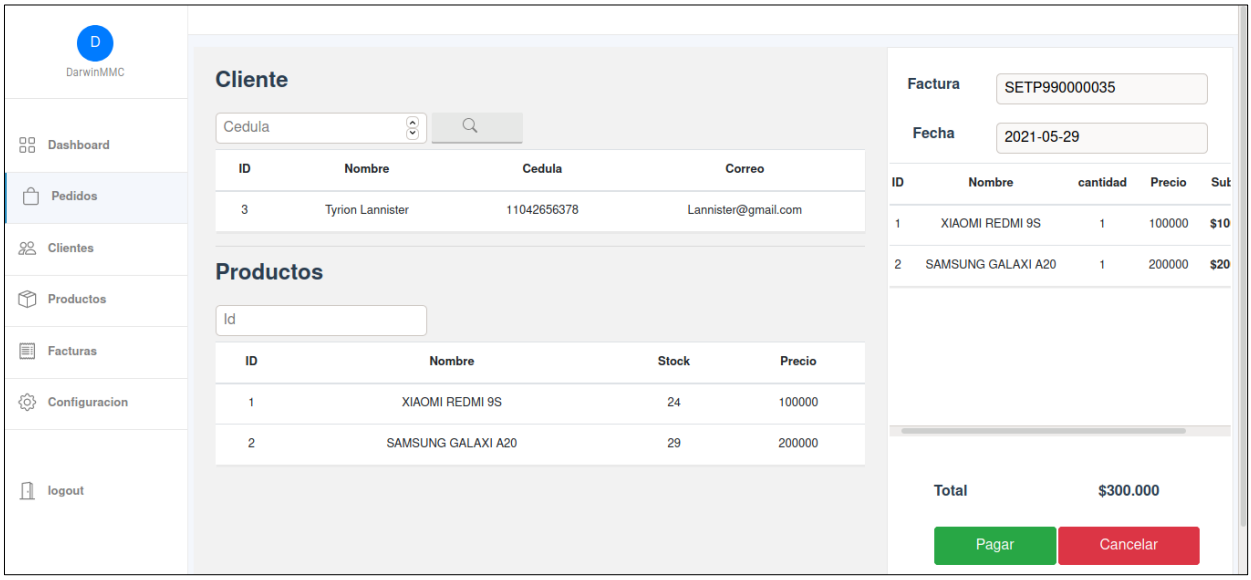


Ilustración 86 aplicación web Pedidos

La opción de clientes permite visualizar los clientes registrados por medio de una tabla donde se podrán eliminar. El panel derecho permite el registro de nuevos clientes ingresando todos los valores solicitados.

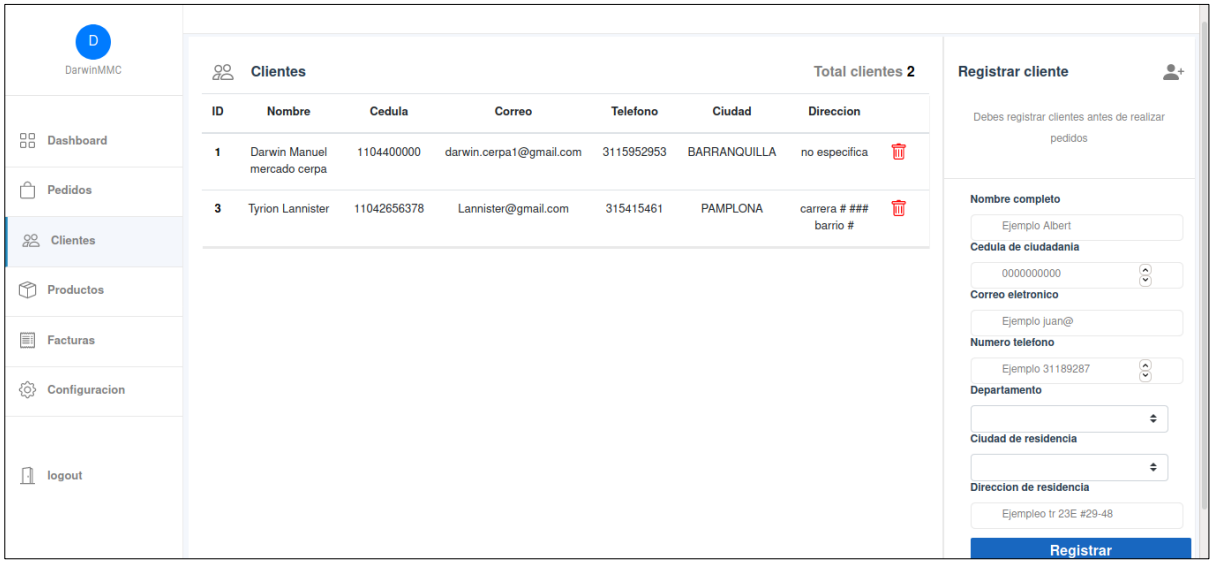


Ilustración 87 aplicación web clientes

El panel producto permite visualizar, registrar y eliminar productos estos productos también son visualizados en la opción de pedido.

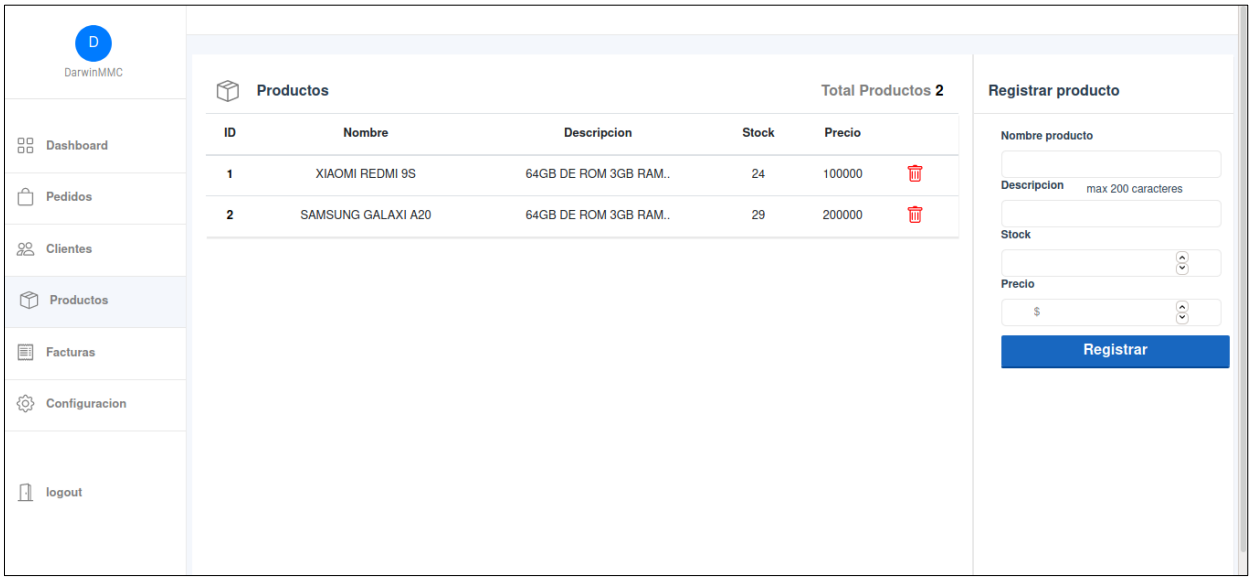


Ilustración 88 aplicación web productos

La opción factura permite visualizar los pedidos realizados y el estado de validación ante el web service de la DIAN. El color rojo indica que la factura no aprobó el proceso de validación, el color naranja indica que la factura se encuentra en cola para el proceso de validación, el color verde indica que la factura fue validada y emitida correctamente.

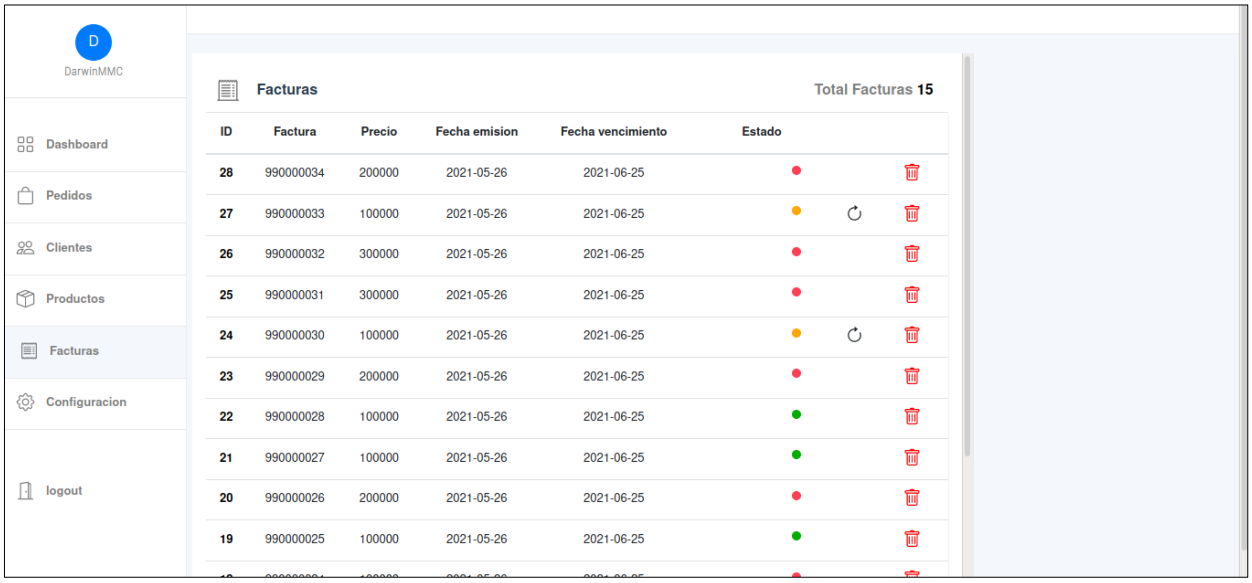


Ilustración 89 aplicación web facturas

La opción configuración debe ser diligenciado ante de realizar cualquier proceso de emisión de factura. Se deberán completar las 4 opciones requeridas, primero se debe seleccionar el entorno en el que se enviarán las facturas, los cuales pueden ser en entorno de habilitación, se usa para la realizar el set de pruebas en los servicios web de la DIAN para habilitarse como facturador electrónico, el segundo entorno es el de producción este entorno se utiliza para enviar las facturas validas a los métodos del servico web para la validación de facturas y registro de facturas electrónicas.

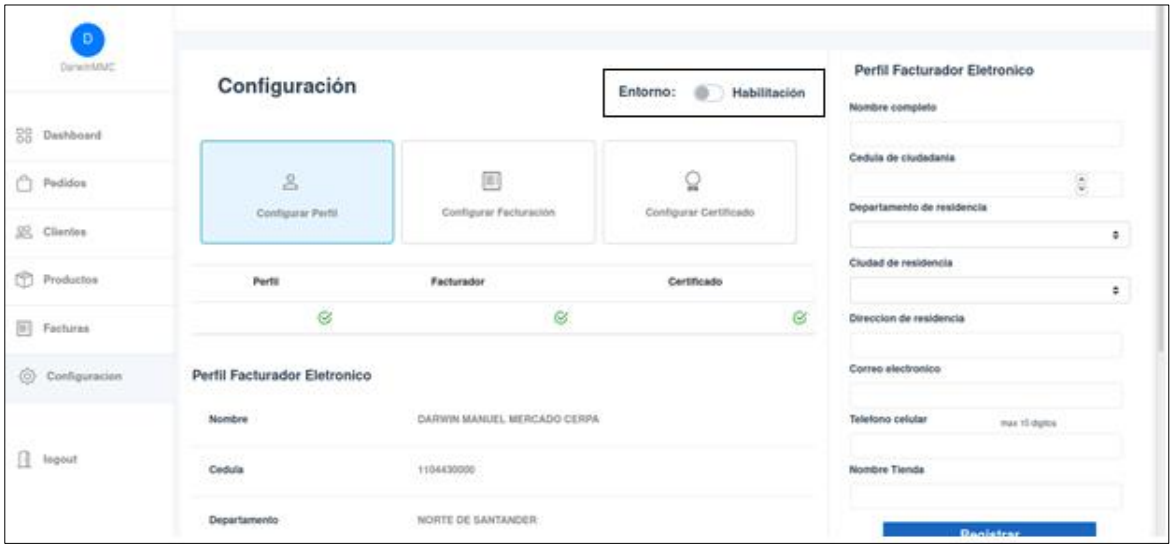


Ilustración 90 aplicación web configuración-entorno

Posterior a la selección del entorno se debe configurar el perfil del facturador en el formulario ubicado en la parte derecha, este contendrá información del obligado a facturar estos datos deben ser llenado con la información presente el RUT.

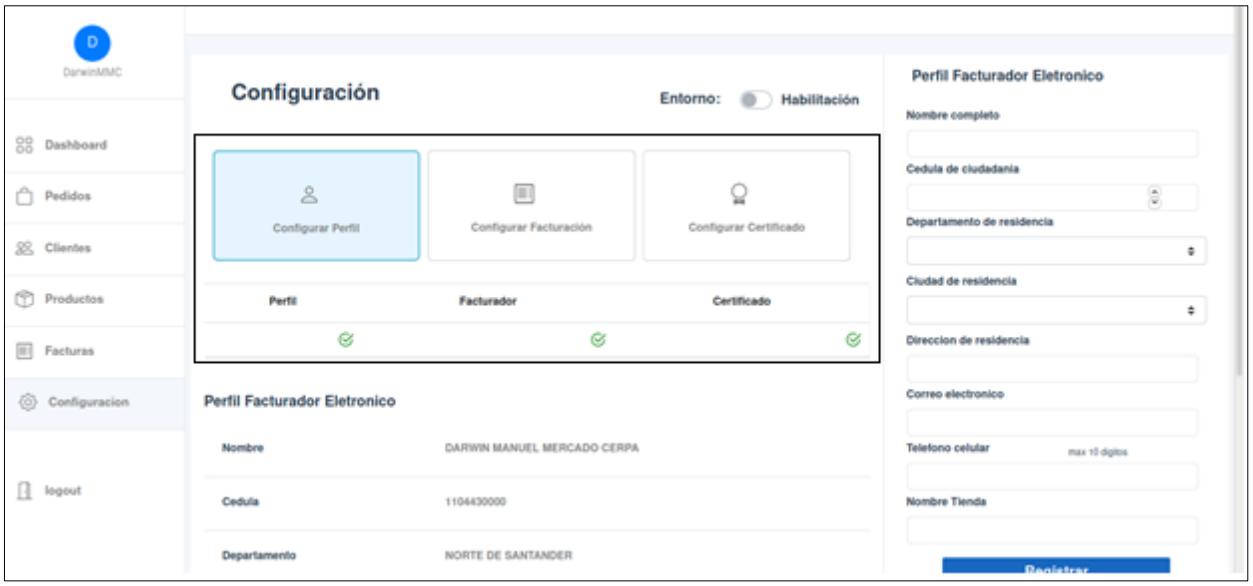


Ilustración 91 aplicación web configuración

La opción de configurar facturador contendrá la información técnica requerida para él envió de facturas electrónicas, esta información es suministrada por el sistema de habilitación de la DIAN, para tener acceso a este sistema debe estar registrado en el RUT e ingresar a habilitación para facturación electrónica e ingresar su documento de identificación, este enviará un token al correo registrado en el RUT, con este podrá acceder a este sistema.

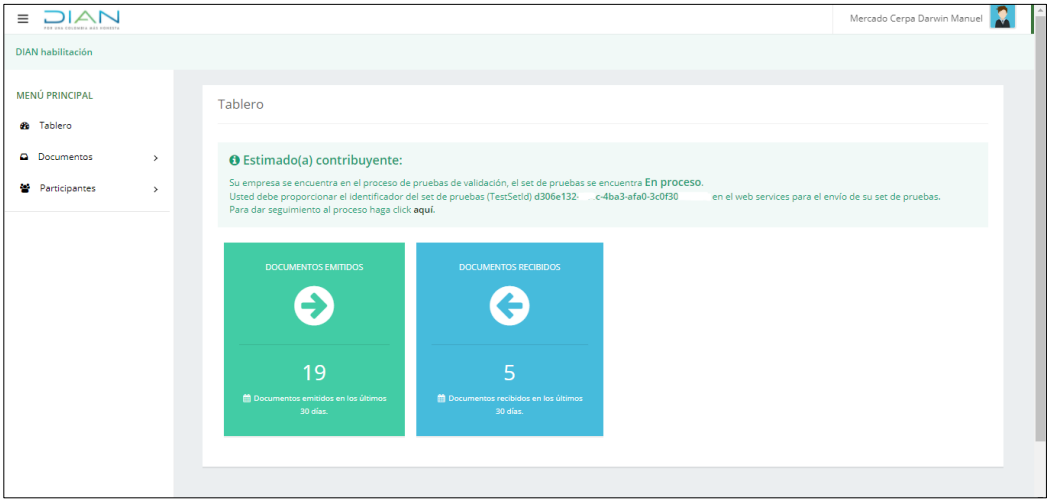


Ilustración 92 sistema DIAN habilitación

Una vez ingresado a este sistema debemos dirigirnos a la opción Participantes, facturador.

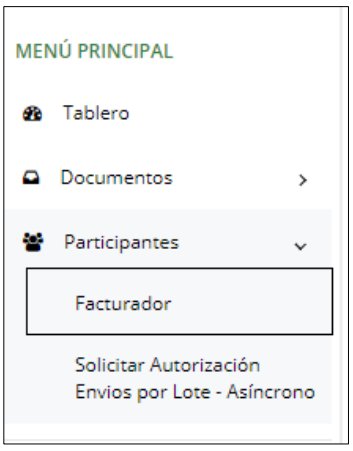


Ilustración 93 sistema DIAN facturador

Dar clic en configurar modo de operación.

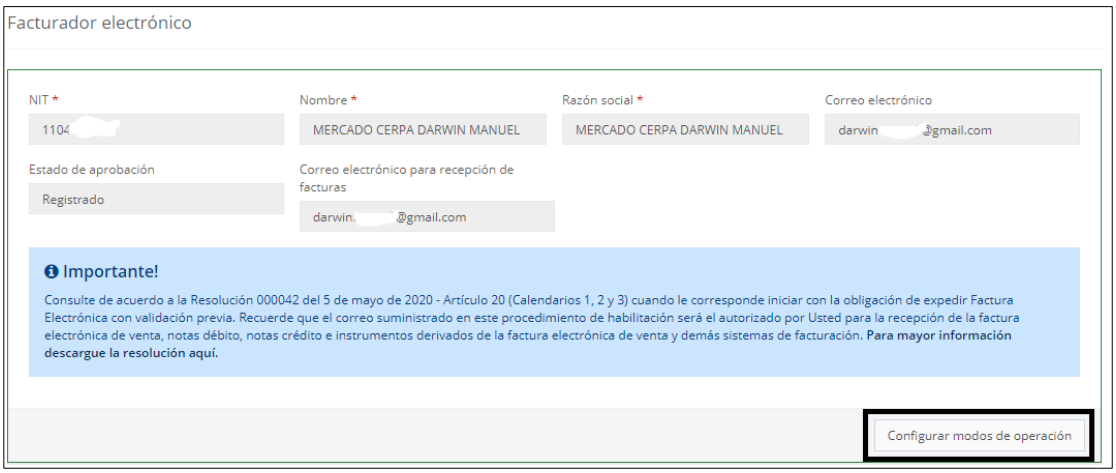


Ilustración 94 sistema DIAN habilitación configurar modo

A continuación, debe registrar el sistema y asignar y pin aleatorio.

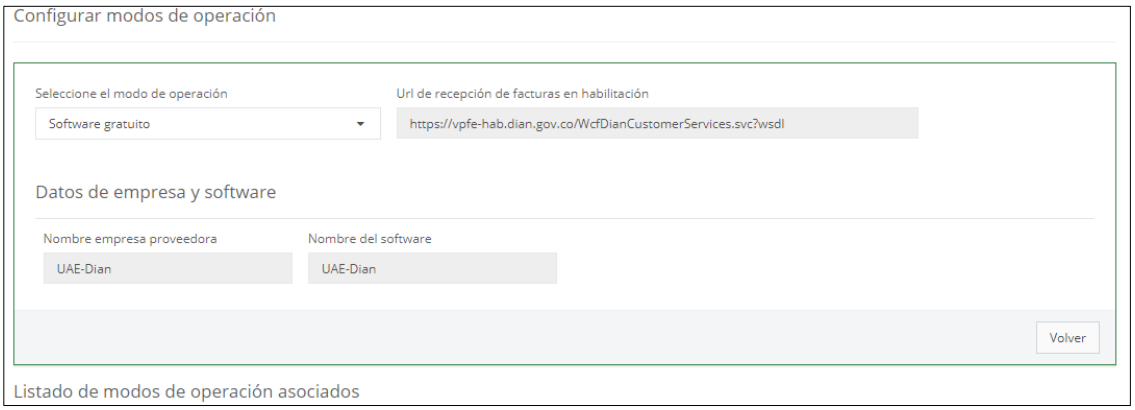


Ilustración 95 sistema DIAN registrar software

Una vez registrado encontraremos la información en la parte inferior de la página.

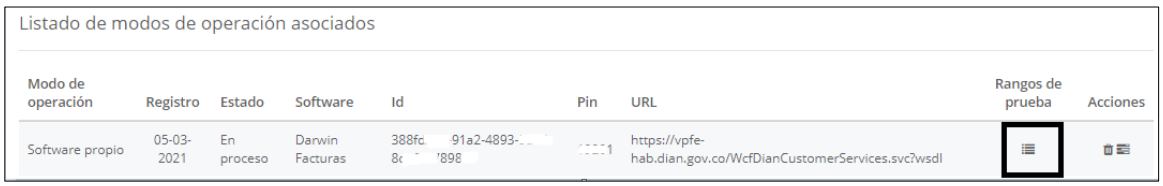


Ilustración 96 sistema DIAN información técnica

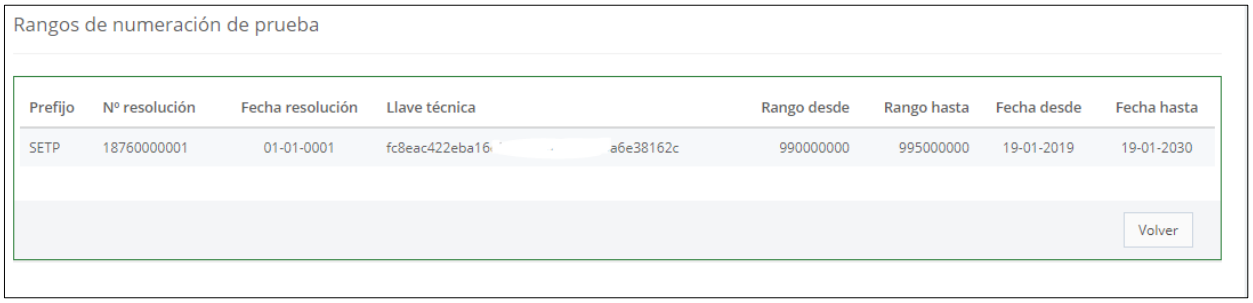


Ilustración 97 sistema DIAN Rango de prueba

Como ultima configuración debemos registrar un certificado digital con extensión p12 y la contraseña del certificado digital.

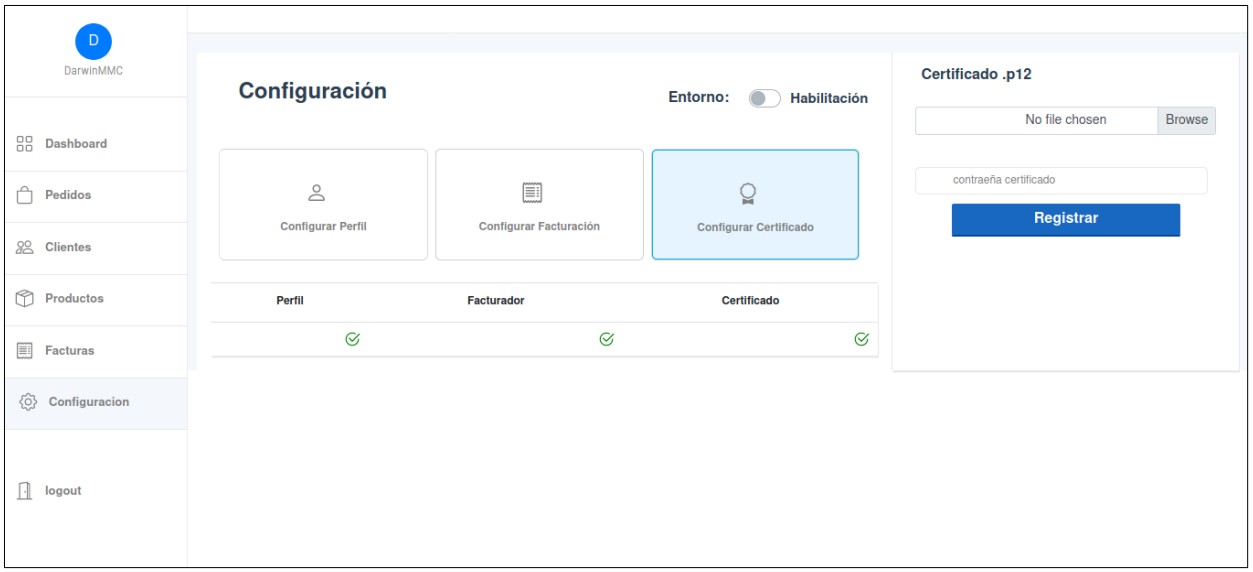


Ilustración 98 aplicación web configuración certificado

4.2 VALIDACIÓN DE FACTURAS ELECTRÓNICAS

El proceso de validación de facturas se efectúa en el web service de la DIAN donde este valida la estructura del documento UBL invoice y firma y devuelve el resultado de la validación dependiendo del tipo de método utilizado.

4.2.1 Emisión de facturas factura

Antes de generar facturas se debe configurar el modo operación, para habilitación se debe usar el método sendTestSetAsync este será el método seleccionado cuando se configura el entorno como habilitación, si el entorno se encuentra en producción este hará uso del método síncrono SendBillSync, algo a tener en cuenta es que este método se utiliza para él envió de facturas cuando se está habilitado como facturador, si se hace uso de este método no se tendrán en cuentas las facturas generadas con este método, pero puede ser usado para validar las facturas y no agotar la cantidad minina de facturas requeridas en el método sendTestSetAsync para habilitarse como facturador.

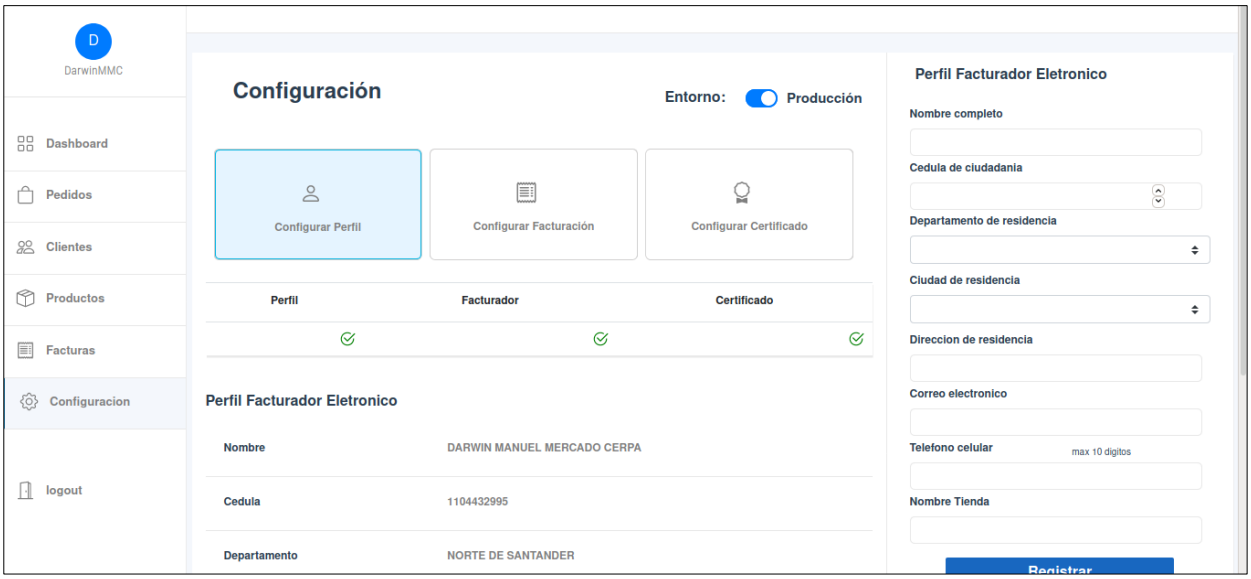


Ilustración 99 configuración de entorno

Para emitir las facturas se necesita buscar el adquiriente por medio del numero de documento como fue registrado anteriormente en el sistema y seleccionar los productos, solo se necesita dar click en pagar y este realizara el proceso de generar firmar y enviar la factura con los datos del adquieriente y el emisor de la factura.

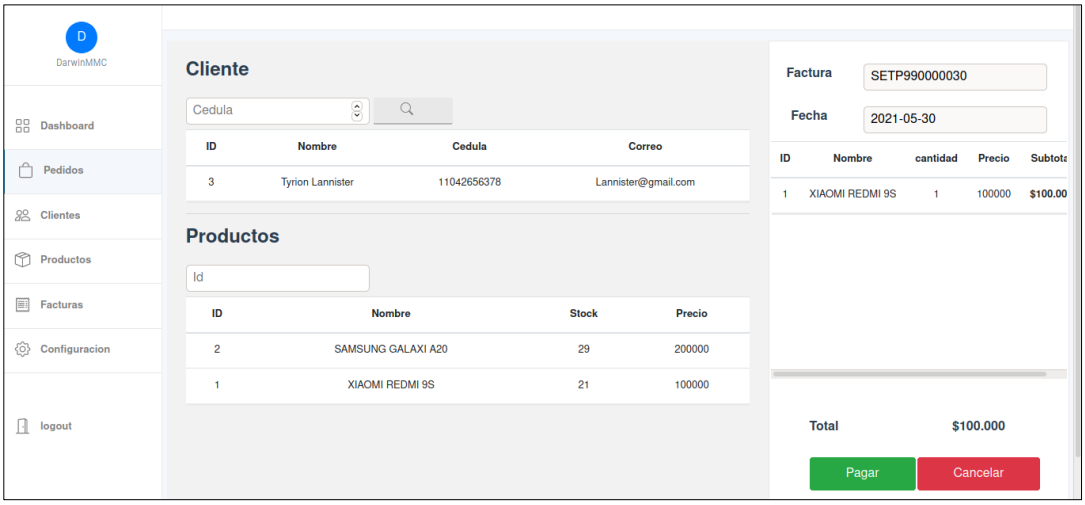


Ilustración 100 Generar pedido

La prueba se realiza en entorno de produccion por lo que hace uso del metodo sincrono lo queiere decir que el sistema de la DIAN retornara la informacion de la validacion de forma inmediata.

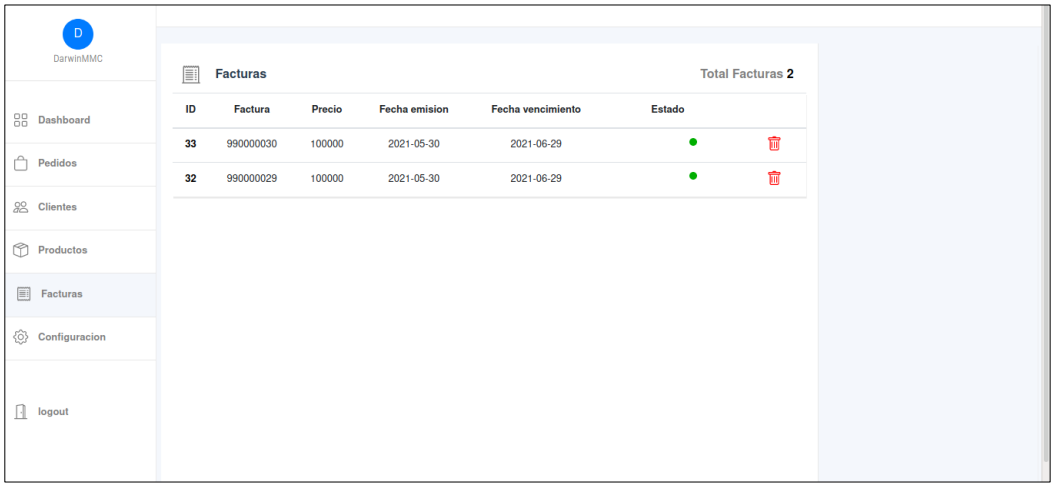


Ilustración 101 facturas generadas

El sistema de validacion retorna como respuesta un XML donde se encuentra toda la información correspondiente a la validacion. Para una factura sin errores se devolvera un valor 00 y si la factura contiene cualquiere clase de error devolvera 99 dentro del elemento StatusCode.

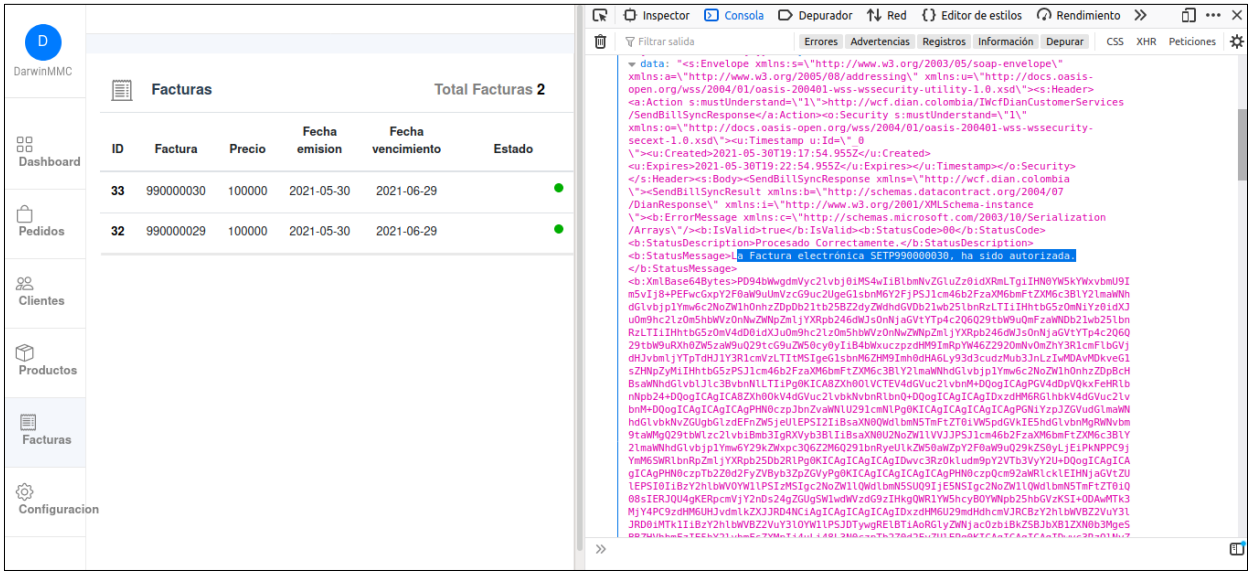


Ilustración 102 respuesta de validación

Una vez las facturas son aprobadas estas se pueden visualizar dentro del sistema de la DIAN, donde lista todos los documentos generados.

Ilustración 103 sistema DIAN facturas emitidas

Todas las facturas emitidas pueden ser descargadas tanto por el emisor como por el adquierenete por medio del enlace que se encuentra en el codigo QR como lo indica el anexo tecnico. Pero se debe tener encuenta que solo se podra acceder a las facturas cuando se esta habilitado como facturador electronico.

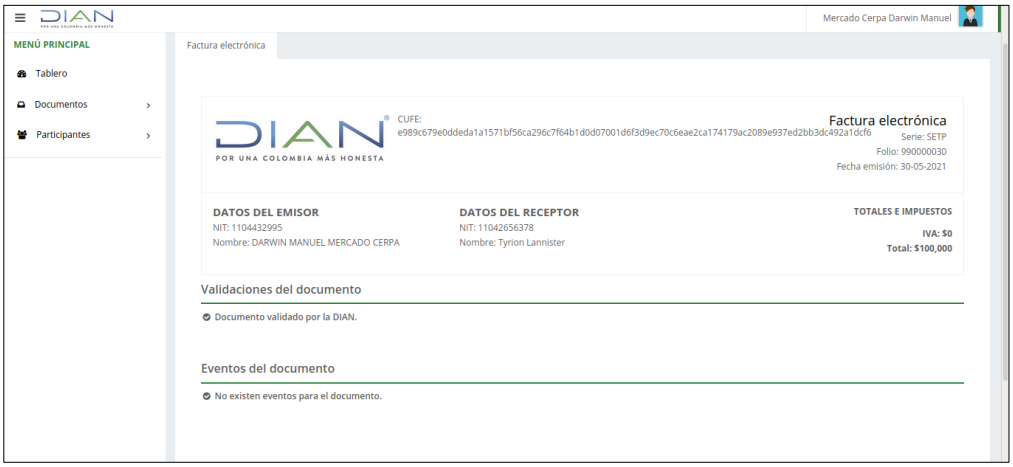


Ilustración 104 información de factura emitida

Una vez se recibe el resultado de la validación se hace él envió de la factura en formato PDF y XML al adquirente por medio de SMTP.

Ilustración 105 correo electrónico factura

La factura en formato PDF contendrá la información acerca de la operación y un código QR donde se encuentra información acerca de la factura y la ruta de consulta para el documento ante la DIAN.

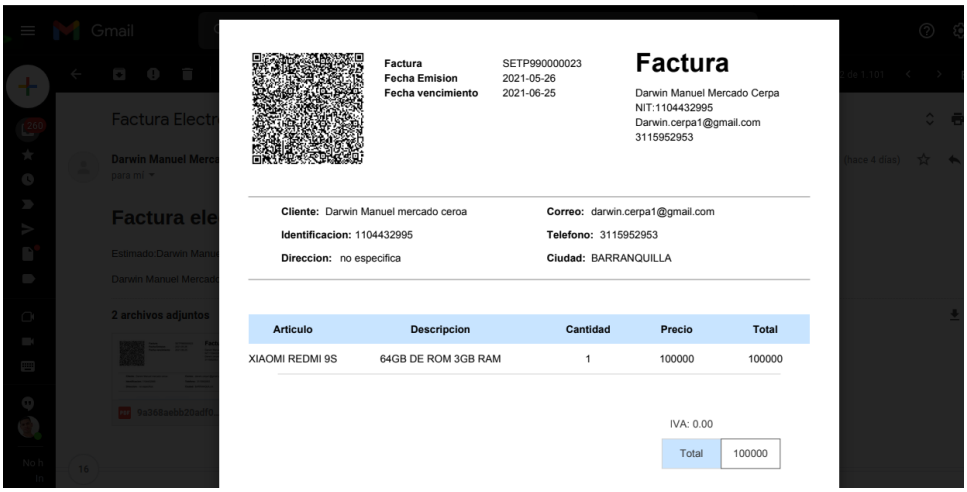


Ilustración 106 correo factura PDF

en el correo también se encontrará adjunto la factura XML que fue aprobada.

Ilustración 107 correo factura XML

En el proceso de habilitación se deben enviar una cantidad mínima de facturas electrónicas, notas crédito y notas debito aprobadas para habilitarse como facturador, el siguiente grafico indica el avance de este proceso.

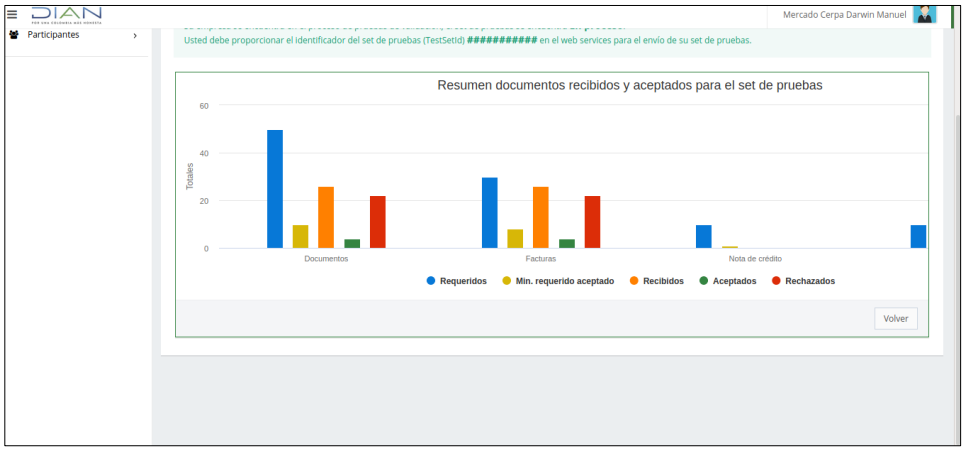


Ilustración 108 grafico habilitación

Ilustración 109 grafica habilitación aceptadas

La mayoría de la validación se ha realizado con el método SendBilllSync el cual, valida los documentos, pero no se tienen en cuenta para habilitación con el objetivo de no agotar la cantidad mínima de facturas requeridas. la barra de color verde indica la cantidad de facturas aceptadas para la etapa de habilitación como facturador electrónico.

La información con respecto a la cantidad de documentos requeridos para aprobar el proceso de habilitación se encuentra en la opción de configuración de modo de operación.

Modo de operación	Registro	Estado	Software	Id	Pin	URL	Rangos de prueba	Acciones
Software propio	05-03-2021	En proceso	Darwin Facturas	[redacted]		https://vpfe-hab.dian.gov.co/WcfDianCustomerServices.svc?wsdl		

Ilustración 110 sistema dian información proceso de habilitación

Ilustración 111 sistema dian cantidad de documentos requeridos

Para habilitarme como facturador autorizado por la DIAN para emitir facturas electrónicas de ventas debo superar la cantidad de documentos correctos.

CAPÍTULO V: CONCLUSIONES

- Es sumamente importante dirigirse a las fuentes oficiales para adquirir fundamentación técnica y detallada. de las tecnologías utilizadas.
- En el proceso de firma XAdES es importante que los nombres de espacio del documento padre coincidan con los nombres de espacios del elemento al que se le calcula el resumen o se aplica firma, en el caso de la factura Invoice este contendrá el formato de firma por lo es necesario que todos los nombres de espacios incluidos en el elemento antes de calcular el resumen sean los mismo que están el documento Invoice.
- Durante el proceso de firma es importante contar con una herramienta que permitan validar la estructura de la firma debido a que el sistema de validación de la DIAN no notifica el problema que contiene la firma, este solo indica que existe un error en el formato.
- Para la construcción de la factura Invoice no siempre es necesario hacer uso de DOM debido a que resulta más complejo y tedioso crear toda la factura, en el caso del lenguaje de programación php permite convertir un string en un objeto DOM lo que resulta más conveniente crear una plantilla en string y convertir esta en un objeto DOM para hacer uso de método canonical XML C14N, una alternativa a este proceso es construir toda la plantilla canonizada siguiendo las reglas de canonización.
- En el proceso de envió del sobre SOAP es importante que las fecha y hora del equipo en que se ejecuta el sistema este sincronizado con el reloj de la súper intendencia de industria y comercio el cual determina la hora legal en Colombia, de lo contrario el sistema de validación de la DIAN indicara un mensaje de error al comprobar la seguridad del sobre SOAP.

REFERENCIAS

[1] "Leyes desde 1992 - Vigencia expresa y control de constitucionalidad [ESTATUTO_TRIBUTARIO]." http://www.secretariasenado.gov.co/senado/basedoc/estatuto_tributario.html (accessed May 19, 2021).

[2] "Decreto 358 del 2020." https://www.dian.gov.co/impuestos/factura-electronica/Documents/Decreto_358_05032020.pdf (accessed May 19, 2021).

[3] *RESOLUCIÓN NÚMERO 000042 05 de Mayo de 2020-DIAN*. .

[4] M. Castanedo Doña, "Capítulo 2. Introducción a XML 2.1 Introducción."

[5] M. M. Henry S. Thompson, David Beech, "W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures." https://www.w3.org/TR/xmlschema11-1/#xsover (accessed May 19, 2021).

[6] "XML - Informacion General - Tutorialspoint." https://www.tutorialspoint.com/es/xml/xml_overview.htm (accessed May 19, 2021).

[7] H. S. T. Tim Bray, Dave Hollander, Andrew Layman, Richard Tobin, "Namespaces in XML 1.0 (Third Edition)." https://www.w3.org/TR/xml-names/ (accessed May 19, 2021).

[8] S. L. Mora, "XML Namespaces." Accessed: May 19, 2021. [Online]. Available: http://gplsi.dlsi.ua.es/~slujan/.

[9] "What is the Document Object Model?" https://www.w3.org/TR/WD-DOM/introduction.html (accessed May 19, 2021).

[10] F. H. John Boyer, Glenn Marcy, Pratik Datta, "Canonical XML Version 2.0." https://www.w3.org/TR/xml-c14n2/ (accessed May 19, 2021).

[11] M. Castanedo Doña, "Canonización Capítulo 5. Canonización 5.1 Introducción."

[12] F. Y. Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, "Extensible Markup Language (XML) 1.0 (Fifth Edition)." https://www.w3.org/TR/xml/ (accessed May 19, 2021).

[13] G. K. H. Jon Bosak, Tim McGrath, "Universal Business Language Version 2.1." http://docs.oasis-open.org/ubl/UBL-2.1.html (accessed May 19, 2021).

[14] "ISO - ISO 15000-5: 2014 - Lenguaje de marcado extensible de negocios electrónicos (ebXML) - Parte 5: Especificación de componentes centrales (CCS)." https://www.iso.org/standard/61433.html (accessed May 31, 2021).

[15] "RFC 4648." https://datatracker.ietf.org/doc/html/rfc4648 (accessed May 19, 2021).

[16] A. Lorena and A. Bazurto, "Características y aplicaciones de las funciones resumen criptográficas en la gestión de contraseñas," Universidad de alicante.

[17] J. Weiss, *Java Cryptography Extensions: Practical Guide for Programmers*. Elsevier Inc., 2004.

[18] J. A. M. Fernández, "Sistemas seguros de acceso y transmisión de datos (MF0489_3) - Juan Andres Maíllo Fernández - Google Libros." https://books.google.com.co/books/about/Sistemas_seguros_de_acceso_y_transmisió.html?id=lo6fDwAAQBAJ&printsec=frontcover&source=kp_read_button&redir_esc=y#v=onepage&q&f=false (accessed May 20, 2021).

[19] "¿Sabías que existen distintos tipos de cifrado para proteger la privacidad de nuestra información en Internet? | Oficina de Seguridad del Internauta." https://www.osi.es/es/actualidad/blog/2019/07/10/sabias-que-existen-distintos-tipos-de-cifrado-para-proteger-la-privacidad (accessed May 20, 2021).

[20] M. N. Takeshi Imamura, Blair Dillaway, Ed Simon, Kelvin Yiu, "XML Encryption Syntax and Processing Version 1.1." https://www.w3.org/TR/xmlenc-core1/ (accessed May 20, 2021).

[21] E. S. Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, "XML Signature Syntax and Processing Version 1.1." https://www.w3.org/TR/xmldsig-core1/ (accessed May 20, 2021).

[22] "XML Advanced Electronic Signatures (XAdES)," 2002. Accessed: May 20, 2021. [Online]. Available: https://uri.etsi.org/01903/v1.1.1/ts_101903v010101p.pdf.

[23] "EUR-Lex - 31999L0093 - EN - EUR-Lex." https://eur-lex.europa.eu/legal-content/ES/ALL/?uri=CELEX%3A31999L0093 (accessed May 20, 2021).

[24] "Simple Object Access Protocol (SOAP) 1.1." https://www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383486 (accessed May 20, 2021).

[25] G. A. HERNANDO, "DISEÑO DE UN SOFTWARE CONTABLE CON LA INCLUSIÓN DE FACTURACIÓN ELECTRÓNICA PARA PEQUEÑAS Y MEDIANAS EMPRESAS," UNIVERSIDAD DE PAMPLONA, 2018.

[26] "Lenguaje de dominio específico para generar facturas electrónicas de acuerdo a los requerimientos técnicos de la DIAN - invoiceQL." https://repositorio.unal.edu.co/handle/unal/78681 (accessed May 30, 2021).

[27] K. xiomara Chipana Luna and J. B. Camacho Medina, "Análisis, Diseño e Implementación de Facturación Electrónica para la optimización de los procesos tributarios en la Empresa CORPORACIO´ N AGROLATINA S.A.C.," 2018.

[28] "Orden Administrativa 4 de 1989 DIAN - Dirección de Impuestos y Aduanas Nacionales - Colombia." https://www.redjurista.com/Documents/orden_administrativa_4_de_1989_dian_-_direccion_de_impuestos_y_aduanas_nacionales.aspx#/ (accessed May 28, 2021).

[29] "GSE – Gestión de Seguridad Electrónica." https://gse.com.co (accessed May 21, 2021).

[30] "RFC 2396." https://datatracker.ietf.org/doc/html/rfc2396 (accessed May 22, 2021).

[31] "Chercker de conformidad con firma electrónica ETSI." https://signatures-conformance-checker.etsi.org/pub/index.php (accessed May 24, 2021).

[32] Nadalin, "Web Services Security: 2 SOAP Message Security 1.0," 2004. Accessed: May 24, 2021. [Online]. Available: http://www.oasis-open.org/committees/wss.

[33] "Web Services Addressing (WS-Addressing)." https://www.w3.org/Submission/ws-addressing/ (accessed May 25, 2021).

[34] "proceso-factura-electronica.png (1013×505)." https://cdn.actualicese.com/Imagen-editoriales/proceso-factura-electronica.png (accessed May 31, 2021).

[35] "curl." https://curl.se/ (accessed May 29, 2021).