

MODELO PARA EL MANEJO Y SEGURIDAD DE LOS USUARIOS MEDIANTE EL USO
DE UN MICROSERVICIO PARA LA CONSTRUCCIÓN DE UNA PLATAFORMA NO-
CODE

Autor

JOAN MANUEL MONTAÑEZ MORANTES

Directora

ALIN ORJUELA DUARTE

PhD Educación

INGENIERIA DE SISTEMAS

INGENIERIAS E.E.S.T

INGENIERIAS Y ARQUITECTURA



UNIVERSIDAD DE PAMPLONA

PAMPLONA, junio 6 de 2022

DEDICATORIA

Gracias a mis padres, a mi hija, a mi familia y amigos.

Resumen del Proyecto

Este documento presenta un modelo para el manejo y seguridad de los usuarios, con el uso de la arquitectura de microservicios consumidos desde una plataforma especializada en No-Code.

Este modelo se obtuvo como parte de la práctica profesional llevada a cabo en HelloGuru Inc, dentro de la cual se estableció el estado del arte respecto a los objetos de estudio necesario para la definición del modelo, se definió los requerimientos de la plataforma No-Code, y se validó el funcionamiento del modelo.

Palabras Clave:

No-Code, arquitectura de microservicios.

Abstract

This document presents a model for the management and security of users, with the use of microservices architecture consumed from a platform specialized in No-Code.

This model was obtained as part of the professional practice carried out at HelloGuru Inc, within which the state of the art was established regarding the objects of study necessary for the definition of the model, the requirements of the No-Code platform were defined, and the performance of the model was validated.

Keywords

No-Code, microservices architecture.

Tabla de contenido

1	Introducción	3
1.1	Planteamiento del problema	3
1.2	Justificación	4
1.3	Objetivos.....	5
1.3.1	Objetivo General	5
1.3.2	Objetivos Específicos	5
1.4	Acotaciones	5
2	Marco teórico	7
2.1	Estado del Arte	7
2.1.1	Microservicios, La Clave Para Evolucionar Hacia Una Arquitectura Que Favorece La Innovación.....	7
2.1.2	Microservices in the Enterprise, 2021	7
2.1.3	Guía de Microservicios Punto de Vista.....	8
2.1.4	Investigación y Desarrollo de la Arquitectura Orientada a Microservicios Para la Implementación de Módulos en un Sistema Web	9
2.1.5	Introducción a la Seguridad Informática y el Análisis de Vulnerabilidades	10
2.2	Referentes Teóricos	11
2.2.1	Microservicios un Enfoque Integrado	11
2.2.2	Arquitectura SOA.....	13
2.2.3	Arquitectura Monolítica	14
2.2.4	Arquitectura de Microservicios	17

2.2.5	Diferencias Entre la Arquitectura Monolítica y la Arquitectura de Microservicios	21
2.3	No-Code.....	23
2.4	Herramientas que se Utilizaron en el Microservicio	24
2.4.1	Kotlin.....	24
2.4.2	Mapeo Objeto-Relacional (ORM).....	25
2.4.3	Api de Persistencia de Java (JPA).....	26
2.4.4	Spring	26
2.4.5	Arquitectura rest.....	27
2.4.6	Springboot	29
2.4.7	Postgresql	30
2.5	Tecnologías que Facilitan el Manejo y la Seguridad de los Usuarios	30
2.5.1	Algoritmos de Cifrado.....	32
2.5.2	Algoritmo Simétrico Fernet	35
2.5.3	Standard Json Web Token (JWT)	40
2.5.4	Fernet Vs Json Web Token	43
2.5.5	Algoritmo de Hasheo de Contraseñas ARGON2	44
2.5.6	Cortafuegos o Firewall	45
2.5.7	Zona Desmilitarizada	46
2.5.8	Firewall de Aplicaciones Web (Waf).....	47
2.5.9	Balanceador de Carga.....	47
2.5.10	Grupos de Seguridad de Amazon	47
2.5.11	Proxy.....	48

3	Modelo Para el Manejo y Seguridad de los Usuarios Mediante el Uso del Microservicio Users	49
3.1	Requerimientos del Sistema Para el Desarrollo del Modelo	49
3.2	Análisis de los Requerimientos del Sistema Para el Desarrollo del Modelo.....	50
3.3	Diagramas de Casos de Uso Del Microservicio Users	51
3.3.1	Flujo General del Microservicio Users	52
3.3.2	Iniciar Sesión.....	52
3.3.3	Regístrate	53
3.3.4	Cambiar Contraseña	54
3.3.5	Recuperar Contraseña	55
3.3.6	Actualizar Información Usuario.....	57
3.3.7	Mostrar Información del Usuario	58
3.3.8	Obtener Token de Invitado.....	59
3.4	Diagramas de Secuencia Del Microservicio Users.....	59
3.4.1	Obtener Token Invitado	60
3.4.2	Iniciar Sesión.....	61
3.4.3	Registrarse.....	62
3.4.4	Cambiar Contraseña	63
3.4.5	Recuperar Contraseña	64
3.4.6	Actualizar Información Usuario.....	65
3.4.7	Listar Información Usuario	66
3.5	Descripción Técnica de los Casos de Uso	66

3.5.1	Iniciar Sesión.....	67
3.5.2	Registrar Usuario.....	69
3.5.3	Cambiar Contraseña	71
3.5.4	Recuperar Contraseña	73
3.5.5	Actualizar Información del Usuario	75
3.5.6	Listar Información del Usuario	77
3.5.7	Token de Invitado.....	78
3.6	Diseño del Modelo.....	80
3.6.1	Diagrama De La Conexión Entre Microservicios	87
3.6.2	Diagrama de Clases	87
3.7	Validación del Modelo	88
3.7.1	Durante el desarrollo	88
3.7.2	Antes del despliegue a producción.....	90
4	Conclusiones	92
5	Referencias Bibliográficas	93

Tabla Ilustraciones

Ilustración 1 Referentes teóricos (Autoría propia).....	12
Ilustración 2 Arquitecturas (Autoría propia).....	12
Ilustración 3 Arquitectura Soa.....	13
Ilustración 4 Arquitectura Monolítica (irvingjuarez, 2021).....	15
Ilustración 5 Arquitectura de Microservicios (irvingjuarez, 2021).....	18
Ilustración 6 Herramientas que se utilizaron en el microservicio (Autoría propia)	24
Ilustración 7 Tecnologías que facilitan el manejo y seguridad de los usuarios (Autoría propia) ..	32
Ilustración 8 Diferencias entre algoritmo simétrico y asimétrico (Savvy Security, 2021)	34
Ilustración 9 Usando la clase cryptography.fernet (Cryptography, 2021)	36
Ilustración 10 contraseñas con Fernet (Cryptography, 2021)	40
Ilustración 11 token JWT firmado	42
Ilustración 12 contenido de un token JWT.....	42
Ilustración 13 Explicación gráfica de un cortafuegos (incibe, 2019).....	46
Ilustración 14 ejemplo de una red local con una DMZ (incibe, 2019).....	46
Ilustración 15 flujo general del sistema.....	52
Ilustración 16 diagrama de secuencia obtener token invitado.....	60
Ilustración 17 diagrama de secuencia iniciar sesión	61
Ilustración 18 diagrama de secuencia registrarse	62
Ilustración 19 diagrama de secuencia cambiar contraseña.....	63
Ilustración 20 diagrama de secuencia recuperar contraseña	64

Ilustración 21	diagrama de secuencia actualizar información usuario	65
Ilustración 22	diagrama de secuencia listar información usuario	66
Ilustración 23	Modelo para el manejo y seguridad de los usuarios mediante el uso del microservicio Users. (Autoría propia).....	81
Ilustración 24	Capa DMZ del modelo. (Autoría propia).....	83
Ilustración 25.	capa de aplicaciones del modelo. (Autoría propia).....	84
Ilustración 26.	Capa de almacenamiento del modelo (Autoría propia).....	85
Ilustración 27	Diagrama de la conexión entre microservicios	87
Ilustración 28	diagrama de clases.....	88

Tabla De Tablas

Tabla 1 Diferencias entre arquitectura de microservicios y monolítica (Cantó, 2018).....	23
Tabla 2 Ventajas y desventajas de Kotlin (Fernandez, 2018)	25
Tabla 3 Descripción de la clase cryptography.fernet (Cryptography, 2021)	37
Tabla 4 método generate_key() de Fernet (Cryptography, 2021).....	37
Tabla 5 método encrypt (data) de Fernet (Cryptography, 2021).....	37
Tabla 6 método encrypt_at_time (data) de Fernet (Cryptography, 2021).....	38
Tabla 7 método decrypt (token, ttl) de Fernet (Cryptography, 2021)	38
Tabla 8 método decrypt_at_time (token, ttl, current_time) de Fernet (Cryptography, 2021).....	39
Tabla 9 método extract_timestamp (token) (Cryptography, 2021).....	39
Tabla 10 Descripción del método argo2.hash (Ciphr Team, 2021)	45
Tabla 11 caso de uso iniciar sesión	53
Tabla 12 caso de uso registrarse.....	54
Tabla 13 caso de uso cambiar contraseña	55
Tabla 14 caso de uso recuperar contraseña	57
Tabla 15 caso de uso actualizar información usuario	58
Tabla 16 caso de uso mostrar información usuario	59
Tabla 17 caso de uso obtener token invitado	59
Tabla 18 descripción técnica caso de uso iniciar sesión	68
Tabla 19 descripción técnica caso de uso registrar usuario	71
Tabla 20 descripción técnica caso de uso cambiar contraseña.....	73
Tabla 21 descripción técnica caso de uso recuperar contraseña	75
Tabla 22 descripción técnica caso de uso actualizar información del usuario	77

Tabla 23 descripción técnica caso de uso listar información del usuario78

Tabla 24 descripción técnica caso de uso token de invitado79

1 Introducción

En esta sección se contextualiza el desarrollo del trabajo de grado, donde principalmente se aborda el problema a resolver. Con este fin, se plantearon objetivos que se desarrollaron en cada uno de los ítems, que dieron como resultado exitoso el desarrollo de un modelo para el manejo y seguridad de los usuarios mediante el uso de un microservicio, que ayudó en la creación e implementación de una plataforma No-Code empleando la misma arquitectura que se desarrollo en este proyecto.

1.1 Planteamiento del problema

En años anteriores las plataformas web se construían con una arquitectura monolítica que se estructuraba de una forma en la que todos los aspectos funcionales del mismo quedan ensamblados en una base de código única (Aplyca, 2018). Por lo tanto, dicha estructura trae grandes complicaciones para plataformas que tengan unan gran escalabilidad, además, es un reto para el equipo de desarrollo, debido a que es difícil trabajar en diversos ambientes simultáneamente.

El inconveniente más grande que tiene, es que al dañarse algo importante en el código o si el servidor se llegara a caer, esto ocasionaría que toda la plataforma dejara de funcionar, trayendo consecuencias negativas para los usuarios y la misma empresa dueña de la plataforma.

¿Cómo se podría solucionar esta problemática? actualmente, las nuevas empresas implementan una arquitectura de microservicios en sus plataformas, que busca desacoplar o independizar los componentes individuales de una aplicación en sí misma. Los microservicios utilizan APIs para poder comunicarse entre sí, debido al caso los desarrolladores pueden trabajar de manera independiente en cada microservicio, utilizando diferentes lenguajes de programación, permitiendo que la plataforma crezca de manera eficiente y sostenible, ya que cada microservicio puede desplegarse de forma autónoma. La mayor ventaja que tiene esta arquitectura es que si algún microservicio falla o se compromete el código, este no afecta el funcionamiento ni la estabilidad de los otros, y por consecuencia, tiene una mayor eficiencia en la disponibilidad de la plataforma.

Uno de estos microservicios es tiene por nombre Users, que se encarga de la autorización, autenticación, seguridad y manejos de los usuarios de la plataforma. Por tal motivo se diseñó y desarrolló un modelo para el manejo y seguridad de los usuarios mediante el uso del microservicio anteriormente mencionado que ayudó en la construcción de una plataforma No-Code.

1.2 Justificación

La empresa donde se desarrolló esta plataforma es una empresa que presta los servicios de University y Laboratory que tiene como objetivo enseñar y desarrollar tecnologías a partir de herramientas o plataformas No-Code. Para lograr este objetivo se utilizaban los servicios de terceros que generaba beneficios económicos para la empresa, pero no los suficientes para tener una estabilidad.

Por lo anterior, la empresa se veía en la necesidad de crear e implementar su propia herramienta o plataforma No-Code para dejar de depender de los servicios de terceros. Por tal motivo se vio en la tarea de contratar personas con conocimiento en el área para el desarrollo de esta propuesta.

En el proceso de investigación, se llegó a la conclusión que, debido a los requerimientos de la plataforma, se veía la necesidad de implementar la arquitectura denominada microservicios, ya que, gracias a esta nueva arquitectura, se puede dividir un sistema complejo en unidades independientes, donde cada uno cumple con una función particular, permitiendo evolucionar de manera más sencilla el crecimiento del software y por consecuente de la plataforma, también permite la separación de un microservicio ya existente cuando este crece más allá de sus capacidades en elementos más pequeños.

En el proceso de construcción del modelo de la plataforma me fue asignada la tarea de diseñar y desarrollar un modelo para el manejo y seguridad de los usuarios mediante el uso de la arquitectura de microservicios. El microservicio Users es necesario porque en cualquier plataforma se debe verificar la autenticidad de la persona por cuestión de seguridad para los beneficiarios, así como la seguridad de sus datos personales. Este proceso se realiza con Tokens de verificación y autenticación requeridos para el registro o inicio sesión del usuario, donde esta llave es única e irrepetible, que cambia dependiendo del lugar donde se accede y los datos de la persona, esta acción

permite que los interesados en acceder al sitio web tengan la certeza que sus datos personales y proyectos que lleven a cabo dentro de esta, estén totalmente a salvo de robos o intentos de sabotaje por parte de terceros y así garantizar la confiabilidad de los usuarios con la plataforma y a su vez la empresa que la desarrolla.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar y desarrollar un modelo para el manejo y seguridad de los usuarios mediante el uso de arquitectura de microservicios para la construcción de una plataforma No-Code.

1.3.2 Objetivos Específicos

Establecer el estado del arte del modelo para el manejo y seguridad de los usuarios mediante el uso de arquitectura de microservicios para la construcción de una plataforma No-Code.

Definir los requerimientos de la plataforma para el manejo y seguridad de los usuarios mediante el uso de arquitectura de microservicios para la construcción de una plataforma No-Code.

Validar el funcionamiento del modelo para el manejo y seguridad de los usuarios mediante el uso de arquitectura de microservicios para la construcción de una plataforma No-Code.

1.4 Acotaciones

No se utilizaron herramientas, tecnologías, base de datos y lenguajes diferentes a los que ha interpuesto la empresa para la construcción del microservicio, las cuales son: AWS (Amazon Web Services), JPA, Kotlin, Java, PostgreSQL, SpringBoot.

El trabajo realizado contribuye con la construcción de una plataforma No-Code, a través del diseño y desarrollo de un modelo para el manejo y seguridad de los usuarios, por lo tanto, no se mostrará el funcionamiento general de dicha plataforma.

Por lo tanto, por políticas de la empresa, el documento no contiene código fuente de la plataforma.

Este modelo fue validado en la empresa, de dos formas. La primera se llevó a cabo durante el desarrollo de la implementación del modelo en el microservicio users, donde se realizaron diferentes pruebas de rendimiento, estabilidad y calidad. La segunda y más importante se realizó antes del despliegue a producción, fue de gran importancia ya que estas pruebas la implementaron el equipo de QA de la empresa, donde verificaron el funcionamiento correcto del microservicio users, obteniendo como resultado el cumplimiento de los requerimientos hechos por la empresa para el manejo y seguridad de los usuarios.

El equipo de QA realizó otras pruebas internas del microservicio users y la infraestructura de la plataforma, donde solo se compartió el resultado, pero no el procedimiento.

2 Marco teórico

2.1 Estado del Arte

De acuerdo al primer objetivo del proyecto, se realizó un estado del arte que ayudó en el desarrollo del modelo para el manejo y seguridad de los usuarios. Por tal motivo, seguidamente se expondrá los libros, artículos, y demás referencias que contribuyeron en el logro de la meta.

2.1.1 Microservicios, La Clave Para Evolucionar Hacia Una Arquitectura Que Favorece La Innovación

Autores. Gerardo Barbosa

Año. 2021

Link. <https://www.ituser.es/opinion/2021/09/microservicios-la-clave-para-evolucionar-hacia-una-arquitectura-que-favorece-la-innovacion>

Resumen. De acuerdo con un estudio de Statista, el 45% de las empresas encuestadas asegura que en 2021 sus aplicaciones de análisis de datos e inteligencia de negocio utilizan microservicios. Por otra parte, según la consultora Markets and Markets, para 2023 este mercado moverá a 1.800 millones de dólares, lo que supone que crecerá a una tasa anual compuesta superior al 22%. (Barbosa, 2021)

De este post se toma la iniciativa que ayudó con la decisión de realizar una plataforma con estructura de microservicios, esto gracias a la estructuración y facilidad de implementarlo.

2.1.2 Microservices in the Enterprise, 2021

Autores. IBM Market Development & Insights

Año. 2021

Link. <https://www.ibm.com/cloud/learn/microservices>

Resumen. Recently, the IBM Market Development & Insights team conducted a series of surveys that recorded the perceptions and real-world experiences of microservices users and those considering adoption. These included more than 1,200 IT executives, developer executives and developers from large and midmarket companies that are currently using a microservices approach, as well as nonusers who are exploring or planning to adopt this approach in the near future. The results offer great insight into the real-world opportunities and challenges for implementing a microservices development approach. (Insights, 2021)

Abstract. Recientemente, el equipo de IBM Market Development & Insights llevó a cabo una serie de encuestas que registraron las percepciones y las experiencias del mundo real de los usuarios de microservicios y aquellos que están considerando la adopción. Estos incluyeron a más de 1200 ejecutivos de TI, ejecutivos de desarrollo y desarrolladores de empresas grandes y medianas que actualmente utilizan un enfoque de microservicios, así como a no usuarios que están explorando o planeando adoptar este enfoque en un futuro próximo. Los resultados ofrecen una gran comprensión de las oportunidades y los desafíos del mundo real para implementar un desarrollo de microservicios.

Con este libro se tiene un enfoque más claro de lo que es una estructura de microservicios, sus usos y el aporte significativo para las nuevas tecnologías.

2.1.3 Guía de Microservicios Punto de Vista

Autores. Roland Barcia (IBM), Kyle Brown (IBM), Richard Osowski (IBM)

Link. <https://www.ibm.com/downloads/cas/5O8YOPKN>

Resumen. Las aplicaciones de microservicios se componen de módulos independientes conectados en red, denominados microservicios. El estilo arquitectónico de microservicios es una evolución del estilo arquitectónico SOA (Arquitectura Orientada a Servicios). Las aplicaciones construidas usando servicios SOA se enfocan en problemas técnicos de integración y el nivel de los servicios implementados son a menudo interfaces específicas de programación de aplicaciones (API). Por el contrario, el enfoque de microservicios implementa capacidades comerciales claras a través de API comerciales generales.

Sirvió como base para la implementación de un modelo de microservicios en la plataforma No-Code.

2.1.4 Investigación y Desarrollo de la Arquitectura Orientada a Microservicios Para la Implementación de Módulos en un Sistema Web

Autores. Ricardo Muñoz

Año. 2018

Link. http://repobib.ubiobio.cl/jspui/bitstream/123456789/2672/1/Munoz_Hernandez_Ricardo.pdf

Resumen. Este documento plantea una nueva forma de desarrollo de sistemas, la cual está enfocada a la arquitectura orientada a microservicios. El objetivo de este informe es introducir al lector en el patrón de arquitectura, sus ventajas, características y además exponer el desarrollo e integración de nuevos módulos desarrollados bajo el patrón de arquitectura. Los conceptos definidos en este informe se demostrarán en el sistema informático llamado Zentown, de la consultora informática Zenta.

El informe tiene una estructura compuesta por los siguientes tópicos:

- Definición de la arquitectura monolítica.
- Definición de la arquitectura orientada a microservicios.

- Análisis y desarrollo de módulos bajo el patrón arquitectura orientado a microservicios.

Con la ayuda de esta tesis, se tuvo una idea más conveniente frente a la estructura del proyecto, además contribuyó en gran medida sobre el modelado de microservicios.

2.1.5 Introducción a la Seguridad Informática y el Análisis de Vulnerabilidades

Autores. Martha Irene Romero Castro, Grace Liliana Figueroa Morán, Denisse Soraya Vera Navarrete, José Efraín Álava Cruzatty, Galo Roberto Parrales Anzúles, Christian José Álava Mero, Ángel Leonardo Murillo Quimiz, Miriam Adriana Castillo Merino

Año. 2018

Link. <https://www.3ciencias.com/wp-content/uploads/2018/10/Seguridadinform%C3%A1tica.pdf>

Resumen. Este libro tiene como objetivo principal conocer los diversos conceptos de la seguridad informática, se estudiarán los mecanismos de prevención y correctivos de las amenazas que se puedan suscitar. Este trabajo de investigación está dirigido a estudiantes de informática, profesionales de la seguridad y docentes que incursionan en el mundo de la seguridad informática, ya que las herramientas descritas son usadas en varias áreas del conocimiento de la seguridad. El contenido de la obra en sus diferentes capítulos aporta conocimientos sobre los fundamentos de la ciberseguridad, conceptos sobre riesgos, amenazas y las diferentes vulnerabilidades que se pueden encontrar en la infraestructura de una organización y las posibles soluciones para mitigar estas amenazas.

Con este libro se logró entender la gran importancia sobre la seguridad informática. Fue un referente importante para la adopción de las tecnologías que se utilizaron para construir el modelo para el manejo y seguridad de los usuarios.

2.2 Referentes Teóricos

Antes de abordar los demás referentes teóricos, se expondrá un libro que da un mayor contexto sobre el uso de microservicios en el mundo de la tecnología.

2.2.1 *Microservicios un Enfoque Integrado*

Autores. David Roldán Martínez, Pedro J. Valderas Aranda, Victoria Torres Bosch

Año. 2020

Link. https://books.google.com.co/books?id=2o6fDwAAQBAJ&printsec=frontcover&dq=Microservicios+Un+enfoque+integrado&hl=es19&sa=X&redir_esc=y#v=onepage&q=Microservicios%20Un%20enfoque%20integrado&f=false

Resumen. Una de las tendencias más actuales en el desarrollo de aplicaciones informáticas son los microservicios. Este paradigma de programación se encuentra prácticamente por todas partes y compañías como Google, Netflix o Amazon, por citar algunas, se han enfocado en los microservicios desde hace ya algunos años.

Presenta una visión básica y aplicada sobre el desarrollo de aplicaciones web basadas en microservicios. Se incluye un ejemplo práctico completo, disponible en la nube, que el lector podrá ir construyendo desde cero a medida que va avanzando en su dominio de este tipo de aplicaciones.

Es, sin duda, una obra eminentemente práctica con explicaciones teóricas allá donde se ha estimado necesario. (David Roldán Maritinez, 2020)

De este libro se toma la decisión de realizar una plataforma con la estructura de microservicios, debido a los requerimientos obtenidos.

A continuación, se mostrará una ilustración que permite entender la estructura general de los referentes teóricos y cada uno de sus componentes.

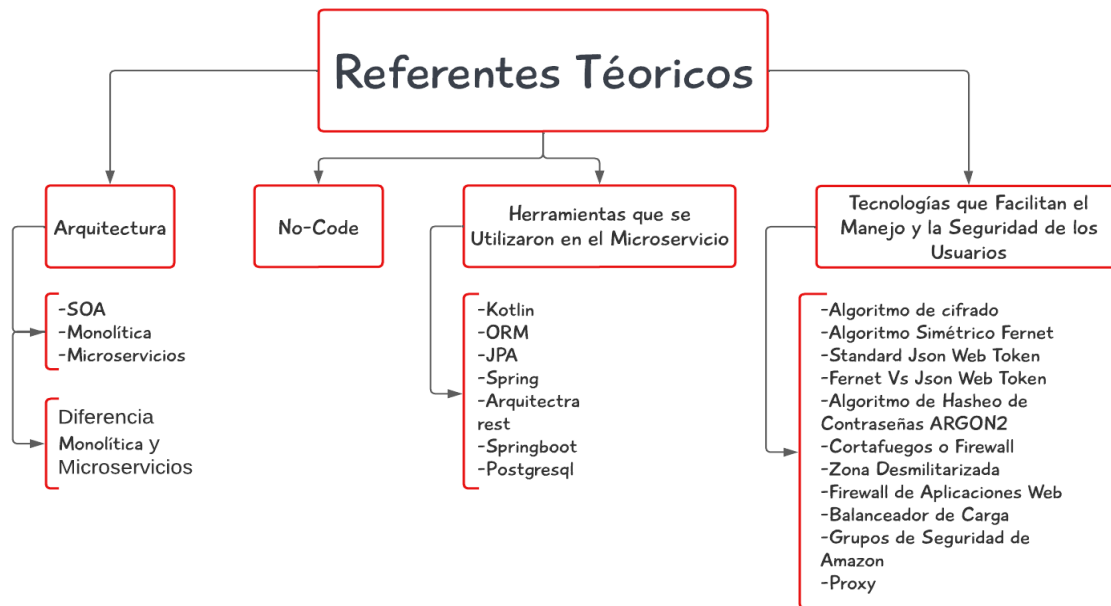


Ilustración 1 Referentes teóricos (Autoría propia)

En este ítem se explicarán algunas arquitecturas que han sido implementadas en los proyectos de desarrollo del software a lo largo de la historia. A continuación, se mostrará un anexo donde expone en orden cronológico el año de publicación de cada referente teórico.

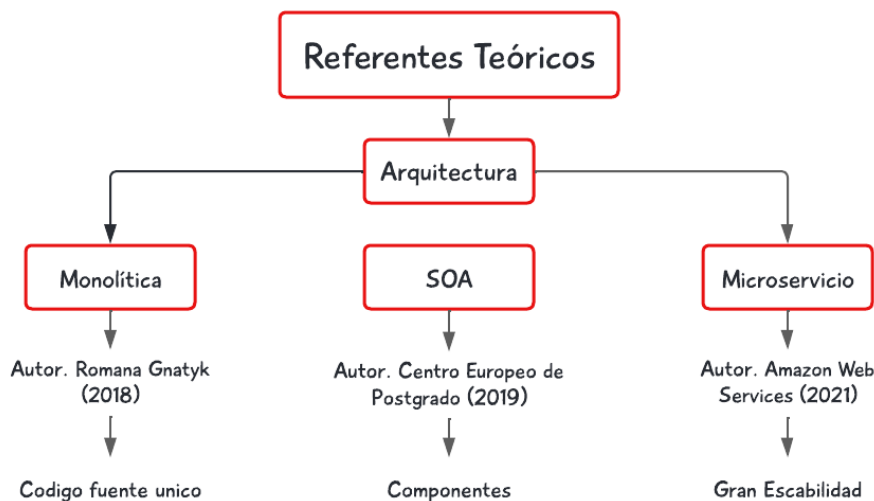


Ilustración 2 Arquitecturas (Autoría propia)

2.2.2 Arquitectura SOA

La arquitectura Orientada a Servicios representa un cambio en los sistemas que utilizan una plataforma de desarrollo que incluye muchos componentes, pero tiene sus propios elementos de diseño, con patrones de lenguaje, diseños arquitectónicos, conceptos, tecnología y el entorno del sistema asociado.

SOA propone un modelo arquitectónico que mejora la eficiencia, efectividad y productividad de un negocio, posicionando los servicios como lo primero que se debe considerar cuando se requiere presentar una lógica que resuelve un problema, o para lograr los objetivos estratégicos de la empresa, asociado a una programación orientada a servicios.

Por lo tanto, esta arquitectura SOA estará compuesta por la combinación de tecnología, producto, extensión Apis (Interfaz de aplicación relacional) de tecnología de asistencia y varias partes (Ceupe, 2019).

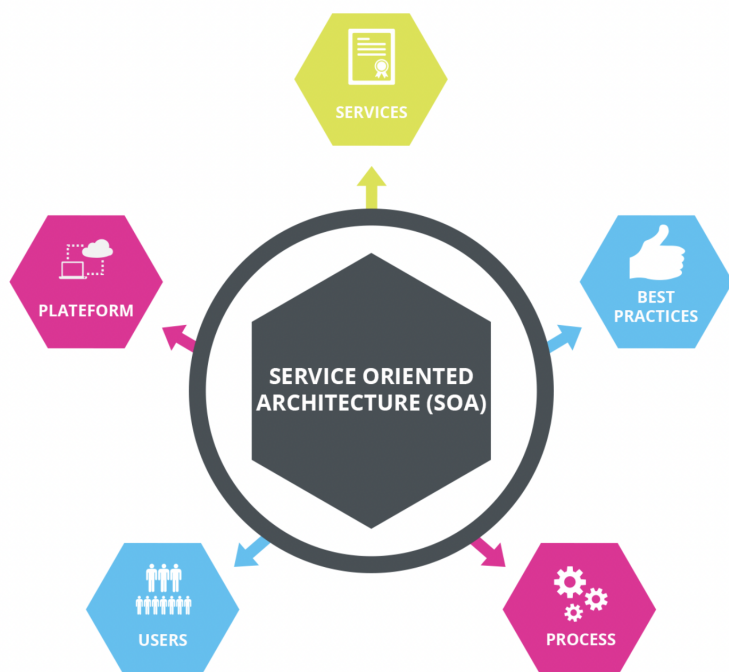


Ilustración 3 Arquitectura Soa

(Software Development Community, 2019)

Según el centro europeo de postgrado CEUPE (Ceupe, 2019), SOA define las siguientes capas de software:

- Aplicaciones básicas: sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier propiedad.
- Presentación de funciones: donde se exponen las funciones de la capa de aplicación como un servicio (generalmente un servicio web).
- Integración de servicios: Facilitan el intercambio de datos entre los elementos de la capa de aplicación hacia procesos de negocio internos o colaborativos.
- Componente de proceso: que define el proceso de acuerdo con la empresa y sus necesidades, y que varía de una empresa a otra.
- Despliegue: donde se implementan los servicios para los usuarios finales.

Por lo anterior se puede diferir que hay pocas diferencias entre la arquitectura SOA y la arquitectura basada en microservicios, puesto que las dos arquitecturas cumplen en su gran mayoría con los mismos objetivos. Según Juan Cantó integrante del centro de competencias digitales Castilla de la Mancha (Cantó, 2018), ninguna arquitectura supera a otra en calidad, la elección depende del objetivo y propósito del proyecto a desarrollar. En términos generales, SOA suele recomendarse para organizaciones grandes que trabajan a partir de la integración de aplicaciones. Los Microservicios se caracterizan por ser sistemas fraccionados y más pequeños que ofrecen un mayor control al desarrollador de software.

2.2.3 Arquitectura Monolítica

La arquitectura monolítica es un modelo de desarrollo unificado para aplicaciones de software. Tiene tres componentes:

- Interfaz de usuario del lado del cliente
- Aplicación del lado del servidor
- Interfaz de datos

Las tres partes interactúan con una sola base de datos. El software creado en este modelo trabaja con una base de código. Por lo tanto, cada vez que las partes involucradas desean realizar actualizaciones o cambios, acceden al mismo conjunto de código. Esto puede tener un efecto dominó que afecte el rendimiento del lado del usuario (Xplenty, 2021)

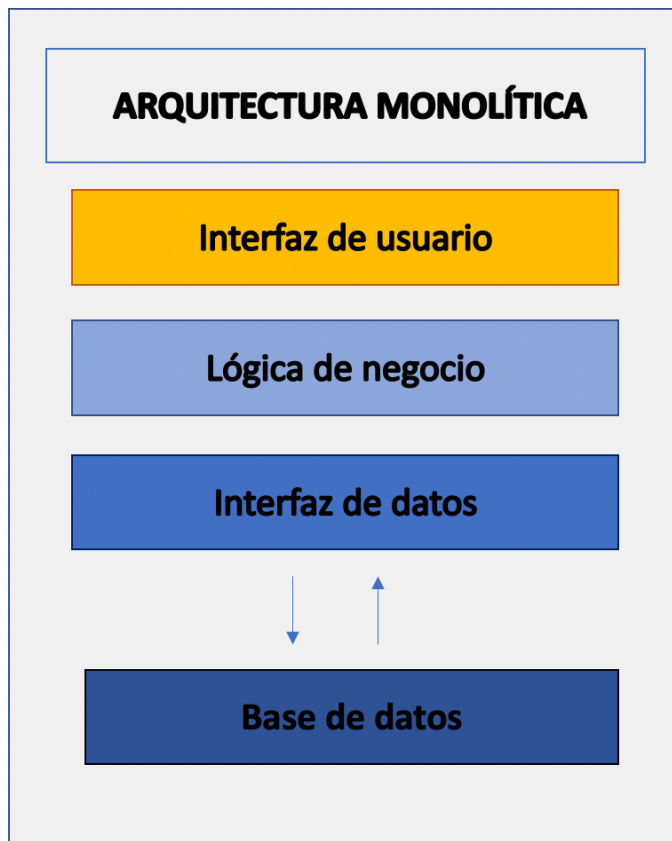


Ilustración 4 Arquitectura Monolítica (irvingjuarez, 2021)

2.2.3.1 Principales características de la arquitectura monolítica:

- Tiene un código fuente único.
- Se conecta a una sola base de datos.
- Debido a que todo el código está unificado, la comunicación entre sus funciones es más fácil.
- Para realizar cambios en la aplicación en desarrollo, es necesario realizar un análisis de todo el sistema, ya que se trata de una única fuente de código, el cambio realizado puede afectar a todo el sistema y generar errores en la aplicación.

Según Romana Gnatyk de la empresa N-iX LLC (empresa subcontratadora de desarrollo de software) (Gnatyk, 2018) las principales ventajas y desventajas de la arquitectura monolítica son:

2.2.3.2 Ventajas de la Arquitectura Monolítica:

- Menos preocupaciones transversales: Las preocupaciones transversales son las encargadas de afectar a toda la aplicación, como el almacenamiento en caché, el registro, el manejo, y la supervisión del rendimiento. En una aplicación monolítica, esta área de funcionalidad corresponde solo a una aplicación, lo que facilita su manejo.
- Depuración y prueba más fáciles: Comparado con la arquitectura de microservicios, las aplicaciones monolíticas son mucho más simples de probar y depurar. Debido a que es una sola unidad indivisible, puede implementar pruebas de un extremo a otro de manera eficaz.
- Sencillo de implementar: Una de las ventajas más importantes de esta arquitectura, es su sencillez, ya que permite construir sistemas sin tantas complicaciones de modelado e

implementación de código. Cuando hablamos de estructuras monolíticas, no tiene que administrar múltiples implementaciones, solo un archivo o directorio.

- Sencillo de desarrollar: Cualquier equipo de ingeniería tiene las habilidades y el conocimiento adecuado para desarrollar una aplicación monolítica. (Gnatyk, 2018)

2.2.3.3 Desventajas de la arquitectura monolítica

- Comprensión: Cuando este tipo de aplicaciones se amplía, se vuelve demasiado complicado de entender. Por otra parte, un sistema complejo de código dentro de una aplicación es difícil de manipular.
- Haciendo cambios: Es más difícil realizar cambios en una aplicación tan grande y compleja con un acoplamiento muy ajustado. El mínimo cambio de código repercute a todo el sistema, por lo que debe coordinarse minuciosamente. Esto provoca que el proceso de desarrollo general tarde mucho más tiempo.
- Escalabilidad: No puede escalar componentes de forma independiente, solo la aplicación completa.
- Barreras de nuevas tecnologías: Es extremadamente problemático aplicar una nueva tecnología en una aplicación monolítica porque entonces toda la aplicación tiene que ser reescrita. (Gnatyk, 2018)

2.2.4 Arquitectura de Microservicios

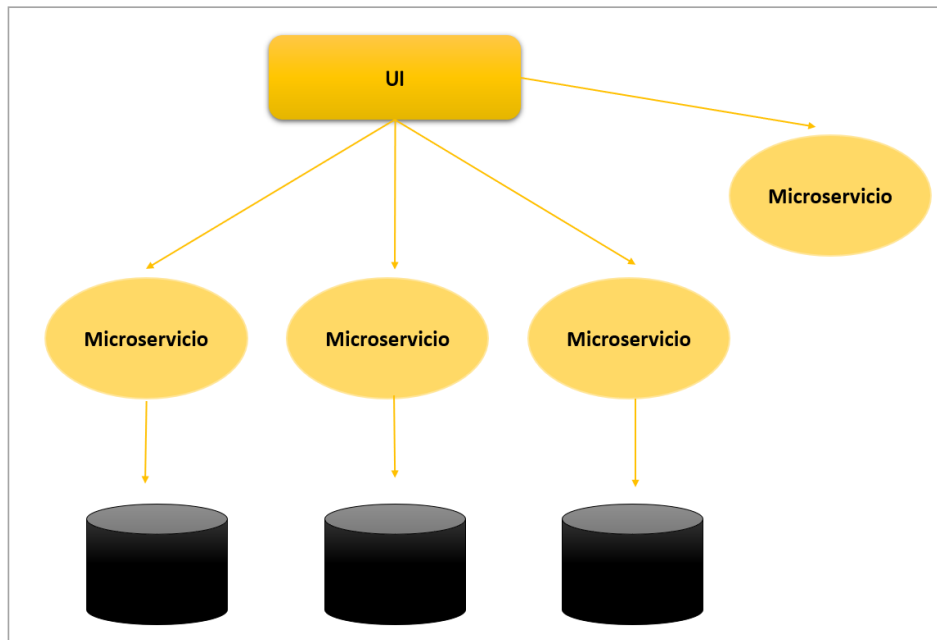


Ilustración 5 Arquitectura de Microservicios (irvingjuarez, 2021)

Según Amazon Web Services (Amazon Web Services, Inc., 2021), una de las empresas de tecnología líderes en el mundo, la arquitectura orientada a microservicios es un enfoque arquitectónico y organizativo para el desarrollo de software en el que el sistema se distribuye en componentes, un servicio pequeño e independiente que se comunica a través de API bien definidas. Los propietarios de estos servicios son pequeños grupos independientes. La arquitectura de Microservices ayuda a que las aplicaciones escalen y crezcan más rápido. Esto permite la innovación y acelera el tiempo de comercialización de nuevas funciones.

2.2.4.1 Principales Características de la Arquitectura de Microservicios

- Independiente: Cada parte presente en un microservicio se puede desarrollar, implementar, operar y extender sin afectar el funcionamiento de otros servicios. Los servicios tienen la necesidad de compartir su código o implementación con otros servicios. Toda la comunicación entre componentes individuales pasa por API definidas.

- Especializado: Cada servicio está diseñado para resolver problemas específicos, contruidos por un conjunto de competencias. Si el microservicio se vuelve extenso a lo largo del tiempo y su complejidad aumenta, se puede dividir en servicios más pequeños. (Amazon Web Services, Inc., 2021)

2.2.4.2 Beneficios de la Arquitectura de Microservicios

- Agilidad: Su estructura se basa en una organización de pequeñas partes independientes que se apropian de los servicios. Los equipos operan en un contexto ajustado, bien entendido y empoderado para trabajar de manera más independiente y rápida. Esto acorta el tiempo del ciclo de desarrollo. Se beneficia considerablemente de un mayor rendimiento organizacional.
- Escalabilidad flexible: Permiten que cada componente escale de forma independiente que permite satisfacer las necesidades de la funcionalidad de la aplicación. Esto permite a los desarrolladores saciar las necesidades de infraestructura, medir con precisión el costo de una función y mantener la disponibilidad si un servicio experimenta una mayor demanda.
- Fácil de implementar: Los microservicios permiten la integración y la entrega continuas, lo que facilita probar nuevas ideas y retroceder cuando surgen problemas. El bajo costo de los errores permite la experimentación, facilita la actualización de su código y hace que las nuevas características sean más rápidas en el mercado.
- Libertad tecnológica: La arquitectura de microservicios no sigue un enfoque de "talla única". Los equipos son libres de elegir la mejor herramienta para resolver sus problemas

particulares. Como resultado, los desarrolladores que construyen microservicios pueden elegir la mejor herramienta para cada tarea.

- **Código reutilizable:** La división del software en pequeños fragmentos y bien estructurados, que permite a los equipos utilizar funciones para diferentes propósitos. Un servicio escrito para una característica se puede utilizar como componente básico de otra característica. Esto permite que una aplicación sea más ordenada y llevadera, ya que los desarrolladores pueden crear nuevas funciones sin tener que escribir código desde cero.
- **Resiliencia:** La independencia del servicio aumenta la tolerancia a fallos de una aplicación. En una arquitectura monolítica, la falla de un solo componente puede hacer que falle toda la aplicación. Si ocurre un error en todo el servicio, no todos sus componentes se verían afectados, por tal motivo los desarrolladores se enfocarán en arreglar estos problemas sin preocuparse que todo el servicio falle. (Amazon Web Services, Inc., 2021)

2.2.4.3 Debilidades de la Arquitectura de Microservicios

- **Más complejidad:** Debido a que la arquitectura de microservicios es un sistema distribuido, debe seleccionar y configurar las conexiones entre todos los módulos y la base de datos. Además, siempre que la aplicación incluya servicios separados, todos deben implementarse de forma independiente.
- **Disposición del sistema:** La arquitectura de microservicios es un sistema complejo formado por muchos módulos y bases de datos, por lo que todas las conexiones deben ser tratadas con cuidado.

- Preocupaciones transversales: Al crear una aplicación de microservicios, se enfrentará a una serie de problemas transversales. Estos incluyen configuración subcontratada, registro, métricas, comprobaciones de estado y más.
- Experimento: La multitud de componentes implementables de forma independiente hace que probar una solución basada en microservicios sea mucho más difícil. (Gnatyk, 2018)

2.2.5 Diferencias Entre la Arquitectura Monolítica y la Arquitectura de Microservicios

Microservicio vs monolítico	
#1. Lenguaje	
Microservicio	Monolítico
Cada servicio se puede desarrollar de forma independiente utilizando diferentes lenguajes de programación.	Completamente desarrollado en un solo lenguaje de programación.
#2. Código Base	
Microservicio	Monolítico
Hay varias bases de código y cada servicio tiene una base de código separada	Solo una base de código
#3. Comprensibilidad	

Microservicio	Monolítico
Gran claridad y fácil mantenimiento	Es muy confuso y difícil de entender
#4. Escalado de la aplicación	
Microservicio	Monolítico
El escalado de la aplicación es simple, ya que cada servicio se puede aplicar individualmente sin aplicar la aplicación completa.	Escalar la aplicación es muy difícil porque es necesario escalar toda la aplicación.
#5 Desarrollo y despliegue	
Microservicio	Monolítico
El desarrollo y el despliegue continuos son posibles.	El desarrollo y la implementación del código continuo es muy complicado.
#6 Inicio del servicio	
Microservicio	Monolítico
Inicio rápido del servicio.	Puesta en marcha del servicio que lleva tiempo.
#7 Modelo de datos	
Microservicio	Monolítico

Tiene un modelo de datos federado, que permite que cada servicio adopte su propio modelo de datos.	Tiene un modelo de datos centralizado.
#8 Consistencia y disponibilidad	
Microservicio	Monolítico
Altamente consistente y fácilmente disponible.	Relativamente inconsistente y disponible porque cualquier actualización requeriría desarrollo desde el principio.

Tabla 1 Diferencias entre arquitectura de microservicios y monolítica (Cantó, 2018)

2.3 No-Code

De acuerdo con Félix Gonzalo (Gonzalo, 2021), que hace parte del startup llamado nocoders ACADEMY, El No-Code es una filosofía digital cuyo propósito es facilitar que cualquier persona tenga acceso a herramientas que permitan la creación de aplicaciones y software sin saber codificar.

Esto se llama programación sin código o programación visual. Es decir, distribuir con lenguajes de codificación para que la creación sea un proceso más intuitivo y sencillo.

Para crear una aplicación sin programación, se utilizan varias herramientas de desarrollo para proporcionar un entorno visual.

Ejemplos de plataformas especializadas en No-Code: Bubble, AppGyver, Adalo, Zapier, Webflow.

2.4 Herramientas que se Utilizaron en el Microservicio

En esta sección se hizo un breve resumen de las herramientas que normalmente se utilizan para la construcción de microservicios con el lenguaje Kotlin. A continuación, se mostrará un anexo donde expone en orden cronológico el año de publicación de cada referente teórico.

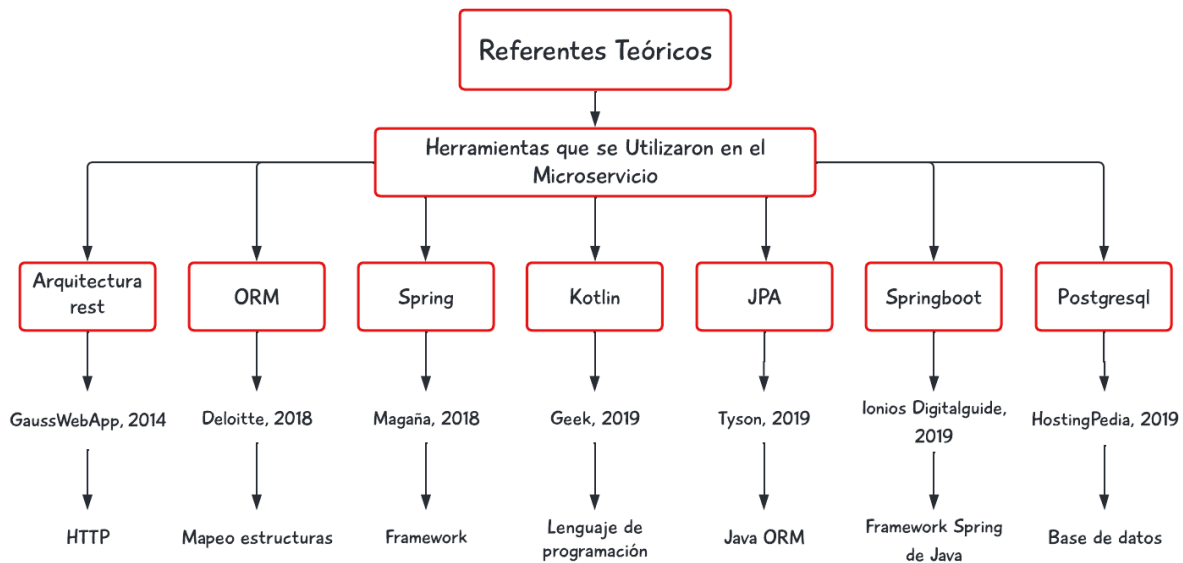


Ilustración 6 Herramientas que se utilizaron en el microservicio (Autoría propia)

2.4.1 Kotlin

Es un lenguaje de programación práctico diseñado para funcionar con Máquina Virtual de Java (JVM) y Android. Alternativamente, se puede compilar a código fuente de JavaScript.

Se destaca por una perfecta combinación de características de manera evidente orientadas a la funcionalidad durante la programación, centrándose en la seguridad, la claridad y la interoperabilidad.

Los programas en Kotlin pueden recurrir a los frameworks y librerías de Java existentes. También, su interoperabilidad no requiere capas de adaptación. (Geek, 2019)

2.4.1.1 Ventajas y Desventajas de Kotlin

Ventajas	Desventajas
Ofrece todas las características de un lenguaje de estilo orientado a objetos con un enfoque funcional	Coincidencia de patrones débil
Funciones de orden superior, son aquellas que toman una función como argumento y tipo de retorno	Tamaño extra de tiempo de ejecución
La codificación en Kotlin es muy corta, concisa y directa. Mismo trabajo con menos líneas de código que en Java	Legibilidad inicial del código se vuelve un poco difícil y entender al principio
Respaldo por JetBrains, Google y Spring Framework, definitivamente está aquí para quedarse, crecer y evolucionar	Comunidad de soporte pequeña, Kotlin es más joven que Scala, tiene menos bibliotecas, publicaciones en blogs y tutoriales

Tabla 2 Ventajas y desventajas de Kotlin (Fernandez, 2018)

2.4.2 Mapeo Objeto-Relacional (ORM)

Es un modelo de programación que facilita el mapeo de las estructuras de base de datos relacionales (mysql, postgresql, oracle, entre otros), en adelante RDBMS (relational database management

system), sobre la distribución lógica de entidades para simplificar y acelerar el desarrollo de nuestras aplicaciones. (Deloitte, 2018)

2.4.3 Api de Persistencia de Java (JPA)

El estándar Java ORM para acceder, almacenar y administrar objetos Java en una base de datos relacional, se encarga de la persistencia, que en contexto generales significa cualquier mecanismo por el cual los objetos de Java soportan el proceso de aplicación que los creó. No es necesario mantener todos los objetos Java, pero la mayoría de las aplicaciones conservan los objetos comerciales clave. La especificación JPA admite establecer qué objetos deben persistir y cómo deben persistir esos objetos en sus aplicaciones Java.

Por sí mismo, JPA no es una herramienta o un marco; más bien, define un conjunto de conceptos que pueden ser implementados por cualquier herramienta o marco. Aunque el modelo de mapeo relacional de objetos (ORM) de JPA se fundamenta originalmente en Hibernate, es evidente que ha evolucionado. De igual manera, aunque JPA se creó inicialmente para su uso con bases de datos relacionales / SQL, algunas implementaciones de JPA se han ampliado para su uso con almacenes de datos NoSQL. Un marco que admite JPA con NoSQL es EclipseLink, la implementación de referencia para JPA 2.2. (Tyson, 2019)

2.4.4 Spring

Es un Framework de código abierto para crear aplicaciones empresariales Java, con soporte para Groovy y Kotlin. Tiene una estructura modular y gran flexibilidad para implementar diferentes tipos de arquitecturas en función de las necesidades de la aplicación. (Magaña, 2018)

Estructura Modular de Spring: No es solo un framework de inyección de dependencias, es una familia completa de proyectos que cubren muchas áreas: desarrollo de aplicaciones web, aplicaciones web receptivas, seguridad, servicios web, microservicios, Android, etc.

Además, en algunos de estos proyectos ver que tiene una estructura modular, es decir, pretende poder tener diferentes módulos agrupando diferentes funciones, programación orientada a

aspectos, acceso a datos, entre otros. Por lo tanto, permite una mejor organización de los proyectos o microservicios que utilicen dicha tecnología. (Magaña, 2018)

Flexibilidad de Spring: Según Luis Magaña (Magaña, 2018), Spring tiene la flexibilidad de:

Aplicaciones de escritorio, aplicaciones de línea de comandos, aplicaciones web clásicas, web receptiva, microservicios.

Las aplicaciones acceden a la base de datos a través de SQL directo, a través de algún tipo de base de datos ORM, NoSQL.

Aplicaciones con el clásico esquema de seguridad donde almacenamos elementos de seguridad, claves o identificadores en nuestra base de datos, pero también con otros sistemas como OAuth 2.0, JWT, LDAP.

Aplicaciones pequeñas, medianas y grandes, aplicaciones que tienden a evolucionar, aplicaciones que se gestionan a través de contenedores, aplicaciones que necesitan desplegarse en la nube.

2.4.5 Arquitectura rest

Usualmente esta arquitectura está presente en muchos modelos de microservicios, puesto que los microservicios se comunican entre sí por medio de llamados HTTP, por tal motivo es importante conocer el concepto y algunos tips para implementarlo.

Rest es una arquitectura de desarrollo web que se puede utilizar en cualquier cliente HTTP. Además, es mucho más sencillo que otras arquitecturas existentes, como XMLRPC o SOAP. Esta simplicidad se logra porque utiliza una interfaz web que usa hipermedia para representar y transformar información.

Esta arquitectura fue iniciada por Roy Fielding en 2000, quien ayudo también en la especificación del protocolo HTTP. (GaussWebApp, 2014)

La principal ventaja de esta arquitectura es que ha aportado mayor escalabilidad a la web, es decir, soporta una mayor cantidad de componentes y sus interacciones. Esta ventaja se debe a una serie de características que presenta la arquitectura REST:

Este es un protocolo sin estado porque la información no se almacena en el servidor. Es decir, toda la información será enviada por el cliente en cada mensaje HTTP, lo que guarda las variables de sesión y el almacenamiento interno del servidor.

Presenta un conjunto bien definido de operaciones, principalmente GET, POST, PUT y DELETE, que se utilizan en todos los recursos.

Utilice URI únicos de acuerdo con una sintaxis común. Estos URI están sujetos a ciertas pautas que visualizaremos más adelante.

Utilice meta media para representar información, normalmente HTML, XML o JSON.

(GaussWebApp, 2014)

Para la arquitectura Rest, los recursos son uno de los elementos más importantes. Estos datos se identifican mediante un URI (Identificador de recurso único), que debe tener las siguientes características:

Las URI no pueden implicar una acción en su nombre, es decir, se debe evitar colocar verbos en ellas:

Ejemplo:

https://...../user/1092019/borrar **Incorrecto**

https://...../user/1092019 **Correcto**

Deben ser únicos, no se puede tener más de una URI para identificar el mismo recurso.

Deben ser independientes del formato, es decir, no deben representar ninguna extensión:

Ejemplo:

https://...../user/1092019.json **Incorrecto**

https://...../user/1092019 **Correcto**

Deben mantener una jerarquía lógica. La jerarquía es el criterio por el cual se ordenan los elementos. El siguiente ejemplo muestra la diferencia:

Ejemplo:

`https://...../info/user/1092019` **Incorrecto**

`https://...../user/1092019/info` **Correcto**

En el caso incorrecto se podría leer como «info contiene al user 1092019», situación poco lógica. En cambio, en el caso correcto, podríamos leerlo como «usuario 1092019 contiene info», algo que suena más lógico.

Los filtrados de información de un recurso no se hacen en la URI:

Ejemplo:

`https://...../user/name/Joan` **Incorrecto**

`https://...../user?name=Joan` **Correcto**

En este caso se utilizaría un RequestParam para especificar el valor.

(GaussWebApp, 2014)

Por lo anterior, es bien sabido que la arquitectura REST está basada en recursos, donde cada recurso se identifica de manera única.

Para interactuar con estos recursos, se utilizan principalmente los siguientes métodos HTML:

GET: permite el acceso a varios recursos.

POST: se utiliza para realizar acciones para crear nuevos recursos.

PUT: se utiliza para actualizar los datos actuales.

DELETE: se utiliza para eliminar recursos.

(GaussWebApp, 2014)

2.4.6 Springboot

SpringBoot, lanzado en 2012, es una solución para el framework Spring de Java que se rige por el principio de “convención sobre configuración” y minimiza la complejidad del desarrollo de nuevos proyectos basados en Spring. Para ello, SpringBoot suministra la estructura básica configurada del proyecto, que contiene las pautas para usar el marco y cada una de las bibliotecas de terceros relevantes para la aplicación, lo que nos prepara el camino para comenzar a desarrollar de manera inmediata. De esta manera se facilita la creación de aplicaciones independientes y reproducibles, por lo que muchas de las nuevas aplicaciones fundamentadas en Spring se desarrollan con SpringBoot. (Ionios Digitalguide, 2019)

Según la guía digital IONOS (Ionios Digitalguide, 2019) las características de SpringBoot se pueden resumir de la siguiente manera:

Incorporación directa de aplicaciones de servidores web/contenedores como Apache Tomcat o Jetty, excluyendo la necesidad de adicionar archivos WAR (Web Application Archive).

Springboot Configura automáticamente spring siempre que sea posible.

Características no funcionales, como configuraciones externalizadas o métricas.

2.4.7 Postgresql

Es un gestor de bases de datos relacional y orientado a objetos. (HostingPedia, 2019)

2.5 Tecnologías que Facilitan el Manejo y la Seguridad de los Usuarios

Desde hace algunos años, la vida cotidiana de los seres humanos ha tomado un cambio significativo en todos sus aspectos. Todos estos cambios se han visto por la llegada del internet y por consecuencia el aumento del uso que le damos a las TIC (Tecnologías de información y comunicación).

Según el congreso de la república de Colombia (Art. 6 Ley 1341 de 2009), las tecnologías de la información y la comunicación (TIC) son un conjunto de recursos, herramientas, dispositivos, programas informáticos, aplicaciones, redes e instalaciones; permite recopilar, procesar, almacenar y transmitir información como: voz, datos, texto, video e imágenes.

El uso de estas tecnologías ha traído beneficios para la mayor parte de la sociedad, pues debido a estas, se puede acceder a la información de una forma fácil y gratuita. Pero no todo son beneficios, ya que al tratarse de una tecnología que opera con la información, también lo hace con la de los usuarios que quieren acceder a ella, de esta manera se crea una grieta en la seguridad de los usuarios, pues, cualquier persona con los conocimientos necesarios podría acceder a ella sin el consentimiento de la persona afectada.

Por el problema anterior, nace un tema indispensable en el manejo de las TIC, ese tema es la “Seguridad Informática” que, según UNIR (Unir, 2021) la Universidad en internet, La seguridad informática se refiere a la retención de información y, en particular, al procesamiento de información para evitar la manipulación de datos y procesos por personas no autorizadas. Su objetivo principal es que las personas como la tecnología y los equipos de datos estén protegidas contra los males y amenazas desarrollados por terceros.

Este tema está cada vez más presente en las empresas informáticas que prestan servicios de acuerdo al asunto, debido a que los usuarios que ingresan a estas herramientas, buscan que la seguridad y la integridad de sus datos personales sean protegidos con la mayor seguridad.

Debido a la gran importancia de garantizar la seguridad de los datos de los usuarios, las empresas informáticas que prestan servicios TIC implementan algoritmos de cifrado especializados en brindar una buena seguridad a los usuarios.

A continuación, se mostrará un anexo donde expone en orden cronológico el año de publicación de cada referente teórico.

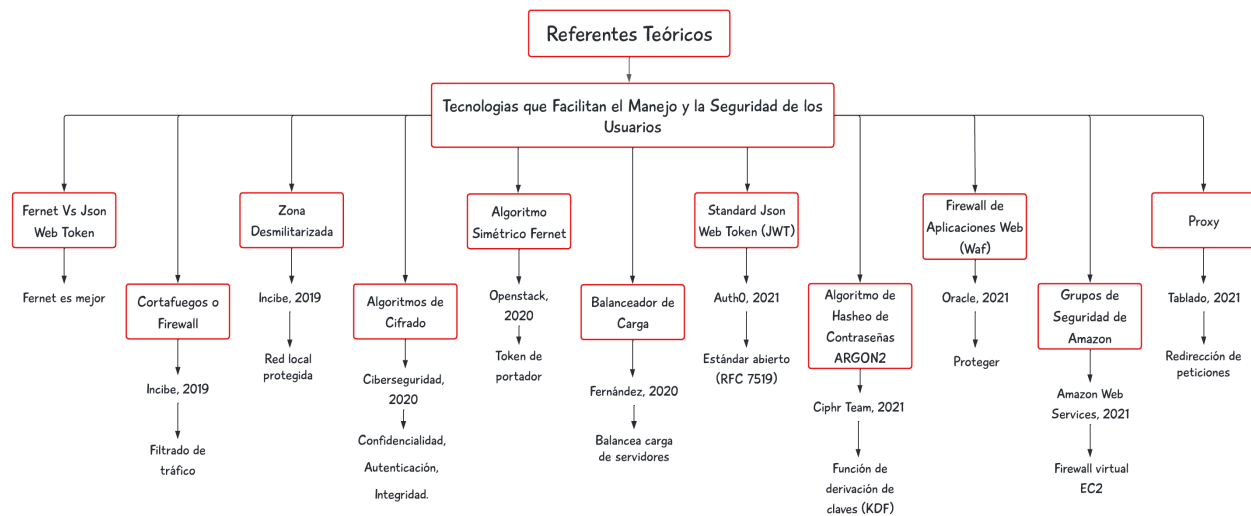


Ilustración 7 Tecnologías que facilitan el manejo y seguridad de los usuarios (Autoría propia)

2.5.1 Algoritmos de Cifrado

Según la página especializada en Ciberseguridad [ciberseguridad.com](https://www.ciberseguridad.com) (Ciberseguridad, 2020) un algoritmo de cifrado es un componente para el equilibrio del transporte electrónico de datos. Los pasos lógicos matemáticos reales se toman y se enlistan al desarrollar algoritmos para objetivos de encriptación, y se usan diferentes cifrados de bloque para encriptar datos o números electrónicos.

Los algoritmos de cifrado se encargan de prevenir la falsificación de datos, como el perpetrado por piratas informáticos que obtienen ilegalmente información financiera electrónica. Estos algoritmos son parte de los protocolos de gestión de riesgos de cualquier organización.

Ayudan en el proceso de transformación de escrito sin formato en escrito cifrado y luego de vuelta a escrito sin formato con el fin de proteger los datos electrónicos cuando se transportan mediante redes. Al codificar o cifrar datos, los hackers u otros usuarios no autorizados primordialmente no pueden entrar a dicha información.

Además, la página [ciberseguridad.com](https://www.ciberseguridad.com) (Ciberseguridad, 2020) menciona algunas de las funciones de los algoritmos de cifrado:

- Confidencialidad: Cifra el contenido del mensaje.
- Autenticación: Comprueba el origen del mensaje.
- Integridad: Prueba que el contenido del mensaje no ha cambiado desde que se envió.
- No repudio: Evita que los remitentes nieguen que enviaron el mensaje cifrado.

2.5.1.1 Cifrado Simétrico

En resumen, el cifrado simétrico o también llamado como algoritmo de cifrado de clave secreta, es un algoritmo que usa una clave única para cifrar y descifrar. Las dos partes que se comunican, deben utilizar la misma clave, ya que el remitente al cifrar y enviar un mensaje, el destinatario solo puede descifrar dicho mensaje con la misma clave con la que el destinatario lo cifró. (Fernandez M., 2020)

Según Peter Smirmoff (Peter Smirmoff, 2019) existen dos tipos de cifrado simétrico:

- Algoritmo de bloque: Las longitudes de bits establecidas se cifran en bloques de datos electrónicos utilizando una clave secreta específica. Dado que los datos están encriptados, el sistema mantiene los datos en su memoria mientras espera que los bloques se llenen.
- Algoritmos de transmisión: Los datos se cifran a medida que se transmiten en lugar de almacenarse en la memoria del sistema.

2.5.1.2 Cifrado Asimétrico

Este algoritmo es una alternativa para el algoritmo simétrico, pues el algoritmo asimétrico contiene una llave publica, de esta manera la distribución de las claves es más fácil y segura, dejando la llave privada de uso exclusivo para el dueño. Pero, existen inconvenientes al utilizar esta clase de algoritmo, pues, se utiliza un mayor tiempo en el proceso de encriptación, como también se aumenta el costo de utilización, porque las claves son de mayor tamaño que las simétricas. Su funcionamiento se basa en que utiliza dos claves diferentes pero relacionadas. La primera llave, la

llave pública, se utiliza para el cifrado y la otra, la clave privada, para el descifrado. Como se indica en el nombre, la clave privada debe ser privada para que solo el destinatario autenticado pueda descifrar el mensaje. (Savvy Security, 2021)

2.5.1.3 Diferencia Entre Algoritmo Simétrico y Asimétrico

Cifrado simétrico	Cifrado asimétrico
El cifrado simétrico consta de una clave para el cifrado y el descifrado.	El cifrado asimétrico consta de dos claves criptográficas conocidas como clave pública y clave privada.
el cifrado simétrico es mucho más rápido en comparación con el método asimétrico	El cifrado asimétrico incorpora dos claves independientes, el proceso se ralentiza considerablemente.
RC4	RSA
AES	Diffie-Hellman
DES	ECC
3DES	El Gamal
QUAD	DSA

Ilustración 8 Diferencias entre algoritmo simétrico y asimétrico (Savvy Security, 2021)

2.5.2 Algoritmo Simétrico Fernet

Para poder entender el concepto de Fernet y sus funcionalidades, se debe explicar brevemente los conceptos derivados de este.

¿Qué es un Token Fernet? Es un token de portador que representa la autenticación del usuario. Los tokens de Fernet contienen una cantidad limitada de datos de identidad y autorización en una carga útil MessagePacked. Luego, la carga útil se envuelve como un mensaje de Fernet para el transporte, donde Fernet proporciona las características de seguridad web necesarias para su uso en URL y encabezados. Los datos dentro de un token fernet están protegidos mediante claves de cifrado simétricas o claves fernet. (Openstack, 2020)

¿Qué Es Una Llave Fernet? Se requiere un repositorio de claves para crear tokens fernet. Estas claves se utilizan para cifrar y descifrar la información que forma la carga útil del token. Cada clave de repositorio puede tener uno de tres estados. El estado de la clave determina cómo el keystone usa una clave con tokens fernet. De acuerdo con la página web openstack.com (Openstack, 2020), los diferentes tipos son los siguientes:

Clave principal: Solo hay una clave principal en un depósito de claves. La clave principal puede cifrar y descifrar tokens. Esta clave siempre se denomina como el índice más alto del repositorio.

Clave secundaria: Una clave secundaria fue una vez una clave principal, pero se ha degradado en lugar de otra clave principal. Solo se pueden descifrar los tokens. Dado que en un momento fue el principal, se justifica su existencia en el repositorio de claves. Keystone debe poder descifrar los tokens creados con las claves primarias antiguas.

Staged Key: Es una clave especial que comparte algunas similitudes con las claves secundarias. Solo puede haber una clave preparada en un repositorio y debe existir. Al igual que las claves secundarias, las claves de fase tienen la capacidad de descifrar tokens. A diferencia de las claves secundarias, las claves de fase nunca han sido una clave principal. En realidad, son opuestos ya que la clave por etapas siempre será la siguiente clave primaria. Esto ayuda a aclarar el nombre porque son la siguiente clave preparada para ser la clave principal. Esta clave siempre se denomina 0 en el almacén de claves.

2.5.2.1 Fernet Y Sus Principales Funciones

Es importante saber que Fernet hace parte de una librería llamada cryptography (Cryptography, 2021), esta incluye recetas de alto nivel e interfaces de bajo nivel para algoritmos criptográficos comunes, como cifrados simétricos, resúmenes de mensajes y funciones de derivación de claves. A continuación, se dará una explicación de Fernet y sus funcionalidades principales.

Fernet garantiza que un mensaje cifrado con él no se puede manipular ni leer sin la clave. Fernet es una implementación de la criptografía de autenticación simétrica (también conocida como "clave secreta"). Fernet también admite la rotación de claves a través de MultiFernet.

Usando la clase `cryptography.fernet.Fernet`

```
>>> from cryptography.fernet import Fernet
>>> key = Fernet.generate_key()
>>> f = Fernet(key)
>>> token = f.encrypt(b"my deep dark secret")
>>> token
b'...'
>>> f.decrypt(token)
b'my deep dark secret'
```

Ilustración 9 Usando la clase `cryptography.fernet` (Cryptography, 2021)

Esta clase permite cifrar y descifrar data.

Parámetros	Retorno	Errores	Descripción
Key (bytes o str)			una clave de 32 bytes codificada en base64 segura para URL. Esto debe mantenerse en

			secreto. Cualquiera con esta clave puede crear y leer mensajes.
--	--	--	---

Tabla 3 Descripción de la clase cryptography.fernet (Cryptography, 2021)

Métodos de la clase cryptography.fernet.fernet:

generate_key ():

Parámetros	Retorno	Errores	Descripción
()	Clave Fernet		Este método genera una clave Fernet nueva, por seguridad esta clave se debe mantener bien segura, ya que con esta clave se descifran los mensajes. Llegando a las manos equivocadas crearía una brecha de seguridad incalculable.

Tabla 4 método generate_key () de Fernet (Cryptography, 2021)

encrypt(data):

Parámetros	Retorno	Errores	Descripción
Data (Bytes)	Token Fernet	TypeError (Se genera si los datos no son bytes)	Cifra la data, sólidas garantías de privacidad y autenticidad.

Tabla 5 método encrypt (data) de Fernet (Cryptography, 2021)

encrypt_at_time (data, current_time)

Parámetros	Retorno	Errores	Descripción
data (Bytes), current_time(int)	Token Fernet	TypeError (Se genera si la data no son bytes o si current_time no es int)	Cifra los datos transmitidos utilizando la hora actual transmitida explícitamente.

Tabla 6 método encrypt_at_time (data) de Fernet (Cryptography, 2021)

decrypt (token, ttl=None)

Parámetros	Retorno	Errores	Descripción
token(bytes), ttl(int)	La data original	cryptography.fernet.InvalidToken (Si el token no es válido), TypeError (Si el token no es de tipo bytes)	Descifra un token Fernet. Si se descifra correctamente, recibirá la data como resultado. Es seguro utilizar estos datos de inmediato, ya que Fernet verifica que los datos no hayan sido manipulados antes de devolverlos.

Tabla 7 método decrypt (token, ttl) de Fernet (Cryptography, 2021)

decrypt_at_time (token, ttl, current_time)

Parámetros	Retorno	Errores	Descripción
token(bytes), ttl(int)	La data original	cryptography.fernet.InvalidToken (Si el token no es válido), TypeError (Si el token no es de tipo bytes)	Descifra un Token de Fernet utilizando la hora actual.

Tabla 8 método decrypt_at_time (token, ttl, current_time) de Fernet (Cryptography, 2021)

Extract_timestamp (token)

Parámetros	Retorno	Errores	Descripción
token(bytes)	La hora de caducación del Token	cryptography.fernet.InvalidToken (Si el token no es válido), TypeError (Si el token no es de tipo bytes)	Este método se usa para saber exactamente la hora de caducación de un Token

Tabla 9 método extract_timestamp (token) (Cryptography, 2021)

2.5.2.2 Usando Contraseñas con Fernet

Para hacer esto, necesita ejecutar la contraseña a través de una función de derivación de clave como PBKDF2HMAC, bcrypt o Scrypt. (Cryptography, 2021)

Ejemplo de procedimiento:

```
>>> import base64
>>> import os
>>> from cryptography.fernet import Fernet
>>> from cryptography.hazmat.primitives import hashes
>>> from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
>>> password = b"password"
>>> salt = os.urandom(16)
>>> kdf = PBKDF2HMAC(
...     algorithm=hashes.SHA256(),
...     length=32,
...     salt=salt,
...     iterations=100000,
... )
>>> key = base64.urlsafe_b64encode(kdf.derive(password))
>>> f = Fernet(key)
>>> token = f.encrypt(b"Secret message!")
>>> token
b'...'
>>> f.decrypt(token)
b'Secret message!'
```

Ilustración 10 contraseñas con Fernet (Cryptography, 2021)

2.5.2.3 Limitaciones de Fernet

Fernet es ideal para cifrar datos que se pueden almacenar fácilmente en la memoria. Como característica de diseño, no expone bytes no autenticados. Esto significa que todo el contenido del mensaje debe estar disponible en la memoria, por lo que Fernet generalmente no es adecuado para archivos muy grandes en este momento. (Cryptography, 2021)

2.5.3 Standard Json Web Token (JWT)

Según auth0 (auth0, 2021) una empresa argentina especializada en seguridad, JWT es un estándar abierto (RFC 7519) que define una forma compacta e independiente de transmitir información de forma segura entre las partes como un objeto JSON. Esta data se puede admitir porque está firmada digitalmente. Este se puede firmar aumentando su seguridad con el algoritmo HMAC o un par de claves pública / privada usando RSA o ECDSA.

JWT se puede utilizar para los siguientes escenarios:

Autorización: Es donde más se utiliza JWT, ya que, al iniciar sesión, el Token quedara expuesto para que las siguientes solicitudes del usuario estén permitidas en todo el sistema.

Intercambio de información: Como los JWT pueden ser firmados, esto permite autenticar la identidad del remitente, también permite verificar que el contenido del token no haya sido manipulado.

Los JWT tienen una estructura compacta separada por (.):

- Encabezado (aaaa)
- Carga útil (bbbb)
- Firma (cccc)

Dando como resultado la siguiente estructura:

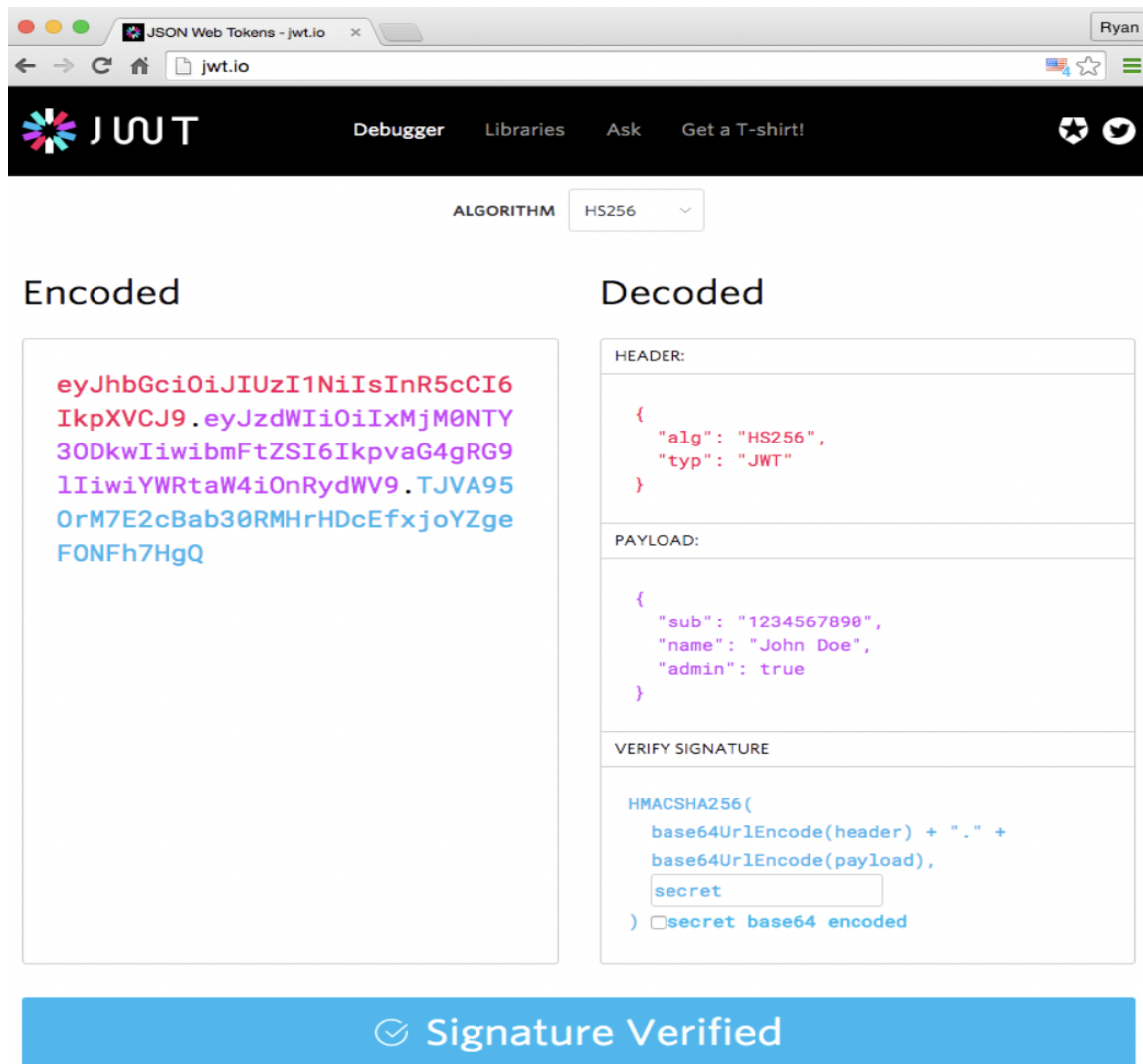
- aaaa.bbbb.cccc

Cuyo resultado es un conjunto de cadenas de URL Base64. A continuación se mostrará un ejemplo de un JWT que está firmado:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4  
gRG9lIiwiaXNjb2NpYWwiOiJhbnRydWV9.  
4pcPyMD09o1PSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

Ilustración 11 token JWT firmado

Para poder ver el contenido de cada uno de los elementos de la estructura, se puede tomar el token y llevarlo a la página oficial de JWT (jwt.io). A continuación, se mostrará un ejemplo con el resultado del Token anterior:



The screenshot shows the JWT.io website interface. At the top, there's a navigation bar with the JWT logo, links for 'Debugger', 'Libraries', 'Ask', and 'Get a T-shirt!', and a user profile 'Ryan'. Below the navigation bar, there's a dropdown menu for 'ALGORITHM' set to 'HS256'. The main content area is split into two columns: 'Encoded' and 'Decoded'. The 'Encoded' column displays a long string of base64-encoded characters. The 'Decoded' column shows the decoded structure of the token, including the header, payload, and signature verification details. A blue banner at the bottom indicates 'Signature Verified'.

Encoded

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoiTjVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
```

Decoded

HEADER:

```
{  "alg": "HS256",  "typ": "JWT"}
```

PAYLOAD:

```
{  "sub": "1234567890",  "name": "John Doe",  "admin": true}
```

VERIFY SIGNATURE

```
HMACSHA256(  base64UrlEncode(header) + "." +  base64UrlEncode(payload),  secret  ) ☐ secret base64 encoded
```

Signature Verified

Ilustración 12 contenido de un token JWT

2.5.3.1 Funcionamiento de Json Web Token.

Luego de que el usuario haya iniciado sesión en un sistema que utiliza este algoritmo, este devolverá un JWT. Por seguridad, el resultado no se debe guardar por mucho tiempo, para prevenir brechas de seguridad, ya que este token contiene las credenciales del usuario. Al haber obtenido el token, el usuario puede acceder a todos los servicios del sistema a los que tiene autorización para acceder, esto se debe ya que internamente el sistema devuelve el Token en una cabecera llamada “Authorization” seguido de la palabra Bearer, quedando de la siguiente manera: Authorization: Bearer <token>, de esta manera el sistema puede verificar la identidad del usuario. (auth0, 2021)

2.5.3.2 ¿Son los Tokens Jwt Seguros?

Según un artículo del banco BBVA (Bbva, 2020) los tokens no están encriptados y la cadena que se visualiza es solo una cadena serializada usando la codificación base64 url, que se puede decodificar fácilmente para ver claramente el contenido JSON del token. De tal manera la seguridad o confiabilidad de un JWT depende de la configuración que se le dé al sistema.

2.5.4 Fernet Vs Json Web Token

Al estudiarse estas dos tecnologías que ayudan con el manejo y seguridad de los usuarios, se puede observar que Fernet es una buena alternativa de JWT, esto debido a las siguientes razones:

- Implementar JWT con una buena seguridad es más complicado que hacerlo con Fernet.
- El tiempo de desarrollo es más rápido con Fernet.
- Fernet propone MultiFernet, que es de doble ciclo, importante para mejorar la seguridad.
- Fácil implementación con Fernet.
- La librería que contiene a la clase Fernet está disponible para múltiples lenguajes.
- Fernet encripta y firma con claves el token que se obtiene, por su parte JWT solo lo firma por sí solo.

2.5.5 Algoritmo de Hasheo de Contraseñas ARGON2

Para el uso y manejo de las contraseñas de los usuarios que sirvieron para poder ingresar a la plataforma, se encontró la siguiente alternativa:

Argon2 es un algoritmo criptográfico especializado en hash de contraseñas, específicamente, es una función de derivación de claves (KDF) diseñada para ser resistente a la GPU, permitiendo mantener una buena seguridad al tratar las contraseñas de un sistema, además de ser fácil de utilizar y utilizar pocos recursos del entorno donde se implementa. (Ciphr Team, 2021)

Usando Argon2 en Kotlin. En este proceso se realizó un ejemplo de uso para Argon2:

Se debe instalar las dependencias, en este caso con Gradle:

```
implementation("de.mkammerer:argon2-jvm:2.10.1")
```

Importar la librería en la clase donde se va a utilizar:

```
import de.mkammerer.argon2.Argon2Factory
```

Creando una función donde se llama al método y se encripta un string:

```
fun encrypt (password: String): String {  
    val argon2 = Argon2Factory.create()  
    val password1 = argon2.hash(10, 65536, 1, password.toCharArray())  
    argon2.wipeArray(password.toCharArray())  
    return password1  
}
```

Explicación del método `argon2.hash(iterations, memory, parallelism, password)`

Parámetros	Retorno	Errores	Descripción
------------	---------	---------	-------------

iterations(int), memory(int), parallelism(int), password (char [])	Devuelve un string hashed	TypeError, si los parámetros no son del tipo especificado	Este método recibe una lista de char y retorna un string encriptado.
---	---------------------------------	---	---

Tabla 10 Descripción del método argo2.hash (Ciphr Team, 2021)

Donde:

- Iterations: Es el número de iteraciones.
- Memory: Establece el uso de memoria en x kibibytes.
- Parallelism: Número de subprocesos y contraseña de carriles de cálculo.
- Password: La lista de char que desea hashear.

2.5.6 Cortafuegos o Firewall

De acuerdo con el instituto nacional de seguridad (incibe), se trata de dispositivos de seguridad cuya función principal es filtrar el tráfico de red entrante y saliente con una serie de reglas, permitiendo o negando ese tráfico. Cuando una comunicación llega al firewall, como una solicitud a un servidor web corporativo, se puede aceptar o rechazar, según cómo se hayan configurado las reglas. (incibe, 2019)

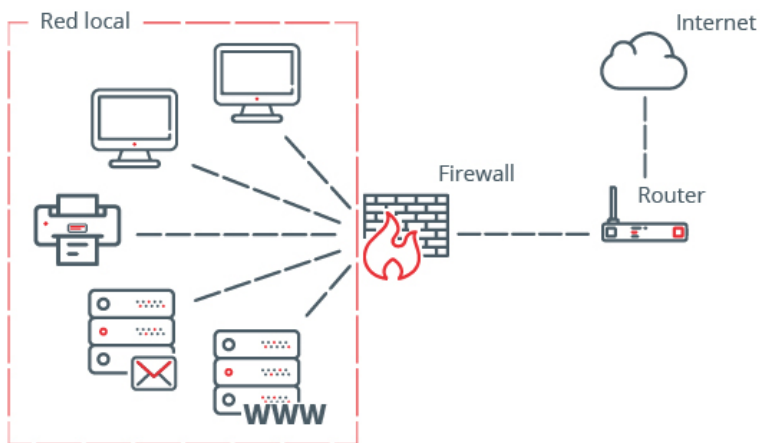


Ilustración 13 Explicación gráfica de un cortafuegos (incibe, 2019)

2.5.7 Zona Desmilitarizada

Es una red separada que forma parte de la intranet de una organización. Todos los recursos de la empresa a los que se debe acceder desde Internet, como la web o los servidores de correo, se encuentran allí por separado. (incibe, 2019)

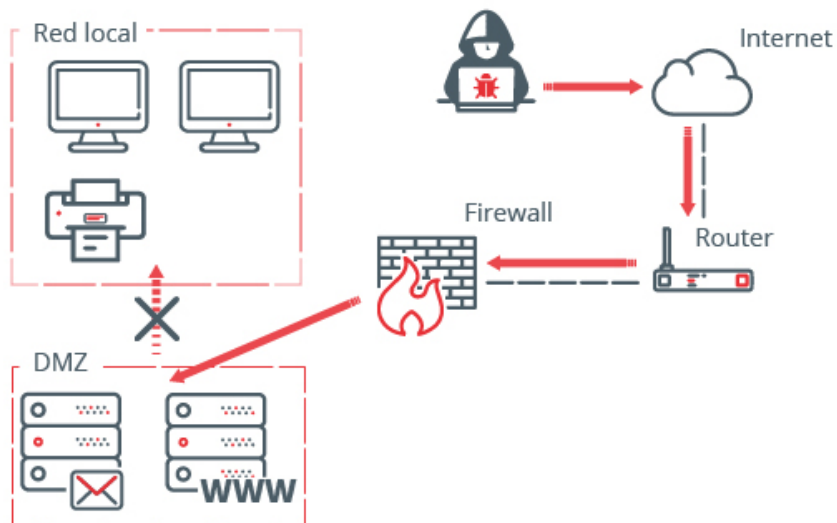


Ilustración 14 ejemplo de una red local con una DMZ (incibe, 2019)

2.5.8 Firewall de Aplicaciones Web (Waf)

Protege contra múltiples ataques contra servidores de aplicaciones web en el backend. Su funcionalidad es la de garantizar la seguridad del servidor web mediante el análisis de los paquetes de solicitud HTTP / HTTPS y los patrones de tráfico.

Waf monitoriza cada solicitud enviada al servidor, antes de que llegue a la aplicación, para asegurarse de que cumpla con las. Funciones WAF que se pueden implementar:

- En software: instalar la aplicación en el sistema operativo.
- En hardware: integrar funciones en la solución del dispositivo. (Oracle, 2021)

2.5.9 Balanceador de Carga

Así es como se distribuyen las solicitudes de Internet en una fila de servidores. Existen varios métodos para implementar el equilibrio de carga. Desde el simple "Round Robin" (que distribuye todas las solicitudes provenientes de Internet entre los servidores disponibles para dicho servicio) hasta las computadoras que reciben la solicitud, recolectan información, en tiempo real, sobre el desempeño de la computadora, y se utilizan para enrutar estas solicitudes individualmente al servidor que está mejor ubicado para proporcionar el servicio adecuado. Los balanceadores de carga pueden ser soluciones de hardware como enrutadores y conmutadores que incluyen software de balanceo de carga preparado para él y soluciones de software instaladas en la parte posterior del servidor del dispositivo. (Fernández, 2020)

2.5.10 Grupos de Seguridad de Amazon

De acuerdo con Amazon Web Services "Un grupo de seguridad funciona como un firewall virtual para las instancias EC2 para controlar el tráfico entrante y saliente. Las reglas de entrada controlan el tráfico entrante a la instancia y las reglas de salida controlan el tráfico saliente desde la instancia. Al lanzar una instancia puede especificar uno o varios grupos de seguridad. Si no especifica un

grupo de seguridad, Amazon EC2 utiliza el grupo de seguridad predeterminado. Puede añadir reglas a cada grupo de seguridad que permitan el tráfico a o desde sus instancias asociadas. Puede modificar las reglas de un grupo de seguridad en cualquier momento. Las reglas nuevas y modificadas se aplican automáticamente a todas las instancias asociadas al grupo de seguridad. Cuando Amazon EC2 decide si se permite que el tráfico llegue a una instancia, evalúa todas las reglas de todos los grupos de seguridad asociados a la instancia". (Amazon Web Services, 2021)

2.5.11 Proxy

Consiste en una computadora que actúa como intermediario entre el cliente y el servidor de destino. Se encarga de recibir las solicitudes de los clientes y las reenvía al servidor web al que desea acceder.

El servidor proxy evita que haya una conexión directa entre el usuario y el destino. Es decir, permiten enmascarar la dirección IP del usuario para que no se pueda conocer el sitio web visitado. En otras palabras, gracias a un proxy, se beneficia de un mayor anonimato. (Tablado, 2021)

3 Modelo Para el Manejo y Seguridad de los Usuarios

Mediante el Uso del Microservicio Users

3.1 Requerimientos del Sistema Para el Desarrollo del Modelo

Se realizó el proceso que cumplió con el objetivo número dos del proyecto, en donde se definieron los requerimientos de la plataforma para la creación del modelo. Estos requerimientos están estipulados en los siguientes tópicos:

- Funcionalidad que permita a un usuario el ingreso a la plataforma no-code de forma segura.
- Funcionalidad que permita a un usuario iniciar sesión en la plataforma no-code.
- Funcionalidad que permita a un usuario registrarse en la plataforma no-code.
- Funcionalidad que permita a un usuario recuperar el acceso a la plataforma no-code
- Funcionalidad que permita a un usuario visualizar su información en la plataforma no-code.
- Funcionalidad que permita a un usuario actualizar su información de manera segura en la plataforma no-code de forma segura.

Estas actividades se llevaron a cabo en la primera semana del Offsite1, donde el equipo de producto expuso de manera general los requerimientos exhibidos con anterioridad y el diseño requerido para la implementación de la plataforma. Por tal motivo se requirió que el microservicio que manejó el tema de los usuarios, llamado microservicio users haya establecido la mayor seguridad para el manejo, almacenamiento, autenticación, integridad y autenticidad de los datos de las personas que utilizarán la plataforma.

3.2 Análisis de los Requerimientos del Sistema Para el Desarrollo del Modelo

Por lo descrito por el equipo de producto en los requerimientos para el microservicio users se puede deducir que es uno de los microservicios más importantes que se incluyeron en la plataforma no-code, esto es debido a que los demás microservicios dependen de este para poder comunicarse entre sí, debido a que si un usuario intenta acceder a alguna funcionalidad y no se ha registrado o ha iniciado sesión en la plataforma, este no lo podrá hacer, porque para hacerlo debe autenticarse en el sistema y así detecte la autenticidad e integridad de sus datos. Por lo tanto, además de las tecnologías que se implementaron en el microservicio, también se implementaron tecnologías en la arquitectura de la plataforma, esto con el fin de garantizar una mayor seguridad e integridad de la información suministrada por el usuario. Dichas tecnologías fueron facilitadas por la empresa donde se llevó a cabo el proyecto, las cuales ya fueron expuestas en el ítem de referentes teóricos. A continuación, se explicará el uso de dichas tecnologías en la arquitectura.

Waf: Este es un firewall de aplicación web que se encarga de verificar si las peticiones tienen algún tipo de ataque o algún componente que pueda llegar con algo malicioso, este posee reglas y algoritmos que se crean para alertar sobre el tráfico que se considera sospechoso.

Balanceador: Se dedica a tomar peticiones https y manda la solicitud al servidor o aplicación que corresponda.

Zona desmilitarizada: Esta tecnología es aplicada en la plataforma para aislar el balanceador de la plataforma, es decir la zona desmilitarizada permite que la petición pueda acceder al balanceador para que este redirija la petición al microservicio u otra aplicación. De esta manera obliga a la petición llegar al balanceador y no consumir directamente las funcionalidades del microservicio.

Grupo de seguridad: Este es un firewall virtual dedicado al control del tráfico información entrante y saliente.

Redes privadas: Estas redes en la infraestructura se implementan en cada capa de ella con un grupo de seguridad que están segmentadas para aislarse una de otras.

EC2: Esta es una aplicación de Amazon Web Services la cual está dedicada al hosting, donde se pueden correr las aplicaciones, microservicios.

Estas tecnologías mencionadas se implementan con ayuda de Amazon Web Services que permite darle las configuraciones pertinentes para su buen funcionamiento.

Posteriormente, se explicarán las tecnologías utilizadas en el microservicio users, que ayudan con la seguridad y el manejo de la información de los usuarios.

Fernet: Es utilizado para generar tokens, que contienen información del usuario para realizar su autenticación. Además de esto Fernet proporciona otras funcionalidades que ayudan a incrementar la seguridad de los usuarios en el microservicio users.

Argon2: Esta tecnología fue utilizada para encriptar y desencriptar las contraseñas de los usuarios en la plataforma no-code. Para mejorar la seguridad en la base de datos de usuarios, se guardó esa clave encriptada para evitar robo de información.

En conclusión, estas tecnologías son impuestas por la empresa para el manejo y seguridad de la información de los usuarios, en el microservicio users.

3.3 Diagramas de Casos de Uso Del Microservicio Users

A continuación, se describirá las funcionalidades de los servicios que se propusieron en los requerimientos y posteriormente analizados para la construcción del modelo que ayudó en el manejo y seguridad de los usuarios en la plataforma no-code.

3.3.1 Flujo General del Microservicio Users

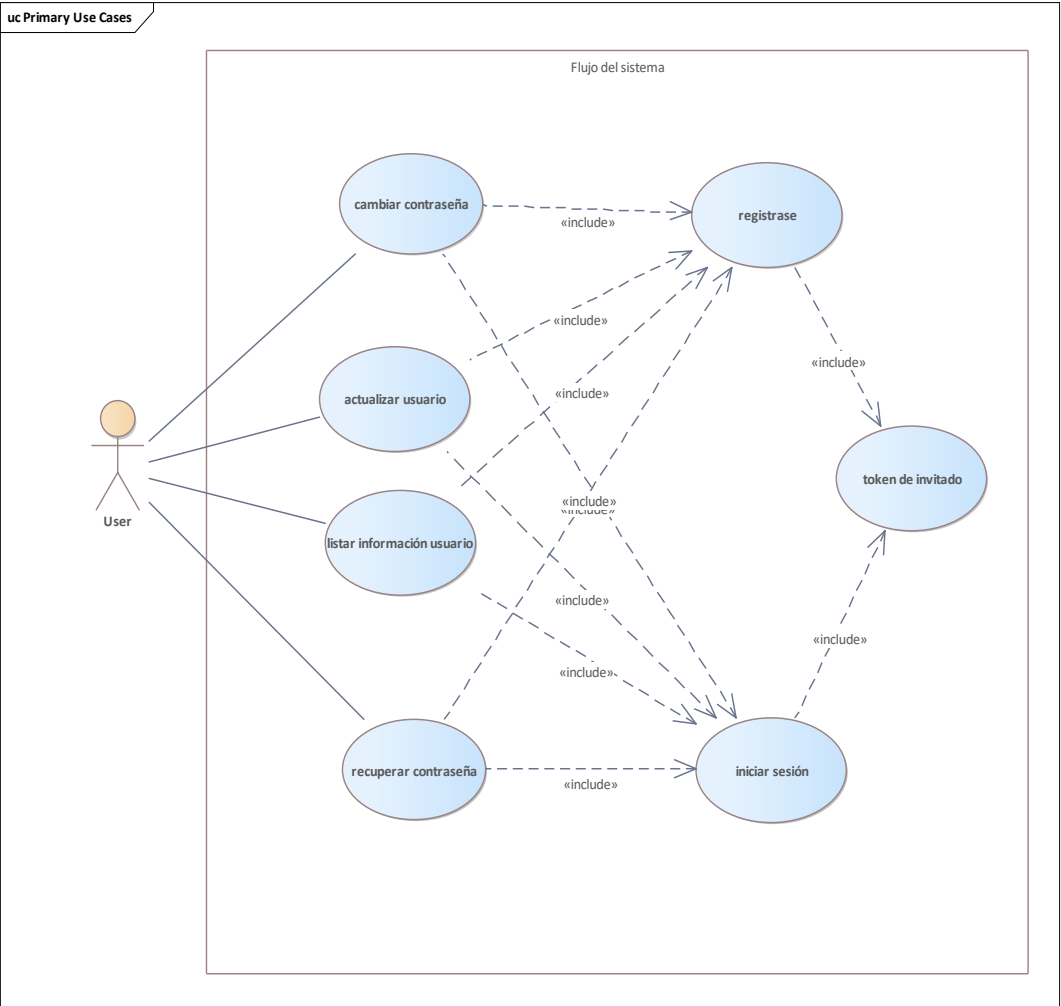


Ilustración 15 flujo general del sistema

3.3.2 Iniciar Sesión

Nombre del caso de uso	Iniciar Sesión
------------------------	----------------

Actor participante	Usuario
Condición inicial	El usuario entra a la página
Flujo de eventos	Se muestra ventana con opciones iniciar sesión o registrase. Oprime botón iniciar sesión. Muestra ventana iniciar sesión con formulario y botones de iniciar sesión o ¿has olvidado la contraseña? Usuario ingresa los campos del formulario (nombre de usuario, contraseña).
Condición de salida	Se muestra la ventana principal con sus funcionalidades.
Errores	Si el usuario ingresa las credenciales mal, se retornará un error.

Tabla 11 caso de uso iniciar sesión

3.3.3 Registrase

Nombre del caso de uso	Registrar usuario
Actor participante	Usuario

Condición inicial	1. El usuario entra a la página
Flujo de eventos	Se muestra ventana con opciones iniciar sesión o registrase. Oprime botón Registrarse. Muestra ventana registrarse con formulario y botón registrase. Usuario ingresa los campos del formulario (nombre de la cuenta, nombre de usuario, contraseña).
Condición de salida	Se muestra la ventana principal con sus funcionalidades.
Errores	Si el username no es de tipo email, se retornará un error. Si el username ya existe, se retornará un error. Si no se llena todos los campos requeridos del formulario, se retornará un error.

Tabla 12 caso de uso registrarse

3.3.4 Cambiar Contraseña

Nombre del caso de uso	Cambiar contraseña
Actor participante	Usuario

Condición inicial	1. Muestra ventana principal con sus funcionalidades.
Flujo de eventos	El usuario oprime botón perfil usuario. Muestra ventana perfil usuario con funcionalidades. Oprime botón cambiar contraseña. Muestra ventana cambiar contraseña con formulario y un botón. Ingresa los datos del formulario (contraseña actual, nueva contraseña) y oprime botón cambiar. Muestra popup con mensaje “Se ha cambiado la contraseña” más botón aceptar. Oprime el botón aceptar.
Condición de salida	Muestra ventana principal con sus funcionalidades
Errores	Si el usuario ingresa mal la contraseña actual, retornará un error. Si el usuario ingresa una contraseña que ya había tenido en el campo “nueva contraseña”, retornará un error. Si no se llena todos los campos requeridos del formulario, se retornará un error.

Tabla 13 caso de uso cambiar contraseña

3.3.5 Recuperar Contraseña

Nombre del caso de uso	Recuperar contraseña
------------------------	----------------------

Actor participante	Usuario
Condición inicial	El usuario entra a la página
Flujo de eventos	Se muestra ventana con opciones iniciar sesión o registrase. Oprime botón iniciar sesión. Muestra ventana iniciar sesión con formulario botones de iniciar sesión o ¿has olvidado la contraseña? Oprime botón ¿has olvidado la contraseña? Muestra ventana recuperar contraseña con formulario y botones de enviar email o volver. Ingresa los datos del formulario (nombre de usuario) y oprime botón enviar email. Muestra popup con mensaje “se ha enviado un correo que contiene una clave para recuperar contraseña” más formulario y botón recuperar contraseña. Ingresa formulario (clave) y oprime botón recuperar contraseña. Muestra ventana con formulario ingresar nueva contraseña más botón aceptar. Ingresa formulario (nueva contraseña) y oprime botón aceptar.
Condición de salida	Muestra ventana principal con sus funcionalidades más mensaje “se ha cambiado la contraseña satisfactoriamente”.

Errores	Si no ingresa su nombre de usuario correctamente, en ningún momento le llegará un correo a su correo. Si el usuario ingresa una clave diferente a la que le llegó al correo, retornará error. Si el usuario ingresa una contraseña que ya había tenido en el campo “nueva contraseña”, retornará un error.
---------	---

Tabla 14 caso de uso recuperar contraseña

3.3.6 Actualizar Información Usuario

Nombre del caso de uso	Actualizar información
Actor participante	Usuario
Condición inicial	Muestra ventana principal con sus funcionalidades.
Flujo de eventos	El usuario oprime botón perfil usuario. Muestra ventana perfil usuario con funcionalidades. Oprime botón cambiar actualizar información. Muestra ventana actualizar información con un formulario y un botón aceptar.

	Ingresa los datos del formulario que desea cambiar (nombre, apellido, teléfono) y oprime botón aceptar. Muestra popup con mensaje “Está seguro que desea cambiar la información” más botón aceptar. Oprime el botón aceptar.
Condición de salida	Muestra ventana principal con mensaje “se ha cambiado la información satisfactoriamente” más sus funcionalidades.
Errores	Si el usuario deja un campo vacío del formulario, se retornará un error.

Tabla 15 caso de uso actualizar información usuario

3.3.7 Mostrar Información del Usuario

Nombre del caso de uso	Información del usuario
Actor participante	Usuario
Condición inicial	Muestra ventana principal con sus funcionalidades.
Flujo de eventos	El usuario oprime botón perfil usuario.
Condición de salida	Muestra ventana perfil usuario con funcionalidades.

Errores	
---------	--

Tabla 16 caso de uso mostrar información usuario

3.3.8 Obtener Token de Invitado

Nombre del caso de uso	Información del usuario
Actor participante	Usuario
Condición inicial	Usuario ingresa al link de la plataforma.
Flujo de eventos	Muestra ventana con botón iniciar sesión o registrase.
Condición de salida	El usuario puede oprimir cualquiera de los dos botones.
Errores	

Tabla 17 caso de uso obtener token invitado

3.4 Diagramas de Secuencia Del Microservicio Users

Según la página especializada en diagramas Lucidchart, un diagrama de secuencia es un tipo de diagrama descriptivo pues relaciona cómo, y en qué orden, un grupo de asuntos trabaja en conjunto. Los equipos de software o equipos corporativos utilizan estos diagramas para comprender los

requisitos de un nuevo sistema o para documentar un proceso existente. Estos diagramas también son conocidos como diagramas de eventos o escenarios de eventos. A continuación, se mostrarán los diagramas de flujo de los servicios, describiendo su paso a paso. (Lucidchart, 2021)

3.4.1 Obtener Token Invitado

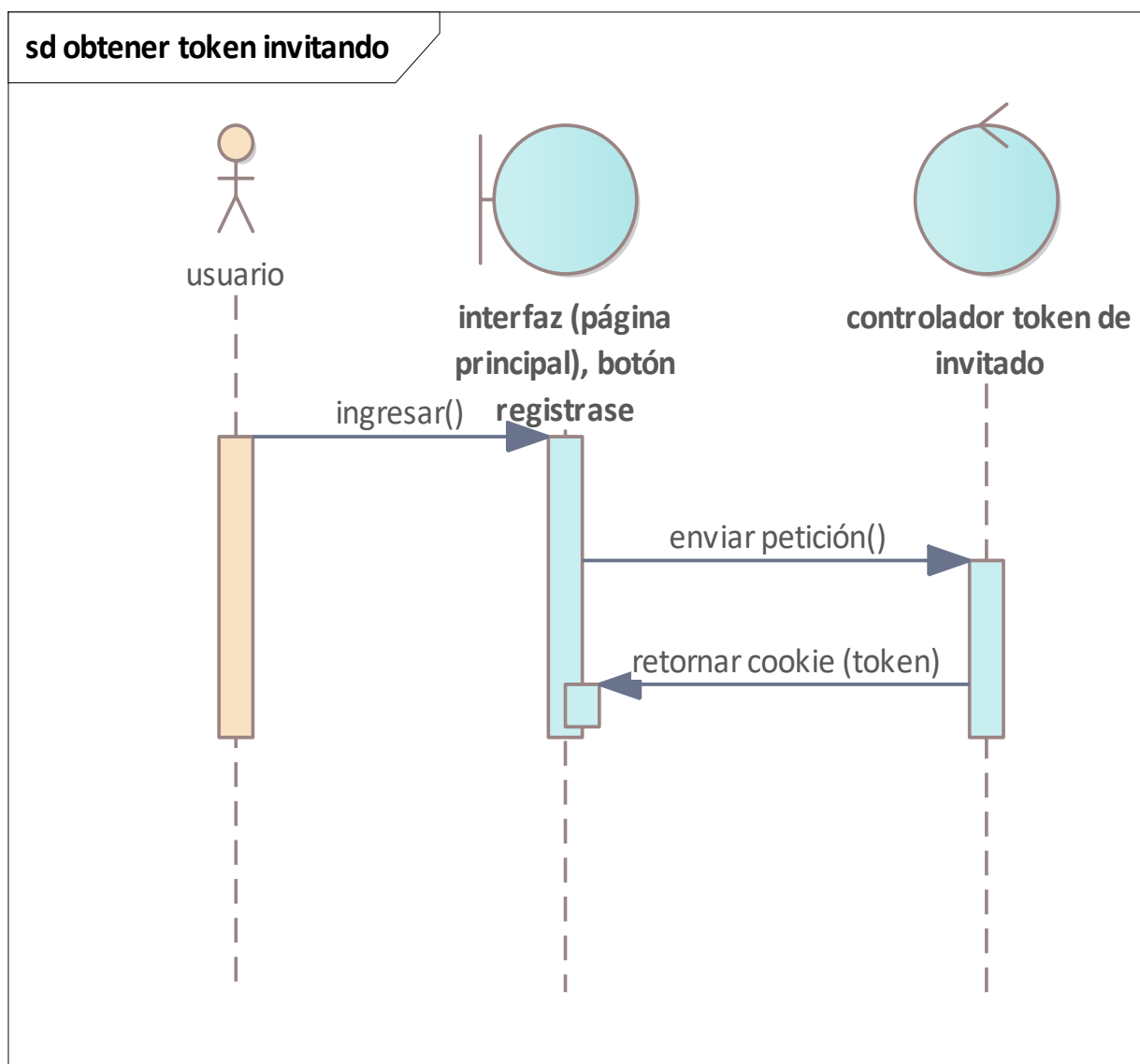


Ilustración 16 diagrama de secuencia obtener token invitado

3.4.2 Iniciar Sesión

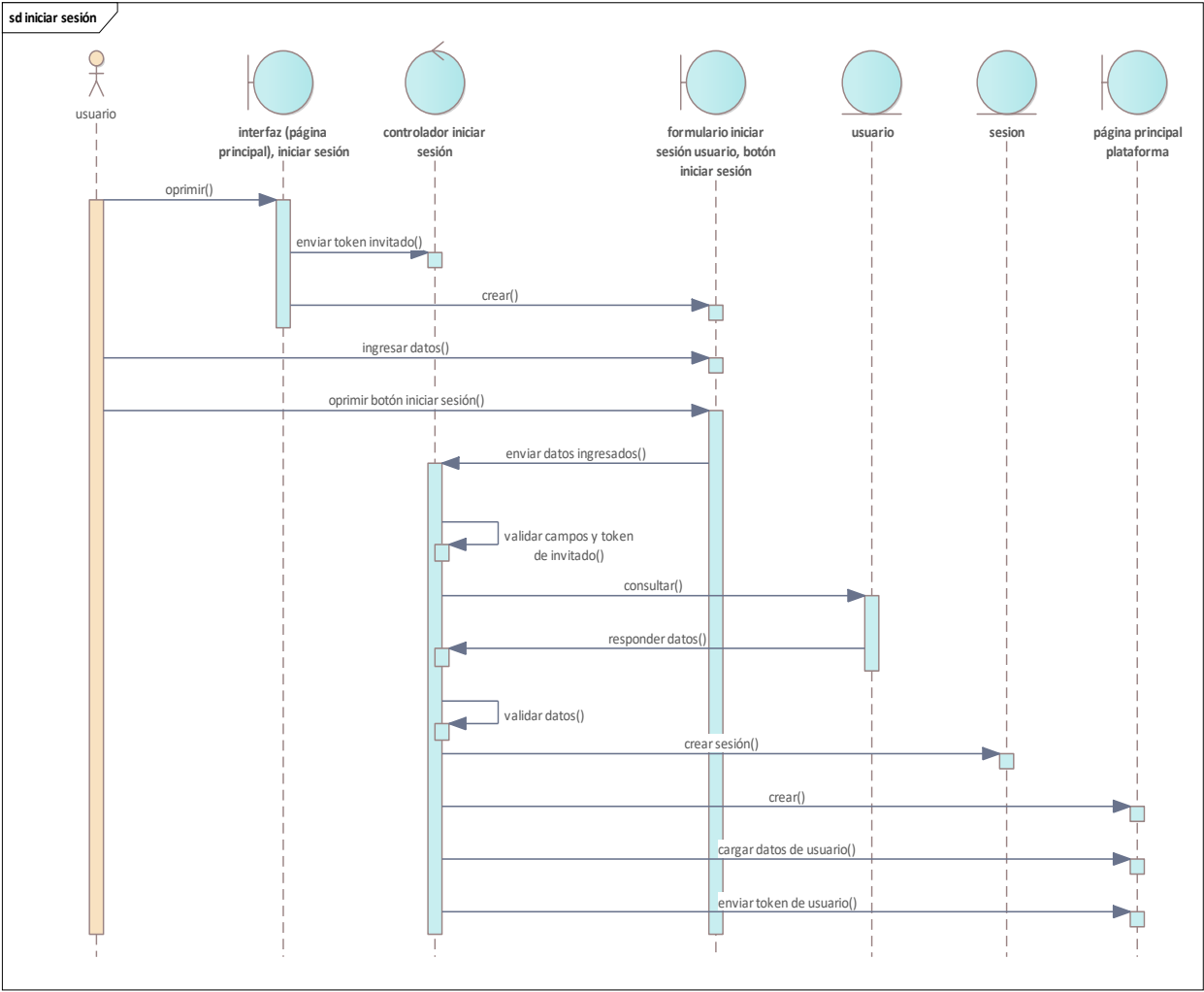


Ilustración 17 diagrama de secuencia iniciar sesión

3.4.3 Registrarse

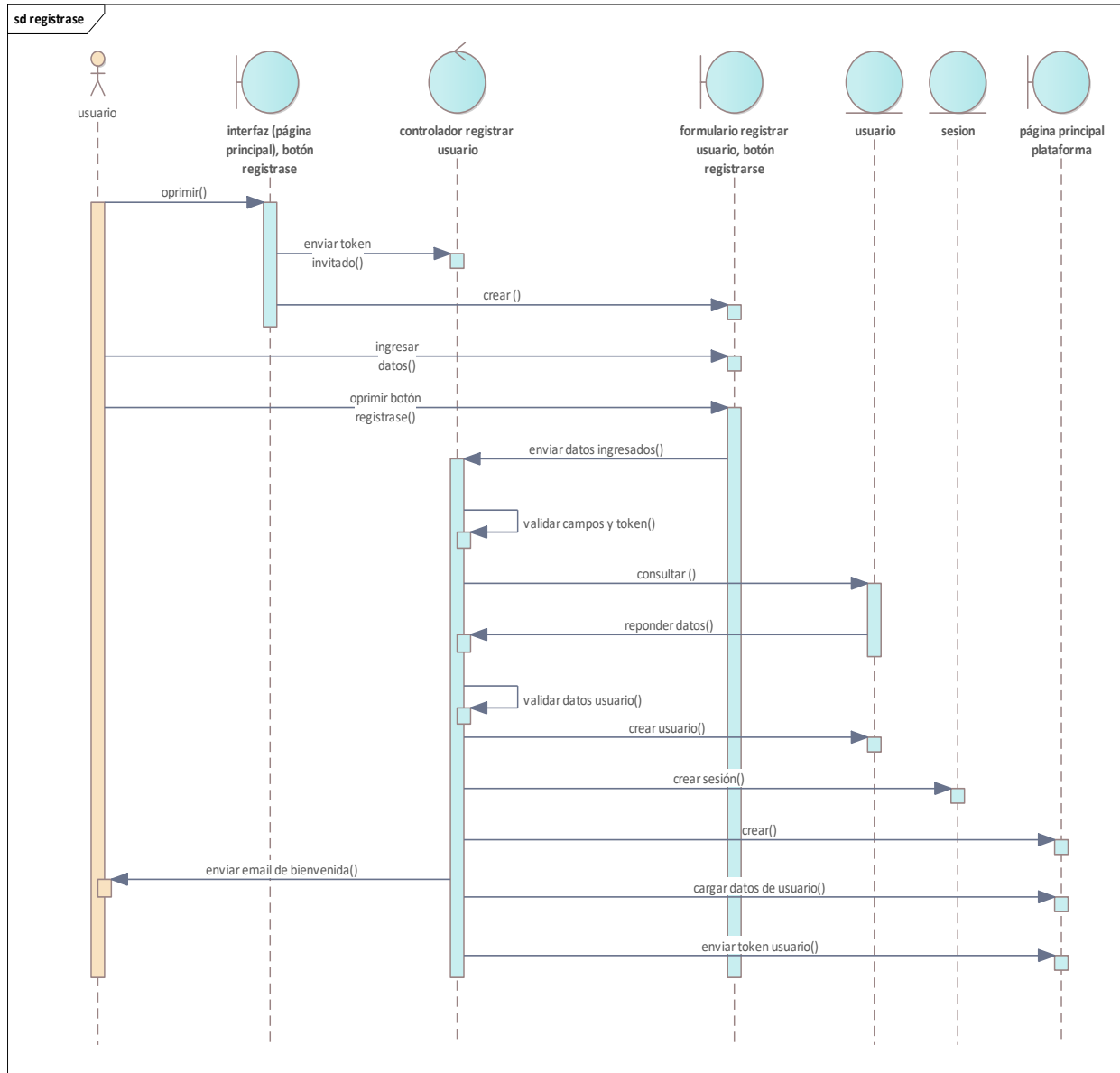


Ilustración 18 diagrama de secuencia registrarse

3.4.4 Cambiar Contraseña

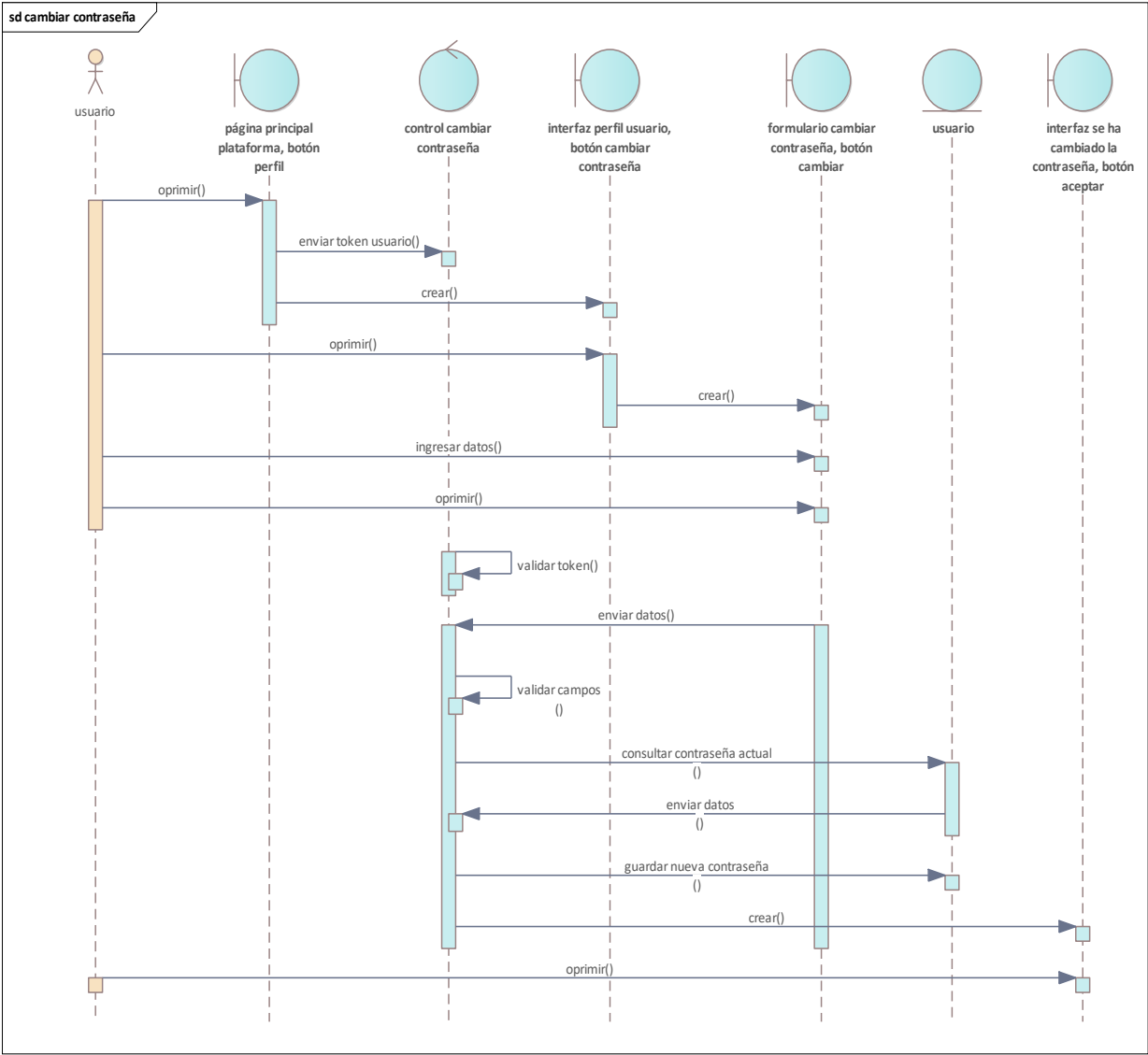


Ilustración 19 diagrama de secuencia cambiar contraseña

3.4.5 Recuperar Contraseña

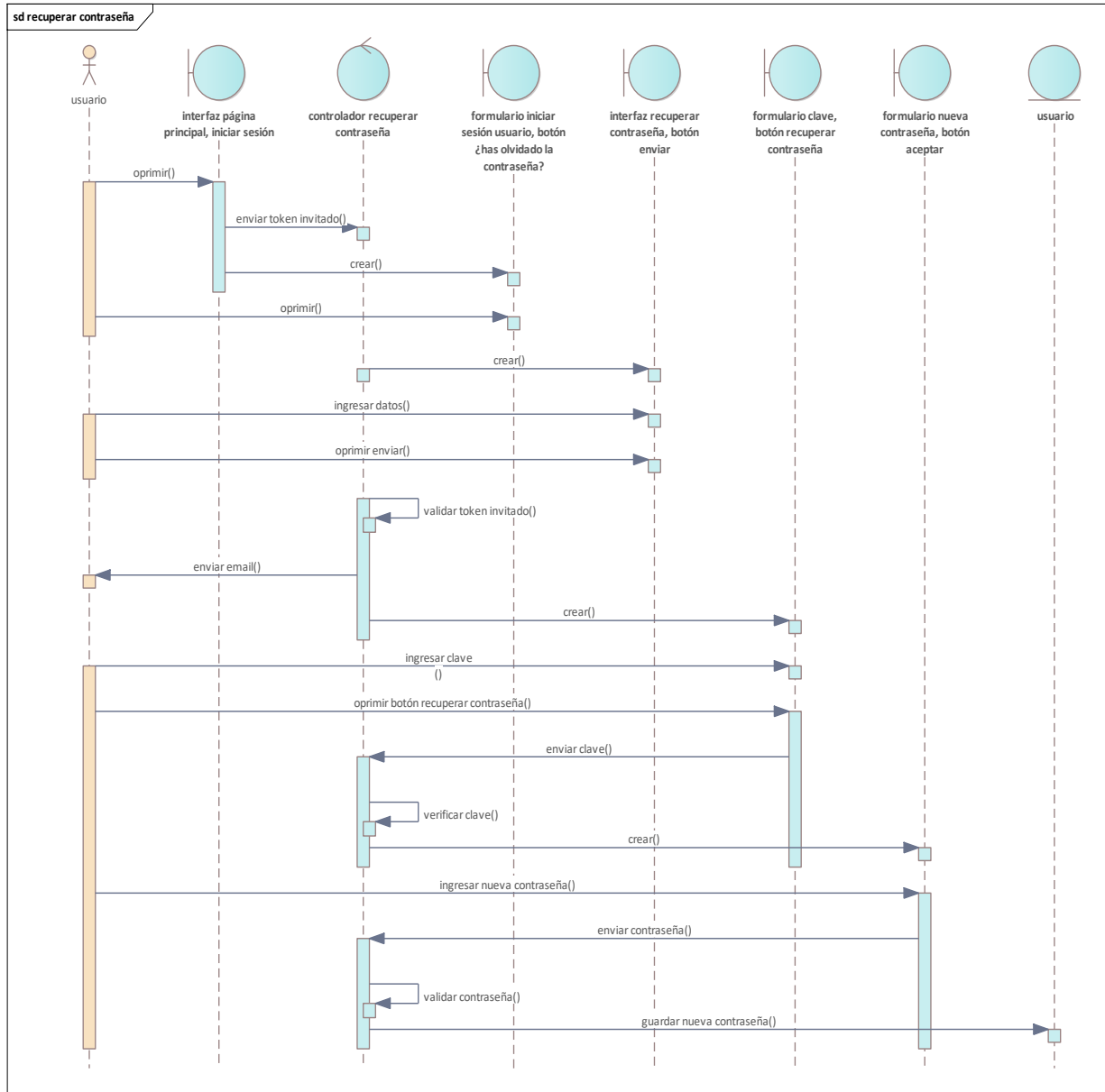


Ilustración 20 diagrama de secuencia recuperar contraseña

3.4.6 Actualizar Información Usuario

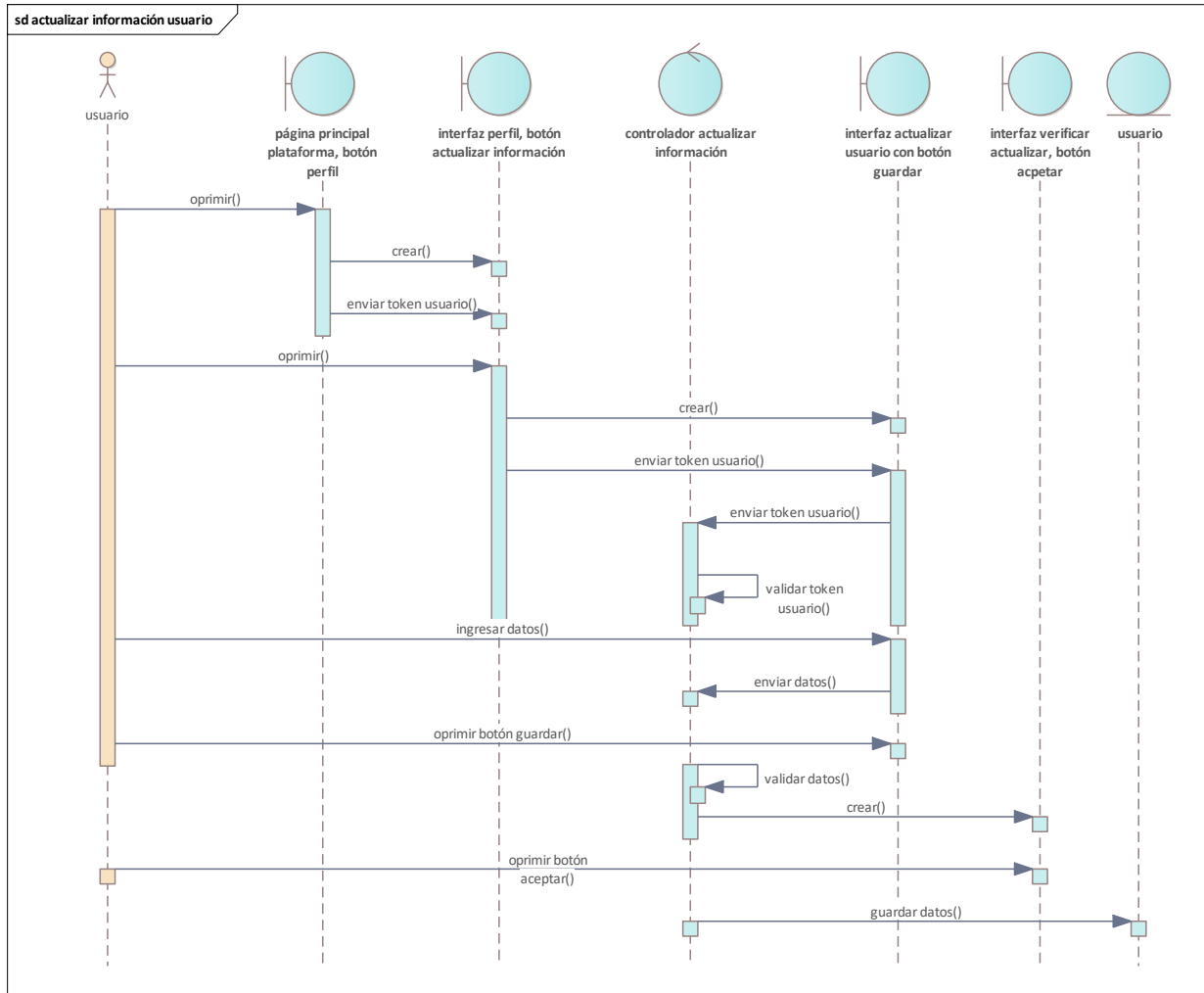


Ilustración 21 diagrama de secuencia actualizar información usuario

3.4.7 Listar Información Usuario

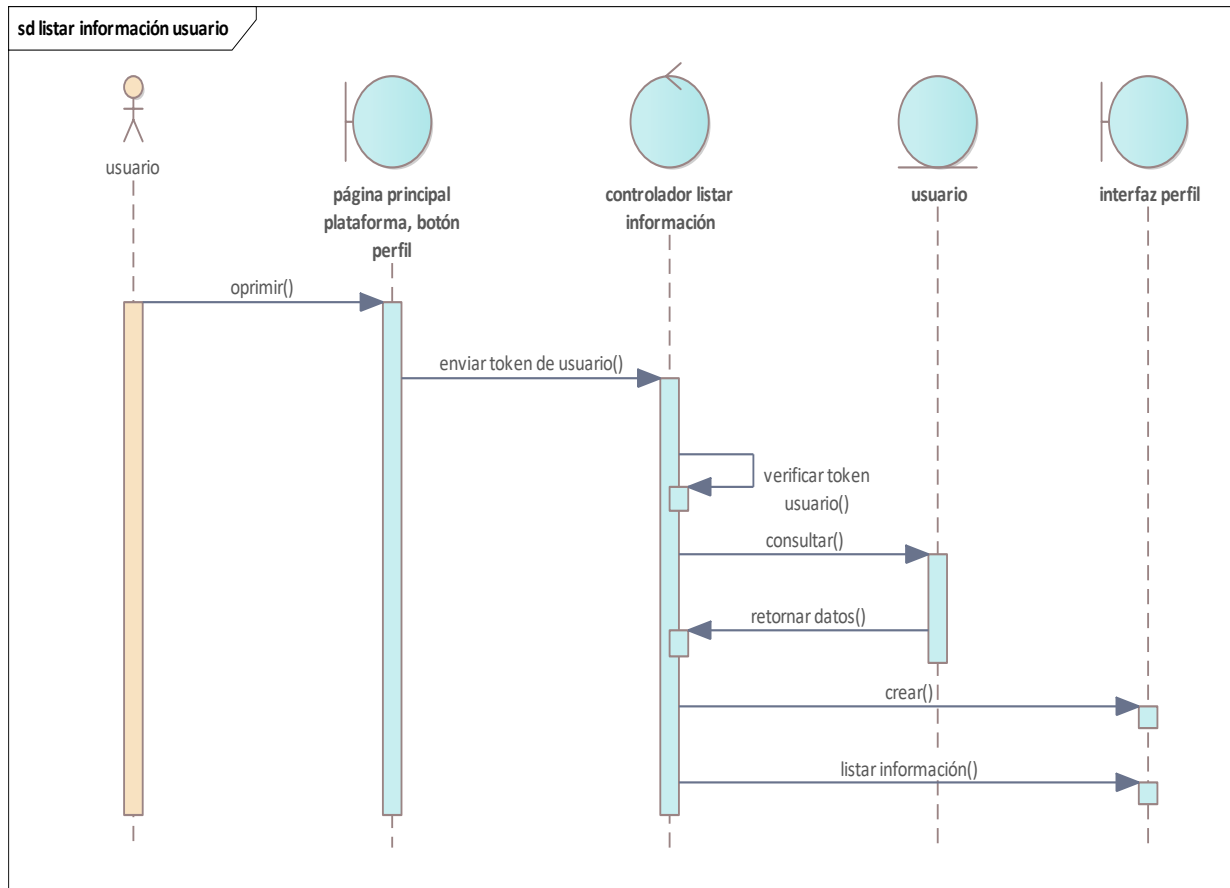


Ilustración 22 diagrama de secuencia listar información usuario

3.5 Descripción Técnica de los Casos de Uso

En las siguientes tablas se va describió de forma técnica cada caso de uso presente en los diagramas de secuencia, para que el lector tenga una mejor idea de su funcionamiento interno y de las herramientas tecnológicas usadas.

3.5.1 Iniciar Sesión

Caso de uso	Iniciar sesión
Descripción	Este proceso tiene como objetivo la validación de las credenciales del usuario que quiere ingresar a la plataforma, al obtener una respuesta positiva guarda la información de la sesión en la tabla sesión. Por último, permite ingresar al usuario a la plataforma.
Datos y acciones de entradas	Recibe un token de invitado, nombre de usuario y contraseña.
Procesos internos del sistema	Para garantizar la seguridad del usuario, el sistema realiza los siguientes sub procesos: Al usuario entrar a la página principal de la plataforma, el sistema le asigna un token de invitado. Al usuario oprimir el botón iniciar sesión, el sistema verifica si el usuario ya existe en la base de datos, al hacerlo, después se asegura que la contraseña obtenida es la correspondiente del usuario, esto lo hace con una función del algoritmo Argon2. También comprueba que el token de invitado sea válido. Antes de dirigir a la página principal al usuario, el sistema toma los datos del usuario tomados de

	la base de datos para retornar un Token de tipo usuario, que contiene la información del usuario cifrada. El sistema introduce en las cookies de la plataforma el Token retornado.
Herramientas tecnológicas utilizadas	Algoritmo de encriptación Argon2.
Procedimientos especiales requeridos	Si nombre de usuario es inválido o la contraseña, el sistema retorna un error: “Las credenciales son incorrectas, por favor inténtalo de nuevo”. Si algún usuario intenta ingresar con un token de invitado no válido, aparecerá un error de tipo “El usuario no tiene los permisos para acceder o el token de invitado es inválido”. Si el usuario tarda mucho tiempo sin registrarse (3600 segundos), el token de invitado se vencerá, por tal motivo lanzará un mensaje: “El tiempo de espera ya caduco, por favor vuelva a cargar la página”
Resultados esperados	El usuario debe poder iniciar sesión correctamente en la plataforma, de tal manera poder acceder a esta.

Tabla 18 descripción técnica caso de uso iniciar sesión

3.5.2 Registrar Usuario

Caso de uso	Registro del usuario
Descripción	Este caso se encarga de registrar el usuario, guardar su información en la base de datos Users en la tabla usuario, así como la información de la sesión en la tabla sesión. Además, permite ingresar al usuario a la plataforma.
Datos y acciones de entradas	Recibe un token de invitado, el nombre de la cuenta, nombre de usuario y contraseña.
Procesos internos del sistema	Para garantizar la seguridad del usuario, el sistema realiza los siguientes sub procesos: Al usuario entrar a la página principal de la plataforma, el sistema le asigna un token de invitado. Al usuario oprimir el botón registrarse, el sistema verifica si el usuario ya existe en la base de datos, también verifica que el token de invitado sea válido. Al usuario registrarse, el sistema encripta la contraseña con el algoritmo de encriptación Argon2 y guarda los datos en la tabla usuario, así como la información de la sesión en la tabla sesión.

	Antes de dirigir a la página principal al usuario, el sistema toma los datos que el usuario registró y los encripta con el algoritmo de encriptación Fernet, retornando un Token de tipo usuario, que contiene la información del usuario cifrada. El sistema introduce en las cookies de la plataforma el Token retornado. Al terminar el proceso, el sistema envía un correo de confirmación al usuario.
Herramientas tecnológicas utilizadas	Algoritmo de encriptación Fernet. Algoritmo de encriptación Argon2.
Procedimientos especiales requeridos	El nombre de usuario debe ser de tipo email, si no es así retorna un error: “El nombre de usuario no es un correo válido”. Si nombre de usuario ya existe, el sistema retorna un error: “El nombre de usuario ya existe, por favor intenta con otro nombre de usuario”. Si algún usuario intenta ingresar con un token de invitado no válido, aparecerá un error de tipo “El usuario no tiene los permisos para acceder o el token de invitado es inválido”. Si el usuario tarda mucho tiempo sin registrarse (3600 segundos), el token de invitado se vencerá, por tal motivo lanzará un mensaje: “El tiempo de

	espera ya caduco, por favor vuelva a cargar la página”.
Resultados esperados	El usuario debe poder registrarse correctamente en la plataforma, de tal manera poder acceder a esta y recibir un correo de bienvenida.

Tabla 19 descripción técnica caso de uso registrar usuario

3.5.3 Cambiar Contraseña

Caso de uso	Cambiar Contraseña
Descripción	La finalidad de este servicio es poder cambiar la contraseña del usuario.
Datos y acciones de entradas	Recibe un token de usuario, contraseña actual, nueva contraseña.
Procesos internos del sistema	Para garantizar la seguridad del usuario, el sistema realiza los siguientes sub procesos: Al usuario oprimir el botón actualizar contraseña, el sistema verifica la identidad y autenticidad de este, esto se hace gracias a que en las cookies de la plataforma existe un token de usuario, puesto allí después de haber iniciado sesión o registrase. El servicio obtiene el token y lo verifica con el algoritmo de encriptación Fernet.

	Al usuario oprimir el botón cambiar y su identidad haya sido verificada con el paso anterior, el sistema chequea que la contraseña actual escrita en el formulario sea la correcta con el algoritmo Argon2, prosigue con la encriptación de la nueva contraseña, después guarda los cambios en la base de datos.
Herramientas tecnológicas utilizadas	Algoritmo de encriptación Argon2. Algoritmo de encriptación Fernet.
Procedimientos especiales requeridos	Si la contraseña actual es incorrecta, retornará un mensaje de error: “La contraseña actual es incorrecta, por favor intenta de nuevo”. Si algún usuario intenta ingresar con un token de usuario no válido, aparecerá un error de tipo “El usuario no tiene los permisos para acceder o el token de usuario es inválido”. Si la contraseña nueva suministrada por el usuario ya existía en la base de datos, el sistema retornará un mensaje de error: “Ya has utilizado esta contraseña anteriormente, por favor intenta con otra”. Si el usuario tarda mucho tiempo sin estar presente en la plataforma (3600 segundos), el token de usuario se vencerá, por tal motivo lanzará un mensaje: “El tiempo de espera ya caduco, por favor vuelva a cargar la página”.

Resultados esperados	El usuario debe poder cambiar la contraseña correctamente y permanecer en la plataforma sin tener que volver a iniciar sesión.
----------------------	--

Tabla 20 descripción técnica caso de uso cambiar contraseña

3.5.4 Recuperar Contraseña

Caso de uso	Recuperar contraseña
Descripción	Durante este proceso, el usuario debería recuperar su contraseña que por alguna u otra razón no tiene para poder iniciar sesión en la plataforma y poder disfrutar de sus servicios.
Datos y acciones de entradas	Recibe un token de usuario, nombre de usuario, clave enviada por email, y nueva contraseña.
Procesos internos del sistema	Para garantizar la seguridad del usuario, el sistema realiza los siguientes sub procesos: Al usuario entrar a la página principal de la plataforma, el sistema le asigna un token de invitado. Una vez que el usuario haya introducido el nombre de usuario, y haya oprimido el botón enviar correo, el sistema enviará un correo

	electrónico que contiene una clave secreta única para ese proceso. Cuando el usuario ingresa la clave que fue resultado del anterior paso, el sistema verifica que sea correcta. Si es verdadera dejará cambiar la contraseña al usuario. Una vez que el usuario ingresa la nueva contraseña, el sistema verifica en la base de datos si esa contraseña ya había sido usada, si no ha sido usada, procede a encriptar la nueva contraseña con el algoritmo Argon2, por último, actualiza la data en la base de datos.
Herramientas tecnológicas utilizadas	Algoritmo de encriptación Argon2. Algoritmo de encriptación Fernet.
Procedimientos especiales requeridos	En caso que la clave ingresada en el formulario no coincide con el que fue enviado al correo, se retornará un error: “La clave suministrada no coincide con la enviada por correo electrónico, por favor inténtalo de nuevo” Si la contraseña nueva suministrada por el usuario ya existía en la base de datos, el sistema retornará un mensaje de error: “Ya has utilizado está contraseña anteriormente, por favor intenta con otra”. Si algún usuario intenta ingresar con un token de invitado no válido, aparecerá un error de tipo “El

	usuario no tiene los permisos para acceder o el token de invitado es inválido”. Si el usuario tarda mucho tiempo sin estar presente en la plataforma (3600 segundos), el token de invitado se vencerá, por tal motivo lanzará un mensaje: “El tiempo de espera ya caduco, por favor vuelva a cargar la página”.
Resultados esperados	El usuario debe poder cambiar la contraseña correctamente y permanecer en la plataforma sin tener que volver a iniciar sesión.

Tabla 21 descripción técnica caso de uso recuperar contraseña

3.5.5 Actualizar Información del Usuario

Caso de uso	Actualizar usuario
Descripción	El proceso de actualizar usuario permite modificar la información actual del usuario (nombre, teléfono, entre otros).
Datos y acciones de entradas	Recibe un token de usuario y datos a cambiar.
Procesos internos del sistema	Para garantizar la seguridad del usuario, el sistema realiza los siguientes sub procesos: Al usuario oprimir el botón actualizar datos, el sistema verifica la identidad y autenticidad de

	este, esto se hace gracias a que en las cookies de la plataforma existe un token de usuario, puesto allí después de haber iniciado sesión o registrase. El servicio obtiene el token y lo verifica con el algoritmo de encriptación Fernet. Luego de que el usuario haya ingresado los datos que desea cambiar y haya oprimido guardar, el sistema guarda los cambios en la base de datos.
Herramientas tecnológicas utilizadas	Algoritmo de encriptación Argon2. Algoritmo de encriptación Fernet.
Procedimientos especiales requeridos	Si el usuario deja algún campo del formulario vacío y este es requerido, el sistema retornará un error: “El campo “xxxx” es requerido, por favor ingrese un valor”. Si algún usuario intenta ingresar con un token de usuario no válido, aparecerá un error de tipo “El usuario no tiene los permisos para acceder o el token de usuario es inválido”. Si el usuario tarda mucho tiempo sin estar presente en la plataforma (3600 segundos), el token de usuario se vencerá, por tal motivo lanzará un mensaje: “El tiempo de espera ya caduco, por favor vuelva a cargar la página”.
Resultados esperados	El usuario puede actualizar la información de su perfil correctamente.

Tabla 22 descripción técnica caso de uso actualizar información del usuario

3.5.6 Listar Información del Usuario

Caso de uso	Listar información
Descripción	En este procedimiento, el usuario puede ver la información actual de su perfil.
Datos y acciones de entradas	Recibe un token de usuario.
Procesos internos del sistema	Para garantizar la seguridad del usuario, el sistema realiza los siguientes sub procesos: Al usuario oprimir el botón ver perfil, el sistema verifica la identidad y autenticidad de este, esto se hace gracias a que en las cookies de la plataforma existe un token de usuario, puesto allí después de haber iniciado sesión o registrase. El servicio obtiene el token y lo verifica con el algoritmo de encriptación Fernet. El sistema lista la información del usuario.
Herramientas tecnológicas utilizadas	
Procedimientos especiales requeridos	Si algún usuario intenta ingresar con un token de usuario no válido, aparecerá un error de tipo “El

	usuario no tiene los permisos para acceder o el token de usuario es inválido”. Si el usuario tarda mucho tiempo sin estar presente en la plataforma (3600 segundos), el token de usuario se vencerá, por tal motivo lanzará un mensaje: “El tiempo de espera ya caduco, por favor vuelva a cargar la página”.
Resultados esperados	El usuario puede visualizar la información de su perfil correctamente.

Tabla 23 descripción técnica caso de uso listar información del usuario

3.5.7 Token de Invitado

Caso de uso	OBTENER TOKEN DE INVITADO
Descripción	En este procedimiento, el sistema le asigna un token de invitado al usuario que ingresa a la página principal de la plataforma, para que cuando el usuario inicie sesión o se registre sea permitido.
Datos y acciones de entradas	
Procesos internos del sistema	Para garantizar la seguridad del usuario, el sistema realiza los siguientes sub procesos:

	Al usuario ingresar a la página principal de la plataforma, el sistema le asigna automáticamente un token de invitado en las cookies del navegador, para que pueda iniciar sesión o registrarse. Sin este token no podrá ingresar a las funcionalidades de la plataforma.
Herramientas tecnológicas utilizadas	Algoritmo de encriptación Fernet.
Procedimientos especiales requeridos	Si algún usuario intenta ingresar con un token de usuario no válido, aparecerá un error de tipo “El usuario no tiene los permisos para acceder o el token de usuario es inválido”. Si el usuario tarda mucho tiempo sin estar presente en la plataforma (3600 segundos), el token de usuario se vencerá, por tal motivo lanzará un mensaje: “El tiempo de espera ya caduco, por favor vuelva a cargar la página”.
Resultados esperados	El usuario puede ingresar a la página principal correctamente y tener en las cookies de su navegador el token de invitado.

Tabla 24 descripción técnica caso de uso token de invitado

3.6 Diseño del Modelo

Durante el desarrollo de este ítem, se llevó a cabo el modelo del microservicio Users para el manejo y seguridad de los usuarios. En el transcurso se desarrollaron casos de uso y otros diagramas para cada una de las funcionalidades que se obtuvieron al realizar los dos pasos anteriores (requerimientos y análisis). El objetivo de este proceso era el de crear un modelo en lo posible sin errores e inconsistencias que se adapte a los requerimientos del equipo de producto, de esta manera se logró crear un microservicio orientado al manejo y seguridad de los usuarios que se guio por el modelo que se creó en este paso.

A continuación, se mostrará el modelo realizado. Donde se explicará de forma general su funcionamiento y también cada uno de sus componentes.

Dado los requerimientos expuestos por el equipo de producto, el análisis realizado y la construcción de los casos de uso y tablas técnicas, se llegó al siguiente modelo como resultado:

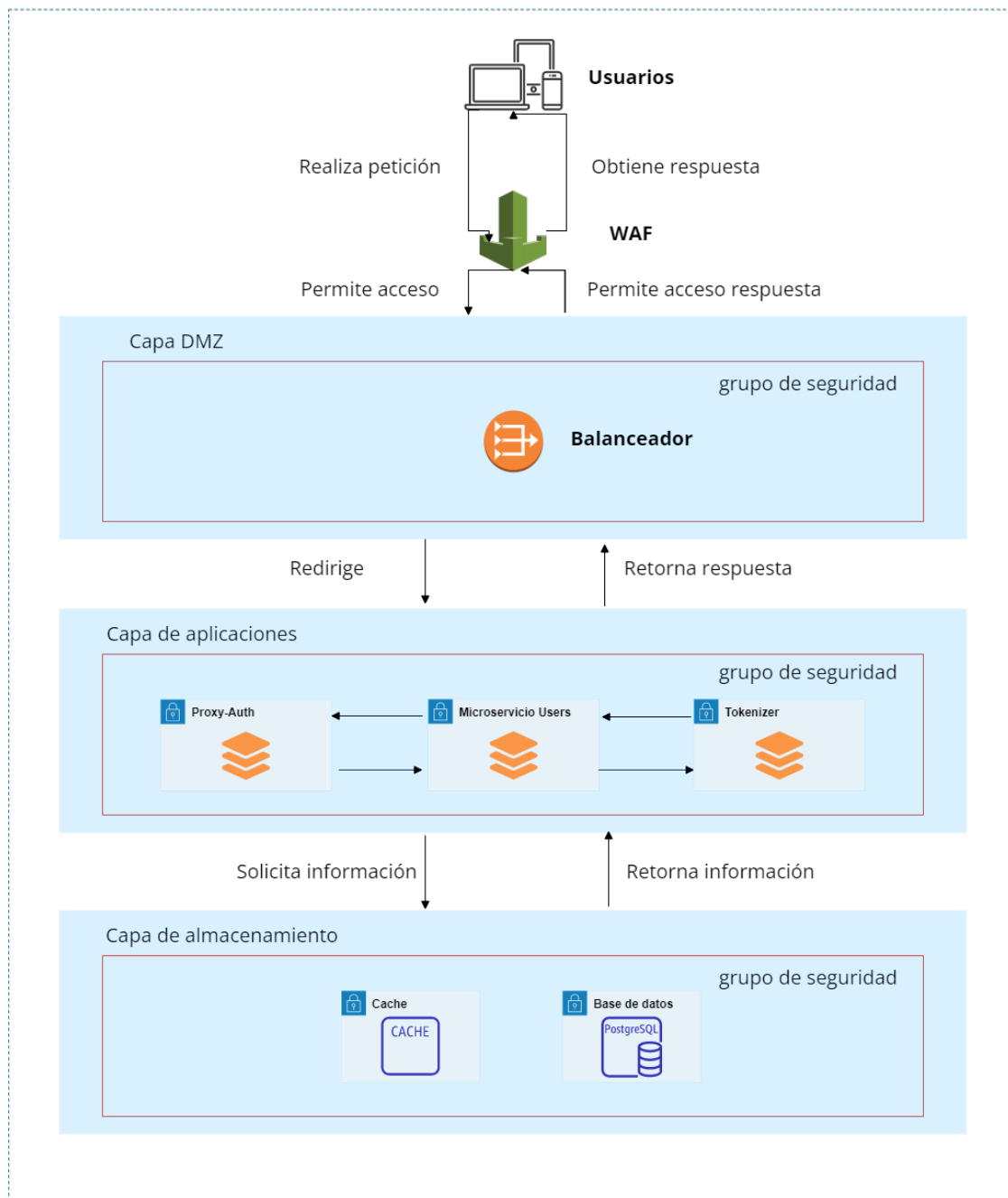


Ilustración 23 Modelo para el manejo y seguridad de los usuarios mediante el uso del microservicio Users. (Autoría propia)

Antes de explicar con más detalle el comportamiento interno de las capas, se debe entender de forma general el modelo.

Para el manejo y seguridad de los usuarios, se aplica el uso de diferentes tecnologías que se encargan de evitar filtraciones y ataques a la aplicación, de esta manera se puede garantizar la seguridad de la información de los usuarios.

Según la ilustración 21, el modelo tiene su inicio cuando un usuario realiza una petición dentro de la plataforma No-Code, la petición pasa por una tecnología integrada llamada WAF, este es un firewall de aplicación web que se encarga de verificar si las peticiones tienen algún tipo de ataque o algún componente que pueda venir con algo malicioso, este posee reglas y algoritmos que se crean para alertar el tráfico que está sospechoso. Al cumplir con estas reglas, la petición llega al balanceador de carga de la plataforma, que se dedica a tomar las peticiones por https y manda la redirección a la capa de aplicaciones, donde inicialmente pasa al Proxy-Auth, este valida las cookies o cualquier otro mecanismo que se encargue de la autenticación. Al validar la autenticación, traduce y genera un nuevo token y lo reenvía hacia el microservicio Users, además de conocer a cuál microservicio se debe enviar el token, realiza un balanceo de carga de cada uno de los componentes. Al llegar al microservicio Users, este puede enviar o recibir peticiones de otro microservicio, así como conectarse a la capa de almacenamiento que puede ser la base de datos del microservicio o al cache. Por último, el microservicio retorna una respuesta a la petición del usuario, que pasa por el proxy, este retorna la respuesta acompañada de un Token, luego pasa por el balanceador que finaliza el proceso entregando la respuesta al cliente con su respectivo Token.

En la explicación anterior se dio una idea general del funcionamiento del modelo, además de las tecnologías mencionadas, el modelo incluye capas donde se encuentran los componentes de este. Estas capas son redes privadas que están segmentadas para aislar ciertas partes del modelo. Esto quiere decir que cada red privada posee un grupo de seguridad cuya funcionalidad permite garantizar el control de acceso.

En la gráfica se puede observar el flujo de acceso entre las redes privadas. Este flujo limita la disponibilidad de información o servicios que posee una red, por tal motivo no se puede acceder directamente desde internet (usuarios) hasta la capa de almacenamiento o desde la capa

desmilitarizada (capa DMZ) hasta capa de almacenamiento, así como desde internet (usuarios) no se podría acceder directamente a los servicios de la capa de aplicación. Las bases de datos solo pueden ser accedidas si alguien tiene el grupo de seguridad de tipo capa de aplicación, y las aplicaciones solo pueden ser accedidas si tienen el grupo de seguridad de capa DMZ. Además de esto los grupos de seguridad funcionan como cortafuegos que se encarga de redireccionar el tráfico.

Capa desmilitarizada (capa dmz)

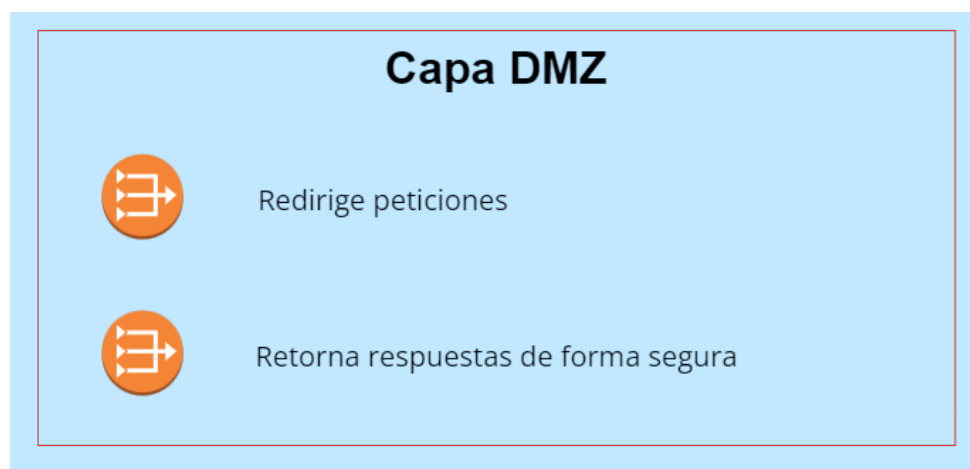


Ilustración 24 Capa DMZ del modelo. (Autoría propia)

Es una red pública para acceder a internet donde generalmente son puentes o servicios utilizados para permitir el acceso a las demás redes privadas. Dentro de esta se encuentra el balanceador de carga de la plataforma no-code. Para entender mejor el funcionamiento de esta capa un ejemplo sencillo es: que un usuario ya ha sido registrado en el sistema y se encuentra en la sección del perfil de usuario, este modifica su nombre y actualiza los datos. Cuando el usuario ejecuta dicha acción, empieza a ejecutarse el modelo, posterior la petición viaja al WAF donde envía dicha petición al balanceador de carga, si dicha petición no llega e intenta consumir algún servicio directamente a la capa de aplicaciones la petición será denegada. Esto se da debido a:

La capa desmilitarizada es la única capa a la que se puede acceder desde internet.

El balanceador que se encuentra en esta capa es el encargado de enviar las peticiones al microservicio o aplicación adecuado.

Si la petición no entra a la capa desmilitarizada, la petición no obtendrá el grupo de seguridad de dicha capa, por lo tanto, la petición no pasaría a la capa de aplicaciones debido a que esta capa requiere de un grupo de seguridad de la capa desmilitarizada.

Capa de aplicaciones

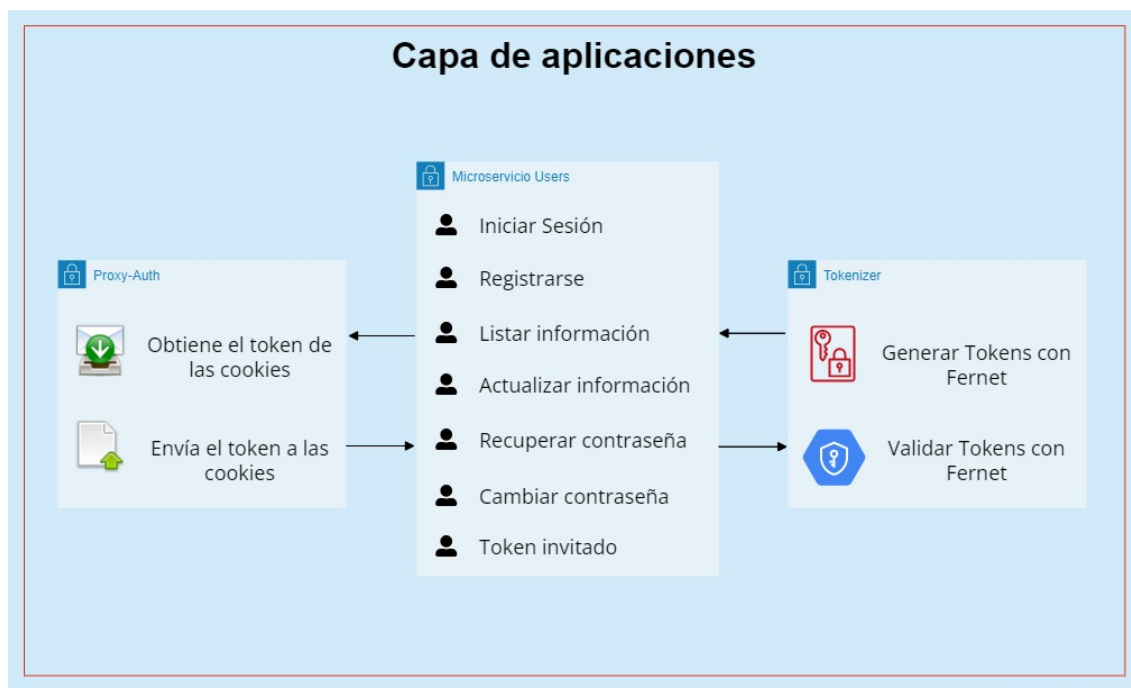


Ilustración 25. capa de aplicaciones del modelo. (Autoría propia)

Esta capa contiene los servidores donde se alojan los microservicios. Esta es la capa más importante, pues en esta se encuentra el microservicio Users, que se encarga de la validación, autenticación, manejo de la información y la obtención de los Tokens de los usuarios. El funcionamiento interno de esta, inicia en el Proxy-Auth, explicado anteriormente. Posteriormente pasa al microservicio Users, que se encarga de recibir y responder a las peticiones de los usuarios. El microservicio Users se conecta a otro microservicio llamado Tokenizer, que solo se encarga de

generar los Tokens para la autenticación de los usuarios. La forma en que es generado los tokens es por medio de la librería fernet-java8 que se utilizó con el lenguaje de programación kotlin, lo cual se implementó utilizando sus funcionalidades, por ejemplo, para generar un token se utiliza la siguiente instrucción `Token.generate(fernetKey, data)`, donde `fernetKey` es la clave con la que se puede desencriptar y `data` es la información del usuario que se quiere guardar en el token. Además, el microservicio users se conecta con la capa de almacenamiento, donde se encuentran las bases de datos y el cache de este, que se necesita para consultar la información de los usuarios requerida en sus servicios o funcionalidades (Véase en la ilustración 21).

Capa de almacenamiento

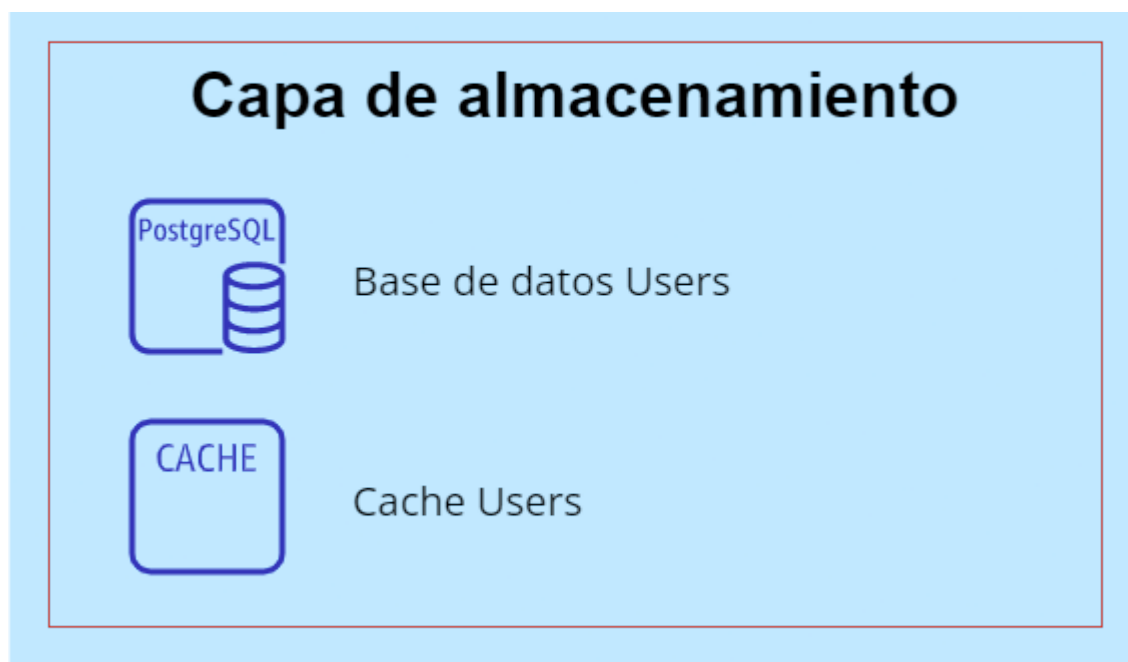


Ilustración 26. Capa de almacenamiento del modelo (Autoría propia)

Como se ve en la ilustración 24, en esta capa se encuentra el storage del microservicio users, distribuido por una base de datos y un cache, donde la base de datos es construida con el motor de PostgreSQL. Además, la cache es construida con el motor de base de datos Redis.

La capa de almacenamiento solo puede ser accedida desde la capa de aplicaciones, visto anteriormente, para comprender mejor, visualizar la ilustración 21.

Para entender mejor cómo funciona este flujo con el microservicio Users, a continuación, se va a explicar de forma textual el flujo del servicio token de invitado.

El usuario entra a la plataforma.

La plataforma detecta el ingreso, lo cual permite que se genere el envío de una petición al microservicio.

La petición se recibe en el balanceador de carga, este redirecciona la petición https al proxy.

El proxy recibe la petición, en este caso el proxy al verificar, sabe que este servicio es público, por tal motivo no verifica las cookies u otro mecanismo para la autenticación. Crea un token y reenvía la petición al microservicio Users con su token.

El microservicio users toma la petición y llama a otro microservicio nombrado Tokenizer, que se encarga de generar los tokens. Users recibe el token y lo inyecta en las cookies. Por último, envía de vuelta la respuesta, que en este caso son cookies.

El proxy recibe la respuesta y verifica la autenticidad. Al verificar envía la respuesta al balanceador de carga.

El balanceador recibe y retorna la respuesta al cliente, que en este caso son las cookies que se inyectan en el navegador del cliente.

3.6.1 Diagrama De La Conexión Entre Microservicios

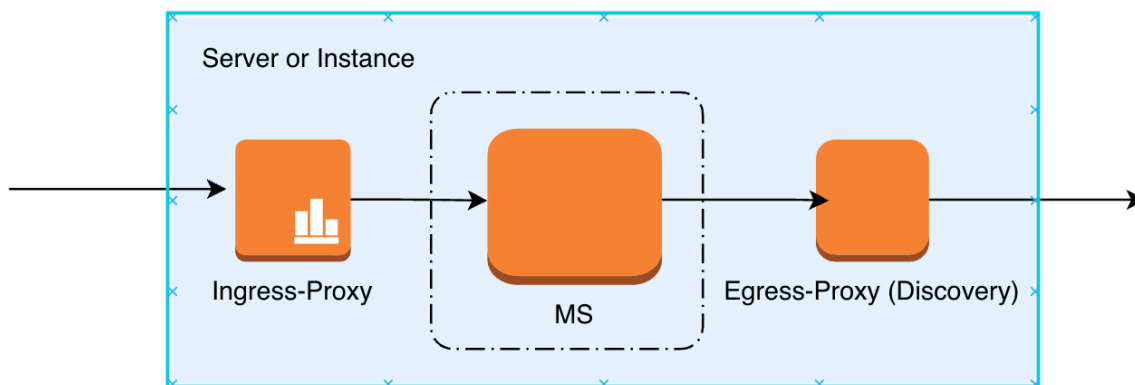


Ilustración 27 Diagrama de la conexión entre microservicios

Los microservicios son alojados en servidores de Amazon Web Services, a estos servidores se les pueden añadir reglas o componentes para aumentar la seguridad de la plataforma. Una de las reglas que se les implementan a estos, son dos proxys internos, proxy de ingreso (Ingress-Proxy) y un proxy de salida (Egress-Proxy). El primero se encarga de la autenticación del mecanismo utilizado para la seguridad, que en este caso son cookies que contienen tokens. Al verificar su autenticidad deja consumir los servicios de los microservicios. El segundo proxy se encarga de tomar las cookies que verifica el primer proxy, para posteriormente crear un nuevo token interno y retornarlo, además este proxy se encarga de saber la ruta o ubicación del microservicio al que quiere acceder o conectarse.

3.6.2 Diagrama de Clases

En el siguiente diagrama se especificó el diagrama de clases de la base de datos Users, que es parte fundamental del modelo realizado.

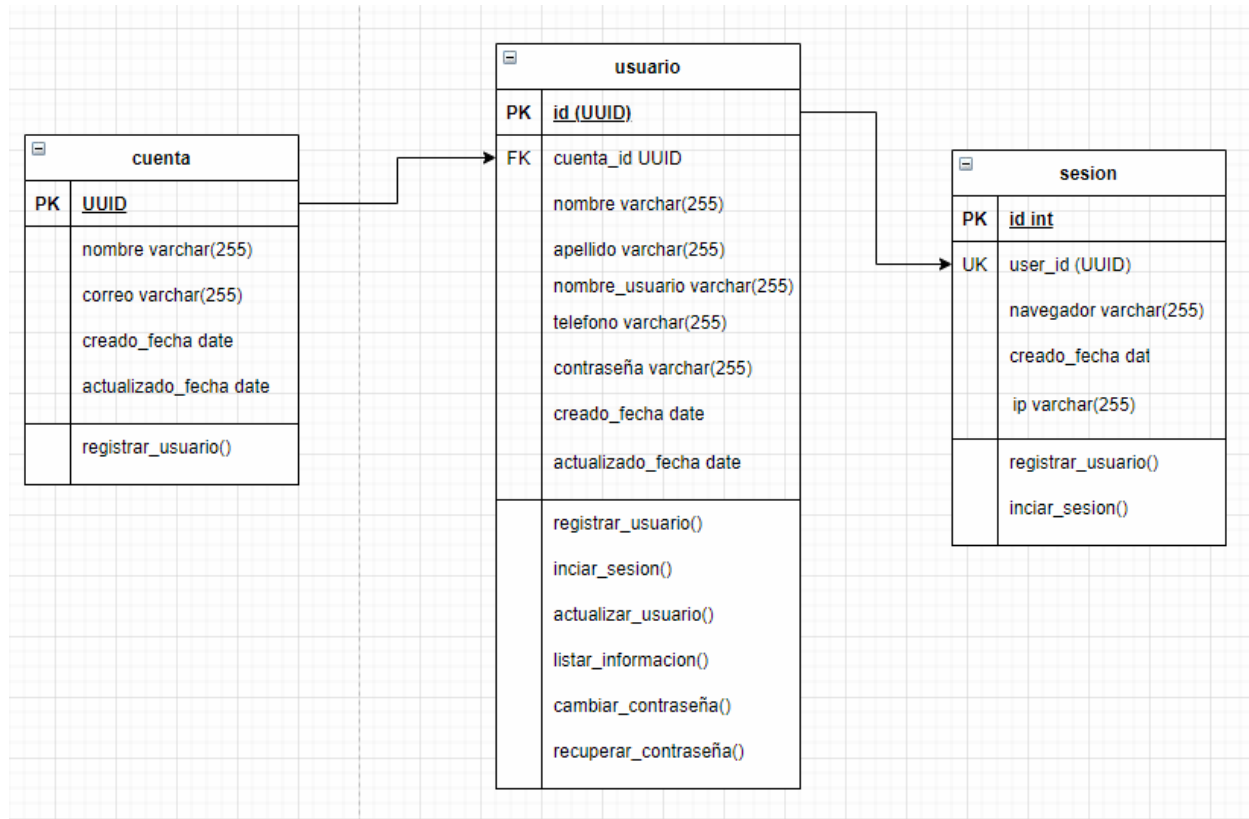


Ilustración 28 diagrama de clases

3.7 Validación del Modelo

Para la validación del modelo del microservicio users, se desarrolló la infraestructura y un microservicio cumpliendo la definición a cabalidad del modelo realizado, de esta manera se logró probar el modelo indirectamente por medio de pruebas de software que se le hicieron a cada uno de estos. Estas ocurrieron en dos momentos diferentes:

3.7.1 Durante el desarrollo

Las pruebas de validación realizadas durante este proceso se hicieron en dos situaciones diferentes:

Construcción de la arquitectura: Para implementar el microservicio users, se debía construir la infraestructura que soportaría el microservicio y parte de la seguridad de toda la plataforma. Una vez terminada la arquitectura, se hicieron las siguientes pruebas para validar su correcto funcionamiento:

- Prueba de rendimiento: Con ayuda del programa JMeter especializado en este tipo de seguimiento, se ejecutaron pruebas que albergaban desde 10.000 peticiones por minuto para medir el rendimiento de la arquitectura, donde se detectaban fallas para ser solucionadas prematuramente, con el fin de evitar estos inconvenientes en el ambiente de producción (ambiente real con usuarios).
- Prueba de seguridad (Accediendo a data store): Con la realización de esta prueba de seguridad, se verificó la protección y seguridad de los datos del usuario. Se llevó a cabo realizando una conexión a la data store del microservicio users, que retornaba un error de autorización debido al firewall implementado en la arquitectura. La única forma de acceder a esta información era por medio de una VPN que permitía conectarse a la red local de la infraestructura y así crear la conexión con la data store. Lo anterior dio como resultado el buen funcionamiento del mecanismo de seguridad (firewall) implementado.

Construcción del microservicio: Durante el proceso de desarrollo del microservicio se realizaron las siguientes pruebas para validar su correcto funcionamiento:

- Pruebas HTTP: Se realizaron pruebas a cada uno de los servicios del microservicio, con el fin de encontrar y solucionar errores de su funcionamiento por medio de llamadas HTTP a los endpoints de cada uno de estos, esta prueba se realizó con la ayuda del programa Postman, especializado en Api, donde se construye un llamado HTTP con gran facilidad.

- Pruebas de integración: La finalidad de estas pruebas de integración era de verificar el buen funcionamiento de cada servicio o funcionalidad del microservicio users, integrando todas las conexiones que se tenían con servicios externos. Se llevó a cabo con la ayuda del lenguaje Kotlin, que emplea funcionalidades para realizar las pruebas de integración.
- Pruebas unitarias: Con estas pruebas se verificaba el funcionamiento de la lógica de negocio de cada uno de los servicios o funcionalidades del microservicio users, es decir, solo se verificaba el funcionamiento interno de los servicios. Para poder llevar a cabo estas pruebas, se utilizó una funcionalidad específica del lenguaje Kotlin que permite imitar el comportamiento de las conexiones externas de forma controlada, un ejemplo sencillo en crear un usuario sería el siguiente: Para crear un usuario se requerían de parámetros como el nombre del usuario, apellido y correo, si en la prueba se ingresaba un parámetro mal, esta retornaba un error, ese es el ejemplo más claro de la validación de la lógica de negocio.

3.7.2 Antes del despliegue a producción

Este proceso de validación fue el más importante de la validación del modelo del microservicio users para el manejo y seguridad de los usuarios, ya que este proceso lo llevó a cabo un equipo especializado de la empresa donde se desarrolló las prácticas empresariales, llamado equipo QA. Este equipo tuvo la finalidad de implementar pruebas de software a la infraestructura y microservicio desarrollados a partir del modelo presentado y así validarlo de forma indirecta.

Los casos de uso y las tablas técnicas presentadas en la sección del desarrollo del modelo, fueron piezas claves para el equipo de QA, ya que gracias a estos anexos pudieron verificar el funcionamiento paso a paso de cada uno de los servicios del microservicio y de la seguridad general del modelo presentado. Las pruebas que se realizaron fueron con la ayuda del programa Postman, especializado en Apis y son las siguientes:

- Prueba HTTP (Crear usuario): Una de las funcionalidades más importantes del microservicio users es la de crear un usuario, ya que en esta prueba se llevaba a cabo la autenticación y verificación del usuario al ingresar a la plataforma, es por esto que el equipo de QA realizó múltiples pruebas para validar su correcto funcionamiento. Por ejemplo, intentar crear un usuario y después de crearlo verificar que se pueda acceder a las demás funcionalidades con el token que responde.
- Pruebas HTTP generales: El equipo de QA realizó pruebas http generales a los demás servicios o funcionalidades del microservicio users, donde verificó su correcto funcionamiento, así como la seguridad y el manejo de los usuarios.

Además de las pruebas anteriores, el equipo de QA realizó otras pruebas internas del microservicio users y la infraestructura de la plataforma, donde solo se compartió el resultado, pero no el procedimiento.

4 Conclusiones

De acuerdo con el estudio exhaustivo que se realizó en la sección vista del estado del arte, se pudo concluir que la mejor forma de manejar la seguridad de los usuarios en una plataforma No-Code es utilizando la arquitectura de microservicios, a partir de esto se inició con el diseño del modelo del microservicio users.

En virtud de lo expuesto por el equipo de producto de la empresa en la primera semana del offsite1, donde se explicó de forma general los requerimientos para el manejo y seguridad de los usuarios para la plataforma No-Code y posteriormente un análisis exhaustivo de estos requerimientos, se llegó como resultado final los requerimientos específicos del sistema para el desarrollo del modelo del microservicio users que se encargó del manejo y seguridad de los usuarios.

Tras la validación por el equipo de QA de la empresa, se llegó a la conclusión de que el microservicio users superó con éxitos los requerimientos proporcionados por esta, dando como resultado la aprobación para llevar a producción el microservicio.

Gracias a todo lo anterior, se puede interpretar que para el manejo y seguridad de los usuarios en una plataforma No-Code, la mejor arquitectura a desarrollar es la de microservicios, por lo tanto, se cumple con éxito el diseño y desarrollo del modelo del microservicio users y de esta manera dando por terminada satisfactoriamente la práctica empresarial.

5 Referencias Bibliográficas

Amazon Web Services, Inc. (2021). Amazon. Obtenido de <https://aws.amazon.com/es/microservices/#:~:text=Los%20microservicios%20son%20un%20enfoque,servicios%20son%20equipos%20peque%C3%B1os%20independientes.>

Aplyca. (2018). Aplyca. Obtenido de De sistemas monolíticos a microservicios: <https://www.aplyca.com/es/blog/aplicaciones-monoliticas-o-microservicios>

Atlassian. (2021). atlassian. Obtenido de <https://www.atlassian.com/es/software/bitbucket>

Atlassian. (2021). Atlassian. Obtenido de <https://www.atlassian.com/es/software/jira/guides/use-cases/what-is-jira-used-for#Jira-for-requirements-&-test-case-management>

auth0. (2021). JWT.IO. Obtenido de <https://jwt.io/introduction>

Barbosa, G. (14 de 09 de 2021). ituser.es. Obtenido de <https://www.ituser.es/opinion/2021/09/microservicios-la-clave-para-evolucionar-hacia-una-arquitectura-que-favorece-la-innovacion>

Bbva. (17 de 06 de 2020). bbva. Obtenido de <https://www.bbva.com/es/json-web-tokens-jwt-claves-para-usarlos-de-manera-segura/>

Blancarte, O. (s.f.). www.oscarblancarteblog.com. Obtenido de <https://www.oscarblancarteblog.com/tutoriales/java-persistence-api-jpa/>

Blakehead, T. (2017). Cómo Construir Microservicios: Los Diez Principales Trucos Para Modelar, Integrar Y Desplegar Microservicios

Cantó, J. C. (28 de 11 de 2018). Blib. Obtenido de <https://www.bilib.es/actualidad/blog/noticia/articulo/arquitectura-web-soa-vs-microservices-aplicaciones-y-diferencias/>

Ceupe (2019). Ceupe. Obtenido de <https://www.ceupe.com/blog/que-es-soa.html>

Ciberseguridad. (26 de 03 de 2020). Ciberseguridad. Obtenido de <https://ciberseguridad.com/servicios/algoritmos-cifrado/#AES>

Ciphr Team. (26 de 01 de 2021). Ciphr. Obtenido de <https://ciphr.io/blog/post/what-argon2-and-how-does-it-protect-my-data>

Cryptography. (2021). Cryptography. Obtenido de <https://cryptography.io/en/latest/>

Cvosoft It Academy. (16 de 10 de 2021). vvosoft. Obtenido de <https://www.scribbr.es/detector-de-plagio/generador-apa/new/webpage/>

David Roldán Martínez, P. J. (2020). Microservicios Un enfoque integrado. Ra-Ma.

Deloitte. (28 de 06 de 2018). Deloitte. Obtenido de <https://www2.deloitte.com/es/es/pages/technology/articles/que-es-orm.html>

Ecured. (2021). Ecured. Obtenido de https://www.ecured.cu/Enterprise_Architect

Fernandez, D. (31 de 07 de 2018). bigeek. Obtenido de <https://blog.bi-geek.com/introduccion-a-kotlin/>

Fernandez, M. (15 de 12 de 2020). Mailfence Blog. Obtenido de <https://blog.mailfence.com/es/cifrado-simetrico-vs-asimetrico/>

GaussWebApp. (02 de 10 de 2014). GaussWebApp. Obtenido de <https://gausswebapp.com/arquitectura-rest.html#:~:text=REST%20es%20una%20arquitectura%20de,utilizada%20en%20cualquier%20cliente%20HTTP.&text=Es%20decir%2C%20toda%20la%20informaci%C3%B3n,y%20almacenamiento%20interno%20del%20servidor.>

Geek, E. (15 de OCTUBRE de 2019). <https://ifgeekthen.nttdata.com/es/>

Gnatyk, R. (03 de 10 de 2018). n-ix. Obtenido de https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/?utm_source=xp&utm_medium=blog&utm_campaign=content

Gonzalo, F. (30 de 03 de 2021). ncoders.academy. Obtenido de <https://www.ncoders.academy/blog/que-es-nocode>

HostingPedia. (07 de 02 de 2019). hostingpedia.net. Obtenido de <https://hostingpedia.net/postgresql.html#:~:text=PostgreSQL%20es%20un%20gestor%20de,de%20forma%20libre%20y%20desinteresadamente>.

Insights, I. M. (2021). Microservices in the enterprise, 2021:. Obtenido de <https://www.ibm.com/cloud/learn/microservices>

Ionos Digitalguide. (04 de 09 de 2019). ionos. Obtenido de <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/spring-boot-tutorial/>

Irvingjuarez. (2021). Platzi. Obtenido de <https://platzi.com/tutoriales/1247-arquitectura-software/9250-que-es-un-microservicio/>

Juárez, I. (02 de 2021). Platzi. Obtenido de Platzi: https://platzi.com/tutoriales/1247-arquitectura-software/9250-que-es-un-microservicio/?utm_source=google&utm_medium=cpc&utm_campaign=12915366154&utm_adgroup=&utm_content=&gclid=aw.ds

Magaña, L. M. (03 de 12 de 2018). OpenWebinars.net. Obtenido de <https://openwebinars.net/blog/que-es-spring-framework/>

Mancomun. (17 de 11 de 2021). mancomun. Obtenido de

<https://www.mancomun.gal/es/solucion-tic/draw-io/>

Mayanga, C. (19 de 02 de 2020). carlosmayanga. Obtenido de

https://www.carlosmayanga.com/2020/02/que-es-smartgit-requisito-de-empleo_19.html

Openstack. (04 de 06 de 2020). Openstack. Obtenido de

<https://docs.openstack.org/keystone/pike/admin/identity-fernet-token-faq.html>

Perry, J. S. (11 de 05 de 2017). IBM Developer. Obtenido de

<https://developer.ibm.com/es/tutorials/j-spring-boot-basics-perry/>

Peter Smirmoff, D. T. (18 de 06 de 2019). cryptomathic. Obtenido de

[https://www.cryptomathic.com/news-events/blog/symmetric-key-encryption-why-where-and-how-its-used-in-](https://www.cryptomathic.com/news-events/blog/symmetric-key-encryption-why-where-and-how-its-used-in-banking#:~:text=Symmetric%20encryption%20is%20a%20type,encrypt%20and%20decrypt%20electronic%20information.&text=By%20using%20symmetric%20encryption%20al)

[banking#:~:text=Symmetric%20encryption%20is%20a%20type,encrypt%20and%20decrypt%20electronic%20information.&text=By%20using%20symmetric%20encryption%20al](https://www.cryptomathic.com/news-events/blog/symmetric-key-encryption-why-where-and-how-its-used-in-banking#:~:text=Symmetric%20encryption%20is%20a%20type,encrypt%20and%20decrypt%20electronic%20information.&text=By%20using%20symmetric%20encryption%20al)

Savvy Security. (19 de 06 de 2021). Savvy Security. Obtenido de

<https://cheapsslsecurity.com/blog/what-is-asymmetric-encryption-understand-with-simple-examples/#:~:text=Asymmetric%20Encryption%20C%20also%20known%20as,'%20and%20a%20'Private%20Key.>

Software Development Community. (15 de 02 de 2019). Medium. Obtenido de <https://medium.com/@SoftwareDevelopmentCommunity/what-is-service-oriented-architecture-fa894d11a7ec>

Sparxsystems. (2020). Sparxsystems.. Obtenido de <http://www.sparxsystems.com.ar/tic.portal>. (s.f.). tic.PORTAL. Obtenido de <https://www.ticportal.es/temas/cloud-computing/amazon-web-services>

Tyson, M. (02 de 04 de 2019). InfoWorld. Obtenido de <https://www.infoworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html>

Unir. (15 de 06 de 2021). Unir. Obtenido de <https://ecuador.unir.net/actualidad-unir/que-es-seguridad-informatica/>

Webempresa.Mx. (2021). Webempresa.Mx. Obtenido de <https://www.webempresa.mx/hosting/que-es-github.html>

Xplenty. (2021). Xplenty. Obtenido de <https://www.xplenty.com/glossary/what-is-monolithic-architecture/>