UNIVERSIDAD NACIONAL DEL SANTA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas e Informática

“Implementación de un API REST de notificaciones en tiempo real con gestión de fallos usando colas en la empresa TK Business Online SAC”

Trabajo de Suficiencia Profesional para Obtener el
Título Profesional de Ingeniero de Sistemas e Informática

Autor:

Bach. Cisneros Baca, Benny Lando

Asesor:

Ms. Macedo Alcántara, Dayan Fernando
DNI N°: 32941877
ORCID: 0000-0003-1190-4032

NUEVO CHIMBOTE – PERÚ
2025

UNIVERSIDAD NACIONAL DEL SANTA

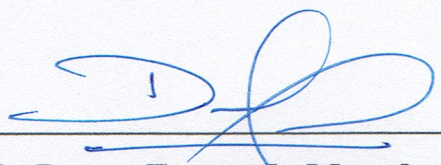
FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas e Informática

"Implementación de un API REST de notificaciones en tiempo real con gestión de fallos usando colas en la empresa TK Business Online SAC"

Trabajo de Suficiencia Profesional para Obtener el Título Profesional de Ingeniero de Sistemas e Informática

Revisado y Aprobado por el Asesor:



Ms. Dayan Fernando Macedo Alcántara

DNI N°: 32941877
ORCID: 0000-0003-1190-4032
ASESOR

NUEVO CHIMBOTE – PERÚ

2025

UNIVERSIDAD NACIONAL DEL SANTA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas e Informática

"Implementación de un API REST de notificaciones en tiempo real con gestión de fallos usando colas en la empresa TK Business Online SAC"

Trabajo de Suficiencia Profesional para Obtener el Título Profesional de Ingeniero de Sistemas e Informática

Revisado y Aprobado por el Jurado Evaluador:

Dr. Juan Pablo Sánchez Chávez
DNI N°: 17808722
ORCID: 0000-0002-3521-7037
PRESIDENTE

Ms. Camilo Ernesto Suarez Rebaza
DNI N°: 32978627
ORCID: 0000-0002-6870-4296
SECRETARIO

Ms. Dayan Fernando Macedo Alcántara
DNI N°: 32941877
ORCID: 0000-0003-1190-4032
INTEGRANTE

NUEVO CHIMBOTE – PERÚ

2025

FACULTAD DE INGENIERÍA
ESCUELA PROFESIONAL INGENIERÍA DE SISTEMAS E INFORMÁTICA

ACTA DE SUSTENTACIÓN INFORME DE TRABAJO DE SUFICIENCIA PROFESIONAL

A los dieciséis días del mes de julio del año dos mil veinticinco, siendo las 10:00 am. En el aula S-2 del Pabellón de la Escuela Profesional de Ingeniería Sistema e Informática-FI-UNS, se instaló el Jurado Evaluador designado mediante Resolución 170-2025-UNS-CFI, y de expedito según Resolución Decanal N° 438-2025-UNS-FI integrado por los docentes: : **Dr. Juan Pablo Sánchez Chávez (presidente), Ms. Camilo Ernesto Suarez Rebaza (secretario) y Ms. Dayan Fernando Macedo Alcántara (Integrante**), para dar inicio a la sustentación de la Tesis titulada **"IMPLEMENTACION DE UN API REST DE NOTIFICACIONES EN TIEMPO REAL CON GESTION DE FALLOS USANDO COLAS EN LA EMPRESA TK BUSINESS ONLINE SAC",** perteneciente al Bachiller: **CISNEROS BACA BENNY LANDO**, con código de matrícula N°201514054, quien fue asesorado por el **Ms DAYAN FERNANDO MACEDO ALCANTARA** , según Resolución Decanal N.º 680-2024-UNS-FI.

El Jurado Evaluador, después de deliberar sobre aspectos relacionados con el trabajo, contenido y sustentación del mismo, y con las sugerencias pertinentes en concordancia con el Reglamento General de Grados y Títulos, vigente, declaran aprobar:

BACHILLER	PROMEDIO VIGESIMAL	PONDERACIÓN
CISNEROS BACA BENNY LANDO	16	REGULAR

Siendo las 11 a m del mismo día, se dio por terminado el acto de sustentación, firmando la presente acta en señal de conformidad.

Nuevo Chimbote,16 julio de 2025

Dr. Juan Pablo Sánchez Chávez
PRESIDENTE

Ms. Camilo Ernesto Suarez Rebaza
SECRETARIO

Ms. Dayan Fernando Macedo Alcántara
INTEGRANTE

Recibo digital

Este recibo confirma quesu trabajo ha sido recibido por Turnitin. A continuación podrá ver la información del recibo con respecto a su entrega.

La primera página de tus entregas se muestra abajo.

Autor de la entrega:	Benny Lando Cisneros Baca
Título del ejercicio:	Tesis 2025
Título de la entrega:	TRABAJO_SUFICIENCIA.pdf
Nombre del archivo:	TRABAJO_SUFICIENCIA.pdf
Tamaño del archivo:	4.91M
Total páginas:	142
Total de palabras:	16,767
Total de caracteres:	99,875
Fecha de entrega:	16-mar-2026 08:47a. m. (UTC-0500)
Identificador de la entrega:	2904780852

UNIVERSIDAD NACIONAL DEL SANTA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas e Informática

UNS
UNIVERSIDAD NACIONAL DEL SANTA

"Implementación de un API REST de notificaciones en tiempo real con gestión de fallos usando colas en la empresa TK Business Online SAC"

Trabajo de Suficiencia Profesional para Obtener el Título Profesional de Ingeniero de Sistemas e Informática

Autor:

Bach. Cisneros Baca, Benny Lando

Asesor:

Ms. Macedo Alcántara, Dayan Fernando
DNI N°: 32941877
ORCID: 0000-0003-1190-4032

NUEVO CHIMBOTE – PERÚ
2025

9% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Texto citado
- Texto mencionado
- Coincidencias menores (menos de 15 palabras)

Exclusiones

- N.º de coincidencias excluidas

Fuentes principales

9% Fuentes de Internet
0% Publicaciones
0% Trabajos entregados (trabajos del estudiante)

Fuentes principales

Las fuentes con el mayor número de coincidencias dentro de la entrega. Las fuentes superpuestas no se mostrarán.

1	Internet	hdl.handle.net 4%
2	Internet	repositorio.uns.edu.pe 4%
3	Internet	alicia.concytec.gob.pe <1%
4	Internet	repositorio.unicesar.edu.co <1%
5	Internet	repositorio.upeu.edu.pe <1%
6	Internet	repositorio.upao.edu.pe <1%
7	Internet	cybertesis.unmsm.edu.pe <1%
8	Internet	ruc.udc.es <1%
9	Internet	repositorio.ucv.edu.pe <1%

AGRADECIMIENTO

Quisiera manifestar mi más profundo agradecimiento a:

A mis padres, cuyo respaldo incondicional
hizo posible que completara mis estudios
universitarios.

A los docentes de la EPISI, cuyas enseñanzas
y orientación me permitieron crecer
profesionalmente a lo largo de estos años.

INDICE GENERAL

LISTA DE TABLAS

LISTA DE FIGURAS

PRESENTACIÓN

En el competitivo mercado de las casas de cambio online en Perú, la rapidez y efectividad en la comunicación con clave para garantizar operaciones seguras y eficientes. TK Business Online SAC ha decidido fortalecer su infraestructura tecnológica mediante la implementación de un API REST de notificaciones en tiempo real con gestión de fallos utilizando colas, con el fin de mejorar la interacción entre sus sistemas internos como con sus clientes.

Con este propósito, la estructura del trabajo se presenta de la siguiente manera: En el Capítulo I, se expone el tema tratado; en el Capítulo II, se proporciona el contexto de la experiencia profesional; en el Capítulo III, se desarrolla la importancia para el ejercicio de la carrera profesional; en el Capítulo IV, se detallan los objetivos generales y específicos alcanzados; en el Capítulo V, se examina el sustento teórico del tema tratado; en el Capítulo VI, se registra la organización y sistematización de las experiencias logradas; en el Capítulo VII, se analiza la ubicación de las experiencias en el marco del sustento teórico; en el Capítulo VIII, se explican los aportes logrados para el desarrollo del centro laboral; en el Capítulo IX, de profundiza en los aportes para la formación profesional; y en el Capítulo X, se presentan las conclusiones y recomendaciones.

Esta implementación sirve como precedente para que empresas, especialmente casas de cambio online, mejoren la comunicación de sus sistemas, optimizando el envío de notificaciones en tiempo real y garantizando una experiencia más confiable para los usuarios.

RESUMEN

El presente trabajo de suficiencia profesional fue desarrollado en la empresa TK Business Online SAC, con el objetivo de implementar un API REST de notificaciones en tiempo real con gestión de fallos usando colas. Para ello se diseñó una arquitectura capaz de garantizar el envío de notificaciones ante la ocurrencia de errores de conectividad entre cliente y servidor. Se utilizó la metodología de desarrollo ICONIX para realizar el análisis y diseño del sistema. A continuación, el desarrollo del API se realizó utilizando: el lenguaje NodeJs con el framework NestJs, la librería BullMQ para gestionar las colas y MongoDB como base de datos no relacional. Finalmente se realizaron las pruebas correspondientes de los casos de uso para verificar su correcto funcionamiento.

Palabras clave: API REST, Notificaciones en tiempo real, Gestión de fallos, Colas de mensajes.

AUTOR:

- Bach. BENNY LANDO CISNEROS BACA

ASESOR:

- Ms. DAYAN FERNANDO MACEDO ALCANTARA

ABSTRACT

The present professional competency work was developed at the company TK Business Online SAC, with the objective of implementing a REST API for real-time notifications with fault management using queues. To achieve this, an architecture was designed to ensure the delivery of notifications in the event of connectivity errors between the client and server. The ICONIX development methodology was used for system analysis and design. Then, the API was developed using the NodeJs language with the NestJs framework, the BullMQ library for queue management, and MongoDB as a non-relational database. Finally, the corresponding tests for the use cases were carried out to verify their correct functioning.

Keywords: REST API, Real-time notifications, Fault tolerance, Message queues.

AUTHOR:

- Bach. BENNY LANDO CISNEROS BACA

ADVICER:

- Ms. DAYAN FERNANDO MACEDO ALCANTARA

I. TEMA ESPECÍFICO ABORDADO

IMPLEMENTACIÓN DE UN API REST DE NOTIFICACIONES EN TIEMPO REAL CON GESTIÓN DE FALLOS USANDO COLAS EN LA EMPRESA TK BUSINESS ONLINE SAC

II. CONTEXTUALIZACIÓN DE LA EXPERIENCIA PROFESIONAL

2.1. CRONOLOGÍA DE LA TRAYECTORIA PROFESIONAL

2.1.1. TK Business Online SAC

Es una casa de cambio online en Perú que ofrece servicios de cambio de divisas, como dólares a soles, al mejor tipo de cambio. Se destacan por su rapidez, seguridad y personalización en las transacciones. Además, están inscritos en la Superintendencia de Banca, Seguros y AFP, lo que garantiza su confiabilidad.

Tiempo en la empresa: Del 01/07/2021 al 28/02/2025

Puesto: Ingeniero de Software

2.1.1.1. Principales actividades

- Desarrollar interfaces de usuario intuitivas y funcionales, asegurando una experiencia óptima para los clientes.
- En el backend, implementar estructuras robustas y seguras que permitan un rendimiento eficiente, integrando servicios y bases de datos para un flujo continuo de información.
- Desarrollo y mantenimiento de la aplicación móvil para mejorar la experiencia de los usuarios.
- Configurar, mantener y monitorear los servidores para garantizar su rendimiento, seguridad y disponibilidad.
- Diseñar, implementar y optimizar las bases de datos para gestionar la información de manera eficiente.
- Resolver problemas técnicos y ofrecer orientación para el uso eficiente de plataformas

- Generar reportes y apoyar la toma de decisiones estratégicas.

2.1.1.2. Logros obtenidos

- Se implementó un rediseño del proceso de registro y operación separando perfiles persona y empresa.
- Se desarrolló la aplicación móvil para uso de los clientes usando el framework Flutter, el cual asegura la compatibilidad con dispositivos Android y iOS.
- Se implementó la integración con el servicio Host to Host de Interbank con la finalidad de automatizar las transferencias bancarias a las cuentas bancarias de los clientes.
- Se automatizó el proceso de compilación y publicación de la aplicación móvil mediante la plataforma Codemagic, lo cual permitió reducir el tiempo de desarrollo y despliegue.
- Se desarrolló una automatización para enviar notificaciones por Whatsapp cuando el cliente abandona una operación, lo cual permite retener e incentivar a los clientes a realizar su operación.

2.1.1.3. Aprendizaje

- Desarrollo y administración de sitios con Wordpress.
- Integración y despliegue continuo con Gitlab CI/CD.
- Desarrollo de aplicaciones móviles para Android y iOS usando el framework Flutter.
- Integración de Firebase Cloud Messaging para envío de notificaciones push.
- Desarrollo y gestión de base de datos MongoDB.
- Automatización de tareas con N8N.

2.2. CONTEXTO EN QUE SE DESARROLLÓ LA EXPERIENCIA

2.2.1. La empresa

2.2.1.1. TK Business Online SAC

Es una casa de cambio online en Perú que ofrece servicios de cambio de divisas, como dólares a soles, al mejor tipo de cambio. Se destacan por su rapidez, seguridad y personalización en las transacciones. Además, están inscritos en la Superintendencia de Banca, Seguros y AFP, lo que garantiza su confiabilidad.

2.2.1.2. Misión

Ofrecer servicios de cambio de divisas de manera rápida, segura y confiable, con las mejores tasas del mercado y una experiencia de usuario excepcional.

2.2.1.3. Visión

Ser la plataforma líder en cambio de divisas en Perú, reconocida por su innovación, transparencia y excelencia en el servicio al cliente, contribuyendo al crecimiento económico y financiero de nuestros usuarios y del país.

2.2.1.4. Ubicación

Av. La Molina Nro. 3365 Int. 110 Urb. Mástil de las Lagunas Lima - Lima - La Molina

Figura N° 1: Ubicación geográfica de TK Business Online SAC
Fuente: Google Maps

2.2.1.5. Logotipo

Figura N° 2: Logotipo de TKambio
Fuente: Sitio web de TKambio

2.2.1.6. Organigrama de TKambio

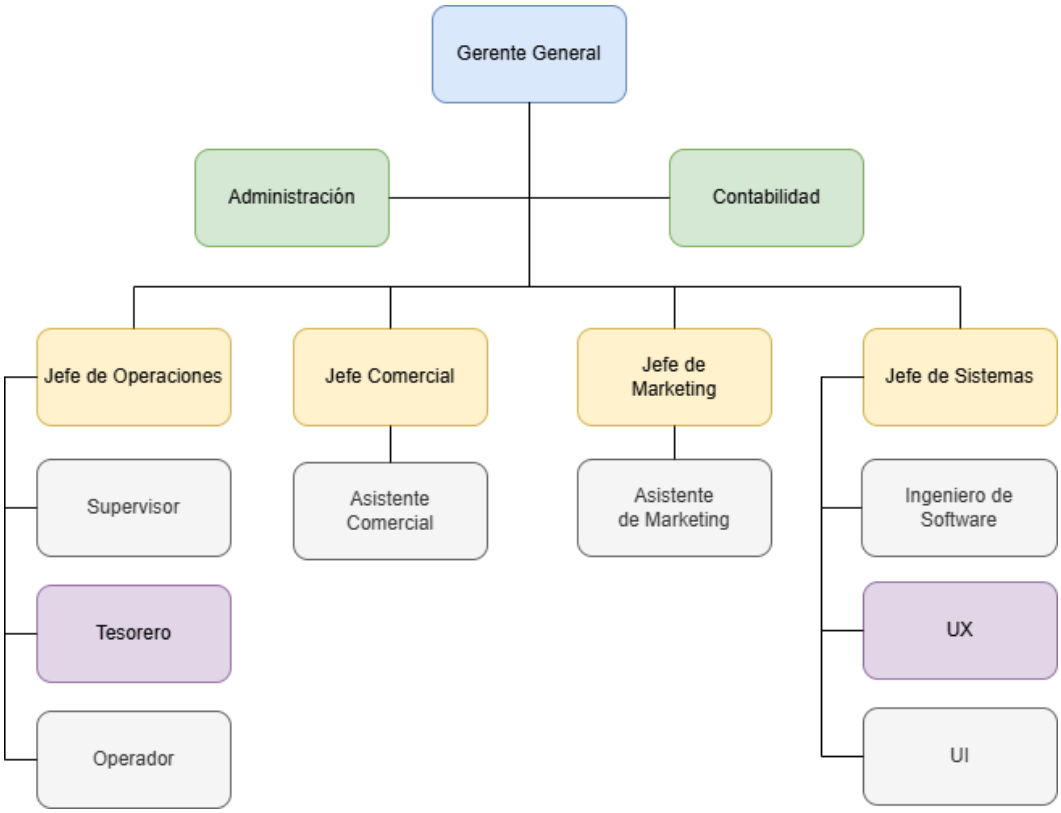


Figura N° 3: Organigrama de la TKambio
Fuente: Creado por el autor

2.2.2. Contexto del proyecto

En una casa de cambio online, la implementación de un sistema de notificaciones multicanal juega un rol crucial para optimizar la experiencia del cliente y aumentar la interacción con la plataforma. Este sistema permite informar a los usuarios sobre las fluctuaciones en el tipo de cambio y las promociones relevantes en tiempo real, incentivándolos a aprovechar oportunidades únicas para comprar o vender dólares. Los canales de notificación cumplen diferentes funciones según las necesidades de los clientes:

- **Websocket**: Ideal para notificaciones en tiempo real dentro de la plataforma web, ofreciendo a los usuarios actualizaciones instantáneas mientras navegan o realizan operaciones en el sitio.
- **Push notifications**: Estas son útiles para los clientes que usan la aplicación móvil, permitiendo alertas rápidas y personalizadas

directamente en sus dispositivos, incluso cuando no están activamente usando la app.

- **Email**: Este canal es perfecto para enviar información detallada y estructurada, como boletines con las mejores tasas de cambio o descuentos exclusivos, manteniendo a los clientes informados de manera profesional.

- **SMS**: Es una opción efectiva para mensajes concisos y urgentes, como el envío de un código de verificación al registrarse en la plataforma.

TK Business Online SAC, comprometida con la mejora constante de sus procesos y servicios tecnológicos, ha desarrollado e implementado un API REST que garantiza el envío de notificaciones en tiempo real con una gestión eficiente de fallos mediante el uso de colas. Este sistema permite a la empresa garantizar la entrega eficiente y segura de notificaciones a sus clientes, incluso ante eventuales interrupciones o errores en la comunicación. Con esta solución, se mejora la experiencia del cliente al mantenerlo informado en todo momento, reforzando la confianza y fidelidad hacia los servicios de la empresa.

Labor realizada: En este proyecto es fundamental la participación del Ingeniero de Software de TK Business Online SAC. Su conocimiento y experiencia en desarrollo de sistemas y en la integración de software será clave para garantizar la confiabilidad y eficiencia del sistema de notificaciones.

III. IMPORTANCIA PARA EL EJERCICIO DE LA CARRERA PROFESIONAL

Según la Universidad Nacional del Santa (2025), el egresado de la carrera de Ingeniería de Sistemas e Informática tiene las siguientes competencias: "Analiza, diseña, desarrolla, implementa y evalúa sistemas computacionales, para la solución de problemas empresariales utilizando con pertinencia herramientas de ingeniería de software, redes y comunicaciones, sistemas inteligentes, gestión de conocimientos, gestión de Tecnología de Información y Comunicaciones, y de gestión empresarial". Por lo tanto, la implementación del presente proyecto en una casa de cambio online tiene una gran importancia para el ejercicio de la carrera debido a los siguientes puntos clave:

- **Desarrollo de soluciones tecnológicas modernas:** Este proyecto permite aplicar conocimientos avanzados en el diseño e implementación de API REST, integrando múltiples canales como websocket, ush notifications, email y SMS, demostrando la capacidad del ingeniero para crear sistemas adaptados a las necesidades del mercado financiero.

- **Garantía de continuidad y confiabilidad:** Los mecanismos de gestión de fallos con colas aseguran la estabilidad y robustez del sistema, fortaleciendo habilidades esenciales en el manejo de arquitectura escalable y estrategias de recuperación ante errores.

- **Optimización de la experiencia del cliente**: La habilidad de implementar canales de comunicación efectivos impacta directamente en la satisfacción del cliente, elevando el valor estratégico del profesional en la creación de experiencias fluidas y confiables para los usuarios.

- **Visión empresarial y adaptabilidad**: Los ingenieros deben comprender las dinámicas del mercado financiero y las expectativas de los clientes para diseñar sistemas que no solo sean funcionales, sino que también impulsen la interacción comercial y las operaciones.

- **Enfoque en la seguridad de la información:** El manejo de datos sensibles en este tipo de sistemas subraya la importancia de garantizar la integridad, confidencialidad y disponibilidad de la información, una responsabilidad clave para el ingeniero de sistemas en cualquier proyecto relacionado con servicios financieros

- **Mejoras en procesos empresariales:** Este tipo de proyecto ayuda a agilizar procesos internos y a facilitar la toma de decisiones estratégicas en tiempo real, demostrando el impacto positivo de la tecnología en la operación empresarial.

IV. OBJETIVOS LOGRADOS

4.1. Objetivo General

Implementar un API REST de notificaciones en tiempo real con gestión de fallos usando colas en la empresa TK Business Online SAC.

4.2. Objetivos Específicos

- Garantizar una tasa de éxito de entrega de notificaciones superior al 99%.
- Mantener un tiempo promedio de espera en cola en el envío de notificaciones menor a 4 segundos.
- Asegurar un tiempo promedio de entrega de notificaciones menor a 5 segundos.

V. SUSTENTO TEÓRICO DEL TEMA ABORDADO

5.1. ANTECEDENTES

5.1.1. Investigaciones Internacionales:

Tesis 01

Autor: Mora Vega, Jeordy O.

Título: SISTEMA DE ADMINISTRACIÓN DE NOTIFICACIONES PARA LA AUTOMATIZACIÓN DE TAREAS EN LA EMPRESA ECOPETROL

Año: 2022

Resumen:

El proyecto se centró en crear un sistema de gestión de notificaciones para automatizar tareas en Ecopetrol. Utilizó herramientas de Microsoft Power Platform, como Power Apps, Power Automate y Power BI, para simplificar procesos manuales. La solución incluyó microservicios, un API centralizado y una base de datos en Dataverse, lo que permitió gestionar notificaciones y reportes en tiempo real de manera eficiente. Además, mediante la metodología ágil Scrum, se diseñó una solución escalable que mejora la trazabilidad,

minimiza errores y apoya la toma de decisiones con reportes dinámicos e interactivos.

Relación:

Esta tesis obtenida como antecedente de referencia tiene mucho en común con este proyecto, teniendo como relación que ambas tesis tienen el objetivo de realizar un sistema de información centralizado para la gestión de notificaciones minimizando errores.

Tesis 02

Autor: García, Andrés y Torres, Jefrey

Título: PROPUESTA DE MODELOS DE COMUNICACIÓN DESACOPLADOS PARA SISTEMAS DE INFORMACIÓN CON ARQUITECTURA ORIENTADA A MICROSERVICIOS

Año: 2024

Resumen:

Este proyecto aborda los retos de comunicación en sistemas desacoplados basados en arquitecturas de microservicios, a través de la implementación de un broker de mensajería dentro del proyecto de Renovación de Sistemas de Información (RSI). Para ello, se identificaron los requerimientos específicos, se evaluaron distintas tecnologías de mensajería y se seleccionó la más adecuada, desarrollando una solución personalizada que fue implementada y probada en dos microservicios clave. Las pruebas funcionales y de carga demostraron mejoras en escalabilidad, eficiencia y confiabilidad, concluyendo que una correcta elección tecnológica puede optimizar significativamente la comunicación asíncrona en entornos distribuidos.

Relación:

Esta tesis sirve como referente ya que demuestra cómo el uso de un broker de mensajería mejora la comunicación en sistemas distribuidos, permitiendo mayor escalabilidad y tolerancia a fallos. Su enfoque en microservicios y colas se alinea con los objetivos del presente proyecto.

5.1.2. Investigaciones Nacionales

Tesis 01

Autor: Dextre Pineda, Christian

Título: IMPLEMENTACIÓN DE UN PORTAL WEB Y MIGRACION AL MOTOR DE REGLAS PARA EL PROCESO DE NOTIFICACIONES AUTOMÁTICAS DEL SECTOR DE TELECOMUNICACIONES BAJO EL MARCO DE ARQUITECTURA SOA Y METODOLOGÍA SCRUM

Año: 2021

Resumen:

Este trabajo de suficiencia profesional presenta la implementación de un nuevo portal web dirigido a los usuarios de áreas corporativas, permitiéndoles configurar mensajes promocionales y códigos IPCC asociados. Además, se desarrolló un flujo de notificaciones para clientes corporativos, utilizando canales como SMS o llamadas de asesores, basado en reglas de negocio según el evento. También se diseñó un nuevo modelo de datos adaptable a herramientas BRMS, permitiendo a los usuarios modificar sus reglas sin intervención técnica, mejorando así la calidad del servicio y la gestión de promociones.

Relación:

Este trabajo se relaciona con el presente proyecto ya que aborda la implementación de flujos de notificación orientados a mejorar la comunicación con los usuarios mediante la automatización, la personalización de mensajes y la eficiencia en la entrega.

Tesis 02

Autor: Arias Mendoza, Angélica Milagros

Título: IMPLEMENTACIÓN DE UN SISTEMA INTEGRAL PARA LA MEJORA DEL PROCESO DE NOTIFICACIONES DE LA ENTIDAD RECAUDADORA DE TRIBUTOS E IMPUESTO DEL PERU, UTILIZANDO LAS TECNOLOGÍAS INFORMÁTICAS ECM/CCM

Año: 2021

Resumen:

Esta propuesta plantea la implementación de un sistema de notificaciones para mejorar el proceso de comunicación en una entidad recaudadora de impuestos, con el fin de enfrentar la baja en el cumplimiento de metas de recaudación. A pesar de inversiones en tecnología, el problema persistía debido a deficiencias en la definición y gestión del proceso de notificaciones. La solución consiste en integrar herramientas como ECM y CCM para optimizar la generación y distribución de documentos, mejorando así la eficiencia del proceso recaudador.

Relación:

Este trabajo sirve como antecedente para el presente proyecto porque demuestra cómo la implementación de un sistema de notificaciones puede resolver problemas críticos de comunicación dentro de una organización. Estos aprendizajes inspiraron el uso de tecnologías orientadas a la confiabilidad y automatización en la entrega de notificaciones.

Tesis 03

Autor: Willian Huamanttupa, Merling Ramirez y Yobel Cañazaca

Título: APLICACIÓN WEB PARA EL CONTROL DE LLEGADA DE CAMIONES Y LA EVALUACIÓN DE USABILIDAD DEL SOFTWARE BASADO EN ISO/IEC 25010

Año: 2023

Resumen:

Esta investigación desarrolló e implementó una aplicación web en tiempo real para el control de llegada de unidades vehiculares, utilizando MEAN Stack y la metodología ágil Scrum. Dirigida a organizaciones que aún gestionan procesos manualmente, la solución fue evaluada bajo la norma ISO/IEC 25010, destacando aspectos como usabilidad y accesibilidad. Con un nivel de confianza de 0.778, los resultados mostraron una alta satisfacción de los usuarios, concluyendo que la aplicación mejoró significativamente los procesos y la eficiencia en la gestión empresarial.

Relación:

Esta tesis se relaciona con el presente proyecto porque ambos buscan optimizar procesos mediante soluciones web en tiempo real. Además, el uso de tecnologías modernas y metodologías ágiles en la tesis sirve como referente para el desarrollo eficiente y centrado en el usuario del presente proyecto.

5.2. FUNDAMENTOS TEÓRICOS DE LA INVESTIGACIÓN

5.2.1. API

Una API, o interfaz para la programación de aplicaciones, consiste en un conjunto de reglas y especificaciones utilizadas para desarrollar e integrar el software de distintas aplicaciones. (RedHat, 2023).

En el ámbito de los servicios financieros, que abarcan préstamos, seguros, comercio electrónico, pagos, información bancaria, historiales de transacciones, autenticación y transferencias, las API abiertas desempeñan un papel clave al proporcionar acceso a estos datos a quienes lo requieran, agilizando así los procesos operativos (Cámara de Comercio de Bogotá, 2025).

5.2.1.1. Tipos de API

Según Nida (2024) hay diversas API, cada una diseñada con un propósito y funcionalidad específicos. Algunos de los tipos más frecuentes incluyen:

- **API RESTful**: Son APIs basadas en la web que emplean solicitudes HTTP para acceder y modificar datos. Se utilizan frecuentemente en el desarrollo web, permitiendo la transmisión de información de manera estandarizada y comprensible.
- **API SOAP**: Este tipo de API también opera en la web, pero utiliza XML para el intercambio de datos. Es común en aplicaciones empresariales y facilita interacciones avanzadas entre sistemas.
- **API GraphQL**: Es un lenguaje de consulta para APIs desarrollado por Facebook, que permite a los desarrolladores especificar la estructura de datos que necesitan y recibir solo la información solicitada.
- **API abiertas**: Son APIs públicas accesibles por cualquier usuario. Se usan principalmente para la integración de aplicaciones de terceros con servicios existentes.
- **API internas**: Funcionan dentro de una organización, facilitando la comunicación entre distintos departamentos o aplicaciones internas.

- **API de socios**: Están diseñadas para permitir la interacción entre empresas y sus socios o proveedores.

5.2.2. REST

Según Tomás (2022), REST (Transferencia de Estado Representacional) es una arquitectura que se utiliza principalmente sobre el protocolo HTTP para la solicitud de recursos, en lugar de operaciones o mensajes, como lo hace el protocolo SOAP.

AppMaster (2023) añade que "REST fomenta el intercambio de datos entre el cliente y el servidor mediante representaciones de estado. Esto brinda flexibilidad y permite una integración sencilla con distintas aplicaciones y plataformas".

Tomás (2022) señala que los principios de REST son:

- La transmisión de datos se realiza a través de HTTP, empleando sus operaciones principales: GET, POST, PUT y DELETE.
- Los servicios se acceden mediante un espacio de URI unificado, donde cada URI representa un recurso en Internet. Este enfoque ha demostrado ser eficaz, simple y versátil, contribuyendo significativamente al éxito de la Web.
- La identificación del formato de los datos se establece mediante tipos MIME, como text/html o image/gif, siendo XML (text/xml) el estándar de codificación más utilizado.

Por su parte, AppMaster (2023) añade los siguientes lineamientos para la denominación y estructuración de recursos:

- **Sustantivos para recursos:** Los nombres de los recursos deben representarse con sustantivos, no con verbos, para reflejar entidades y no acciones.
- **Nombres simples y descriptivos**: Los recursos deben tener nombres claros y comprensibles que indiquen su propósito de forma precisa.

- **Pluralidad en colecciones**: Los nombres de recursos que representan conjuntos deben ir en plural para indicar que se trata de una colección de elementos.
- **Anidamiento para relaciones**: Las estructuras jerárquicas entre recursos deben representarse mediante rutas anidadas que reflejen sus relaciones de forma lógica y ordenada.

Las ventajas de REST se deben a su diseño simple. Sus beneficios incluyen tiempos de respuesta más eficientes y menos trabajo para el cliente y el servidor. Además, ofrece mayor estabilidad ante modificaciones posteriores y facilita el desarrollo de clientes, ya que solo necesitan realizar peticiones HTTP y codificar datos en XML (Tomás & Lloret, 2022).

5.2.3. JSON

Según Flanagan (2020), JSON, que corresponde a "JavaScript Object Notation", es un formato que emplea la sintaxis literal de objetos y arreglos del lenguaje JavaScript para transformar estructuras de datos en forma de objetos y listas en cadenas de texto. Este formato permite representar valores primitivos como números, strings, true, false y null, así como estructuras más complejas formadas a partir de esos mismos valores.

JSON se utiliza ocasionalmente como un formato de archivo de configuración legible por humanos. Su sintaxis es un subconjunto muy estricto de JavaScript: no admite comentarios y los nombres de las propiedades deben ir entre comillas dobles, incluso en situaciones en las que JavaScript no lo requiere (Flanagan, 2020).

5.2.4. NOTIFICACIÓN

En el contexto de la informática, un sistema de notificación integra software y hardware para enviar mensajes a un grupo de destinatarios. Generalmente, refleja actividad vinculada a una cuenta y es un componente esencial en las aplicaciones web actuales (Academia Lab, 2025).

Según EcuRed (2020) el uso de las tecnologías de notificaciones se dan en los siguientes escenarios:

- **Aplicaciones e-commerce**: Una tienda en línea envía notificaciones en tiempo real a los usuarios para informar sobre el estado de sus pedidos, promociones exclusivas o carritos abandonados, mejorando la experiencia del cliente y aumentando las conversiones.
- **Aplicaciones bancarias**: Los sistemas bancarios notifican de inmediato a los usuarios sobre movimientos en sus cuentas, transacciones sospechosas o vencimientos de pagos, brindando seguridad y control financiero.
- **Sistemas de monitoreo de servidores**: Herramientas como Prometheus envían notificaciones automáticas a administradores cuando se detectan fallos, caídas o picos inusuales en los servidores, permitiendo una respuesta rápida.
- **Gestión de turnos en salud**: Plataformas médicas notifican a los pacientes sobre sus citas, recordatorios de tratamientos o cambios de horario, optimizando la asistencia y reduciendo ausencias.

EcuRed (2020) señala que la estructura fundamental de un sistema de notificaciones se compone de los siguientes elementos esenciales:

- **Emisor u origen del evento**: Es el sistema, servicio o aplicación que detecta una condición o evento relevante que debe ser comunicado.
- **Gestor de notificaciones**: Procesa los eventos, genera las notificaciones y gestiona su envío según reglas definidas.
- **Canal de entrega**: Medio por el cual se transmite la notificación (correo, SMS, push, etc.).
- **Receptor o destinatario**: Usuario o sistema que recibe la notificación final
- **Registro y monitoreo**: Guarda el historial de notificaciones y permite supervisar su estado (entregado, fallido, pendiente).

5.2.5. WEBSOCKET

Según AppMaster (2023) “El protocolo WebSocket es un mecanismo de comunicación en tiempo real que permite el intercambio bidireccional de datos entre un cliente y un servidor mediante una única conexión persistente. A diferencia del HTTP tradicional, WebSocket admite una comunicación full-duplex, lo que significa que los datos pueden enviarse y recibirse simultáneamente en ambas direcciones, optimizando el rendimiento de la red y la eficiencia de las aplicaciones”.

5.2.5.1. Etapas de la comunicación

Según Thierry (2023) la comunicación mediante WebSocket se desarrolla en tres fases principales:

1. El proceso comienza con el handshake, donde el cliente inicia la conexión y ambas partes intercambian información de identificación para establecer la comunicación.
2. En la segunda etapa, la comunicación se establece completamente, permitiendo que ambas partes intercambien datos de manera independiente y simultánea.
3. El proceso finaliza cuando cualquiera de las partes decide terminar la comunicación y cerrar la conexión.

5.2.5.2. Casos de uso

AppMaster (2023) señala que WebSocket se utiliza en los siguientes escenarios:

- **Aplicaciones de chat**: permiten enviar y recibir mensajes de texto en forma instantánea sin necesidad de actualizaciones manuales ni consultas constantes.
- **Notificaciones en tiempo real**: envían alertas instantáneas sobre eventos como correos electrónicos, actualizaciones de tareas o interacciones en sistemas colaborativos.

- **Herramientas de colaboración en vivo**: posibilitan la edición simultánea de documentos, hojas de cálculo o presentaciones, facilitando el trabajo en equipo y el control de versiones.

5.2.6. NOTIFICACIÓN PUSH

Según (Tomás y otros, 2019), "es un tipo de comunicación entre el servidor y el cliente en donde el servidor es el que realiza la notificación en el momento en el que ocurre el evento, mientras que el cliente solo espera a recibir mensajes sin necesidad de consultar al servidor constantemente".

5.2.6.1. Servicio Google Cloud Messaging

De acuerdo con (Tomás y otros, 2019), es una plataforma que facilita a los desarrolladores la transferencia de datos desde sus servidores hacia sus aplicaciones en dispositivos Android.

Según (Tomás y otros, 2019), las notificaciones push con GCM requieren de tres actores:

- **Servicio GCM**: Su función será registrar el dispositivo para que pueda recibir notificaciones de nuestro proyecto, proporcionando un identificador de registro y activando el servicio. En cuanto dar de baja al dispositivo, esta acción evitará que reciba más notificaciones. Además, actuará como intermediario en el envío de mensajes, encargándose de la transmisión y gestión de la cola de mensajes. Este actor es administrado por Google.

- **Aplicación Android**: Se encargará de inscribir y retirar el dispositivo en el GCM, además de gestionar la recepción de notificaciones.

- **Servidor web**: Se encargará de la transmisión de notificaciones, requiriendo la gestión de un almacenamiento con los identificadores de los dispositivos registrados en el proyecto.

5.2.7. COLAS

Una cola es una estructura de datos que organiza los elementos de manera ordenada, donde las eliminaciones ocurren en un extremo denominado frente, y las incorporaciones de nuevos elementos se realizan en el otro extremo,

llamado fondo. En este sistema, el primer elemento en ingresar es también el primero en salir, por lo que se conoce como una lista FIFO (First In, First Out) (Joyanes & Zahonero, 1998).

Muchos sistemas del mundo real, tanto físicos como lógicos, siguen el modelo de cola. Ejemplos de ello incluyen la gestión de trabajos de impresión en un servidor, la asignación de prioridades en viajes, el funcionamiento de programas de simulación y la administración en sistemas operativos. Además, las colas se emplean comúnmente como buffers de datos, facilitando la transferencia de información entre componentes de distinta velocidad, como de la memoria de una computadora a una impresora (Joyanes & Zahonero, 1998).

Las colas pueden resolver muchos problemas diferentes de una manera elegante, desde suavizar los picos de procesamiento hasta crear canales de comunicación sólidos entre microservicios o descargar el trabajo pesado de un servidor a muchos procesos más pequeños, y muchos otros casos de uso (Taskforce.sh Inc., 2025).

5.2.8. BULLMQ

BullMQ es una librería de NodeJs que implementa un sistema de colas rápido y robusto construido sobre Redis para ayudar a implementar muchas arquitecturas de microservicios actuales (Taskforce.sh Inc., 2025).

La librería está diseñada para cumplir los siguientes objetivos:

- La semántica de la cola exactamente una vez, es decir, intenta entregar cada mensaje exactamente una vez, pero entregará al menos uno en el peor de los casos.
- Fácil de escalar horizontalmente, agregando más workers para procesar jobs en paralelo.
- Consistencia.
- Alto rendimiento. Intenta obtener el mayor rendimiento posible de Redis mediante la combinación de scripts .lua eficientes y canalización.

BullMQ se basa en 4 clases que, juntas, se pueden utilizar para resolver muchos problemas diferentes. Estas clases son Queue, Worker, QueueEvents y FlowProducer.

La clase Queue representa una cola y se puede usar para agregar jobs a la cola, así como para alguna otra manipulación básica, como pausar, limpiar u obtener datos de la cola.

Los jobs son estructuras de datos definidas por el usuario y almacenadas en una cola. Estos jobs son gestionados por workers, instancias encargadas de su procesamiento. Se pueden ejecutar múltiples workers, ya sea dentro del mismo proceso de Node.js, en procesos independientes o incluso en distintas máquinas. Todos los workers consumen jobs de la cola y registran su estado como completado o con error.

5.2.9. NOSQL

Según Bender y otros (2014), el término NoSQL: "se emplea para clasificar sistemas de gestión de bases de datos que difieren de los modelos relacionales tradicionales. Su evolución ha sido impulsada por tendencias como Big Data, Big Users y Cloud Computing, que han fomentado el avance de esta tecnología".

Parker y otros (2013) afirman que: "las bases de datos NoSQL surgieron para satisfacer las demandas de la era digital, facilitando la gestión de grandes volúmenes de información en aplicaciones con millones de usuarios diarios. Estas bases capturan datos personales, redes sociales, contenido generado por usuarios y datos geolocalizados. Se caracterizan por no requerir esquemas fijos, ser fáciles de instalar, emplear lenguajes no declarativos, ofrecer alto rendimiento y disponibilidad, evitar operaciones de junturas, soportar paralelismo y escalar horizontalmente en estructuras distribuidas".

Según Bender y otros (2014) las bases de datos NoSQL se pueden clasificar en:

- **Almacenamientos clave-valor**: El modelo de datos utiliza pares clave-valor, organizados en una tabla hash donde cada clave apunta al objeto correspondiente. Su diseño es simple y eficiente,

permitiendo escalabilidad y tolerancia a fallos mediante la distribución de registros y replicación. Además, las transacciones emplean registros simples, evitando la necesidad de protocolos complejos.

- **Almacenamientos de documentos**: El modelo de datos se basa en tuplas formadas por una clave y un documento. Aunque guarda similitud con el almacenamiento clave-valor, exige que los datos del documento sigan un formato como JSON, XML u otros formatos semi estructurados. Los documentos suelen consistir en colecciones clave-valor con valores anidados relacionados con cada clave.

- **Sistemas de base de datos de grafos**: En estos sistemas el modelo de datos está compuesto por aristas y nodos. Los nodos contienen propiedades, mientras que las aristas contienen etiquetas y definen roles. La información en los grafos se guarda en tuplas con diversos atributos.

5.2.10. MONGO DB

MongoDB es un sistema de gestión de base de datos de tipo NoSQL de código abierto y gratuito, diseñada para el almacenamiento de documentos. Utiliza JSON como formato de datos, pero los almacena en una versión binaria llamada BSON. Cada documento puede compararse con una fila de una tabla relacional y admite valores anidados con profundidad variable (Bender y otros, 2014).

Según Banker y otros (2016) las principales características de MongoDB son las siguientes:

- **Modelo de datos de documento**: La información se almacena en documentos con formato BSON, los que a su vez se agrupan en colecciones. Un documento no tiene una estructura fija, así que sus atributos pueden cambiar sin afectar a otros documentos.

- **Consultas ad hoc**: Permite realizar consultas flexibles sin necesidad de definir de antemano qué consultas aceptará la base de datos.

- **Índices**: Permite crear índices B-tree en los campos de los documentos, con el fin de realizar consultas rápidamente.
- **Réplicas**: Mediante la replicación de los datos en nodos secundarios, la base de datos puede mejorar la velocidad de lectura de datos y recuperarse de fallos automáticamente.
- **Velocidad y durabilidad**: Brinda un buen balance entre velocidad de escritura y la garantía de escribir permanentemente los datos en disco, incluso luego de una caída abrupta en el servidor.
- **Escalamiento**: Permite escalar la base de datos de forma horizontal de manera transparente y con recuperación automática ante fallos.

Banker y otros (2016) afirma que los principales casos de uso de MongoDB son los siguientes:

- **Aplicaciones web**: MongoDB es una opción adecuada para sitios web con una gran cantidad de tráfico. Además, puede escalar fácilmente conforme la aplicación web crece.
- **Desarrollo ágil**: Permite ahorrar tiempo de desarrollo ya que no es necesario estructurar la información de antemano como pasa con una base de datos relacional.
- **Analítica y logs**: Las actualizaciones atómicas permiten incrementar contadores y agregar valores a un array de forma más eficiente, lo que es útil para analítica. Por otro lado, almacenar logs en una base de datos en lugar de archivos, provee una mejor organización y facilita las consultas.
- **Caché**: MongoDB cuenta con una alta velocidad de lectura y los documentos permiten crear estructuras más complejas en comparación a otros sistemas de caché.
- **Estructura variable**: Es posible consumir un API JSON y almacenar directamente esa respuesta como un documento ya que no es necesario conocer la estructura de esos datos.

VI. ORGANIZACIÓN Y SISTEMATIZACIÓN DE LAS EXPERIENCIAS LOGRADAS

6.1. ANÁLISIS DE REQUISITOS

El primer paso en la metodología ICONIX es la creación del "modelo de dominio"; sin embargo, antes de ello, es necesario elaborar un listado de requisitos, ya que estos constituyen la base principal para su desarrollo. En esta sección se detallará la especificación de los requisitos funcionales y no funcionales del sistema a implementar.

6.1.1. Requisitos Funcionales

RF001: Para utilizar la API el usuario debe autenticarse mediante su nombre de usuario y contraseña.

RF002: El administrador será capaz de registrar un usuario.

RF003: El administrador será capaz de modificar un usuario.

RF004: El administrador será capaz de consultar un usuario.

RF005: El administrador será capaz de listar usuarios.

RF006: El administrador será capaz de eliminar a un usuario.

RF007: El usuario será capaz de registrar un cliente.

RF008: El usuario será capaz de modificar un cliente.

RF009: El usuario será capaz de consultar un cliente.

RF010: El usuario será capaz de listar clientes.

RF011: El usuario será capaz de reiniciar la clave de acceso de un cliente.

RF012: El usuario será capaz de eliminar un cliente.

RF013: Para enviar y recibir notificaciones el cliente debe autenticarse mediante su identificador y clave de acceso.

RF014: El usuario será capaz de registrar una plantilla de notificación.

RF015: El usuario será capaz de modificar una plantilla de notificación.

RF016: El usuario será capaz de consultar una plantilla de notificación.

RF017: El usuario será capaz de listar plantillas de notificación.

RF018: El usuario será capaz de eliminar una plantilla de notificación.

RF019: El usuario será capaz de registrar una etiqueta.

RF020: El usuario será capaz de modificar una etiqueta.

RF021: El usuario será capaz de consultar una etiqueta.

RF022: El usuario será capaz de listar etiquetas.

RF023: El usuario será capaz de eliminar una etiqueta.

RF024: El cliente será capaz de enviar una notificación.

RF025: El cliente será capaz de establecer conexión con el servidor de websockets.

RF026: El usuario será capaz de cancelar el envío de una notificación.

RF027: El usuario será capaz de consultar una notificación.

RF028: El usuario será capaz de listar notificaciones.

RF029: El usuario será capaz de eliminar una notificación.

RF030: El sistema será capaz de reintentar el envío de una notificación fallida.

RF031: El administrador será capaz de listar las configuraciones.

RF032: El administrador será capaz de actualizar una configuración.

6.1.2. Requisitos no Funcionales

- RNF01: La API debe seguir estándares como RESTful para garantizar interoperabilidad.
- RNF02: La API debe contar con un sistema de logging para registrar las solicitudes y respuestas.
- RNF03: Toda la comunicación debe realizarse sobre HTTPS para asegurar la privacidad de los datos.
- RNF04: Los errores deben ser manejados de manera uniforme, devolviendo códigos HTTP estándar.
- RNF05: Debe ser posible agregar nuevos endpoints sin interrumpir los servicios existentes.
- RNF06: La latencia máxima para una solicitud no debe exceder los 200 ms bajo cargas normales.
- RNF07: La API debe implementar autenticación y autorización utilizando el protocolo JWT.

6.1.3. Modelado del Dominio

Relación de potenciales objetos o clases de dominio

- **Cliente:** Aplicación o sistema que hace solicitudes a la API.
- **Configuración:** Parámetros que determinan el funcionamiento del envío de notificaciones.
- **Etapa de Notificación:** Cada uno de los estados del envío de la notificación.
- **Etiqueta:** Identificador o marcador usado para categorizar, organizar o filtrar notificaciones según sus atributos o propósito.
- **Notificación:** Mensaje o alerta enviada para informar a clientes o sistemas sobre un evento o cambio específico.
- **Plantilla:** Formato predefinido utilizado para estructurar y personalizar el contenido de las notificaciones enviadas.
- **Usuario:** Persona que interactúa con la API.

6.1.4. Modelo de Dominio Inicial

En el siguiente diagrama se muestra el Dominio Inicial del sistema propuesto

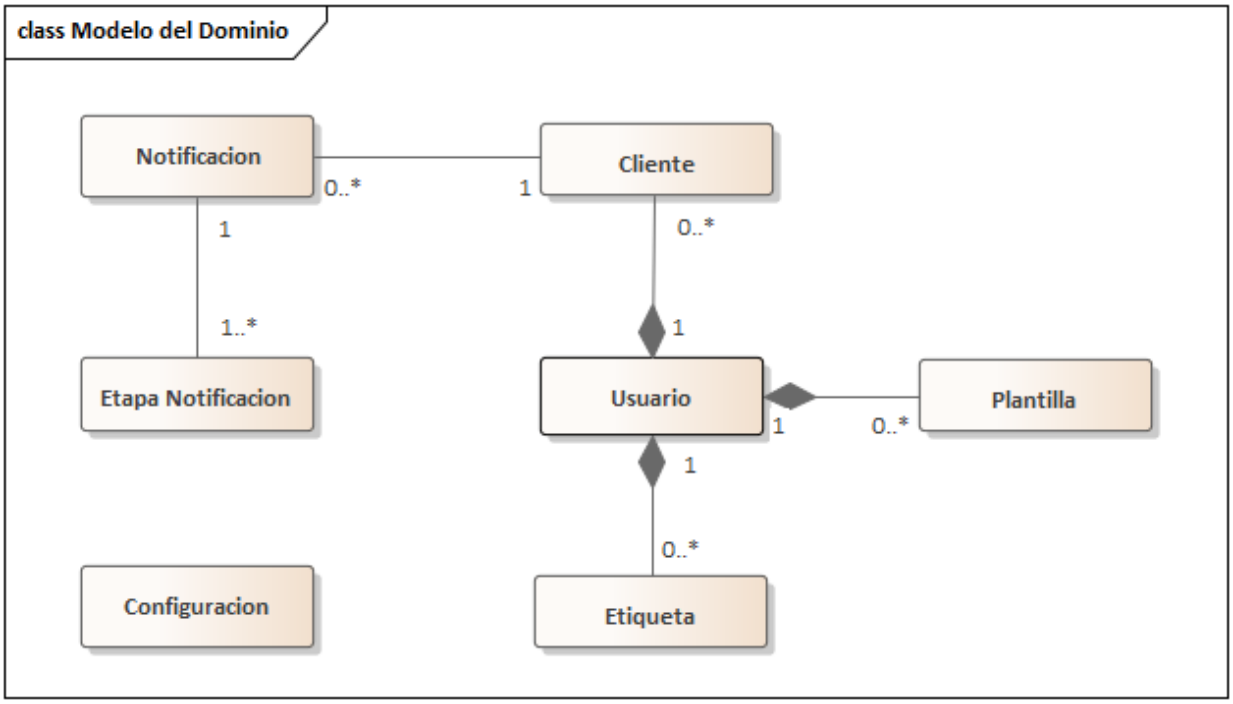


Figura N° 4: Modelo del Dominio
Fuente: Creado por el autor

6.1.5. Modelado de Casos de Uso

6.1.5.1. Prototipo de Interfaz de Usuario

Para identificar los casos de uso, se desarrollaron diversos esquemas de prototipos de interfaz de usuario, los cuales servirán de referencia para la especificación de dichos casos de uso.

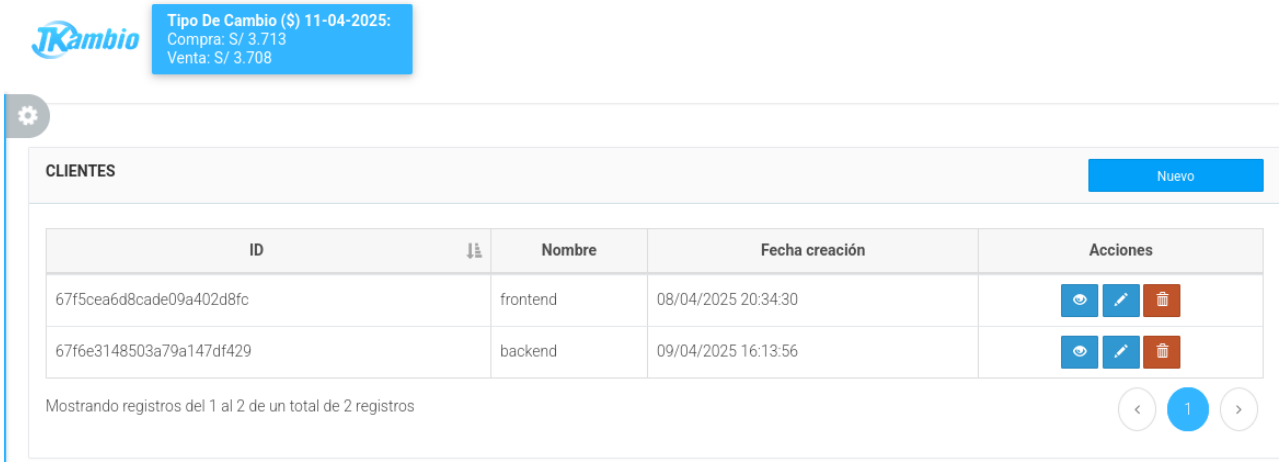


Figura N° 5: Prototipo Listar Clientes
Fuente: Creado por el autor

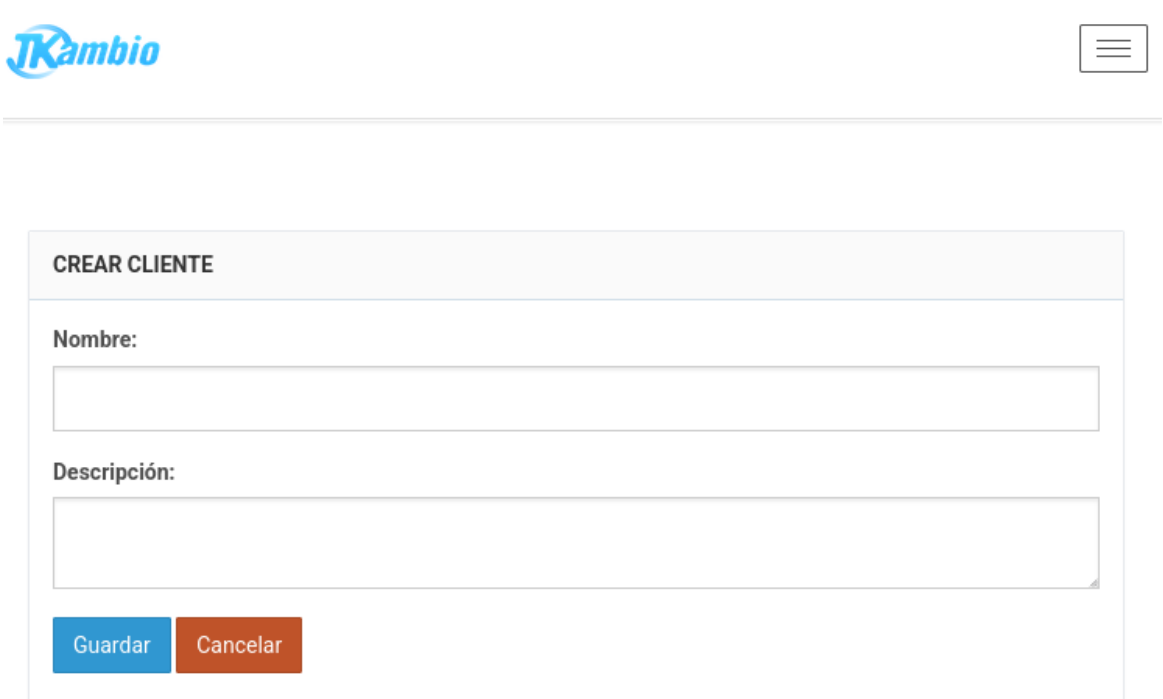


Figura N° 6: Prototipo Crear Cliente
Fuente: Creado por el autor

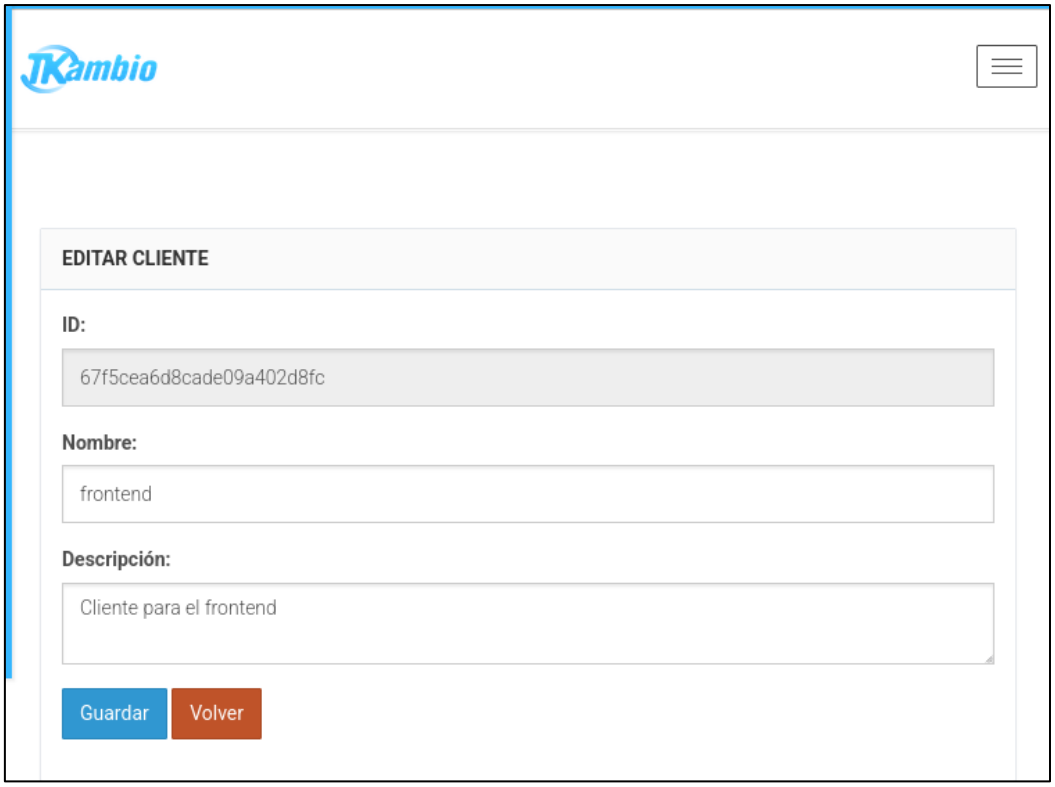


Figura N° 7: Prototipo Editar Cliente
Fuente: Creado por el autor

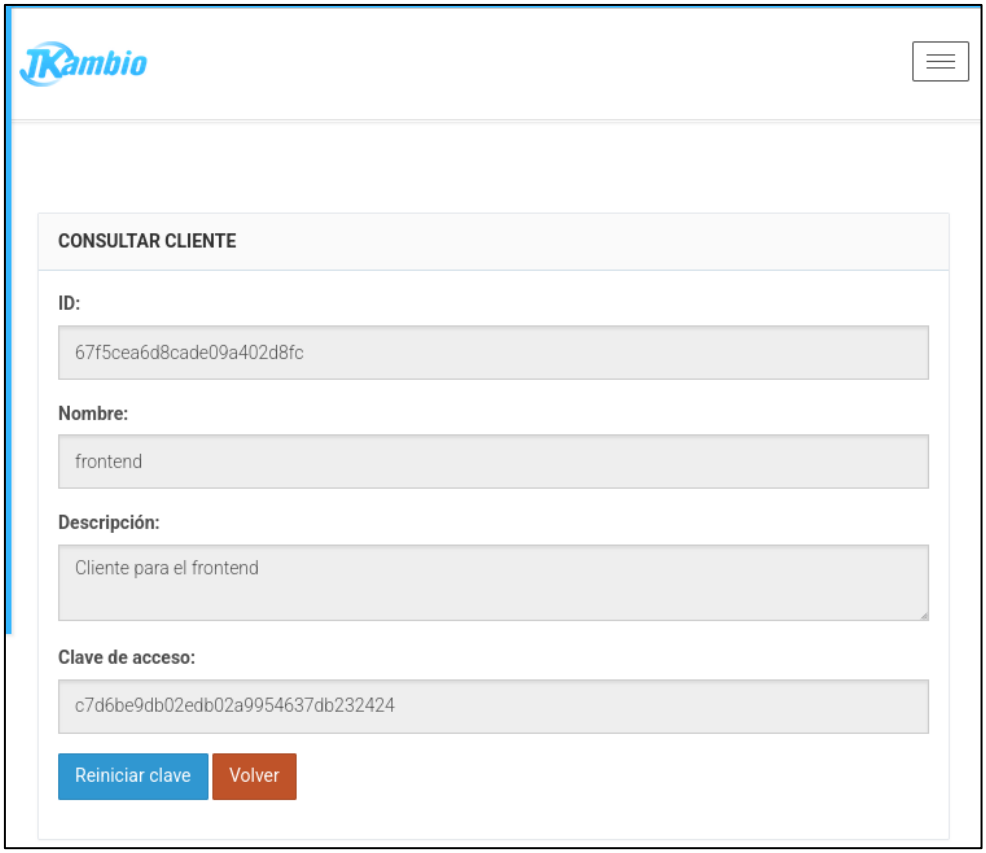


Figura N° 8: Prototipo Consultar Cliente
Fuente: Creado por el autor

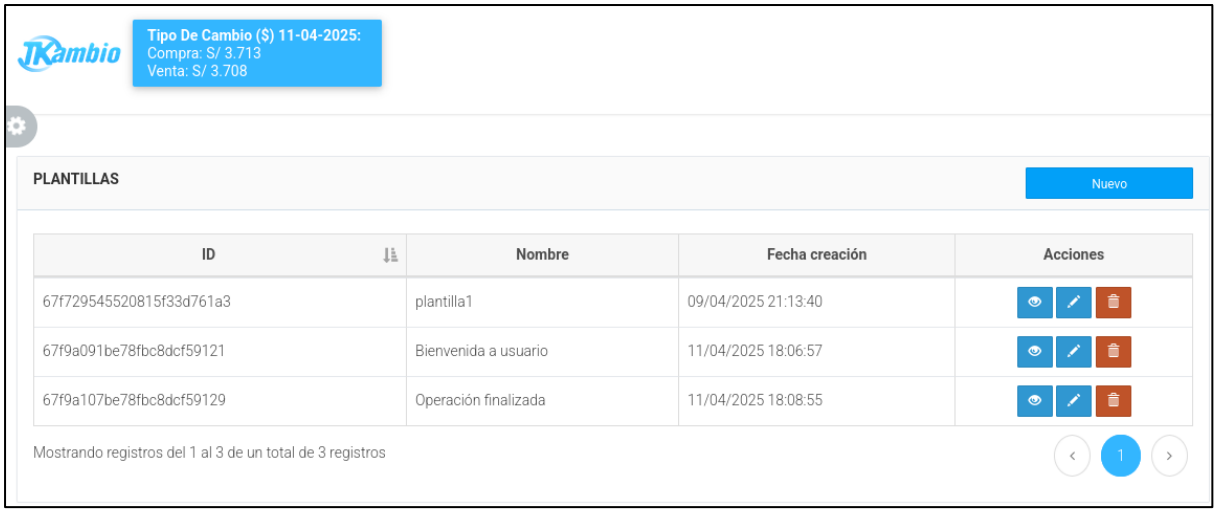


Figura N° 9: Prototipo Listar Plantillas
Fuente: Creado por el autor

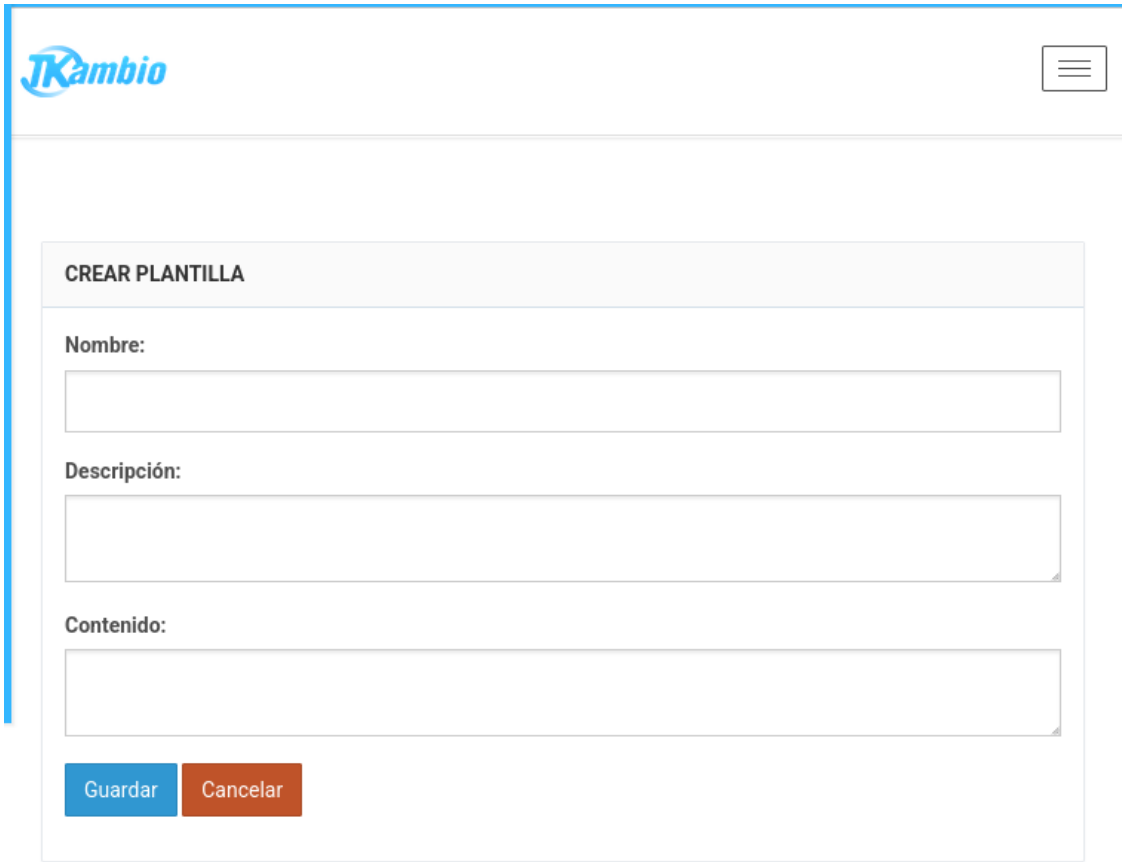


Figura N° 10: Prototipo Crear Plantilla
Fuente: Creado por el autor

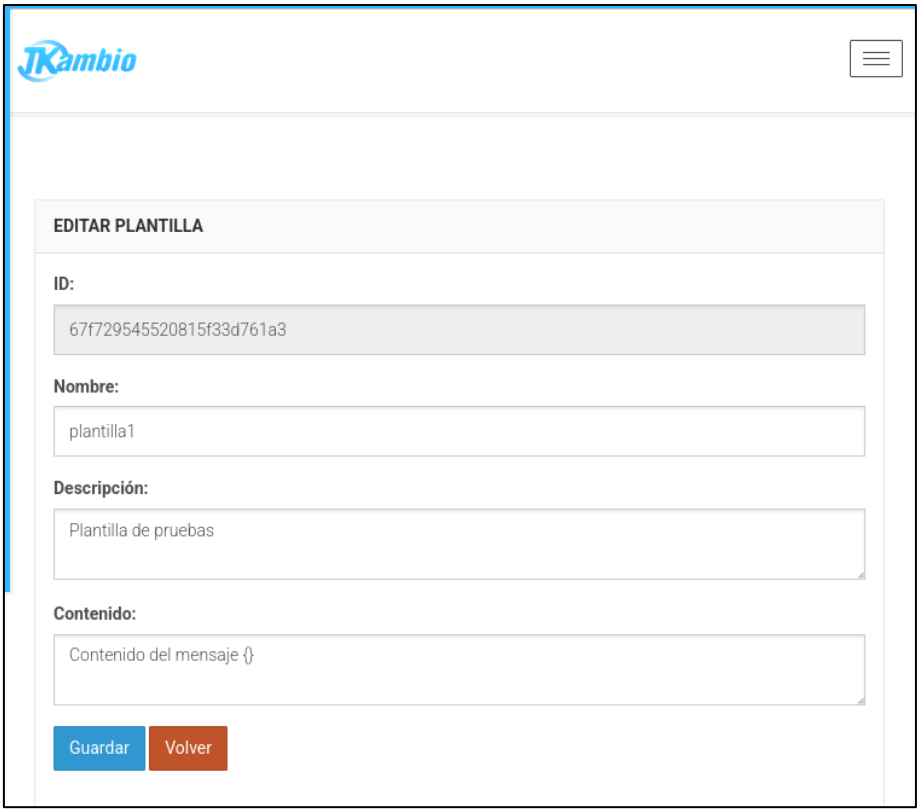


Figura N° 11: Prototipo Editar Plantilla
Fuente: Creado por el autor

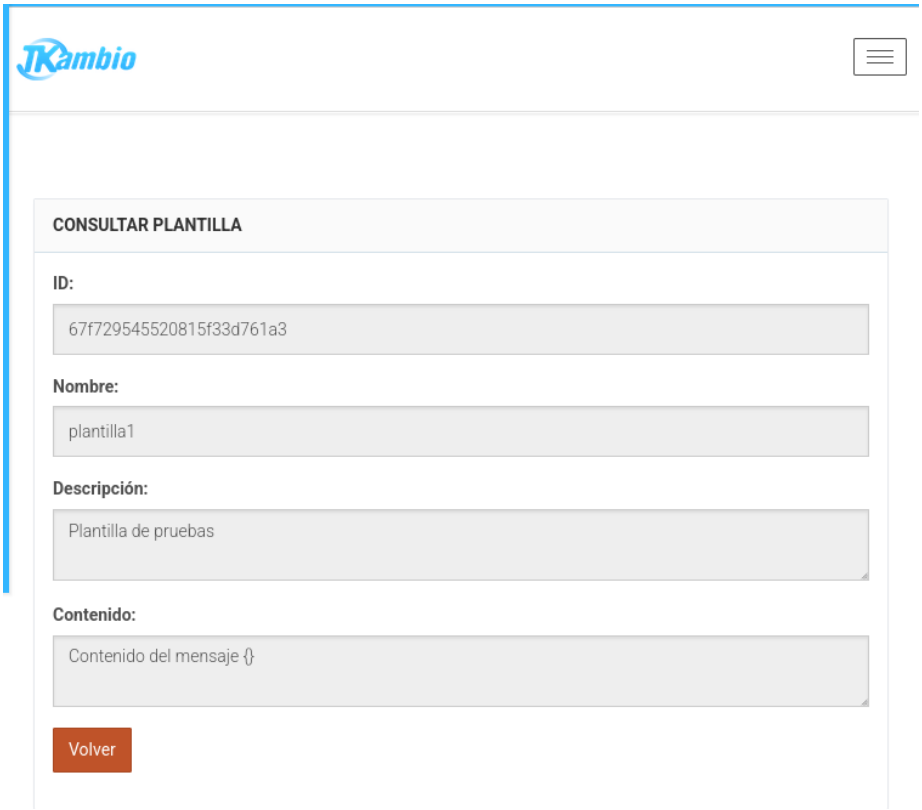


Figura N° 12: Prototipo Consultar Plantilla
Fuente: Creado por el autor

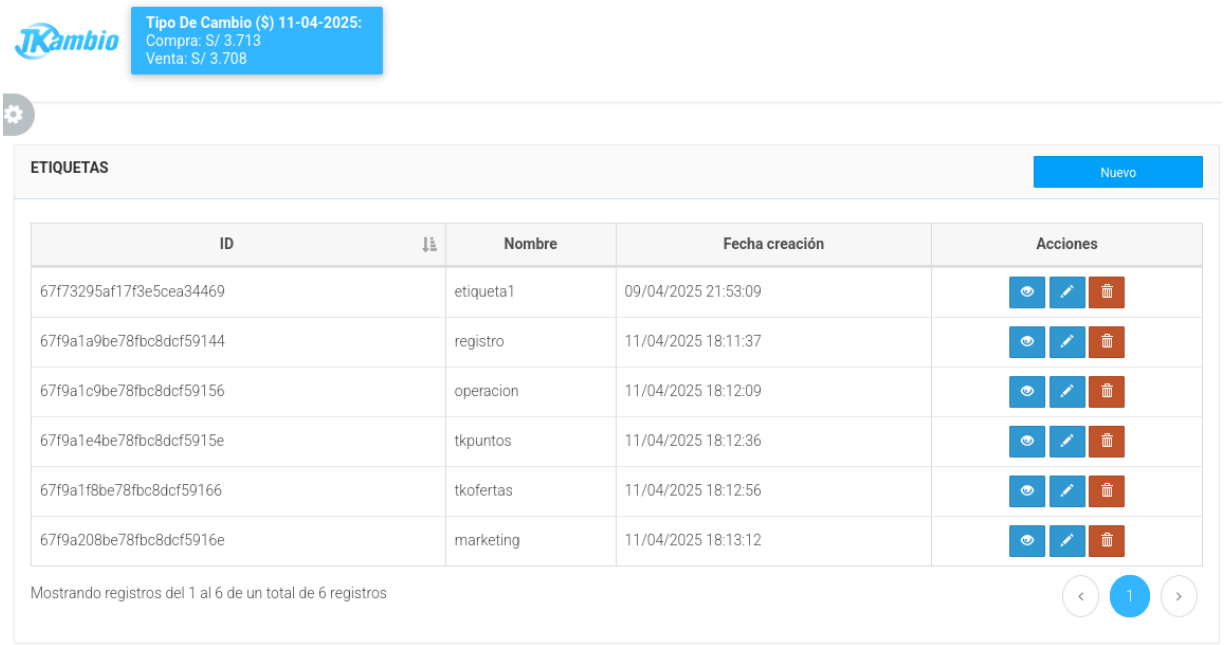


Figura N° 13: Prototipo Listar Etiquetas
Fuente: Creado por el autor

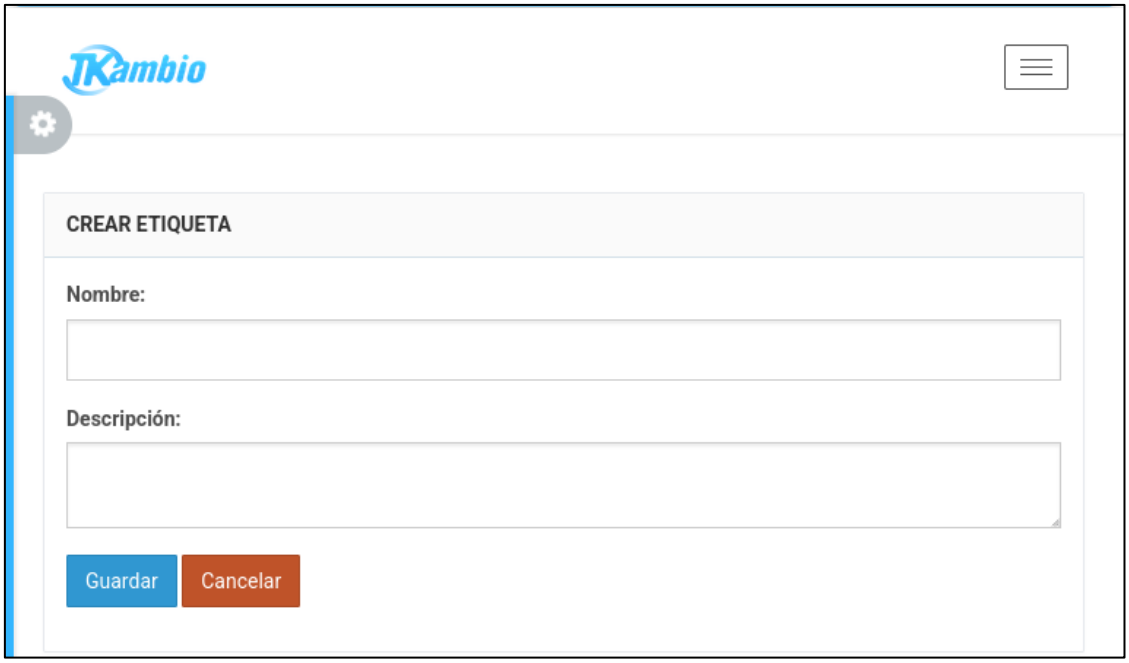


Figura N° 14: Prototipo Crear Etiqueta
Fuente: Creado por el autor

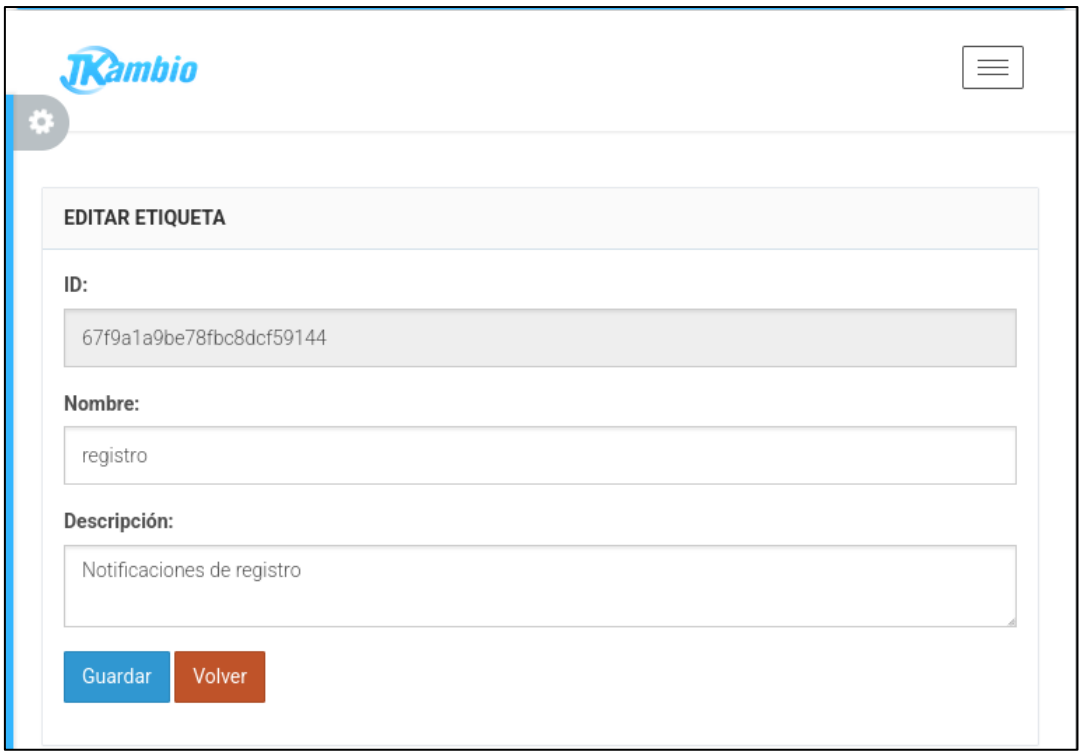


Figura N° 15: Prototipo Editar Etiqueta
Fuente: Creado por el autor

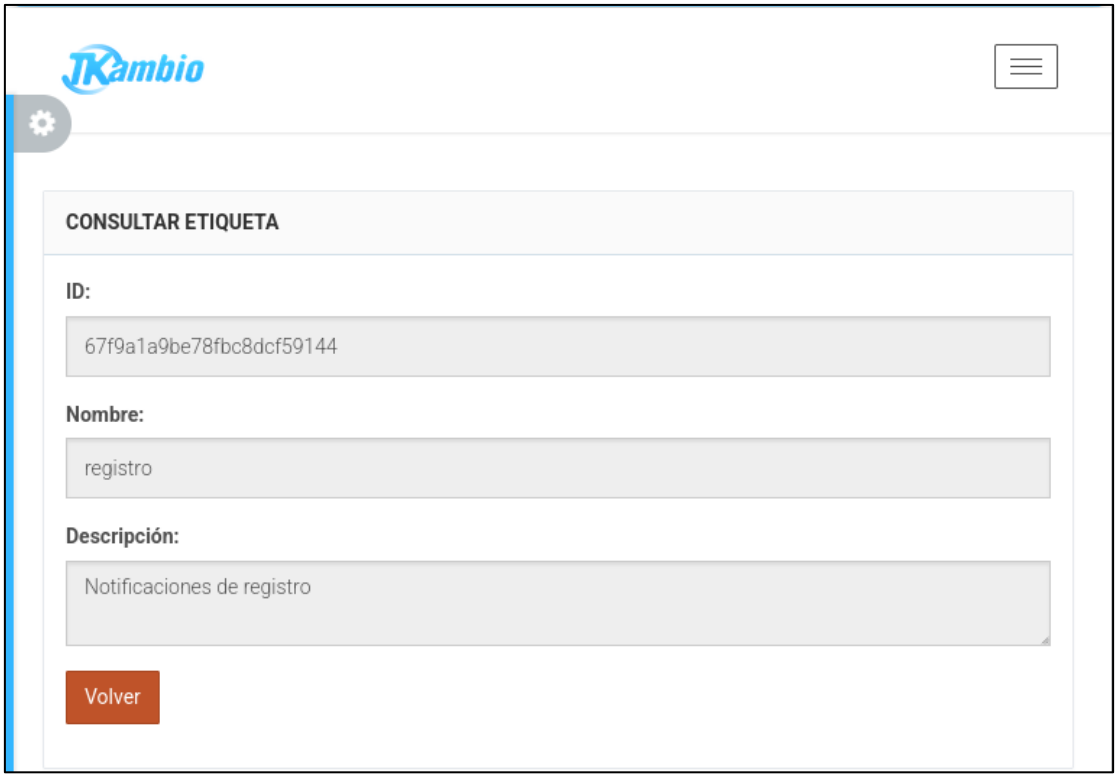


Figura N° 16: Prototipo Consultar Etiqueta
Fuente: Creado por el autor

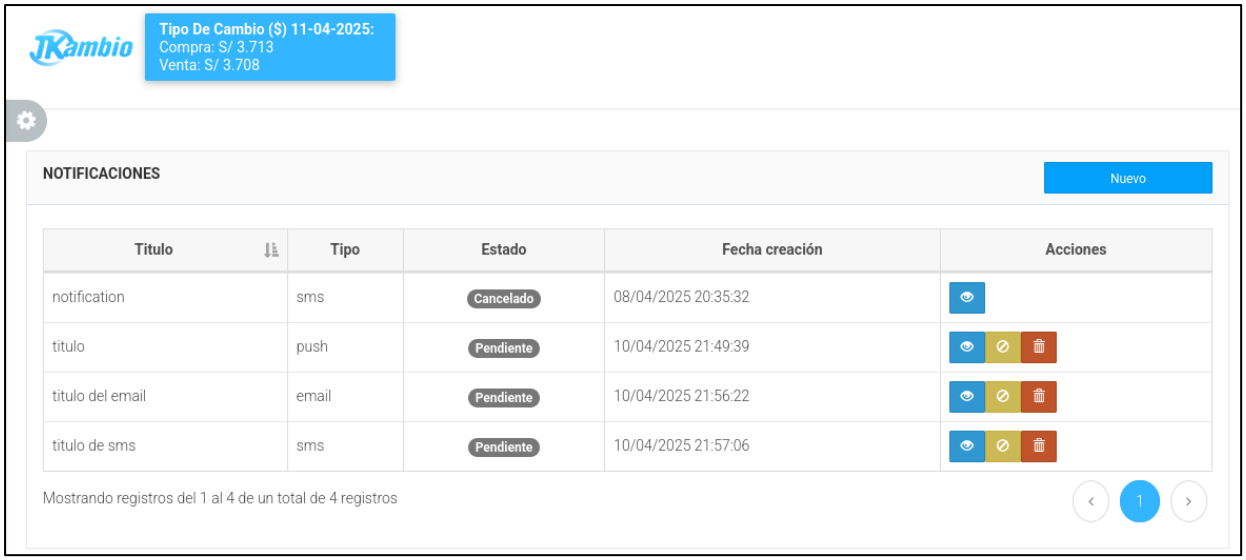


Figura N° 17: Prototipo Listar Notificaciones
Fuente: Creado por el autor

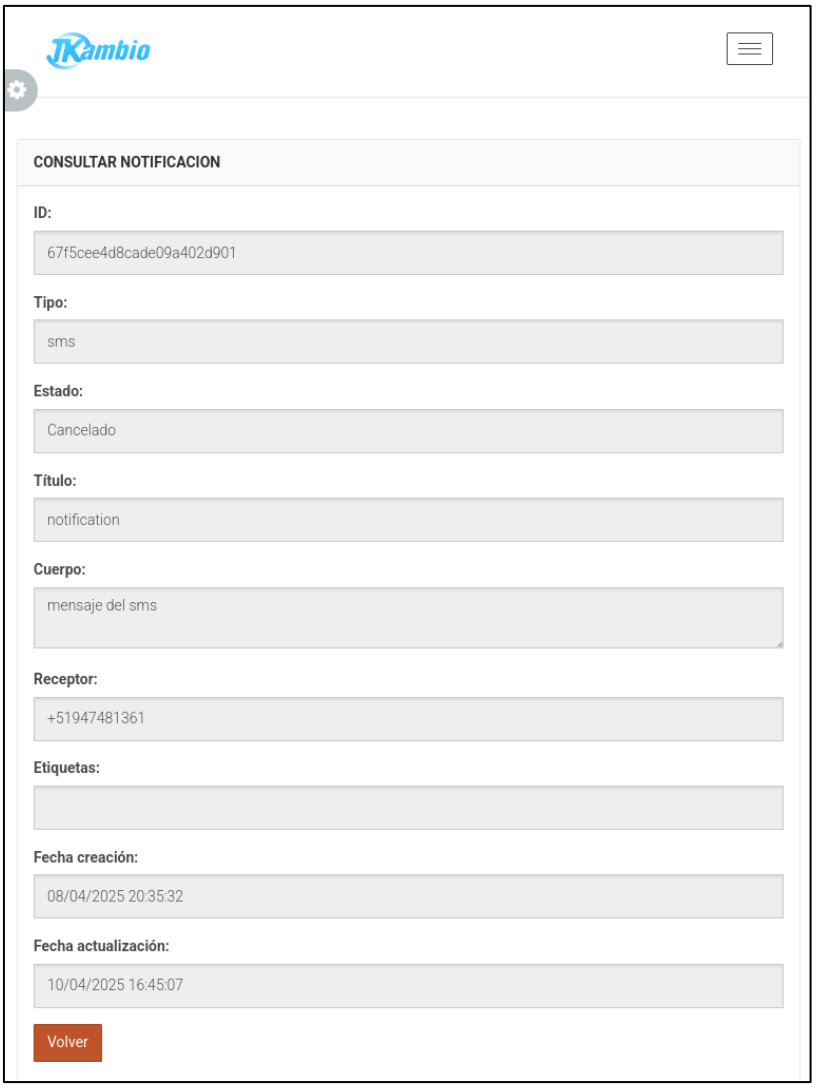


Figura N° 18: Prototipo Consultar Notificación
Fuente: Creado por el autor

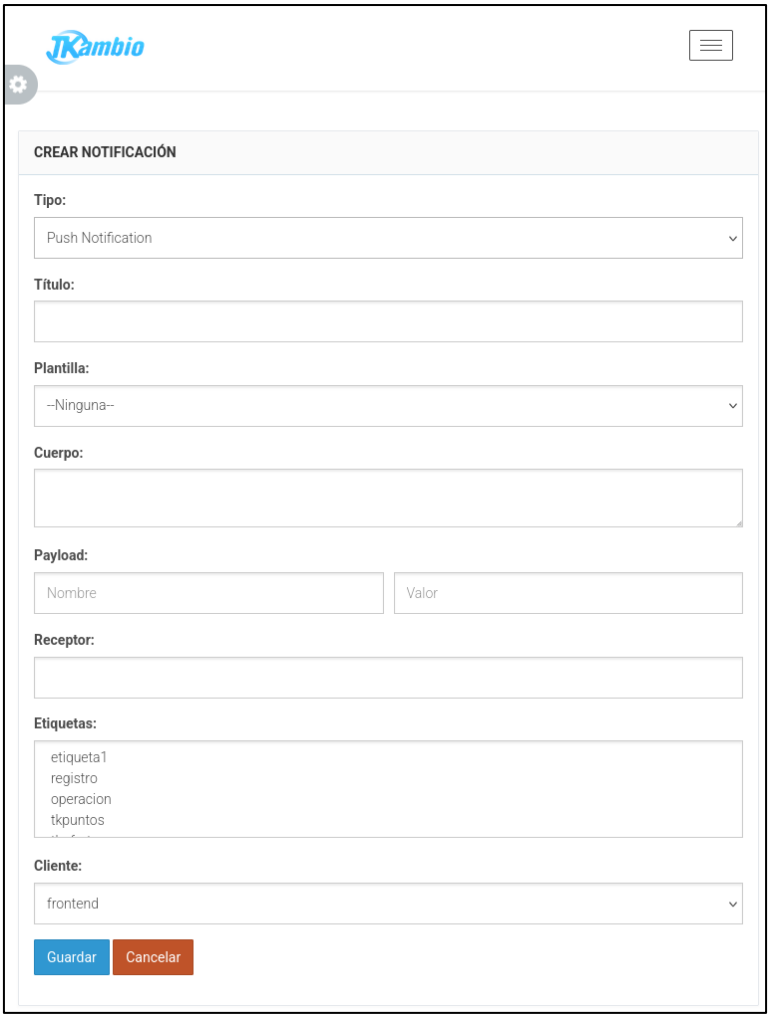


Figura N° 19: Prototipo Crear Notificación
Fuente: Creado por el autor

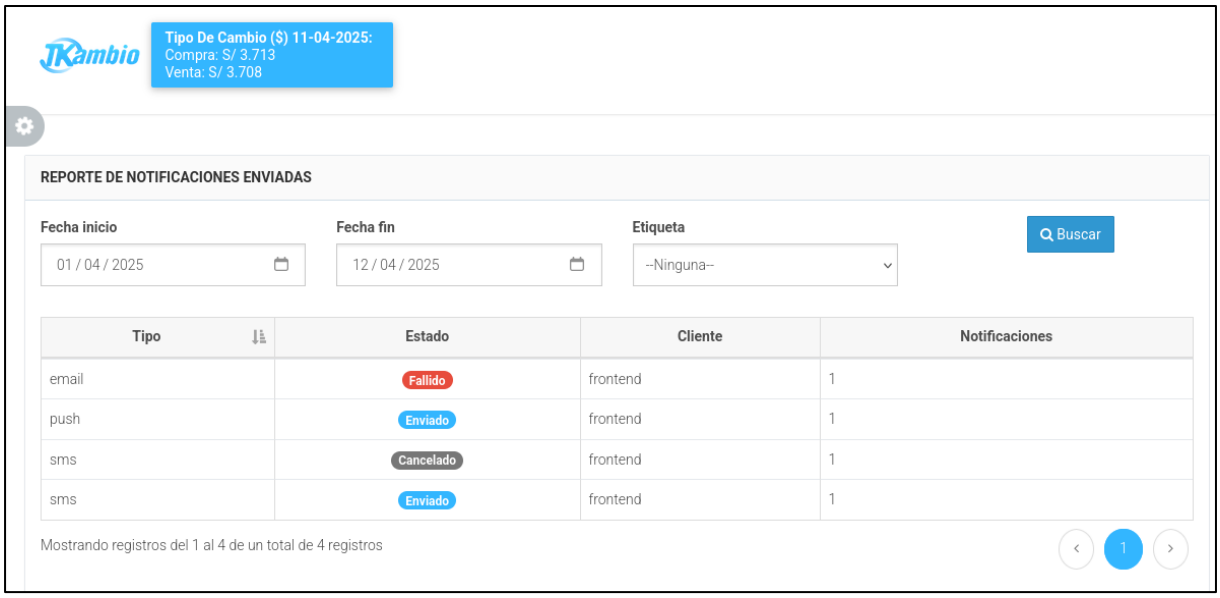


Figura N° 20: Prototipo Reporte de Notificaciones Enviadas
Fuente: Creado por el autor

Figura N° 21: Prototipo Reporte de Tiempos de Envío
Fuente: Creado por el autor

6.1.5.2. Diagrama de Paquetes de Caso de Uso

En el siguiente diagrama se muestran los paquetes de casos de uso.

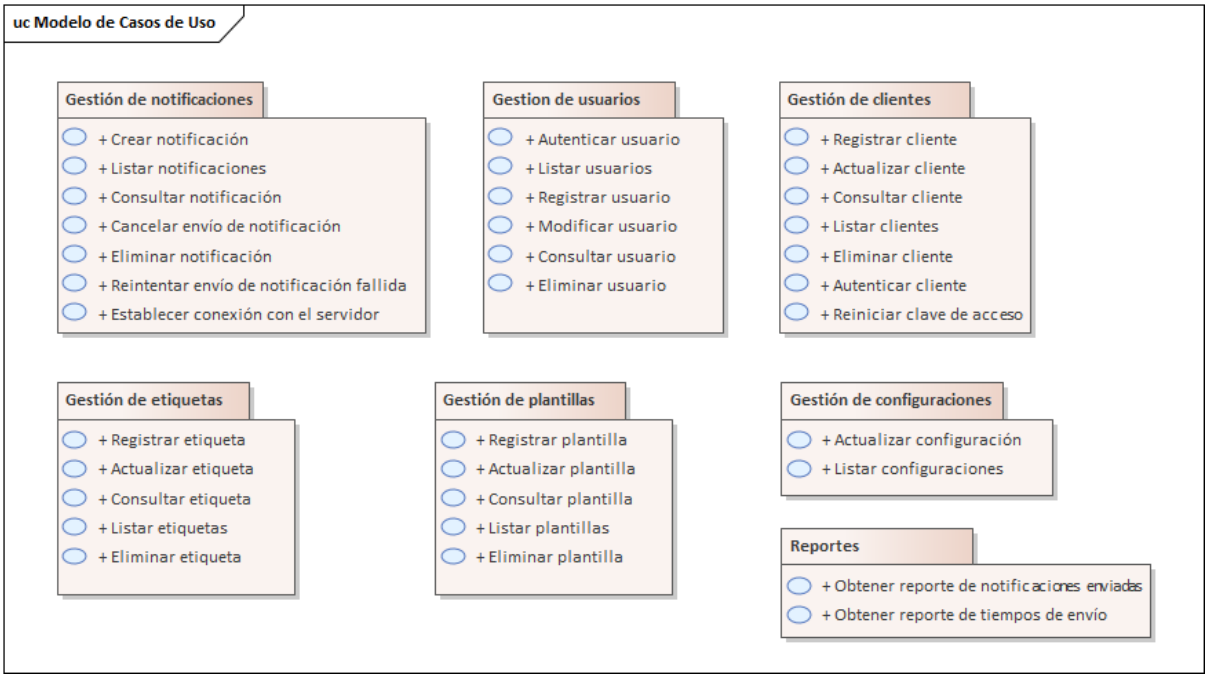


Figura N° 22: Diagrama de Paquetes de Casos de Uso
Fuente: Creado por el autor

6.1.5.3. Diagrama de Caso de Uso

En los siguientes diagrmas se presentan los casos de uso del sistema estructurados en paquetes de análisis.

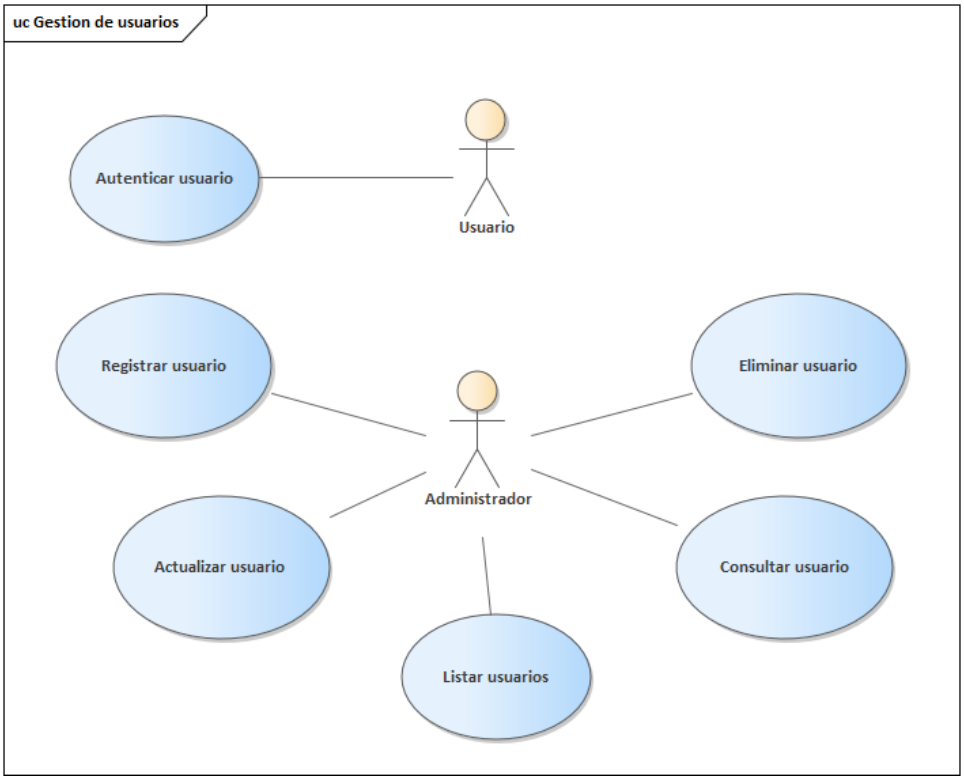


Figura N° 23: DCU Gestión de Usuarios
Fuente: Creado por el autor

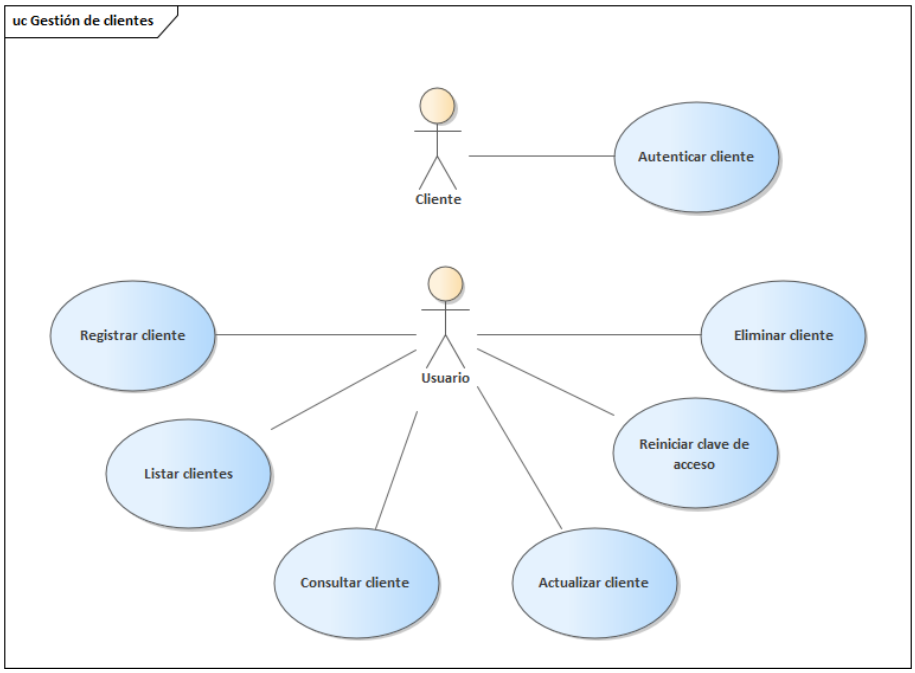


Figura N° 24: DCU Gestión de Clientes
Fuente: Creado por el autor

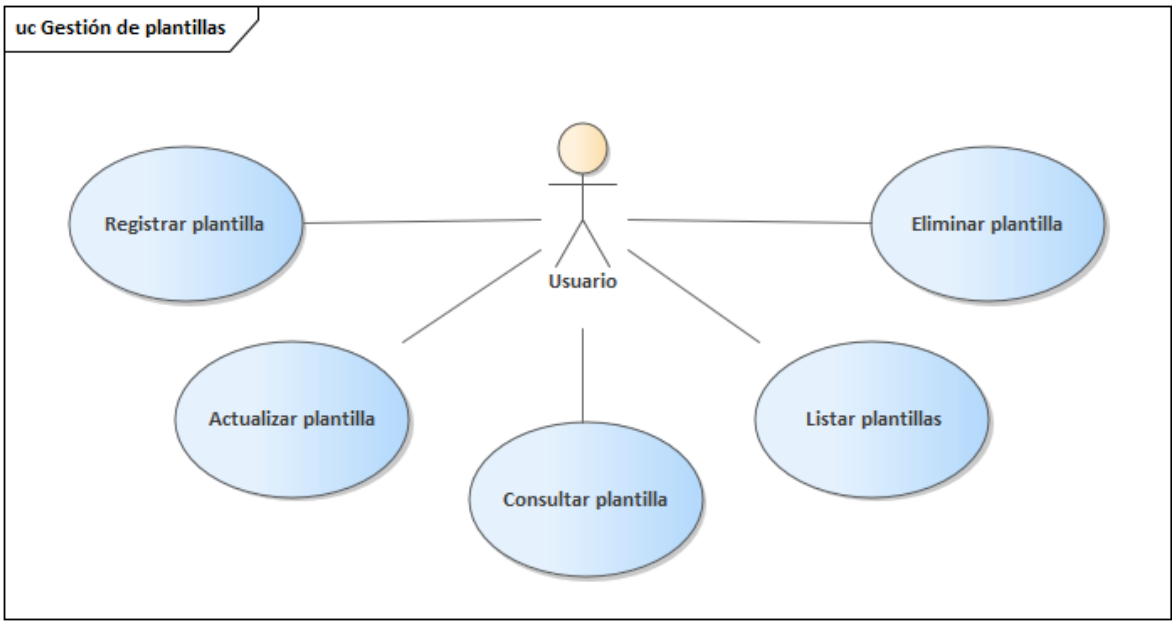


Figura N° 25: DCU Gestión de Plantillas
Fuente: Creado por el autor

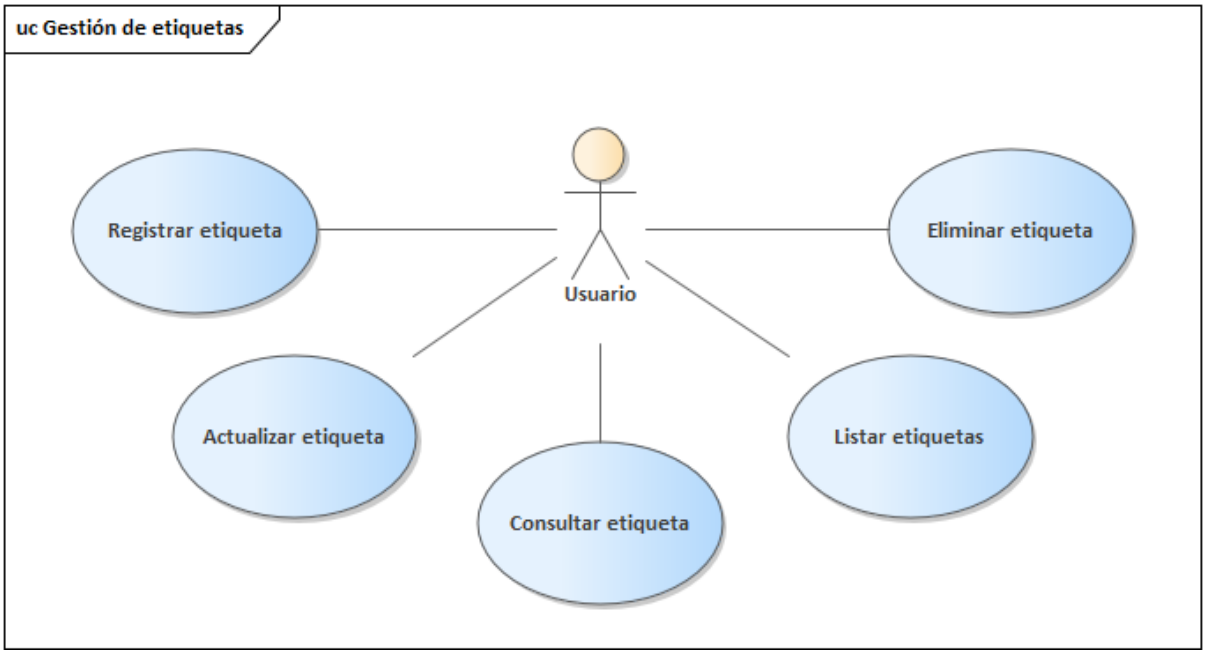


Figura N° 26: DCU Gestión de Etiquetas
Fuente: Creado por el autor

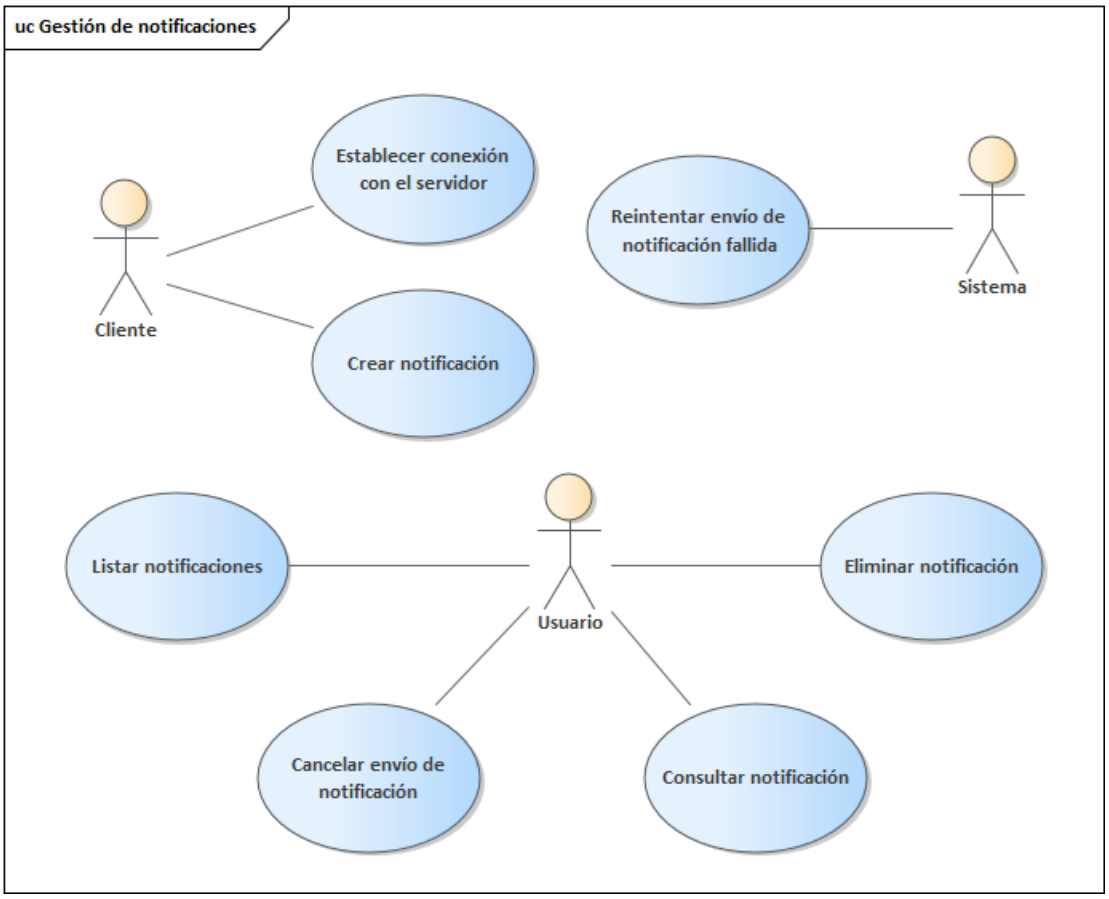


Figura N° 27: DCU Gestión de Notificaciones
Fuente: Creado por el autor

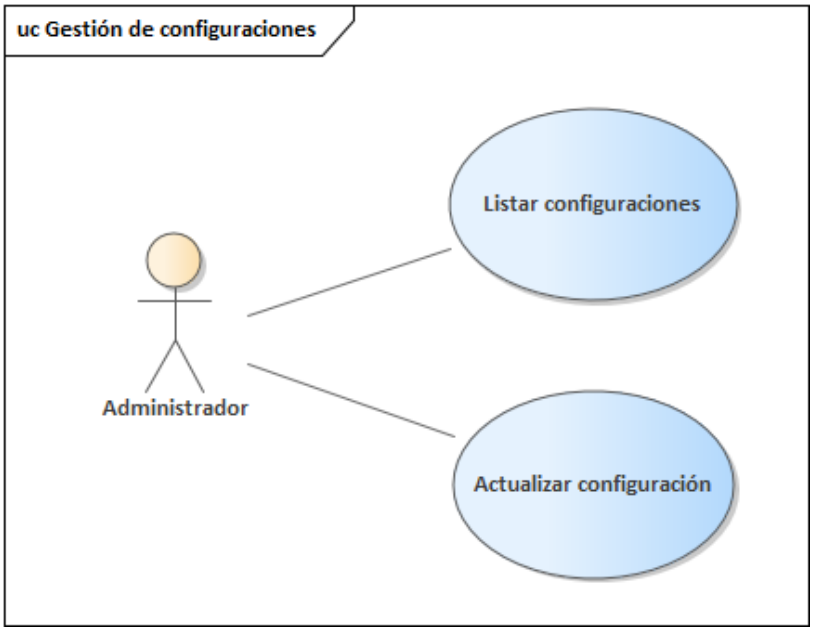


Figura N° 28: DCU Gestión de Configuraciones
Fuente: Creado por el autor

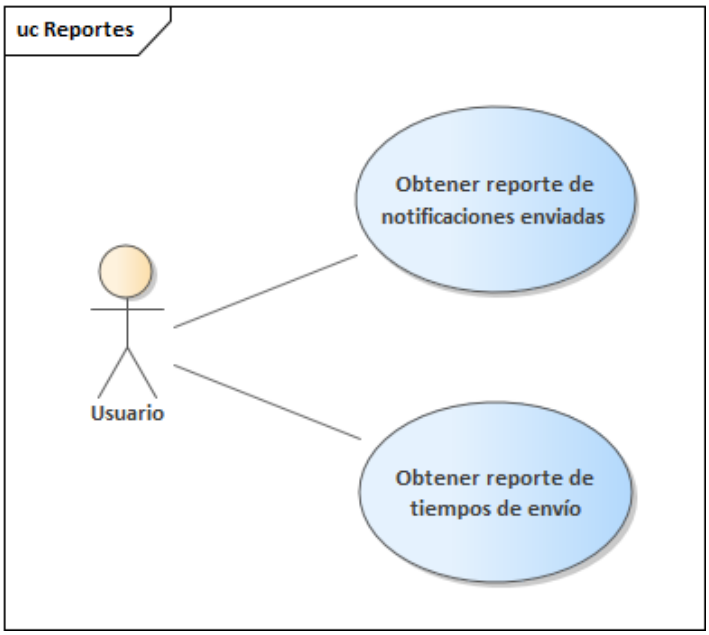


Figura N° 29: DCU Reportes
Fuente: Creado por el autor

6.2. ANÁLISIS Y DISEÑO PRELIMINAR

6.2.1. Especificación de Caso de Uso

En esta etapa se proporciona una descripción detallada y textual de cada caso de uso, presentada en las siguientes tablas, que servirá de referencia para identificar los artefactos dentro del diagrama de robustez.

Tabla N° 1: Especificación de caso de uso Autenticar Usuario

Especificación de Caso de Uso		
Caso de Uso	Autenticar Usuario	
Precondiciones	El usuario a autenticar debe existir en la base de datos.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud POST con su nombre de usuario y contraseña al endpoint correspondiente. La API valida los datos recibidos. La API genera un token de acceso de tipo JWT. La API devuelve una respuesta con el código de éxito (200 – OK) y el token de acceso generado.
	Alternativos	• Si los datos enviados son inválidos, la API responde con un error (401 - Unauthorized). • Si el usuario no existe, la API responde con un error (401 - Unauthorized).
Post condiciones	El token de acceso es enviado al usuario	

Fuente: Creado por el autor

Tabla N° 2: Especificación de caso de uso Registrar Usuario

Especificación de Caso de Uso		
Caso de Uso	Registrar Usuario	
Precondiciones	El administrador debe estar autenticado.	
Flujo de Eventos	**Básico**	El administrador envía una solicitud POST con los datos del nuevo usuario al endpoint correspondiente. La API valida los datos recibidos. La API almacena el nuevo usuario en la base de datos. La API devuelve una respuesta con el código de éxito (201 - Created) y los datos del nuevo usuario.
	Alternativos	• Si los datos enviados son inválidos, la API responde con un error (400 - Bad Request). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	El nuevo usuario queda registrado en la base de datos.	

Fuente: Creado por el autor

Tabla N° 3: Especificación de caso de uso Actualizar Usuario

Especificación de Caso de Uso		
Caso de Uso	Actualizar Usuario	
Precondiciones	El administrador debe estar autenticado. El usuario a actualizar debe existir en la base de datos.	
Flujo de Eventos	**Básico**	El administrador envía una solicitud PUT con los datos actualizados al endpoint correspondiente. La API valida los datos recibidos. La API actualiza la información del usuario en la base de datos. La API devuelve una respuesta con el código de éxito (200 - OK) y los datos actualizados del usuario.
	Alternativos	• Si los datos enviados son inválidos, la API responde con un error (400 - Bad Request). • Si el usuario no existe, la API responde con un error (404 - Not Found). • Si el administrador no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	La información del usuario es actualizada en la base de datos.	

Fuente: Creado por el autor

Tabla N° 4: Especificación de caso de uso Listar Usuarios

Especificación de Caso de Uso		
Caso de Uso	Listar Usuarios	
Precondiciones	El administrador debe estar autenticado.	
Flujo de Eventos	**Básico**	El administrador envía una solicitud GET al endpoint de listado. La API consulta la base de datos para obtener todos los usuarios registrados. La API devuelve una respuesta con el código de éxito (200 - OK) y una lista de los usuarios encontrados.
	Alternativos	• Si no existen usuarios para listar, la API responde con una lista vacía y el código de éxito (200 - OK). • Si el administrador no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	El administrador recibe los datos de los usuarios disponibles	

Fuente: Creado por el autor

Tabla N° 5: Especificación de caso de uso Consultar Usuario

Especificación de Caso de Uso		
Caso de Uso	Consultar Usuario	
Precondiciones	El administrador debe estar autenticado.	
Flujo de Eventos	**Básico**	El administrador envía una solicitud GET al endpoint correspondiente. La API busca el usuario en la base de datos. La API devuelve una respuesta con el código de éxito (200 - OK) y los datos del usuario solicitado.
	Alternativos	• Si el usuario no existe, la API responde con un error (404 - Not Found). • Si el administrador no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	Los datos del usuario son enviados al solicitante.	

Fuente: Creado por el autor

Tabla N° 6: Especificación de caso de uso Eliminar Usuario

Especificación de Caso de Uso		
Caso de Uso	Eliminar Usuario	
Precondiciones	El administrador debe estar autenticado. El usuario que se desea eliminar debe existir en la base de datos.	
Flujo de Eventos	**Básico**	El administrador envía una solicitud DELETE al endpoint correspondiente. La API verifica la existencia del usuario en la base de datos. La API actualiza el estado del usuario para marcarlo como eliminado. La API devuelve una respuesta con el código de éxito (200 - OK) indicando que el usuario ha sido eliminado de forma lógica.
	Alternativos	• Si el usuario no existe o ya está marcado como eliminado, la API responde con un error (404 - Not Found). • Si el administrador no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	El usuario no será visible en las consultas regulares.	

Fuente: Creado por el autor

Tabla N° 7: Especificación de caso de uso Registrar Cliente

Especificación de Caso de Uso		
Caso de Uso	Registrar Cliente	
Precondiciones	El usuario debe estar autenticado.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud POST con los datos del nuevo cliente al endpoint correspondiente. La API valida los datos recibidos. La API almacena el nuevo cliente en la base de datos. La API devuelve una respuesta con el código de éxito (201 - Created) y los datos del nuevo cliente.
	Alternativos	• Si los datos enviados son inválidos, la API responde con un error (400 - Bad Request). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	El nuevo cliente queda registrado en la base de datos.	

Fuente: Creado por el autor

Tabla N° 8: Especificación de caso de uso Actualizar Cliente

Especificación de Caso de Uso		
Caso de Uso	Actualizar Cliente	
Precondiciones	El usuario debe estar autenticado. El cliente a actualizar debe existir en la base de datos.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud PUT con los datos actualizados del cliente al endpoint correspondiente. La API valida los datos recibidos. La API actualiza la información del cliente en la base de datos. La API devuelve una respuesta con el código de éxito (200 - OK) y los datos actualizados del cliente.
	Alternativos	• Si los datos enviados son inválidos, la API responde con un error (400 - Bad Request). • Si el cliente no existe, la API responde con un error (404 - Not Found). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	La información del cliente es actualizada en la base de datos.	

Fuente: Creado por el autor

Tabla N° 9: Especificación de caso de uso Consultar Cliente

Especificación de Caso de Uso		
Caso de Uso	Consultar Cliente	
Precondiciones	El usuario debe estar autenticado.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud GET al endpoint correspondiente con el identificador del cliente. La API busca el cliente en la base de datos. La API devuelve una respuesta con el código de éxito (200 - OK) y los datos del cliente solicitado.
	Alternativos	• Si el cliente no existe, la API responde con un error (404 - Not Found). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	Los datos del cliente son enviados al usuario.	

Fuente: Creado por el autor

Tabla N° 10: Especificación de caso de uso Listar Clientes

Especificación de Caso de Uso		
Caso de Uso	Listar Clientes	
Precondiciones	El usuario debe estar autenticado.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud GET al endpoint de listado. La API consulta la base de datos para obtener todos los clientes registrados. La API devuelve una respuesta con el código de éxito (200 - OK) y una lista de los clientes encontrados.
	Alternativos	• Si no existen clientes para listar, la API responde con una lista vacía y el código de éxito (200 - OK). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	El usuario recibe los datos de los clientes disponibles.	

Fuente: Creado por el autor

Tabla N° 11: Especificación de caso de uso Eliminar Cliente

Especificación de Caso de Uso		
Caso de Uso	Eliminar Cliente	
Precondiciones	El usuario debe estar autenticado. El cliente que se desea eliminar debe existir en la base de datos.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud DELETE al endpoint correspondiente. La API verifica la existencia del cliente en la base de datos. La API actualiza el estado del cliente para marcarlo como eliminado La API devuelve una respuesta con el código de éxito (200 - OK) indicando que el cliente ha sido eliminado de forma lógica.
	Alternativos	• Si el cliente no existe o ya está marcado como eliminado, la API responde con un error (404 - Not Found). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	El cliente no será visible en las consultas regulares.	

Fuente: Creado por el autor

Tabla N° 12: Especificación de caso de uso Autenticar Cliente

Especificación de Caso de Uso		
Caso de Uso	Autenticar Cliente	
Precondiciones	El cliente a autenticar debe existir en la base de datos.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud POST con el identificador y clave de acceso del cliente al endpoint correspondiente. La API valida los datos recibidos. La API genera un token de acceso de tipo JWT. La API devuelve una respuesta con el código de éxito (200 – OK) y el token de acceso generado.
	Alternativos	• Si los datos enviados son inválidos, la API responde con un error (401 - Unauthorized). • Si el cliente no existe, la API responde con un error (401 - Unauthorized).
Post condiciones	El token de acceso es enviado al usuario	

Fuente: Creado por el autor

Tabla N° 13: Especificación de caso de uso Reiniciar Clave de Acceso de Cliente

Especificación de Caso de Uso		
Caso de Uso	Reiniciar Clave de Acceso de Cliente	
Precondiciones	El usuario debe estar autenticado. El cliente a actualizar debe existir en la base de datos.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud PUT con el identificador del cliente al endpoint correspondiente. La API genera una nueva clave de acceso y actualiza la información del cliente en la base de datos. La API devuelve una respuesta con el código de éxito (200 - OK) y los datos actualizados del cliente.
	Alternativos	• Si el cliente no existe, la API responde con un error (404 - Not Found). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	La clave de acceso del cliente es actualizada en la base de datos.	

Fuente: Creado por el autor

Tabla N° 14: Especificación de caso de uso Registrar Plantilla

Especificación de Caso de Uso		
Caso de Uso	Registrar Plantilla	
Precondiciones	El usuario debe estar autenticado.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud POST con los datos de la nueva plantilla al endpoint correspondiente. La API valida los datos recibidos. La API almacena la nueva plantilla en la base de datos. La API devuelve una respuesta con el código de éxito (201 - Created) y los datos de la nueva plantilla.
	Alternativos	• Si los datos enviados son inválidos, la API responde con un error (400 - Bad Request). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	La nueva plantilla queda registrada en la base de datos.	

Fuente: Creado por el autor

Tabla N° 15: Especificación de caso de uso Consultar Plantilla

Especificación de Caso de Uso		
Caso de Uso	Consultar Plantilla	
Precondiciones	El usuario debe estar autenticado.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud GET al endpoint correspondiente con el identificador de la plantilla. La API busca la plantilla en la base de datos. La API devuelve una respuesta con el código de éxito (200 - OK) y los datos de la plantilla solicitada.
	Alternativos	• Si la plantilla no existe, la API responde con un error (404 - Not Found). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	Los datos de la plantilla son enviados al usuario.	

Fuente: Creado por el autor

Tabla N° 16: Especificación de caso de uso Actualizar Plantilla

Especificación de Caso de Uso		
Caso de Uso	Actualizar Plantilla	
Precondiciones	El usuario debe estar autenticado. La plantilla a actualizar debe existir en la base de datos.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud PUT con los datos actualizados de la plantilla al endpoint correspondiente. La API valida los datos recibidos. La API actualiza la información de la plantilla en la base de datos. La API devuelve una respuesta con el código de éxito (200 - OK) y los datos actualizados de la plantilla.
	Alternativos	• Si los datos enviados son inválidos, la API responde con un error (400 - Bad Request). • Si la plantilla no existe, la API responde con un error (404 - Not Found). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	La información de la plantilla es actualizada en la base de datos.	

Fuente: Creado por el autor

Tabla N° 17: Especificación de caso de uso Listar Plantillas

Especificación de Caso de Uso		
Caso de Uso	Listar Plantillas	
Precondiciones	El usuario debe estar autenticado.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud GET al endpoint de listado. La API consulta la base de datos para obtener todas las plantillas registradas. La API devuelve una respuesta con el código de éxito (200 - OK) y una lista de las plantillas encontradas.
	Alternativos	• Si no existen plantillas para listar, la API responde con una lista vacía y el código de éxito (200 - OK). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	El usuario recibe los datos de las plantillas disponibles	

Fuente: Creado por el autor

Tabla N° 18: Especificación de caso de uso Eliminar Plantilla

Especificación de Caso de Uso		
Caso de Uso	Eliminar Plantilla	
Precondiciones	El usuario debe estar autenticado. La plantilla que se desea eliminar debe existir en la base de datos.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud DELETE al endpoint correspondiente. La API verifica la existencia de la plantilla en la base de datos. La API actualiza el estado de la plantilla para marcarla como eliminada. La API devuelve una respuesta con el código de éxito (200 - OK) indicando que la plantilla ha sido eliminada de forma lógica.
	Alternativos	• Si la plantilla no existe o ya está marcada como eliminada, la API responde con un error (404 - Not Found). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	La plantilla no será visible en las consultas regulares.	

Fuente: Creado por el autor

Tabla N° 19: Especificación de caso de uso Registrar Etiqueta

Especificación de Caso de Uso		
Caso de Uso	Registrar Etiqueta	
Precondiciones	El usuario debe estar autenticado.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud POST con los datos de la nueva etiqueta al endpoint correspondiente. La API valida los datos recibidos. La API almacena la nueva etiqueta en la base de datos. La API devuelve una respuesta con el código de éxito (201 - Created) y los datos de la nueva etiqueta.
	Alternativos	• Si los datos enviados son inválidos, la API responde con un error (400 - Bad Request). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	La nueva etiqueta queda registrada en la base de datos.	

Fuente: Creado por el autor

Tabla N° 20: Especificación de caso de uso Consultar Etiqueta

Especificación de Caso de Uso		
Caso de Uso	Consultar Etiqueta	
Precondiciones	El usuario debe estar autenticado.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud GET al endpoint correspondiente con el identificador de la etiqueta. La API busca la etiqueta en la base de datos. La API devuelve una respuesta con el código de éxito (200 - OK) y los datos de la etiqueta solicitada.
	Alternativos	• Si la etiqueta no existe, la API responde con un error (404 - Not Found). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	Los datos de la etiqueta son enviados al usuario.	

Fuente: Creado por el autor

Tabla N° 21: Especificación de caso de uso Actualizar Etiqueta

Especificación de Caso de Uso		
Caso de Uso	Actualizar Etiqueta	
Precondiciones	El usuario debe estar autenticado. La etiqueta a actualizar debe existir en la base de datos.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud PUT con los datos actualizados de la etiqueta al endpoint correspondiente. La API valida los datos recibidos. La API actualiza la información de la etiqueta en la base de datos. La API devuelve una respuesta con el código de éxito (200 - OK) y los datos actualizados de la etiqueta.
	Alternativos	• Si los datos enviados son inválidos, la API responde con un error (400 - Bad Request). • Si la etiqueta no existe, la API responde con un error (404 - Not Found). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	La información de la etiqueta es actualizada en la base de datos.	

Fuente: Creado por el autor

Tabla N° 22: Especificación de caso de uso Listar Etiquetas

Especificación de Caso de Uso		
Caso de Uso	Listar Etiquetas	
Precondiciones	El usuario debe estar autenticado.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud GET al endpoint de listado. La API consulta la base de datos para obtener todas las etiquetas disponibles. La API devuelve una respuesta con el código de éxito (200 - OK) y una lista de las etiquetas encontradas.
	Alternativos	• Si no existen etiquetas para listar, la API responde con una lista vacía y el código de éxito (200 - OK). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	El usuario recibe los datos de las etiquetas disponibles.	

Fuente: Creado por el autor

Tabla N° 23: Especificación de caso de uso Eliminar Etiqueta

Especificación de Caso de Uso		
Caso de Uso	Eliminar Etiqueta	
Precondiciones	El usuario debe estar autenticado. La etiqueta que se desea eliminar debe existir en la base de datos.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud DELETE al endpoint correspondiente. La API verifica la existencia de la etiqueta en la base de datos. La API actualiza el estado de la etiqueta para marcarla como eliminada. La API devuelve una respuesta con el código de éxito (200 - OK) indicando que la etiqueta ha sido eliminada de forma lógica.
	Alternativos	• Si la etiqueta no existe o ya está marcada como eliminada, la API responde con un error (404 - Not Found). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	La etiqueta no será visible en las consultas regulares.	

Fuente: Creado por el autor

Tabla N° 24: Especificación de caso de uso Enviar Notificación

Especificación de Caso de Uso		
Caso de Uso	Enviar Notificación	
Precondiciones	El cliente debe estar autenticado.	
Flujo de Eventos	**Básico**	El cliente envía una solicitud POST con los datos de la nueva notificación al endpoint correspondiente. La API valida los datos recibidos. La API obtiene la plantilla y/o etiquetas de la base de datos. La API almacena la nueva notificación en la base de datos y agrega la notificación a la cola para que se realice el envío por el canal correspondiente. La API devuelve una respuesta con el código de éxito (201 - Created) y los datos de la nueva notificación.
	Alternativos	• Si los datos enviados son inválidos, la API responde con un error (400 - Bad Request). • Si el cliente no está autenticado, la API responde con un error (401 - Unauthorized)..
Post condiciones	La nueva notificación queda registrada en la base de datos.	

Fuente: Creado por el autor

Tabla N° 25: Especificación de caso de uso Listar Notificaciones

Especificación de Caso de Uso		
Caso de Uso	Listar Notificaciones	
Precondiciones	El usuario debe estar autenticado.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud GET al endpoint de listado. La API consulta la base de datos para obtener todas las notificaciones disponibles. La API devuelve una respuesta con el código de éxito (200 - OK) y una lista de las notificaciones encontradas.
	Alternativos	• Si no existen notificaciones para listar, la API responde con una lista vacía y el código de éxito (200 - OK). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	El usuario recibe los datos de las notificaciones disponibles.	

Fuente: Creado por el autor

Tabla N° 26: Especificación de caso de uso Consultar Notificación

Especificación de Caso de Uso		
Caso de Uso	Consultar Notificación	
Precondiciones	El usuario debe estar autenticado.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud GET al endpoint correspondiente con el identificador de la notificación. La API busca la notificación en la base de datos. La API devuelve una respuesta con el código de éxito (200 - OK) y los datos de la notificación solicitada.
	Alternativos	• Si la notificación no existe, la API responde con un error (404 - Not Found). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	Los datos de la notificación son enviados al usuario.	

Fuente: Creado por el autor

Tabla N° 27: Especificación de caso de uso Cancelar Envío de Notificación

Especificación de Caso de Uso		
Caso de Uso	Cancelar Envío de Notificación	
Precondiciones	El usuario debe estar autenticado. La notificación debe existir en la base de datos y aún no haber sido enviada.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud PUT con el identificador de la notificación al endpoint correspondiente. La API busca la notificación en la base de datos y verifica su estado. La API actualiza el estado de la notificación en la base de datos y cancela su envío. La API devuelve una respuesta con el código de éxito (200 - OK).
	Alternativos	• Si la notificación no existe, la API responde con un error (404 - Not Found). • Si el estado de la notificación no es el correcto, la API responde con un error (403 - Forbidden). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	El estado de la notificación es actualizado en la base de datos. La notificación no es enviada.	

Fuente: Creado por el autor

Tabla N° 28: Especificación de caso de uso Eliminar Notificación

Especificación de Caso de Uso		
Caso de Uso	Eliminar Notificación	
Precondiciones	El usuario debe estar autenticado. La notificación que se desea eliminar debe existir en la base de datos.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud DELETE al endpoint correspondiente. La API verifica la existencia de la notificación en la base de datos. La API actualiza el estado de la notificación para marcarla como eliminada. La API devuelve una respuesta con el código de éxito (200 - OK) indicando que la notificación ha sido eliminada de forma lógica.
	Alternativos	• Si la notificación no existe o ya está marcada como eliminada, la API responde con un error (404 - Not Found). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	La notificación no será visible en las consultas regulares.	

Fuente: Creado por el autor

Tabla N° 29: Especificación de caso de uso Establecer Conexión con el Servidor

Especificación de Caso de Uso		
Caso de Uso	Establecer Conexión con el Servidor	
Precondiciones	El cliente debe estar registrado en la base de datos	
Flujo de Eventos	**Básico**	El cliente inicia la conexión al servidor mediante el protocolo WebSocket, enviando su identificador y clave de acceso. El servidor busca el cliente en la base de datos y verifica las credenciales. El servidor establece un canal de comunicación con el cliente.
	Alternativos	• Si los datos enviados son inválidos, el servidor responde con un error (Unauthorized). • Si el cliente no existe, el servidor responde con un error (Unauthorized).
Post condiciones	El cliente autenticado puede recibir mensajes del servidor.	

Fuente: Creado por el autor

Tabla N° 30: Especificación de caso de uso Reintentar envío de notificación fallida

Especificación de Caso de Uso		
Caso de Uso	Reintentar envío de notificación fallida	
Precondiciones	La notificación no debe superar el máximo de intentos de envío	
Flujo de Eventos	**Básico**	El sistema obtiene la notificación fallida de la cola de notificaciones. El sistema obtiene los datos actualizados de la notificación de la base de datos. El sistema verifica el número de intentos realizados para enviar la notificación. El sistema realiza el envío de la notificación por el canal correspondiente. El sistema actualiza los intentos y estado de la notificación en la base de datos.
	Alternativos	• Si la notificación no existe, el sistema no realiza el envío de la notificación. • Si la cantidad de intentos no es válida, el sistema no realiza el envío de la notificación. • Si el envío de la notificación falla nuevamente, el sistema regresa la notificación a la cola para un nuevo intento.
Post condiciones	El sistema actualiza el estado y los intentos de la notificación en la base de datos.	

Fuente: Creado por el autor

Tabla N° 31: Especificación de caso de uso Listar Configuraciones

Especificación de Caso de Uso		
Caso de Uso	Listar Configuraciones	
Precondiciones	El administrador debe estar autenticado.	
Flujo de Eventos	**Básico**	El administrador envía una solicitud GET al endpoint de listado. La API consulta la base de datos para obtener todas las configuraciones disponibles. La API devuelve una respuesta con el código de éxito (200 - OK) y una lista de las configuraciones encontradas.
	Alternativos	• Si no existen configuraciones para listar, la API responde con una lista vacía y el código de éxito (200 - OK). • Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	El administrador recibe los datos de las configuraciones disponibles.	

Fuente: Creado por el autor

Tabla N° 32: Especificación de caso de uso Actualizar Configuración

Especificación de Caso de Uso		
Caso de Uso	Actualizar Configuración	
Precondiciones	El administrador debe estar autenticado. La configuración a actualizar debe existir en la base de datos.	
Flujo de Eventos	**Básico**	El administrador envía una solicitud PUT con el valor actualizado de la configuración al endpoint correspondiente. La API valida los datos recibidos. La API actualiza la información de la configuración en la base de datos. La API devuelve una respuesta con el código de éxito (200 - OK) y los datos actualizados de la configuración.
	Alternativos	• Si los datos enviados son inválidos, la API responde con un error (400 - Bad Request). • Si la configuración no existe, la API responde con un error (404 - Not Found). • Si el administrador no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	El valor de la configuración es actualizado en la base de datos.	

Fuente: Creado por el autor

Tabla N° 33: Especificación de caso de uso Obtener Reporte de Notificaciones Enviadas

Especificación de Caso de Uso		
Caso de Uso	Obtener Reporte de Notificaciones Enviadas	
Precondiciones	El usuario debe estar autenticado.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud GET al endpoint correspondiente con el intervalo de fechas para el reporte. La API consulta la base de datos para obtener todas las notificaciones enviadas durante el intervalo de fechas especificado. La API genera el reporte de notificaciones enviadas y fallidas según cliente y canal de envío. La API devuelve una respuesta con el código de éxito (200 - OK) y los datos del reporte.
	Alternativos	• Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	El usuario recibe los datos del reporte.	

Fuente: Creado por el autor

Tabla N° 34: Especificación de caso de uso Obtener Reporte de Tiempos de Envío

Especificación de Caso de Uso		
Caso de Uso	Obtener Reporte de Tiempos de Envío	
Precondiciones	El usuario debe estar autenticado.	
Flujo de Eventos	**Básico**	El usuario envía una solicitud GET al endpoint correspondiente con el intervalo de fechas para el reporte. La API consulta la base de datos para obtener todas las notificaciones enviadas durante el intervalo de fechas especificado. La API genera el reporte de tiempos de envío y demoras según cliente y canal de envío. La API devuelve una respuesta con el código de éxito (200 - OK) y los datos del reporte.
	Alternativos	• Si el usuario no está autenticado, la API responde con un error (401 - Unauthorized).
Post condiciones	El usuario recibe los datos del reporte.	

Fuente: Creado por el autor

6.2.2. Análisis de Robustez

El inicio de esta fase requiere contar con un borrador de las especificaciones de cada caso de uso, el cual será empleado en los respectivos diagramas de robustez.

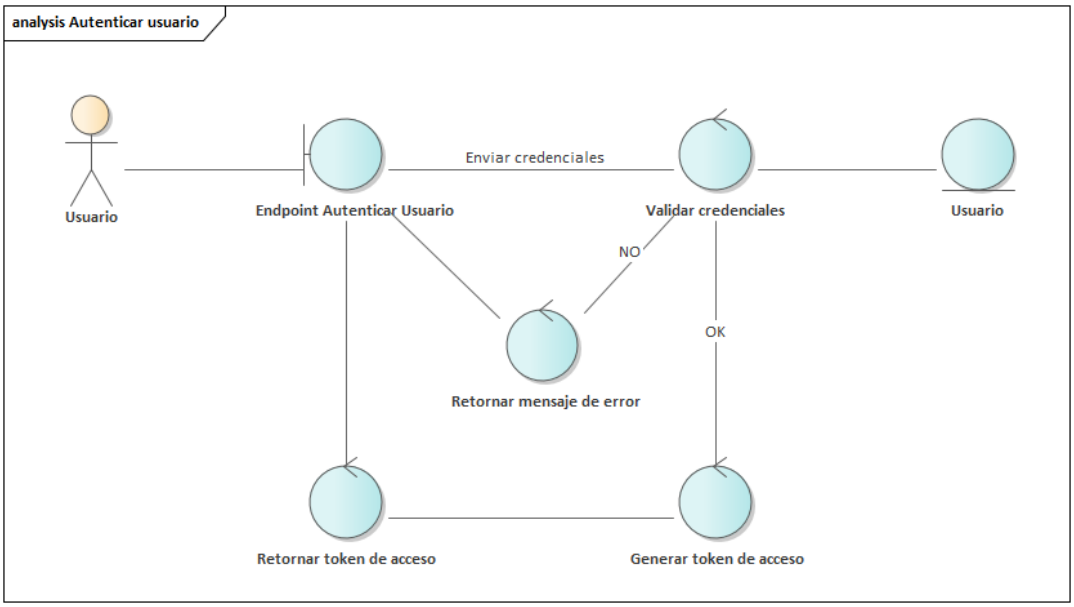


Figura N° 30: Diagrama de robustez Autenticar Usuario
Fuente: Creado por el autor

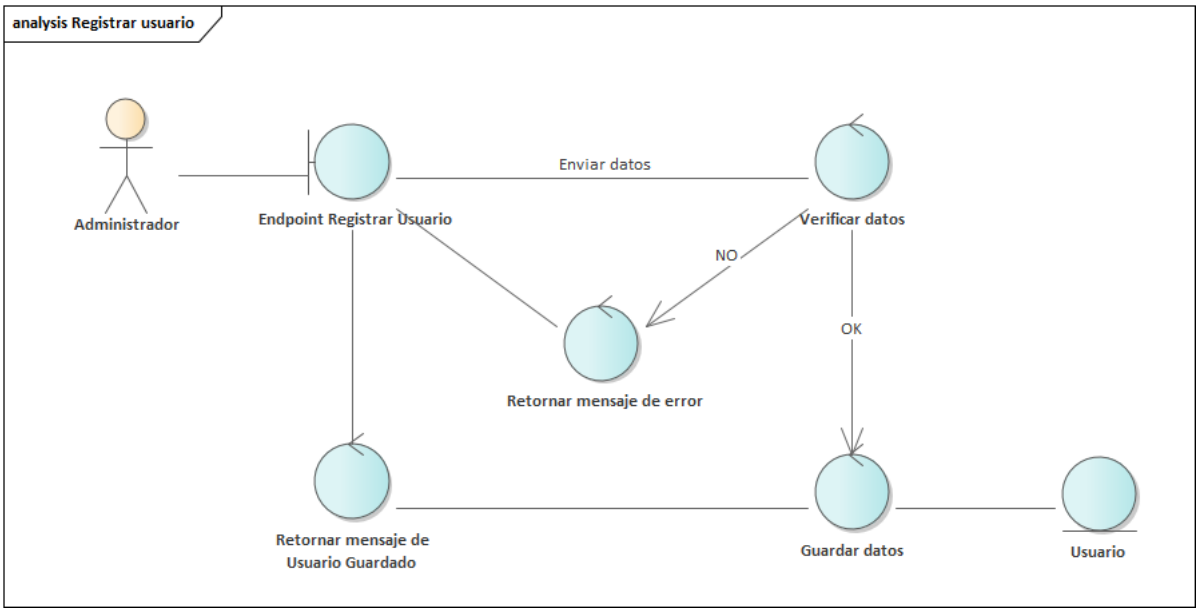


Figura N° 31: Diagrama de robustez Registrar Usuario
Fuente: Creado por el autor

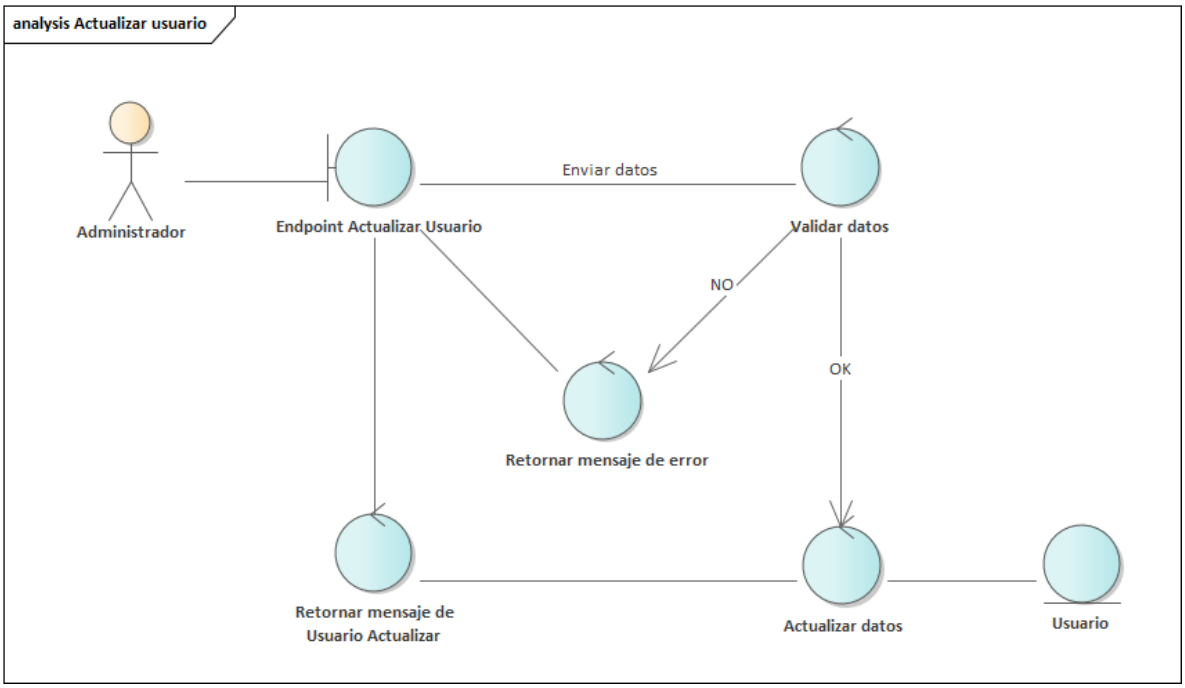


Figura N° 32: Diagrama de robustez Actualizar Usuario
Fuente: Creado por el autor

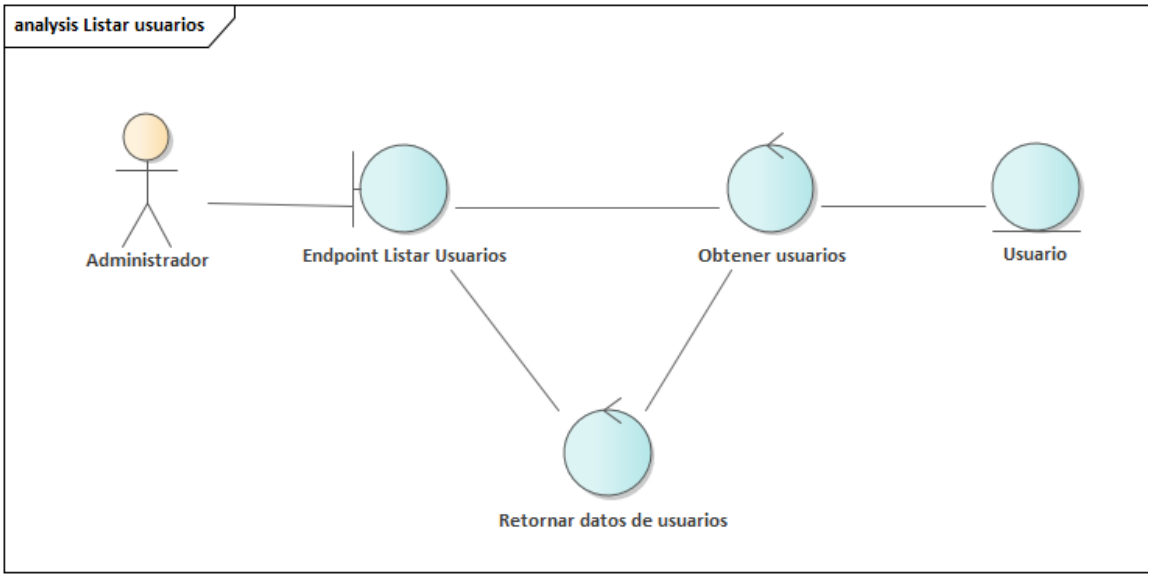


Figura N° 33: Diagrama de robustez Listar Usuarios
Fuente: Creado por el autor

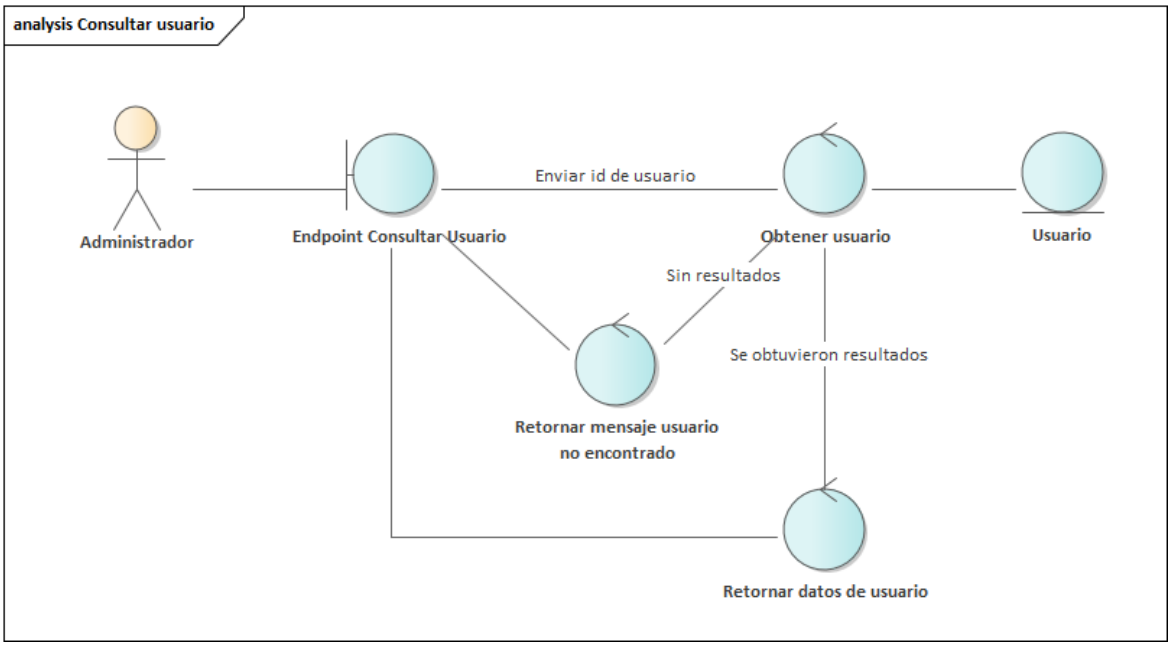


Figura N° 34: Diagrama de robustez Consultar Usuario
Fuente: Creado por el autor

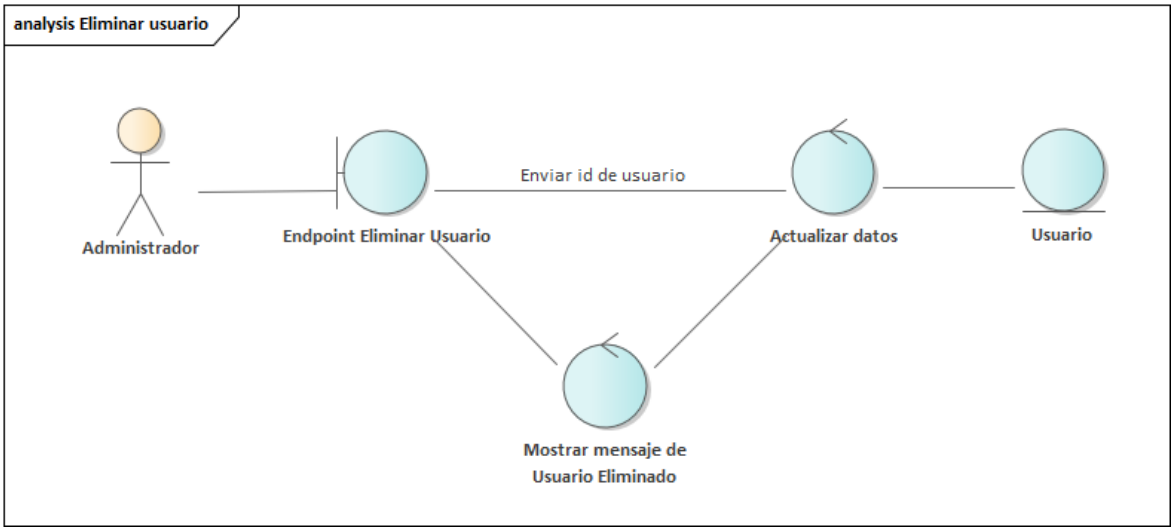


Figura N° 35: Diagrama de robustez Eliminar Usuario
Fuente: Creado por el autor

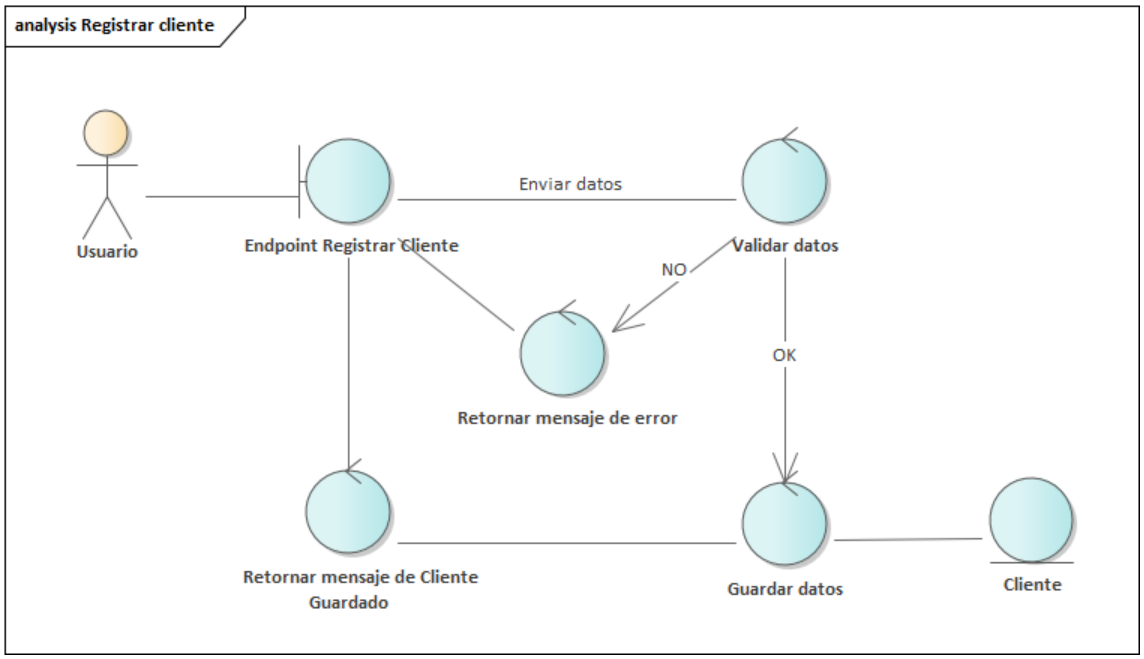


Figura N° 36: Diagrama de robustez Registrar Cliente
Fuente: Creado por el autor

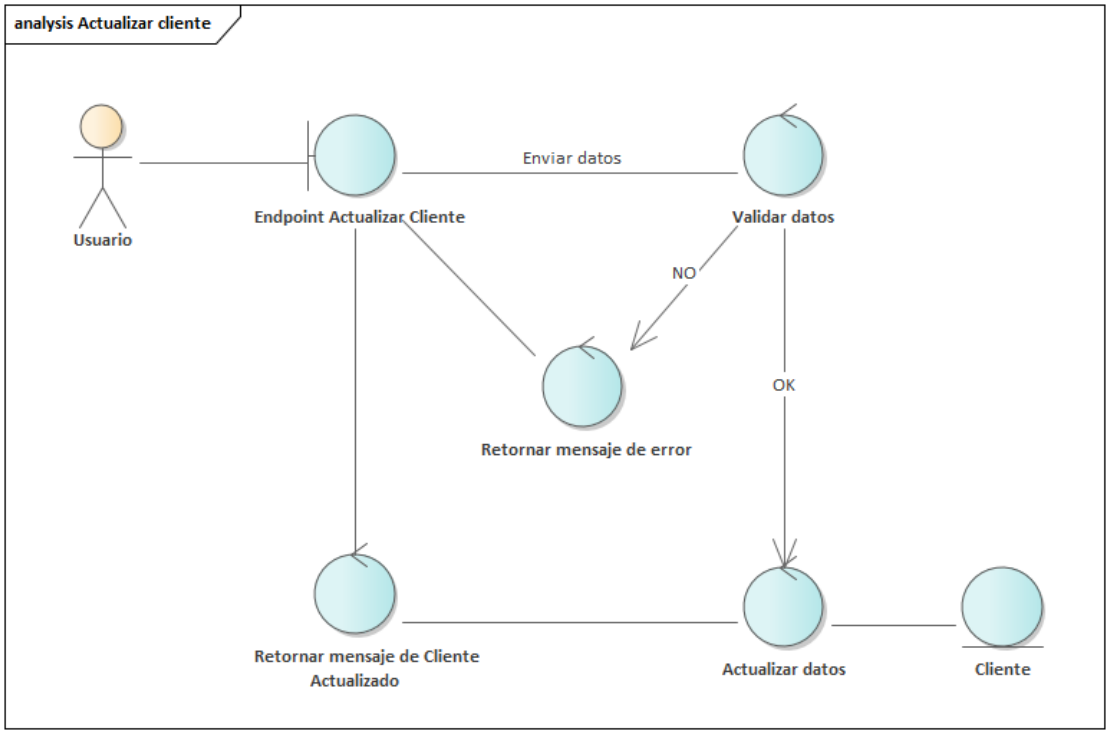


Figura N° 37: Diagrama de robustez Actualizar Cliente
Fuente: Creado por el autor

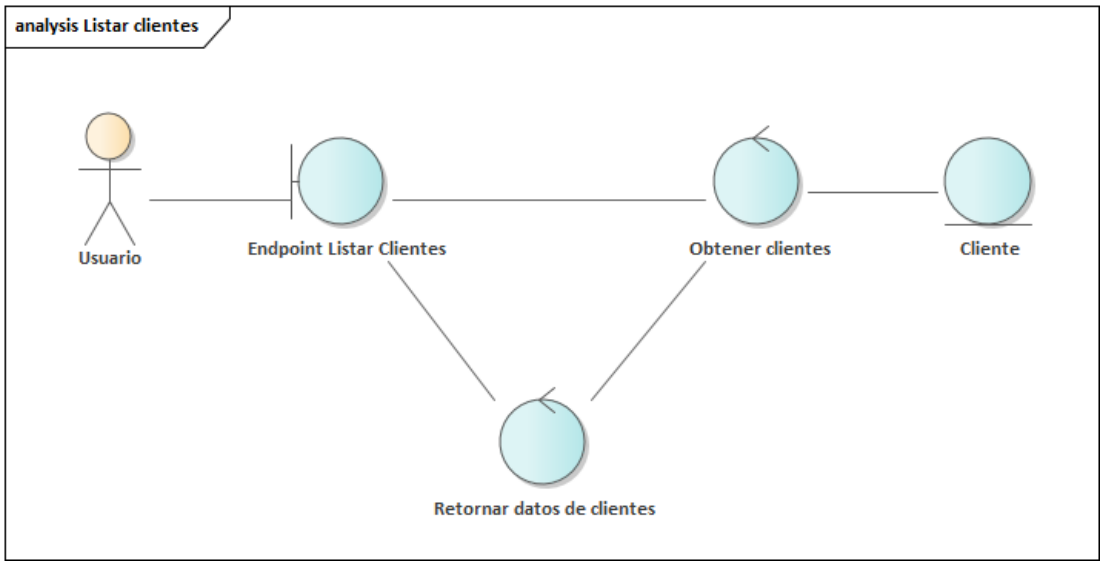


Figura N° 38: Diagrama de robustez Listar Clientes
Fuente: Creado por el autor

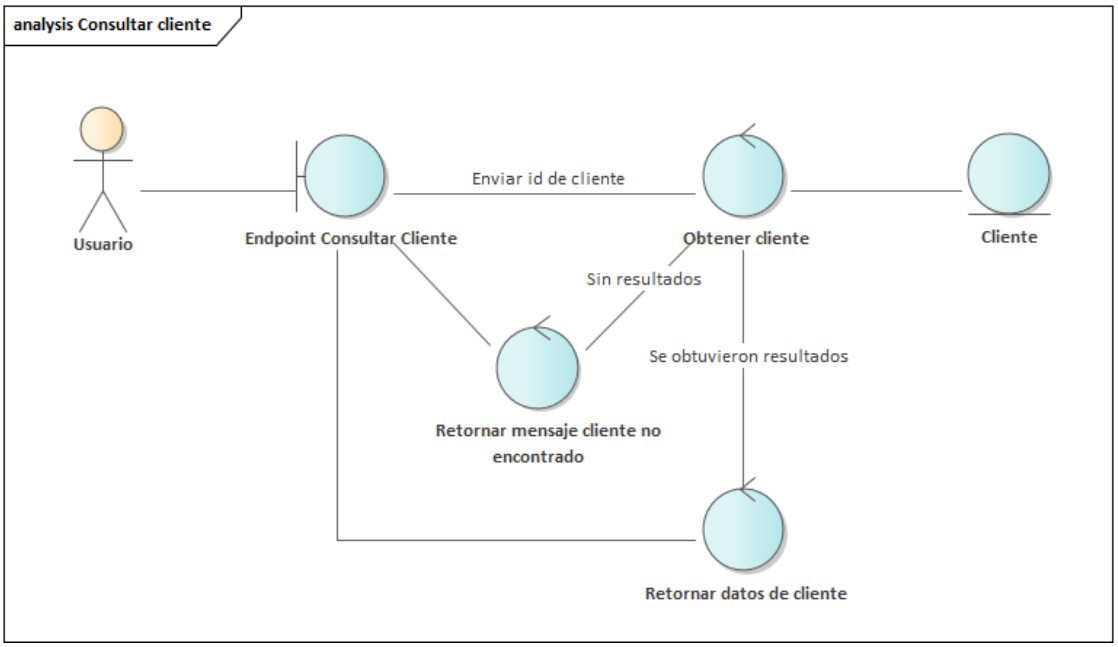


Figura N° 39: Diagrama de robustez Consultar Cliente
Fuente: Creado por el autor

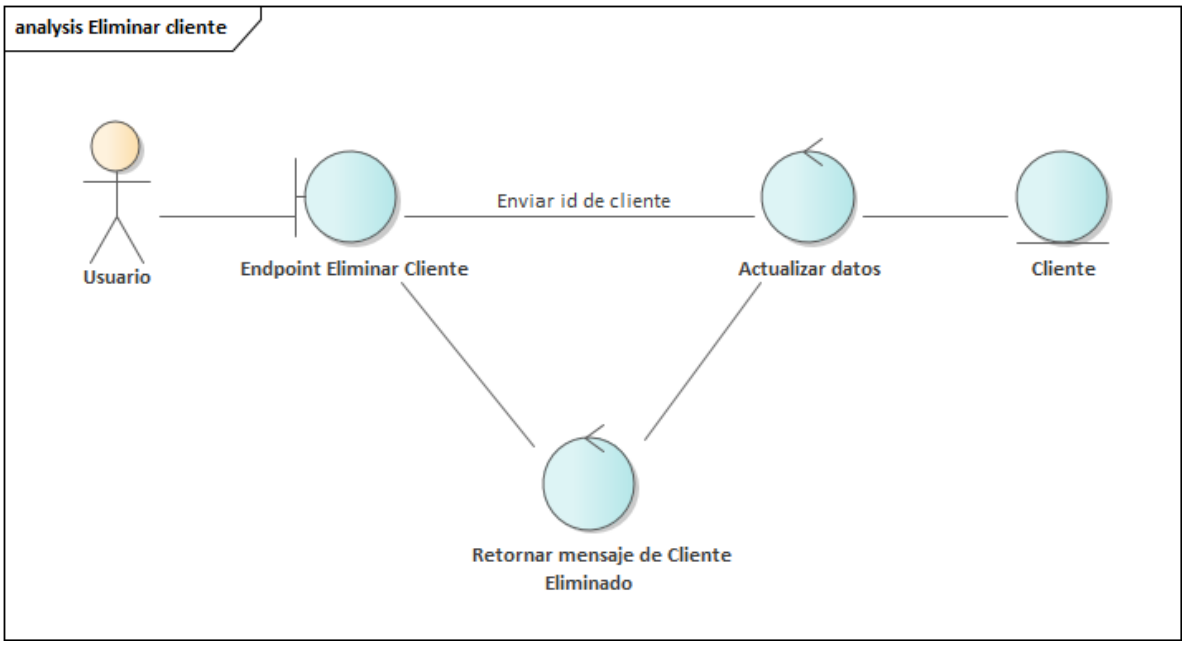


Figura N° 40: Diagrama de robustez Eliminar Cliente
Fuente: Creado por el autor

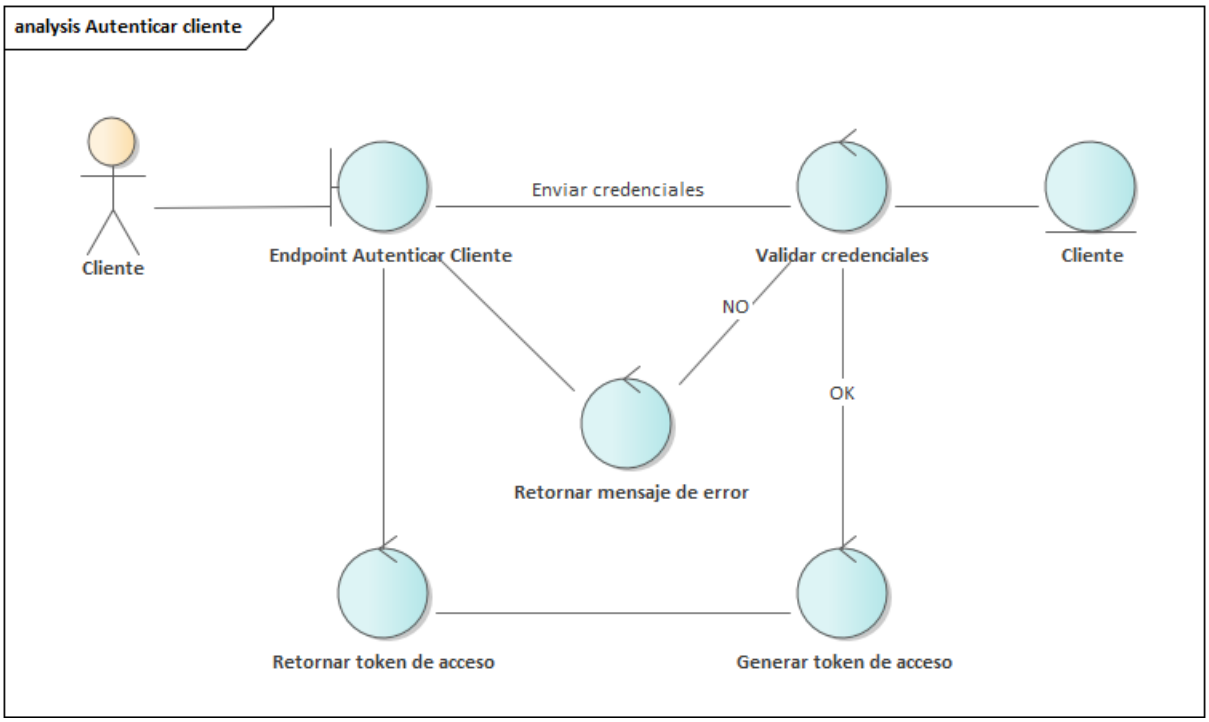


Figura N° 41: Diagrama de robustez Autenticar Cliente
Fuente: Creado por el autor

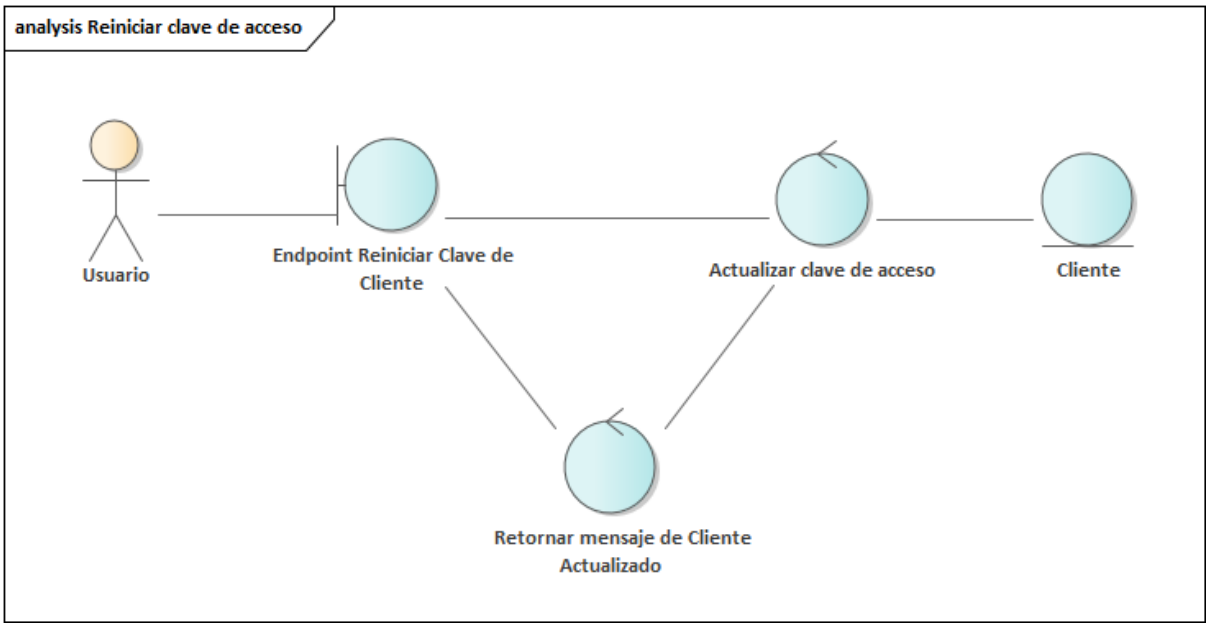


Figura N° 42: Diagrama de robustez Reiniciar Clave de Acceso de Cliente
Fuente: Creado por el autor

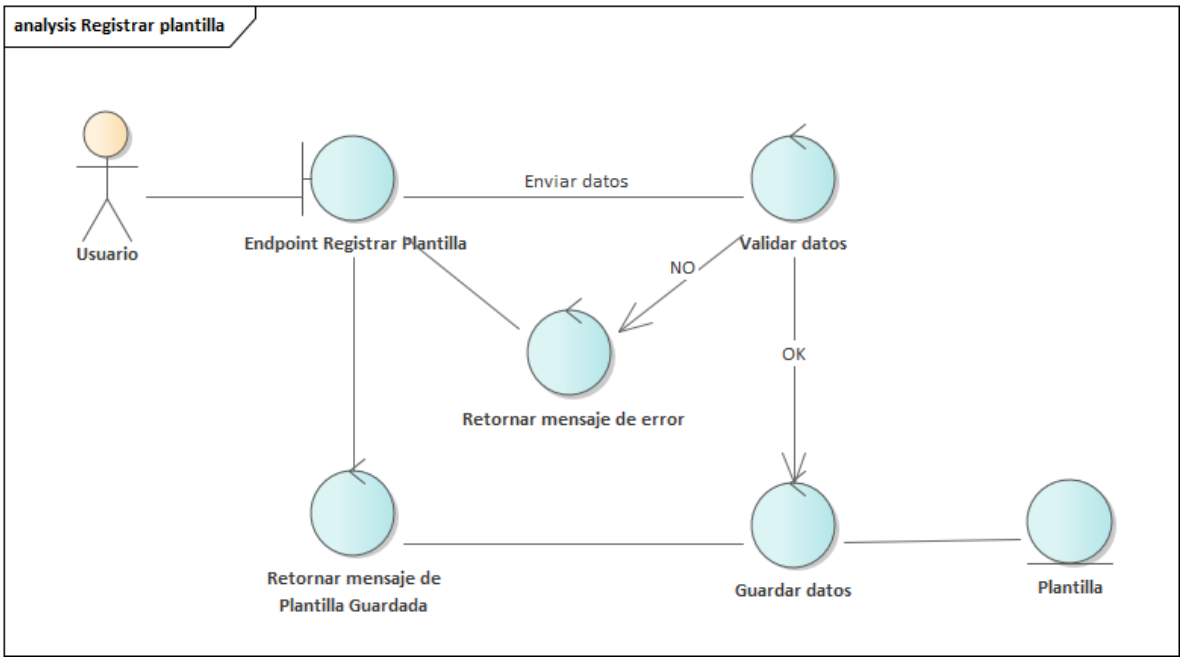


Figura N° 43: Diagrama de robustez Registrar Plantilla
Fuente: Creado por el autor

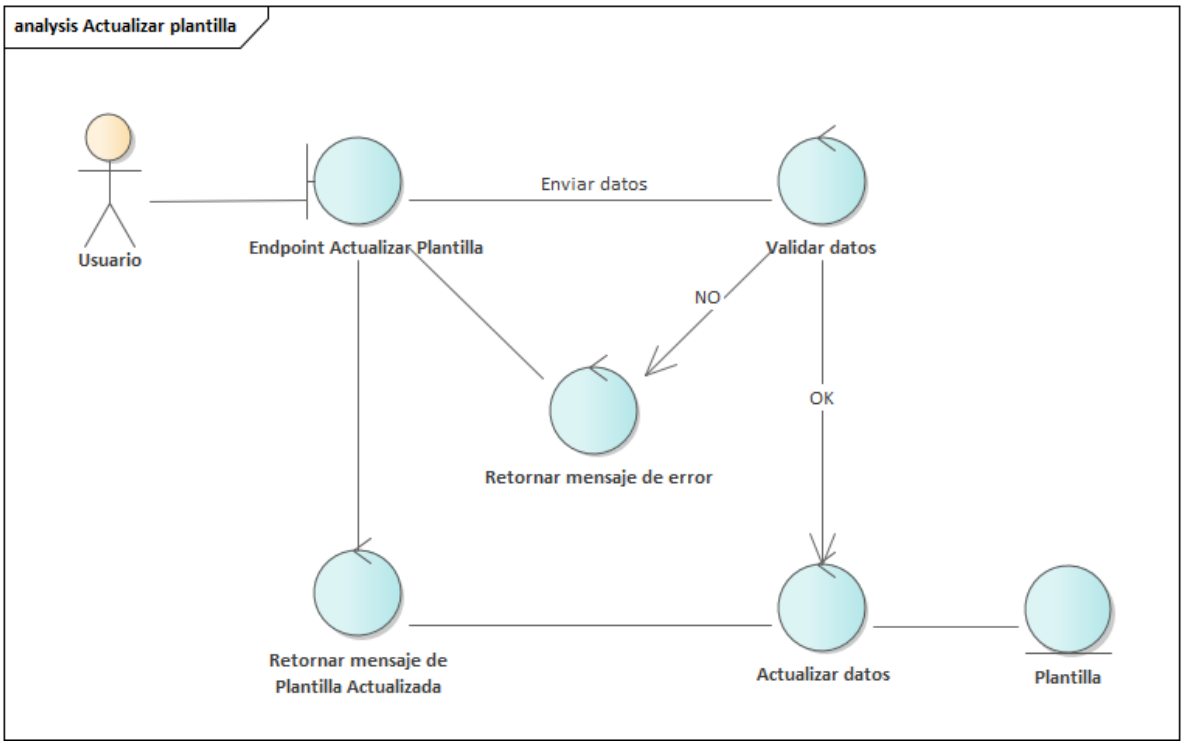


Figura N° 44: Diagrama de robustez Actualizar Plantilla
Fuente: Creado por el autor

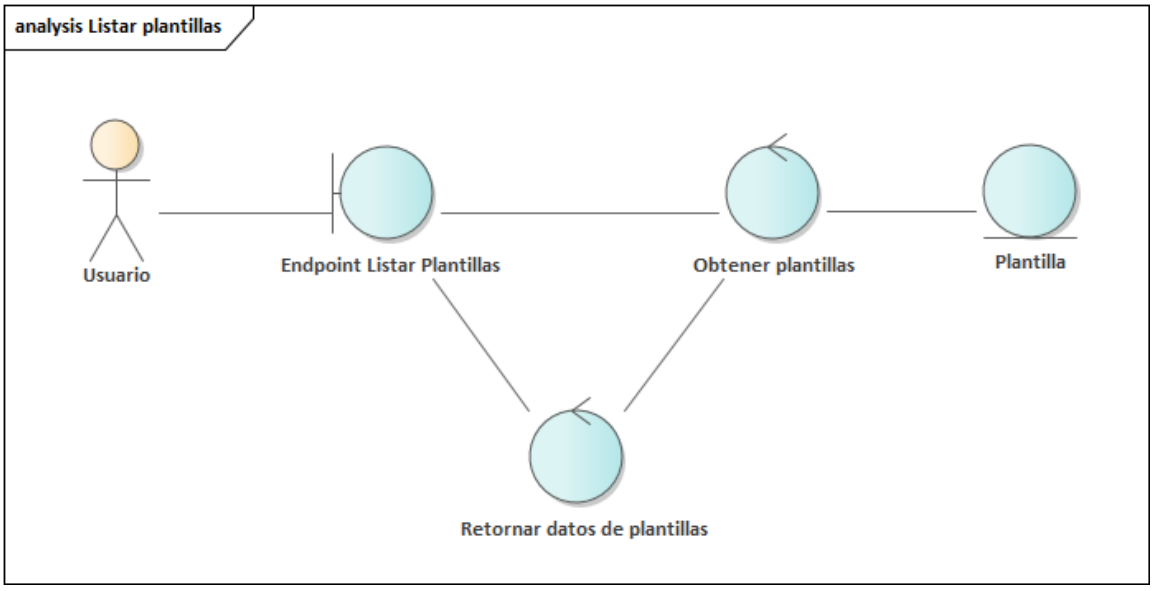


Figura N° 45: Diagrama de robustez Listar Plantillas
Fuente: Creado por el autor

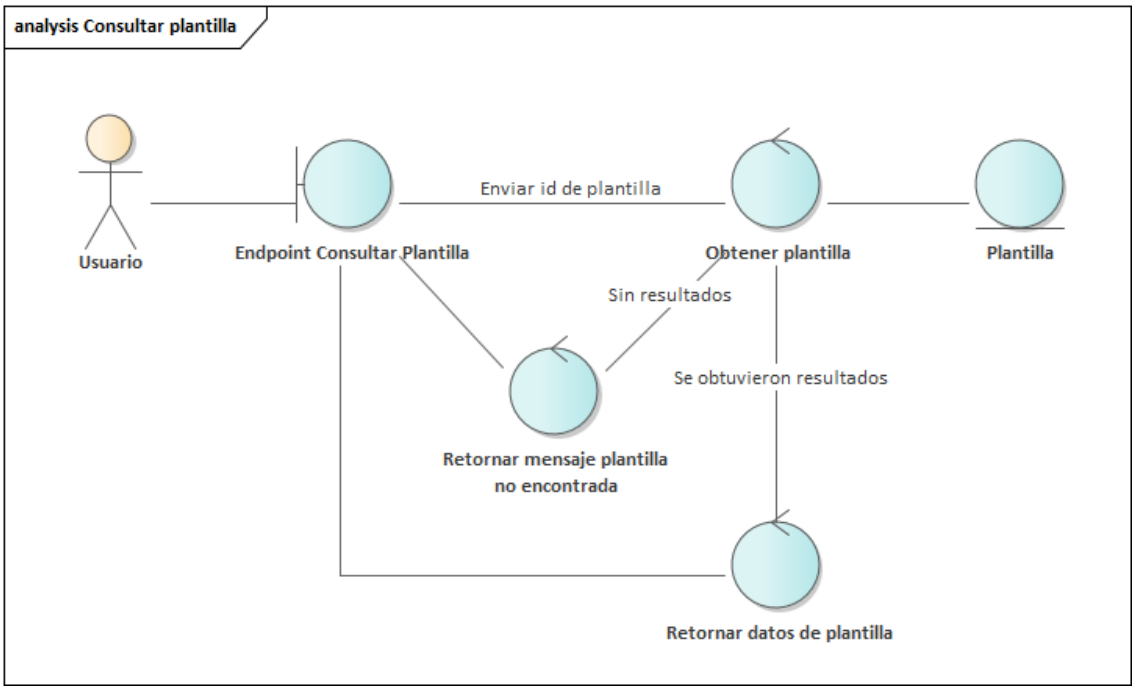


Figura N° 46: Diagrama de robustez Consultar Plantilla
Fuente: Creado por el autor

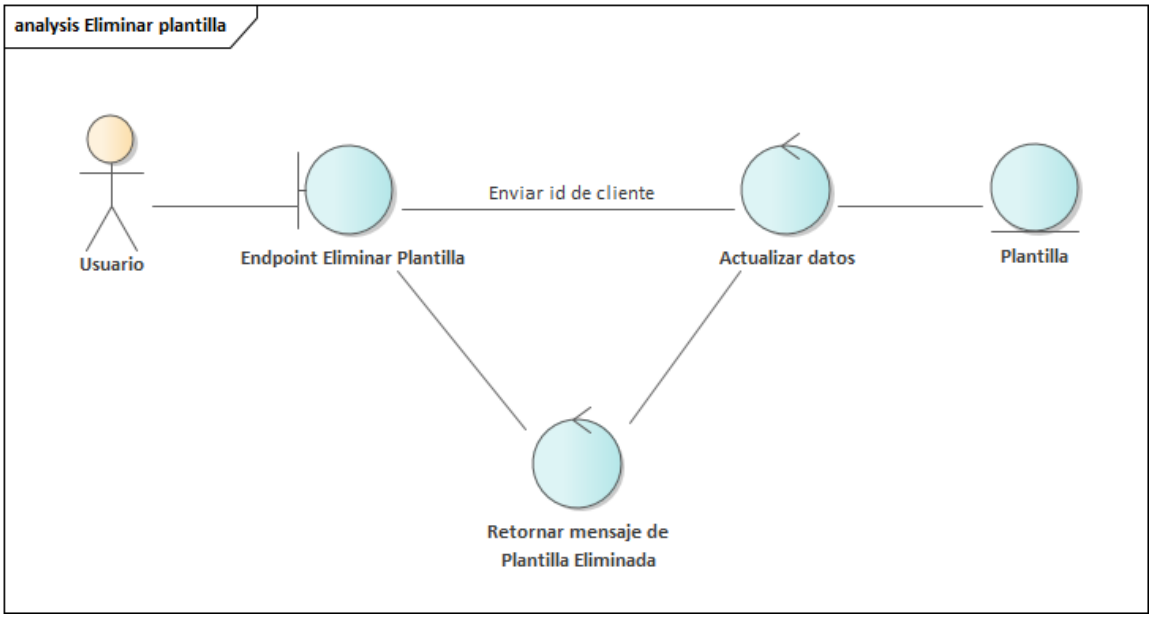


Figura N° 47: Diagrama de robustez Eliminar Plantilla
Fuente: Creado por el autor

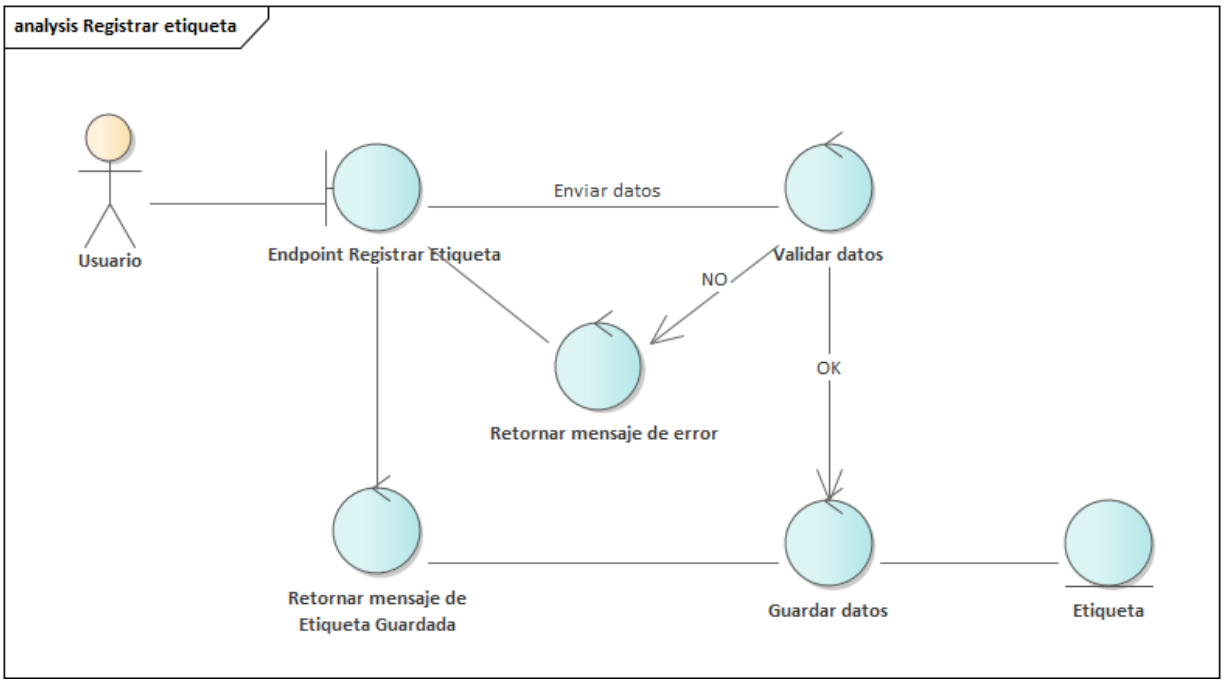


Figura N° 48: Diagrama de robustez Registrar Etiqueta
Fuente: Creado por el autor

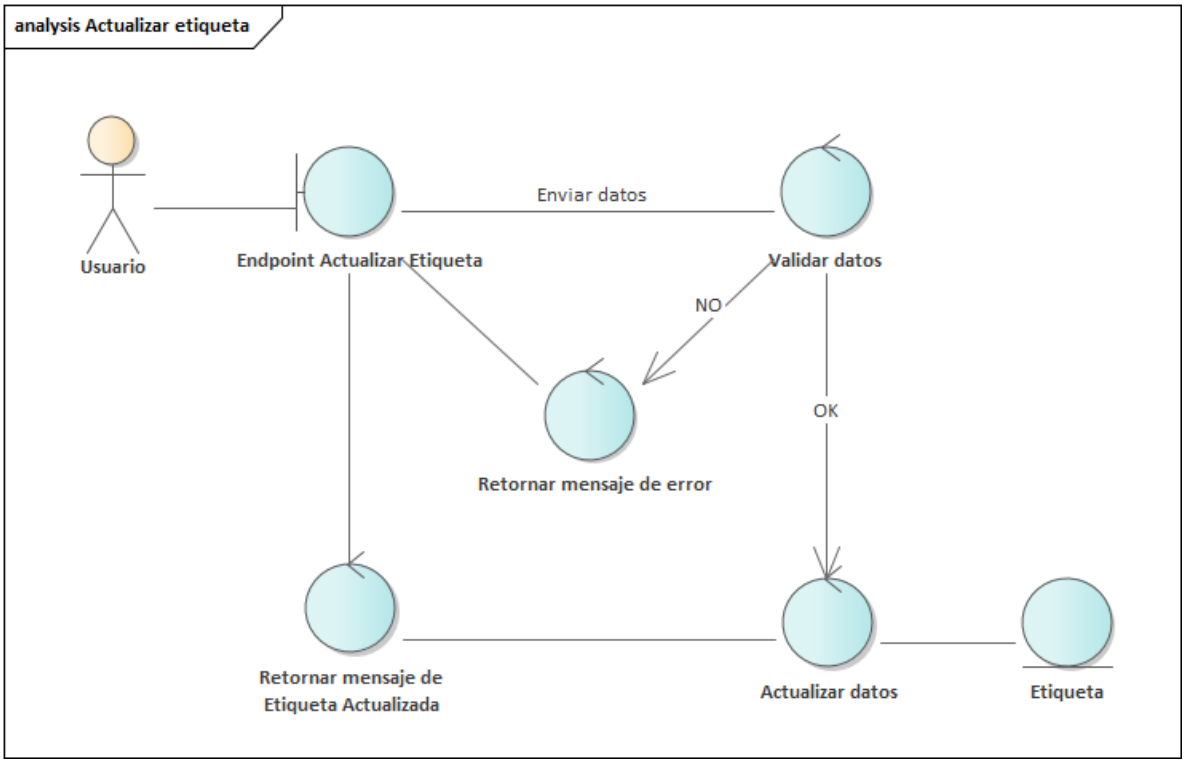


Figura N° 49: Diagrama de robustez Actualizar Etiqueta
Fuente: Creado por el autor

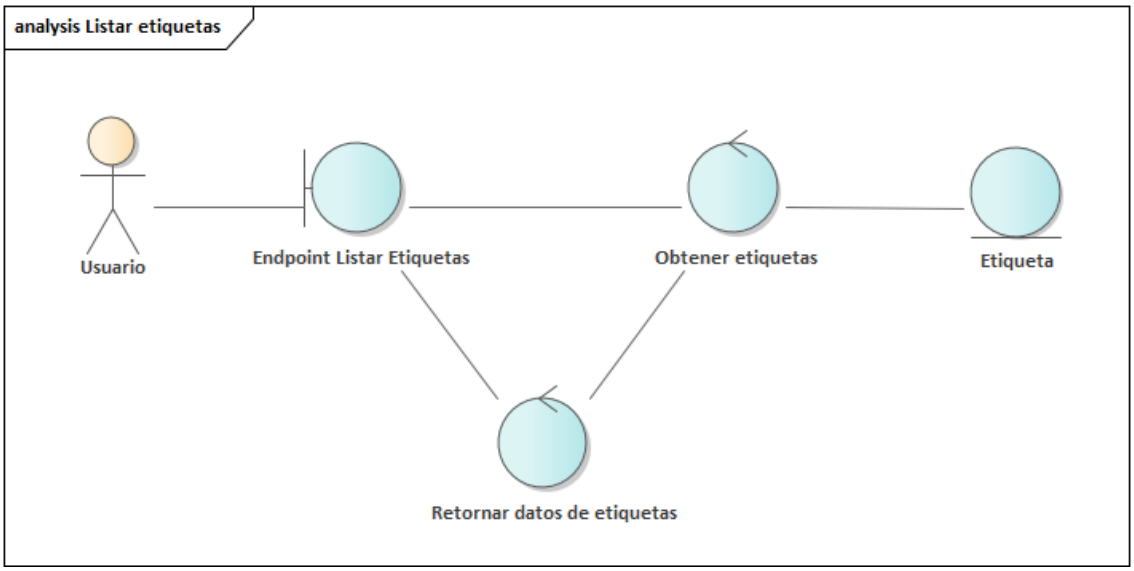


Figura N° 50: Diagrama de robustez Listar Etiquetas
Fuente: Creado por el autor

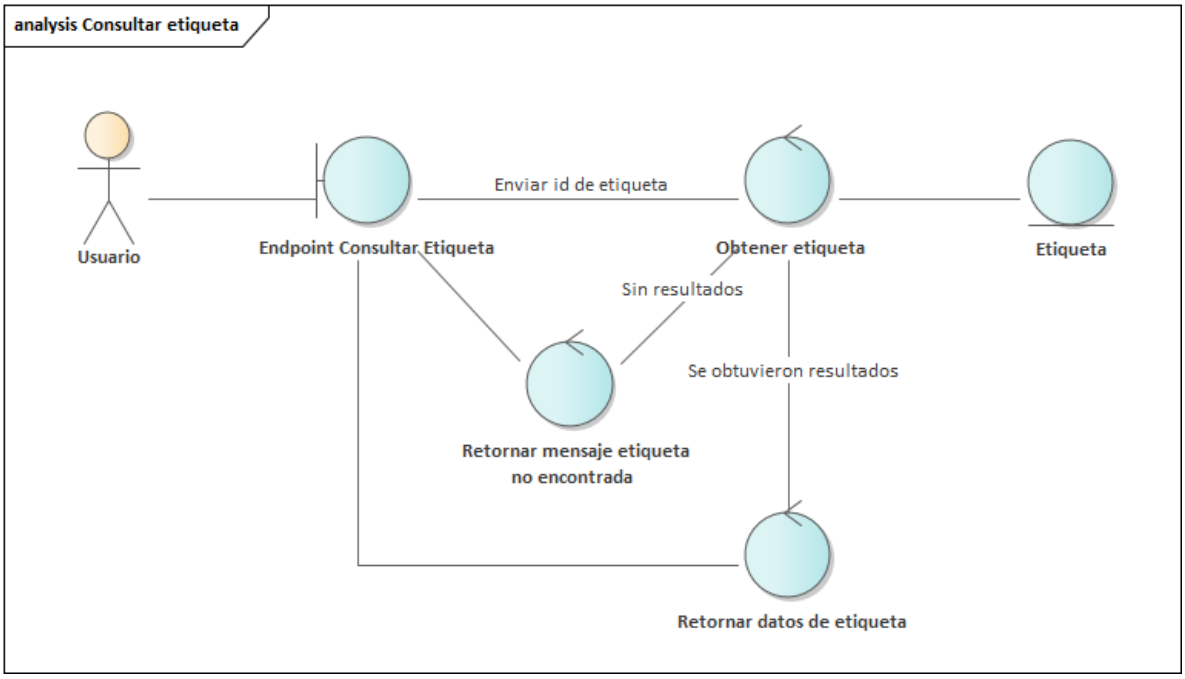


Figura N° 51: Diagrama de robustez Consultar Etiqueta
Fuente: Creado por el autor

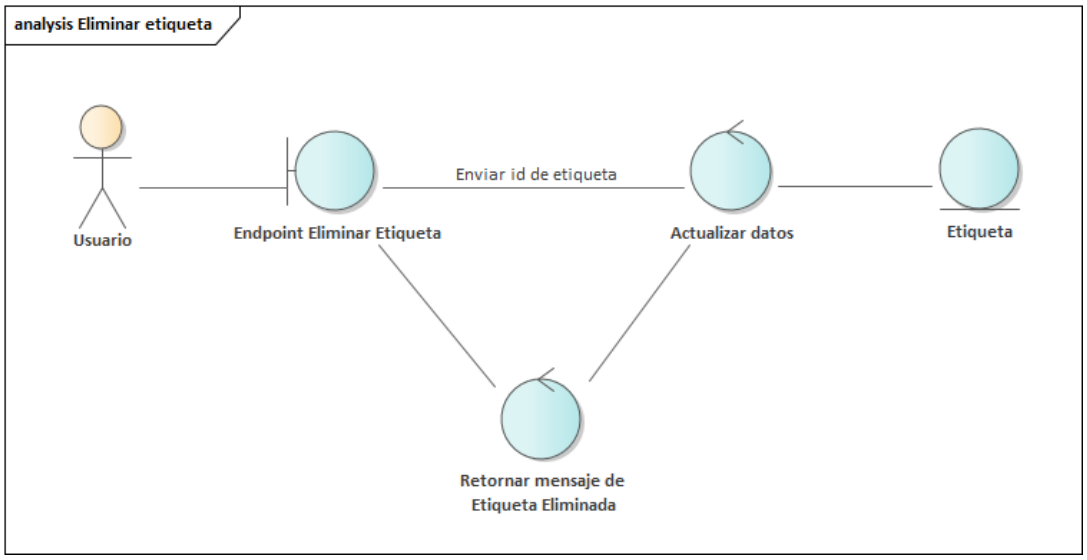


Figura N° 52: Diagrama de robustez Eliminar Etiqueta
Fuente: Creado por el autor

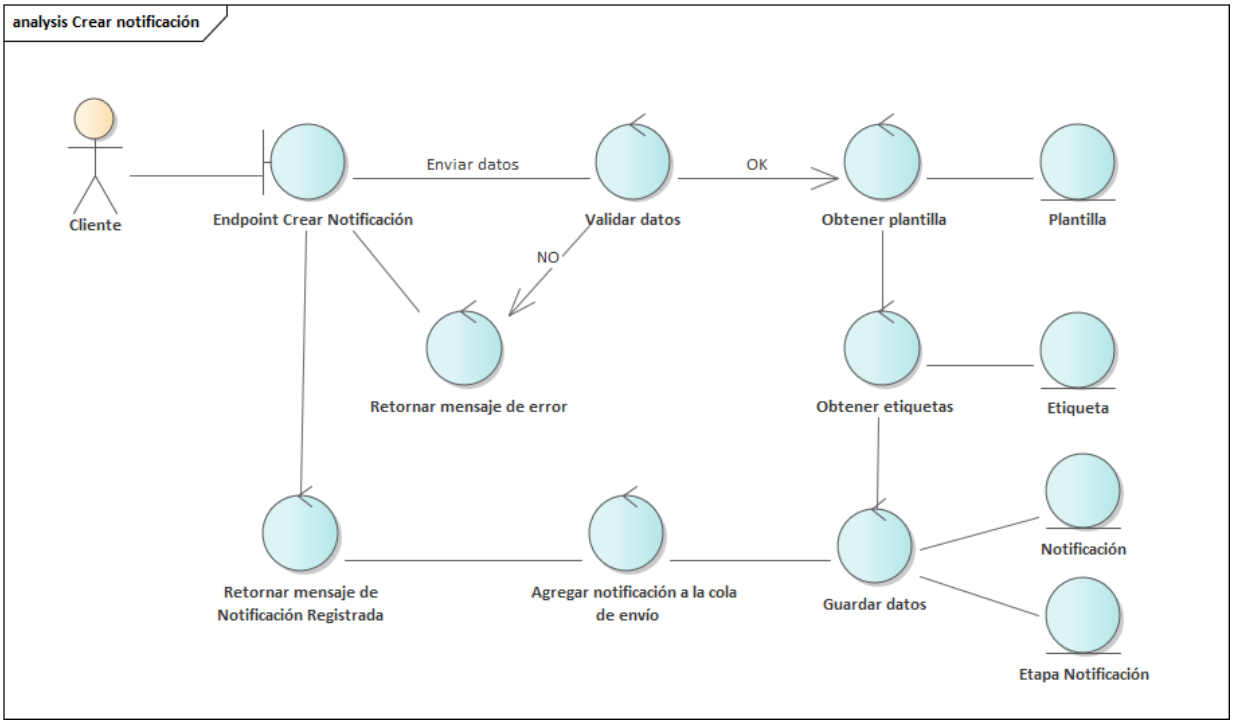


Figura N° 53: Diagrama de robustez Crear Notificación
Fuente: Creado por el autor

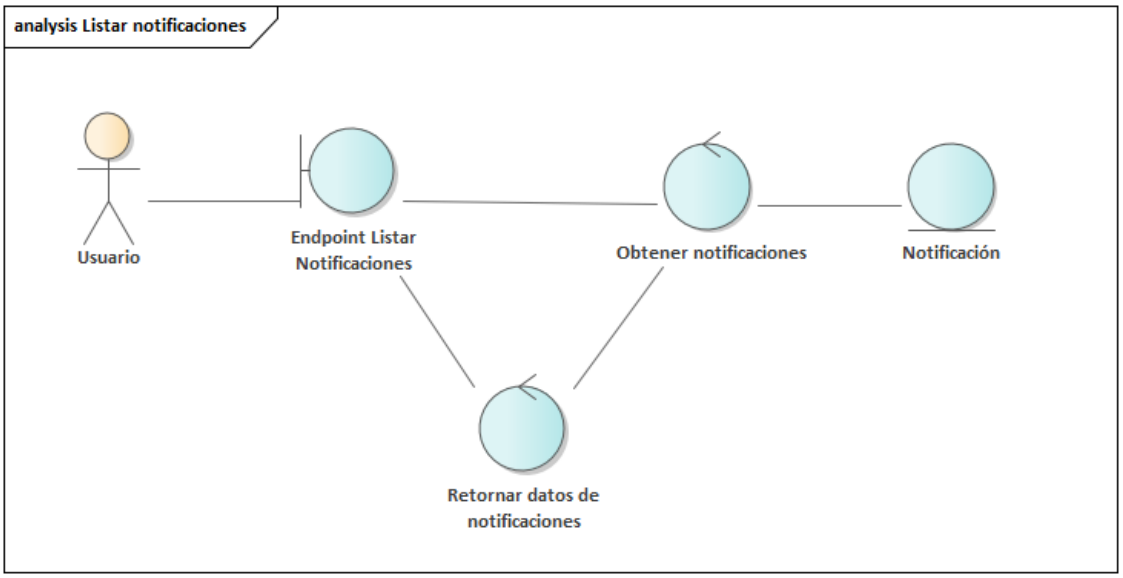


Figura N° 54: Diagrama de robustez Listar Notificaciones
Fuente: Creado por el autor

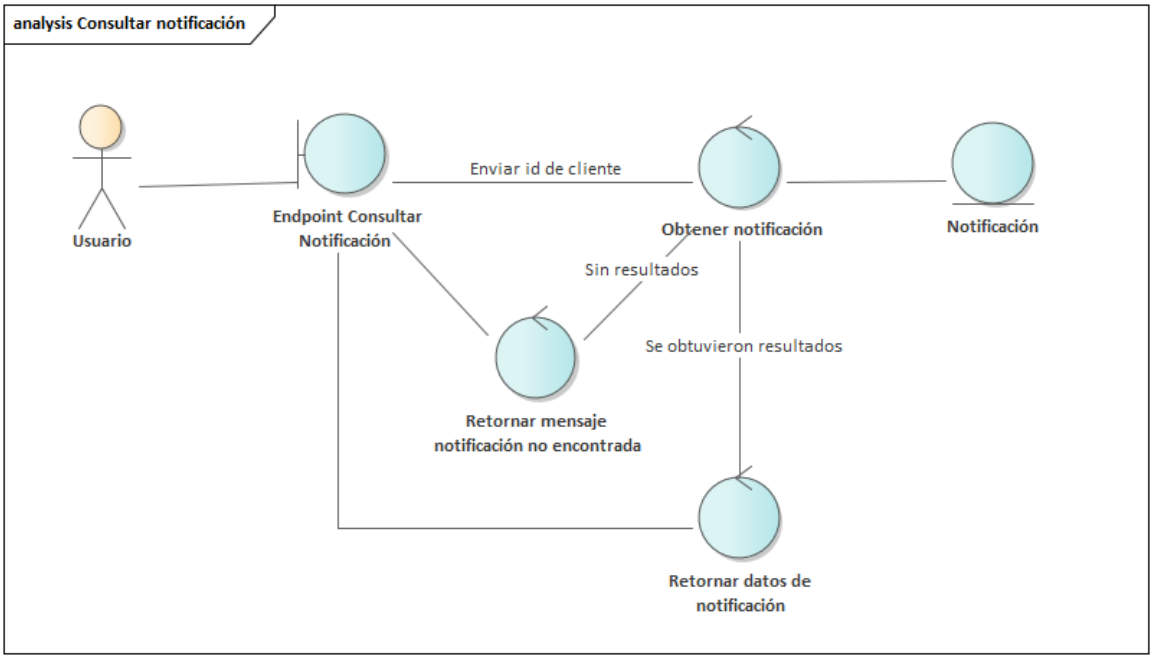


Figura N° 55: Diagrama de robustez Consultar Notificación
Fuente: Creado por el autor

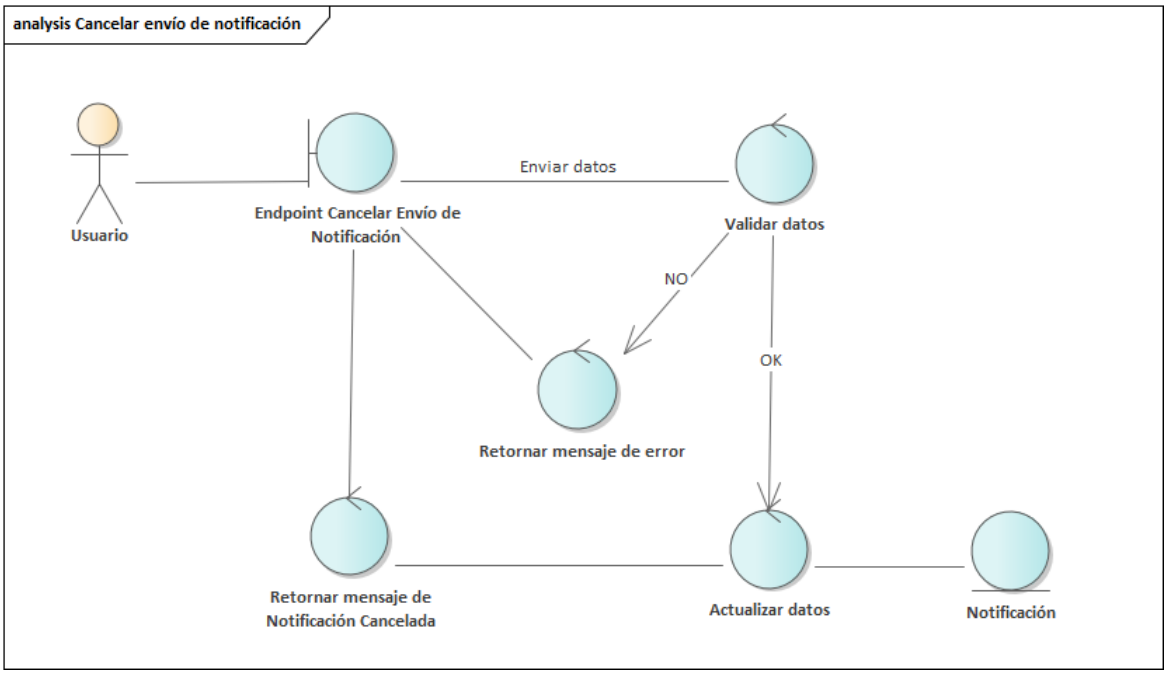


Figura N° 56: Diagrama de robustez Cancelar Envío de Notificación
Fuente: Creado por el autor

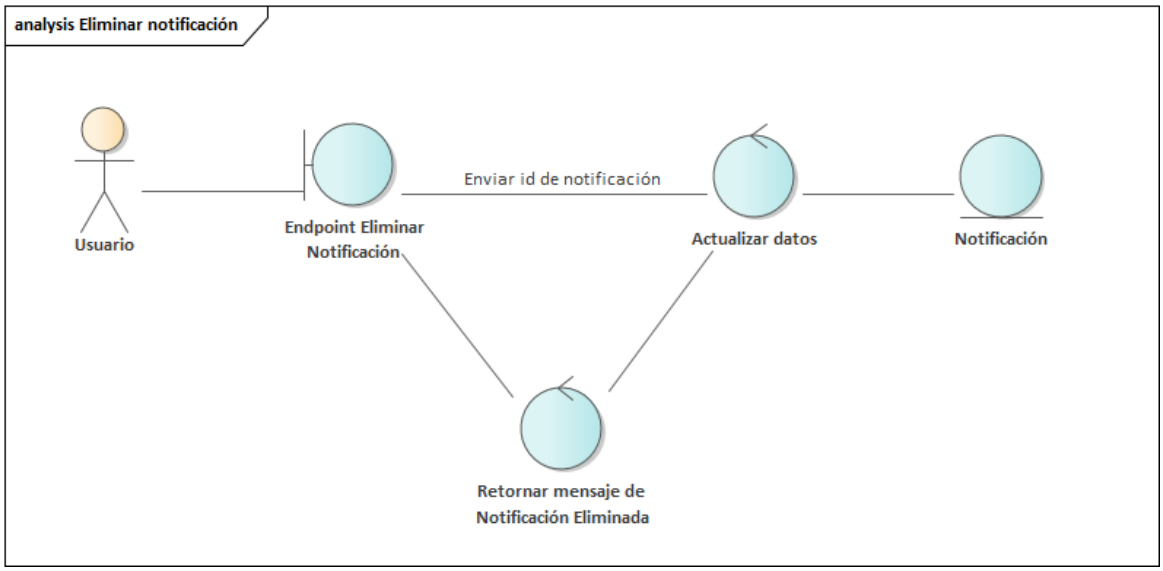


Figura N° 57: Diagrama de robustez Eliminar Notificación
Fuente: Creado por el autor

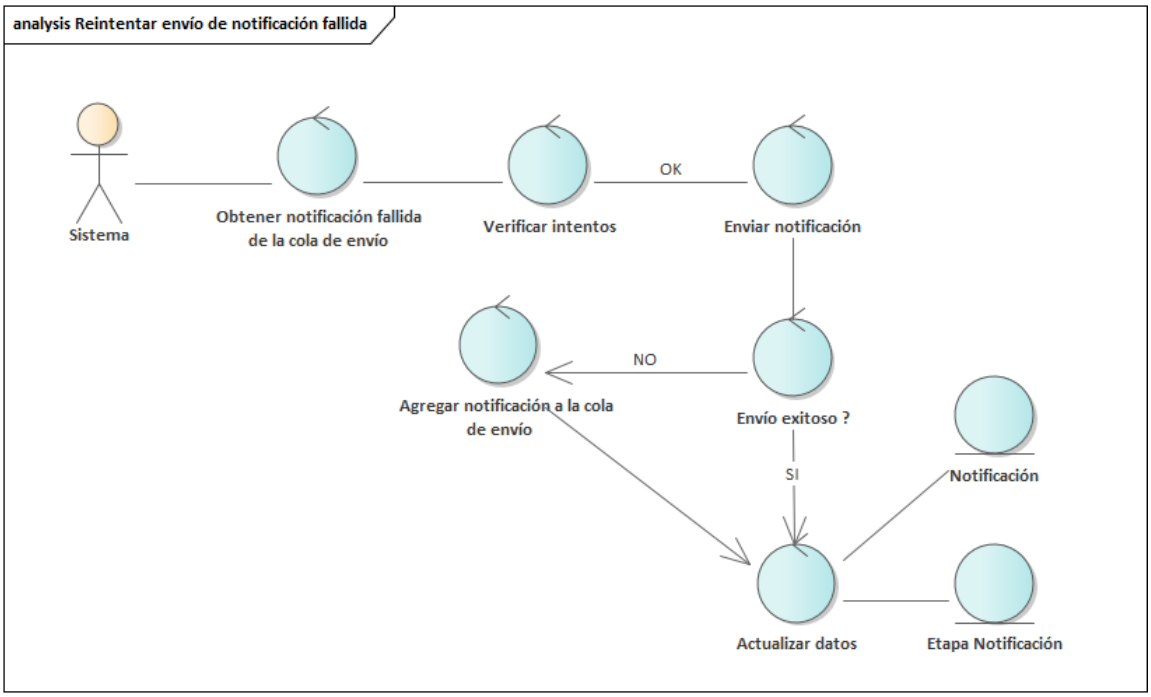


Figura N° 58: Diagrama de robustez Reintentar Envío de Notificación Fallida
Fuente: Creado por el autor

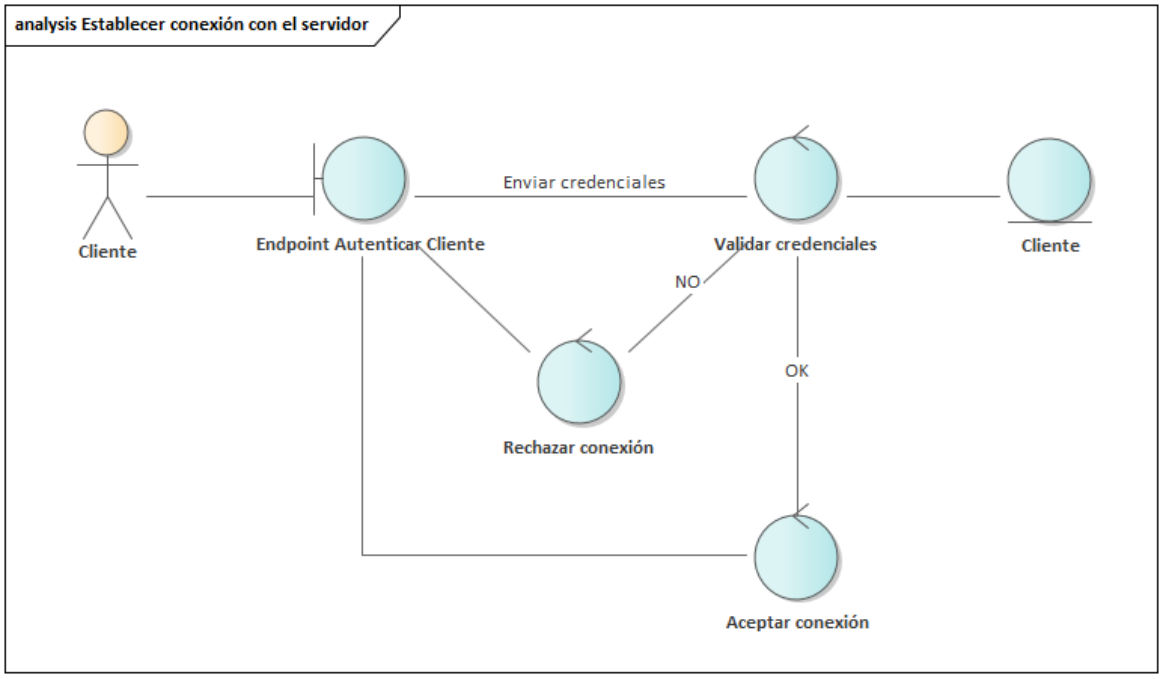


Figura N° 59: Diagrama de robustez Establecer conexión con el servidor
Fuente: Creado por el autor

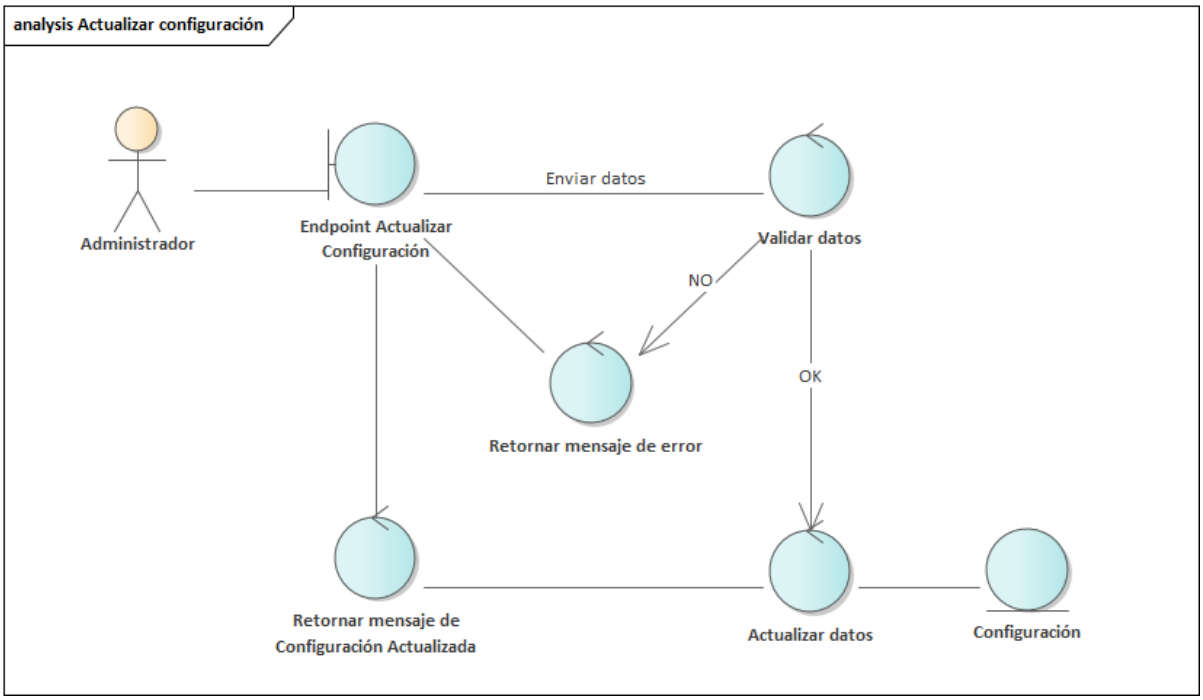


Figura N° 60: Diagrama de robustez Actualizar Configuración
Fuente: Creado por el autor

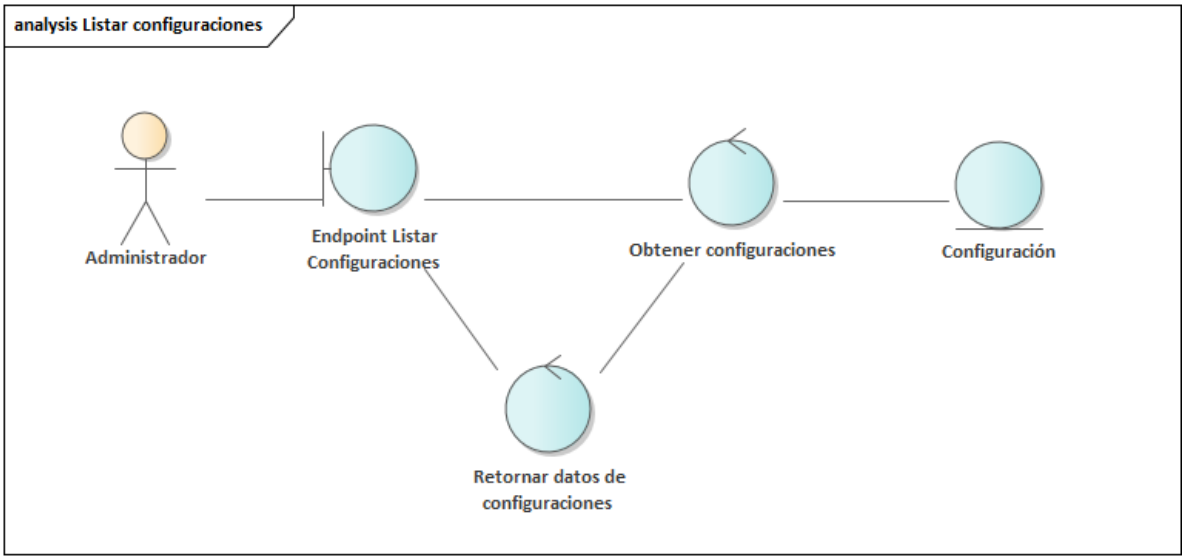


Figura N° 61: Diagrama de robustez Listar Configuraciones
Fuente: Creado por el autor

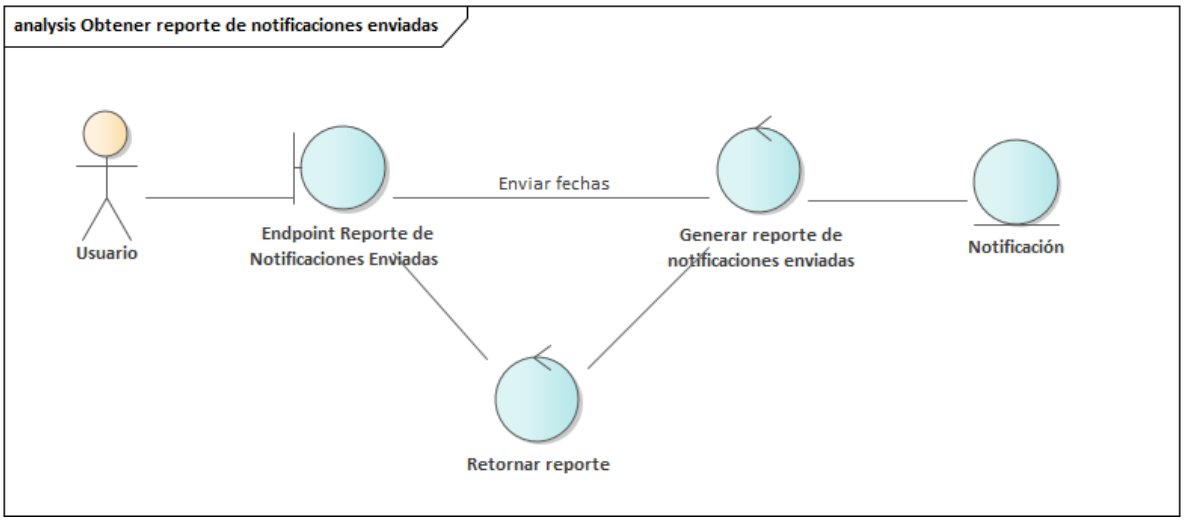


Figura N° 62: Diagrama de robustez Obtener Reporte de Notificaciones Enviadas
Fuente: Creado por el autor

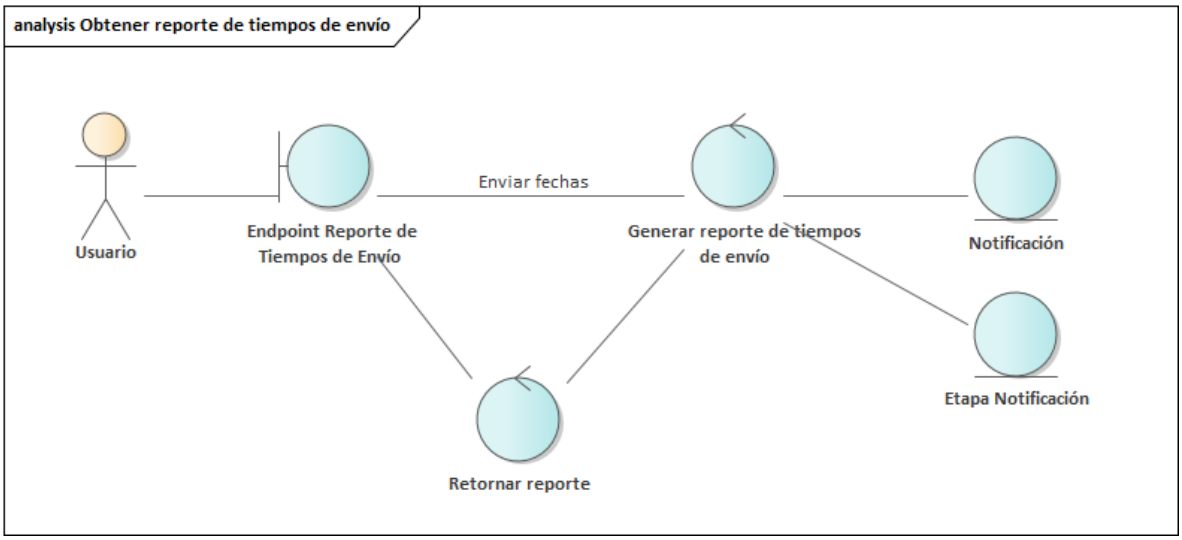


Figura N° 63: Diagrama de robustez Obtener Reporte de Tiempos de Envíos
Fuente: Creado por el autor

6.3. ARQUITECTURA TÉCNICA

La arquitectura técnica tiene como finalidad ofrecer una visión de la infraestructura tecnológica que se empleará en el desarrollo de la API.

Su función es definir la estructura del sistema a construir. Esta arquitectura se diseña para cumplir con los requisitos de negocio y los estándares de nivel de servicio establecidos para el sistema en desarrollo.

Además, abarca la topología del sistema, incluyendo aspectos como los nodos del servidor, su ubicación en la red, la selección del servidor de aplicaciones y el gestor de base de datos, entre otros.

6.3.1. Arquitectura de la Aplicación

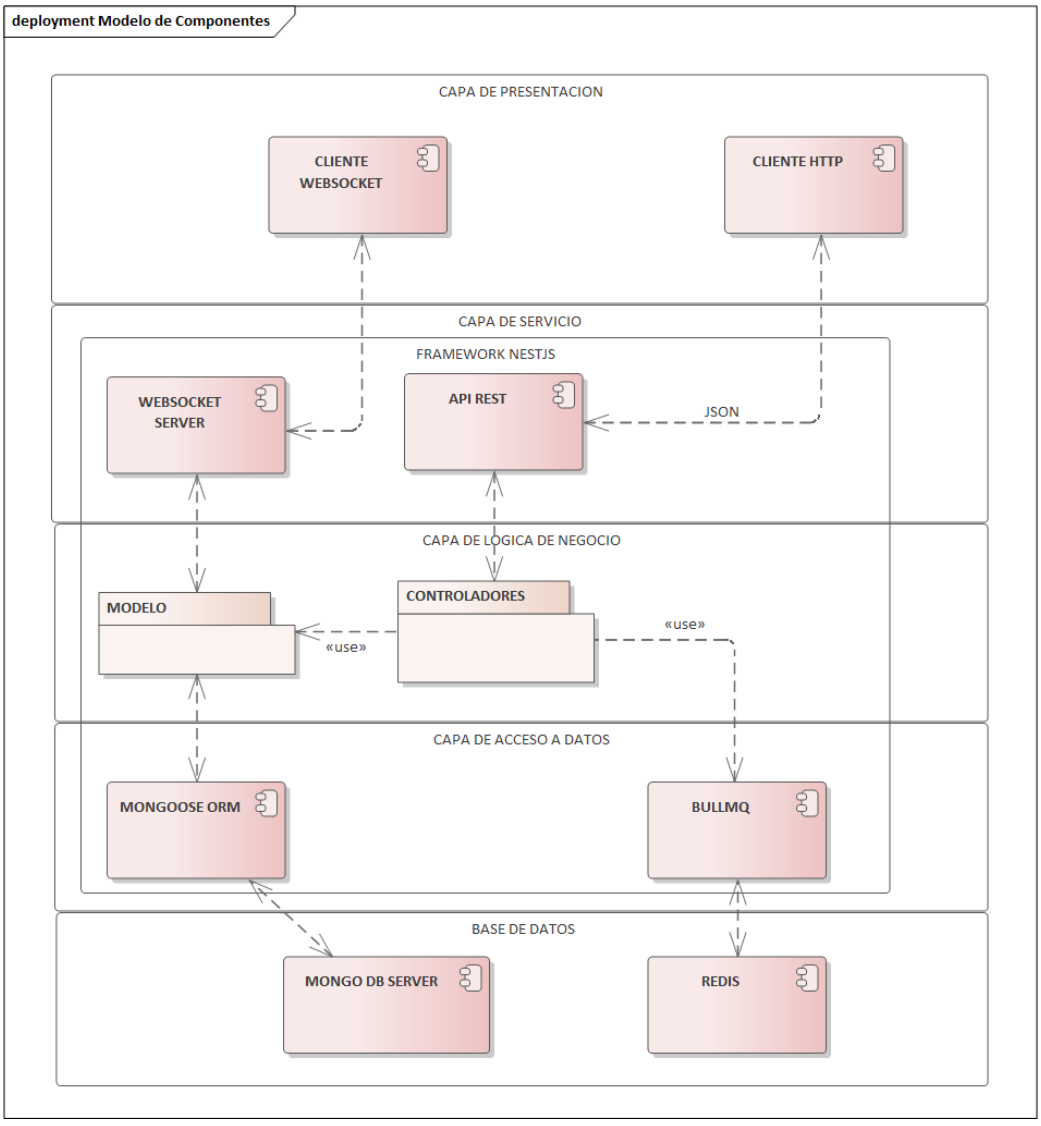


Figura N° 64: Diagrama de componentes del sistema

Fuente: Creado por el autor

6.4. DISEÑO DETALLADO

Durante esta fase, se llevará a cabo un diseño detallado de los casos de uso a través de la implementación de diagramas de secuencia.

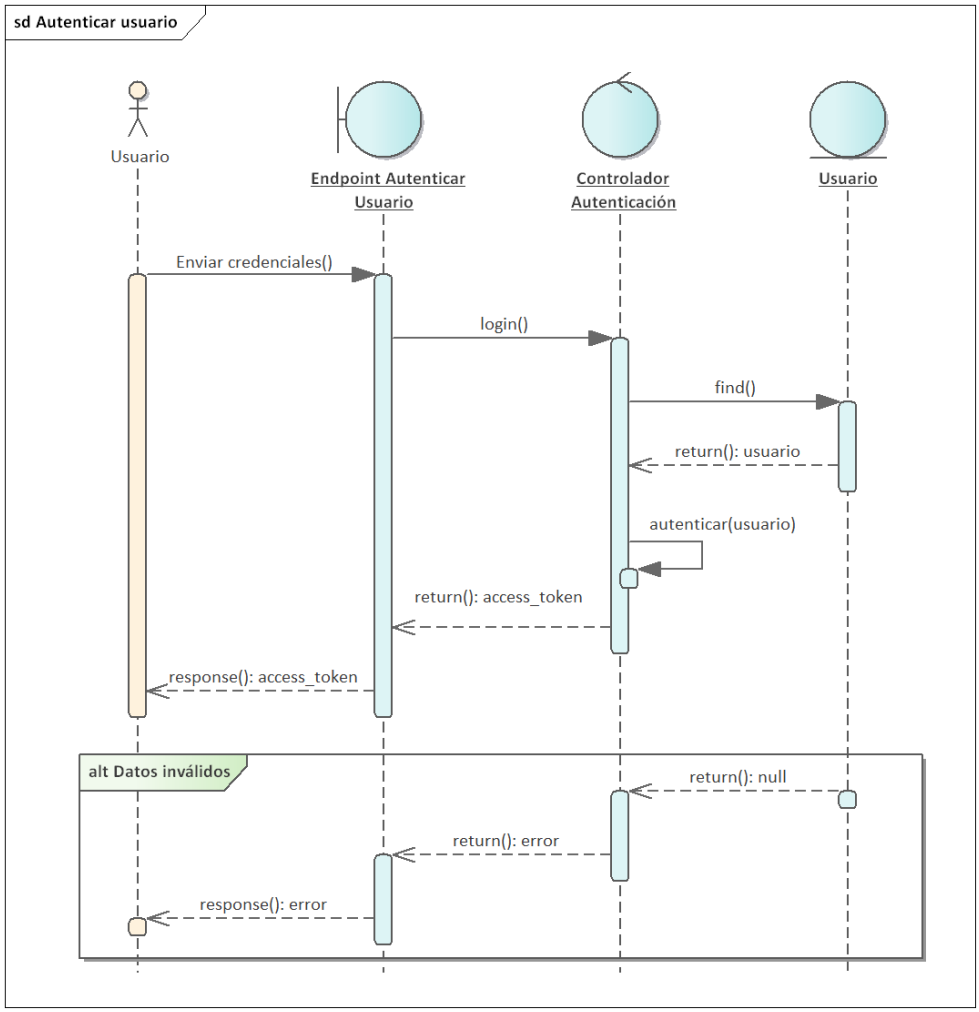


Figura N° 65: Diagrama de secuencia Autenticar Usuario
Fuente: Creado por el autor

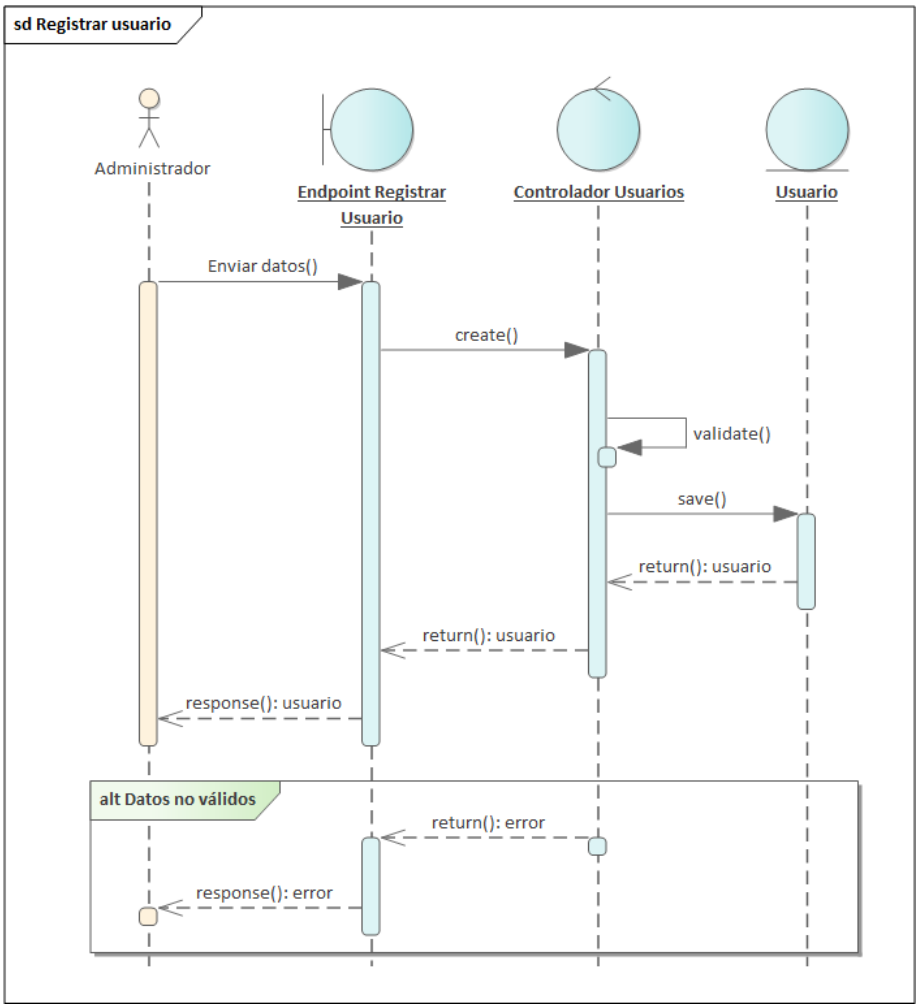


Figura N° 66: Diagrama de secuencia Registrar Usuario
Fuente: Creado por el autor

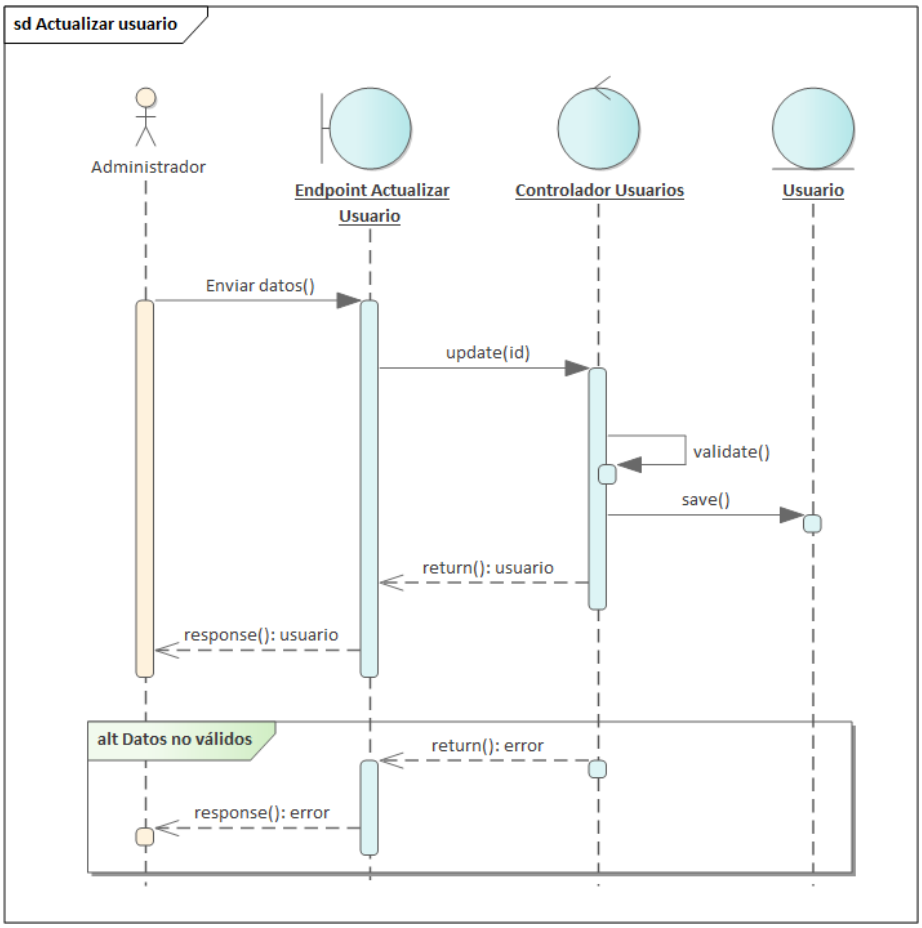


Figura N° 67: Diagrama de secuencia Actualizar Usuario
Fuente: Creado por el autor

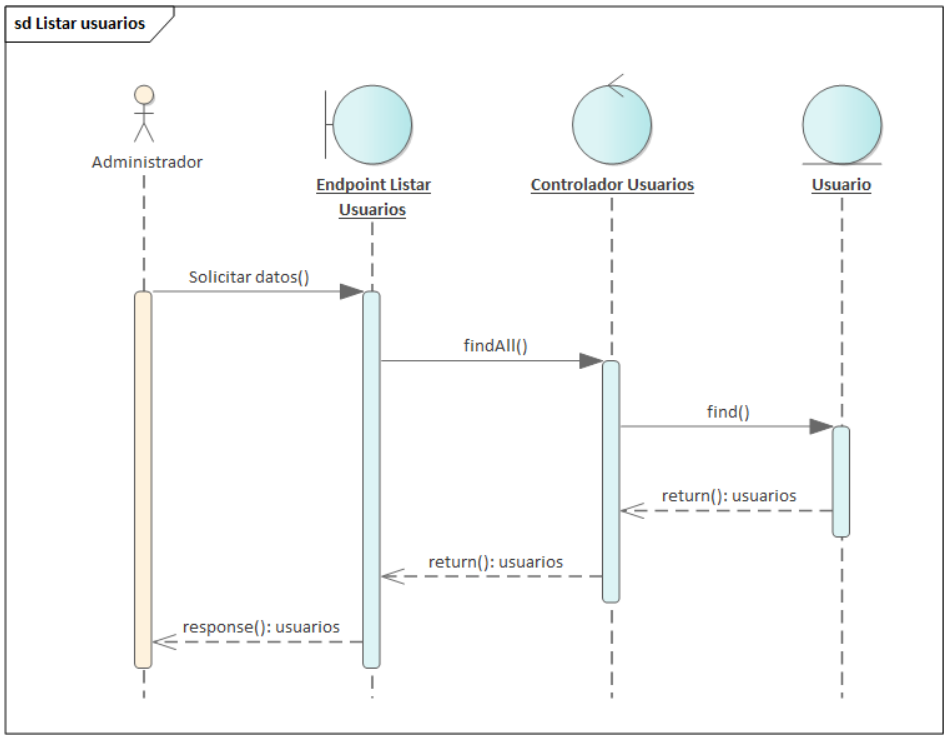


Figura N° 68: Diagrama de secuencia Listar Usuarios
Fuente: Creado por el autor

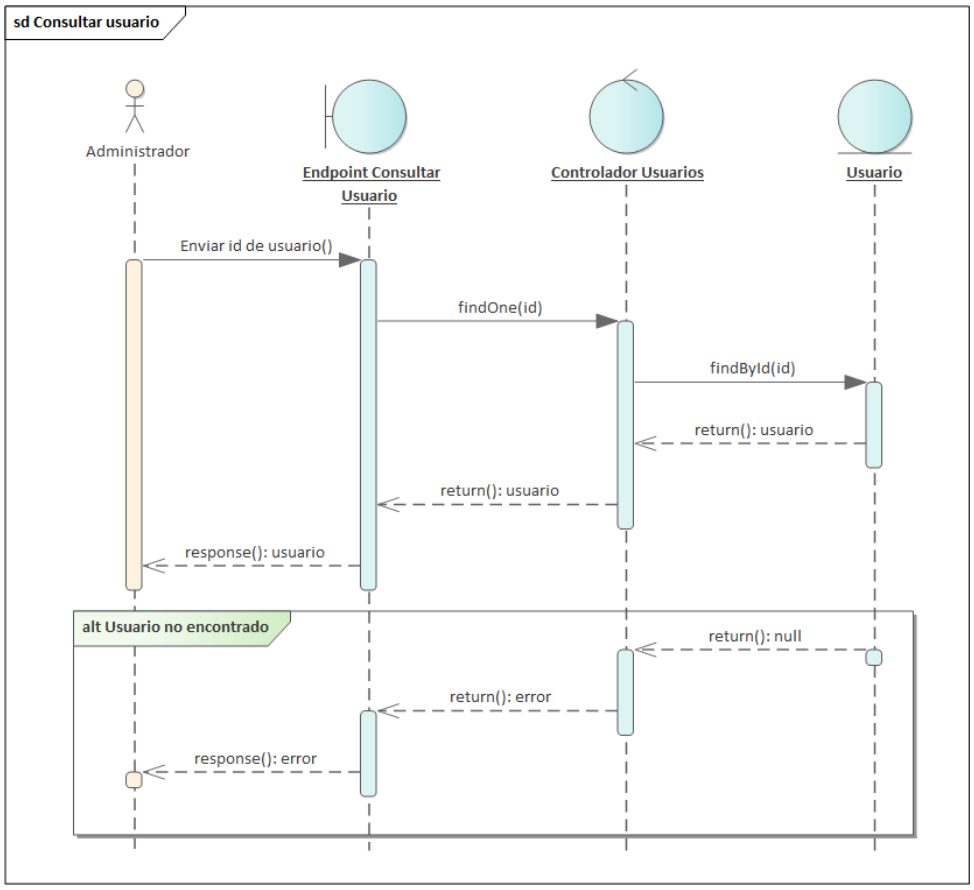


Figura N° 69: Diagrama de secuencia Consultar Usuario
Fuente: Creado por el autor

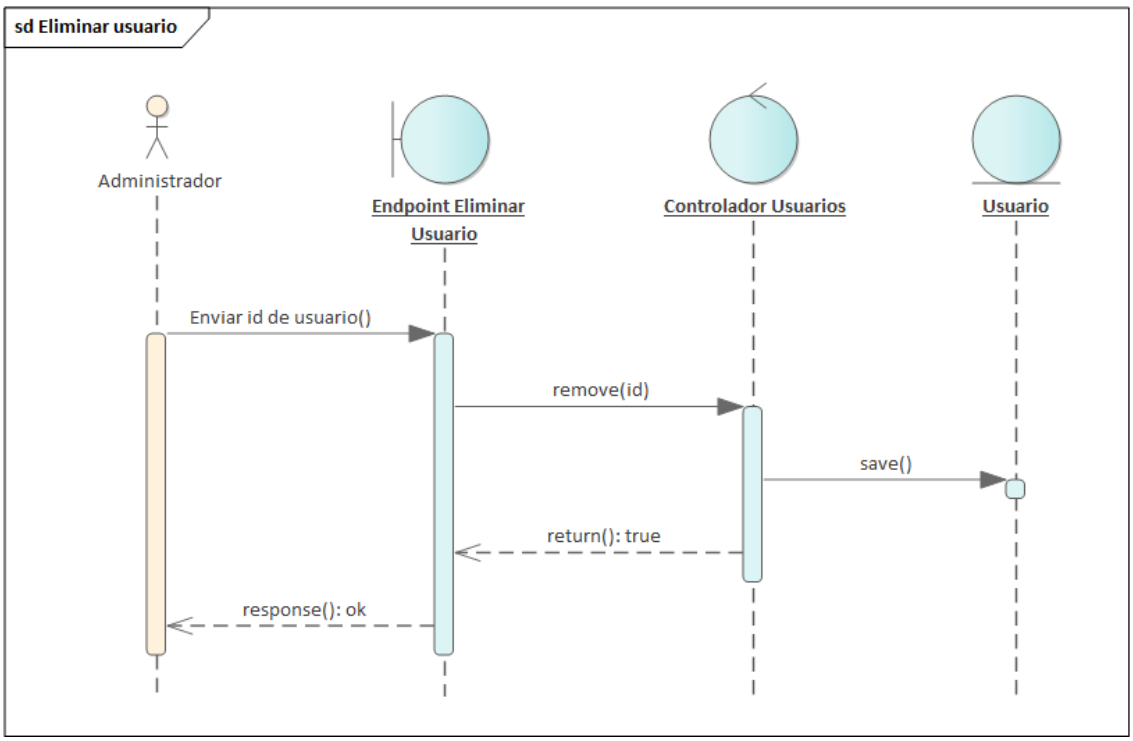


Figura N° 70: Diagrama de secuencia Eliminar Usuario
Fuente: Creado por el autor

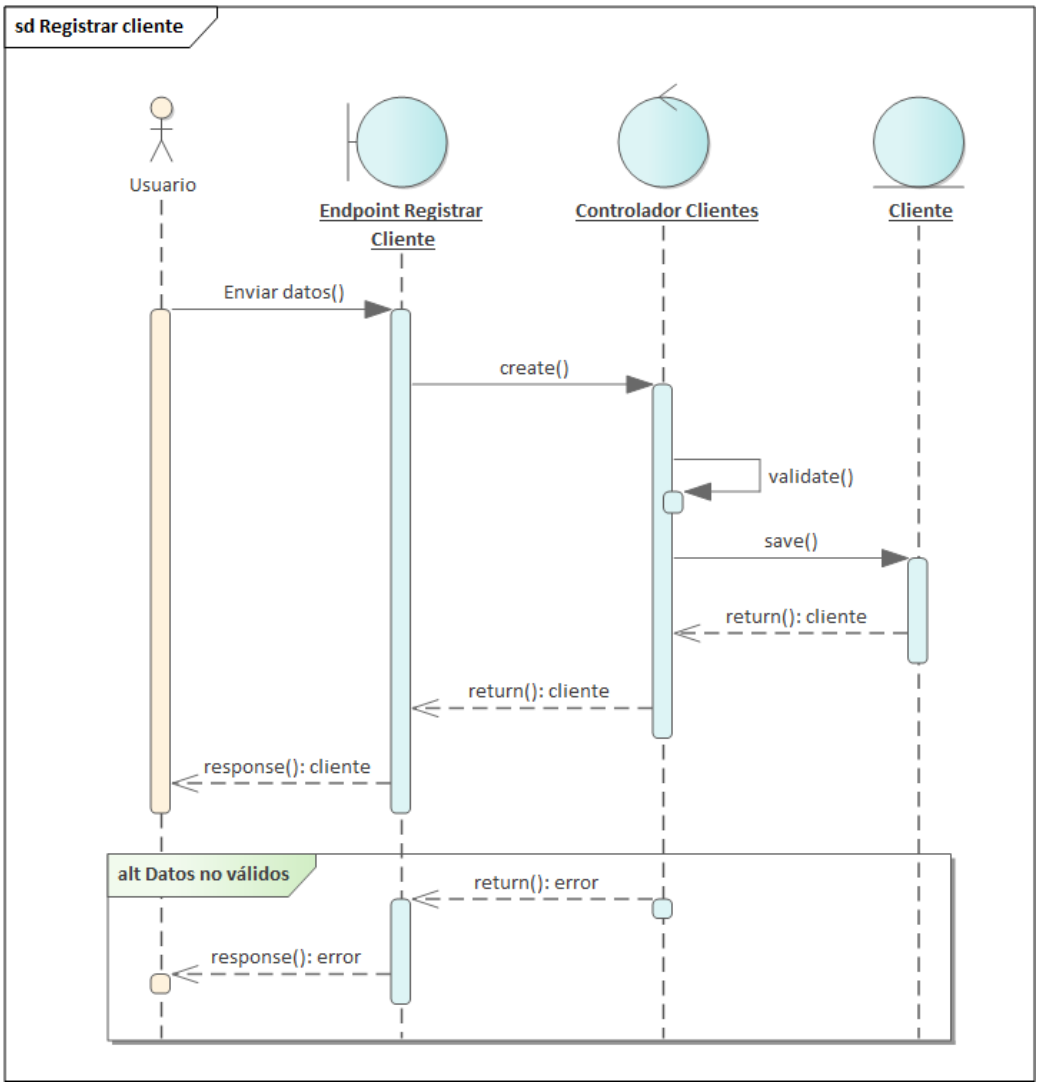


Figura N° 71: Diagrama de secuencia Registrar Cliente
Fuente: Creado por el autor

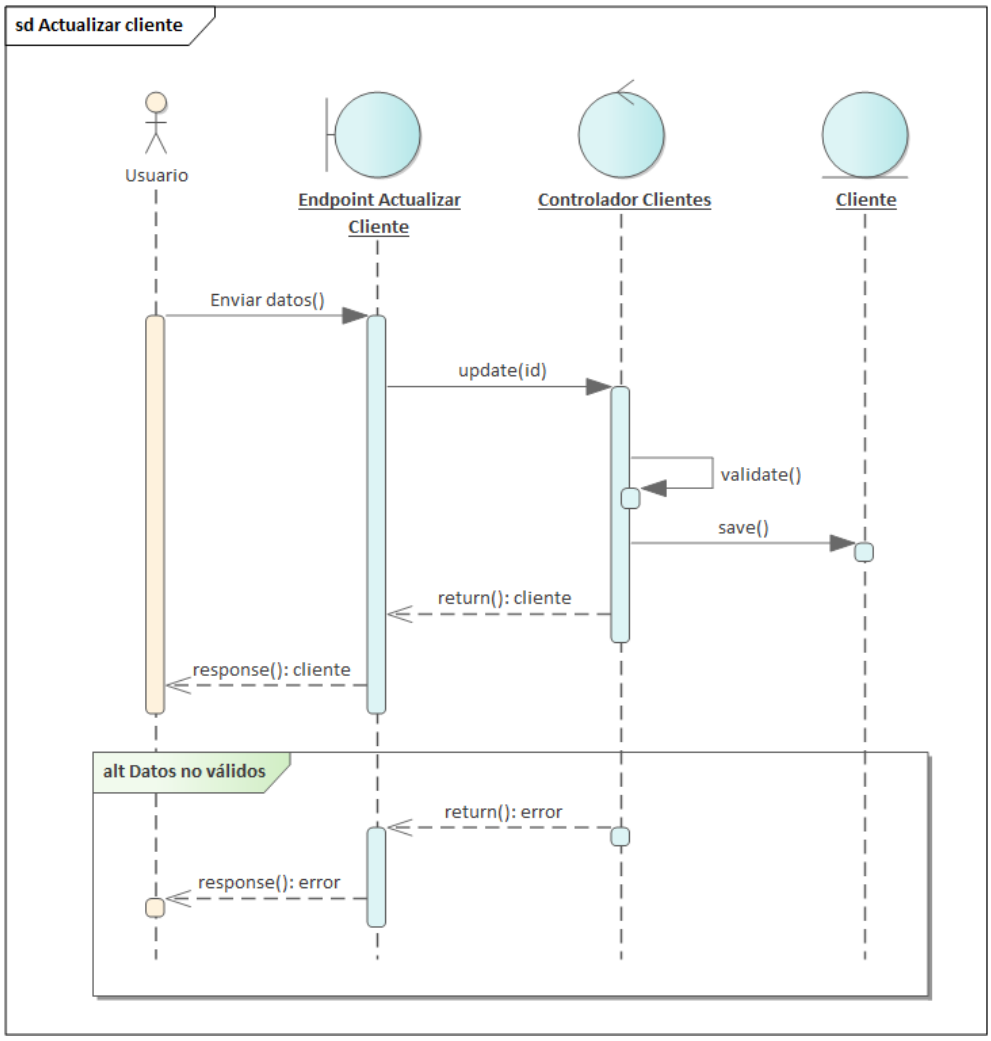


Figura N° 72: Diagrama de secuencia Actualizar Cliente
Fuente: Creado por el autor

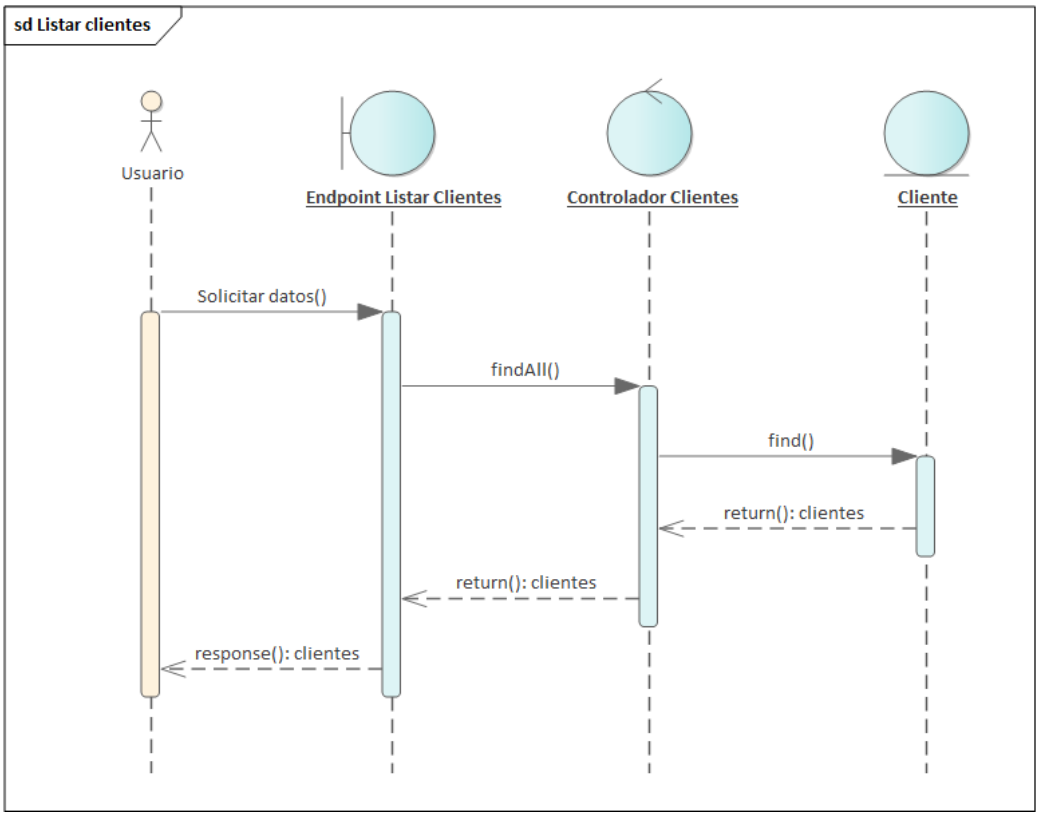


Figura N° 73: Diagrama de secuencia Listar Clientes
Fuente: Creado por el autor

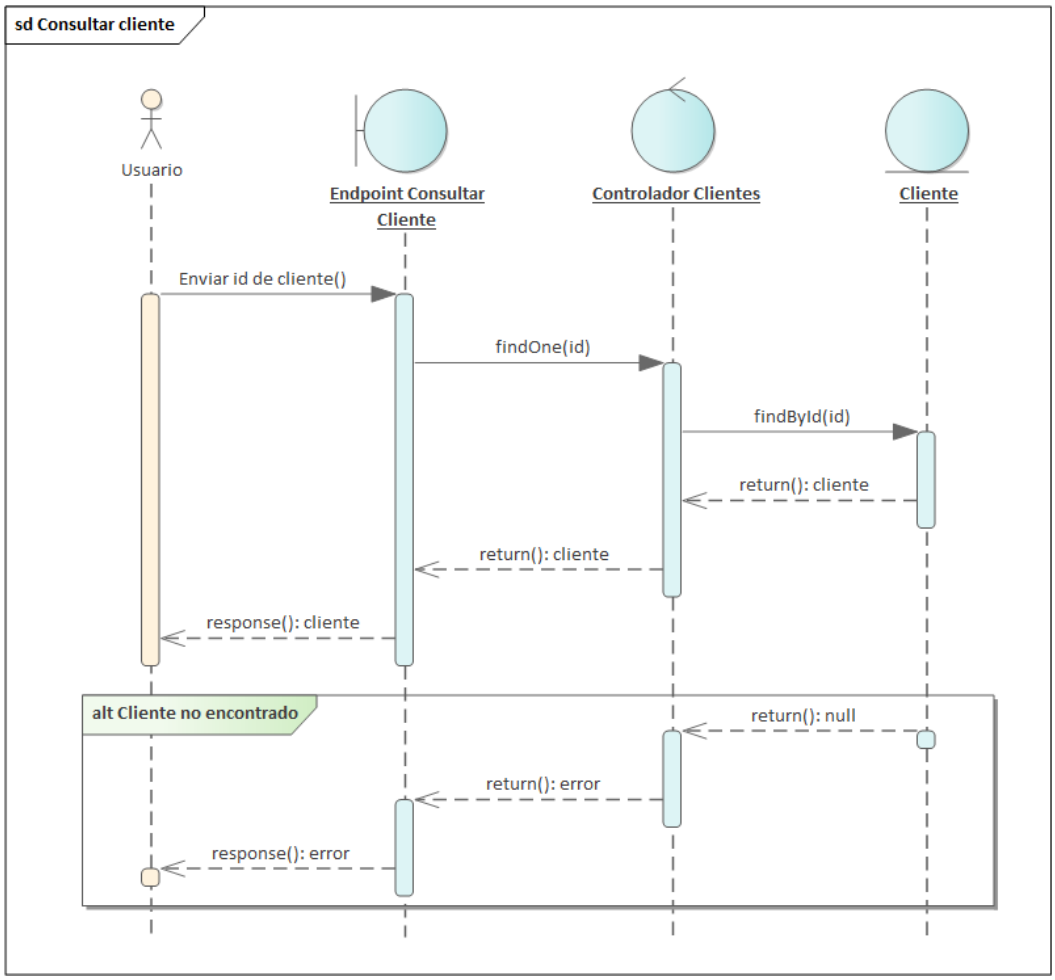


Figura N° 74: Diagrama de secuencia Consultar Cliente
Fuente: Creado por el autor

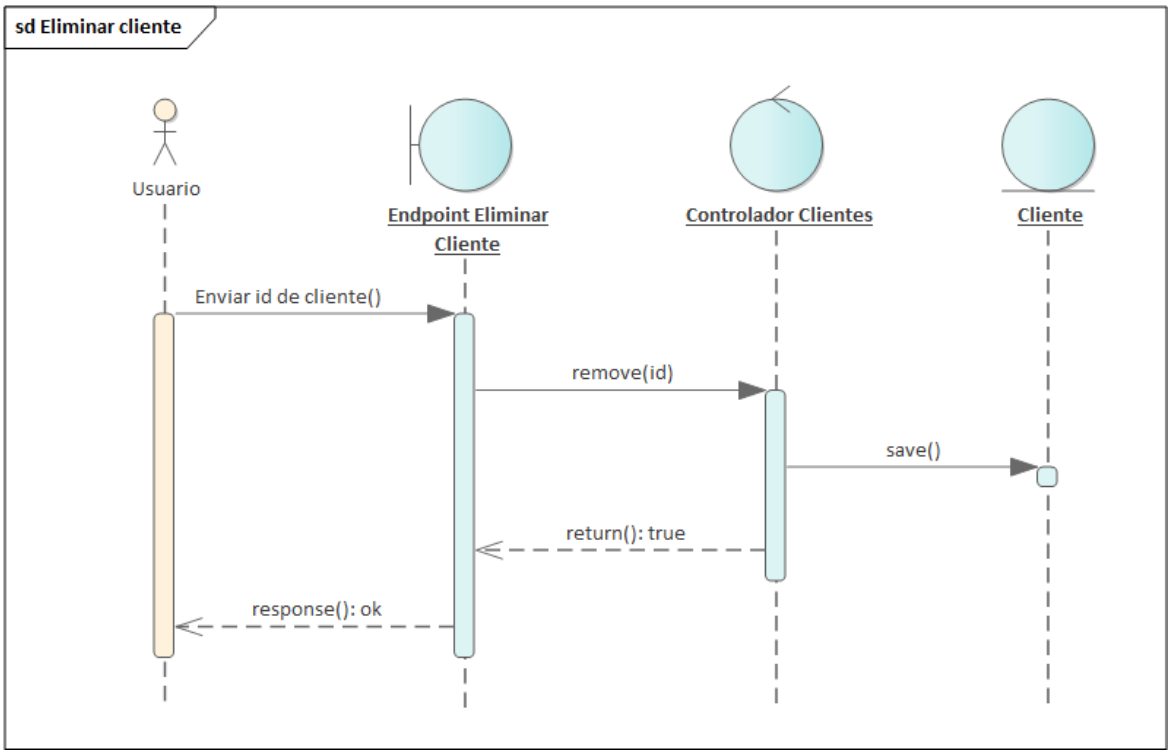


Figura N° 75: Diagrama de secuencia Eliminar Cliente
Fuente: Creado por el autor

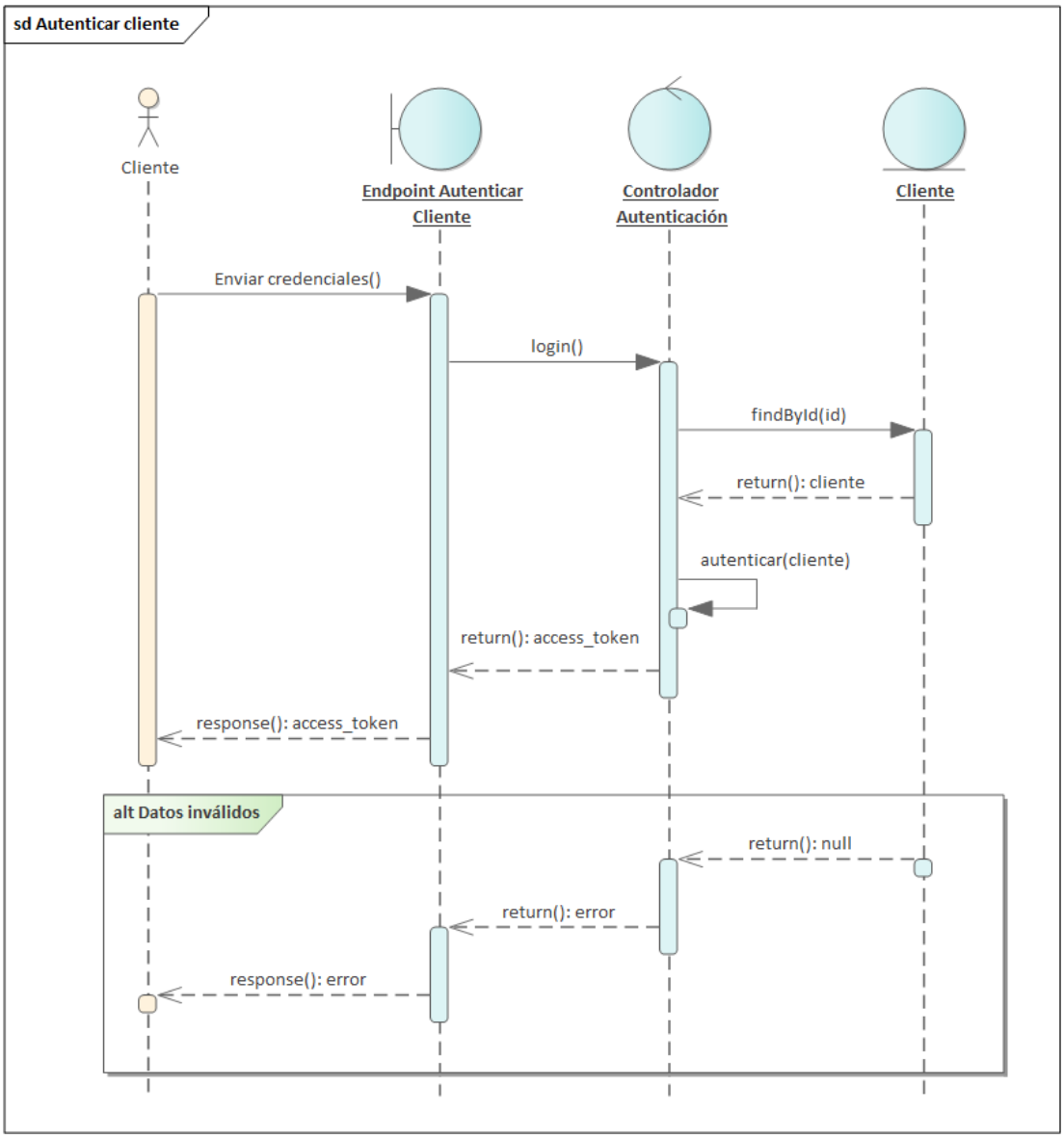


Figura N° 76: Diagrama de secuencia Autenticar Cliente
Fuente: Creado por el autor

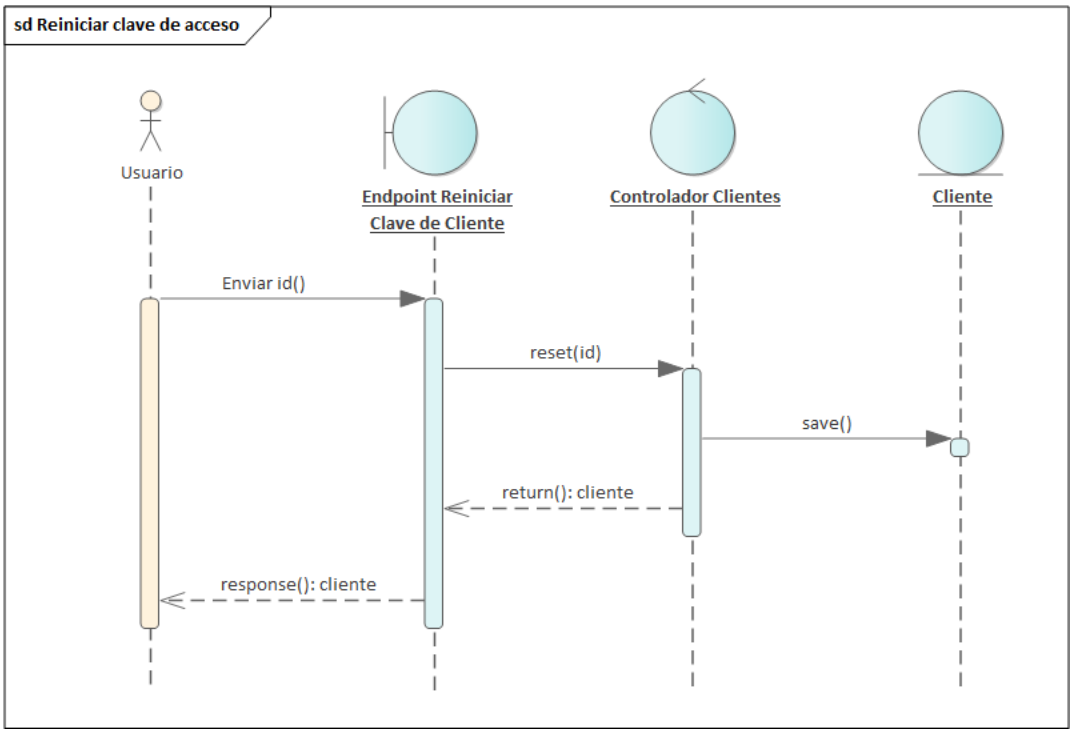


Figura N° 77: Diagrama de secuencia Reiniciar Clave de Acceso de Cliente
Fuente: Creado por el autor

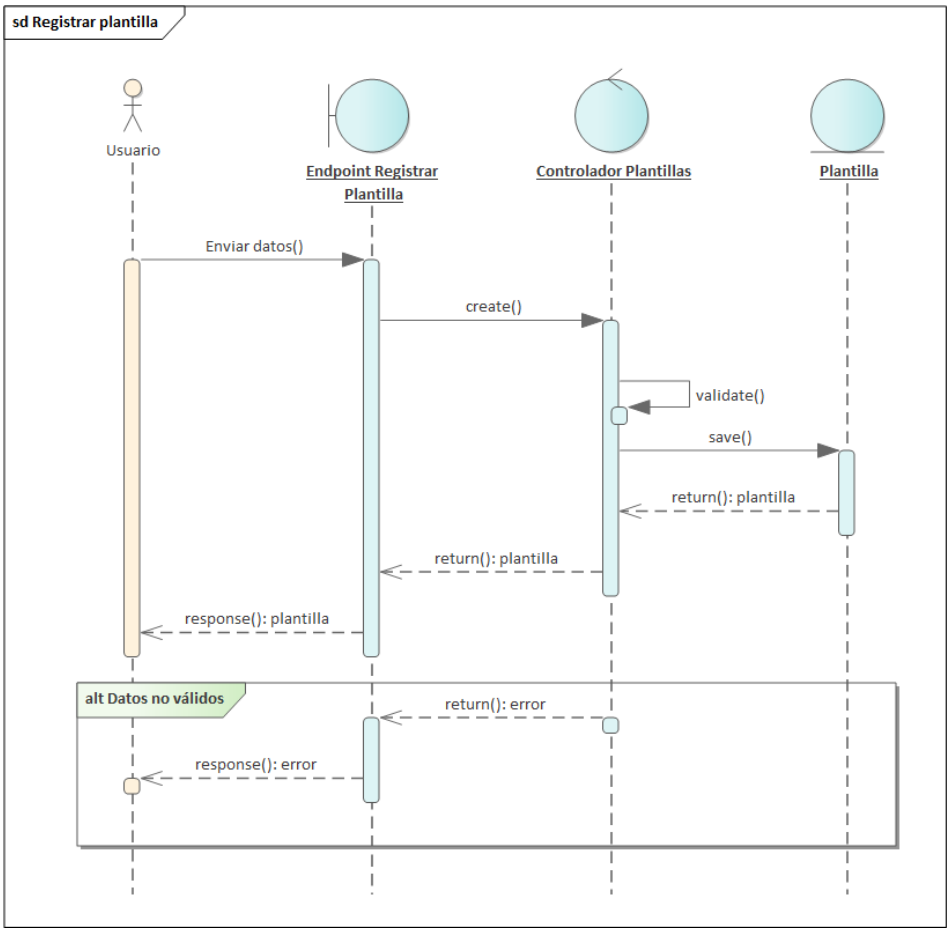


Figura N° 78: Diagrama de secuencia Registrar Plantilla
Fuente: Creado por el autor

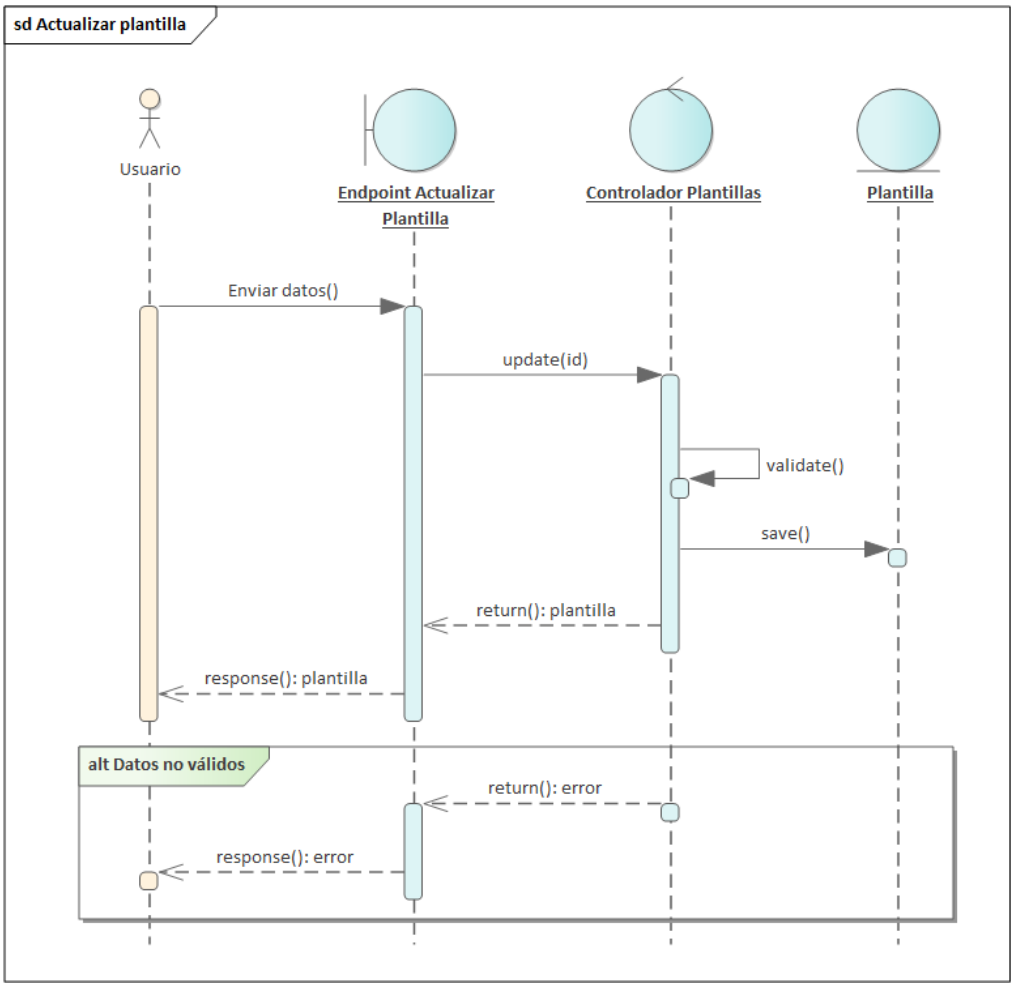


Figura N° 79: Diagrama de secuencia Actualizar Plantilla
Fuente: Creado por el autor

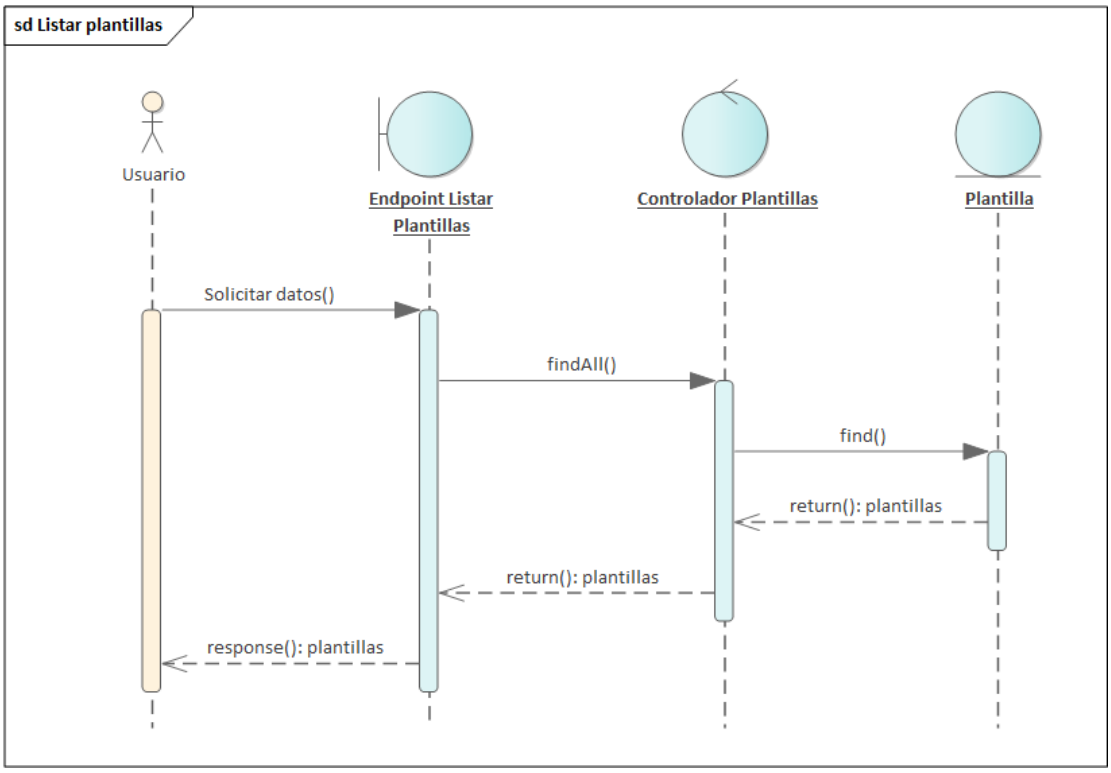


Figura N° 80: Diagrama de secuencia Listar Plantillas
Fuente: Creado por el autor

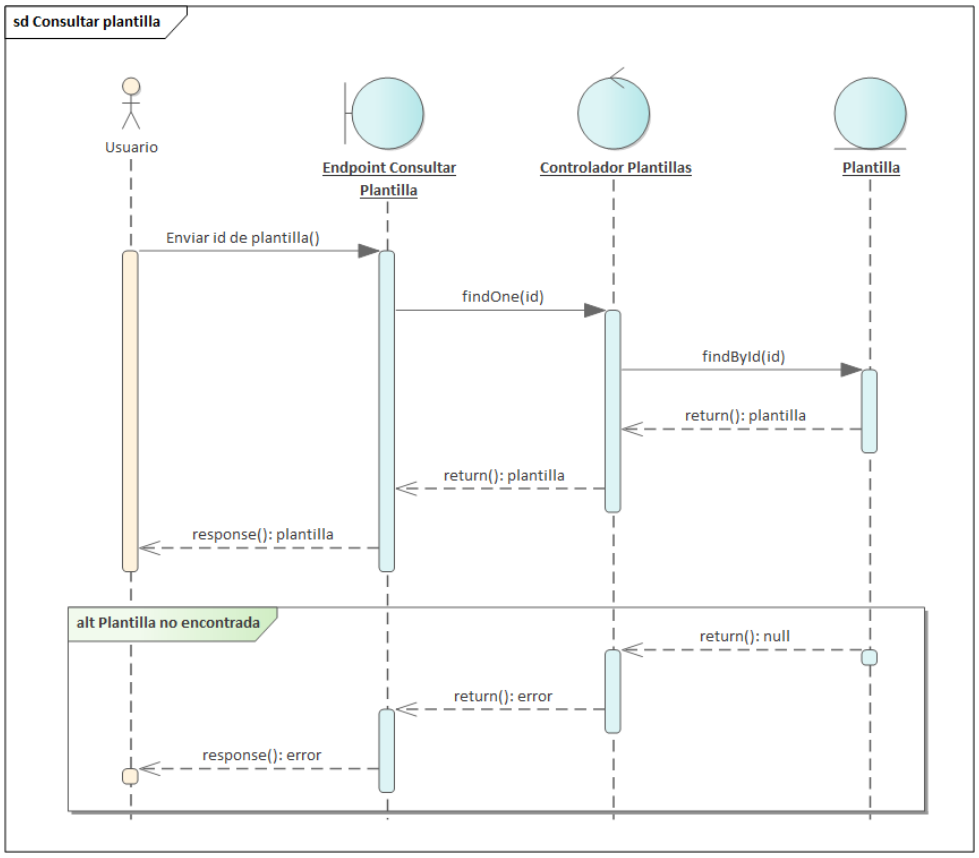


Figura N° 81: Diagrama de secuencia Consultar Plantilla
Fuente: Creado por el autor

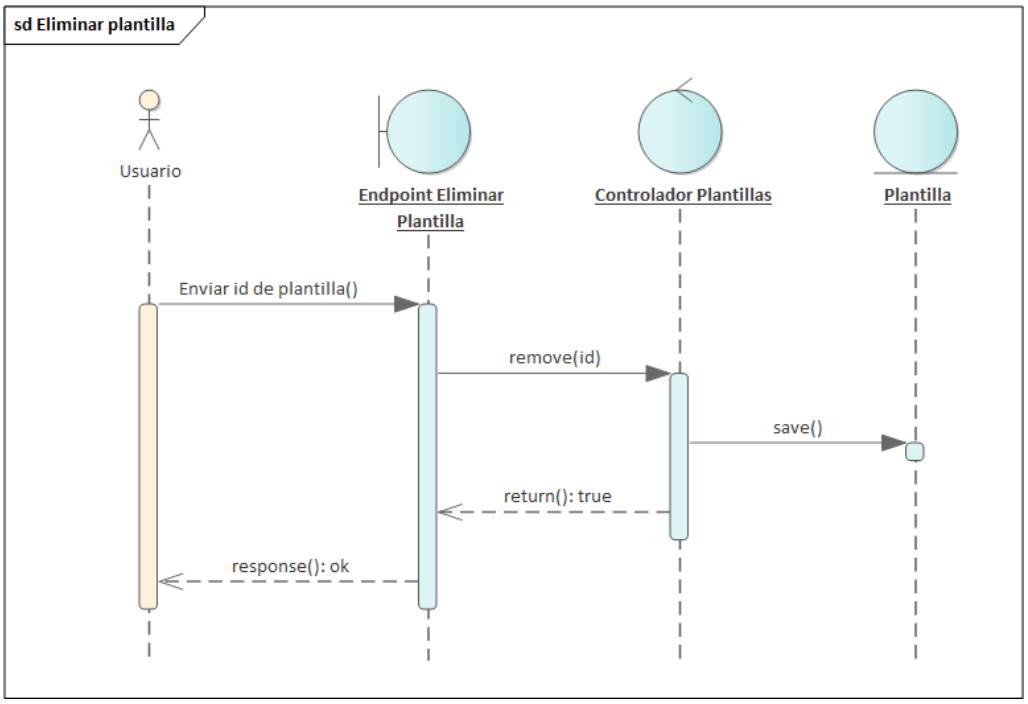


Figura N° 82: Diagrama de secuencia Eliminar Plantilla
Fuente: Creado por el autor

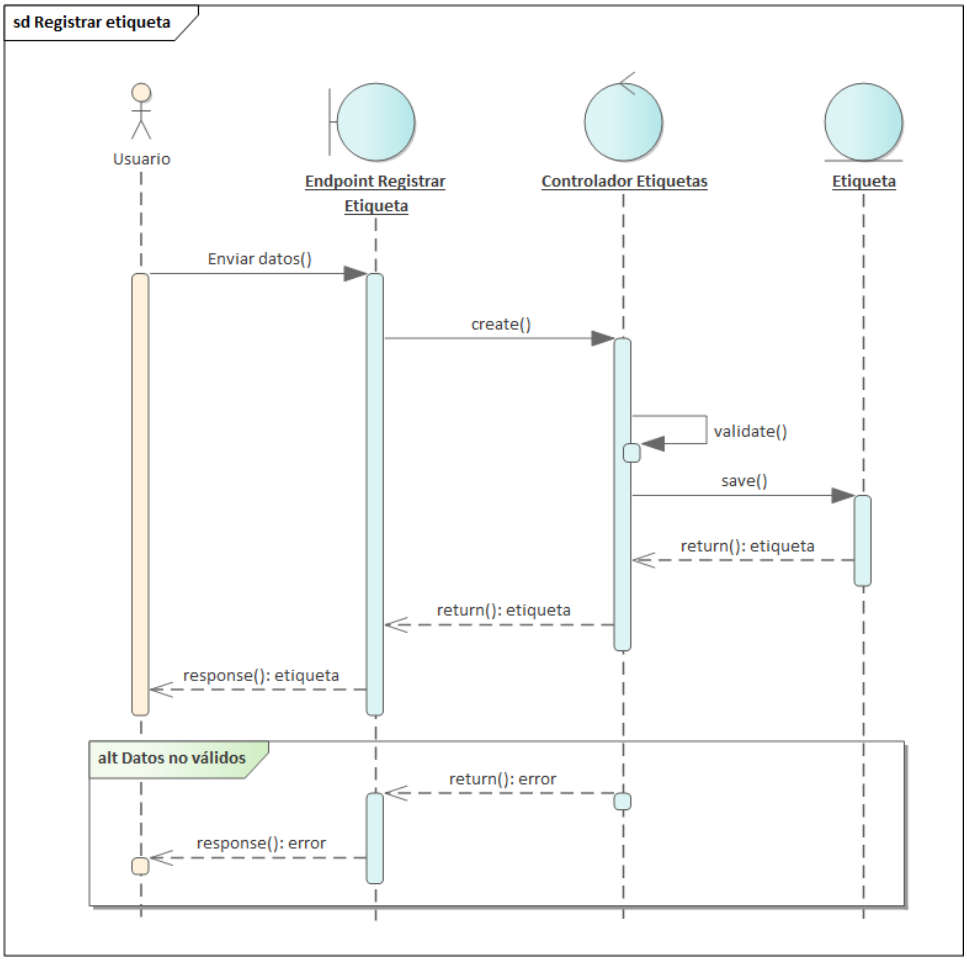


Figura N° 83: Diagrama de secuencia Registrar Etiqueta
Fuente: Creado por el autor

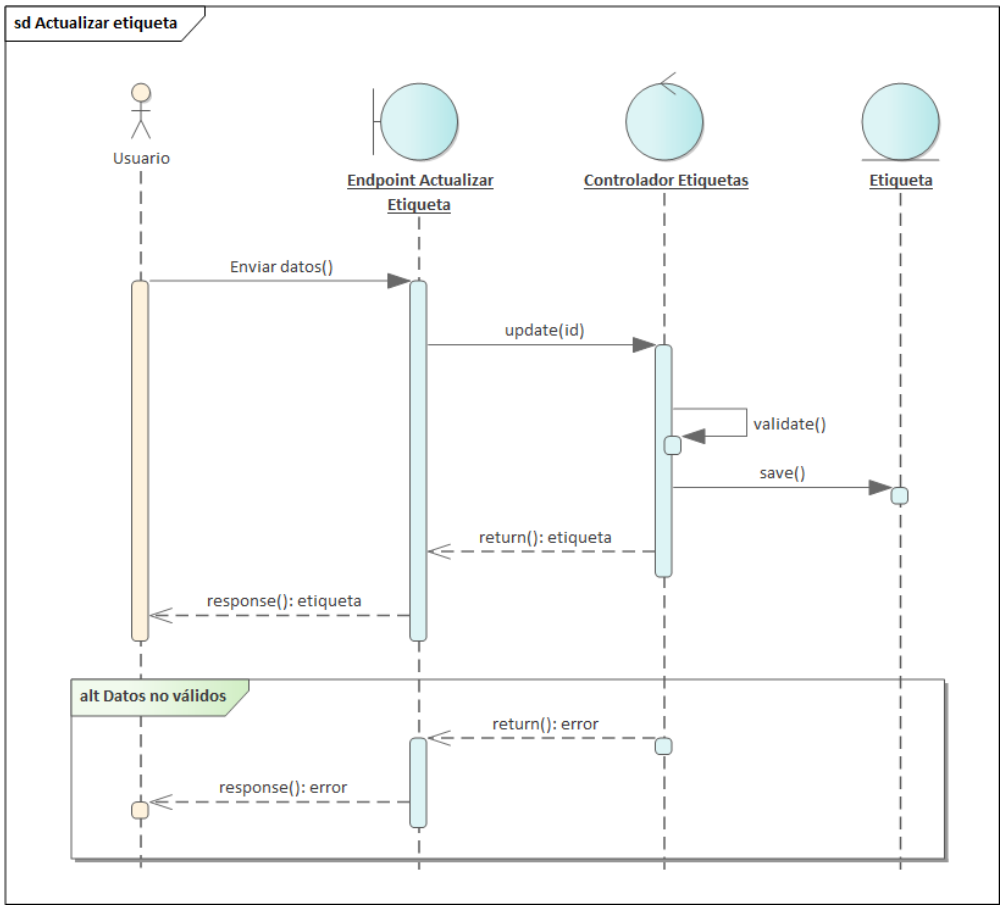


Figura N° 84: Diagrama de secuencia Actualizar Etiqueta
Fuente: Creado por el autor

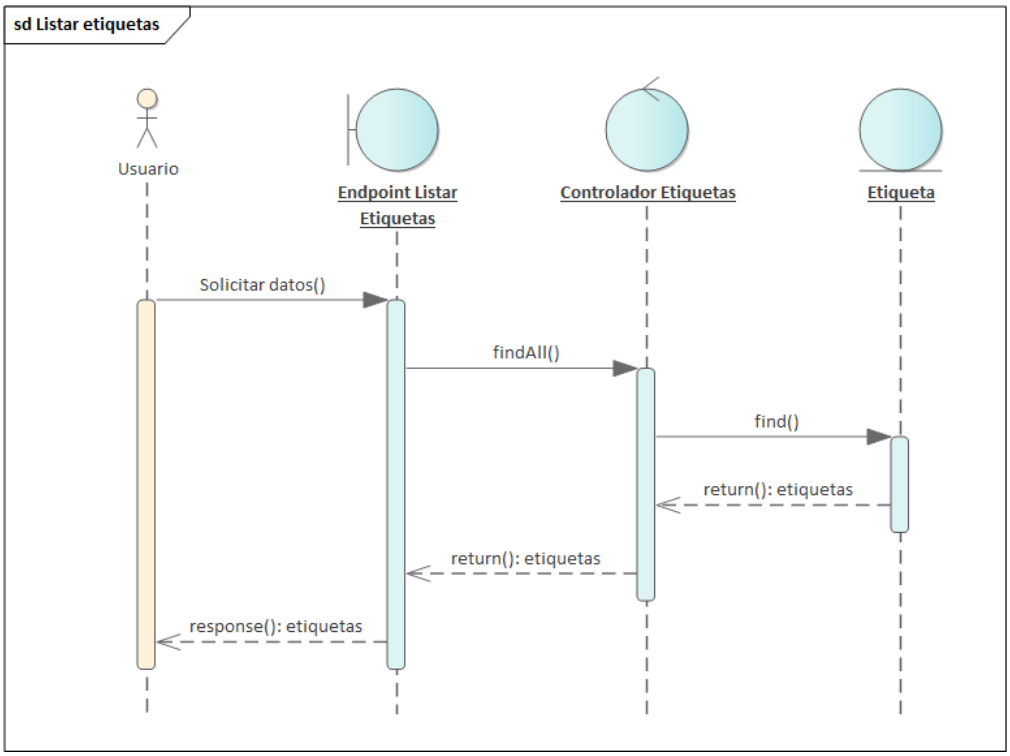


Figura N° 85: Diagrama de secuencia Listar Etiquetas
Fuente: Creado por el autor

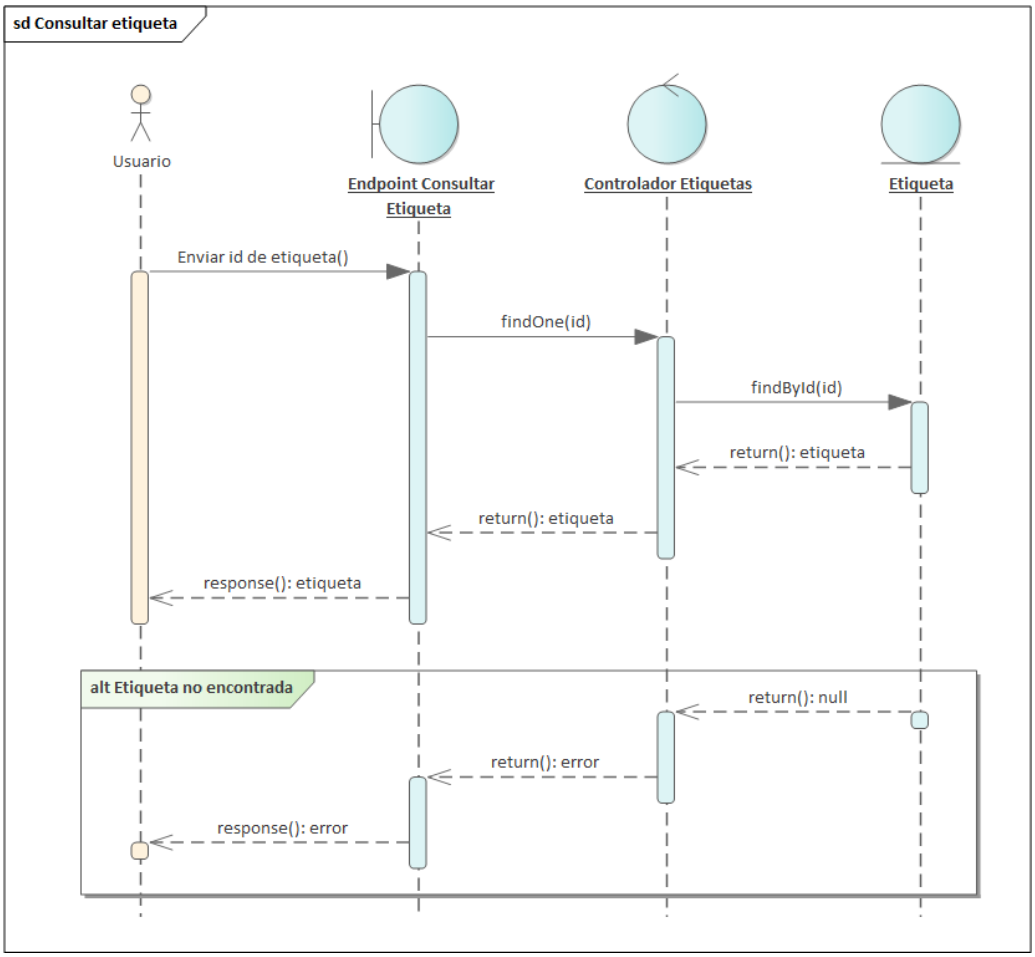


Figura N° 86: Diagrama de secuencia Consultar Etiqueta
Fuente: Creado por el autor

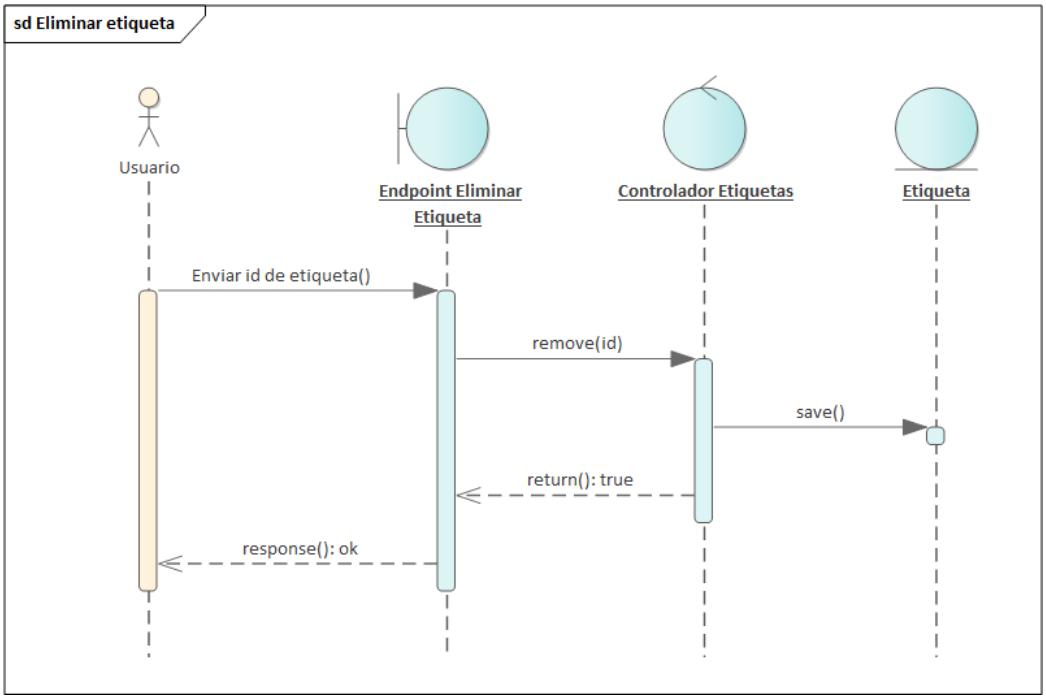


Figura N° 87: Diagrama de secuencia Eliminar Etiqueta
Fuente: Creado por el autor

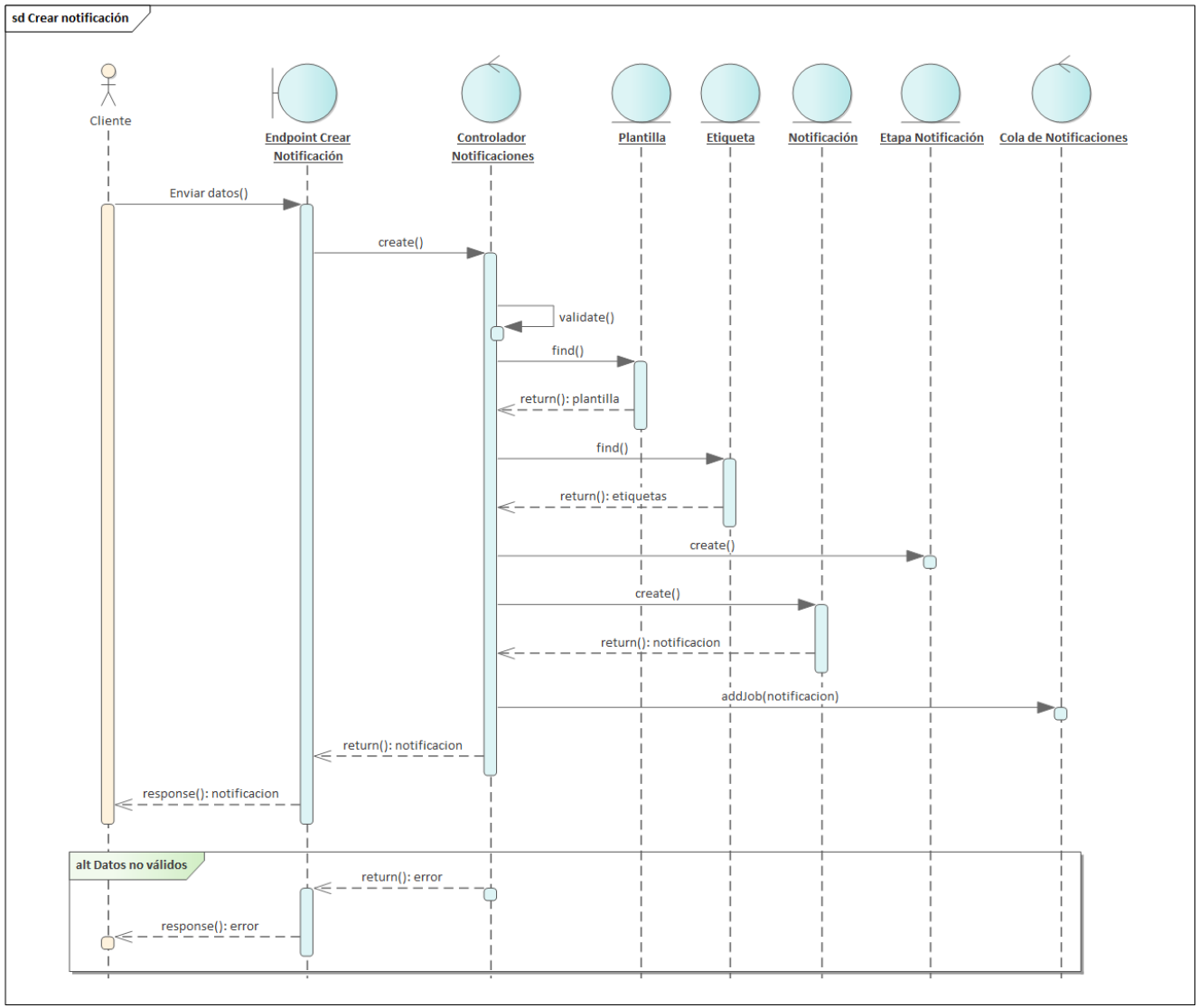


Figura N° 88: Diagrama de secuencia Crear Notificación
Fuente: Creado por el autor

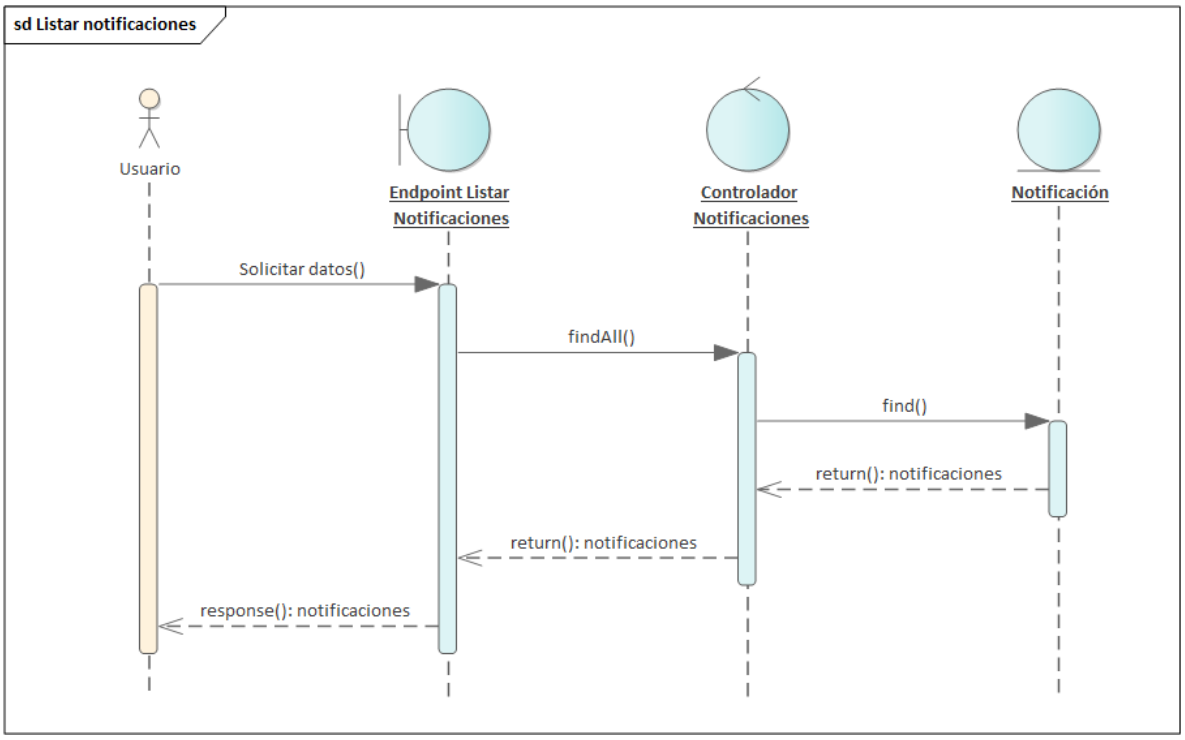


Figura N° 89: Diagrama de secuencia Listar Notificaciones
Fuente: Creado por el autor

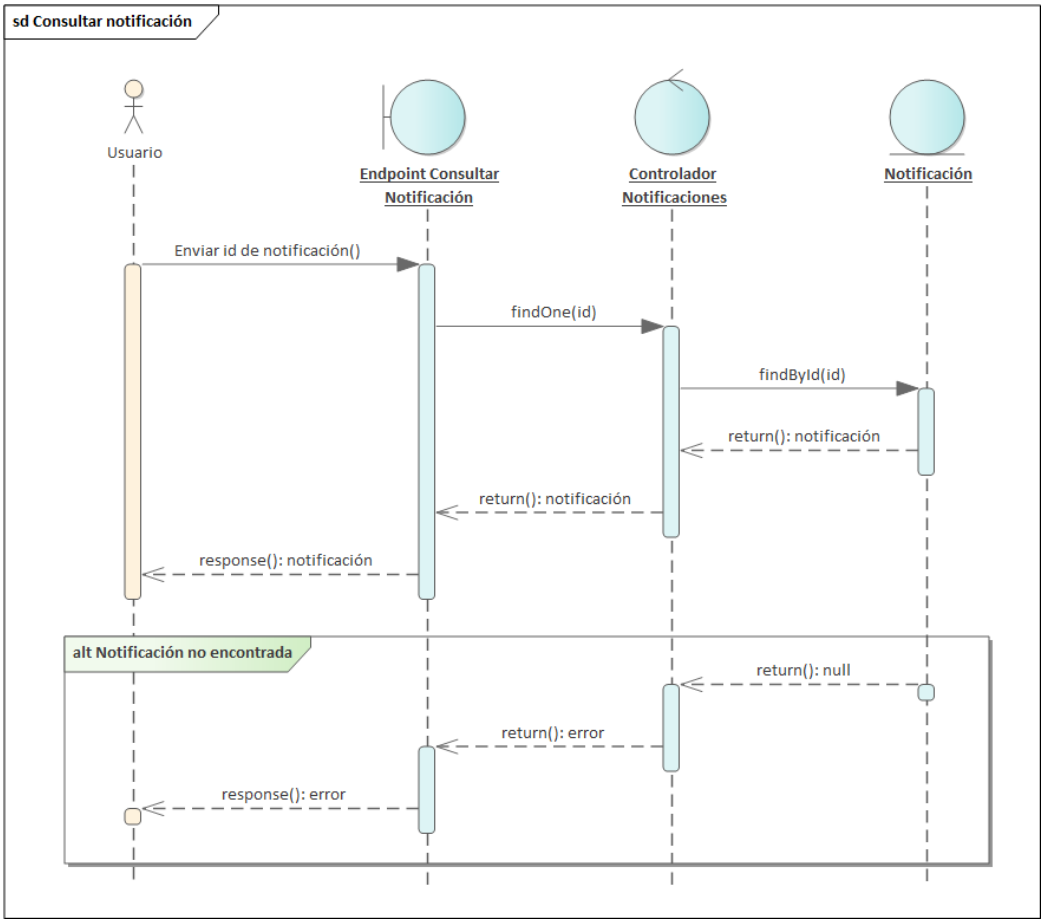


Figura N° 90: Diagrama de secuencia Consultar Notificación
Fuente: Creado por el autor

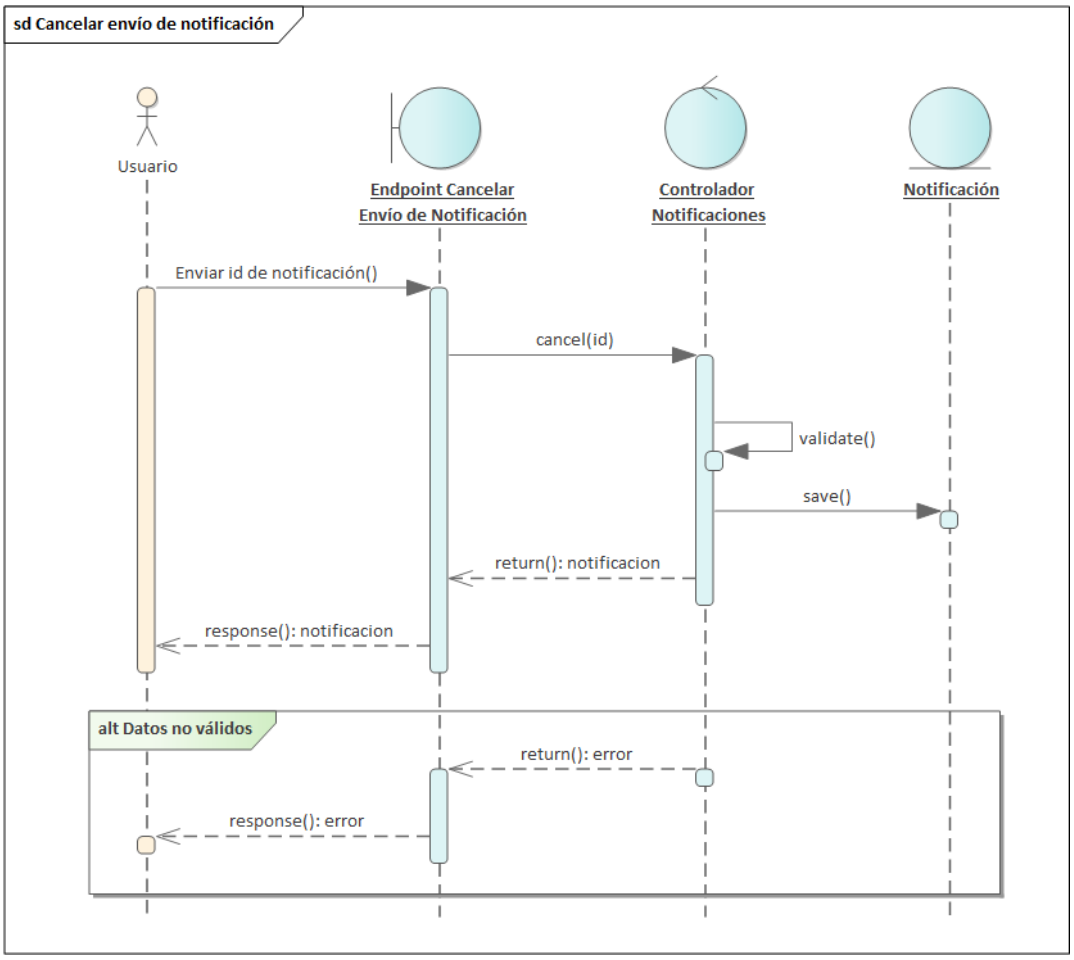


Figura N° 91: Diagrama de secuencia Cancelar Envío de Notificación
Fuente: Creado por el autor

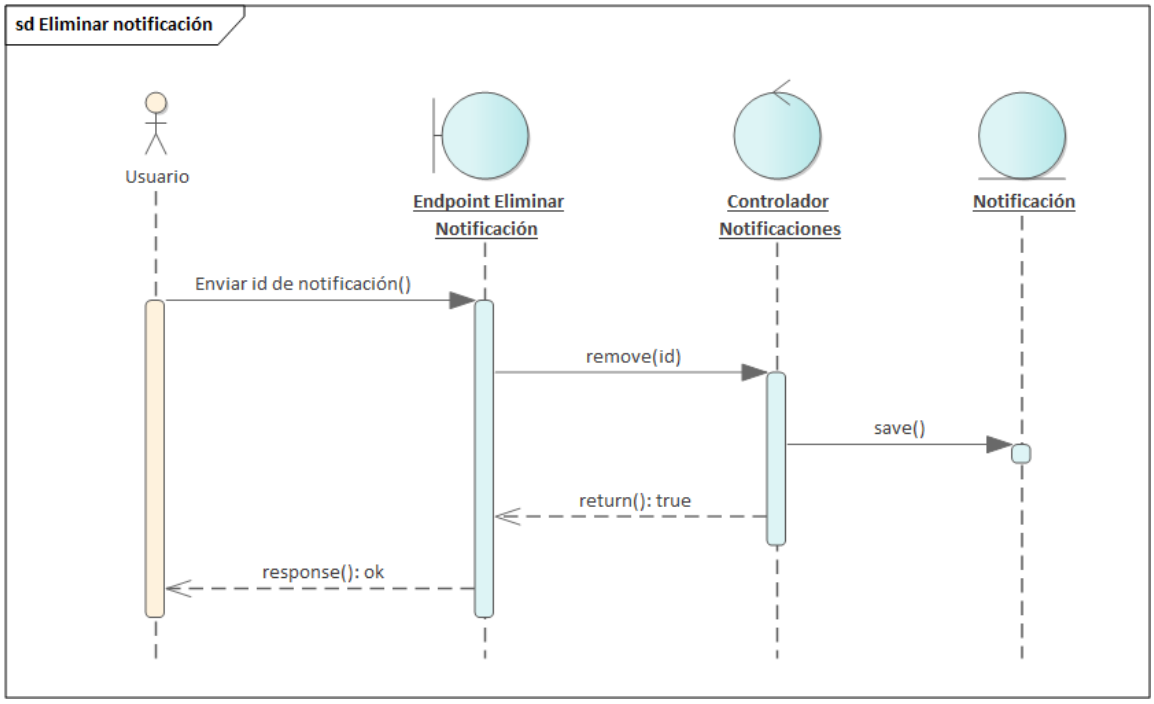


Figura N° 92: Diagrama de secuencia Eliminar Notificación
Fuente: Creado por el autor

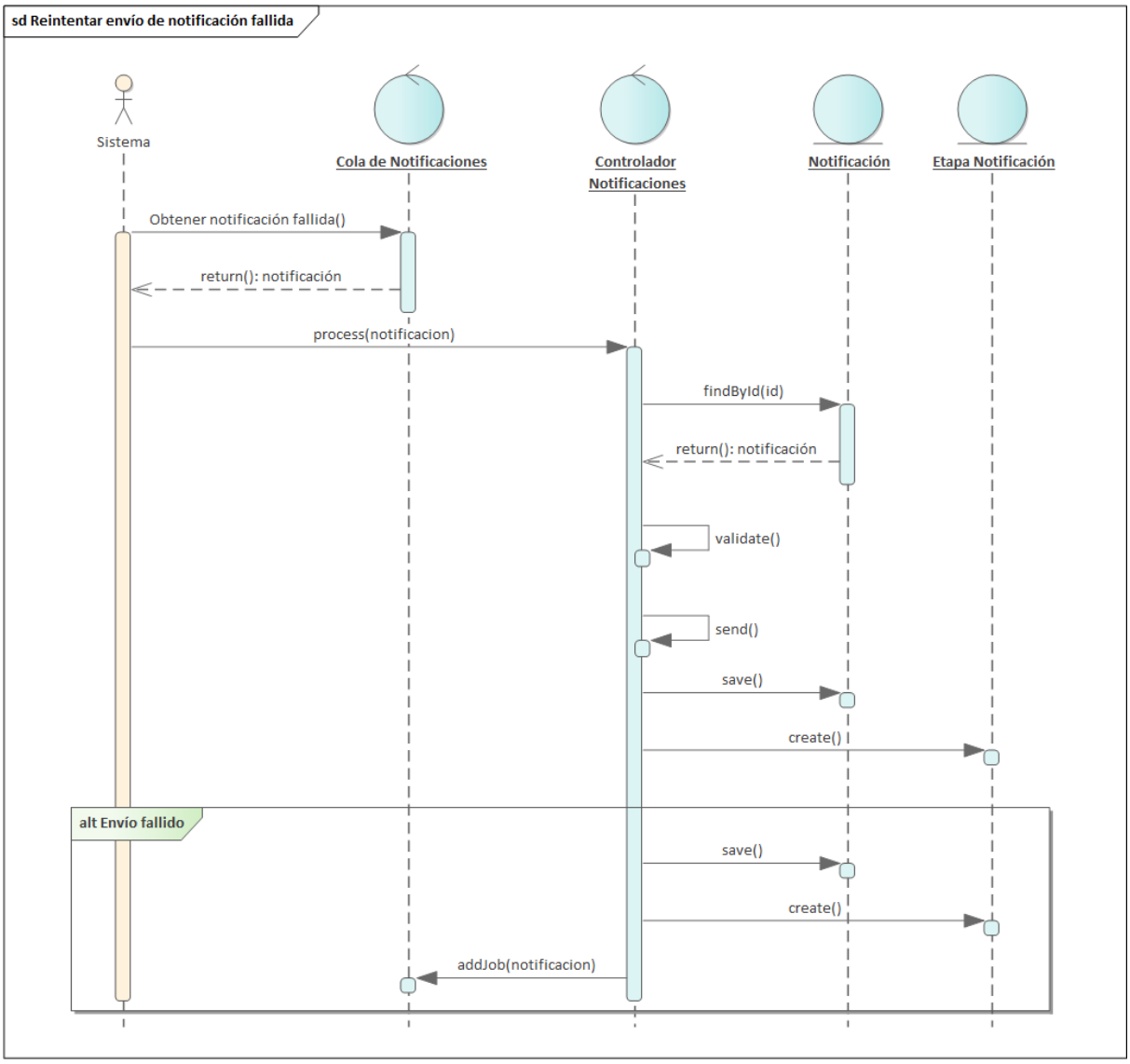


Figura N° 93: Diagrama de secuencia Reintentar Envío de Notificación Fallida
Fuente: Creado por el autor

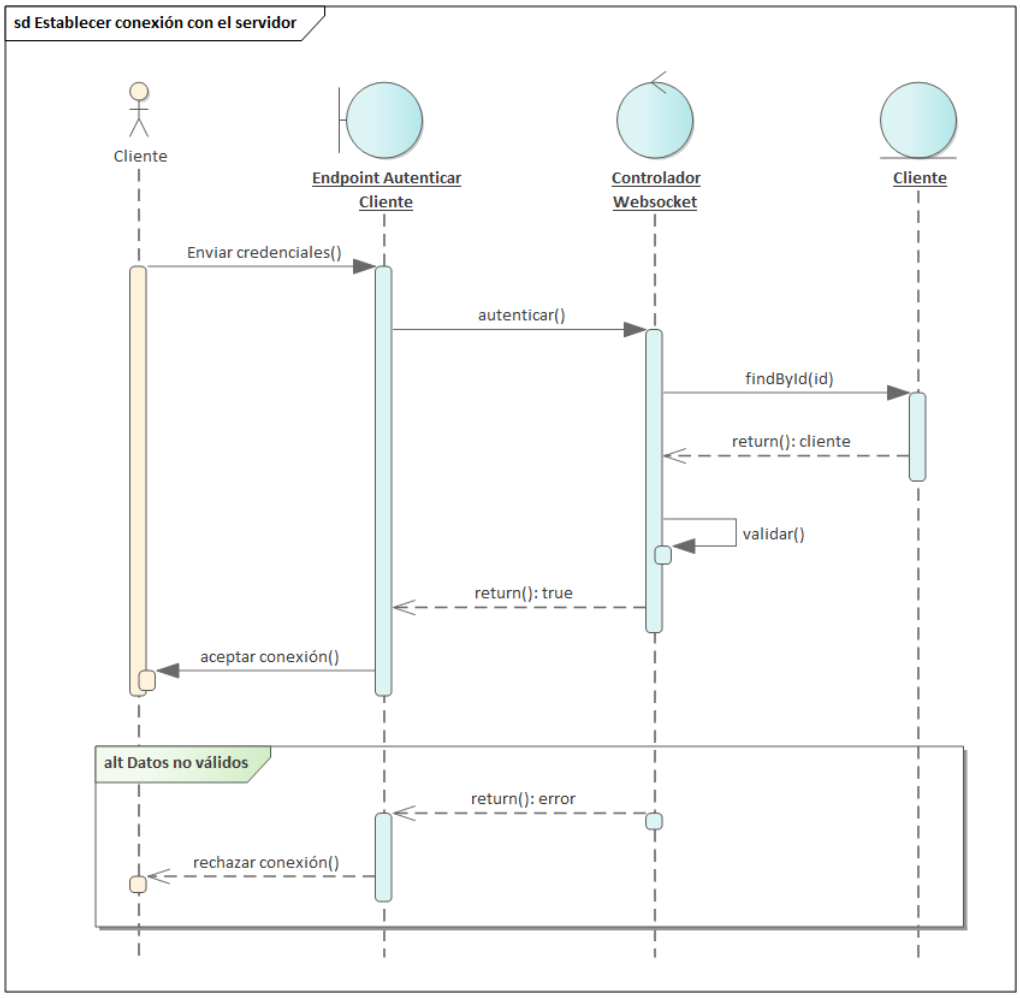


Figura N° 94: Diagrama de secuencia Establecer Conexión con el Servidor
Fuente: Creado por el autor

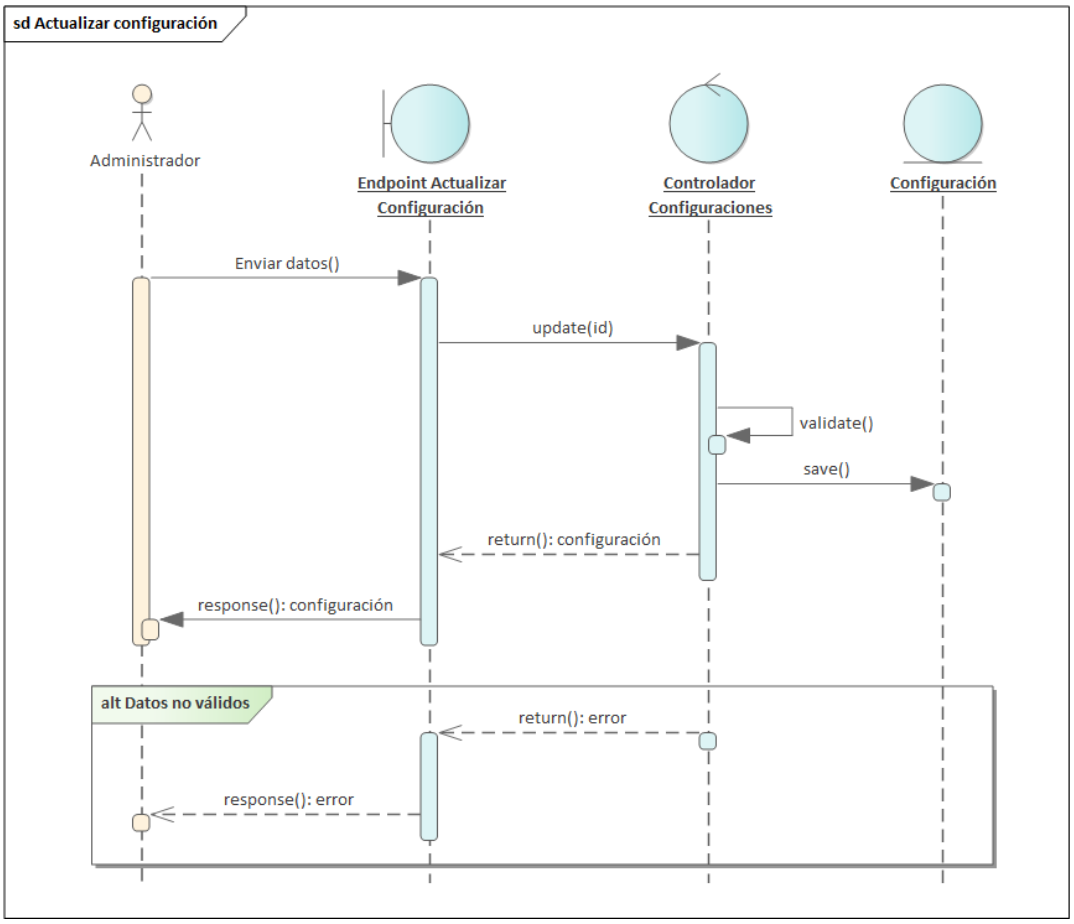


Figura N° 95: Diagrama de secuencia Actualizar Configuración
Fuente: Creado por el autor

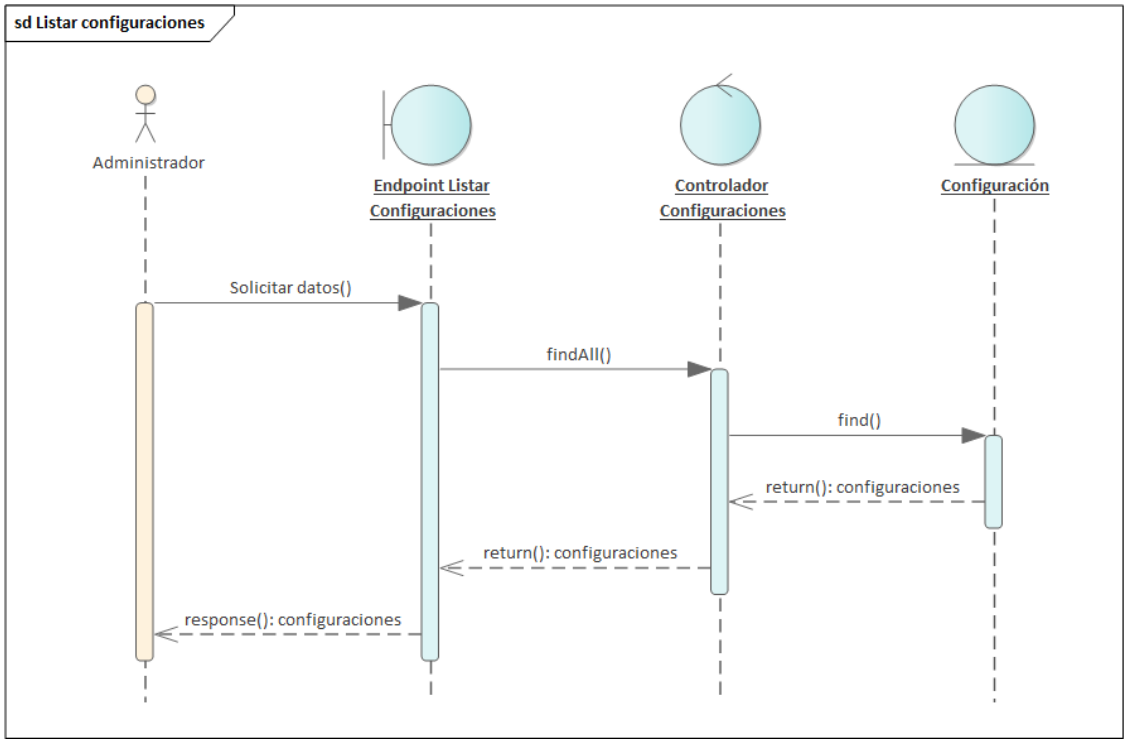


Figura N° 96: Diagrama de secuencia Listar Configuraciones
Fuente: Creado por el autor

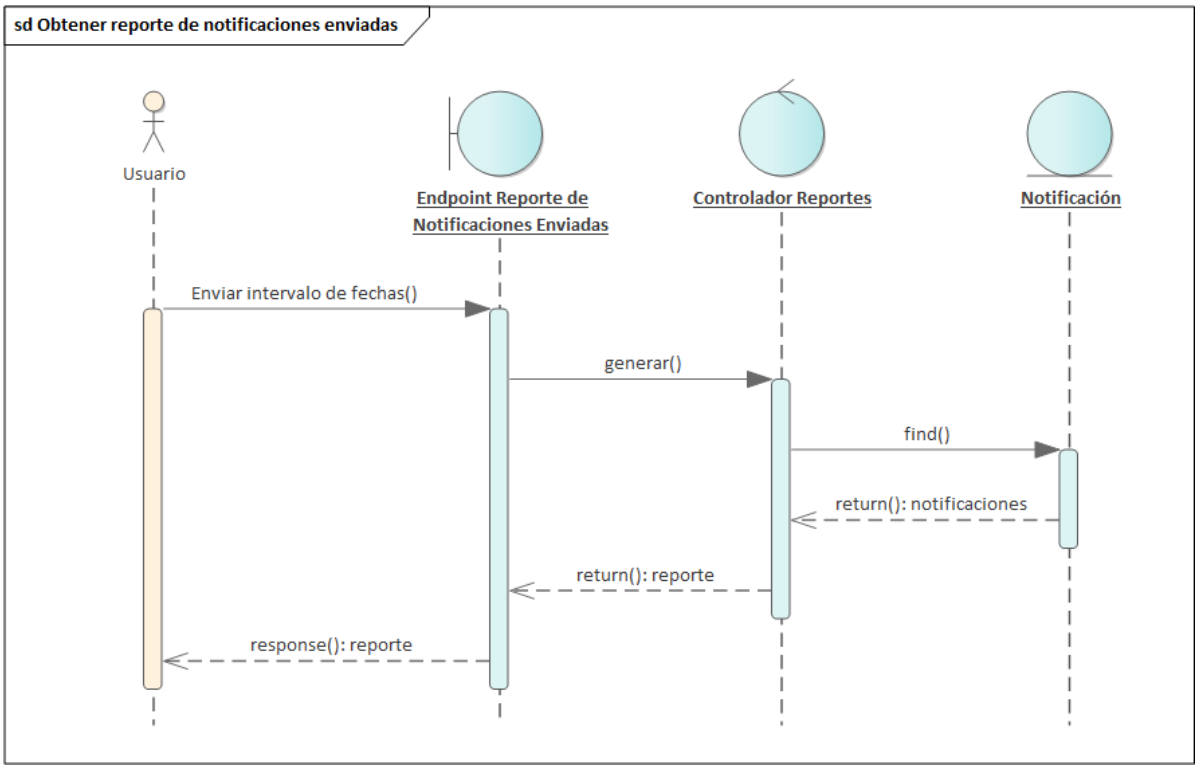


Figura N° 97: Diagrama de secuencia Obtener Reporte de Notificaciones Enviadas
Fuente: Creado por el autor

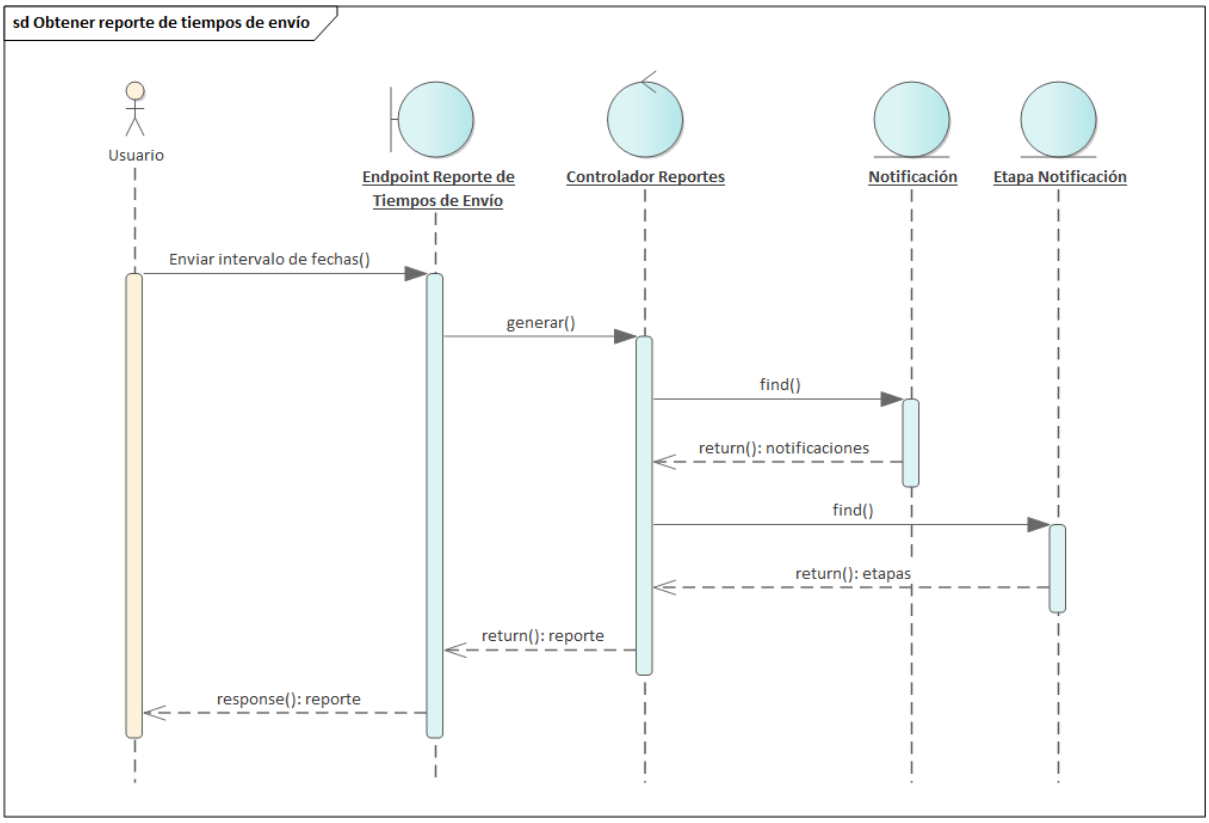


Figura N° 98: Diagrama de secuencia Obtener Reporte de Tiempos de Envío
Fuente: Creado por el autor

6.5. DIAGRAMA DE CLASES

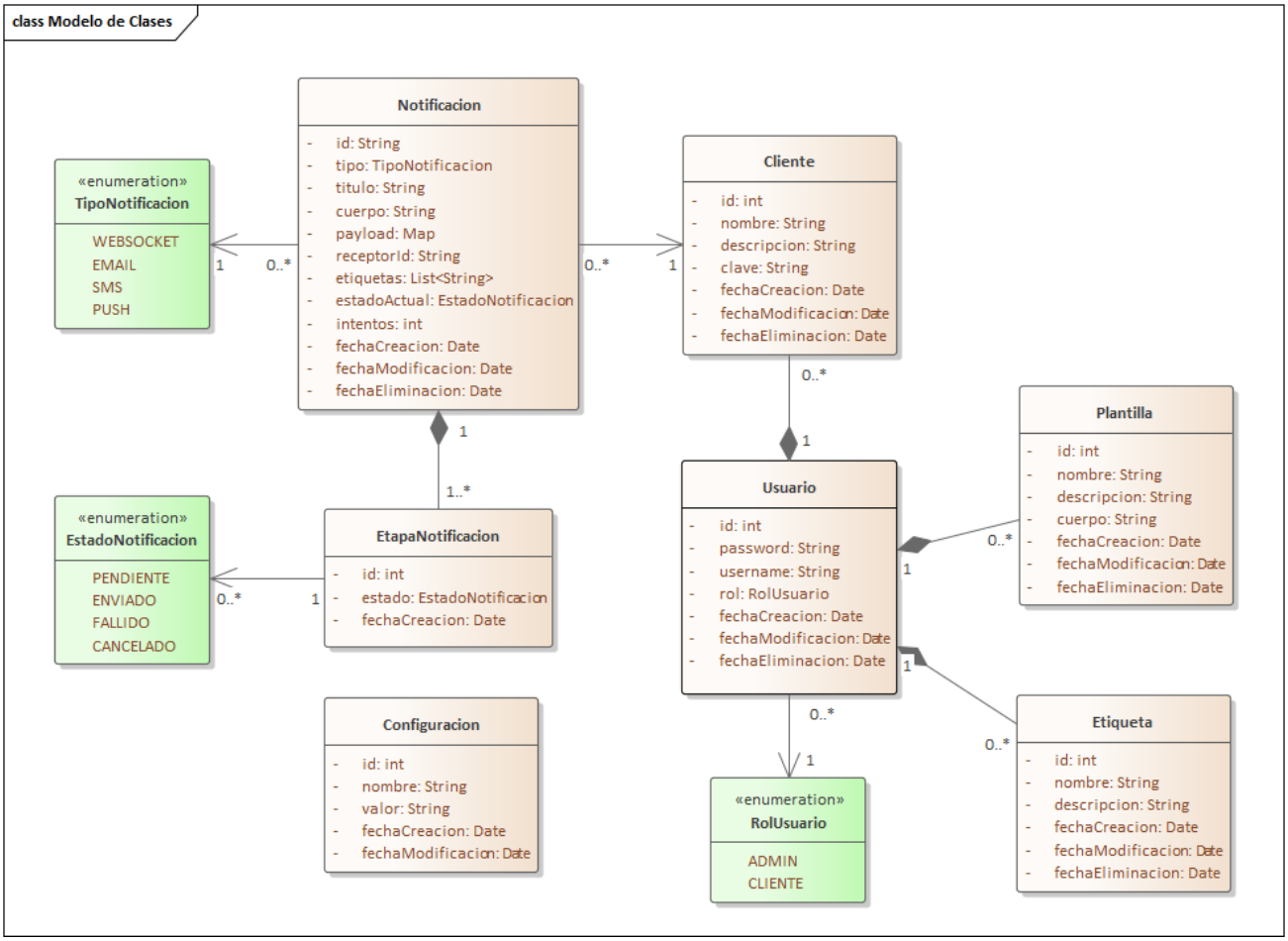


Figura N° 99: Diagrama de clases del sistema
Fuente: Creado por el autor

6.6. DIAGRAMA DE BASE DE DATOS

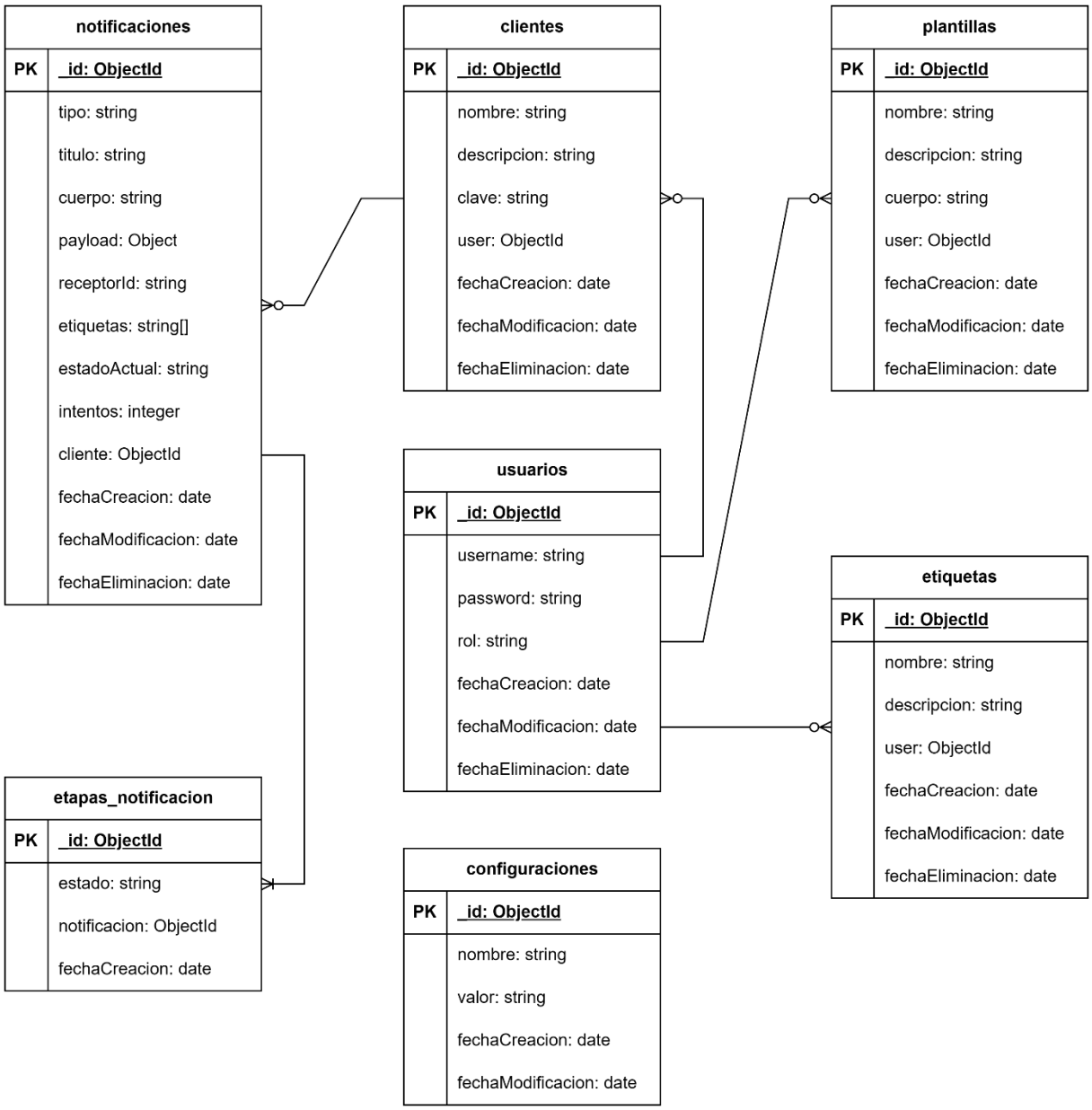


Figura N° 100: Diagrama de base de datos
Fuente: Creado por el autor

6.7. PRUEBAS

Para verificar que el sistema operara correctamente, se llevaron a cabo pruebas de integración para cada caso de uso.

Tabla N° 35: Casos de prueba

Caso de prueba	Caso de uso	Descripción	Resultado Esperado	Resultado Obtenido	Estado
CP01	Autenticar Usuario	Verificar la generación del token de acceso	Correcto	Correcto	Concluido
CP02	Registrar Usuario	Verificar el correcto registro de datos	Correcto	Correcto	Concluido
CP03	Actualizar Usuario	Verificar la correcta actualización de los datos	Correcto	Correcto	Concluido
CP04	Listar Usuarios	Verificar la correcta visualización de resultados	Correcto	Correcto	Concluido
CP05	Consultar Usuario	Verificar la correcta visualización de los datos	Correcto	Correcto	Concluido
CP06	Eliminar Usuario	Verificar la correcta eliminación del usuario	Correcto	Correcto	Concluido
CP07	Registrar Cliente	Verificar el correcto registro de datos	Correcto	Correcto	Concluido
CP08	Actualizar Cliente	Verificar la correcta actualización de los datos	Correcto	Correcto	Concluido
CP09	Consultar Cliente	Verificar la correcta visualización de los datos	Correcto	Correcto	Concluido
CP10	Listar Clientes	Verificar la visualización de resultados	Correcto	Correcto	Concluido
CP11	Eliminar Cliente	Verificar la correcta eliminación del cliente	Correcto	Correcto	Concluido
CP12	Autenticar Cliente	Verificar la generación del token de acceso	Correcto	Correcto	Concluido
CP13	Reiniciar Clave de Acceso de Cliente	Verificar la correcta actualización de los datos	Correcto	Correcto	Concluido
CP14	Registrar Plantilla	Verificar el correcto registro de datos	Correcto	Correcto	Concluido
CP15	Consultar Plantilla	Verificar la correcta visualización de los datos	Correcto	Correcto	Concluido
CP16	Actualizar Plantilla	Verificar la correcta actualización de los datos	Correcto	Correcto	Concluido
CP17	Listar Plantillas	Verificar la visualización de resultados	Correcto	Correcto	Concluido

CP18	Eliminar Plantilla	Verificar la correcta eliminación de la plantilla	Correcto	Correcto	Concluido
CP19	Registrar Etiqueta	Verificar el correcto registro de datos	Correcto	Correcto	Concluido
CP20	Consultar Etiqueta	Verificar la correcta visualización de los datos	Correcto	Correcto	Concluido
CP21	Actualizar Etiqueta	Verificar la correcta actualización de los datos	Correcto	Correcto	Concluido
CP22	Listar Etiquetas	Verificar la visualización de resultados	Correcto	Correcto	Concluido
CP23	Eliminar Etiqueta	Verificar la correcta eliminación de la etiqueta	Correcto	Correcto	Concluido
CP24	Enviar Notificación	Verificar el correcto envío de la notificación	Correcto	Correcto	Concluido
CP25	Listar Notificaciones	Verificar la visualización de resultados	Correcto	Correcto	Concluido
CP26	Consultar Notificación	Verificar la correcta visualización de los datos	Correcto	Correcto	Concluido
CP27	Cancelar Envío de Notificación	Verificar la correcta actualización de los datos	Correcto	Correcto	Concluido
CP28	Eliminar Notificación	Verificar la correcta eliminación de la notificación	Correcto	Correcto	Concluido
CP29	Establecer Conexión con el Servidor	Verificar la conexión con el servidor	Correcto	Correcto	Concluido
CP30	Reintentar envío de notificación fallida	Verificar el correcto envío de la notificación	Correcto	Correcto	Concluido
CP31	Listar Configuraciones	Verificar la visualización de resultados	Correcto	Correcto	Concluido
CP32	Actualizar Configuración	Verificar la correcta actualización de los datos	Correcto	Correcto	Concluido
CP33	Obtener Reporte de Notificaciones Enviadas	Verificar la visualización de resultados	Correcto	Correcto	Concluido
CP34	Obtener Reporte de Tiempos de Envío	Verificar la visualización de resultados	Correcto	Correcto	Concluido

Fuente: Creado por el autor

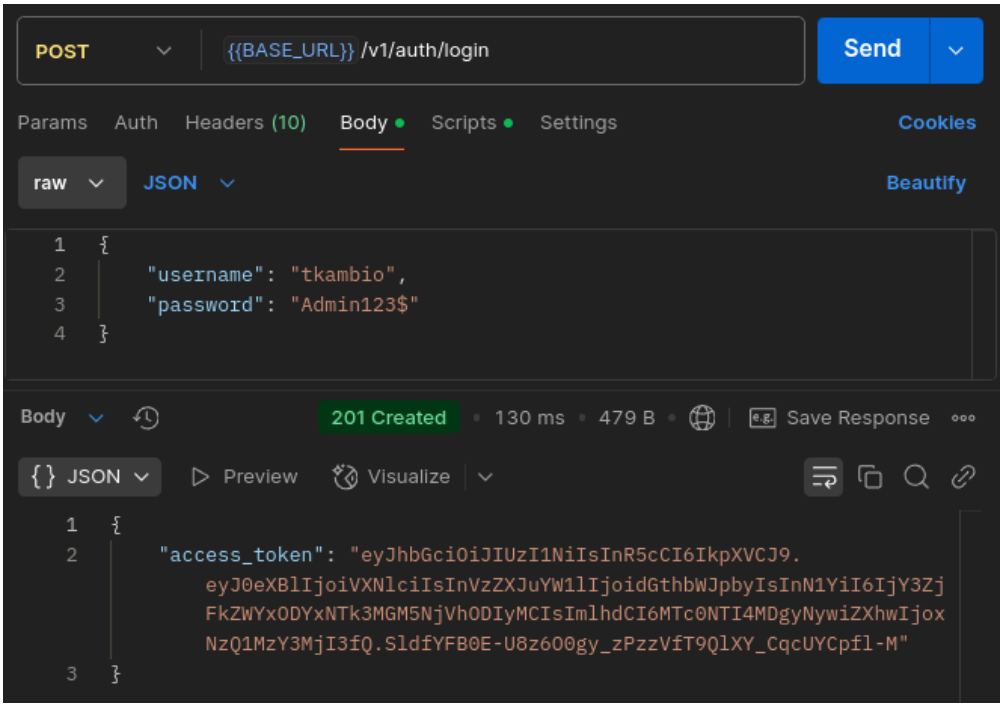


Figura N° 101: Caso de Prueba Autenticar Usuario
Fuente: Creado por el autor

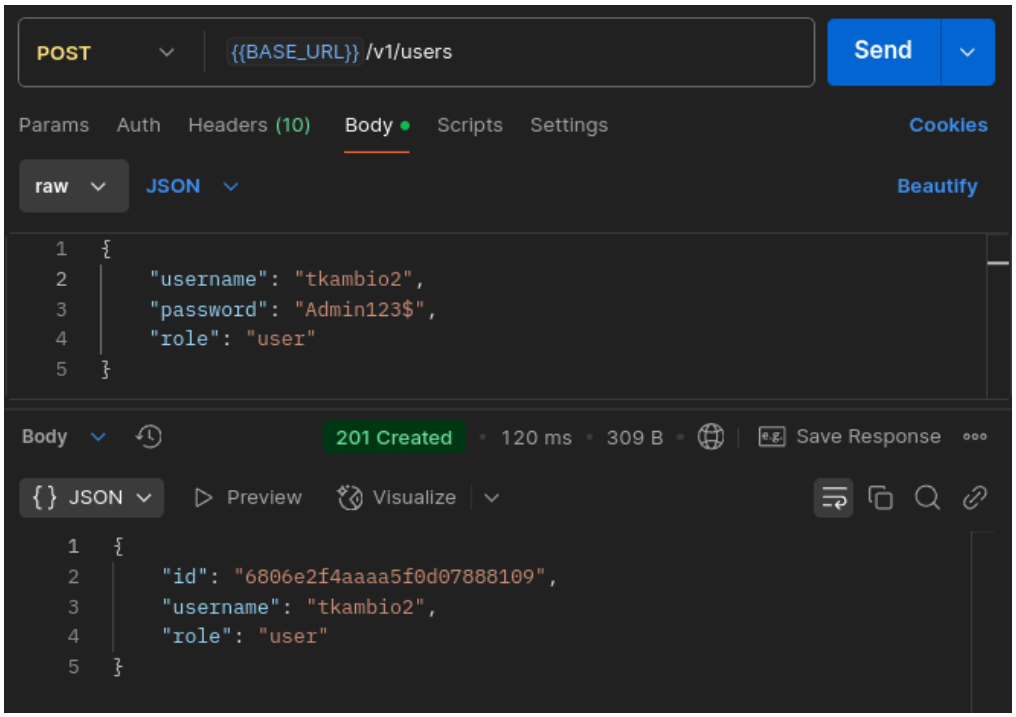


Figura N° 102: Caso de Prueba Registrar Usuario
Fuente: Creado por el autor

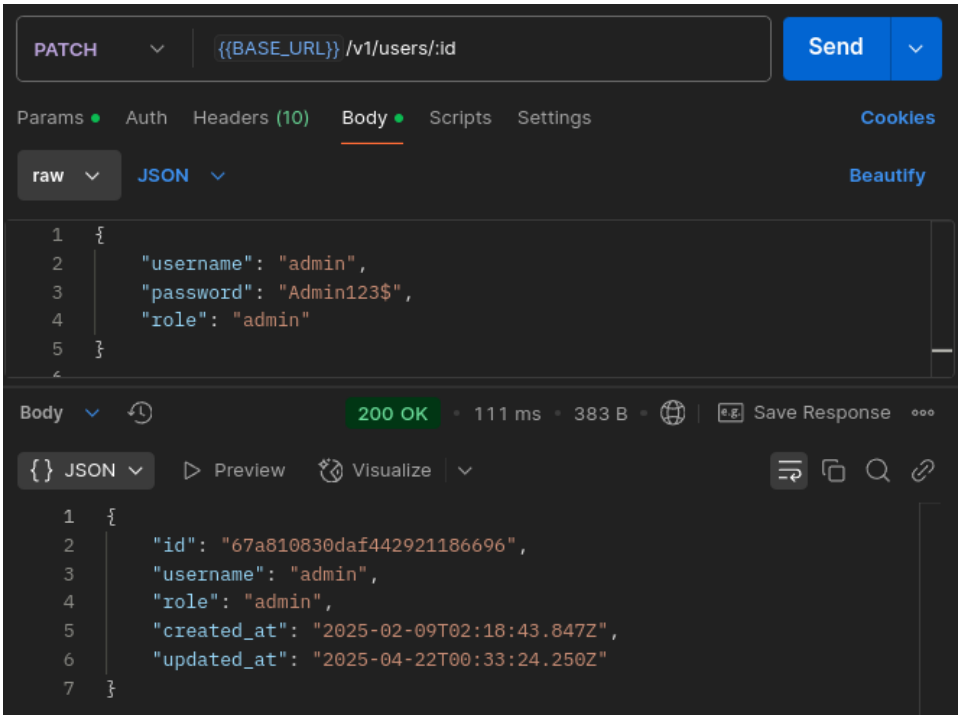


Figura N° 103: Caso de Prueba Actualizar Usuario
Fuente: Creado por el autor

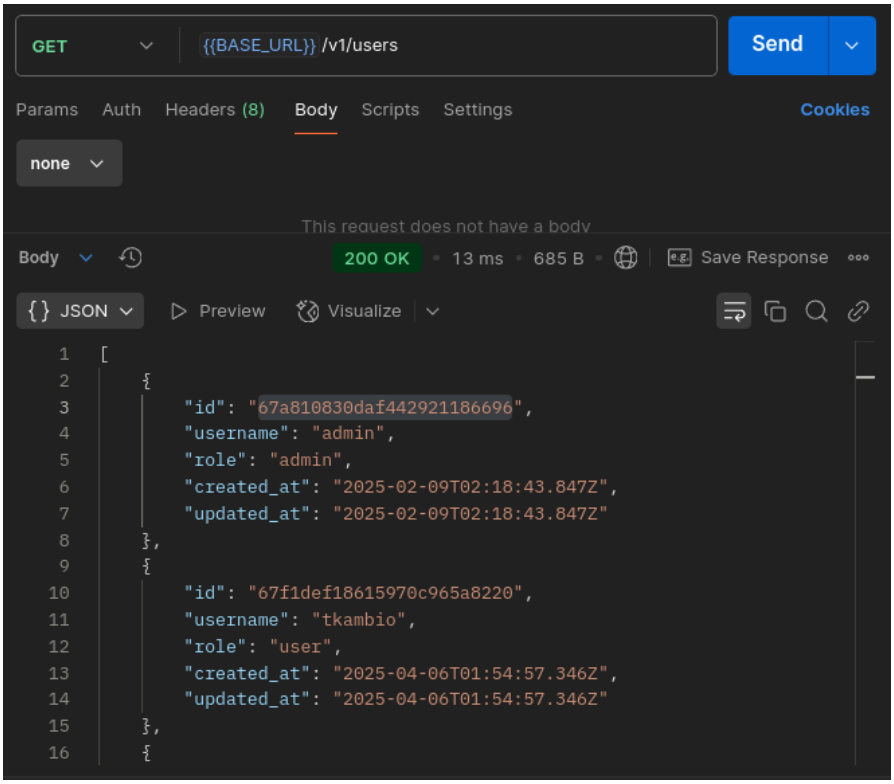


Figura N° 104: Caso de Prueba Listar Usuarios
Fuente: Creado por el autor

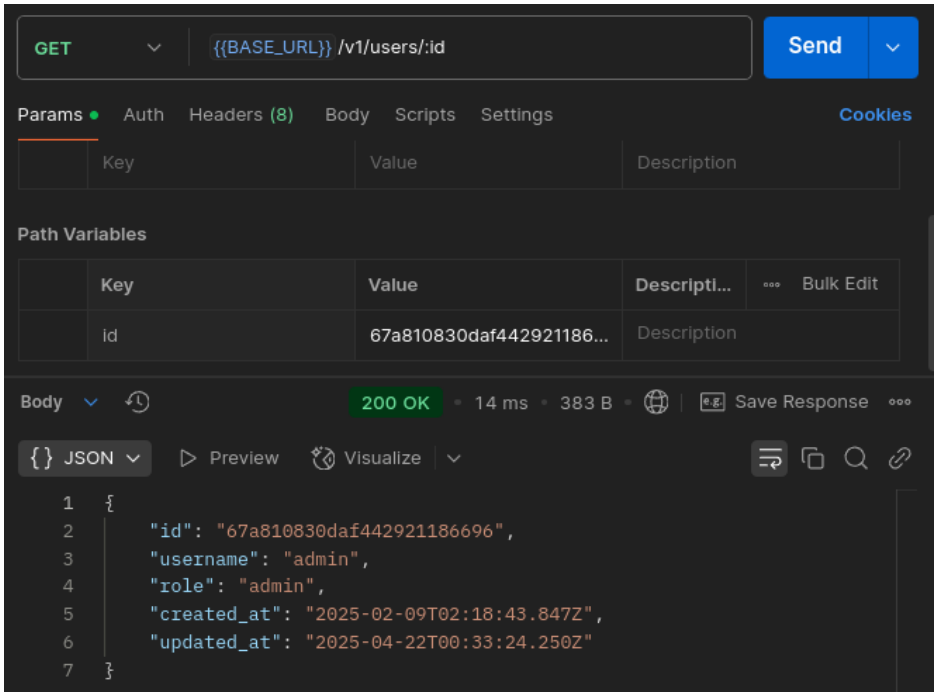


Figura N° 105: Caso de Prueba Consultar Usuario
Fuente: Creado por el autor

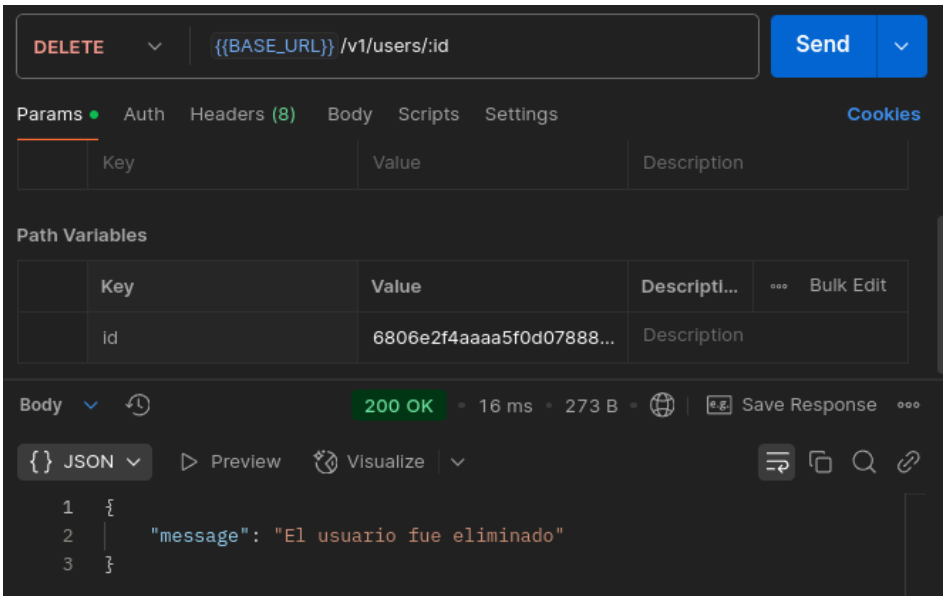


Figura N° 106: Caso de Prueba Eliminar Usuario
Fuente: Creado por el autor

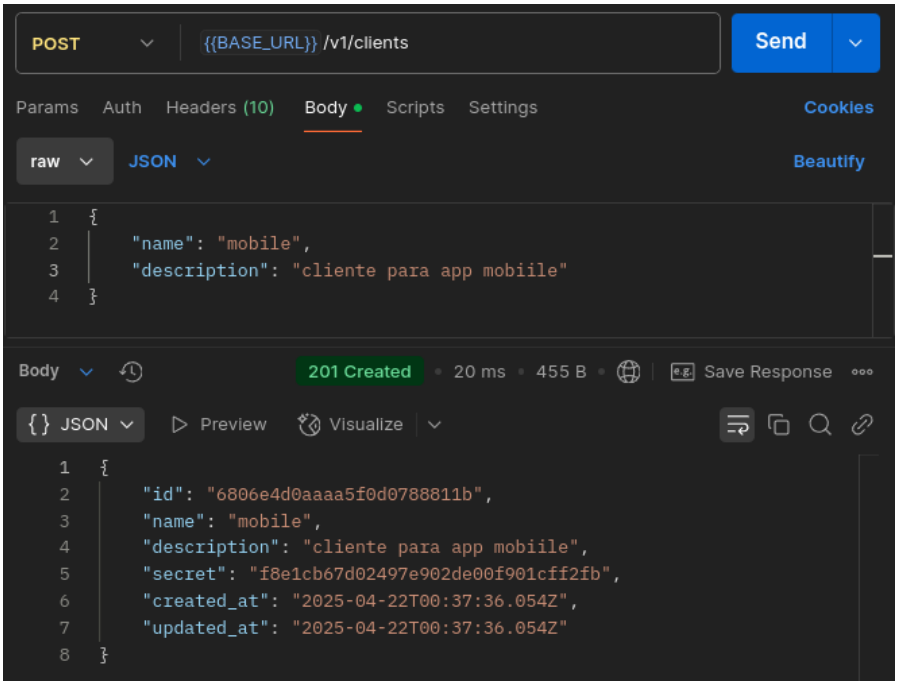


Figura N° 107: Caso de Prueba Registrar Cliente
Fuente: Creado por el autor

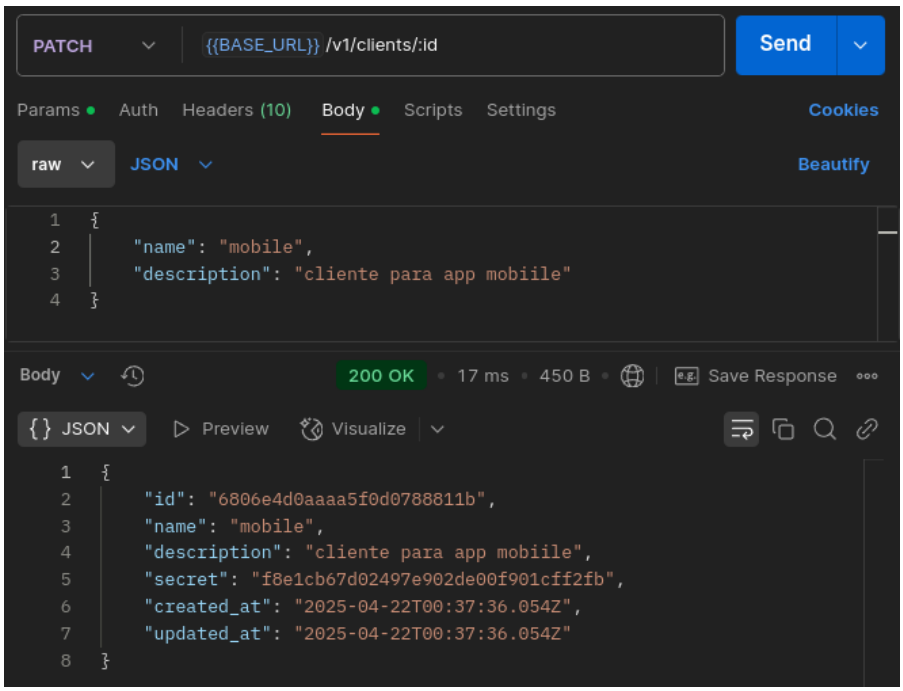


Figura N° 108: Caso de Prueba Actualizar Cliente
Fuente: Creado por el autor

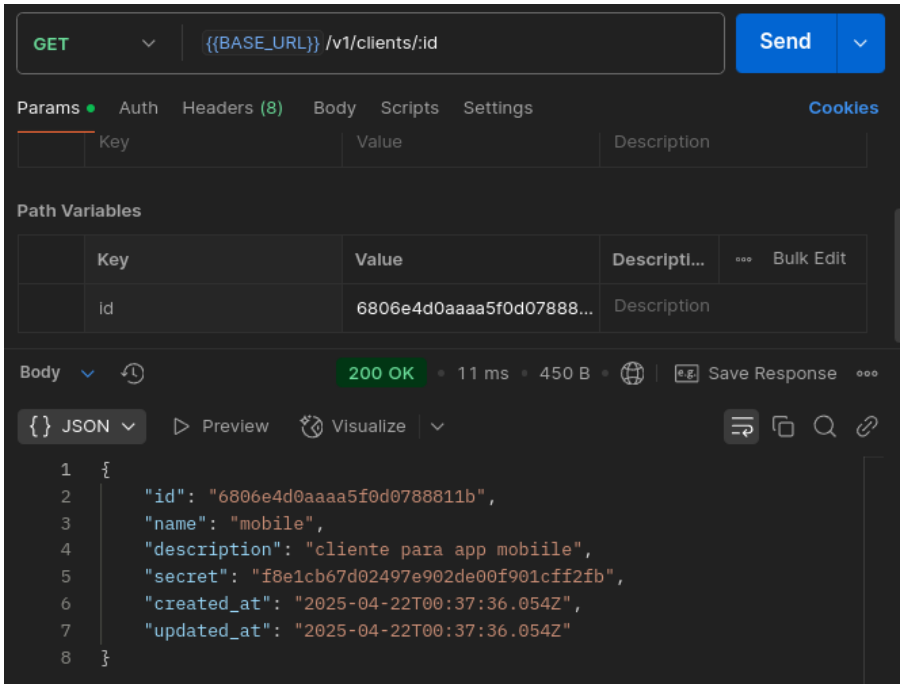


Figura N° 109: Caso de Prueba Consultar Cliente
Fuente: Creado por el autor

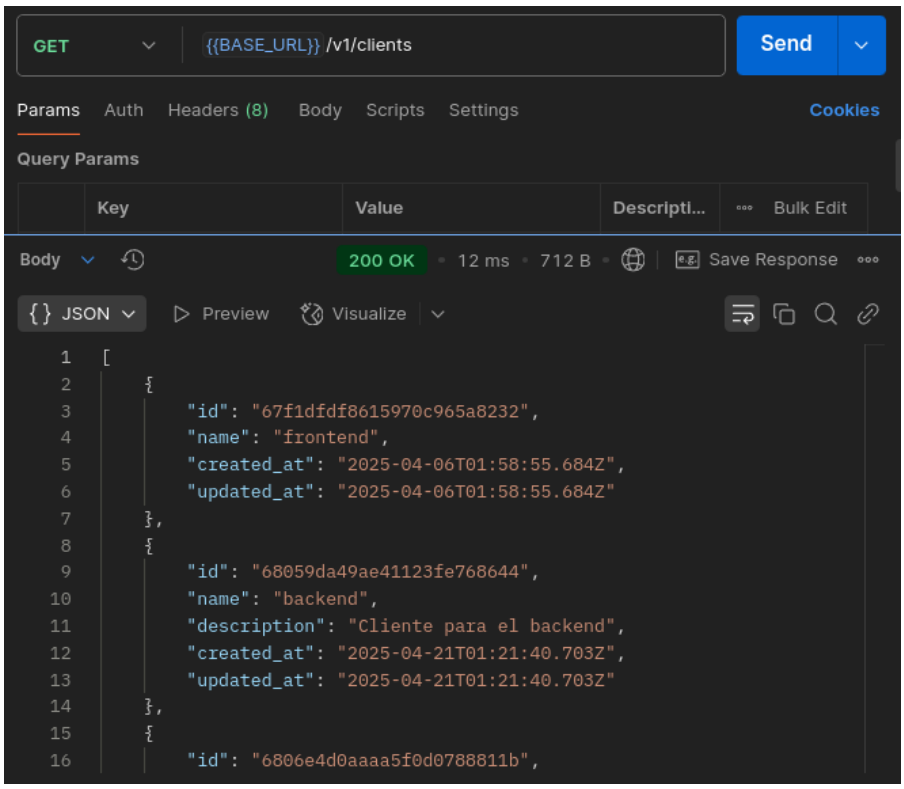


Figura N° 110: Caso de Prueba Listar Clientes
Fuente: Creado por el autor

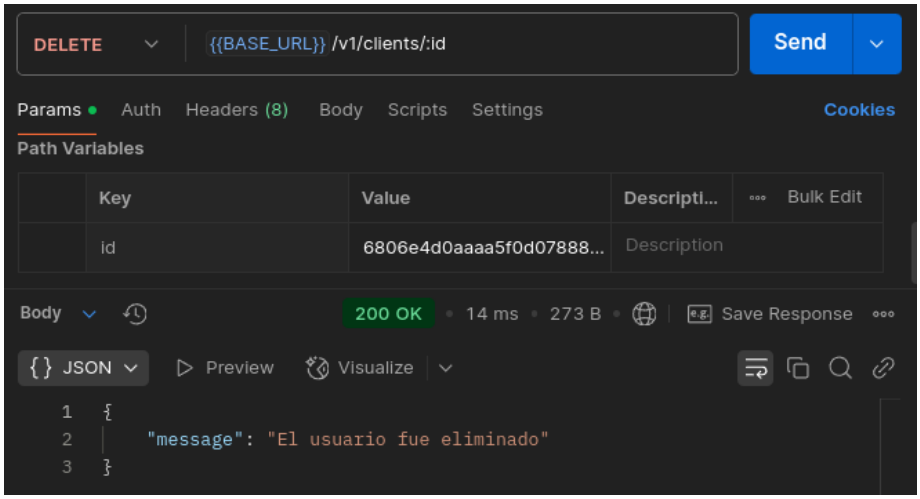


Figura N° 111: Caso de Prueba Eliminar Cliente
Fuente: Creado por el autor

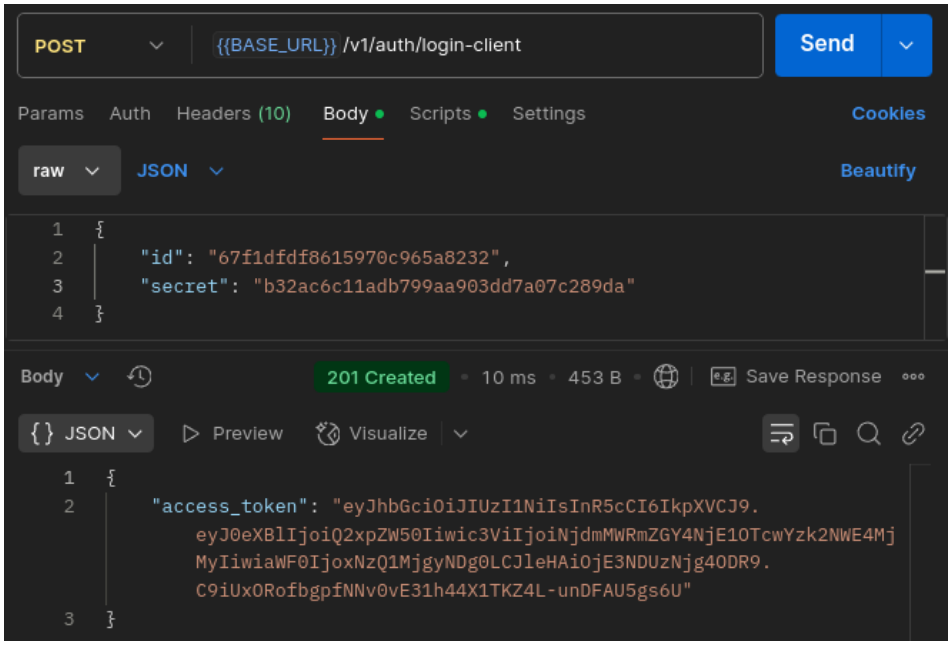


Figura N° 112: Caso de Prueba Autenticar Cliente
Fuente: Creado por el autor

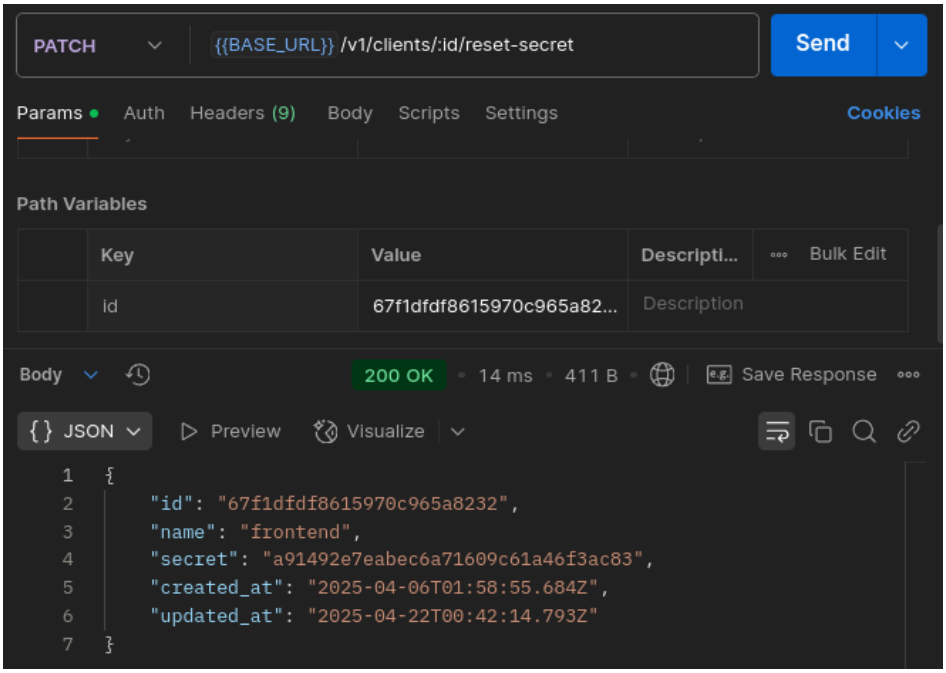


Figura N° 113: Caso de Prueba Reiniciar Clave de Acceso Cliente
Fuente: Creado por el autor

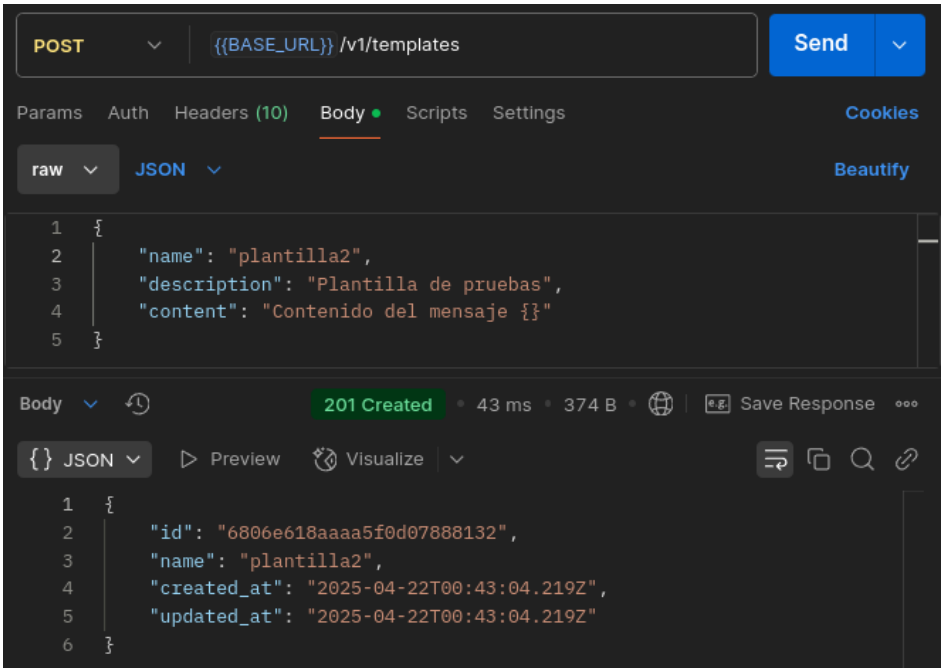


Figura N° 114: Caso de Prueba Registrar Plantilla
Fuente: Creado por el autor

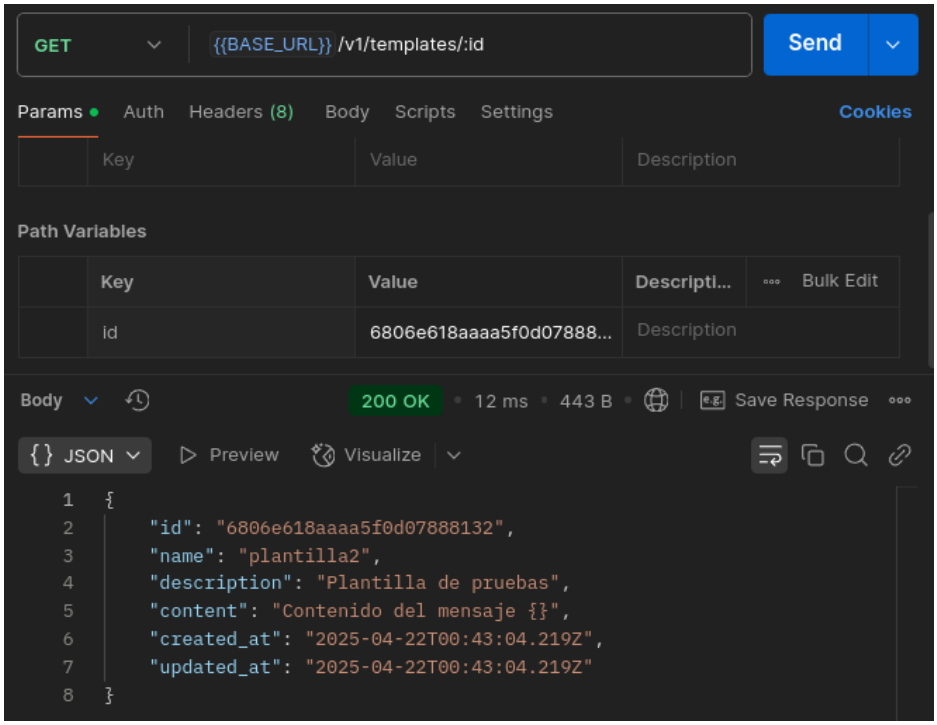


Figura N° 115: Caso de Prueba Consultar Plantilla
Fuente: Creado por el autor

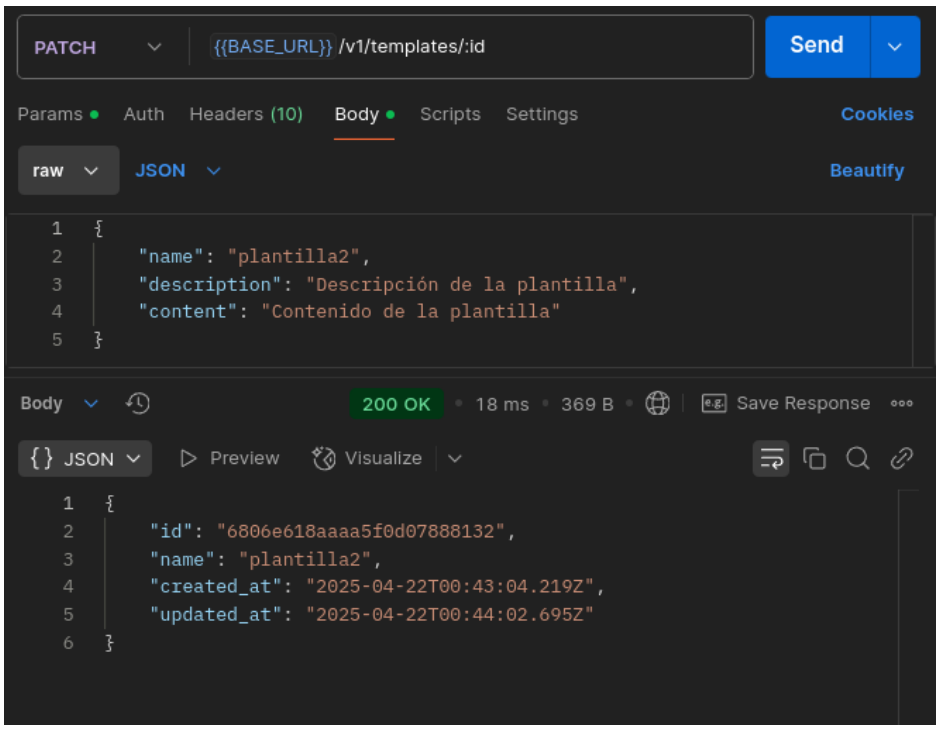


Figura N° 116: Caso de Prueba Actualizar Plantilla
Fuente: Creado por el autor

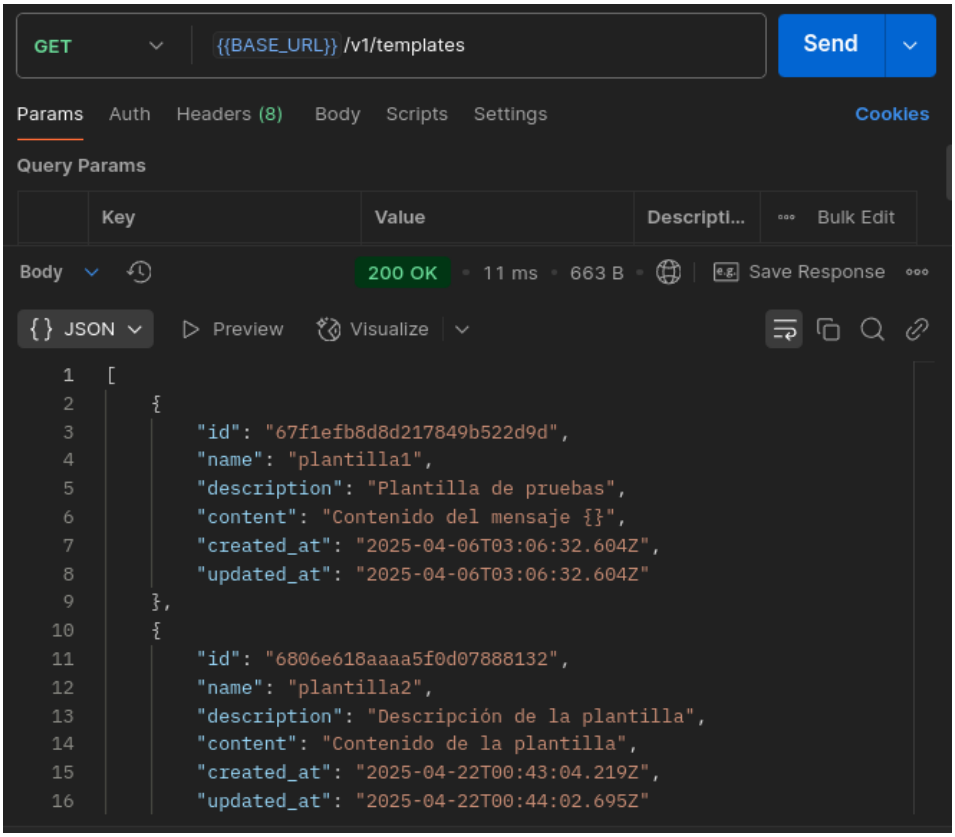


Figura N° 117: Caso de Prueba Listar Plantillas
Fuente: Creado por el autor

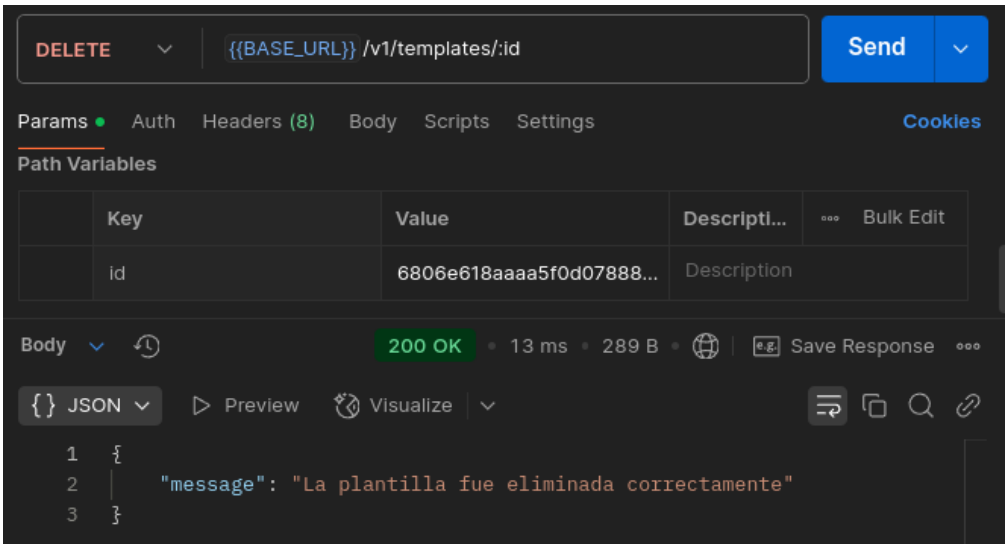


Figura N° 118: Caso de Prueba Eliminar Plantilla
Fuente: Creado por el autor

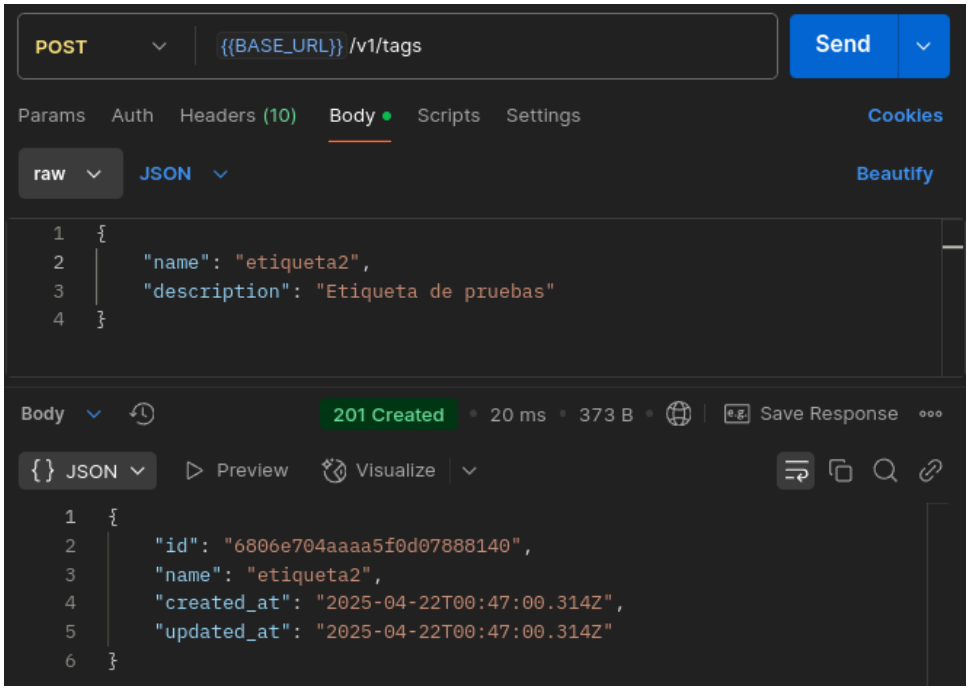


Figura N° 119: Caso de Prueba Registrar Etiqueta
Fuente: Creado por el autor

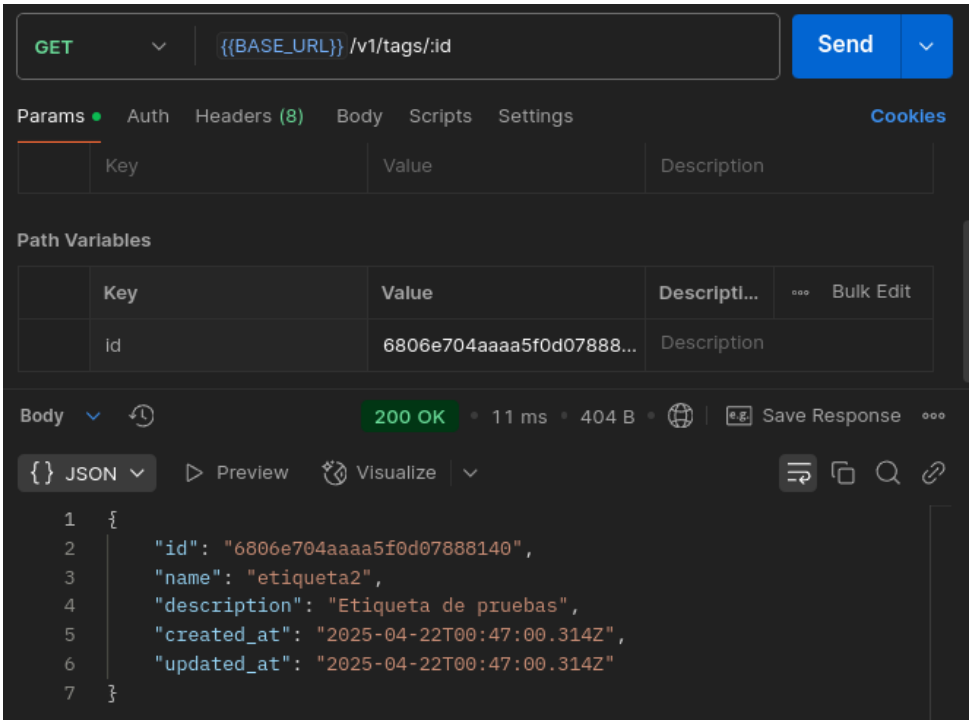


Figura N° 120: Caso de Prueba Consultar Etiqueta
Fuente: Creado por el autor

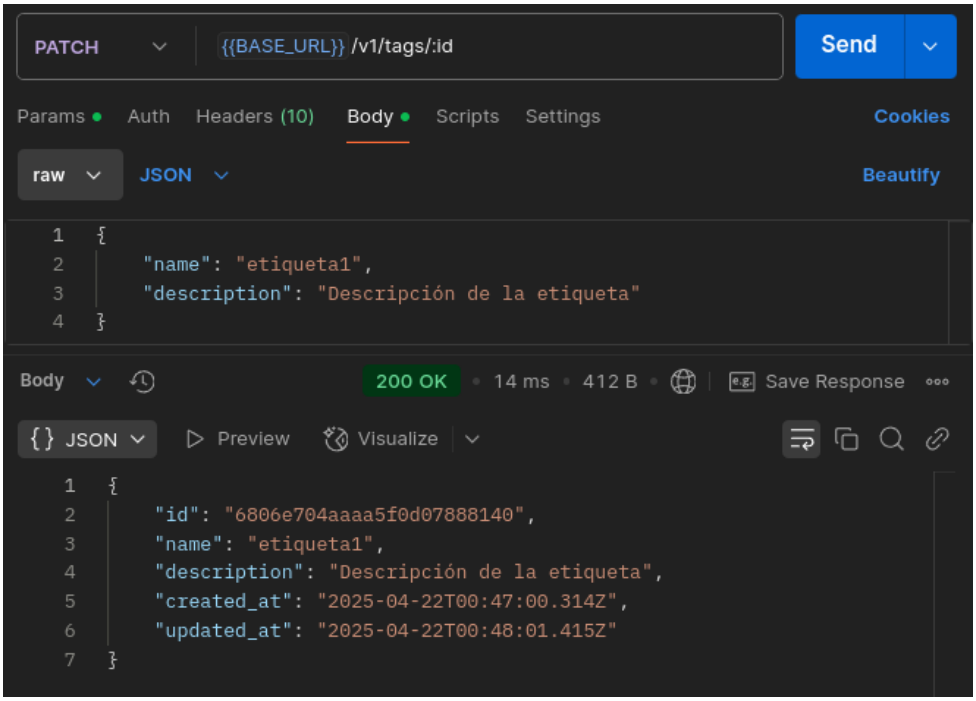


Figura N° 121: Caso de Prueba Actualizar Etiqueta
Fuente: Creado por el autor

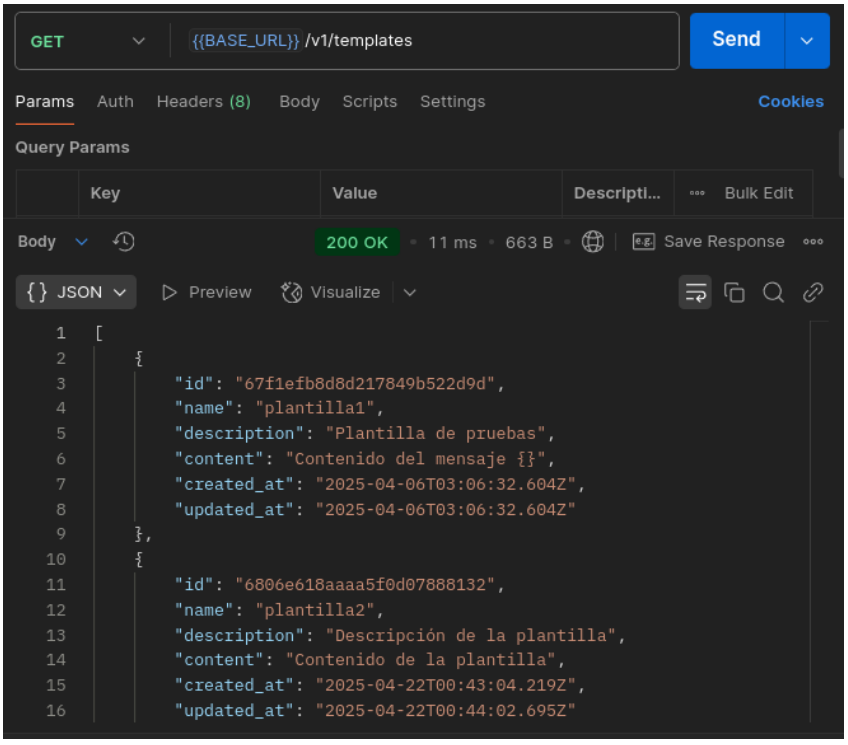


Figura N° 122: Caso de Prueba Listar Etiquetas
Fuente: Creado por el autor

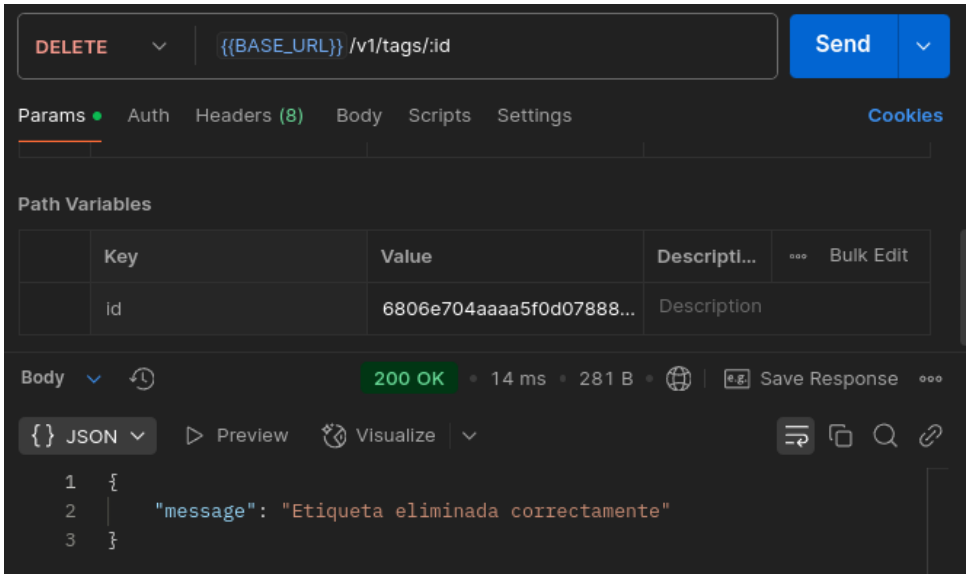


Figura N° 123: Caso de Prueba Eliminar Etiqueta
Fuente: Creado por el autor

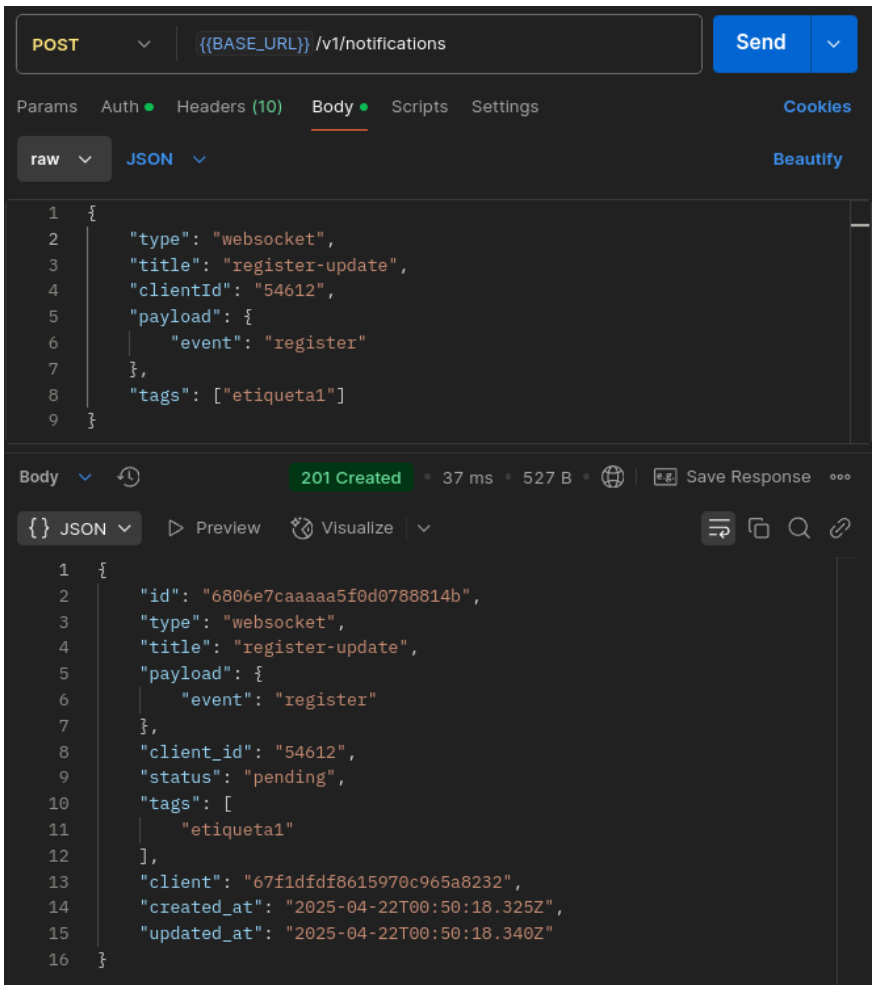


Figura N° 124: Caso de Prueba Enviar Notificación
Fuente: Creado por el autor

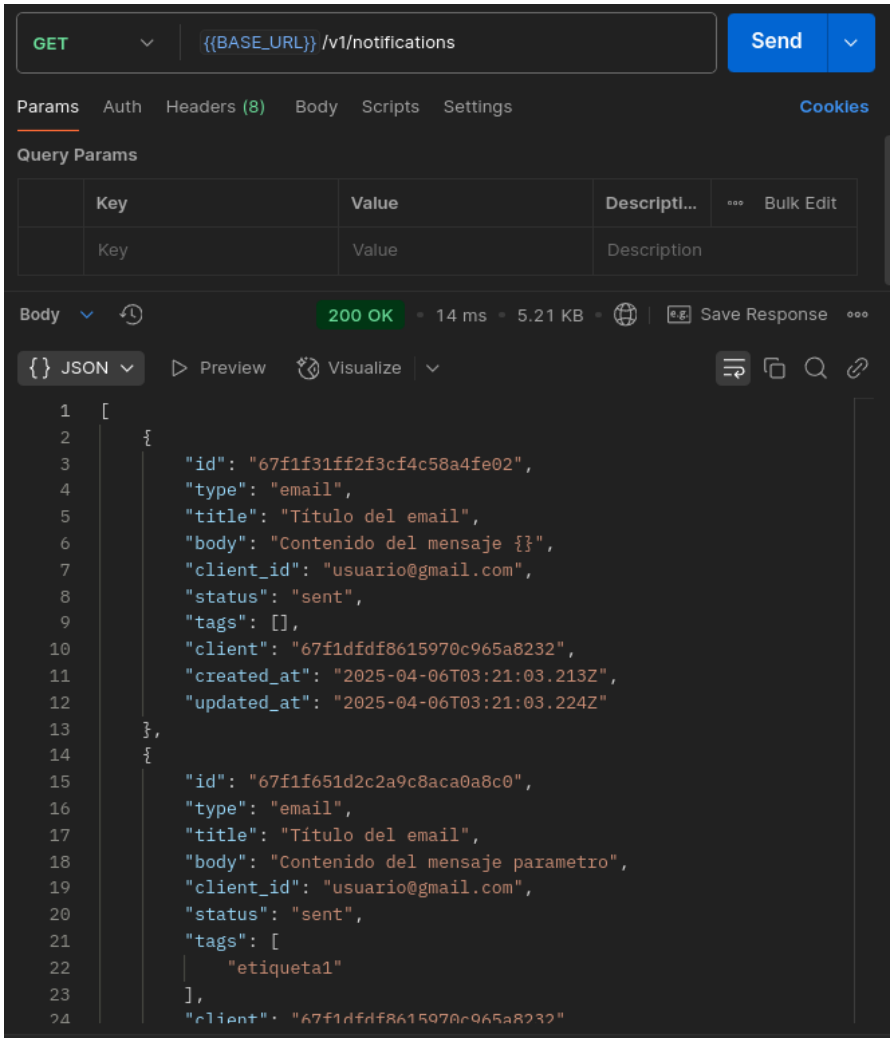


Figura N° 125: Caso de Prueba Listar Notificaciones
Fuente: Creado por el autor

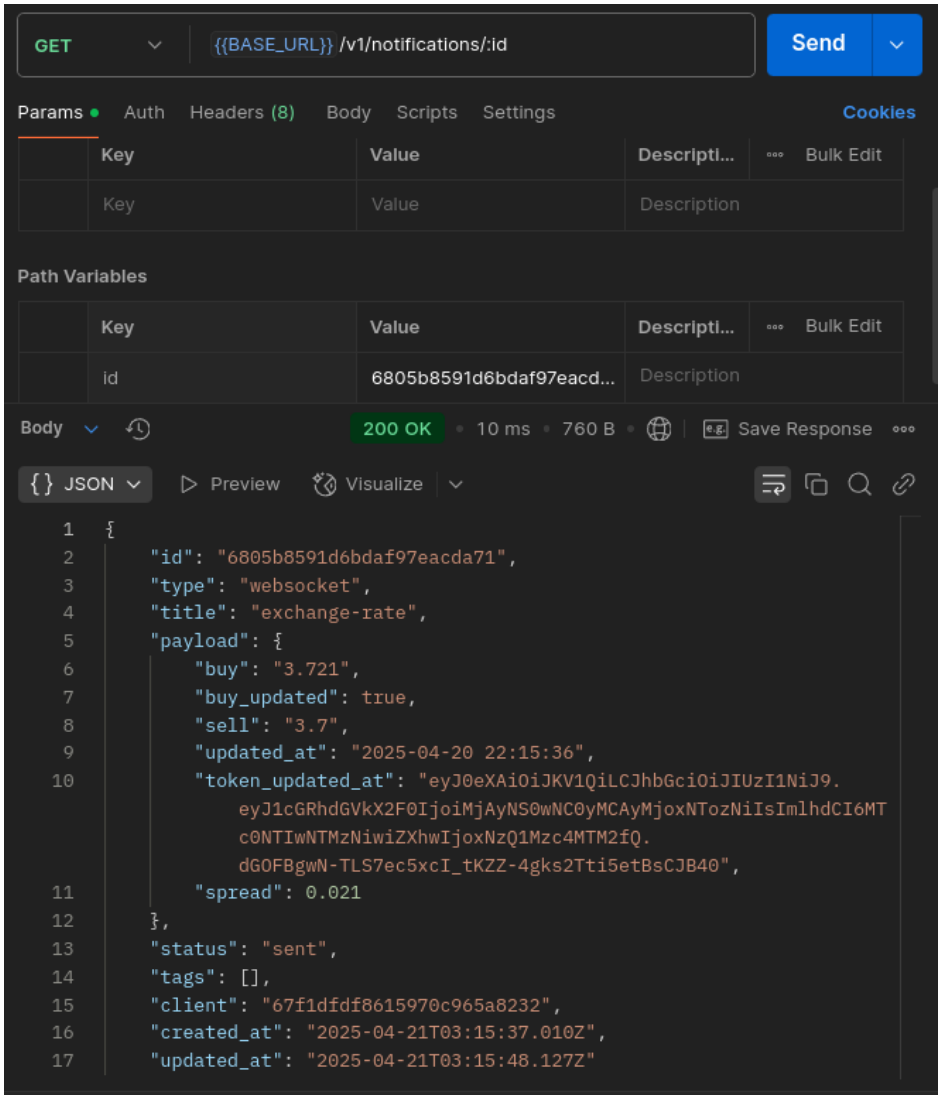


Figura N° 126: Caso de Prueba Consultar Notificación
Fuente: Creado por el autor

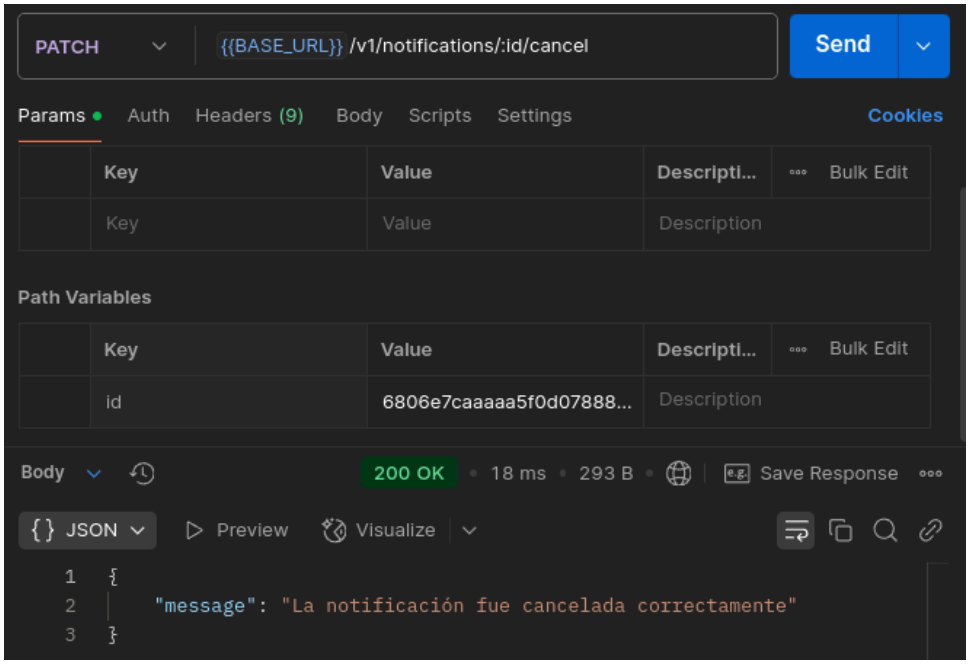


Figura N° 127: Caso de Prueba Cancelar Envío de Notificación
Fuente: Creado por el autor

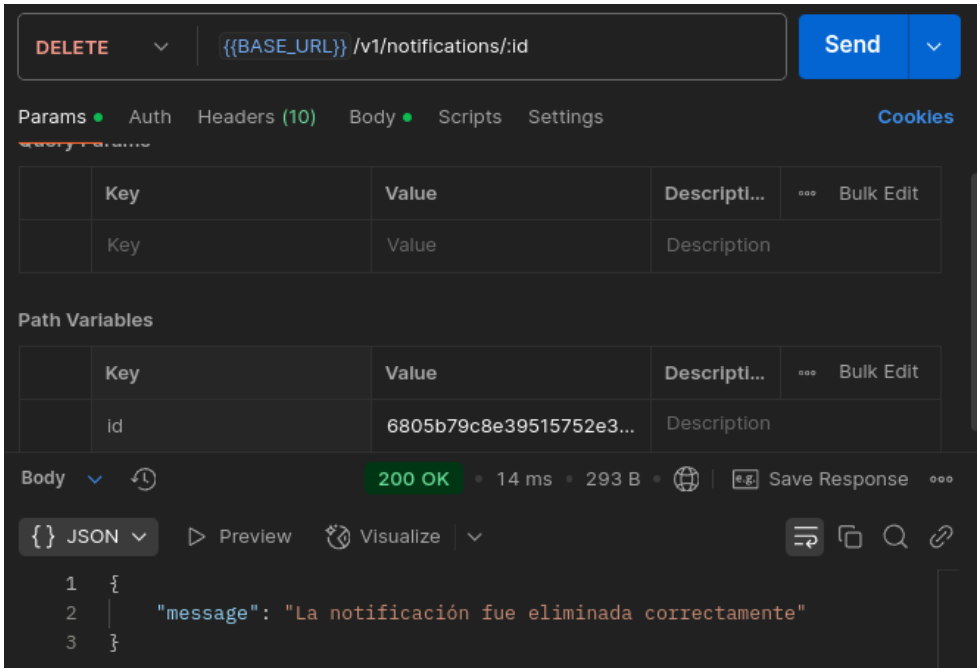


Figura N° 128: Caso de Prueba Eliminar Notificación
Fuente: Creado por el autor

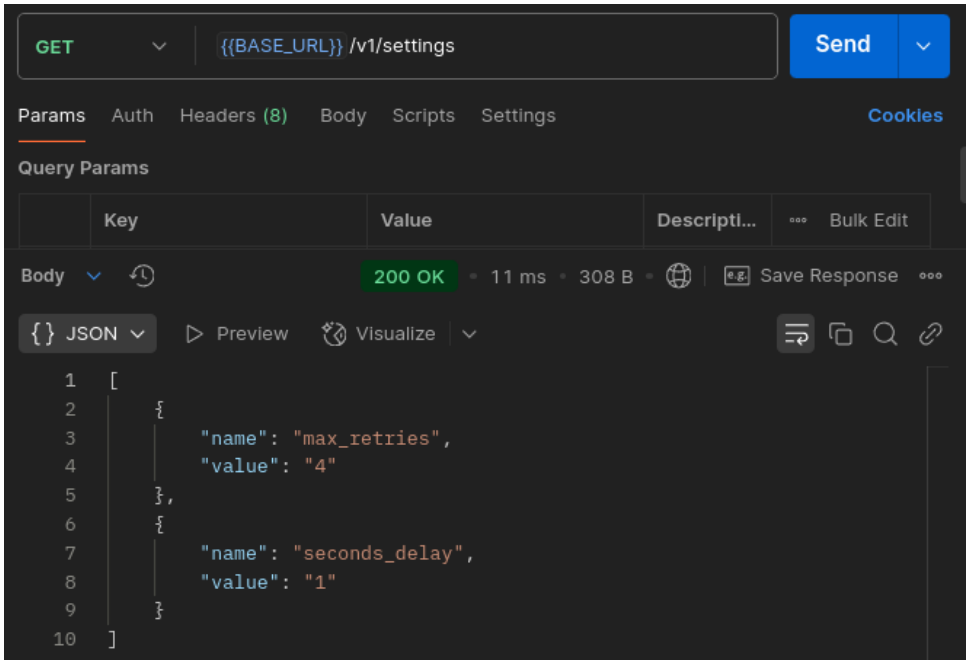


Figura N° 129: Caso de Prueba Listar Configuraciones
Fuente: Creado por el autor

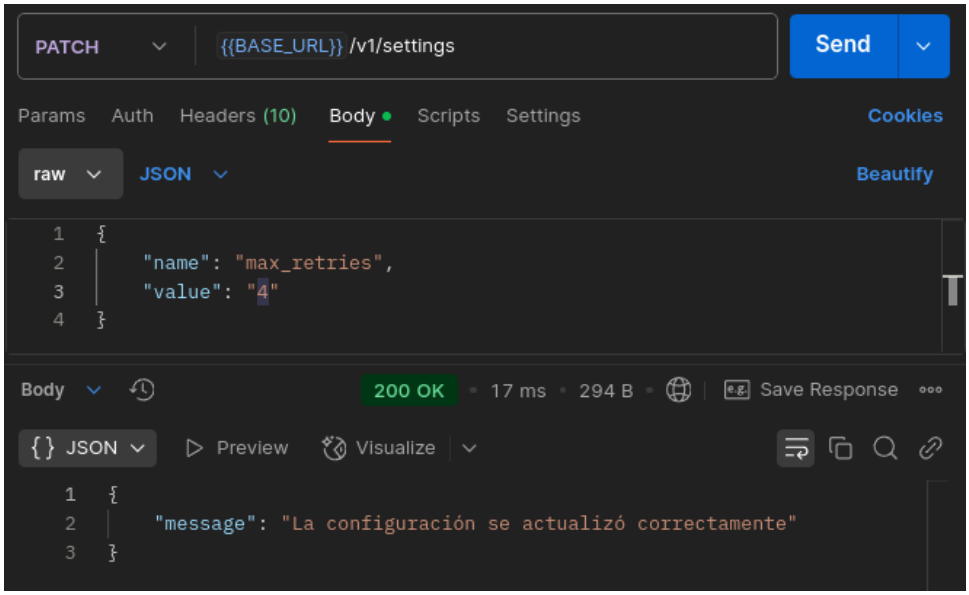


Figura N° 130: Caso de Prueba Actualizar Configuración
Fuente: Creado por el autor

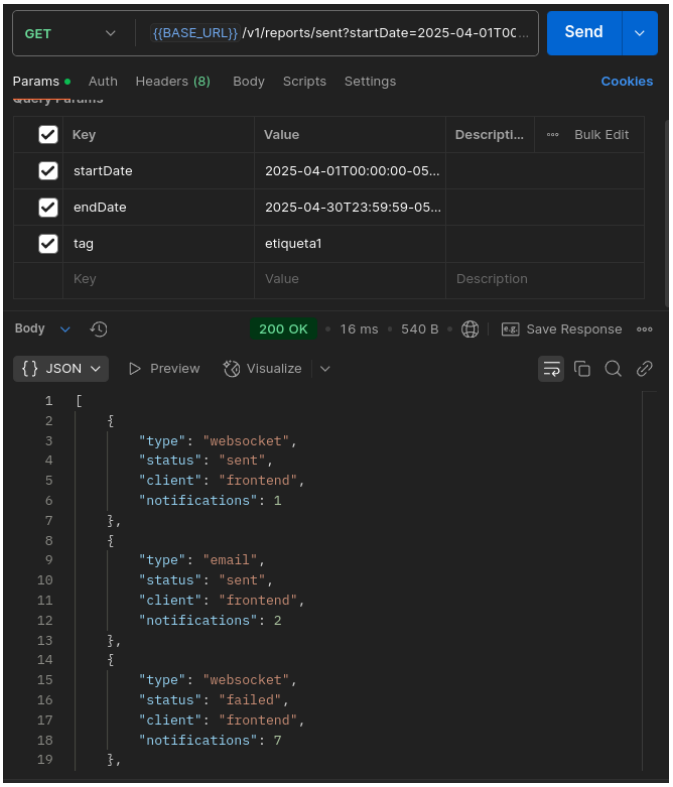


Figura N° 131: Caso de Prueba Obtener Reporte de Notificaciones Enviadas
Fuente: Creado por el autor

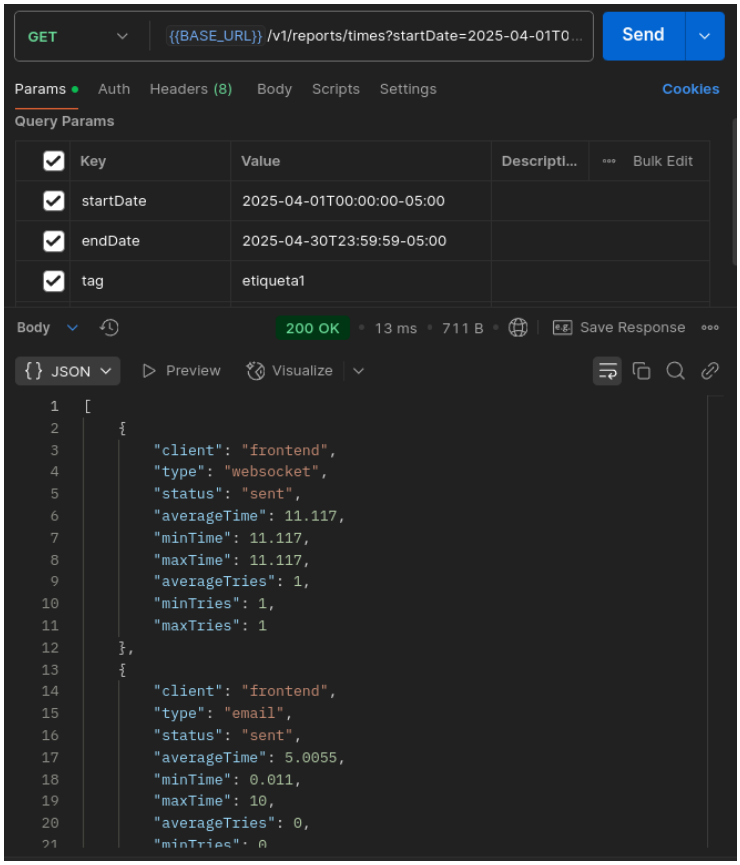


Figura N° 132: Caso de Prueba Obtener Reporte de Tiempos de Envío
Fuente: Creado por el autor

Para verificar el rendimiento y los tiempos de entrega del API, se ejecutaron 5 pruebas en las cuales se enviaron consecutivamente 500 notificaciones simulando aleatoriamente una tasa de error del 20% en la comunicación con el receptor. El API se configuró con un máximo de 3 reintentos por notificación. En cada prueba se registró el número de envíos exitosos y fallidos, así como el porcentaje de envíos exitosos. Adicionalmente, se midieron el tiempo promedio de espera en la cola de notificaciones y el tiempo promedio de entrega de las notificaciones. En la siguiente tabla se muestran los resultados obtenidos.

Tabla N° 36: Resultados de pruebas de envío de notificaciones

N°	**Envíos exitosos**	**Envíos fallidos**	**Porcentaje de éxito**	**Tiempo de espera promedio (milisegundos)**	**Tiempo de entrega promedio (milisegundos)**
1	496	4	99.20	3325.65	3416.10
2	497	3	99.40	2770.99	3118.78
3	497	3	99.40	2673.00	3017.15
4	499	1	99.80	2445.04	2531.36
5	497	3	99.40	3228.92	3160.52
Promedio			99.44	2888.72	3048.78

Fuente: Creado por el autor

El análisis de los resultados muestra que el API presenta un alto porcentaje de éxito, con un promedio general del 99.44%. El tiempo promedio de espera en la cola de notificaciones varió entre 2445.04 milisegundos y 3325.65 milisegundos con un promedio general de 2888.72 milisegundos, mientras que el tiempo promedio de entrega de notificaciones varió entre 2531.36 milisegundos y 3416.10 milisegundos con un promedio general de 3048.78 milisegundos. Se observa que la prueba con mayor eficiencia fue la número 4, con 99.8% de éxito y los tiempos más bajos en espera y envío. Por otro lado, las demás pruebas muestran una consistencia en los valores, lo que sugiere estabilidad en el rendimiento del API.

VII. UBICACIÓN DE LAS EXPERIENCIAS EN EL MARCO DEL SUSTENTO TEÓRICO

Contar con una comprensión clara del marco teórico relacionado con el problema de investigación fue esencial durante el desarrollo del proyecto. Enseguida, se presenta una visión general sobre cómo se incorporan las diversas experiencias dentro de este marco teórico.

- **API:** Es la interfaz del sistema de notificaciones, la cual servirá para que otras aplicaciones de la empresa la puedan integrar y hacer uso de ella para el envío de notificaciones de acuerdo con sus propios procesos.
- **REST:** El proyecto de notificaciones hace uso de la arquitectura REST para la transmisión de información de manera estandarizada y comprensible mediante HTTP. Esto facilita la integración con otras aplicaciones de la empresa.
- **JSON:** Es el formato estándar para el intercambio de datos entre el API de notificaciones y los clientes que hacen uso de ella. Esto garantiza una comunicación eficiente y rápida entre los componentes del sistema ya que una amplia cantidad de lenguajes de programación lo soportan.
- **Notificación:** Es el núcleo del sistema, ya que representa el mecanismo mediante el cual se informa a los usuarios sobre eventos importantes en tiempo real. En este proyecto se implementan los canales: websocket, push, email y sms. La API debe gestionar el envío eficiente de las notificaciones y gestionar posibles fallos.
- **Websocket:** Es el protocolo que permite la comunicación bidireccional y en tiempo real entre el servidor y los clientes. A diferencia de las peticiones HTTP tradicionales, Websocket establece una conexión persistente. En el presente proyecto se implementó un servidor de Websocket para enviar notificaciones mediante este canal gestionando los fallos en la conectividad con los clientes.
- **Notificación push:** Es el canal de notificación que permite enviar mensajes a los clientes sin que estos tengan que estar consultando activamente la aplicación. La API de notificaciones se integra con el servicio Firebase Cloud Messaging (FCM) para realizar el envío de este tipo de notificaciones.

- **Colas:** Es la tecnología que permite la gestión eficiente del envío de notificaciones ya que procesan cada notificación de forma ordenada, evitando la sobrecarga del sistema y asegurando que cada mensaje se entregue sin perderse, incluso en caso de fallos. En el presente proyecto se utilizó la librería BullMQ para gestionar las colas con reintentos automáticos.
- **MongoDB:** Es una base de datos NoSQL orientada a documentos, la cual es adecuada para gestionar una gran cantidad de notificaciones en este proyecto ya que ofrece flexibilidad en la definición de los modelos, escalabilidad horizontal, velocidad de lectura y mecanismos de recuperación automática ante fallos.

VIII. APORTES LOGRADOS PARA EL DESARROLLO DEL CENTRO LABORAL

Los aportes del proyecto permiten lo siguiente:

- Disminuir la cantidad de notificaciones en tiempo real perdidas debido a problemas de conectividad entre el cliente y el servidor, especialmente en horarios de alta demanda de operaciones.
- Centralizar en un solo microservicio la funcionalidad de envío de notificaciones mediante múltiples canales, lo cual permite desacoplar dicha funcionalidad de otras aplicaciones y facilitar el trabajo de mantenimiento al equipo de desarrollo.
- La eficacia de la API permite a la empresa hacer un mayor uso de las notificaciones para mantener a los clientes informados sobre: variaciones en el tipo de cambio, cupones, promociones, etc. Esto eleva la confianza de los usuarios en la plataforma y los incentiva a hacer un mayor uso de esta.

IX. APORTES PARA LA FORMACIÓN PROFESIONAL

A partir de la experiencia ganada en la implementación de este proyecto y mi recorrido profesional, he logrado:

- Diseñar una arquitectura que implementa gestión de fallos mediante colas con BullMQ y Redis, lo cual mejora la resiliencia del sistema, evitando interrupciones y asegurando la entrega efectiva de mensajes.
- Diseñar una API REST escalable con NestJS, mediante el uso de módulos reutilizables, controladores y servicios optimizados para procesamiento en tiempo real.
- Trabajar con una base de datos no relacional orientada a documentos como es MongoDB, diseñada para realizar consultas rápidas y eficientes con grandes volúmenes de datos.
- Implementar la integración con Firebase para el envío de notificaciones push a dispositivos móviles.
- Implementar el envío de SMS utilizando el proveedor Brevo.

X. CONCLUSIONES Y RECOMENDACIONES

10.1. CONCLUSIONES

- Se logró garantizar una tasa de éxito de entrega de notificaciones de 99.44 en promedio, la cual es superior al 99%.
- Se logró mantener un tiempo promedio de espera en cola en el envío de notificaciones de 2888.72 milisegundos, la cual es menor a 4 segundos.
- Se logró asegurar un tiempo promedio de entrega de notificaciones de 3048.78 milisegundos, la cual es menor a 5 segundos.

10.2. RECOMENDACIONES

- Implementar nuevos canales de notificación, como: Whatsapp, Telegram o Slack, de acuerdo con las necesidades de la empresa.
- Agregar la posibilidad de adjuntar archivos multimedia en el envío de las notificaciones.
- Implementar el uso de big data y analytics para ajustar estrategias de envío de notificaciones en tiempo real.

XI. REFERENCIAS BIBLIOGRÁFICAS

Academia Lab. (2025). *Sistema de notificación*. Retrieved 15 de abril de 2025, from https://academia-lab.com/enciclopedia/sistema-de-notificacion/

AppMaster. (4 de septiembre de 2023). *Principios de diseño REST*. Retrieved 27 de mayo de 2025, from https://appmaster.io/es/blog/principios-de-diseno-descansan

AppMaster. (25 de octubre de 2023). *Protocolo WebSocket: una inmersión profunda en su funcionamiento*. Retrieved 15 de abril de 2025, from https://appmaster.io/es/blog/protocolo-websocket-como-funciona-es

Banker, K., Bakkum, P., Verch, S., Garrett, D., & Hawkins, T. (2016). *MongoDB in Action.* Shelter Island: Manning Publications Co.

Bender, C., Deco, C., González, J., Hallo, M., & Ponce, J. (2014). *Tópicos avanzados de Bases de datos.* Proyecto LATIn.

Cámara de Comercio de Bogotá. (15 de abril de 2025). *El mundo conectado por las API.* https://bibliotecadigital.ccb.org.co/server/api/core/bitstreams/87d206e6-ba8c-4f63-a858-8d2e3027eafe/content

EcuRed. (7 de diciembre de 2020). *Sistemas de notificaciones y alertas*. Retrieved 27 de mayo de 2025, from https://www.ecured.cu/Sistemas_de_notificaciones_y_alertas

Flanagan, D. (2020). *JavaScript: The Definitive Guide* (7ma ed.). Madrid: O'Reilly.

Joyanes, L., & Zahonero, I. (1998). *Estructura de Datos. Algoritmos, abstracción y objetos.* McGraw Hill.

Nida, F. (27 de mayo de 2024). *¿Qué significa API y cómo funciona una API?* Retrieved 15 de abril de 2025, from Astera: https://www.astera.com/es/knowledge-center/what-does-api-stand-for/

Parker, Z., Poe, S., & Vrbsky, S. (2013). Comparing NoSQL MongoDB to an SQL DB. *Proceedings of the 51st ACM Southeast Conference.* New York: ACM.

RedHat. (20 de enero de 2023). *¿Qué es una API y cómo funciona?* Retrieved 15 de abril de 2025, from https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

Taskforce.sh Inc. (15 de abril de 2025). *What is BullMQ*. https://docs.bullmq.io/

Thierry, R. (2023). *Jakarta EE Desarrolle aplicaciones web en Java.* Ediciones ENI.

Tomás, J., & Lloret, J. (2022). *El gran libro de Android* (9na ed.). Marcombo.

Tomás, J., Puga, G., Santamaría, D., & Barroso, J. (2019). *El gran libro de Android Avanzado* (5ta ed.). Marcombo.

Universidad Nacional del Santa. (s.f.). *INGENIERÍA DE SISTEMAS E INFORMÁTICA*. Retrieved 14 de mayo de 2025, from https://www.uns.edu.pe/#/ingenieria/ingenieria-de-sistemas-e-informatica