ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE SISTEMA ECOMMERCE CUSTOMER TO CUSTOMER PARA ELECTRODOMÉSTICOS

DESARROLLO BACKEND

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN DESARROLLO DE SOFTWARE

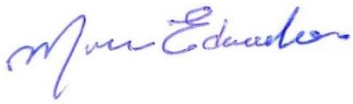
EDUARDO MIGUEL MUZO CANSIGÑA

DIRECTOR: ING YADIRA FRANCO R. Mg.

DMQ, marzo 2023

CERTIFICACIONES

Yo, EDUARDO MIGUEL MUZO CANSIGÑA declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.



EDUARDO MIGUEL MUZO CANSIGÑA

eduardo.muzo@epn.edu.ec

eduardomuzo123456@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por EDUARDO MIGUEL MUZO CANSIGÑA, bajo mi supervisión.

Ing. Yadira Franco R Mg.

DIRECTOR

Yadira.franco@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

EDUARDO MIGUEL MUZO CANSIGÑA

DEDICATORIA

A mi madre que siempre me apoyo en todas mis decisiones, además me enseñó a creer en mí y que siempre me ánimo hasta por el logro más mínimo que obtenía.

A mi familia por apoyarme a seguir adelante, brindándome su amor incondicional y su apoyo constante en todo lo que hago.

A mis queridas mascotas que siempre estuvieron dándome mucho cariño y que son mis mejores amigos.

EDUARDO MIGUEL MUZO CANSIGÑA

AGRADECIMIENTO

Quiero expresar mi más profundo agradecimiento a la Escuela Politécnica Nacional por haberme brindado la oportunidad de formar parte de una de las mejores universidades del país. Gracias a la calidad de la educación que recibí, he podido desarrollarme tanto profesional como personalmente.

Quiero agradecer especialmente a todos los maestros que, a lo largo de mi carrera, me han brindado su apoyo, guía y enseñanzas. Su dedicación y conocimientos me han permitido adquirir las habilidades necesarias para enfrentar los retos de la vida profesional con éxito.

EDUARDO MIGUEL MUZO CANSIGÑA

ÍNDICE DE CONTENIDO

RESUMEN

El objetivo de este proyecto es mejorar la venta de electrodomésticos mediante un sistema web seguro y fácil de usar. Los usuarios pueden registrarse, iniciar sesión, gestionar sus productos, realizar comentarios y reportar sospechas, así como realizar compras y ventas mediante un chat. El sistema también cuenta con un método de pago y garantiza la seguridad de la información de los usuarios.

A diferencia de otras plataformas, este sistema brinda una experiencia amigable para los clientes y vendedores, garantizando la seguridad de la transacción y la información. Para alcanzar estos objetivos, el proyecto se divide en tres secciones: objetivos y alcance, metodología de desarrollo y actividades realizadas con resultados obtenidos, pruebas y conclusiones.

En la sección de objetivos y alcance se definen los objetivos específicos del proyecto y el alcance de las funcionalidades del sistema web. En la metodología de desarrollo se describe el modelado de la base de datos, el diseño de la arquitectura y las herramientas utilizadas para el componente backend. Por último, en la sección de actividades realizadas se presentan los resultados obtenidos, las pruebas realizadas al sistema, las conclusiones y recomendaciones finales.

En resumen, este proyecto busca mejorar la venta de electrodomésticos mediante un sistema web seguro y fácil de usar. El documento se divide en tres secciones para describir los objetivos y alcance, la metodología de desarrollo y las actividades realizadas con resultados obtenidos.

PALABRAS CLAVE: sistema web, seguro, venta, compra, electrodomésticos.

ABSTRACT

The objective of this project is to improve the sale of home appliances through a secure and user-friendly web system. Users can register, log in, manage their products, leave comments, and report suspicious activities, as well as buy and sell through a chat. The system also includes a payment method and ensures the security of users' information.

Unlike other platforms, this system provides a friendly experience for both customers and sellers, guaranteeing transaction and information security. To achieve these goals, the project is divided into three sections: objectives and scope, development methodology, and activities carried out with results, tests, and conclusions.

The objectives and scope section defines the specific goals of the project and the scope of the functionalities of the web system. The development methodology describes the database modeling, architecture design, and tools used for the backend component. Finally, in the activities carried out section, the results obtained, tests carried out on the system, conclusions, and final recommendations are presented.

In summary, this project seeks to improve the sale of home appliances through a secure and user-friendly web system. The document is divided into three sections to describe the objectives and scope, development methodology, and activities carried out with results obtained.

KEYWORDS: web system, secure, sale, purchase, home appliances.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

En Ecuador las compras y ventas por medio de un sitio web o aplicación móvil ha ido en aumento, solo en 2021, el comercio electrónico aumento consideradamente, lo que equivale a un aumento del 30% a comparación al total de ventas online en 2020, por lo que existe un crecimiento alto, de 250.000 compradores, en el que el 42% son clientes nuevos, según el presidente de la Cámara de Comercio electrónico del Ecuador [1].

La cantidad de personas optan por cambiar, vender o comprar electrodoméstico en diferentes estados ya sea nuevo, seminuevo o aceptable. El comercio electrónico podría ir en aumento en Ecuador, aunque existen muchos factores como por ejemplo la inflación, lo cual aumenta el valor de productos electrónicos según INEC, por esta razón se adquiere productos más baratos, lo cual se adquiere por la venta de segundos consumidores [2].

Desde marzo de 2020, en el inicio de la pandemia a nivel nacional, hasta los inicios del 2022 existió un total de alrededor de 3.5 millones de ciudadanos ecuatorianos que por primera vez realizaron compras o vendieron por línea, esto lo dice según estadísticas de industria de comercio electrónico [3].

Dentro del territorio ecuatoriano existen varias páginas para vender o comprar productos online, en la que se puede llegar a miles de clientes sin invertir mucho dinero, ya que son plataformas digitales consolidados, lo cual han demostrado transparencia y seguridad para realizar compras y ventas, como por ejemplo las plataformas más utilizadas en Ecuador son: MercadoLibre, Facebook Marketplace, Yaesta.com, OLX, entre otras [3].

Este componente de integración curricular tiene como objetivo diseñar el backend de un sistema ecommerce de ventas y compras de electrodomésticos entre usuarios, que brinde una alternativa adicional para realizar transacciones de este tipo a nivel nacional. Para lograrlo, se realizará el diseño, la implementación y la administración del sistema de base de datos, así como la definición de la lógica de negocios. También se crearán APIs (*Application Programming Interface*) para permitir que el frontend del sistema pueda consumirlas y, lo más importante, se garantizará la seguridad de la información en la plataforma.

El resultado esperado es que los usuarios puedan agregar productos para vender y comprar productos existentes en la plataforma. Las opciones para comprar los productos consisten en llegar a un acuerdo con el vendedor a través de un chat interno, mientras que para publicar un producto en venta se deben detallar el tipo de entrega, el tipo de compra y los detalles del producto.

El backend permite almacenar la información de productos en venta o compra, usuarios, comentarios, chats, entre otros en una base de datos relacional, en la tabla de productos se podrá modificar detalles, eliminar y agregar nuevos productos, de igual manera si se requiere que el producto sea más visible en el ecommerce se debe realizar un pago mínimo, por lo que el enfoque de este proyecto es poder brindar un servicio para emprendedores que vendan o compren productos, y ayudar a las necesidades de los ciudadanos, garantizar seguridad en la información y en que los usuarios clientes no sean estafados, por ello existe un usuario administrador el encargado de mantener la plataforma y detectar estafas.

1.1 Objetivo general

Desarrollo backend para un sistema ecommerce *customer to customer* para la venta de electrodomésticos.

1.2 Objetivos específicos

Analizar los requerimientos del sistema ecommerce C2C para la venta de electrodomésticos.

Diseñar la base de datos.

Implantar la base de datos en MySQL.

Configurar las APIS del lado del cliente y servidor.

Implementar seguridad e integridad en la base de datos.

Pruebas unitarias sobre el funcionamiento de las APIS en el Frontend.

1.3 Alcance

La idea presente propone el desarrollo de una plataforma en línea tipo ecommerce en el cual se presenta un sistema de compra y venta de electrodomésticos, por lo mismo se decide implementar la opción en el caso que el usuario tenga la voluntad de vender un producto, este usuario debe colocar fotos, descripción y precio, para que de esta manera el usuario comprador, pueda visualizar este producto y tenga la capacidad de poder comunicarse a través de un chat con el vendedor de dicho producto para posteriormente que puedan concretar la venta.

El usuario en general (comprador/vendedor) tendrá que realizar un registro en la plataforma para poder realizar el acto de compra y venta, caso contrario no este no puede acceder a

todas las funcionalidades dentro del sistema, teniendo la misma información de descripción, fotos y precio de cualquier producto que sea del interés del usuario.

Implementación de suscripciones o micro pagos, con la finalidad de que los usuarios interesados en que sus productos sean mucho más visibles dentro de la plataforma y así que los mismos sean vendidos lo más pronto posible haciendo así una denominada “venta rápida”.

El Ecommerce presenta integridad, consistencia y seguridad de toda la información y datos proporcionados, lo que garantiza que la parte del backend sea efectiva, dinámica y adecuada para los productos que se encuentran en venta o compra, la cual está desarrollada con el framework denominado Laravel. Se garantiza un trabajo lípido en el frontend gracias a la ayuda del *framework* de React, además se realiza pruebas individuales o unitarias y de integración, con el objetivo de comprobar la efectividad del sistema web ecommerce y todas las funcionalidades del mismo.

Con el objetivo de llevar a cabo el actual proyecto, se utiliza una metodología ágil como lo es la metodología denominada “*Scrum*”, lo que ha permitido tener un desarrollo flexible y poder obtener soluciones complejas para la codificación y despliegue del sistema web.

La propuesta plantea la lógica interna de una interfaz la cual es utilizada para que diferentes usuarios que deseen vender o comprar electrodomésticos, además se puede buscar un electrodoméstico por medio de un buscador y filtros para satisfacer las necesidades del usuario. Además, existen 2 perfiles de usuario para que puedan visualizar, comprar y administrar respectivamente en los diferentes módulos dentro de la plataforma, a continuación, una demostración del mismo:

Roles disponibles del *backend*:

- Administrador
- Cliente

Usuario Administrador

- Implementación de métodos de inicio de sesión, cierre de sesión, cambio de contraseña y recuperación de contraseña.
- Implementación de métodos para visualizar y cambiar información del perfil.
- Implementación de métodos para gestionar usuarios registrados.
- Implementación de métodos para gestionar solicitudes de suscripción.

- Implementación de métodos para gestionar la plataforma y reportes.

Usuario Cliente

- Implementación de métodos de inicio de sesión, cierre de sesión, cambio de contraseña y recuperación de contraseña.
- Implementación de métodos para visualizar y cambiar información del perfil.
- Implementación de métodos para gestionar la venta de electrodomésticos individuales por cada uno de los usuarios.
- Implementación de métodos para el chat interno.
- Implementación de métodos para la contratación de mayor visualización de electrométricos.
- Implementación de métodos que permita comentar productos en venta.
- Implementación de métodos para reportar clientes.

1.4 Marco Teórico

A lo largo del tiempo la comercialización ha evolucionado hasta llegar a la actualidad con venta y compra de productos mediante tiendas online, el cual es el objetivo de este proyecto. El ecommerce consumidor, se refiere a que un consumidor que vende productos o servicios a otro consumidor en línea. Esta venta se es facilitada por un tercero la cual se encarga de los detalles de la transacción a cambio de una comisión por reunir a las partes de vendedores directos y clientes potenciales [4].

El propósito del ecommerce es permitir que una de las partes venda directamente a los compradores, con el objetivo de generar más ingresos para el vendedor y no gastar dinero en la creación o mantenimiento de alguna plataforma online, lo cual permite que el vendedor se quede con la mayor parte de las ganancias, pero existe opciones para que existan precios más competitivos para que compitan vendedores en la misma plataforma C2C [4].

Dentro del desarrollo web o programas a medida, el backend está directamente enfocado a los procesos internos necesarios para que funcione la web de forma correcta, además también gestiona la información proporcionada por los usuarios dentro de la plataforma web [5]. En el desarrollo de software unas de las funciones principales del backend es ser

encargado de manejar acciones de lógica, conexión de base de datos y el desarrollo de características que simplifica el proceso de desarrollo.

En el mundo del desarrollo web existe una gran cantidad lenguajes de programación, de igual manera framework del lado del backend, cada una con su pros y contras, de igual manera se puede trabajar con diferentes framework, el lenguaje de programación PHP es muy utilizado la cual se puede trabajar dentro de un framework denominado Laravel, la razón de su utilización es porque es un lenguaje que se adapta mucho al desarrollo de aplicaciones web y porque son tecnologías que tiene un alto estándar respecto al backend.

Para el desarrollo de un sistema web, es importante implementar un sistema que permita administrar los datos en la base de datos, las cual debe incluir todas las relaciones que debe tener cada tabla y poder determinar todos los procesos para almacenar los datos. Al diseñar una base de datos, es importante definir reglas de tablas basadas en conceptos de modelo de base de datos [6].

Las bases de datos relacionales se utilizan porque facilita el acceso a los datos relacionados entre sí, el cual hay que definir antes de relacionar las tablas para acceder de forma directa, cada tabla contiene una llave o un ID, también debe contener una serie de atributos con diferentes tipos de datos, para poder establecer relaciones de tablas según la logia de negocio que se evalúa al principio para el desarrollo del proyecto [7].

Una API (*Aplication Programming Interface*), se define como un conjunto de reglas y herramientas de interacciones entre clientes y servidores, enfocada en la lógica de negocios y posteriormente para poder implementar en una página web [8]. Una Api es también la encargada de presentar la información al cliente, depende de las restricciones que existen internamente para que la base de datos muestre la información.

REST es una arquitectura para el desarrollo web el cual debe contener el protocolo HTTP para poder respaldar la seguridad de los datos que salgan o ingresen al sistema web, además debe estar estructurada por reglas el cual debe cumplir el diseño de la API para ponerlo en producción y no exista inconsistencia en el sistema, es importante conocer que una *API REST* es la que va brindar seguridad de un sistema en la web [9].

PHP es un lenguaje de programación del lado del servidor ampliamente utilizado para el desarrollo de aplicaciones web el cual se lo puede introducir dentro de HTML, además de ser un lenguaje de código abierto.

Existen tres principales áreas donde se utiliza el lenguaje de PHP:

- Scripting del lado del servidor para crear aplicaciones web, es donde se utiliza más el lenguaje, a lo largo del tiempo ha logrado convertirse en uno de los lenguajes más utilizados del lado del servidor.
- Secuencia de comandos de línea código, el cual se lo puede realizar para sin requerir algún servidor o navegador.
- Escritura de aplicaciones de escritorio, aunque no es muy utilizada se lo puede aplicar, pero se debe tener un conocimiento más avanzado respecto al lenguaje.

EL framework Laravel de PHP el cual es muy utilizado ya que es muy fácil de manejar como ayudar a la productividad rápida de los programadores con el objetivo de ahorrar tiempo. Laravel se mantiene en constantes actualizaciones con la web moderna que han incorporado características como por ejemplo la comunicación en tiempo real o notificaciones.

Laravel es un marco de aplicación web el cual tiene una estructura expresiva y elegante, además cuenta con un marco progresivo lo que significa que va creciendo respecto al proyecto, es orientado a convenciones sobre configuraciones, es decir, realiza código para que el desarrollador evite configuraciones [10].

JSON es el formato más utilizado para el intercambio de datos, por lo cual es simple y ligero. *JSON* significa en español notación de objetos de JavaScript, el cual es una solución compacta y legible para humanos a su vez representa una estructura de datos compleja y además facilita el intercambio de datos entre sistemas [11].

Respecto al despliegue de una aplicación web es el proceso de publicar el *API REST* a disposición del público en general ha brindado un servicio con aspectos de seguridad respecto a la información personal, también a la compra y venta de electrodomésticos presentados por los clientes dentro del sistema, con un rendimiento respecto al entorno virtual para que pueda ser consumida.

2 METODOLOGÍA

Los métodos agiles de desarrollo de software tienen como objetivo dividir el proyecto en pequeñas actividades, así que se logra definir tiempos para presentar avances en tiempos establecidos ofreciendo mayor satisfacción al cliente respecto al avance, además de desarrollar un trabajo flexible con el equipo de trabajo [12].

Respecto a lo citado, para el desarrollo del componente del backend del presente proyecto de un sistema de ecommerce *custumer to custumer* para electrodomésticos, se definió establecer la metodología Scrum, la razón es porque permite planificar las tareas establecidas en el desarrollo del proyecto, descomponiéndole por partes y definiendo un tiempo establecido para presentar avances, además de crear un ambiente productivo con el equipo de desarrollo existiendo mayor comunicación, para poder obtener como resultado final un producto de calidad y totalmente funcional en los tiempos establecidos.

Adicionalmente, se podría considerar el uso de metodología de casos como complemento a la metodología Scrum, ya que esta última se enfoca principalmente en la gestión de proyectos y en la entrega de resultados de manera iterativa y en ciclos cortos, por lo que la metodología de casos ayuda a obtener una comprensión más profunda de los usuarios y de los procesos específicos de la industria de ecommerce, lo que a su vez podría mejorar la calidad de las decisiones y el diseño del producto final, además de estudiar los desafíos específicos que enfrenta el sector de venta de electrodomésticos en línea y como otras empresas han resulto esos problemas en el pasado. Entonces se podría ayudar a informar los procesos de toma de decisiones en el proyecto y asegurarse de que el diseño del sistema sea eficaz y sostenible a largo plazo.

2.1 Metodología de Desarrollo

Los métodos de desarrollo se utilizan en el ámbito de la programación, con el objetivo de trabajar en equipo de manera organizada y eficiente, en el transcurso del tiempo ha evolucionado, de tal manera que ha pasado por varios procesos de organización para lograr ser una base importante a la hora de desarrollar software, con un conjunto de técnicas y métodos organizativos [13].

La metodología ágil Scrum la cual permite realizar técnicas de organización para poder desarrollar software respecto a proyectos complejos convirtiéndolos muy flexible para el equipo de desarrollo, la cual se estructura para realizar entregas por cada finalización de tarea y poder resolver irregularidades antes de presentar el producto final que anhela el cliente, donde el equipo debe desarrollar habilidades de innovación, también debe ser un equipo competitivo, flexible y con una alta productividad que son fundamentales para lograr

todos los objetivos planteados dentro del proyecto para lograr cumplir con las fechas establecidas por el equipo de trabajo [14].

Roles

En la metodología Scrum el cual trata de dividir el proyecto en Sprint, la cual es asignado a cada uno de los participantes para el desarrollo del proyecto obteniendo con ello una flexibilidad e inmediatez en el avance del proyecto [15].

Los tres roles principales de Scrum son:

Product Owner (PO): Es el responsable de decidir que trabajo necesita hacerse y poder maximizar el valor del proyecto que se está llevando a cabo, con el objetico de siempre mantener la visión del producto final, al mismo tiempo asegura que cada iteración el producto se ajuste a las necesidades del cliente [16].

Scrum Master: Es el líder del proyecto, el cual tiene como acción ayudar al equipo de desarrollo a cumplir con las reglas y a la organización a usar de mejor manera la metodología Scrum. Su objetivo principal es minimizar las interrupciones que retrasen el proyecto, también trabaja en estrecha colaboración con el *Product Owner* para que el proyecto logre tener un mejor desempeño y manejar todas las tareas previstas [16].

Development Team: La misión es construir el proyecto, con conocimientos técnicos necesarios para desarrollar un proyecto enfocado en las historias que aprueba y se compromete al comienzo de cada sprint, por lo que son los principales responsables de la finalización de cada sprint y del proyecto en su totalidad [16].

En la **TABLA I** se encuentra la distribución de roles para el desarrollo del proyecto el cual consta de 3 tipos de roles, asignado a cada uno de los miembros.

TABLA I: División de roles del proyecto.

ROLES	NOMBRES
Product Owner	Ing. Yadira Franco, MSc.
Scrum Master	Ing. Yadira Franco, MSc.
Development Team	Eduardo Muzo

Artefactos

En la metodología Scrum los artefactos se define como aquellos elementos que garantiza la recopilación de información necesaria para el desarrollo del proyecto y controlando el desempeño del equipo de trabajo, por lo que va a evitar errores durante el desarrollo del proyecto garantizando una buena productividad y calidad en el proyecto [17]. Por esta razón, a continuación, se muestran los artefactos utilizados para desarrollar el proyecto del lado del servidor.

Recopilación de Requerimientos

La recopilación de requerimientos es el proceso de analizar las necesidades del proyecto de software para poder conocer el alcance del proyecto respecto a las exigencias y gestionar las expectativas de las partes interesadas del proyecto, para poder ser analizados y registrados [18]. La **TABLA II** es donde se encuentra la recopilación de requerimientos realizados para el proyecto. La tabla completa se encuentra todos los requerimientos en **ANEXO II** de este documento.

TABLA II: Fragmento de la tabla de recopilación de requerimientos.

RECOPILACIÓN DE REQUERIMIENTOS		
TIPO DEL SISTEMA	**ID - RR**	**ENUNCIADO DEL ÍTEM**
backend	**RR001**	Como cliente nuevo necesito registrar usuarios mediante un formulario.

Historias de Usuario

Las historias de usuarios dentro de un proyecto de software es una explicación general de un usuario el cual consta de funcionalidades por cumplir dentro del sistema, esto depende de cada rol y por esa razón se necesita articular como proporciona la función de software al valor del cliente, o en otras palabras describe el resultado deseado [19].

Dentro de la metodología Scrum las historias de usuarios se añade a los Sprint y se van ejecutando conforme como avance cada sprint, esto conduce a mejorar la estimación y la planificación del proyecto, lo que conduce a tener un resultado final más preciso [19]. En la **TABLA III** se encuentra el esquema que se utilizó al momento de realizar las historias de usuarios, y el resto de las historias de usuarios se localiza en la parte de **ANEXO II**.

TABLA III: Esquema utilizado para las Historias de Usuario.

HISTORIA DE USUARIO	
Identificador: HU001	**Usuario:** Cliente
Nombre de historia: Registrar Usuarios.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Iteración asignada: 1	
Responsable: Eduardo Muzo	
Descripción: El *backend* mediante métodos debe proporcionar la función de registrar, mediante el formulario con los siguientes campos: • Nombre y Apellido. • Nombre usuario. • Numero personal. • Numero de domicilio. • Dirección**.** • Email. • Contraseña.	
Observación: El *backend* valida cada campo y por defecto asigna el rol cliente, y así poder acceder al sistema dependiendo del rol.	

Product Backlog

EL *product backlog* o también conocido en español como pila de producto, es un listado de tareas que se deben ir realizando durante el desarrollo de un proyecto. En la metodología Scrum es una lista con características priorizadas, y que contiene descripciones sobre el resultado establecido en conjunto con el equipo, es importante tener primero un *product backlog* ya que es suficiente para poder iniciar con el primer sprint del proyecto, además se tiene la facilidad de poder crear y cambiar el *product backlog* siempre y cuando sea necesario [20]. En la **TABLA IV**, se muestra el diagrama utilizado para el *Product backlog,* y en **ANEXO II** contiene la tabla completa.

TABLA IV: Fragmento de la tabla de *Product Backlog.*

PRODUCT BACKLOG				
ID–HU	**HISTORIA DE USUARIO**	**ITERACIÓN**	**ESTADO**	**PRIORIDAD**
HU000	Configuración Inicial.	0	Finalizado	Alta
HU001	Registrar Usuarios	1	Finalizado	Alta

Sprint Backlog

Es la suma de toda la lista pendiente del *product backlog*, para luego ser representado a través de un tablero de actividades, haciendo visible para todo el equipo de trabajo, permitiendo observar las tareas donde se dificulta el avance del proyecto, para poder tomar decisiones del problema, haciendo visible el avance de todo el proyecto de software [21]. En la **TABLA V** se visualiza el *sprint backlog*, mientras que en la sección **ANEXO II** se puede visualizar todos los detalles del *sprint backlog* del proyecto.

TABLA V: Fragmento del *Sprint Backlog*.

SPRINT BACKLOG						
ID-SB	**NOMBRE**	**MÓDULO**	**ID-HU**	**HISTORIA DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
SB006	Despliegue del backend	• Despliegue del backend en Heroku				10 horas

2.2 Diseño de la arquitectura

Sobre el diseño de la arquitectura es importante ya que permite brindar la solución al sistema el cual consta de un patrón arquitectónico, en el que se presenta los componentes y su estructura interna que requiere el sistema, además de brindar el tipo de funcionamiento y mantenibilidad que tendrá el sistema una vez finalizado. Por lo tanto, la arquitectura para la implementación del proyecto se muestra a continuación.

Patrón arquitectónico

El patrón arquitectónico implementado en el proyecto es MVC (Vista, modelo, controlador), es un modelo en el que se visualiza la organización y la estructura de los componentes requeridos por el sistema, por qué se utiliza es por la razón que dentro de la ingeniería de software se importante crear procesos que garanticen la calidad de los programas a desarrollar, separando el código para obtener una mejor organización y evitar el concepto de código espagueti en el proyecto, ayudando al desarrollo de aplicaciones con una alto estándar de calidad [22].

- Modelo: Es la capa donde maneja los datos de los usuarios del sistema, por lo tanto, es donde se va a acceder a los datos mediante métodos, la cual se encontrará ubicado en una base de datos, además el proyecto consta con una librería denominada ORM, el cual permite abstraer los datos en objetos, evitando usar sentencias SQL [22].

- Vista: Como su nombre lo dice es la capa donde se encuentra el código del sistema que va a producir la visualización de interfaces de usuarios, además se visualiza datos dependiendo de los requerimientos que el usuario los haga poder realizar el llamado de datos de la BD.

- Controlador: Es la capa donde se encuentran las acciones que vaya a solicitar la aplicación, como la visualización de un elemento en específico, realizar una compra o venta entre otras funcionalidades, adaptando todas las funcionalidades a las necesidades de la aplicación, por lo que es un intermediario entre el modelo y la vista [22].

El patrón arquitectónico se lo presenta en la **Fig. 1** usado para el desarrollo del componente backend, el cual contiene el patrón MVC, permitiendo conocer los controladores que contiene la lógica del negocio, y el puente que existe entre los modelos y vistas del proyecto.

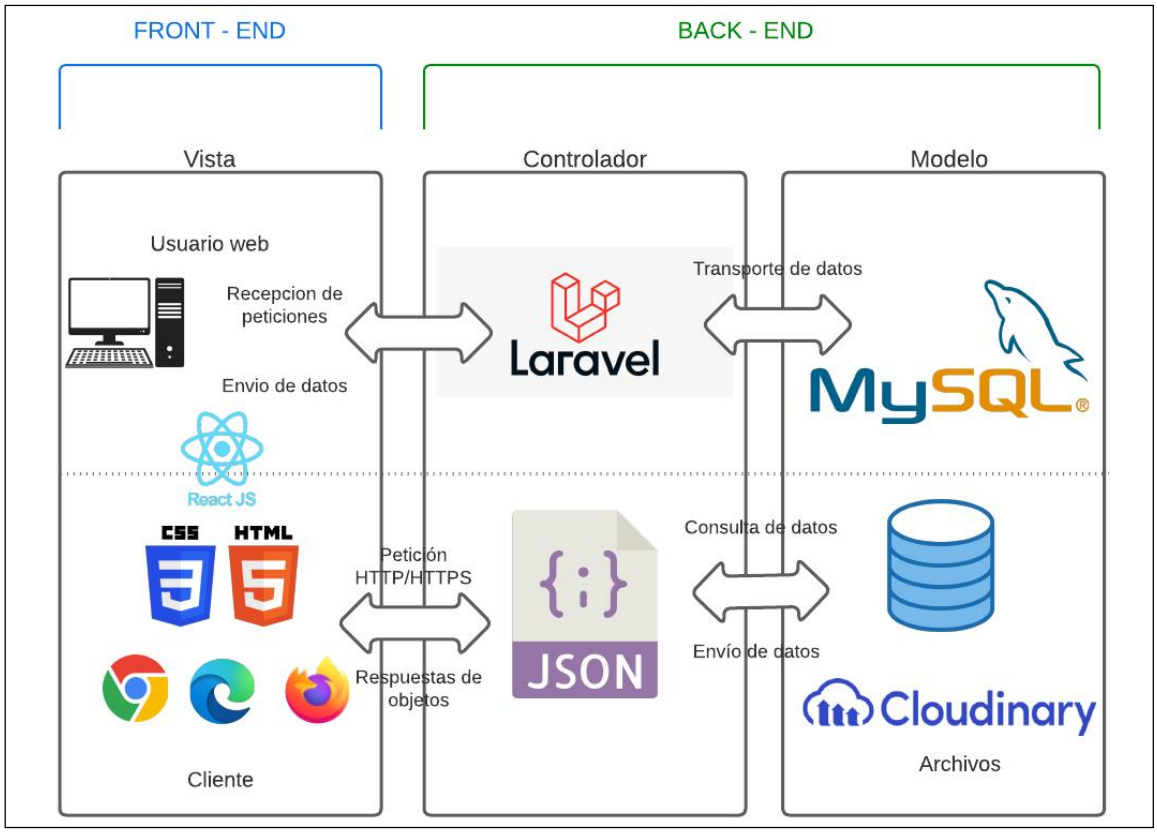


Fig. 1: Arquitectura implementado para el componente backend.

2.3 Herramientas de desarrollo

Una vez determinado el patrón arquitectónico del presente proyecto se lleva a cabo la selección de diferentes herramientas que se utilizaran en el proyecto para el desarrollo de codificación, modelado, diseño, integración y pruebas unitarias del sistema, garantizando un estándar de calidad. La **TABLA VI** a continuación, presenta las herramientas seleccionadas y su respectiva justificación.

TABLA VI: Herramientas seleccionadas para el desarrollo del componente backend.

HERRAMIENTA	JUSTIFICACIÓN
Laravel	Un framework el cual contiene el método ORM para poder acceder a la base de datos sin realizar sentencias SQL, presentando los datos como objetos, esto permite que se tenga un rápido manejo de datos del lado del servidor, por el formato que es muy ligero.

MySQL	Es una base de datos muy reconocida y utilizada, la razón es porque permite trabajar de forma sencilla respecto a las relaciones de tablas, consultas y el almacenamiento de datos, también se puede realizar la visualización completa de las relacionadas de todas las tablas teniendo una expectativa general del manejo de datos.
Visual Studio Code	Es una herramienta para codificar código muy sencillo de utilizar para el desarrollo de proyectos grandes, además lo hace sencillo de manejar porque existen extensiones que se pueden instalar ayudando a realizar código de forma rápida y hacerlo personalizable a nuestros requerimientos.
Postman	Es una herramienta para realizar pruebas y observar las funcionalidades de la API que no contengan fallas; permitiendo realizar consultas *POST, GET, PULL, DELETE*, entre otras, comprobando las necesidades establecidas del proyecto.
Composer	Es un gestor de paquetes que utiliza laravel para poder realizar el llamado de paquetes y librerías de forma rápida y sencilla, dependiendo de las necesidades el proyecto, reduciendo el tiempo y ayudado a enfocarse en los requerimientos del proyecto.
Pusher	Es una herramienta que brinda un servicio en la nube para añadir funcionalidades en tiempo real a las aplicaciones. Trabaja de forma online con un *websockets*, sin necesidad de tener un servidor *WebSockets* propio, con ello se puede crear aplicaciones como chat en tiempo real, juegos multijugadores, envíos de datos en tiempo real, notificaciones, entre otros. [23].
Alwaysdata	Es un host francés el cual puede almacenar una BD de un proyecto en la nube de forma gratuita, teniendo un buen rendimiento y excelente respecto a la seguridad de información [24].
Heroku	Es una herramienta muy utilizada para ayudar a desplegar aplicaciones web, el cual soporta varios lenguajes de programación, logrando con ello almacenar en la nube sin necesidad de enfocarse en el host y si en el desarrollo del proyecto [25].
GitHub	GitHub es una plataforma donde se puede alojar proyectos en la nube usando con ello el sistema de control de versiones GIT, también incluye el método para colaborar con miembros pertenecientes al proyecto.

2.4 Librerías

En la siguiente **TABLA VII** se presenta la lista de las librerías con su respectiva justificación respecto al uso dentro del proyecto desarrollado ayudando a realizar los métodos definidos dentro del proyecto.

TABLA VII: Librerías seleccionadas para el desarrollo del *backend*.

LIBRERÍAS	JUSTIFICACIÓN
Laravel Sanctum	Es una librería que ayuda al proceso de autenticación, generando múltiples *tokens* de la API para el inicio de sesión o para otorgar permisos o para realizar acciones mediante los *tokens*.

3 RESULTADOS

Esta parte del proyecto presenta los resultados obtenidos durante el desarrollo del componente backend el cual va a describir todo el proceso de la implementación de la lógica en el proyecto, además de probar y desplegar el proyecto a producción. Para ello, los resultados se presentan como Sprints.

3.1 Sprint 0. Configuración del ambiente de desarrollo

Como se indica en el *Sprint Backlog*, se realiza la configuración del ambiente de desarrollo para poder comenzar con el proyecto, a continuación, se presentan las actividades del presente sprint:

- Recopilación y levantamiento de requerimientos.
- Estructura del proyecto.
- Diseño de modelo de base de datos en MySQL.
- Identificación de Roles de usuarios.

Recopilación y levantamiento de requerimientos

Implementación de métodos para registro de usuario Cliente.

El componente *backend* permite a los usuarios de perfil cliente registrarse proporcionando información requerida en los campos necesarios. Una vez registrados, los usuarios pueden acceder al sistema.

Implementación de métodos de inicio de sesión, cierre de sesión, cambio de contraseña y recuperación de contraseña.

El componente *backend* implementa métodos de autenticación por *token* para el inicio de sesión de administrador y cliente en el sistema web, también incluye métodos para cerrar sesión, cambiar contraseña y recuperar contraseña por correo electrónico registrado en la base de datos.

Implementación de métodos para visualización y edición del perfil.

El componente *backend* implementa métodos para visualizar y editar información para el perfil administrador y cliente, permitiendo cambios según sus necesidades, y también permite actualizar el avatar.

Implementación de métodos para gestionar ventas de electrodomésticos.

El componente *backend* implementa métodos para que el usuario con rol cliente pueda agregar, editar y eliminar electrodomésticos para su venta en el sistema, ingresando los detalles requeridos y logrando con ello el objetivo de administrar sus propios productos.

Implementación de métodos para buscar y filtrar electrodomésticos en venta.

El componente *backend* implementa métodos para que el usuario con rol cliente le permita buscar electrodomésticos por título y filtrar por distintos campos como precio mínimo y máximo, marca, usuario, categorías, entre otros.

Implementación de métodos para realizar comentarios a electrodomésticos en venta.

El componente *backend* implementa métodos para que el usuario con rol cliente pueda realizar comentarios sobre productos en venta, así como también eliminar y editar sus propios comentarios en el sistema una vez que se encuentre autenticado en el sistema.

Implementación de métodos para reportar electrodomésticos en venta.

El componente *backend* implementa métodos para que el usuario con rol cliente pueda realizar reportes sobre productos en venta, estos reportes pueden ser por motivos de estafa o cualquier problema con el producto o vendedor.

Implementación de métodos para realizar una suscripción.

El componente *backend* implementa métodos para que el usuario con rol cliente pueda realizar una suscripción para mejorar la visibilidad de sus productos en comparación a los de otros clientes que no han pagado. Esto genera ingresos para la plataforma y ayudar a los clientes a ver sus productos más rápido.

Implementación de métodos para crear un chat interno para la compra y venta de electrodomésticos.

El componente *backend* implementa métodos para que el usuario con rol cliente pueda comunicarse entre ellos mediante un chat interno para tratar temas relacionados con la compra o venta de electrodomésticos y llegar a un acuerdo mutuo.

Implementación de métodos para gestionar usuarios registrados.

El componente *backend* implementa métodos para que el usuario con rol administrador pueda gestionar los usuarios registrados en el sistema, con acciones para visualizar la lista de todos los usuarios, y con la acción de deshabilitar cuentas.

Implementación de métodos para gestionar suscripciones.

El componente *backend* implementa métodos para que el usuario con rol administrador pueda manejar las suscripciones recibidas en la plataforma, incluyendo la capacidad de cancelar y aceptar suscripciones.

Implementación de métodos para gestionar reportes de electrodomésticos.

El componente *backend* implementa métodos para que el usuario con rol administrador pueda gestionar los reportes de electrodomésticos recibidos, además la acción de actualizar el estado del mismo.

Implementación de métodos para gestionar comentarios de electrodomésticos.

El componente *backend* implementa métodos para que el usuario con rol administrador pueda gestionar comentarios de electrodomésticos recibidos.

Implementación de métodos para gestionar categorías de electrodomésticos.

El componente *backend* implementa métodos para que el usuario con rol administrador pueda gestionar las categorías y que exista una adecuada organización de electrodomésticos en el sistema.

Implementación de métodos para gestionar categorías de electrodomésticos.

El componente *backend* implementa métodos para que el usuario con rol administrador pueda gestionar las categorías y que exista una adecuada organización de electrodomésticos en el sistema.

Implementación de métodos para gestionar electrodomésticos de perfil cliente.

El componente *backend* implementa métodos para que el usuario con rol administrador pueda ver con detalle y eliminar los electrodomésticos registrados en el sistema.

A continuación, en la **Fig. 2** presenta los métodos divididos para los usuarios existentes en el sistema y como pueden interactuar con el backend.

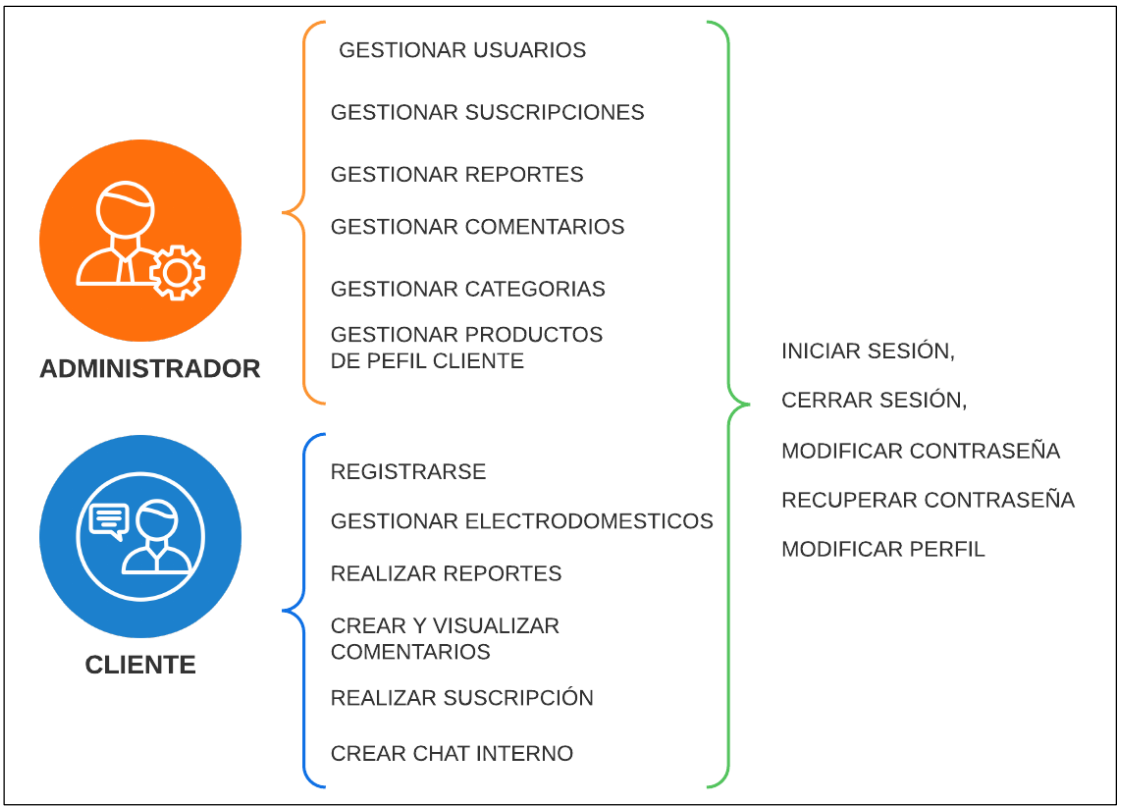


Fig. 2: Funcionalidades designados para los usuarios.

Estructura del proyecto.

Para la estructura del proyecto del componente backend, el cual se lo está realizando en la herramienta de *visual studio code*, además se instaló *composer* el cual es el gestor para las librerías que se utiliza en conjunto con laravel, de igual manera comprobar la conexión del proyecto con MySQL, también se creó un repositorio en GitHub para poder controlar las versiones del proyecto mediante ramas y con estos requerimientos se configuro el ambiente de desarrollo para comenzar con el proyecto. A continuación, en la **Fig. 3** se visualiza la estructura en laravel.

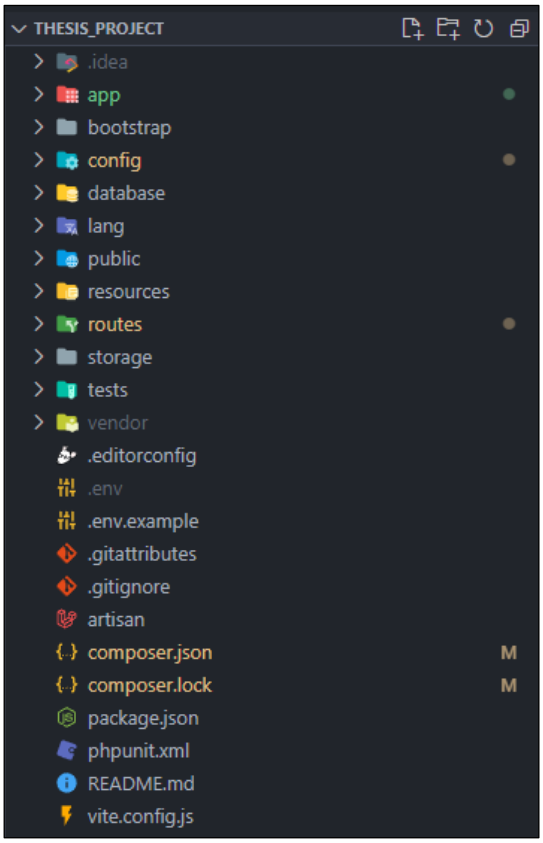


Fig. 3: Estructura para el proyecto en laravel.

Diseño del modelo de base de datos en MySQL.

El modelo para la base de datos mencionado anteriormente es relacional, realizado en MySQL para que toda la información sea integrada mediante relaciones de tabla, un modelo de base de datos se centra en la lógica de negocios, ya que el backend está encargado en realizar peticiones y consultas directamente a la tabla de la base de datos. A continuación, en la **Fig. 4** presenta las tablas implementadas para el diseño de base de datos, mientras que el diagrama lógico se encuentra en **ANEXO II** el cual incluyen todos los atributos de cada tabla con sus claves y las relaciones que existen entre tablas siguiendo las reglas para crear un BD relacional.

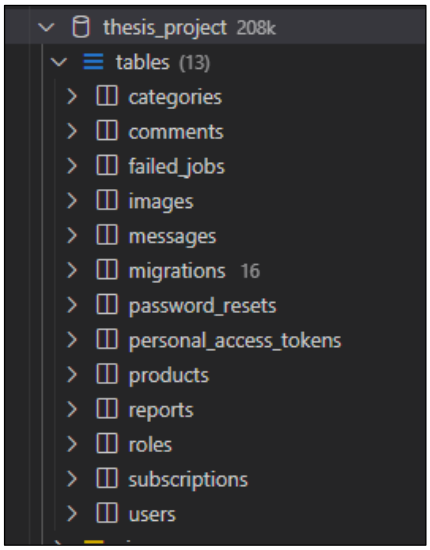


Fig. 4: Tablas de la BD.

Roles de usuarios.

Dentro del sistema existen dos roles, el cliente y administrativo. En la **Fig. 5** muestran los módulos que tienen acceso cada rol en el sistema web.

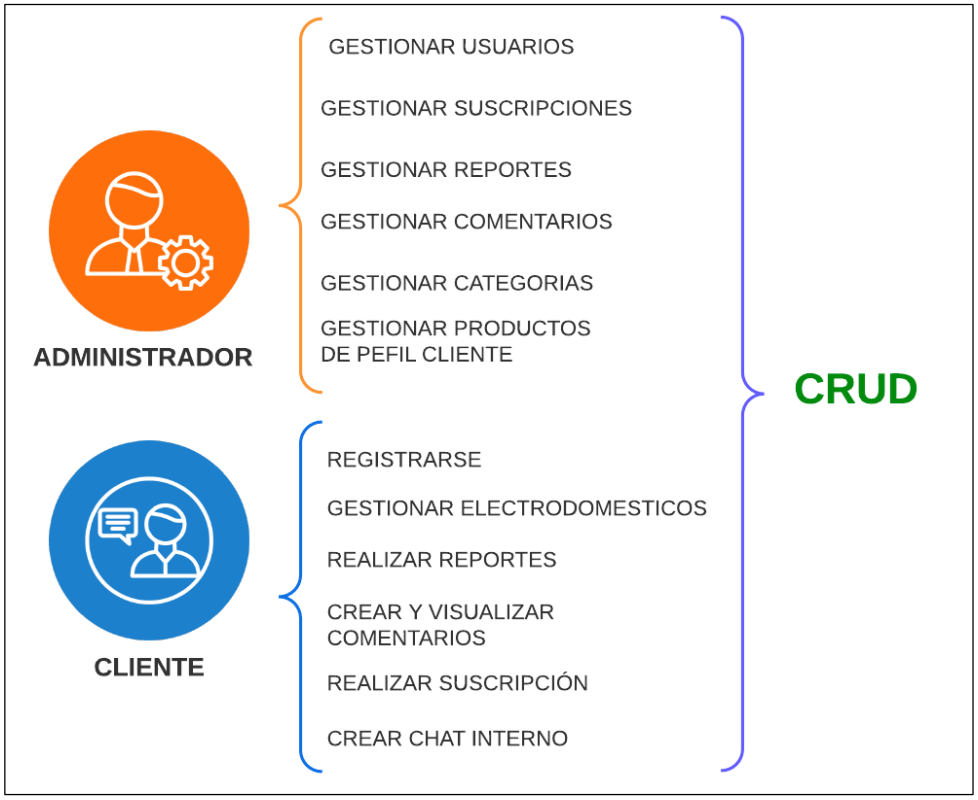


Fig. 5: Roles de usuarios del sistema.

3.2 Sprint 1. Diseño e implementación de métodos para el perfil cliente y administrador.

El presente sprint el cual contiene las siguientes actividades estimulado por el *Sprint Backlog* son las siguientes:

- Tarea 1. Implementar métodos para el registro a través de formularios.
- Tarea 2. Implementar método para visualizar y modificar el perfil.
- Tarea 3. Implementar método para recuperar contraseña.

Tarea 1. Implementar método para el registro a través de formularios.

Para desarrollar el registro de usuarios se implementa varios métodos. El primero consiste en llenar y completar toda la información requerida por el formulario, el cual se encarga de validar el backend respecto a los requerimientos definidos y asignar por defecto el rol cliente. Para completar con el registro, el usuario debe confirmar su correo electrónico enviado al correo proporcionado al registrarse. Esto con el propósito de verificar si el correo electrónico ingresado es válido y, en caso necesario, para la recuperación de la contraseña. La Fig**. 6** muestra los campos requeridos respecto al registro de un nuevo usuario y él envió de información a la base de datos.

```php
public function up()
{
    Schema::create('users', function (Blueprint $table) {
        // ID para la tabla de la BDD
        $table->id();

        // columnas para la tabla BDD
        $table->string('first_name', 50);
        $table->string('last_name', 50);
        $table->string('personal_phone', 10);
        $table->string('address', 50);
        $table->string('password');
        $table->boolean('state')->default(true);

        // columnas que seran unicas para la tabla de la BDD
        $table->string('email')->unique();
        $table->string('username', 50)->unique();

        // columnas que seran podran aceptar regitros null para la tabla de la BDD
        $table->string('home_phone', 9)->nullable();
        $table->timestamp('email_verified_at')->nullable();

        // columnas especiales para la tabla de la BDD
        $table->rememberToken();
        $table->timestamps();
    });
}
```

Fig. 6: Campos requeridos para registrar usuario.

Para el proceso de inicio de sesión, se ha implementado un método que realiza la autenticación mediante la verificación del email y contraseña del usuario. Además, se ha utilizado la librería de *Laravel Sanctum* para generar *tokens* de API que permiten la autenticación en el sistema. La **Fig. 7** ilustra este proceso. Es importante mencionar que el proceso de autenticación es similar para todos los roles del sistema. Por lo tanto, también se ha implementado un método para cerrar sesión, que permite al usuario finalizar la sesión actual, invalidando el *token* anterior. En este caso, el usuario debería iniciar sesión para obtener un nuevo *token*.

En cuanto a los resultados, se logró implementar un método eficiente para el registro de usuarios a través de formulario y verificación de correo electrónico. Los usuarios registrados a través de este método se podrán acceder al sistema de manera segura y eficiente, mejorando la experiencia del usuario.

```php
public function login(Request $request)
{
    // Validación de los datos de entrada
    $request->validate([
        'email' => ['required', 'string', 'email'],
        'password' => ['required', 'string'],
    ]);
    $user = User::where('email', $request['email'])->first();
    if (
        !$user || !$user->state ||
        !Hash::check($request['password'], $user->password)
    ) {
        return $this->sendResponse(message: 'The provided credentials are incorrect.', code: 404);
    }
    if (!$user->tokens->isEmpty()) {
        return $this->sendResponse(message: 'User is already authenticated.', code: 403);
    }
    $token = $user->createToken('auth-token')->plainTextToken;
    return $this->sendResponse(message: 'Successful authentication.', result: [
        'user' => new UserResource($user),
        'access_token' => $token,
        'token_type' => 'Bearer',
    ]);
}
```

Fig. 7: Función para autenticar a un usuario.

Tarea 2. Implementar método para visualizar y modificar el perfil.

Para implementar el método para visualizar y modificar el perfil, se ha creado una ruta específica que permite a los usuarios cliente y administrador acceder a su información de perfil. Para hacer uso de esta ruta, el usuario debe estar autenticado. Utilizando una petición de tipo *GET*, se puede realizar la consulta de la información del perfil. El código correspondiente a esta función se encuentra detallado en **ANEXO III**. Además, una vez que el usuario ha ingresado al sistema, se permite realizar cambios en su perfil mediante una ruta protegida de tipo *POST*. Esta función es común para todos los roles del sistema. La **Fig. 8** ilustra el proceso de modificación, confirmación y actualizando de los datos

ingresados en el perfil. En este proceso, el backend valida los campos y actualiza automáticamente la información en la base de datos.

```php
public function store(Request $request)
{
    $request->validate([
        'username' => ['required', 'string', 'min:5', 'max:20', ValidationRule::unique('users')->ignore
        ($request->user()->id)],
        'first_name' => ['required', 'string', 'min:3', 'max:35'],
        'last_name' => ['required', 'string', 'min:3', 'max:35'],
        'personal_phone' => ['required', 'numeric', 'digits:10'],
        'home_phone' => ['nullable', 'numeric', 'digits:9'],
        'address' => ['required', 'string', 'min:5', 'max:50'],
    ]);
    $user = $request->user();
    $user->update($request->all());
    // Se invoca a la función padre
    return $this->sendResponse('Profile updated successfully');
}
```

Fig. 8: Función para modificar el perfil.

La actualización del avatar es importante dentro de un sistema para distinguirse uno de otros, este método es válido para todos los roles. La función utilizada para la actualización del avatar se muestra en la **Fig. 9**.

```php
public function store(Request $request)
{
    $request->validate([
        'image' => ['required', 'image', 'mimes:jpg,png,jpeg', 'max:512'],
    ]);
    $user = $request->user();
    $file = $request->file('image');
    $obj = Cloudinary::upload($file->getRealPath(), ['folder' => 'avatars']);
    $image_url = $obj->getSecurePath();
    $user->attachImage($image_url);

    return $this->sendResponse('Avatar updated successfully');
}
```

Fig. 9: Función para la actualización del avatar.

Tarea 3. Implementar método para recuperar contraseña.

La tarea 3 en el Sprint incluye la implementación de un método para recuperar la contraseña en caso de que un usuario haya olvidado su contraseña. Para ello, se verifica la validez del correo electrónico ingresado y si este se encuentra registrado en el sistema. Una vez comprobado, se envía un correo electrónico personalizado para confirmar la restauración de la contraseña. Este correo electrónico incluye un *token* que ayuda en el proceso de restauración de la contraseña. La **Fig. 10**. En **ANEXO III** se presenta todas las validaciones correspondientes a los métodos para la restauración de la contraseña por medio del correo electrónico.

```php
public function resendLink(Request $request)
{
    $request->validate([
        'email' => ['required', 'email'],
    ]);
    $status = Password::sendResetLink(
        $request->only('email')
    );
    return $status === Password::RESET_LINK_SENT
        ? $this->sendResponse(__($status))
        : $this->sendResponse(
        message: 'Link reset failure.',
        errors: ['email' => __($status)],
        code: 422
        );
}
```

Fig. 10: Función para restaurar la nueva contraseña.

En relación a la actualización de la contraseña, se valida previamente que el usuario se encuentre autenticado, mediante una ruta estática de tipo *POST*. Es importante mencionar que se requiere validar y confirmar la nueva contraseña antes de realizar la petición La **Fig. 11** y en **ANEXO III** se proporciona detalles adicionales del método para actualizar la contraen y los resultados obtenidos sobre la tarea 3.

```php
public function update(Request $request)
{
    $validated = $request->validate([
        'password' => [
            'required','string','confirmed',
            PasswordValidator::defaults()->mixedCase()->numbers()->symbols()]
    ]);
    $user = $request->user();
    $user->password = Hash::make($validated['password']);
    $user->save();
    return $this->sendResponse('Password updated successfully');
}
```

Fig. 11: Función para actualizar la nueva contraseña.

3.3 Sprint 2. Diseño e implementación de métodos del modelo clientes y electrodomésticos.

En el presente sprint el cual contiene las siguientes actividades estimulado por el *Sprint Backlog* son las siguientes:

- Tarea 1. Implementar método para gestionar electrodomésticos en venta.
- Tarea 2. Implementar método para la búsqueda de electrodomésticos.

- Tarea 3. Implementar método para comentarios y sugerencias de electrodomésticos.
- Tarea 4. Implementar método para reportar electrodomésticos sospechosos.

Tarea 1. Implementar método para gestionar electrodomésticos en venta.

Para implementar el método para gestionar la venta de electrodomésticos, se han implementado un grupo de rutas(*products*) con métodos HTTP, con el objetivo de interactuar con la gestión de electrodomésticos en venta. Estas rutas solo están disponibles para usuarios con el perfil *"customer"*. La **Fig. 12** muestra las rutas definidas.

```php
function () {
    Route::controller(ProductController::class)->group(
        function () {
            Route::get('/', 'index')->name('products.index');
            Route::post('/', 'store')->name('products.store');
            Route::get('/{product}', 'show')->name('products.show');
            Route::post('/{product}/update', 'update')->name('products.update');
            Route::delete('/{product}', 'destroy')->name('products.destroy');
        }
    );
```

Fig. 12: Grupo de rutas para productos.

En el sistema existe dos tipos de rutas GET para visualizar la información de la tabla producto: una función que retorna todos los productos registrados y otra que retorna un producto en específico, recibiendo el parámetro ID del producto. La **Fig. 13** muestra la función para visualizar todos los productos registrados en el sistema.

```php
//Funcion para mostrar todos los productos de la base de datos
0 references | 0 overrides
public function index()
{
    $products = Cache::remember('products_query', 60, function(){
        return Product::all();
    });
        return $this->sendResponse(
    message: "Products returned successfully",
    result: [
        'products' => new ProductCollection($products),
    ]
    );
}
```

Fig. 13: Función para retornar los productos de la BD.

El registro de un nuevo producto, lo pueden realizar únicamente el rol *"Customer"*, el cual consta de un formulario para el registro de información y envío a la base de datos, el cual se muestra en la **Fig. 14**. La función para crear un nuevo electrodoméstico se encarga de validar todos los campos, mientras que la imagen se almacena en la plataforma de Cloudinary el cual retorna una URL de forma inmediata y envía a la tabla *Products* para su almacenamiento.

```php
public function store(Request $request)
{
    $request->validate([
        'title' => 'required|max:50|min:5',
        'price' => 'required|numeric|min:1|max:100000',
        'detail' => 'required',
        'stock' => 'required|numeric|min:1|max:100000',
        'state_appliance' => 'required|max:255',
        'delivery_method' => 'required|max:255',
        'brand' => 'required|max:20|min:3',
        'address' => 'required|max:50|min:5',
        'phone' => 'required|numeric|max:9999999999|min:1000000',
        'categorie_id' => 'required|exists:categories,id',
        'image' => 'required|image'
    ]);
    $file = $request->file('image');
    $obj = Cloudinary::upload($file->getRealPath(), ['folder' => 'products']);
    $image_url = $obj->getSecurePath();
    $product = Product::create([
        'title' => $request->title,
        'price' => $request->price,
        'detail' => $request->detail,
        'stock' => $request->stock,
        'state_appliance' => $request->state_appliance,
        'delivery_method' => $request->delivery_method,
        'brand' => $request->brand,
        'address' => $request->address,
        'phone' => $request->phone,
        'categorie_id' => $request->categorie_id,
        'image' => $image_url,
    ]);
    return $this->sendResponse(
    message: 'Product created successfully',
    code: 201,
    result: [
            'product' => new ProductResource($product),
```

Fig. 14: Función para crear un nuevo electrodoméstico**.**

Las validaciones son las mismas para la función de crear un nuevo electrodoméstico y la función de actualizar electrodomésticos. Además, el backend verifica si el producto pertenece al usuario autenticado. El usuario autenticado tiene la acción de eliminar únicamente sus productos registrados en el sistema, pero el backend únicamente cambia el estado de visualización para los usuarios con perfil cliente, pero sigue visible para el administrador. Todos los detalles sobre los métodos para actualizar observar y eliminar los electrodomésticos se encuentran en **ANEXO III**.

Tarea 2. Implementar método para la búsqueda de electrodomésticos.

Para realizar la búsqueda y filtrado de electrodomésticos, es necesario que el usuario se encuentre autenticado en el sistema. La plataforma cuenta con rutas específicas de método *GET*, que permiten realizar la búsqueda y filtrado de los electrodomésticos de forma segura y eficiente. Estas rutas se aprecia en la **Fig. 15**.

```
Route::get('search', [ProductController::class, 'search'])->name('products.search');
Route::get('filter/products', [ProductController::class, 'filter'])->name('products.filter');
```

Fig. 15: Rutas privadas de búsqueda y filtrado**.**

Para realizar la búsqueda de electrodomésticos, utiliza un parámetro de tipo texto para buscar dentro de la comuna "*title*" de la base de datos y retornar todos los productos con ese nombre. Esto se visualiza en la **Fig. 16**.

Además, se ofrece la posibilidad de realizar búsquedas mediante filtros, recibiendo parámetros que se verifica para realiza la búsqueda interna en las columnas correspondientes. Todos los detalles sobre el método de búsqueda y filtrado de electrodomésticos se encuentra en **ANEXO III**.

```
//Funcion para buscar un producto
0 references | 0 overrides
public function search(Request $request)
{
    $product = Product::active()->where('title', 'like', '%' . $request->title . '%')->get();
    return $this->sendResponse(
    message: "Product returned successfully",
    result: [
            'product' => ProductResource::collection($product),
        ]
    );
}
```

Fig. 16: Función para buscar un producto.

Tarea 3. Implementar método para comentarios y sugerencias de electrodomésticos.

Los usuarios con perfil "customer" pueden realizar comentarios en el sistema, pero primero deben autenticarse. El acceso a las funciones de comentarios se realiza mediante el grupo de rutas llamado *"Comment",* como se puede evidenciar en la **Fig. 17**.

```php
Route::controller(CommentController::class)->group(
    function () {
            Route::get('/{product}/comments', 'index')->name('products.comments.index');
            Route::get('/{product}/comments/{comment}', 'show')->name('products.comments.show');
            Route::post('/{product}/comments', 'store')->name('products.comments.store');
            Route::put('/{product}/comments/{comment}', 'update')->name('products.comments.update');
            Route::delete('/{product}/comments/{comment}', 'destroy')->name('products.comments.destroy');
        }
);
```

Fig. 17: Rutas privadas para Comentarios.

Para poder visualizar los comentarios de un electrodoméstico en específico, los usuarios con perfil *"customer"* pueden acceder a las funciones de tipo GET en el grupo de rutas llamado *"Comment".* La primera función recibe el parámetro del ID del producto y retorna todos los comentarios presentes en ese producto. Por otro lado, la segunda función también recibe el parámetro del ID del producto, pero además recibe el ID del comentario específico y retorna solo ese comentario, como se observa en la **Fig. 18**.

```php
//Funcion para mostrar todos los comentarios  de un producto
0 references | 0 overrides
public function index(Product $product)
{
    //Se retorna la coleccion de comentarios del producto
    return $this->sendResponse(
        message: "Comments returned successfully",
        code: 200,
        result: [
        'comments' => new CommentCollection(Product::find($product->id)->comments),
        ]
    );
}
//Funcion para mostrar un comentario de un producto en especifico
0 references | 0 overrides
public function show(Product $product, Comment $comment)
{
    $comment = $product->comments()->where('id', $comment->id)->firstOrFail();
    return $this->sendResponse(
        message: "Comment returned successfully",
        code: 200,
        result: [
            'comment' => new CommentResource($comment)
        ]
    );
}
```

Fig. 18: Funciones para la visualización de comentarios.

Para generar y actualizar comentarios a un producto en específico, tanto las funciones de crear y actualizar tienen las mismas validaciones de campos y además de comprobar que el usuario autenticado le pertenezca el comentario para editar, la función para generar un nuevo comentario se encuentra en la siguiente **Fig. 19** y todos los detalles sobre el método se encuentra en la sección de **ANEXO III**.

```php
//Funcion para crear un comentario
// Se recibe como parametro el id del producto y los datos del comentario
0 references | 0 overrides
public function store(Request $request, Product $product)
{
        $request->validate([
            'comment' => 'required|string'
        ]);

        $comment = $product->comments()->save(new Comment($request->all()));
        return $this->sendResponse(
            message: "Comment created successfully",
            code: 201,
            result: [
                'comment' => new CommentResource($comment)
            ]
        );
}
```

Fig. 19: Función para crear comentarios.

El usuario que se encuentra autenticado, es el único capaz de eliminar sus comentarios, en la **Fig. 20** se observa la función de eliminar un comentario, el cual verifica que el comentario pertenezca al mismo usuario autenticado en el sistema.

```php
//Funcion para eliminar un comentario
// Se recibe como parametro el id del producto y el id del comentario
0 references | 0 overrides
public function destroy(Product $product, Comment $comment)
{
    $comment = $product->comments()->where('id', $comment->id)->firstOrFail();
    $comment->delete();
    return $this->sendResponse(
        message: "Comment deleted successfully",
        code: 200,
        result: []
    );
}
```

Fig. 20: Función para eliminar un comentario.

Tarea 4. Implementar método para reportar electrodomésticos sospechosos.

Los usuarios con perfil *"customer"* pueden reportar productos sospechosos en el sistema. Para realizar este reporte, deben autenticarse y acceder a la ruta de tipo *POST* llamada *"Report",* según se muestra en la **Fig. 21**. La función para crear un reporte recibe como parámetro el ID del producto sospechoso, y luego se envía el reporte para su revisión por

parte del administrador. La función también se encarga de realizar la validación de los campos del formulario de reporte. En **ANEXO III** se detalla el método para el reporte de electrodomésticos.

```php
//Funcion para crear un reporte
// Se recibe como parametro el id del producto y los datos del reporte
0 references | 0 overrides
public function store(Request $request, Product $product)
{
    $this->authorize('create', Report::class);

    $request->validate([
        'title' => 'required|string',
        'description' => 'required|string',
    ]);
    $report = $product->reports()->save(new Report($request->all()));
    return $this->sendResponse(
    message: "Report created successfully",
    result: [
            'report' => new ReportResource($report)
        ]
    );
}
```

Fig. 21: Función para crear un reporte**.**

3.4 Sprint 3. Diseño e implementación de métodos suscripción y chat interno.

El Sprint 3 se considera el más importante, ya que los objetivos corresponden a las funciones de ventas y generar recursos a la plataforma, a continuación, se presenta las funciones establecidas por el *Sprint Backlog*.

- Tarea 1. Implementar método para solicitar suscripción.
- Tarea 2. Implementar método para interactuar mediante un chat interno.

Tarea 1. Implementar método para solicitar suscripción.

El sistema permite a los clientes solicitar una suscripción para colocar un producto registrado en destacados, lo cual aumenta la visualización del mismo. La **Fig. 22** muestra la función para realizar una suscripción, donde se debe especificar el ID del producto al que se desea destacar. Antes de realizar suscripción, el sistema verifica que el usuario autenticado sea el propietario del producto, que el producto no se encuentre ya en destacados y valida todos los datos de entrada necesarios para realizar el pago correspondiente.

```php
public function store(Request $request)
{
    if (auth()->id() != Product::find($request->product_id)->user_id) {
        return $this->sendResponse(
        message: 'You are not allowed to subscribe this product',
        code: 403,
        result: null
        );
    }
    $request->validate([
        'product_id' => [
            'required',
            'exists:products,id',
            function ($attribute, $value, $fail) {
                $subscription = Subscription::where('product_id', $value)->where('status', 'active')->first();
                if ($subscription) {
                    $fail('This product is already featured');
                }
            }
        ]
    ]);
    $subscription = Subscription::create([
        'user_id' => auth()->id(),
        'product_id' => $request->product_id,
        'status' => 'active',
        'start_date' => now(),
        'end_date' => now()->addMonth(),
        'payment_method' => 'paypal',
        'price' => 4.99,
    ]);
    $subscription->product->update([
        'featured' => true,
    ]);
```

Fig. 22: Función para crear una suscripción.

La **Fig. 23** muestra la función para ver las solicitudes de suscripción del usuario autenticado. Para ello, la función debe verificar si la petición pertenece al usuario y retornar información sobre todas las suscripciones registradas.

```php
// Funcion para obtener las suscripciones del usuario autenticado
0 references | 0 overrides
public function index()
{
    $products = Product::where('user_id', auth()->id())->where('state', 1)->get();
    $subscriptions = Subscription::whereIn('product_id', $products->pluck('id'))->get();
    return $this->sendResponse(
    message: "Subscriptions returned successfully",
    result: [
            'subscriptions' => SubscriptionResource::collection($subscriptions),
        ]
    );
}
```

Fig. 23: Función para mostrar la solicitud de la suscripción.

Tarea 2. Implementar método para interactuar mediante un chat interno.

Para iniciar una conversación mediante un chat interno, el usuario con el perfil de cliente utiliza el método presentado en la **Fig. 24**. La función permite enviar mensajes a otro usuario con el objetivo de llegar a un acuerdo sobre la compra o venta de un

electrodoméstico en particular. Se requiere del ID del cliente y toda la conversación es registrada en la base de datos para su posterior visualización en una ruta específica. La herramienta de Pusher fue implementada para lograr una mensajería en tiempo real y tener una comunicación compleja de manera rápida.

```php
public function sendMessage(Request $request)
{
    $this->authorize('create', Message::class);
    $message = Message::create([
        'to' => $request->to,
        'message' => $request->message
    ]);
    event(new MessageSent($request->to, $request->message));
    return $this->sendResponse(
        message: 'Message sent',
        code: 200);
}
```

Fig. 24: Función para enviar mensajes.

Sólo los usuarios que hayan iniciado una conversación podrán verla y los contactos, también existen rutas privadas de tipo *GET* y *POST* para ello. Todos los detalles relacionados con los métodos del chat se encuentran en el **ANEXO** III.

3.5 Sprint 4. Diseño e implementación de métodos para el perfil administrador.

- Tarea 1. Implementar métodos para gestionar clientes.
- Tarea 2: Implementar métodos para gestionar suscripciones.
- Tarea 3: Implementar métodos para gestionar reportes de electrodomésticos.
- Tarea 4: Implementar métodos para gestionar comentarios.
- Tarea 5. Implementar métodos para gestionar categorías.
- Tarea 6. Implementar métodos para gestionar electrodomésticos de perfil cliente.

Tarea 1. Implementar métodos para gestionar clientes

El administrador tiene la capacidad de visualizar todos los usuarios registrados en el sistema, así como también puede ver un usuario específico y tener permisos para eliminarlo. Para acceder a cada uno de los métodos, existe un grupo de rutas *GET* y *DELETE*, como se muestra en la **Fig. 25**.

```php
// Se hace uso de grupo de rutas para los clientes
Route::controller(CustomerController::class)->group(
    function () {
        Route::get('customers/', 'index')->name('customers.index');
        Route::get('customers/{customer}', 'show')->name('customers.show');
        Route::delete('customers/{customer}', 'destroy')->name('customers.destroy');
    }
);
```

Fig. 25: Grupo de rutas para gestionar los clientes.

El administrador tiene la capacidad de visualizar la lista de todos los usuarios registrados en el sistema, accediendo a una ruta *GET* que retorna información sobre cada uno de los perfiles. Para consultar los detalles de un usuario en particular, se especifica su ID como parámetro y se accede a una ruta *GET*, tal y como se muestra en la **Fig. 26**. Además, el administrador cuenta con los permisos para eliminar a cualquier cliente o perfil con rol 1, lo que implica la eliminación de todos sus datos almacenados en el sistema. Toda la funcionalidad relacionada con la gestión de perfiles se encuentra detallada en el **ANEXO III**.

```php
public function index()
{
    $customers = User::where('role_id', 2)->get();
    return $this->sendResponse(
        message: "Customers returned successfully",
        result: [
            'customers' => $customers,
        ]
    );
}
0 references | 0 overrides
public function show($customer)
{
    $customer = User::find($customer);
    return  $this->sendResponse(
        message: "Customer returned successfully",
        result: [
            'customer' => $customer,
        ]
    );
}
```

Fig. 26: Funciones para mostrar los clientes registrados.

Tarea 2. Implementar métodos para gestionar suscripciones.

El sistema permite a los administradores gestionar las suscripciones de los clientes. Para esto, se han establecido un conjunto de rutas específicas que solo están disponibles para los administradores, como se muestra en la **Fig. 27.** Estas rutas permiten ver todas las solicitudes recibidas, aceptarlas, cancelarlas y eliminarlas. Solo los usuarios con el rol de administrador tienen permiso para acceder a estas rutas, mientras que los clientes solo pueden realizar solicitudes. Para obtener más información sobre la gestión de solicitudes de suscripciones, consulte el **ANEXO III**

```php
Route::controller(AdminSubscriptionController::class)->group(
    function () {
        Route::get('subscriptions/', 'index')->name('admin.subscriptions.index');
        Route::get('subscriptions/{subscription}', 'show')->name('admin.subscriptions.show');
        Route::post('subscriptions/{subscription}', 'acceptSubscription')->name('admin.subscriptions.
        acceptSubscription');
        Route::post('subscriptions/{subscription}/cancel', 'cancelSubscription')->name('admin.subscriptions.
        cancelSubscription');
        Route::delete('subscriptions/{subscription}', 'destroy')->name('admin.subscriptions.destroy');
    }
);
```

Fig. 27: Grupo de rutas para gestionar suscripciones.

Para visualizar las solicitudes se requiere de una función para retorna las solicitudes registradas en el sistema, mientras que para observar los detalles de una suscripción la función recibe el parámetro ID de la solicitud, las funciones se muestran en la **Fig. 28**.

```php
// Funcion para ver todos las suscripciones registrados en la base de datos
0 references | 0 overrides
public function index()
{
    $subscriptions = Subscription::all();
    return $this->sendResponse(
        message: "Subscriptions returned successfully",
        code: 200,
        result: [
            'subscriptions' => $subscriptions,
        ]
    );
}
// Funcion para ver una suscripcion en especifico
0 references | 0 overrides
public function show(Subscription $subscription)
{
    return $this->sendResponse(
        message: "Subscription returned successfully",
        code: 200,
        result: [
            'subscription' => $subscription,
        ]
    );
}
```

Fig. 28: Función para mostrar las solicitudes de suscripción.

Además, el administrador tiene la opción de cancelar una suscripción en caso de un problema o fraude y de activar una suscripción si se resuelve el problema. Toda la información sobre la gestión de suscripciones se encuentra en el **ANEXO III**.

Tarea 3. Implementar métodos para gestionar reportes de electrodomésticos.

En el sistema se ha implementado un grupo de rutas para la gestión de reportes. El sistema permite a la administración recibir y gestionar todos los reportes emitidos por los clientes sobre productos que intenta dañar la integridad de la plataforma. Para ello, se ha implementado métodos que permiten a la administración verificar la veracidad de las solicitudes y tomar acciones en consecuencia en caso de detectarse.

El administrador puede acceder a una ruta que muestra todos los reportes existentes, la función se presenta en la **Fig. 29**, así como también puede ver el detalle de cada uno de ellos y tomar acciones apropiadas. Además, en **ANEXO III** se encuentra detallada la funcionalidad completa relacionada con la gestión de reportes de electrodomésticos en el sistema.

```php
// Funcion para mostrar todos los reportes registrados en la base de datos
0 references | 0 overrides
public function index()
{
    $reports = Report::all();
    return $this->sendResponse(
        message: "Reports returned successfully",
        code: 200,
        result: [
            'reports' => $reports,
        ]
    );
}
```

Fig. 29: Función para listar los reportes en Administración.

Tarea 4. Implementar métodos para gestionar comentarios.

En el sistema se ha implementado un grupo de rutas para la gestión de comentarios. El administrador tiene la responsabilidad de gestionar los comentarios emitidos por los clientes en la plataforma. Para ello, se incluyen métodos que permiten al administrador ver detalles de cada comentario y eliminar aquellos que se consideren inapropiados o dañinos a la integridad de la plataforma. En la **Fig. 30** se muestra la funcionalidad para ver todos los comentarios existentes y en **ANEXO III** se detallan todos los métodos correspondientes a la gestión de comentarios.

```php
//Funcion para mostrar todos los comentarios registrados en la base de datos
0 references | 0 overrides
public function index()
{
    $this->authorize('viewAll', Comment::class);

    return $this->sendResponse(
        message: "Comments returned successfully",
        code: 200,
        result: [
            'comments' => Comment::all(),
        ]
    );
}
```

Fig. 30: Función para listar todos los comentarios en el sistema.

Tarea 5. Implementar métodos para gestionar categorías.

Para la gestión de categorías en la plataforma, se ha implementado un grupo de rutas específicas para el control de las mismas, tal como se muestra en la **Fig. 31**. Estas rutas brindan la posibilidad de crear nuevas categorías, visualizar todas las categorías existentes, obtener detalles de una categoría en particular, actualizarla y eliminarla. Es importante mencionar que la tabla de categorías debe estar en constante actualización y su administración está restringida únicamente al rol número 1.

```php
// Se hace uso de grupo de rutas para las categorias
Route::controller(CategorieController::class)->group(
    function () {
        Route::get('categories/', 'index')->name('categories.index');
        Route::post('categories/', 'store')->name('categories.store');
        Route::get('categories/{categorie}', 'show')->name('categories.show');
        Route::put('categories/{categorie}', 'update')->name('categories.update');
        Route::delete('categories/{categorie}', 'destroy')->name('categories.destroy');
    }
);
```

Fig. 31: Grupo de rutas para gestionar las categorías.

Antes de registrar una nueva categoría, se verifica si ya existe el nombre en la base de datos. Luego, se validan todos los campos, los cuales deben cumplir con ciertos requisitos. Las mismas validaciones se realizan al momento de actualizar una categoría. Si la categoría se registra o actualiza exitosamente, se retornan sus detalles.

Para eliminar una categoría, la función primero verifica si el usuario autenticado tiene la autorización correspondiente. Para llevar a cabo la eliminación, se debe recibir un parámetro ID de la categoría existente en la base de datos, como se muestra en la **Fig. 32**.

Toda la información detallada sobre los métodos implementados para la gestión de la plataforma se encuentra en el **ANEXO III**.

```php
//Funcion para eliminar una categoria en la base de datos
0 references | 0 overrides
public function destroy(Categorie $categorie)
{
    //Se elimina la categoria
    $categorie->delete();

    //Se retorna la categoria eliminada
    return $this->sendResponse(
        message: "Category deleted successfully",
        code: 200,
        result: []
    );
}
```

Fig. 32: Función para eliminar una categoría.

Tarea 6. Implementar métodos para gestionar electrodomésticos de perfil cliente.

El administrador tiene la tarea de gestionar los electrodomésticos de los perfiles de los clientes, para ello cuenta con diversos métodos para realizar esta tarea de manera eficiente y segura. Algunas de las acciones que puede realizar son: ver detalles de los electrodomésticos registrados en la plataforma la función se encuentra en la **Fig. 33** y eliminar un electrodoméstico específico en caso de ser necesario de manera definitiva. Todas estas acciones se llevan a cabo a través de un grupo de rutas específicas para la gestión de electrodomésticos, las cuales están detalladas en **ANEXO III**.

```php
// Funcion para ver todos los productos registrados en la base de datos
0 references | 0 overrides
public function index()
{
    $products = Product::all();
    return $this->sendResponse(
        message: "Products returned successfully",
        code: 200,
        result: [
            'products' => ProductResource::collection($products),
        ]
    );

}
```

Fig. 33: Función para listar todos los productos registrados en el sistema.

3.6 Sprint 5. Pruebas del backend.

- Elaboración de pruebas unitarias de rutas y resultados.
- Elaboración de pruebas de estrés y resultado.
- Elaboración de pruebas de carga y resultado.

Elaboración de pruebas unitarias de rutas y resultados.

Las pruebas unitarias son scripts que se realiza con el objetico de comprobar el correcto funcionamiento de cada una de las rutas, además sirve para mostrar posibles errores y corregirlos. En el framework Laravel incluye por defecto la herramienta necesaria para realizar los test el cual trabaja con *PHPUnit*, y las pruebas se realiza dentro de la carpeta denominada *test* el cual contiene dos subcarpetas denominas *Feature* y *Unit*. En la primera carpeta *Feature* re realiza pruebas complejas, mientras que en *Unit* son pruebas de un fragmento de código en específico [26].

La **Fig. 34** muestra el script para probar la ruta de registro para el ingreso de un nuevo usuario con todos los campos solicitados por el controlador, por lo tanto, si la ruta no tiene ningún error o inconveniente al ingresar los datos, la prueba debe retornar un mensaje de "*OK*" e incluye el tiempo que tomo registrar y el tamaño del almacenamiento de un nuevo registro, como se presenta en la **Fig. 35.**

```php
/** @test register */
0 references | 0 overrides
public function test_register()
{
    $response = $this->post('http://127.0.0.1:8000/api/register', [
        'username' => 'Test 1_1',
        'first_name' => 'Test 1',
        'last_name' => 'Test 1',
        'email' => 'test_1@example.com',
        'home_phone' => '1234567',
        'personal_phone' => '1234567890',
        'address' => 'Test 1',
        'password' => 'Test.123456',
        'password_confirmation' => 'Test.123456',
    ]);
    $response->assertStatus(201);
}
```

Fig. 34: Prueba unitaria para el registro de un usuario.

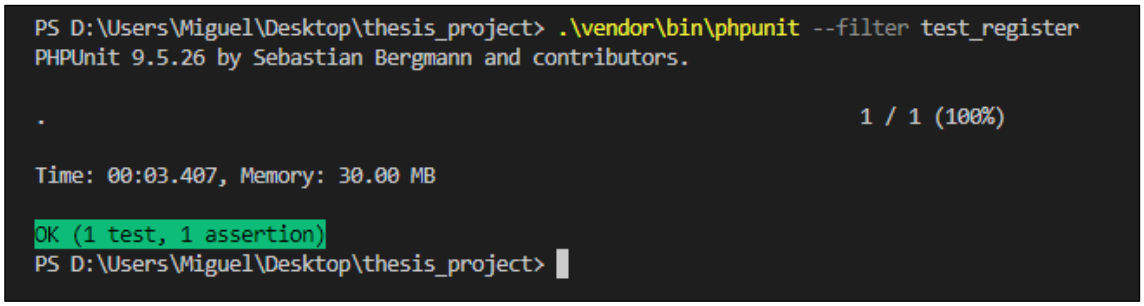


Fig. 35: Resultado de la prueba unitaria del registro de usuario.

En la **TABLA VIII** se presenta los resultados de las pruebas unitarias sobre el registro de un nuevo usuario para verificar el correcto funcionamiento de la ruta. La primera prueba, con todos los campos completos solicitados por el backend, en el que se obtuvo como resultado "*OK*" y con un tiempo de 3 segundos en ejecutarse, con un almacenamiento de 30.00MB, mientras que la segunda prueba realizada a la misma ruta, con campos faltantes, tuvo un resultado "*ERROR*" y tardo en ejecutarse 00.15 segundos.

TABLA VIII: Resultados de la prueba unitaria de la HU001.

Prueba	Resultado	Tiempo de ejecución	Tamaño de almacenamiento
Test de registro con todos los campos	OK	3 segundos	30.00 MB.
Test de registro con campos faltantes	Error	00.15 segundos	-

De esta manera, al realizar pruebas unitarias por cada una de las rutas del proyecto, puedo asegurar de que todas funcionan correctamente antes de su desplegué, lo que me permite detectar y corregir problemas de manera temprana, lo que reduce el tiempo y los costos de mantenimiento. En la sección de **ANEXO II** se puede evidencias todas las pruebas unitarias de rutas y sus resultados correspondientes.

Elaboración de pruebas de estrés y resultado.

La elaboración de pruebas de estrés de la *API REST* es realizar pruebas de carga para poder determinar los límites del sistema mediante varias peticiones por segundo, por lo que se tiene como objetivo verificar la compatibilidad y fiabilidad del sistema en condiciones extremas, obteniendo como resultado el tiempo de respuesta, latencia, carga y otros parámetros de resultados de la prueba de estrés [27] .

En la siguiente **TABLA IX** se puede apreciar los resultados obtenidos y en la **Fig. 36** se muestra el script de la información que se requiere para crear un nuevo usuario, mientras que en **ANEXO II** se encuentra todas las pruebas de estrés con sus respectivas respuestas.

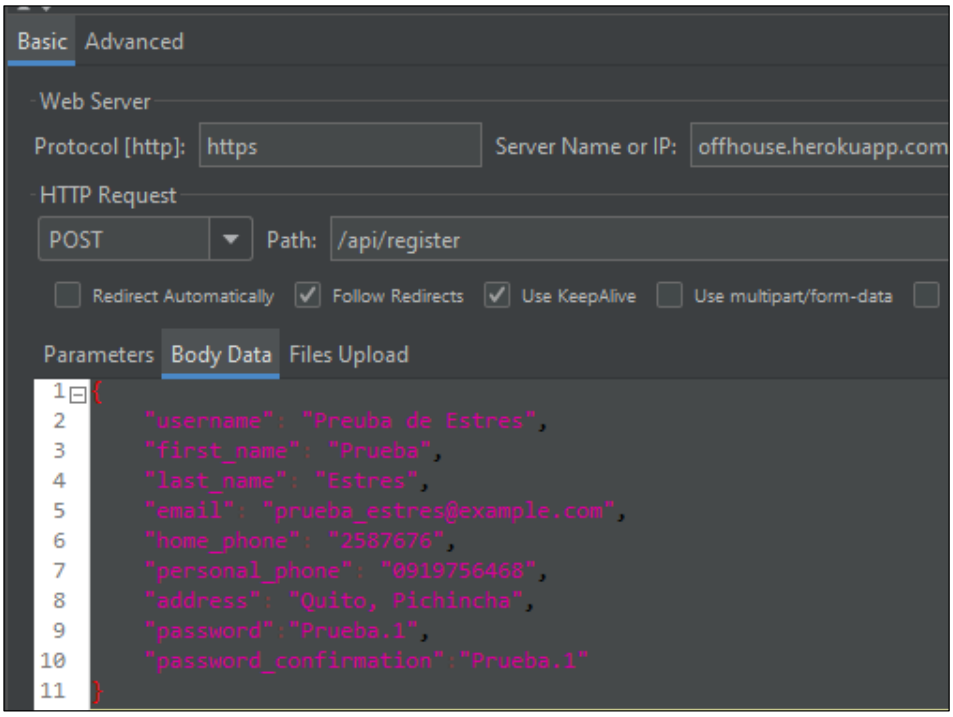


Fig. 36: Prueba de estrés para el registro de usuarios.

Las pruebas de estrés realizadas con apache JMeter, donde los resultados obtenidos muestran un rendimiento considerable de la API REST, con un aumento del doble por prueba. El tiempo promedio de respuesta se mantiene alrededor de 7 segundos, con una respuesta mínima de 2.4 segundos y un máximo de 10.1 segundos. Respecto al porcentaje de error es bueno ya que se mantiene en cero en las primeras pruebas, sin embargo, en el caso de 200 hilos se observa un error del 56 %, mientras que la tasa de error se mantiene en torno a 3.6 - 3.8 peticiones por segundo.

TABLA IX: Resultados de la prueba de estrés de la HU001**.**

Hilos	Periodo de aumento	Promedio	Mínimo	Máximo	Porcentaje de error	Tasa de transferencia
50	10 s	7004 ms	2398 ms	10112 ms	0.00%	3.6/sec
100	10 s	17192 ms	3807 ms	22401 ms	0.00 %	3.7/sec

200	10 s	37387 ms	7233 ms	47777 ms	56.00 %	3.8/sec

Entonces con los resultados obtenidos se puede interpretar que las pruebas de estrés indica que el sistema puede manejar un alto número de peticiones simultáneamente con un tiempo de respuesta aceptable y una tasa de error baja, mientras que en el caso de los 200 hilos se requiere una optimización adicional ya que se presenta un porcentaje de error alto, pero se observa que el sistema admite un numero alto de peticiones.

Elaboración de pruebas de carga y resultado.

La elaboración de pruebas de carga es importante para garantizar la escalabilidad y el rendimiento de la *API REST* en condiciones con una alta concurrencia de información. Con las pruebas de carga, se puede simular diferentes escenarios de uso para medir y evaluar el rendimiento de la API bajo diferentes escenarios y durante un periodo de tiempo en específico, además se puede obtener como resultado el rendimiento de la API y utilizar para evaluar y poder evitar cuellos de botellas y mejorar el redimiendo en áreas críticas.

En la **Fig. 37** se muestra el resultado obtenido sobre las pruebas de carga, para lo cual en el script se definió un escenario con un total de 100 usuarios ejecutando simultáneamente para las rutas de registro e inicio de sesión, durante un periodo de 60 segundos. Los resultados obtenidos es un total de 456 peticiones en 5 segundos, un total de 108 interacciones por segundo, obteniendo una tasa de éxito del 100%. Para visualizar el resto de pruebas de carga realizadas por cada ruta y su respectivo resultado se presentan en la sección de **ANEXO II**.

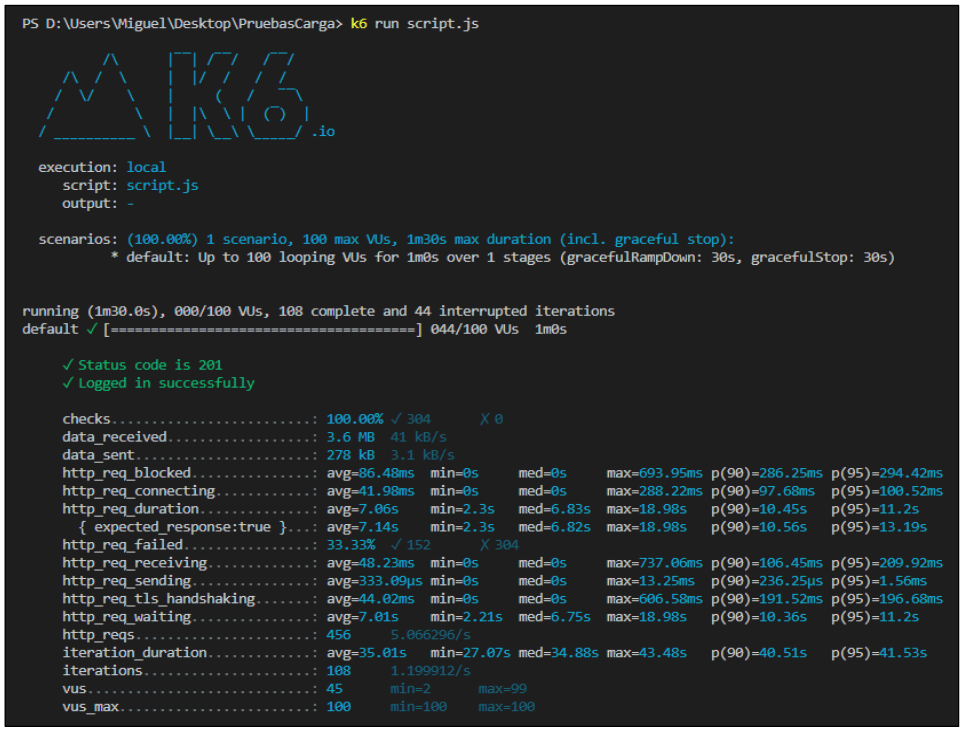


Fig. 37: Resultado de la prueba de carga en K6.

3.7 Sprint 6 Despliegue del backend.

- Deploy de la base de datos en AlwaysData.
- Deploy del sistema backend en Heroku.

Deploy de la base de datos.

Para poder utilizar la base de datos(BD) por el backend sin necesidad de almacenar de forma local, es recomendable desplegar en un servidor en la nube. Para ello, se optó por la herramienta de *AlwaysData* que brinda un servicio de BD de MySQL para almacenar los datos, entonces el servidor en línea funciona como si estuviera de forma local, aunque no tiene la misma velocidad de respuesta en las peticiones, pero tiene varios beneficios en comparación con almacenar de forma local. Uno de los principales beneficios es la disponibilidad en línea, lo que permite acceder a los datos desde cualquier lugar con una conexión a internet, además los servicios en la nube ofrecen una mayor seguridad.

Para migrar la BD a *AlwaysData*, primero se creamos una cuenta y luego una nueva base de datos. Luego se debe colocar en el archivo *env* del proyecto las claves emitidas por la plataforma, como se observa en la **Fig. 38**. Finalmente, con el comando *php artisan migrate* para migrar todas las tablas a la nueva base de datos en línea y utilizarla de manera segura.

```
DB_CONNECTION=mysql
DB_HOST=mysql-miguel.alwaysdata.net
DB_PORT=3306
DB_DATABASE=miguel_thesis_project
DB_USERNAME=miguel
DB_PASSWORD=1229myvztyE
```

Fig. 38: Credenciales para la base de datos.

Deploy del backend.

Finalmente, para que el proyecto pueda ser utilizado por el frontend, se llevó acabo el despliegue en un servidor en la nube, utilizando la plataforma de Heroku, para ello se debe utilizar comandos GIT para poder subir en un repositorio de la herramienta o conectar directamente de un repositorio en GitHub. Además es importante incluir el archivo *Procfile* en la raíz del proyecto como se muestra en la **Fig. 39**.

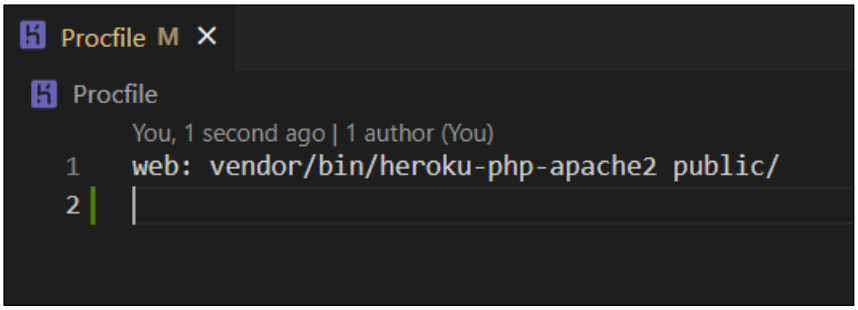


Fig. 39: Configuración del archivo Procfile.

Una vez que el proyecto se desplego se debe colocar en la sección de la app creada en Heroku y configurar las variables de entorno necesarias, ingresando las variables de configuración del proyecto específicamente del archivo *env* como se evidencia en la **Fig. 40.** Con esto, se finalizó el proceso del despliegue y se verifico la funcionad del proyecto, mediante la revisión de la URL emitida.

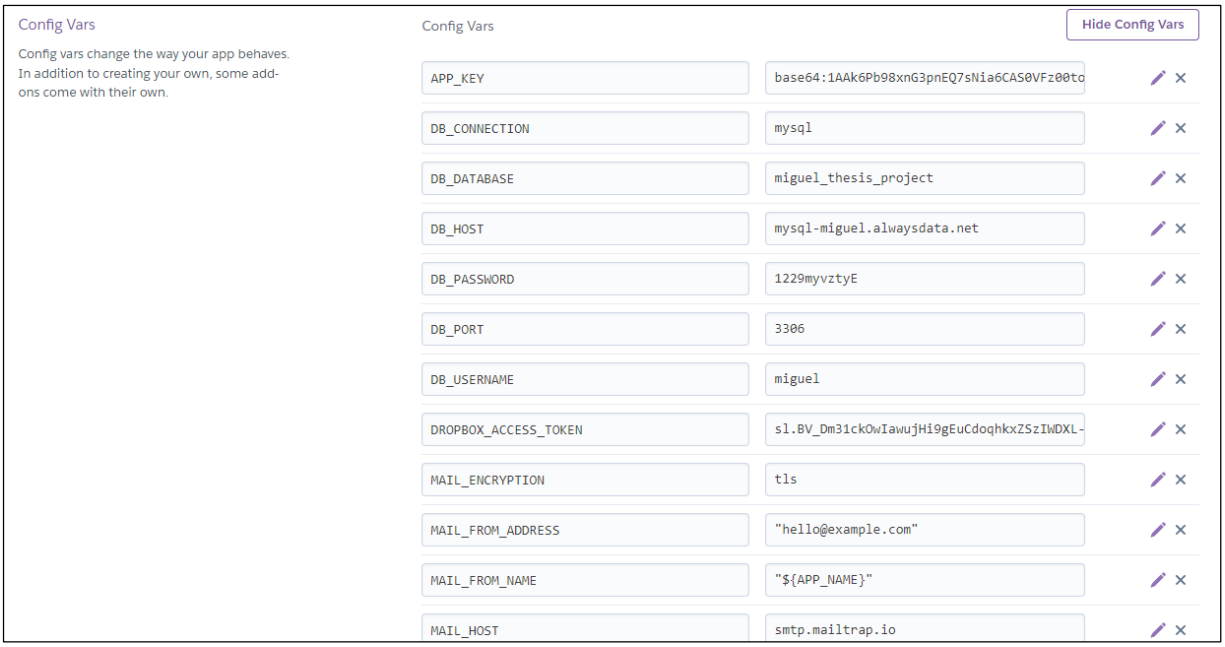


Fig. 40: Variables de entorno en Heroku.

4 CONCLUSIONES

Esta sección del documento se presenta las conclusiones logradas en el proceso del trabajo curricular desarrollado.

- Los requerimientos del proyecto de desarrollo del sistema de un ecommerce customer to customer para la venta de electrodomésticos fue llevado a cabo con éxito. Utilizando el framework de backend Laravel 9 y MySQL como base de datos, se logró implementar todos los métodos necesarios para cumplir con los requerimientos presentados al inicio del proyecto. Además, se logró el objetivo de crear una API REST que pueda ser consumida por el desarrollo Frontend.
- El uso de la metodología de trabajo Scrum fue esencial para el éxito del proyecto, lo cual se dividió mediante *Sprints* lo que permitió una planificación eficiente, una ejecución acertada y un seguimiento constante del avance del proyecto. Esto facilito la toma de decisiones y la adaptación a cambios en el proceso de desarrollo, garantizando el cumplimiento de todos los requerimientos y objetivos establecidos.
- En el diseño de la base de datos fue desarrollada cumpliendo con todos los requerimientos definidos. Se logró una interacción correcta entre las entidades, lo que permitió establecer relaciones y garantizar la seguridad de los datos.
- La arquitectura implementada del MVC fue esencial para el desarrollo del proyecto, lo cual la arquitectura se utilizó para separar de manera clara y ordenada las diferentes capas del sistema, facilitado la organización y mantenimiento del código. Además, la estructura de controladores, vistas y modelos permitió una mayor flexibilidad y escalabilidad del sistema, lo que contribuyó a su facilidad de adaptación a futuros cambios y mejoras.
- Respecto a la seguridad en el proyecto se implementó la autenticación mediante *Laravel Santum*, lo que permitió una mayor seguridad en las peticiones realizadas a la API REST, además se utilizaron medidas adiciones como el uso de *tokens* y autorizaciones para asegurar que solo usuarios autenticados pudieren acceder a los recursos del sistema.
- La seguridad e integridad de la aplicación fue probaba mediante pruebas unitarias, carga y rendimiento, por lo que se aseguró que cumpla con los estándares actuales, con ello garantizando la confidencialidad y privacidad de datos del proyecto, contribuyendo a una mayor confiabilidad del mismo.

5 RECOMENDACIONES

Esta sección del documento se presenta las recomendaciones alcanzadas durante el desarrollo del trabajo curricular desarrollado.

- Se recomienda siempre realizar pruebas de carga, rendimiento y estrés de un sistema, con el fin de evaluar el sistema y corregir antes de colocarlo en producción, con el propósito de evitar cuellos de botellas y garantizar la efectividad del mismo.
- Se recomienda considerar la escalabilidad de la base de datos al momento de planificar el proyecto, ya que es importante asegurar que la misma tenga la función de manejar una gran cantidad de datos, en el caso de un incremento de usuarios y una mayor carga de información.
- Se recomienda implementar un sistema de copias de seguridad y/o recopilación de base de datos para minimizar los riegos de perdida de datos en caso de fallas de hardware o software.
- Se recomienda continuar monitoreando y actualizando la seguridad implementada en el proyecto, ya que es un aspecto fundamental en el desarrollo de una aplicación web, además también seguir indagando sobre las mejorar en prácticas en cuanto a la autenticación y autorización en aplicaciones web, con el fin de mejorar y fortalecer la seguridad e integridad del proyecto.
- Se recomienda la implementación de un sistema de control de versiones con el propósito de facilitar las actualizaciones y mantenimiento del proyecto, además se sugiere la creación de una guía de estilo para garantizar la coherencia y legibilidad del código a medida que va creciendo el proyecto y finalmente documentar todos los recursos utilizados, para facilitar su comprensión y su uso a futuro.

REFERENCIAS BIBLIOGRÁFICAS

[1] Primicias, «Primicias,» 26 Noviembre 2021. [En línea]. Available: https://www.primicias.ec/noticias/economia/comercio-electronico-espera-crecimiento-ventas-ecuador/. [Último acceso: 16 Noviembre 2022].

[2] Primicias, «Primicias,» 06 Mayo 2021. [En línea]. Available: https://www.primicias.ec/noticias/economia/ingresos-couriers-crecieron-ecuador-pandemia/. [Último acceso: 16 Noviembre 2022].

[3] B. Pro, «Bienvenido - Bloguero Pro,» 23 Octubre 2021. [En línea]. Available: https://blogueropro.com/blog/principales-paginas-para-vender-productos-online-en-ecuador. [Último acceso: 16 Noviembre 2022].

[4] C. Ventures, «ERP & CRM Integrations | Clarity Ventures,» 06 Mayo 2020. [En línea]. Available: https://www.clarity-ventures.com/ecommerce/what-is-consumer-to-consumer-ecommerce. [Último acceso: 16 Noviembre 2022].

[5] SEOestudios, SEOestudios, 27 Enero 2020. [En línea]. Available: https://www.seoestudios.es/que-es-backend-web/. [Último acceso: 16 Noviembre 2022].

[6] Lucidchart, «Lucidchart,» 10 Junio 2020. [En línea]. Available: https://www.lucidchart.com/pages/es/que-es-un-modelo-de-base-de-datos. [Último acceso: 16 Noviembre 2022].

[7] Oracle, «Oracle | Cloud Applications and Cloud Platform,» 14 Agosto 2020. [En línea]. Available: https://www.oracle.com/ar/database/what-is-a-relational-database/. [Último acceso: 18 Noviembre 2022].

[8] R. Hat, «Red Hat - We make open source technologies for the enterprise,» 28 Mayo 2019. [En línea]. Available: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces. [Último acceso: 18 Noviembre 2022].

[9] Juanda, «gitbooks.io,» [En línea]. Available: https://juanda.gitbooks.io/webapps/content/api/arquitectura-api-rest.html. [Último acceso: 18 Noviembre 2022].

[10] Laravel, «Laravel - The PHP Framework For Web Artisans,» 13 Septiembre 2019. [En línea]. Available: https://laravel.com/docs/6.x/installation. [Último acceso: 20 Noviembre 2022].

[11] T. J. Validator, «The JSON Validator,» 16 Enero 2008. [En línea]. Available: https://jsonlint.com/. [Último acceso: 20 Noviembre 2022].

[12] R. Hat, «Red Hat - We make open source technologies for the enterprise,» 08 Abril 2020. [En línea]. Available: https://www.redhat.com/es/devops/what-is-agile-methodology. [Último acceso: 20 Noviembre 2022].

[13] S. Universidades, «Becas Santander,» 16 Septiembre 2022. [En línea]. Available: https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html. [Último acceso: 20 Noviembre 2022].

[14] T. f. Innovation, «Thinking for Innovation,» 30 Mayo 2013. [En línea]. Available: https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/. [Último acceso: 20 Noviembre 2022].

[15] G. Digital, «Blog Grupo Digital,» 21 Octubre 2019. [En línea]. Available: https://www.grupodigital.eu/blog/los-roles-de-la-metodologia-agile/. [Último acceso: 20 Noviembre 2022].

[16] I. I. Soluciones, «Integra IT Soluciones,» 28 Abril 2016. [En línea]. Available: https://integrait.com.mx/blog/roles-de-scrum/. [Último acceso: 20 Noviembre 2022].

[17] Viewnext, «Viewnext,» 27 Noviembre 2019. [En línea]. Available: https://www.viewnext.com/artefactos-scrum/. [Último acceso: 22 Noviembre 2022].

[18] I. Zabala, «Enredando Proyectos,» 07 Julio 2019. [En línea]. Available: https://enredandoproyectos.com/recopilar-los-requisitos-de-un-proyecto/. [Último acceso: 24 Noviembre 2022].

[19] Atlassian, «Atlassian,» 25 Septiembre 2019. [En línea]. Available: https://www.atlassian.com/es/agile/project-management/user-stories. [Último acceso: 24 Noviembre 2022].

[20] P. y. más, «Programación y más | Aprende desarrollo web y móvil,» 20 Septiembre 2020. [En línea]. Available: https://programacionymas.com/blog/scrum-product-backlog. [Último acceso: 24 Noviembre 2022].

[21] I. I. Soluciones, «Integra IT Soluciones,» 25 Mayo 2018. [En línea]. Available: https://integrait.com.mx/blog/sprint-y-sprint-backlog/. [Último acceso: 24 Noviembre 2022].

[22] H. d. DesarrolloWeb, «Home de DesarrolloWeb.com,» 02 Enero 2014. [En línea]. Available: https://desarrolloweb.com/articulos/que-es-mvc.html. [Último acceso: 24 Noviembre 2022].

[23] T.Txema Rodríguez, «GENBETA,» 31 03 2013. [En línea]. Available: https://www.genbeta.com/desarrollo/pusher-servicio-en-la-nube-para-gestionar-las-conexiones-y-envio-de-mensajes-mediante-websockets. [Último acceso: 04 01 2023].

[24] alwaysdata <contact@alwaysdata.com>, «alwaysdata,» 23 09 2020. [En línea]. Available: https://help.alwaysdata.com/en/. [Último acceso: 04 01 2023].

[25] R. Ricardocelis, «Platzi,» 03 11 2017. [En línea]. Available: https://platzi.com/blog/que-es-heroku/. [Último acceso: 04 01 2023].

[26] Albert, «cosasdedevs,» 20 01 2021. [En línea]. Available: https://cosasdedevs.com/posts/test-unitarios-unit-test-en-nuestro-blog-con-laravel-8/. [Último acceso: 04 01 2023].

[27] k6.io, «Stress testing,» 22 04 2021. [En línea]. Available: https://k6.io/docs/es/tipos-de-prueba/stress-testing/. [Último acceso: 18 01 2023].

6 ANEXOS

A continuación, se presentan los ANEXOS que se han implementado para el desarrollo del backend, los cuales se encuentran divididos de la siguiente manera:

- **ANEXO I.** Resultados del programa anti plagio Turnitin.
- **ANEXO II.** Manual Técnico.
- **ANEXO III.** Manual de Usuario.
- **ANEXO IV.** Credenciales de acceso y despliegue.

ANEXO I

A continuación, se presenta el certificado que el Director de Tesis ha emitido y en donde se evidencia el resultado que se ha obtenido en la herramienta anti plagio Turnitin.

ESCUELA POLITÉCNICA NACIONAL
ESCUELA DE FORMACIÓN DE TECNÓLOGOS
CAMPUS POLITÉCNICO "ING. JOSÉ RUBÉN ORELLANA"

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 1 de marzo de 2023

De mi consideración:

Yo, Franco Rocha Yadira Guissela, en calidad de Director del Trabajo de Integración Curricular titulado DESARROLLO BACKEND asociado al DESARROLLO DE SISTEMA ECOMMERCE CUSTOMER TO CUSTOMER PARA ELECTRODOMÉSTICOS elaborado por el estudiante **Eduardo Miguel Muzo Cansigña** de la carrera en Tecnología Superior en Desarrollo de Software, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito en las secciones: Descripción del componente desarrollado, Metodología, Resultados, Conclusiones y Recomendaciones, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 8%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el informe generado por la herramienta Turnitin.

Atentamente,

Firmado electrónicamente por:
YADIRA GUISSELA
FRANCO ROCHA

Franco Rocha Yadira Guissela
Profesor Ocasional a Tiempo Completo
Escuela de Formación de Tecnólogos

ANEXO II

Recopilación de requerimientos

En la siguiente **TABLA X** se presenta la recopilación de requerimientos obtenido al inicio del proyecto de acuerdo con lo planificado, entrevistas y solicitado por el *Product Owner*.

TABLA X: Recopilación de requerimientos.

RECOPILACIÓN DE REQUERIMIENTOS		
TIPO DEL SISTEMA	**ID - RR**	**ENUNCIADO DEL ÍTEM**
Backend	**RR002**	Como usuario administrador o cliente necesito implementar varios métodos para: • Iniciar sesión. • Cerrar sesión. • Recuperar contraseña.
	RR003	Como usuario administrador o cliente necesito implementar varios métodos para: • Modificar perfil.
	RR004	Como usuario cliente necesito implementar varios métodos para: • Gestionar electrodomésticos.
	RR005	Como usuario cliente necesito implementar varios métodos para: • Buscar electrodomésticos.
	RR006	Como usuario cliente necesito implementar varios métodos para: • Comentar electrodomésticos.
	RR007	Como usuario cliente necesito implementar varios métodos para: • Reportar electrodomésticos.
	RR008	Como usuario cliente necesito implementar varios métodos para: • Realizar suscripción.

	RR09	Como usuario cliente necesito implementar varios métodos para: • Crear chat interno comprador – vendedor.
	RR010	Como usuario administrador necesito implementar varios métodos para: • Gestionar clientes.
	RR011	Como usuario administrador necesito implementar varios métodos para: • Gestionar suscripción.
	RR012	Como usuario administrador necesito implementar varios métodos para: • Gestionar reportes de electrodomésticos.
	RR013	Como usuario administrador necesito implementar varios métodos para: • Gestionar comentarios.
	RR014	Como usuario administrador necesito implementar varios métodos para: • Gestionar categorías.
	RR015	Como usuario administrador necesito implementar varios métodos para: • Gestionar electrodomésticos de perfil cliente.

Historias de Usuario

En las siguientes tablas se presenta todas las historias de usuario para el proyecto del componente backend basándose en todos los requerimientos de usuarios presentado en la sección anterior.

TABLA XI: Historia de Usuario N:2.

HISTORIA DE USUARIO	
Identificador: HU002	**Usuario:** Administrador y Cliente
Nombre de historia: Iniciar sesión, cerrar sesión y modificar contraseña.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Iteración asignada: 1	
Responsable: Eduardo Muzo	
Descripción: El *backend* mediante métodos debe proporcionar la función de autenticar el usuario para ingresar al sistema mediante: • Correo electrónico. • Contraseña.	
Observación: El *backend* comprueba la existencia del usuario y autoriza el ingreso al sistema dependiendo del rol. Para recuperar la contraseña se envía un correo electrónico.	

TABLA XII: Historia de Usuario N:3.

HISTORIA DE USUARIO	
Identificador: HU003	**Usuario:** Administrador y Cliente
Nombre de historia: Modificar perfil.	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Media
Iteración asignada: 1	
Responsable: Eduardo Muzo	
Descripción: El *backend* mediante métodos debe proporcionar la función de modificar y visualizar el perfil de usuarios: • Nombre y Apellido. • Nombre usuario. • Numero Personal. • Numero de domicilio. • Dirección. • Email.	

• Imagen.
Observación: El *backend* comprueba mediante autenticación al usuario y valida los campos ingresados para la actualización.

TABLA XIII: Historia de Usuario N:4.

HISTORIA DE USUARIO	
Identificador: HU004	**Usuario:** Cliente
Nombre de historia: Gestionar electrodomésticos.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Iteración asignada: 2	
Responsable: Eduardo Muzo	
Descripción: El *backend* mediante métodos debe proporcionar la función de gestionar electrodomésticos: • Añadir electrodomésticos a la venta. • Eliminar electrodomésticos a la venta. • Actualizar electrodomésticos a la venta. • Visualizar electrodomésticos a la venta.	
Observación: El *backend* comprueba al usuario por su rol y asigna al cliente la función de realizar las acciones mencionadas en la descripción. El usuario cliente es el único encargo de gestionar sus propios electrodomésticos.	

TABLA XIV: Historia de Usuario N:5.

HISTORIA DE USUARIO	
Identificador: HU005	**Usuario:** Cliente
Nombre de historia: Buscar electrodomésticos.	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Media
Iteración asignada: 2	
Responsable: Eduardo Muzo	
Descripción: El *backend* mediante métodos debe proporcionar la función de buscar electrodomésticos: • Por título del electrodoméstico.	
Observación: El *backend* debe proporcionar la función de búsqueda por palabra reservada o filtrar campos específicos.	

TABLA XV: Historia de Usuario N:6.

HISTORIA DE USUARIO	
Identificador: HU006	**Usuario:** Cliente
Nombre de historia: Comentar electrodomésticos.	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Media
Iteración asignada: 2	
Responsable: Eduardo Muzo	
Descripción: El *backend* mediante métodos debe proporcionar la función de comentar electrodomésticos sobre alguna duda o sugerencia respecto al electrodoméstico.	
Observación: El *backend* debe proporcionar la función de comentar y eliminar comentarios al usuario cliente respecto a los electrodomésticos existentes en el sistema.	

TABLA XVI: Historia de Usuario N:7.

HISTORIA DE USUARIO	
Identificador: HU007	**Usuario:** Cliente
Nombre de historia: Reportar electrodoméstico.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Media
Iteración asignada: 2	
Responsable: Eduardo Muzo	
Descripción: El *backend* mediante métodos debe proporcionar al cliente la función de reportar a un electrodoméstico respecto a posibles estafas, en cuanto se debe llenar los siguientes campos: • Motivo. • Problema.	
Observación: El *backend* debe proporcionar la función de reportar productos a los clientes, en el cual se debe llenar un formulario y luego evaluado por el administrador.	

TABLA XVII: Historia de Usuario N:8.

HISTORIA DE USUARIO	
Identificador: HU008	**Usuario:** Cliente
Nombre de historia: Realizar suscripción	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Iteración asignada: 3	
Responsable: Eduardo Muzo	
Descripción: El *backend* mediante métodos debe proporcionar al cliente la función de adquirir una suscripción para que un producto tenga mayor visualización en el sistema, para solicitar se debe llenar un formulario con los siguientes campos: • Tipo de transferencia. • Cantidad.	
Observación: El *backend* debe proporcionar la función de solicitar al sistema una suscripción y notificar al cliente que el producto fue asignado en destacados.	

TABLA XVIII: Historia de Usuario N:9.

HISTORIA DE USUARIO	
Identificador: HU009	**Usuario:** Cliente
Nombre de historia: Crear chat interno comprador-vendedor	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Alta
Iteración asignada: 3	
Responsable: Eduardo Muzo	
Descripción: El *backend* mediante métodos debe proporcionar al cliente la función de realizar una comunicación mediante un chat entre comprador y vendedor para realizar la venta de un electrodoméstico en el cual se debe definir: • Método de pago del electrodoméstico. • Método de envió del electrodoméstico	
Observación: El *backend* debe proporcionar la función de crear un chat para el perfil cliente y comunicación entre clientes para poder vender o comprar productos del sistema.	

TABLA XIX: Historia de Usuario N:10.

HISTORIA DE USUARIO	
Identificador: HU010	**Usuario:** Administrador
Nombre de historia: Gestionar clientes	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Media
Iteración asignada: 4	
Responsable: Eduardo Muzo	
Descripción: El *backend* mediante métodos debe proporcionar la función de gestionar a clientes del sistema: • Eliminar usuarios clientes. • Ver detalles usuarios clientes.	
Observación: El usuario administrador es el único para realizar la gestión de clientes.	

TABLA XX: Historia de Usuario N:11.

HISTORIA DE USUARIO	
Identificador: HU011	**Usuario:** Administrador
Nombre de historia: Gestionar de suscripción	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Media
Iteración asignada: 4	
Responsable: Eduardo Muzo	
Descripción: El *backend* mediante métodos debe proporcionar la función de gestionar las solicitudes de suscripción del cliente sobre la suscripción del producto a ser mayormente visualizado en el sistema, el administrador tiene las acciones de: • Visualizar suscripciones. • Aceptar o Cancelar suscripciones. • Verificar pago de suscripciones.	
Observación: El usuario administrador es el encargado de verificar el pago de suscripción o cancelar si existe alguna vulnerabilidad.	

TABLA XXI: Historia de Usuario N:12.

HISTORIA DE USUARIO	
Identificador: HU012	**Usuario:** Administrador
Nombre de historia: Gestionar reportes de electrodomésticos.	
Prioridad en Negocio: Alta	**Riesgo en Negocio:** Alta
Iteración asignada: 4	
Responsable: Eduardo Muzo	
Descripción: El *backend* mediante métodos debe proporcionar la función de gestionar reportes, por lo que puede realizar las siguientes acciones: • Cambiar estado del reporte. • Eliminar reporte. • Ver detalles del reporte.	
Observación: El *backend* debe comprobar mediante autenticación al administrador para poder tener acceso al campo de gestionar la plataforma.	

TABLA XXII: Historia de Usuario N:13.

HISTORIA DE USUARIO	
Identificador: HU013	**Usuario:** Administrador
Nombre de historia: Gestionar comentarios.	
Prioridad en Negocio: Alta	**Riesgo en Negocio:** Alta
Iteración asignada: 4	
Responsable: Eduardo Muzo	
Descripción: El *backend* mediante métodos debe proporcionar la función de gestionar comentarios, por lo que puede realizar las siguientes acciones: • Ver detalles del comentario. • Eliminar comentarios.	
Observación: El *backend* debe comprobar mediante autenticación al administrador para poder tener acceso al campo de gestionar la plataforma.	

TABLA XXIII: Historia de Usuario N:14.

HISTORIA DE USUARIO	
Identificador: HU014	**Usuario:** Administrador
Nombre de historia: Gestionar categorías.	
Prioridad en Negocio: Alta	**Riesgo en Negocio:** Alta
Iteración asignada: 4	
Responsable: Eduardo Muzo	
Descripción: El *backend* mediante métodos debe proporcionar la función de gestionar categorías, por lo que puede realizar las siguientes acciones: • Ver categorías de electrodomésticos. • Actualizar categorías de electrodomésticos. • Crear categorías de electrodomésticos. • Eliminar categorías de electrodomésticos.	
Observación: El *backend* debe comprobar mediante autenticación al administrador para poder tener acceso al campo de gestionar la plataforma.	

TABLA XXIV: Historia de Usuario N:15.

HISTORIA DE USUARIO	
Identificador: HU015	**Usuario:** Administrador
Nombre de historia: Gestionar electrodomésticos de perfil cliente.	
Prioridad en Negocio: Alta	**Riesgo en Negocio:** Alta
Iteración asignada: 4	
Responsable: Eduardo Muzo	
Descripción: El *backend* mediante métodos debe proporcionar la función de gestionar electrodomésticos de perfil cliente, por lo que puede realizar las siguientes acciones: • Eliminar producto. • Ver detalle del producto.	
Observación: El *backend* debe comprobar mediante autenticación al administrador para poder tener acceso al campo de gestionar la plataforma.	

Product Backlog

En esta sección se presenta la prioridad de los requisitos implementados para el desarrollo del componente backend el cual se presenta en la **TABLA XXV**.

TABLA XXV: Tabla de la elaboración del *product backlog*.

ELABORACIÓN DEL *PRODUCT BACKLOG*				
ID – HU	**HISTORIA DE USUARIO**	**ITERACIÓN**	**ESTADO**	**PRIORIDAD**
HU002	Iniciar sesión, cerrar sesión y modificar contraseña.	1	Finalizado	Alta
HU003	Modificar perfil	1	Finalizado	Media
HU004	Gestionar electrodomésticos	2	Finalizado	Alta
HU005	Buscar electrodomésticos	2	Finalizado	Baja
HU006	Comentar electrodomésticos	2	Finalizado	Media
HU007	Reportar electrodomésticos	2	Finalizado	Media
HU008	Realizar suscripción	3	Finalizado	Alta
HU009	Crear chat interno comprador-vendedor	3	Finalizado	Alta
HU010	Gestionar clientes	4	Finalizado	Media
HU011	Gestionar suscripción	4	Finalizado	Alta
HU012	Gestionar reportes de electrodomésticos.	4	Finalizado	Alta

HU013	Gestionar comentarios	4	Finalizado	Media
HU014	Gestionar categorías	4	Finalizado	Alta
HU015	Gestionar electrodomésticos de perfil cliente.	4	Finalizado	Alta

Sprint Backlog

En esta sección se presenta todos los *Sprint backlog* desarrollados durante el proceso del desarrollo backend, donde se aprecia todas las actividades y el tiempo estimado para su presentación establecidos por el *Product Owner*.

TABLA XXVI: Tabla de la Sprint Backlog.

ELABORACIÓN DEL SPRINT BACKLOG						
ID-SB	**NOMBRE**	**MÓDULO**	**ID-HU**	**HISTORIAS DE USUARIOS**	**TAREAS**	**TIEMPO ESTIMADO**
SB000	Configuración del ambiente de desarrollo.		HU000	N/A	• Recopilación de requerimientos. • Instalación y configuración del entorno para el desarrollo del proyecto. • Creación de reposito GitHub. • Diseño del modelo de base de datos MySQL.	20 H
SB001	Diseño e implementación de métodos para el perfil cliente y administrador.	Inicio de sesión	HU001	Registrar Usuarios	• Diseño e implementación de métodos para el registro de usuarios con perfil cliente. • Validación de campos para el registro del usuario con perfil cliente. • Conexión y consulta a la base de datos.	30 H

		Módulo de autenticación al sistema	HU002	Iniciar sesión, cerrar sesión y modificar contraseña.	• Diseño e implementación de métodos para iniciar sesión, cerrar sesión y modificar contraseña para todos los perfiles del sistema. • Consulta a la base de datos y autorización. • Validación de campos requeridos.	
		Modulo perfil	HU003	Modificar perfil	• Diseño e implementación de métodos para visualizar y modificar el perfil. • Consulta a la base de datos y autorización. • Validación de campos requeridos.	
SB002	Diseño e implementación de métodos del modelo clientes y electrodomésticos.	Modulo electrodoméstico	HU004	Gestionar electrodomésticos.	• Diseño e implementación de métodos para ingresar, visualizar, actualizar y eliminar electrodomésticos. • Consulta a la base de datos y autorización. • Validación de campos requeridos.	50 H
		Modulo buscar electrodoméstico	HU005	Buscar electrodomésticos	• Diseño e implementación de métodos para la búsqueda de electrodomésticos. • Consulta a la base de datos.	
		Modulo comentar	HU006	Comentar electrodomésticos	• Diseño e implementación de métodos para añadir, autorizar y eliminar comentarios en publicaciones de electrodomésticos.	

					• Consulta a la base de datos y autorización.	
		Modulo reporte	HU007	Reportar electrodomésticos	• Diseño e implementación de métodos para realizar reportes a electrodomésticos • Consulta a la base de datos y autorización. • Validación de campos requeridos.	
SB003	Diseño e implementación de métodos de suscripción y chat interno.	Modulo suscripción	HU008	Solicitar suscripción	• Diseño e implementación de métodos para solicitar una suscripción. • Consulta a la base de datos y autorización. • Validación de campos requeridos.	40 H
		Modulo chat interno vendedor – comprador	HU009	Crear chat interno comprador-vendedor	• Diseño e implementación de métodos para crear un chat interno en tiempo real para la compra o venta. • Consulta a la base de datos y autorización. • Validación de campos requeridos.	
SB004	Diseño e implementación de métodos para el perfil administrador.	Modulo gestión de clientes	HU010	Gestionar clientes	• Diseño e implementación de métodos para visualizar y eliminar usuarios con perfil cliente. • Consulta a la base de datos y autorización.	40H
		Modulo suscripción	HU011	Gestionar suscripciones	• Diseño e implementación de métodos para visualizar, aceptar o cancelar suscripciones de clientes. • Consulta a la base de datos y autorización.	

		Modulo reportes de electrodomésticos	HU012	Gestionar reportes de electrodomésticos.	• Diseño e implementación de métodos para ver detalles, eliminar y cambiar estado del reporte. • Consulta a la base de datos y autorización.	
		Modulo gestionar comentarios	HU013	Gestionar comentarios	• Diseño e implementación de métodos para ver detalles y eliminar comentarios. • Consulta a la base de datos y autorización.	
		Modulo categorías	HU014	Gestionar categorías	• Diseño e implementación de métodos para ver, actualizar, crear y eliminar categorías. • Consulta a la base de datos y autorización. • Validación de campos requeridos.	
		Modulo gestionar electrodomésticos	HU015	Gestionar electrodomésticos de perfil cliente.	• Diseño e implementación de métodos para ver detalles y eliminar electrodomésticos de los clientes. • Consulta a la base de datos y autorización.	
SB005	Pruebas del Backend.				• Pruebas unitarias • Pruebas de rendimiento	20 H
SB006	Despliegue del Backend.				• Despliegue del backend en heroku y Alwaysdata	10 H
Documentación					• Trabajo de integración curricular • Anexos del proyecto.	30 h
TOTAL						240 H

Diseño de la base de datos SQL.

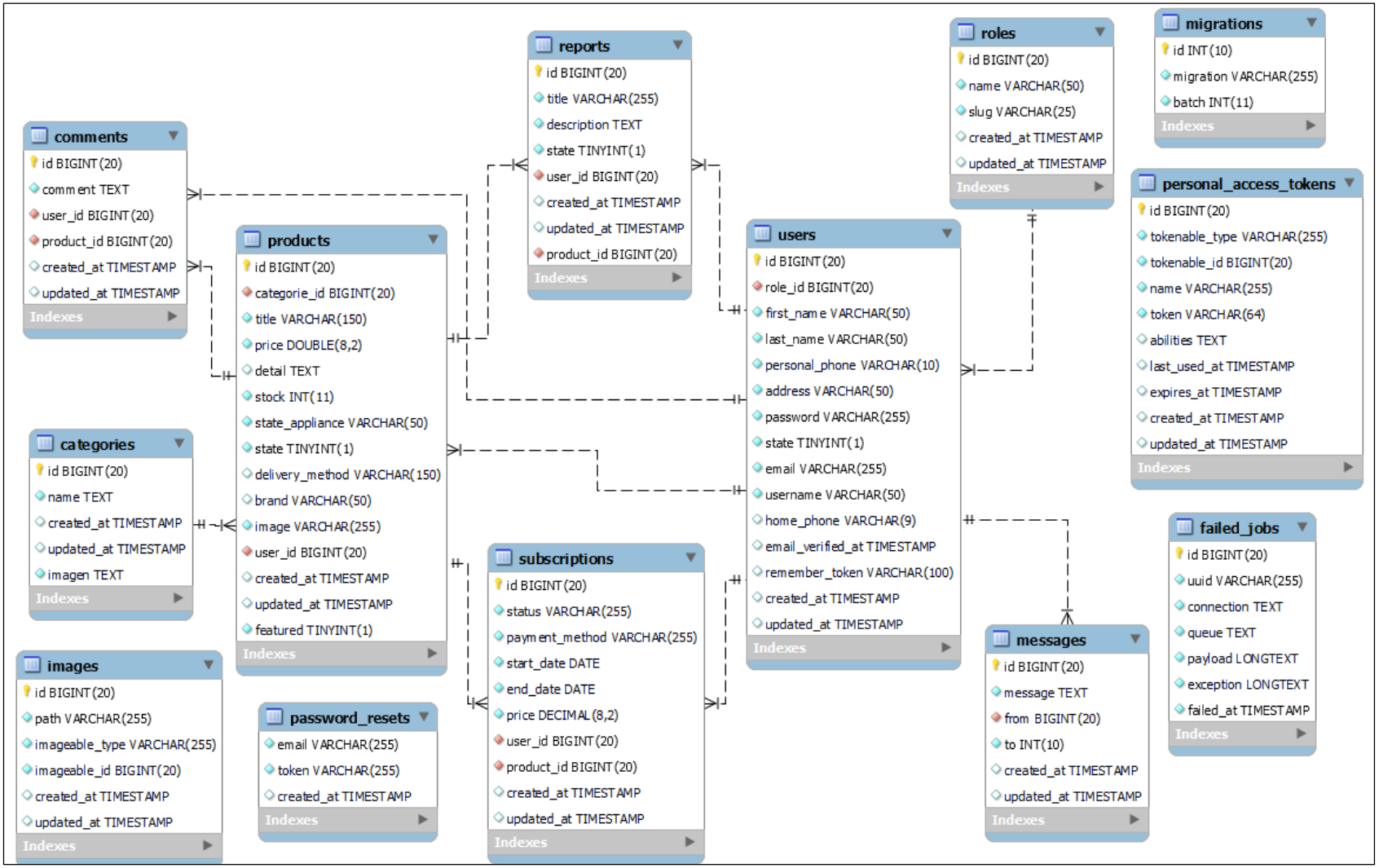


Fig. 41: Diseño de la base de datos en MySQL.

Pruebas

Al final del proyecto y luego de culminar la implementación, se desarrolló las pruebas unitarias, estrés, carga, compatibilidad y aceptación para asegurar la calidad de la aplicación, además, se realizó pruebas de aceptación de clientes por cada una de las historias propuestas con la finalidad de asegurar y comprobar todas las funcionalidades desarrolladas.

Pruebas Unitarias

Para verificar el correcto funcionamiento de las rutas en el sistema, se realizaron pruebas unitarias en las cuales se evaluó la respuesta de cada una de las URLs al cumplir con las historias de usuario. En caso de ser una URL satisfactoria, se espera una respuesta con un estado "OK" y un código de 200; si es un método para crear, se espera una respuesta con un estado "Created" y un código de 201.

```php
/** @test login */
0 references | 0 overrides
public function test_login()
{
    $response = $this->post('http://127.0.0.1:8000/api/login', [
        'email' => 'test@example.com.ec',
        'password' => 'Test.123456'
    ]);
    $response->assertStatus(200);
}
```

Fig. 42: Script para la probar el inicio de sesión.

```php
/** @test register */
0 references | 0 overrides
public function test_register()
{
    $response = $this->post('http://127.0.0.1:8000/api/register', [
        'username' => 'Test 1',
        'first_name' => 'Test 3',
        'last_name' => 'Test 1',
        'email' => 'test@example.com.ec',
        'home_phone' => '1234567',
        'personal_phone' => '1234567890',
        'address' => 'Test 1',
        'password' => 'Test.123456',
        'password_confirmation' => 'Test.123456',
    ]);
    $response->assertStatus(201);
}
```

Fig. 43: Script para la probar el registro de un usuario con perfil cliente.

```php
/** @test view my profile */
0 references | 0 overrides
public function test_profile()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
    ])->get('http://127.0.0.1:8000/api/profile');
    $response->assertStatus(200);
}
```

Fig. 44: Script para la probar la visualización de perfil.

```php
/** @test update my profile */
0 references | 0 overrides
public function test_update_profile()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
    ])->post('http://127.0.0.1:8000/api/profile', [
            "username" => "MiguelMuzo Test",
            "first_name" => "Test 1",
            "last_name" => "Cliente",
            "email" => "example_01@gmail.com",
            "home_phone" => "029570994",
            "personal_phone" => "0919756468",
            "address" => "Guayaquil"
        ]);
    $response->assertStatus(200);
    $response->assertJsonFragment([
        'message' => 'Profile updated successfully',
    ]);
}
```

Fig. 45: Script para probar la actualización de información.

```php
/** @test update avatar */
0 references | 0 overrides
public function test_update_avatar()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
    ])->post('http://127.0.0.1:8000/api/profile/avatar', [
            'image' => $this->getUploadedFile()
        ]);
    $response->assertStatus(200)->assertSee('Avatar updated successfully');
    $data = $response->getData();
    print_r($data);
    return $response->assertJsonFragment([
        'message' => 'Avatar updated successfully',
    ]);

}
```

Fig. 46: Script para probar la actualización del avatar en el sistema.

```php
/** @test views products */
0 references | 0 overrides
public function test_views_products()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
    ])->get('http://127.0.0.1:8000/api/products');

    $response->assertStatus(200);
}
```

Fig. 47: Script para probar el listado de los productos del sistema.

```php
/** @test create product */
0 references | 0 overrides
public function test_create_product()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
    ])->post('http://127.0.0.1:8000/api/products', [
            'title' => 'Test Product',
            'price' => 100,
            'description' => 'Test Product Description',
            'detail' => 'Test Product Detail',
            'stock' => 10,
            'state_appliance' => 'Nuevo',
            'delivery_method' => 'Envio gratis',
            'brand' => 'Test Brand',
            'categorie_id' => 5,
            'image' => $this->getUploadedFile(),
            'address' => 'Quito-Ecuador',
            'phone' => '2787675'
        ]);
    $response->assertStatus(201);
}
```

Fig. 48: Script para probar la creación de un producto.

```php
/**@test update product */
0 references | 0 overrides
public function test_update_product()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
    ])->post('http://127.0.0.1:8000/api/products/103/update', [
            'title' => 'Test Update Product',
            'price' => 100,
            'description' => 'Test Product Description',
            'detail' => 'Test Product Detail',
            'stock' => 10,
            'state_appliance' => 'Nuevo',
            'delivery_method' => 'Envio gratis',
            'brand' => 'Test Brand',
            'categorie_id' => 5,
            'image' => $this->getUploadedFile(),
            'address' => 'Quito-Ecuador',
            'phone' => '2787675'
        ]);
    $response->assertStatus(200);
}
```

Fig. 49: Script para probar la actualización del producto.

```php
/**@test delete product */
0 references | 0 overrides
public function test_delete_product(){
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
        ])->delete("http://127.0.0.1:8000/api/products/103");

    $response->assertStatus(200);
}
```

Fig. 50: Script probar la eliminación de un producto.

```php
/**@test search product */
0 references | 0 overrides
public function test_search_product(){
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
        ])->get("http://127.0.0.1:8000/api/search?title=sistema");

    $response->assertStatus(200);
}
```

Fig. 51: Script para probar la búsqueda de un producto mediante el título.

```php
/**@test filter product */
0 references | 0 overrides
public function test_filter_product(){
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
        ])->get("http://127.0.0.1:8000/api/filter/products?state_appliance=nuevo&categorie_id=1");

    $response->assertStatus(200);
}
```

Fig. 52: Script para probar el filtrado de un producto.

```php
/**@test create report on a product specific*/
0 references | 0 overrides
public function test_create_report_on_a_specific_product(){
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
    ])->post('http://127.0.0.1:8000/api/products/56/reports', [
        "title" => "Titulo Reporte al producto 4",
        "description" =>"Esto es la description"
    ]);
    $response->assertStatus(200);
}
```

Fig. 53: Script para probar el reporte un producto en venta sospechoso.

```php
/**@test create chat */
0 references | 0 overrides
public function test_create_chat(){
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
    ])->post('http://127.0.0.1:8000/api/user/send', [
        "to" => 21,
        "message" => "Hola esto es una prueba unitaria de la ruta para crear un chat"
    ]);
    $response->assertStatus(200);
}
/**@test view messages received */
0 references | 0 overrides
public function test_view_messages_received(){
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
    ])->get('http://127.0.0.1:8000/api/user/received');
    $response->assertStatus(200);
}

/**@test see message on a specific user*/
0 references | 0 overrides
public function test_see_message_on_a_specific_user(){
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
    ])->get('http://127.0.0.1:8000/api/user/21/messages');
    $response->assertStatus(200);
}
/**@test view contacts */
0 references | 0 overrides
public function test_view_contacts(){
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
    ])->get('http://127.0.0.1:8000/api/user/contacts');
    $response->assertStatus(200);
}
```

Fig. 54: Script para probar los métodos de crear un chat.

```php
/**@test view reports */
0 references | 0 overrides
public function test_view_reports()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 11|5gaJX0T4ijo5LelKLk3SDBstjN5L6to4L6GGO36T',
    ])->get('http://127.0.0.1:8000/api/admin/reports');
    $response->assertStatus(200);
}
/**@test update state */
0 references | 0 overrides
public function test_update_state_reports()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 11|5gaJX0T4ijo5LelKLk3SDBstjN5L6to4L6GGO36T',
    ])->get('http://127.0.0.1:8000/api/admin/reports/7',[
        "state" => false
    ]
);
    $response->assertStatus(200);
}
/**@test delete report */
0 references | 0 overrides
public function test_delete_reports()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 11|5gaJX0T4ijo5LelKLk3SDBstjN5L6to4L6GGO36T',
    ])->delete('http://127.0.0.1:8000/api/admin/reports/7');
    $response->assertStatus(200);
}
```

Fig. 55: Script para probar la gestión de reportes.

```php
/**view @test comments on a specific product*/
0 references | 0 overrides
public function test_view_comments_on_a_specific_product()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
    ])->get('http://127.0.0.1:8000/api/products/1/comments');
    $response->assertStatus(200);
}
/**@test create comment on a specfic product */
0 references | 0 overrides
public function test_create_comment_on_a_specific_product(){
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
    ])->post('http://127.0.0.1:8000/api/products/70/comments', [
        'comment' => 'This is a test comment',
    ]);
    $response->assertStatus(201);
}
/**@test update comment on a specfic product */
0 references | 0 overrides
public function test_update_comment_on_a_specific_product(){
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
    ])->put('http://127.0.0.1:8000/api/products/70/comments/9', [
        'comment' => 'This is a test comment updated',
    ]);
    $response->assertStatus(200);
}
/**@test delete comment on a specfic product */
0 references | 0 overrides
public function test_delete_comment_on_a_specific_product(){
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 8|Nff2WFqOIXFxQs5hRXAUb6fDqKUoSKTUjgidO8w9',
    ])->delete('http://127.0.0.1:8000/api/products/70/comments/9');
    $response->assertStatus(200);
}
```

Fig. 56: Script para probar la gestión de comentarios por parte del administrador.

```php
/**@test views admin subscripcions*/
0 references | 0 overrides
public function test_views_admin_subscriptions()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 11|5gaJX0T4ijo5LelKLk3SDBstjN5L6to4L6GGO36T',
        ])->get('http://127.0.0.1:8000/api/admin/subscriptions');
    $response->assertStatus(200);
}
/**@test cancel subscripcions*/
0 references | 0 overrides
public function test_cancel_subscriptions()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 11|5gaJX0T4ijo5LelKLk3SDBstjN5L6to4L6GGO36T',
        ])->get('http://127.0.0.1:8000/api/admin/subscriptions/16/cancel');
    $response->assertStatus(200);
}
/**@test accept subscripcions*/
0 references | 0 overrides
public function test_accept_subscriptions()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 11|5gaJX0T4ijo5LelKLk3SDBstjN5L6to4L6GGO36T',
        ])->get('http://127.0.0.1:8000/api/admin/subscriptions/3');
    $response->assertStatus(200);
}
```

Fig. 57: Script para probar la gestión de suscripción por parte del administrador.

```php
/**@test view categories */
0 references | 0 overrides
public function test_view_categories()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 11|5gaJX0T4ijo5LelKLk3SDBstjN5L6to4L6GGO36T',
    ])->get('http://127.0.0.1:8000/api/admin/categories');
    $response->assertStatus(200);
}
/**@test create categorie */
0 references | 0 overrides
public function test_create_categorie()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 11|5gaJX0T4ijo5LelKLk3SDBstjN5L6to4L6GGO36T',
    ])->post('http://127.0.0.1:8000/api/admin/categories',[
        "name" => "test categorie",
        "imagen" => "https://m.media-amazon.com/images/I/81DIBUvqDtL._SR1260,840_SR630,420_.jpg"
    ]);
    $response->assertStatus(201);
}
/**@test update categorie */
0 references | 0 overrides
public function test_update_categorie()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 11|5gaJX0T4ijo5LelKLk3SDBstjN5L6to4L6GGO36T',
    ])->put('http://127.0.0.1:8000/api/admin/categories/5',[
        "name" => "test categorie",
        "imagen" => "https://m.media-amazon.com/images/I/81DIBUvqDtL._SR1260,840_SR630,420_.jpg"
    ]);
    $response->assertStatus(200);
}
```

Fig. 58: Script para probar la gestión de categorías por parte del administrador.

```php
/**@test views admin products */
0 references | 0 overrides
public function test_views_admin_products()
{
    $response = $this->withHeaders([
        'Authorization' => 'Bearer 11|5gaJX0T4ijo5LelKLk3SDBstjN5L6to4L6GGO36T',
        ])->get('http://127.0.0.1:8000/api/admin/products');

    $response->assertStatus(200);
}
/**@test delete admin products */
0 references | 0 overrides
public function test_delete_admin_product()
{
     $response = $this->withHeaders([
        'Authorization' => 'Bearer 11|5gaJX0T4ijo5LelKLk3SDBstjN5L6to4L6GGO36T',
    ])->delete('http://127.0.0.1:8000/api/admin/products/26');

    $response->assertStatus(200);
}
```

Fig. 59: Script para probar la gestión de electrodomésticos por el administrador.

En las figuras **Fig. 60** e **Fig. 61** se muestran los resultados obtenidos de las pruebas realizadas en diferentes métodos del sistema con respecto a las historias de usuario. Los resultados indican una aceptación completa en todas las pruebas efectuadas.

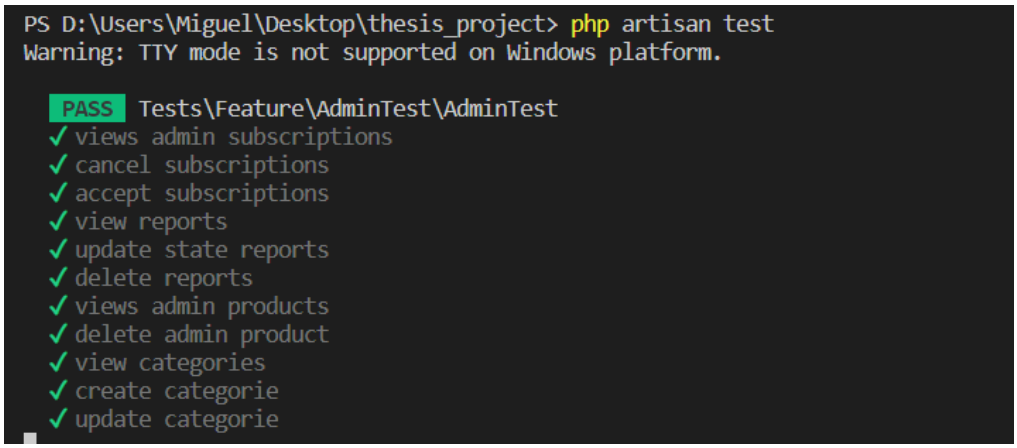


Fig. 60: Resultados de la primera parte de las pruebas unitarias.

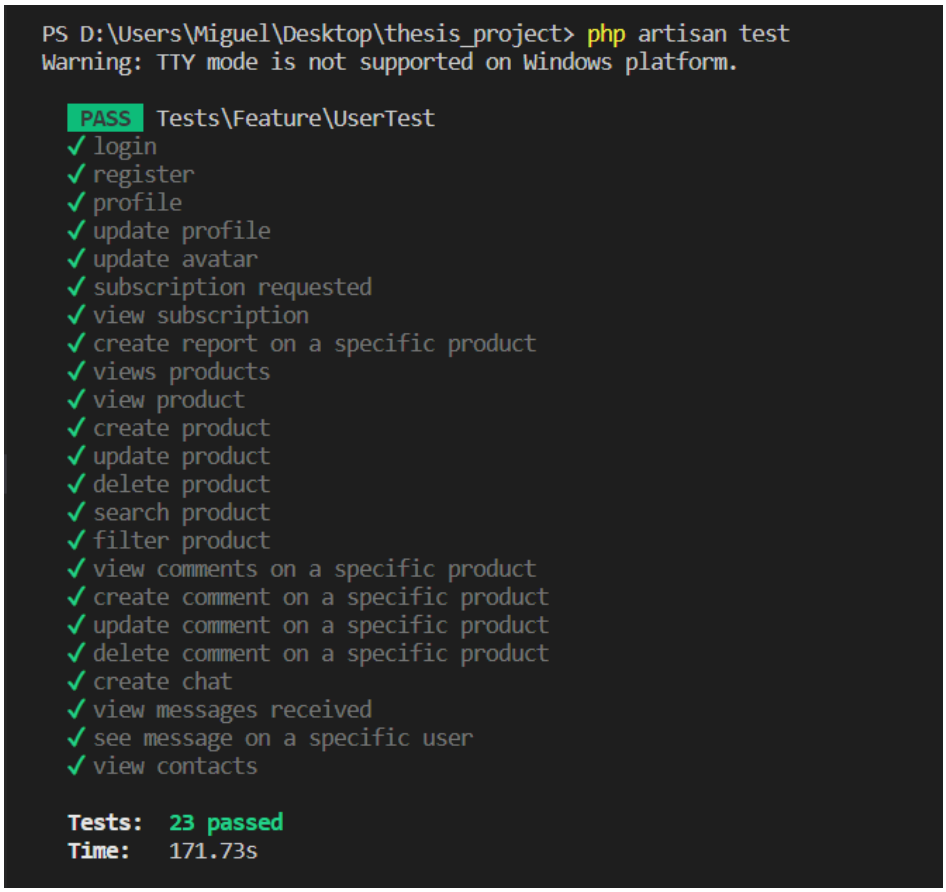


Fig. 61: Resultados de la segunda parte de las pruebas unitarias.

Pruebas de Estrés

En las siguientes figuras, se muestran las pruebas de estrés realizadas en el sistema. Se evaluó el comportamiento de los métodos ante una cantidad determinada de peticiones en un tiempo específico, con el objetivo de verificar el alcance de la API REST.

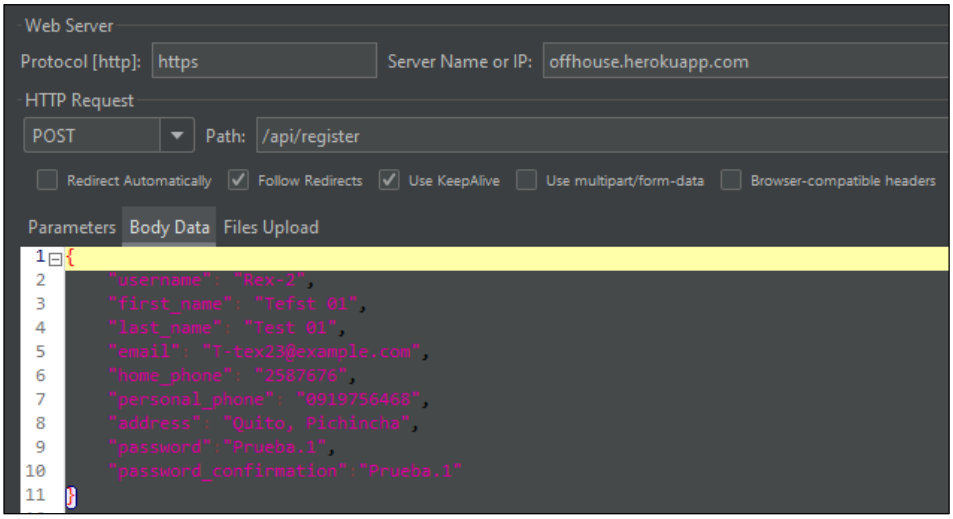


Fig. 62: Prueba de estrés en la historia de usuario N:1.

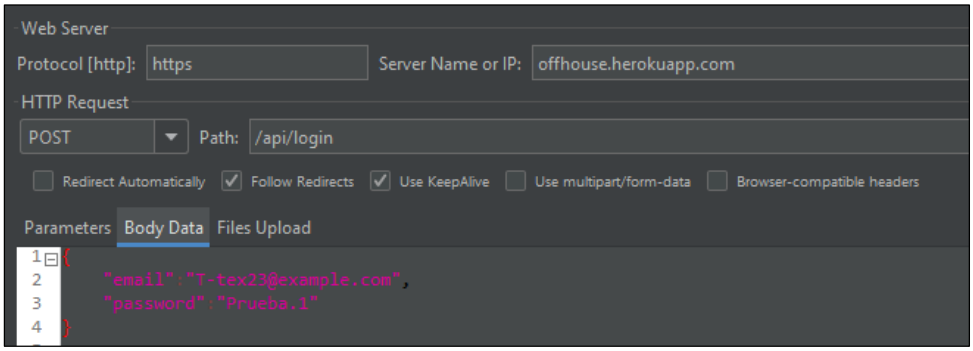


Fig. 63: Prueba de estrés en la historia de usuario N:2.

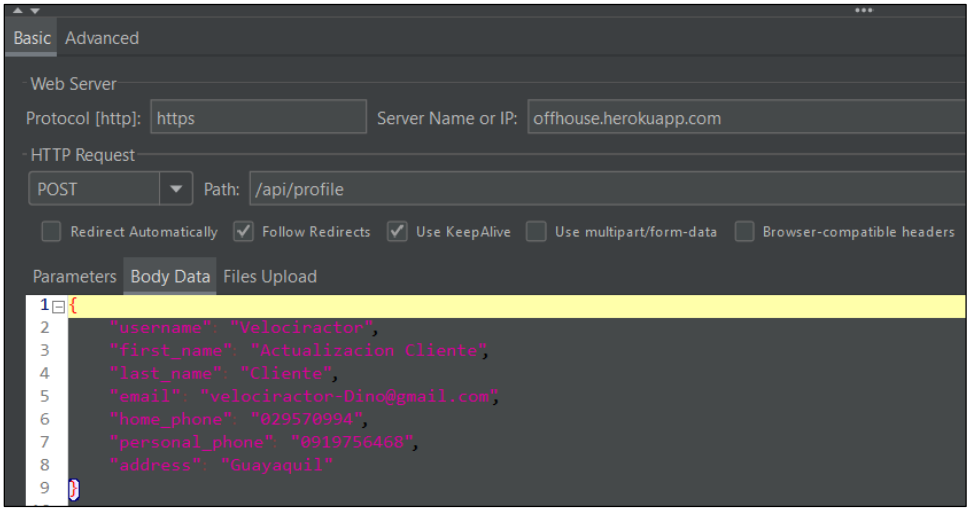


Fig. 64: Prueba de estrés en la historia de usuario N:3.

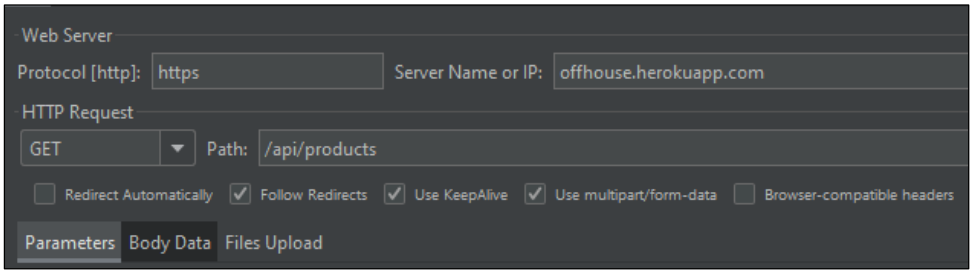


Fig. 65: Prueba de estrés en la historia de usuario N:4.

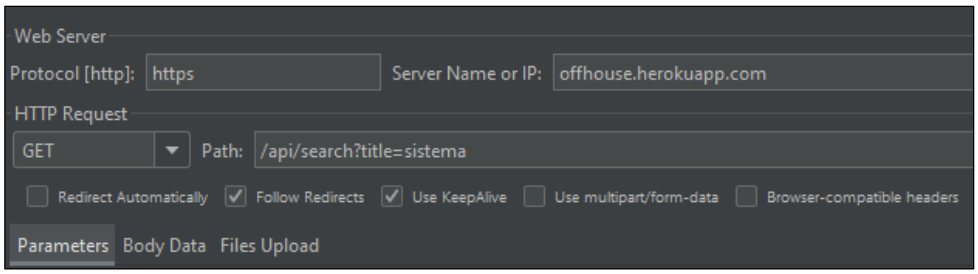


Fig. 66: Prueba de estrés en la historia de usuario N:5.

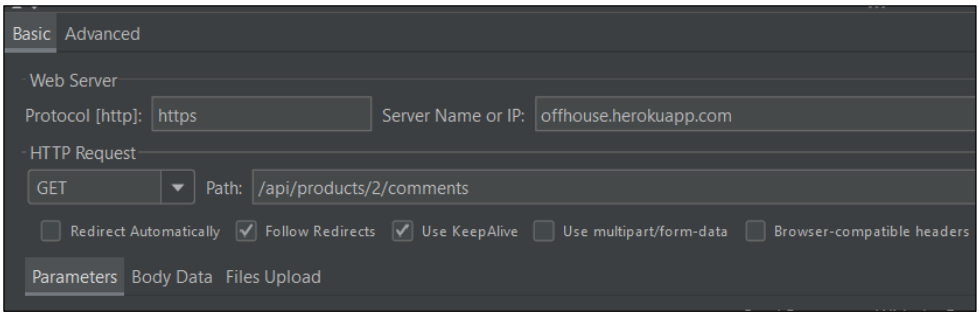


Fig. 67: Prueba de estrés en la historia de usuario N:6.

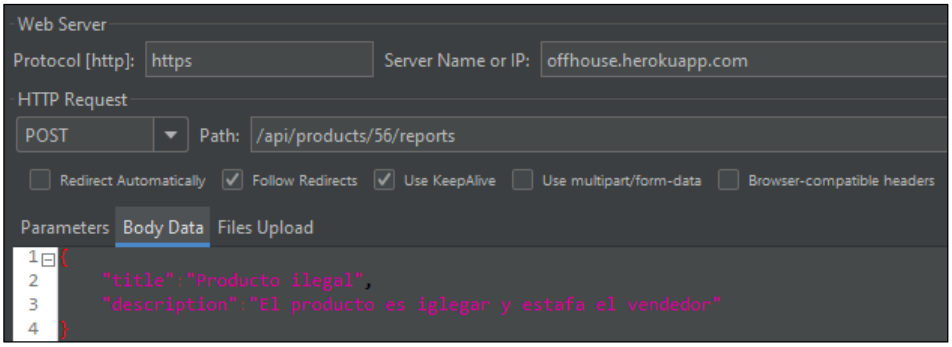


Fig. 68: Prueba de estrés en la historia de usuario N:7.

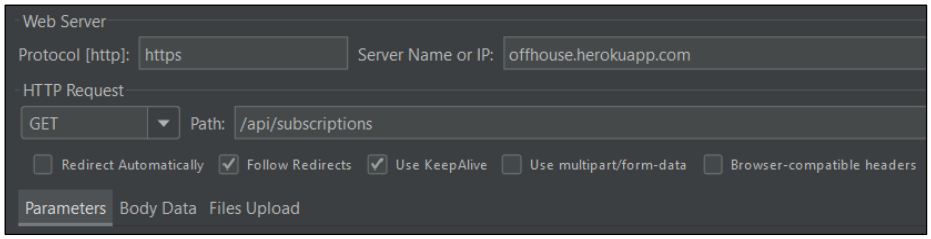


Fig. 69: Prueba de estrés en la historia de usuario N:8.

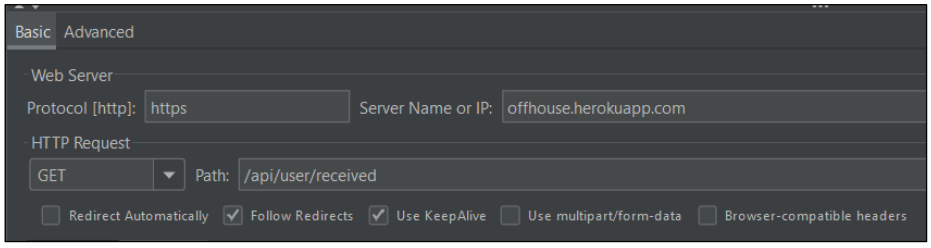


Fig. 70: Prueba de estrés en la historia de usuario N:9.

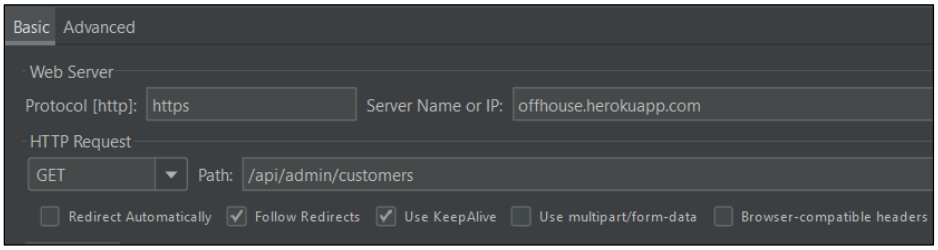


Fig. 71: Prueba de estrés en la historia de usuario N:10.

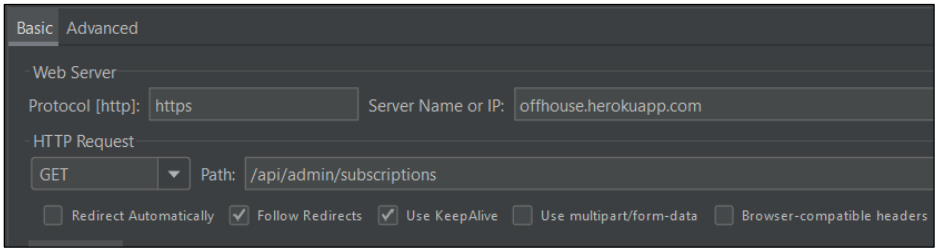


Fig. 72: Prueba de estrés en la historia de usuario N:11.

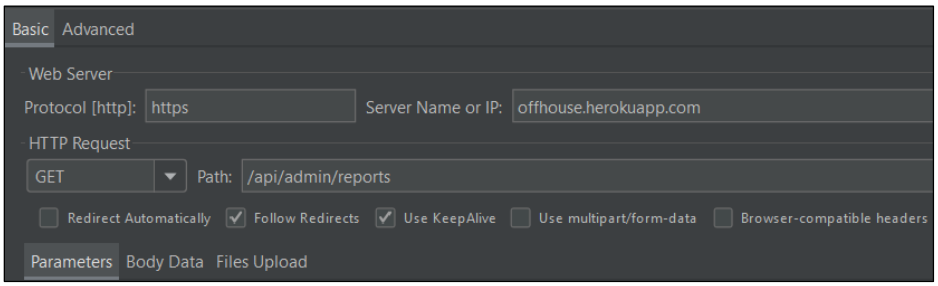


Fig. 73: Prueba de estrés en la historia de usuario N:12.

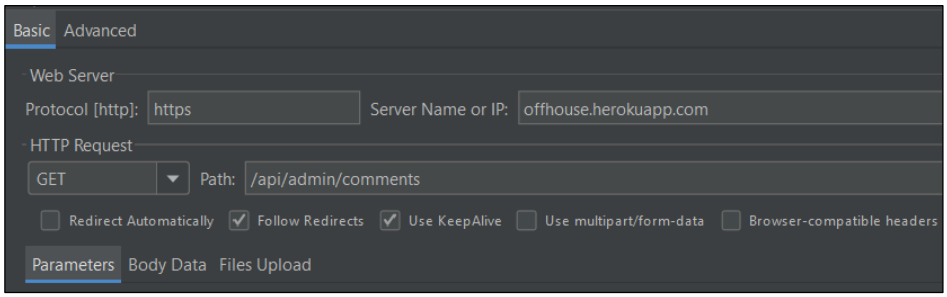


Fig. 74: Prueba de estrés en la historia de usuario N:13.

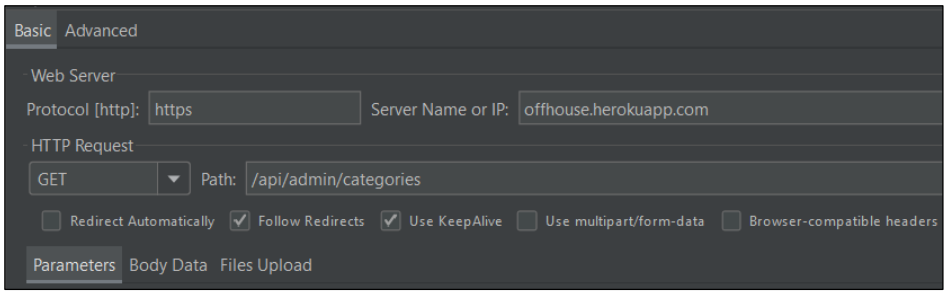


Fig. 75: Prueba de estrés en la historia de usuario N:14.

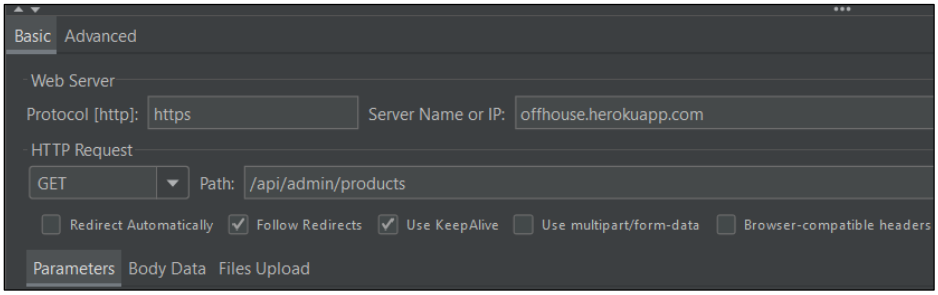


Fig. 76: Prueba de estrés en la historia de usuario N:15.

A continuación, en la **Fig. 77** se muestran los resultados obtenidos de las pruebas unitarias, para la prueba se realizó con un total de 60 hilos y con un periodo de aumento de 10segundos. Como resultado final de todas las pruebas se obtuvo un promedio de aumento del 8024 milisegundo, el tiempo de respuesta mínimo de 642 milisegundos y un máximo de 23073 milisegundos, todo esto con una tasa de error del 0.00%. Con ello se puede concluir que existe una correcta eficiencia respecto a la tasa de error, aunque es importante reducir el tiempo de respuestas en algunas rutas.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec
Register	60	873	652	2960	350.00	0.00%	5.7/sec	111.52	4.17
Login	60	775	642	1225	119.51	0.00%	5.7/sec	111.39	2.87
Profile	60	8847	2850	12558	2563.66	0.00%	3.2/sec	63.31	2.23
Products	60	9625	2536	17796	4210.37	0.00%	2.2/sec	166.78	0.44
Search Products	60	7598	1944	13392	3463.75	0.00%	2.6/sec	37.88	0.56
Comment	60	12614	2487	23073	6226.34	0.00%	1.9/sec	1.06	0.39
Report	60	13572	3714	18349	3972.67	0.00%	2.6/sec	50.83	1.45
Suscription	60	8662	1986	15493	4077.24	0.00%	2.4/sec	0.93	0.50
Chat	60	7557	1760	14072	3565.36	0.00%	2.5/sec	2.00	0.64
Gestionar Users	60	8704	1981	14993	3963.37	0.00%	2.4/sec	63.40	0.51
Gestionar Suscriptions	60	7853	1903	14760	3646.60	0.00%	2.4/sec	49.69	0.52
Gestionar Reports	60	7549	1815	13542	3467.01	0.00%	2.6/sec	12.81	0.54
Gestionar Products	60	9949	2180	17868	4354.04	0.00%	2.2/sec	185.97	0.45
Gestionar Categories	60	7795	1830	13988	3663.47	0.00%	2.5/sec	4.64	0.53
Gestionar Comments	60	8383	2446	14669	3725.36	0.00%	2.5/sec	99.12	0.51
TOTAL	900	8024	642	23073	4978.52	0.00%	23.1/min	9.01	0.12

Fig. 77: Resultados de las pruebas unitarias.

Pruebas de Carga

A continuación, en la **Fig. 78** se presenta las pruebas de carga para comprobar la eficientica del sistema, y ponerlo en producción.

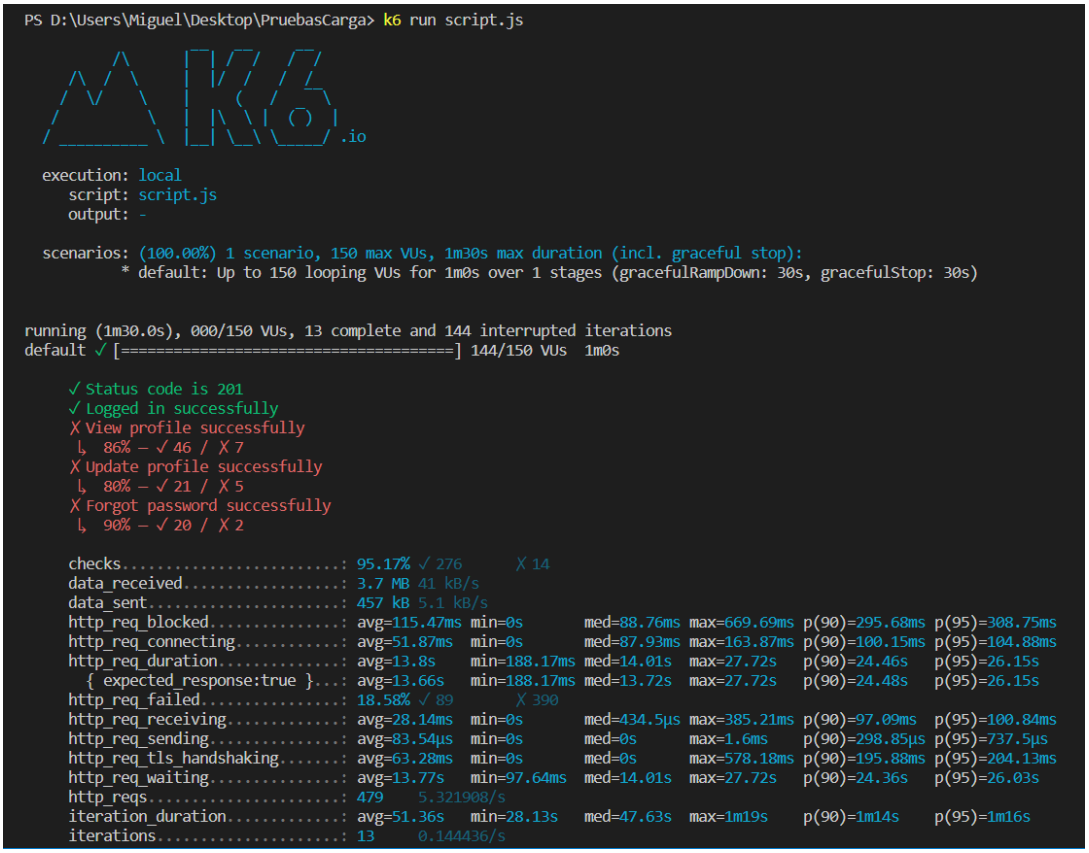


Fig. 78: Resultados de las pruebas de carga en el sistema.

Nota: En el proceso de desarrollo del proyecto, se elaboró un manual técnico detallando la recopilación de requerimientos, historias de usuarios, configuraciones del sistema y pruebas unitarias. Para mayor referencia, el documento se encuentra disponible en el siguiente enlace: ManualTecnico_Miguel_Muzo.pdf

ANEXO III

En esta sección de anexos se encuentra el Manual de usuario del componente backend, el enlace es el siguiente:

https://youtu.be/ohiaiFQjxCU

Se detalla cada una de las funcionalidades desarrolladas por el backend, y de los perfiles que intervienen en la misma, con una explicación más clara y precisa de entender.

ANEXO IV

A continuación, se muestra las credenciales para el acceso del backend, así como el repositorio de GitHub donde se encuentra el código fuente y el manual de instalación del sistema de forma local.

Credenciales de acceso

Para ingresar al sistema OFFHOUSE, se lo hace a través del siguiente enlace:

- **Backend**: https://offhouse.herokuapp.com
- **Frontend:** https://offhouse.vercel.app

A continuación, se presenta las credenciales de los perfiles del sistema:

- Perfil administrador:

  Email: admin_02@offhouse.com

  Contraseña: secreto

- Perfil cliente:

  Email: varela.patricia@example.net

  Contraseña: secreto

Para poder ingresar a *mailtrap* el cual ayuda a manejar correos electrónicos y para la función de recuperar la contraseña se presenta a continuación las credenciales:

Email: dastin.chavez@epn.edu.ec

Contraseña: Polloshermano22

Credenciales PayPal

Email: correoprueba@personal.example.com

Contraseña: prueba1234

Código fuente

El proyecto desarrollado por el componente backend se encuentra en un repositorio de GitHub, que se accede mediante el siguiente enlace:

https://github.com/Miguel-EMC/thesis_project.git