ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE ECOMMERCE PARA LA VENTA DE EQUIPOS COMPUTACIONALES

COMPONENTE BACKEND

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

MARLON ADRIAN TUQUERRES ROMERO

DIRECTORA: ALVAREZ JIMÉNEZ MAYRA ISABEL

DMQ, febrero 2023

CERTIFICACIONES

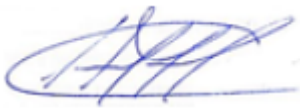
Yo, Marlon Adrian Tuquerres Romero declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

MARLON ADRIAN TUQUERRES ROMERO

marlontuquerres70@gmail.com

marlon.tuquerres@epn.edu.ec

Certifico que el presente trabajo de integración curricular fue desarrollado por Marlon Adrian Tuquerres Romero, bajo mi supervisión.



Ing. Mayra Isabel Alvarez Jiménez MSc.

DIRECTORA

mayra.alvarez@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.

Marlon Adrian Tuquerres Romero

DEDICATORIA

Dedico este proyecto con mucho cariño a mi persona, en el que ha sido de notable aprendizaje con tropiezos y prudencia. Por haberme mantenido en pie y con ánimo por un propósito futuro, siendo este un paso importante para seguir pavimentando mi aprendizaje por muchos años más.

Amado coraje y motivación.

TUQUERRES ROMERO MARLON ADRIAN

AGRADECIMIENTO

Me es grato agradecer a Dios por las circunstancias que se han ido presentando a lo largo de mi vida y acomodando conforme transcurren los años.

Quiero transmitir mi más grande agradecimiento a mi hermana Karylein, quien ha sido la persona que me ha aconsejado sobre mis propósitos, en los momentos difíciles está conmigo y/o me invita a salir a distraerme para que me sienta mejor.

A mis padres, que siempre están apoyándome en mis logros que he venido cosechando. A mi mamá, Fernanda, por confiar en mí desde hace mucho tiempo y ser la persona más tierna en cuanto a palabras para no decaer. Mi papá, Julio, por darme consejos para mantenerme seguro de lo que hago y que “No se pierde nada con intentarlo”.

Asimismo, mis sinceros agradecimientos a la comunidad de docentes de la Escuela de Formación de Tecnólogos, quienes fueron un motivo más del porqué me ha encantado este campo, con sus conocimientos, sugerencias y motivaciones.

Sin olvidar por supuesto a mi tutora, por haber sido parte de este proyecto, y ser una buena guía en la elaboración del mismo.

Finalmente, a una persona especial que aprecio y fue una razón más para terminarlo, debido a mi motivación de viajar a otro país.

TUQUERRES ROMERO MARLON ADRIAN

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Debido a la necesidad de gestionar y automatizar la información junto a los procesos de compra-venta de equipos de computación a través de un sistema web de un establecimiento nace la puesta en marcha de este proyecto. Actualmente una gran cantidad de locales comerciales regulares opta de un método manual para el proceso de compraventa, lo que implica el uso de recursos como personal, papel, entre otros, ya sea para promocionar artículos o para brindar información al cliente, lo que en muchas ocasiones no es óptimo ya que se llega a un número limitado de clientes.

La presente documentación detalla un componente Backend con el desarrollo de la venta de equipos computacionales, trayendo consigo la implementación de la metodología ágil SCRUM, dada su flexibilidad al cambio, obtención de requerimientos, roles definidos, etc. Esto permite el desenvolvimiento del proyecto en tiempos predeterminados. Asimismo, se utiliza el framework Laravel que está fundamentado por PHP, un lenguaje de programación, y al igual que su manejo con el gestor de base de datos relacional MySQL.

El proyecto tiene las opciones de realizar inicio de sesión acorde a roles, registro de usuarios clientes y propietarios, gestión de productos, restablecimiento de contraseñas mediante correo electrónico en caso de olvido de la misma. Además, existe la apertura de la gestión de comentarios, información personal y facilitación de stock disponible de productos conforme los clientes vayan haciendo pedidos. El componente desarrollado permite el almacenamiento y gestión de todos los registros existentes, en cualquier momento en que se requiera.

PALABRAS CLAVE: Laravel, API, Ecommerce, Backend, bases de datos, CRUD, software, sistema web, PHP.

ABSTRACT

Due to the need to manage and automate the information together with the processes of buying and selling computer equipment through a web system of an establishment, the start-up of this project arises. Currently, many regular commercial premises opt for a manual method for the buying and selling process, which implies the use of resources such as personnel, paper, among others, either to promote articles or to provide information to the client, which in many Sometimes it is not optimal since a limited number of clients is reached.

This documentation details a Backend component with the development of the sale of computer equipment, bringing with it the implementation of the agile SCRUM methodology, given its flexibility to change, obtaining requirements, defined roles, etc. This allows the development of the project in predetermined times. Likewise, the Laravel framework is used, which is based on PHP, a programming language, and likes its management with the MySQL relational database manager.

The project has the options to log in according to roles, customer and owner user registration, product management, password reset by email in case of forgetting it. In addition, there is the opening of the management of comments, personal information, and provision of available stock of products as customers place orders. The developed component allows the storage and management of all existing records, at any time it is required.

KEYWORDS: Laravel, API, Ecommerce, Backend, databases, CRUD, software, web system, PHP.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Cada año que pasa la tecnología se vuelve más esencial en la vida del ser humano, hace un par de años atrás un cliente tenía que acercarse a una tienda a realizar una compra, no obstante, hoy en día ya no necesita de esa acción, está a tan solo un clic de obtener un producto deseado. “La tienda física ya no se debe enfocar en ser el único medio para la venta, es necesario tener presencia online, ser parte del movimiento que crece exponencialmente que es el *Ecommerce* desarrollando tu propia tienda online” [1].

Se denota la importancia de la elaboración de un aplicativo *web* para la venta de productos, ya que no solamente se adquiere productos en un establecimiento físico; centro comercial o local. En la ciudad de Quito-Ecuador, varios centros comerciales ya hacen uso de esta herramienta dando como resultado la comodidad y ahorro de tiempo del usuario en adquirir un producto sin que tenga la obligación de salir de casa. Además, el *Ecommerce* se caracteriza por ser una actividad económica que posibilita el crecimiento del comercio de varios productos independientemente de la ubicación geográfica de los potenciales clientes. Por tal motivo, el desarrollo de sitios *web* tiene gran acogida en el campo de la compra y venta de productos e incluso como apoyo para emprendimientos futuros.

Hoy en día existen varias aplicaciones de venta *online* como: Mercado Libre, Linio, etc., cada uno de ellos tiene como particularidad la facilidad de compra, fotografías del producto a detalle, entre otros. No obstante, la aplicación que se propone desarrollar se distingue por la publicación de comentarios por parte de los clientes hacia los productos promocionados por los propietarios y si existe algún comentario inapropiado el administrador puede dar de baja o *banear* a un cliente con toda su información, asimismo con un propietario junto a su negocio. La presentación de cada una de las implementaciones será de interés para cualquiera de ellos, por ejemplo; promocionar, administrar y gestionar, productos. El contacto directo mediante el número de celular del propietario por parte del cliente cuando adquiera un producto.

De este modo, el objetivo del presente proyecto es proveer al público un sistema *web* que tenga como enfoque la compra de equipos computacionales a través de una plataforma de *Ecommerce*, para ello, se realizará un componente de tipo *Backend* que permita la correcta implementación del sitio *web* para que sean ejecutadas apropiadamente por el usuario final.

Además, la aplicación dispone varios tipos de usuarios para una mejor operatividad de los mismos en el sitio *web*, dichos tipos de usuarios son los que previamente se había descrito; Administrador del sistema, perfil propietario y perfil cliente. Igualmente, se tomarán en

cuenta aspectos de importancia como la protección de información, correcta actualización de la base de datos cuando se realice un pedido con respecto al *stock*, además con las funcionalidades que dispondrán los usuarios.

1.1 Objetivo general

Desarrollar un *Backend* de un *Ecommerce* para la venta de equipos computacionales.

1.2 Objetivos específicos

1. Establecer cada uno de los requerimientos funcionales y no funcionales con la finalidad de la elaboración del *Backend*.

2. Esquematizar el modelo para la base de datos relacional.

3. Codificar los *endpoints* basándose en los requerimientos previos, con el fin de la elaboración continua.

4. Verificar los respectivos *endpoints* para su correcto funcionamiento.

5. Desplegar el componente del *Backend* a producción para la disponibilidad de su uso.

1.3 Alcance

La elaboración del proyecto tendrá implementado el desarrollo del componente de un *Backend* de *Ecommerce* para la venta de equipos de cómputo. La aplicación *web* dispondrá de varios perfiles: administrador, propietario y cliente. Cada uno de ellos cumplirá un rol y tareas en específico, como se lo detalla a continuación:

- Administrador global del sistema: Encargado de estar al tanto de los productos promocionados y comentarios de clientes, gestiona los perfiles de los demás usuarios en base a lo mencionado.
- Perfil Propietario: Usuario capaz de gestionar el contenido de su sitio *web* con los productos de su negocio.
- Perfil Cliente: Podrá adquirir productos que se encuentren ofertados por los negocios, publicar comentarios u opiniones de algún producto existente.

La aplicación *web* proveerá una sección de calificación y comentarios por producto, así los usuarios pueden tomar una decisión de compra en base a opiniones de los demás compradores. Por otra parte, a los propietarios se les proporcionará la actualización sobre el número de *stock* disponible de los productos que ellos oferten en base a la cantidad que

realicen los clientes en un pedido, así podrán tomar decisiones futuras sobre la adquisición de más productos. Si existe un producto o comentario no apropiado publicado en el sistema, el administrador obtendrá la información de cualquiera de los usuarios para ponerse en contacto para que lo modifique, y en el caso de que persistiera entonces se lo elimina del sistema de *Ecommerce* con todas sus interacciones existentes.

Asimismo, se garantizará el uso de herramientas de desarrollo modernas para su correcta escalabilidad, integridad y consistencia de los datos. Finalmente, *Scrum*, la metodología ágil de desarrollo estará incorporada en el proyecto para el cumplimento de los objetivos propuestos, con una serie de pruebas para verificar el correcto funcionamiento del *Backend*, llevados consigo al despliegue a producción.

1.4 Marco teórico

En este punto se aborda los conceptos globales de los componentes adicionales y la metodología a aplicarse en el desarrollo del presente Trabajo de Integración Curricular, así como cada una de las herramientas y metodologías que serán consideradas para el desarrollo del *Backend*.

Ecommerce

Un *Ecommerce*, conocido como comercio electrónico (traducción del inglés), es la acción de realizar una compra y/o venta de productos por medio de internet, y es muy común hoy en día por varias tiendas online. Comúnmente no solo se ofrecen productos, ya que también hay empresas que promocionan servicios dando apertura a la posibilidad de variar sus productos con alguna marca. Tienen como características exhibir productos populares, fotos del mismo, opiniones de otros usuarios sobre alguna publicación, navegación cómoda, entre otras.

Por consiguiente, muchos de los aplicativos de *Ecommerce*, poseen un *Backend*, este término se caracteriza porque es la parte del desarrollo en la que el programador se le da la tarea de trabajar en la lógica, para que una página *web* tenga un funcionamiento correcto, por ejemplo, la comunicación del lado del servidor. También, la persona encargada de elaborar el *Backend* tendrá la obligación de asegurar la información y que esta sea transmitida de manera segura, como se describió anteriormente, este componente también accede al servidor, junto al desempeño del producto de modo que no arruine la experiencia de usuario.

Metodología ágil

Es así como, cada vez que existe la implementación y/o desarrollo de una aplicación es fundamental seguir alguna metodología ágil para su correcto orden y ejecución cuando termine el proceso de elaboración. En general, en el desarrollo de *software* se encuentran las metodologías ágiles que se orientan a proporcionar pequeños trozos de *software* en funcionamiento en el menor tiempo posible, con el objetivo de aumentar la satisfacción del usuario final [2].

Se denota a continuación la **TABLA I,** cada una de las metodologías más utilizadas.

TABLA I: Tabla comparativa de metodologías ágiles

	SCRUM	***Extreme Programming***	***Kanban***
¿Qué es?	Proceso en el cual se aplica en trabajo colaborativo un conjunto de prácticas óptimas, con el objetivo de generar en lo posible un buen proyecto de desarrollo de *software*.	Es una metodología de desarrollo en el campo del software orientada a la mejora de calidad y la capacidad de un producto de *software* en respuesta a los requisitos del cliente.	Método visual dirigido al procesamiento y gestión de un trabajo. Previniendo la acumulación de tareas pendientes.
Ventajas	La responsabilidad recae a alguien con experiencia (*Product Owner y Scrum Master*). Roles asignados son predefinidos.	No existen iteraciones. Los códigos de estándar si son utilizados	Iteraciones cortas. Sin estándar de codificación
Desventajas	Iteraciones (*Sprints*) sin modificación futura.	No posee roles prescritos. Responsabilidad grupal del producto. El encargado del proceso es el equipo de trabajo.	No posee roles prescritos. Responsabilidad grupal del producto. El encargado del proceso es el equipo de trabajo.

La metodología *Scrum* tiene como parte de sus artefactos los *Sprints*, este término hace referencia a un periodo de tiempo fijo y breve en donde un equipo efectúa su respectiva

labor para completar una cantidad de trabajo determinada. En este caso, la aplicación web presentará varios *Sprints* para que haya un orden y legibilidad del código al finalizar el proyecto.

En consecuencia, para el proyecto a desarrollarse existe un beneficio y es que se tiene la oportunidad de contar con una persona guía en la elaboración del mismo, lo que agiliza su trabajo en esta ocasión. Es por ello por lo que, de las metodologías descritas anteriormente, la aplicación web a desarrollarse tendrá como implementación *Scrum*, dado que es una de las más agiles y óptimas para un proyecto de desarrollo de *software*, que lleva consigo procesos de control empíricos.

Framework

Es muy común el uso de alguna estructura previa aprovechando librerías e implementaciones para la elaboración de una aplicación, esa estructura se la conoce como *Framework*. En la actualidad se pueden encontrar varios de ellos, tal es el caso de *Laravel*, basado en el lenguaje de programación de PHP. *Laravel* otorga a su desarrollador herramientas robustas para pruebas unitarias, eventos en tiempo real, inyección de dependencias, etc. Igualmente, *Laravel* tiene integrado un buen soporte para sistemas de caché, de la misma manera que dispone de una comunidad con miles de desarrolladores.

Lenguaje de programación

Un lenguaje de programación se conoce como a la agrupación de instrucciones en el cual una computadora puede tener interacción con los humanos [3]. A continuación, se muestra la **TABLA II**, en donde se encuentran algunos lenguajes de programación más útiles.

TABLA II: Ejemplos de lenguajes de Programación

	C++	**PHP**	**HTML**
¿Qué es?	Lenguaje de programación que está orientado a objetos.	• Es un lenguaje de programación interpretado. • Desde su origen fue diseñado hacia la creación de páginas web que sean dinámicas.	Lenguaje que se utiliza en la creación de páginas web.

Ventajas	• En lo que respecta a sistemas complejos resulta un lenguaje muy potente. • Se puede volver a utilizar el código.	• Aplicado para el desarrollo de aplicaciones web dinámicas. • Facilidad de acceso a la información que se encuentra alojada en una base de datos. • Adaptabilidad con el lenguaje HTML.	• Lo admiten todos los exploradores. Archivos pequeños. • Se acopla adecuadamente a lenguajes de *Backend*.
Desventajas	• No es útil en la creación de sitios *web*. • Visualmente no es la mejor opción. • Las librerías tienen un manejo poco cómodo para el desarrollador.	• Es necesario un servidor *web*. • Aprendizaje de un *framework*.	• En distintos navegadores la interpretación puede ser distinta. • Lenguaje estático. • Cuando se trata de archivos pesados resultará lenta.

Considerando la utilidad del lenguaje de programación PHP, se hará su respectivo uso para el desarrollo del proyecto de *Ecommerce*. Se distingue con el trabajo que tiene a lado del servidor, por la facilidad de acceso a la información que se encuentra almacenada en una base de datos. Fuertemente utilizado para proyectos de desarrollo *web* e interfaces gráficas de usuario (GUI) [4].

Gestor de base de datos

Cada *Backend* deberá estar sujeto a una gestión de bases de datos como sistema, por ende, MySQL porta con dicha característica acoplado a PhpMyAdmin (Aplicación *web* que funciona en la administración de bases de datos). En la **TABLA III**, se puede evidenciar algunas de las características donde se hace uso de esta tecnología.

TABLA III: Características de PhpMyAdmin

Características de *PhpMyAdmin*
Importación de datos con formato de archivos CSV y SQL.
Exportación de archivos hacia varios formatos, ya sean CSV, PDF, XML, SQL, etc.
Interfaz *web* cómoda para el usuario

Soporte para MYSQL • Realizar operaciones básicas en un conjunto de datos: CRUD • Administración de procesos almacenados • Mantenimiento de servidor

Railway

Al no ser una página estática no sería optimo alojar un proyecto en plataformas como *GitHub Pages* o similares. Es por ello por lo que la elaboración de este proyecto estará alojada en la plataforma de *Railway*.

Railway, brinda un servicio de computación para el alojamiento en la nube de aplicaciones, llevando consigo un vasto soporte para las aplicaciones y varios lenguajes. De la misma manera, se puede implementar en la nube una infraestructura local. Esta incorporación logra la conexión sin percances a las bases de datos *Mongo*, *Redis*, *MySQL* o *PostgreSQL* alojadas en *Railway* [5].

2 METODOLOGÍA

Cada vez que se llega al tema de metodología se tiende a pensar sobre cada tipo existente; metodología de investigación, tradicional, ágil, entre otros. No obstante, cada uno de ellos tiene algo en común, son disciplinas encargadas de estudiar varios métodos o técnicas para trabajos e investigaciones científicas con el objetivo de alcanzar una meta planteada. Asimismo, cada método está regido a una metodología basada en una teoría comparativa, normativa y descriptiva sobre el método, añadiendo el accionar del investigador [6].

Por tal motivo, en este proyecto no habrá la excepción de su uso, se implementará la metodología ágil *Scrum* con el objetivo de verificar el funcionamiento del componente mediante una serie de pruebas, cada una de estas poseerá dos tipos de resultados; exitoso o no exitoso. De igual modo, las evaluaciones serán de ayuda para modificar y enmendar posibles errores, si es que fuese el caso. Esto lleva consigo la constante comprobación a lo largo del desarrollo del componente de *Backend* para su correcto funcionamiento del sitio *web*.

2.1 Metodología de Desarrollo

En los métodos de información tiene cabida el proceso de desarrollo, por ende, en el campo del desarrollo de *software* una metodología está encargada de llevar una planificación, control y estructuración de un marco de trabajo. A lo largo de los años, cada vez han surgido y evolucionado distintos marcos de trabajo, trayendo consigo debilidades y fortalezas. Sin embargo, cualquiera que fuese la metodología de desarrollo de un sistema no debería ser implementada en todos los proyectos que vayan a realizarse. Deberá tener presente las consideraciones del proyecto y del equipo de desarrollo, ya sean organizacionales y/o técnicas [7].

Por consiguiente, el desarrollo del componente de *Backend* en el presente trabajo pondrá en práctica la metodología ágil *Scrum*, de modo que se describirá cada una de sus implementaciones en el respectivo componente.

Roles

En la elaboración del aplicativo *web* se van a abordar los respectivos roles que plantea *Scrum,* a manera de modelo a seguir se representan por: *Product Owner*, *Scrum Master y Development Team*. Seguidamente se detalla los correspondientes roles.

***Development Team* (Equipo de desarrollo)**

En cada proyecto de desarrollo de *software* siempre habrá un equipo encargado de la implementación de las tareas a realizar, en donde cada uno poseerá avances notorios para la operatividad del producto. Nótese en la **TABLA IV**, en el que se presenta los roles asignados para la elaboración del proyecto de *software*, junto a la persona responsable.

TABLA IV: Roles asignados en el proyecto de software

ROL	INTEGRANTES
Product Owner	Ing. Alvarez Mayra
Scrum Master	Ing. Alvarez Mayra
Development Team	Tuquerres Marlon

Product Owner

En la **TABLA IV**, se denota el rol *Product Owner*, quien es la persona que estará a cargo de guiar al equipo de trabajo para el cumplimiento de las tareas con las funcionalidades del proyecto.

Scrum Master

La persona que explicará las reglas y prácticas que caracteriza a la metodología *Scrum*, ya sea o no que sepan de la misma los integrantes de trabajo. Además, el *Scrum Master* tiende a estar en contacto junto al *Product Owner*, con el objetivo de presentar cambios, avances e ideas para posteriormente hacérselo saber al equipo. En la **TABLA *IV***, se muestra la persona que está a cargo de dicho rol.

Artefactos

En lo que abarca este término en *Scrum,* hace referencia a la manera en el cual la información es manejada, de tal forma que toma como prioridad la transparencia que permite el desarrollo de las actividades adecuadamente. En otras palabras, son los elementos que caracterizan el trabajo y aportan a lo largo del desarrollo del proyecto. A continuación, se presentan dichos elementos que se utilizarán por parte de *Scrum.*

Recopilación de requerimientos

- Los requerimientos dentro de un proyecto de *software* son las especificaciones de lo que se debe hacer en un programa o producto. Estos actuarán como parte de unas pautas con lo que el equipo de desarrolladores creará el programa funcional, con el objetivo de satisfacer la necesidad del usuario final [8]. A continuación, se presenta la **TABLA V**, donde se aprecia una fracción de los requerimientos a implementarse a lo

largo de la elaboración del *Ecommerce*. El resto de los requerimientos se muestran en la sección de ANEXOS.

***TABLA V*:** Modelo para la recolección de requerimientos

Recolección de Requerimientos			
ID	**Nombre**	**Descripción**	**Usuario**
RR-001	Ingreso al sistema	A fin de acceder al sistema, cada uno de los usuarios debe estar vigente con las apropiadas credenciales.	Todos los usuarios
RR-002	Observación y eliminación de todos los usuarios	El Administrador global gestiona a los demás usuarios mediante la eliminación y observación para llevar un control de los mismos.	Administrador

Historias de Usuario

Esta denominación se caracteriza por poseer una justificación y/o descripción ya sea informal o general de alguna función de *software*, que es descrita a partir de la perspectiva de un cliente o usuario final. El objetivo de esta implementación es lograr la articulación de cómo se va a entregar un valor particular al cliente por medio de un elemento de trabajo [9]. Un ejemplo de ello se describe a continuación, **TABLA *VI***. Para más detalle de las historias de usuario diríjase al ANEXO II.

TABLA VI*:* Historia de usuario - Ingreso al sistema

Historia de Usuario	
Identificador (ID): HU001	**Usuario**: Todos los usuarios
Nombre de historia: Ingresar al sistema	
Prioridad: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: La ejecución de esta acción permite el ingreso de los usuarios clientes.	
Observación: Cada uno de los respectivos usuarios podrá incorporarse al sistema según su interés.	

Product Backlog

Otro término que se emplea en la guía *Scrum* es el *Product Backlog,* este consiste en la preparación de todas las tareas a realizarse a lo largo del desarrollo de un producto de *software*, esto con el objetivo de la visibilidad del mismo hacia todo el equipo.

En la **TABLA *VII***, se denota la correspondiente implementación que tendrá su uso a lo largo del desarrollo del componente. Véase en el ANEXO II., toda la recopilación de las mismas.

TABLA VII: Modelo para el listado - *Product Backlog*

ID-HU	Historia de Usuario	Prioridad	Iteración	Estado
HU-001	Ingresar al sistema	Media	1	Pendiente
HU-002	Consultar y eliminar usuarios	Media	1	Pendiente

Sprint Backlog

Por otra parte, todo el trabajo necesario a realizarse se verá representado mediante un tablero de tareas, en el que se debe lograr el compromiso que se había hecho con anterioridad junto al *Product Owner* [10]. Básicamente, son tareas identificadas que se encuentran figuradas en una lista. Nótese en la **TABLA *VIII***, en donde se denota el formato a emplear para cada uno de los *Sprint.* De la misma manera, más adelante se encuentra a más detalle cada una de las tablas en el ANEXO II.

TABLA VIII: *Sprint Backlog* – Formato

ID-SB	Nombre	ID-HU	Historia de Usuario	Tarea	Tiempo estimado
SB-000	Configuración del entorno de desarrollo	N/A	N/A	Puesta en marcha del proyecto en *Visual Studio Code*. Elaboración de la base de datos MySQL Creación de flujo del sistema (diagrama)	20H

2.2 Diseño de arquitectura

En cada proyecto de *software* es primordial seguir un diseño de arquitectura, pues, este utilizará en su mayoría los conocimientos de programación a fin de planear un diseño general, de manera que se delimite un panorama general y comenzar con una elaboración de un prototipo.

Patrón arquitectónico Modelo Vista Controlador (MVC)

Un patrón MVC es muy empleado hacia el diseño de *software*, en especial para la implementación de interfaces de usuario, lógica de control o interfaces de datos. Destaca la separación entre la visualización y la lógica de negocios. Esto genera una mejora para su mantenimiento y la división de trabajo [11].

Se denota arquitectura del patrón a seguir para la elaboración del *Ecommerce* en la **Fig. 1**, que alude en la existencia de un usuario que hará uso de un controlador y todo lo que se vaya actualizando se verá reflejado en la vista. Asimismo, existe un modelo que se verá manipulado por el controlador, y a su vez le notificará de alguna manipulación.

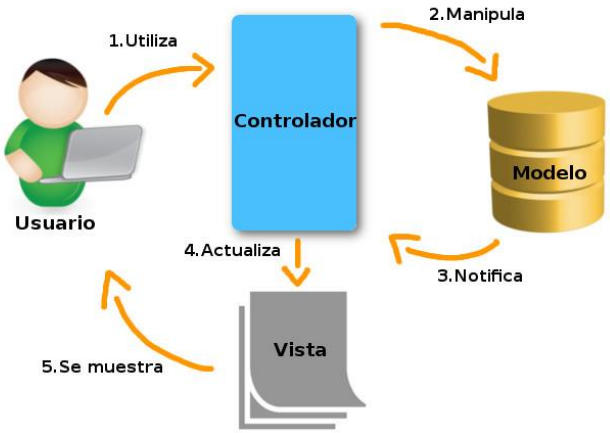


Fig. 1: Patrón arquitectónico MVC

2.3 Herramientas de desarrollo

Las herramientas de desarrollo posibilitan la gestión y el buen rendimiento de las tareas diarias de los datos a través otros *softwares* o aplicaciones. Un buen desarrollador de *software* siempre debe tener en cuenta qué clase de herramientas tiene que utilizar [12]. Por ende, la **TABLA IX**, tiene el registro de las herramientas seleccionadas y cómo será su aporte en la elaboración del *Ecommerce*.

TABLA IX: Registro de Herramientas para la elaboración del proyecto

Herramienta	Justificación
Composer	Gestor de dependencias del lenguaje de programación PHP. Resulta útil debido a la instalación y mantenimiento de las librerías de forma sencilla y rápida, y esto agiliza al momento de desarrollar una aplicación basada en PHP.
Git	Git posee una gran capacidad de realizar fáciles ramificaciones, brindando un flujo de trabajo entre varios usuarios de Git. Esto lleva consigo permitir, realizar, administrar y actualizar cambios en las ramas, con respecto a su código base.
GitHub	Una herramienta gratuita como *GitHub* permite tener el acceso a repositorios sin importar el equipo, debido a que se encuentran alojados en la nube. Además, está integrado con *Git*.
Laravel	*Framework* que contribuye en la elaboración de una aplicación, mediante sus sistemas de paquetes, junto al modelo MVC (Modelo-Vista-Controlador). Asimismo, emplea el formato JSON, lo que genera una mejor optimización por el lado tanto del cliente y servidor para realizar consultas.
MySQL	Sistema gestor de bases de datos y está basada en *Open Source* (código abierto). Se desenvuelve con bases de datos relacionales, debido al múltiple uso de tablas interconectadas que están almacenando información.
Postman	Es una plataforma gratuita que permite hacer uso de *APIs*, además de realizar pruebas de las mismas para corroborar el correcto funcionamiento de un desarrollo *web* orientado a *Backend*.
Railway	Actualmente, muchos servicios de alojamiento poseen su respectivo despliegue con opciones de pago. No obstante, una herramienta gratuita que posibilita el despliegue de una aplicación *Backend* o *Frontend* es *Railway*. Posee una flexibilidad para realizar el desarrollo de una infraestructura local con el respectivo despliegue a la nube.

3 RESULTADOS

Con el objetivo de detallar un resumen de las acciones por cada uno de los *Sprints* realizados, se muestra esta sección, en la que se adjunta sus respectivas pruebas de *API* e ilustraciones.

3.1 *Sprint* 0. Planificación del proyecto

De acuerdo a cada planificación de un proyecto, este poseerá su etapa de arranque correspondiente, por lo que el *Sprint* 0 está enfocado en la configuración del ambiente de desarrollo, y lleva consigo los requerimientos de cada uno de los roles de los distintos clientes. De este modo, se presenta la sistemática de los objetivos del *Sprint* a desarrollarse en esta etapa inicial:

- Estructuración del proyecto.
- Levantamiento de requerimientos.
- Elaboración del *Product Backlog*.
- Creación del proyecto de *Laravel.*
- Prueba unitaria *Sprint* 0.

Estructuración del proyecto

Generar en un sistema una guía de flujo resulta una implementación impecable para representar las decisiones y acciones. Por tal motivo, se tiene que separar los procesos por roles para concretar quiénes son los usuarios encargados del proceso con su respectivo flujo en base a su rol, nótese en la **Fig. 2**.

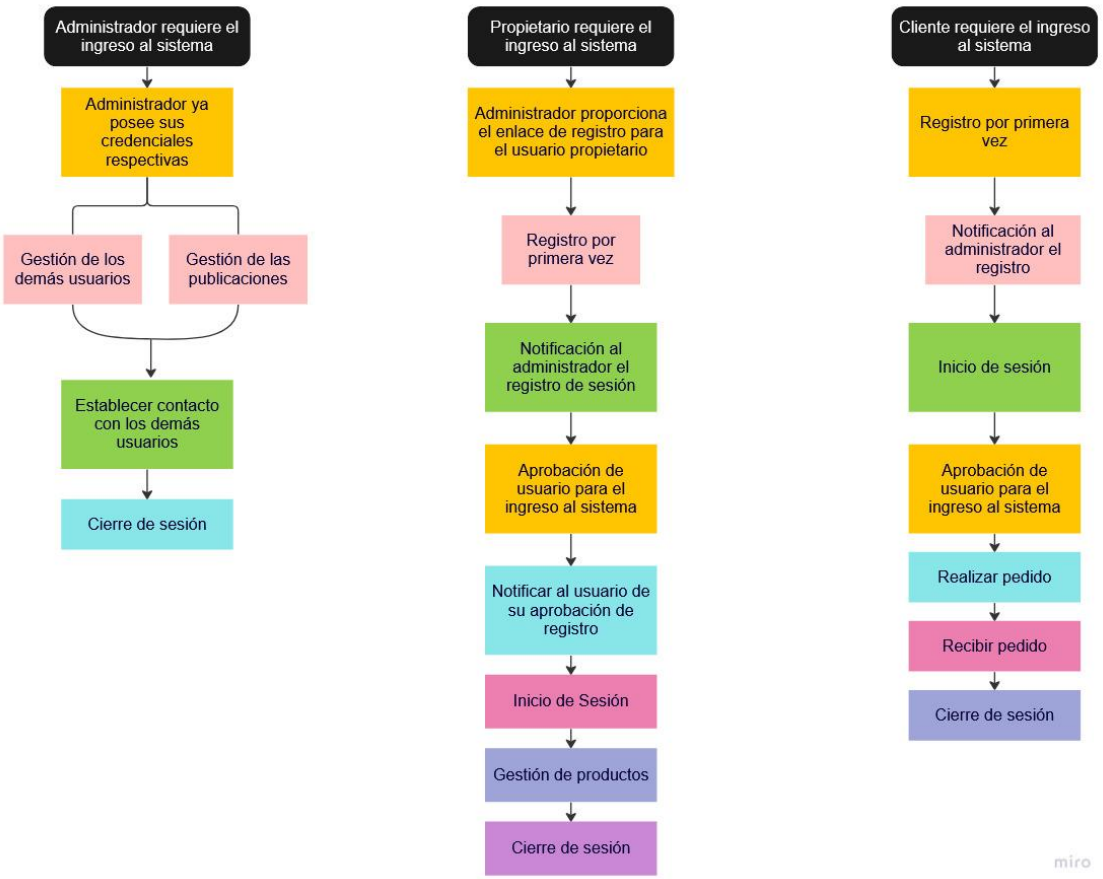


Fig. 2: Diagrama de flujo - proceso con el rol de cada usuario

Levantamiento de requerimientos

Dado que existen varios locales con distintos propietarios se llega a tomar en cuenta cada una de las necesidades que podrían tener en mente cualquier dueño del local. Así, estas deben ser resueltas por parte del aplicativo *Backend*, encargado de la lógica del negocio. De la misma manera, serán de gran utilidad las historias de usuario para la instauración de funcionalidades, así como un requerimiento de búsqueda o adquisición de un producto por el usuario cliente, o la necesidad del propietario en buscar algún producto de su negocio.

Elaboración del *Product Backlog*

Siempre que se realiza un proyecto de *software* es necesario contar con un método guía para la elaboración del mismo. Por tal motivo, la realización del *Product Backlog* en este componente de *Backend* será establecer *Sprints* o etapas de desarrollo, que requerirán ser solventadas en base a su demanda y complejidad de construcción. Cada uno de los archivos del proyecto están gradualmente contenidos hacia el repositorio.

Creación del proyecto de *Laravel*

A fin de crear y configurar el proyecto de *Ecommerce*, se estableció la correcta instalación y/o verificación de las herramientas, como lo son:

- *Visual Studio Code* como editor de código.
- *Composer*, encargado en llevar la gestión de dependencias.
- *MySQL PhpMyAdmin*, en cuanto a base de datos con su respectiva gestión.
- *Postman*, con el objetivo de pruebas de rutas.

Una vez terminado y verificado cada una de las herramientas, se procede a la creación del proyecto con los respectivos comandos en base a la documentación de *Laravel-Framework*, **Fig. *3***

```
PS C:\Users\MATR\Desktop> composer create-project laravel/laravel Ecommerce
```

Fig. 3: Comando para la creación de un proyecto

Prueba unitaria *Sprint* 0

Como ya se mencionó anteriormente, se realizó la creación de un nuevo proyecto de *Backend* en *Laravel*, es por ello por lo que en la **Fig. *4***, se denota la estructura del proyecto creado.

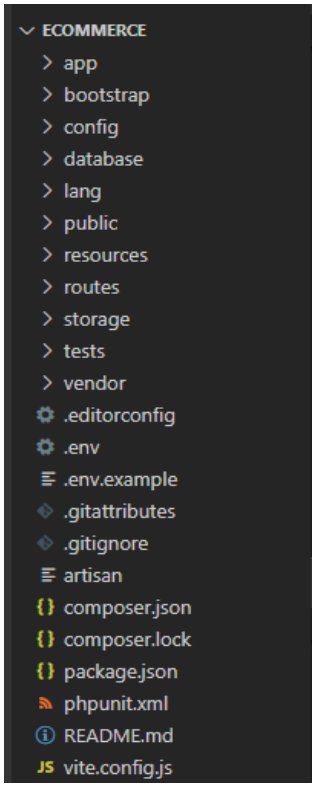


Fig. 4: Estructura del proyecto de *Laravel*

3.2 *Sprint* 1: Diseño de arquitectura y base de datos para el sistema

En base a la creación de *Sprint* del *Product Backlog*, se lleva a cabo la identificación de procesos necesarios para acceder, identificar y gestionar los datos del sistema *web* [13]. De tal manera, las actividades establecidas son:

- Diagrama de flujo.
- Modelo de base de datos.
- Creación del repositorio.
- Prueba unitaria *Sprint* 1.

Diagrama de flujo

En la **Fig. *5***, se presenta a cada uno de los tipos de usuario con sus respectivos requerimientos juntados, guiando al *Frontend* (en el caso de que se lo implemente) de la manera en la que puede hacer uso de los recursos de la *API*.

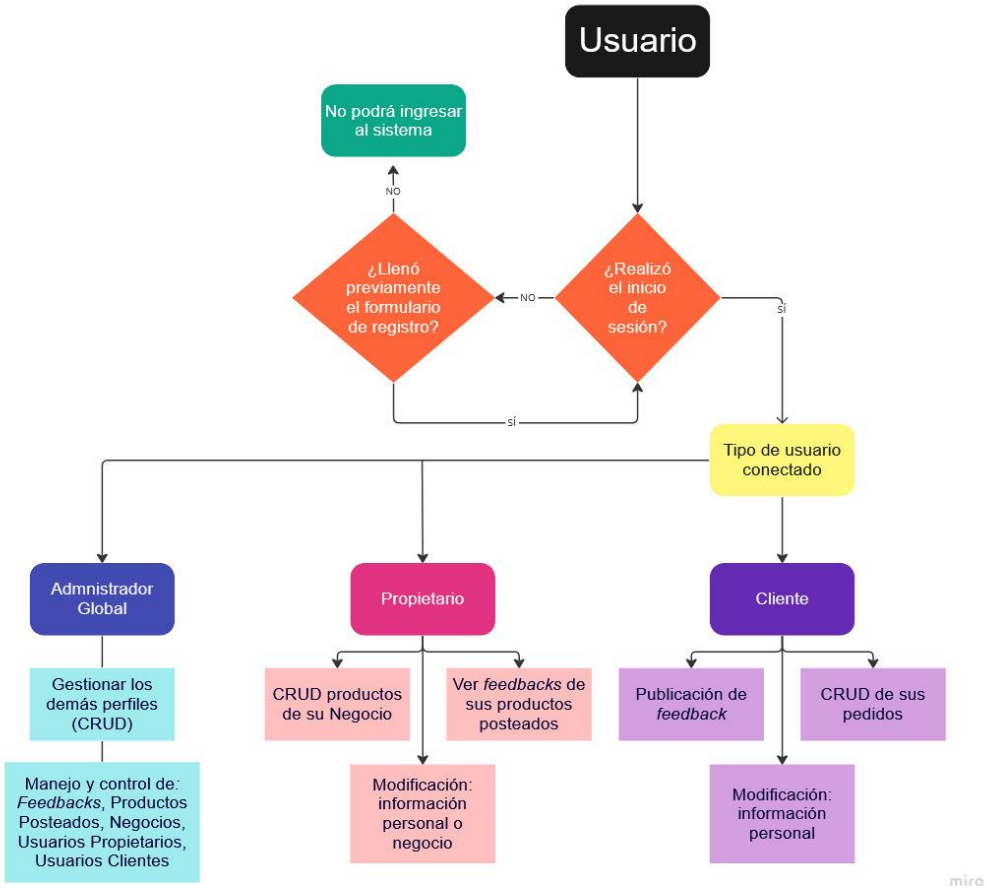


Fig. 5: Diagrama de flujo

Modelo de base de datos

En cada proyecto de este campo es necesario contar con un modelo, puesto que en base a este se podrá seguir de cerca cómo trabaja una base de datos, y en el que se almacenarán cada uno de los datos del sistema de *Ecommerce*. Así, se dará respuesta a futuras consultas y/o peticiones efectuadas a la *API*. Por tal motivo, para el desarrollo de la base de datos se implementó el DBMS (Sistema Gestor de Base de Datos) llamado *MySQL*, porque posee una intuitiva interfaz *web*, soporte para la mayoría de las características del DBMS, y herramientas para el modelado e integración de los datos requeridos en el sistema [14]. Ejemplo de ello, se muestra la **Fig. 6**, conocida como el diagrama de entidad-relación, además, contiene sus correspondientes relaciones y claves.

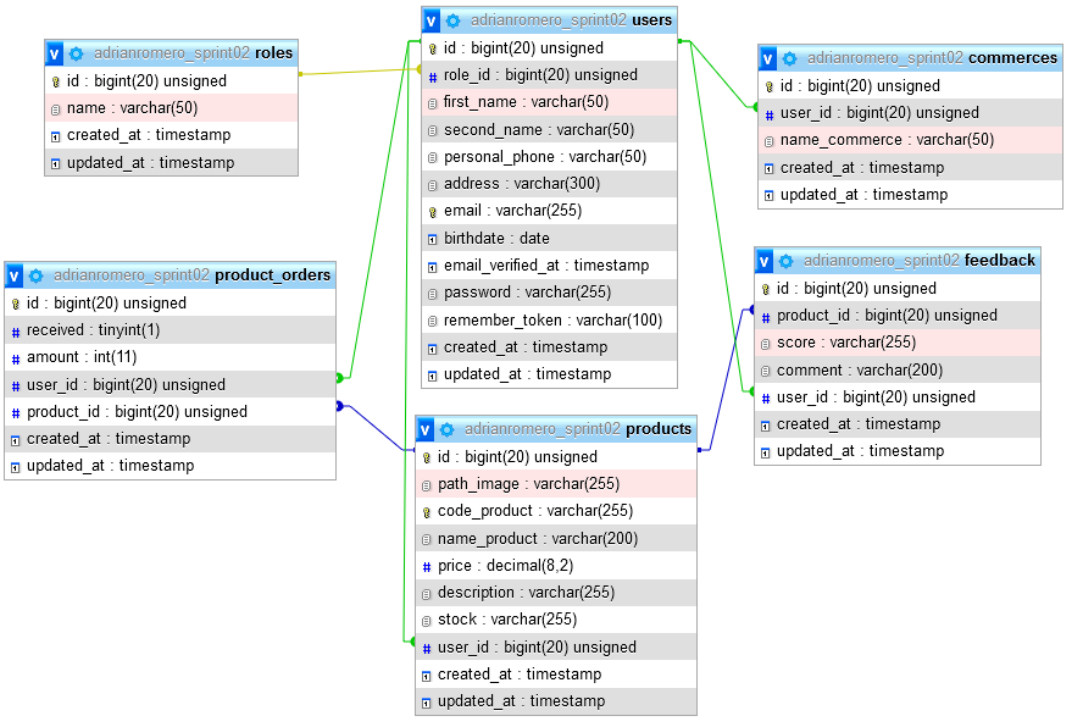


Fig. 6: Diagrama MER

Creación del repositorio

Gradualmente se cargará el proyecto del desarrollo de *Ecommerce* a la plataforma de *GitHub* mediante la herramienta de *Git*. Esto permitirá progresivamente trabajar y llevar un control del avance que se vaya teniendo con los *Pull Request* (ramas secundarias combinadas a la principal, manteniendo integridad en el proyecto). Nótese en la **Fig. 7**, la creación de la misma con dos de los *Sprints* ya cargados.

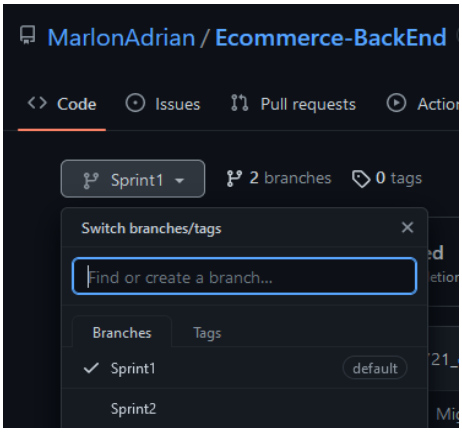


Fig. 7: Proyecto creado en el Repositorio de *GitHub*

Prueba Unitaria *Sprint* 1

Una vez finalizado las anteriores actividades propuestas, se muestra a continuación la **Fig. 8**, con las relaciones existentes en cada una de las entidades hacia la base de datos. Denotando su correspondencia correcta para futuras implementaciones.

Por ejemplo, si a un usuario registrado se lo elimina del sistema, las tablas que también se verán afectadas serán las de *feedback*, *commerces*, *product_orders* y *products*. Asimismo, la tabla de *feedback* lleva relaciones junto a la de *products* y *users*.

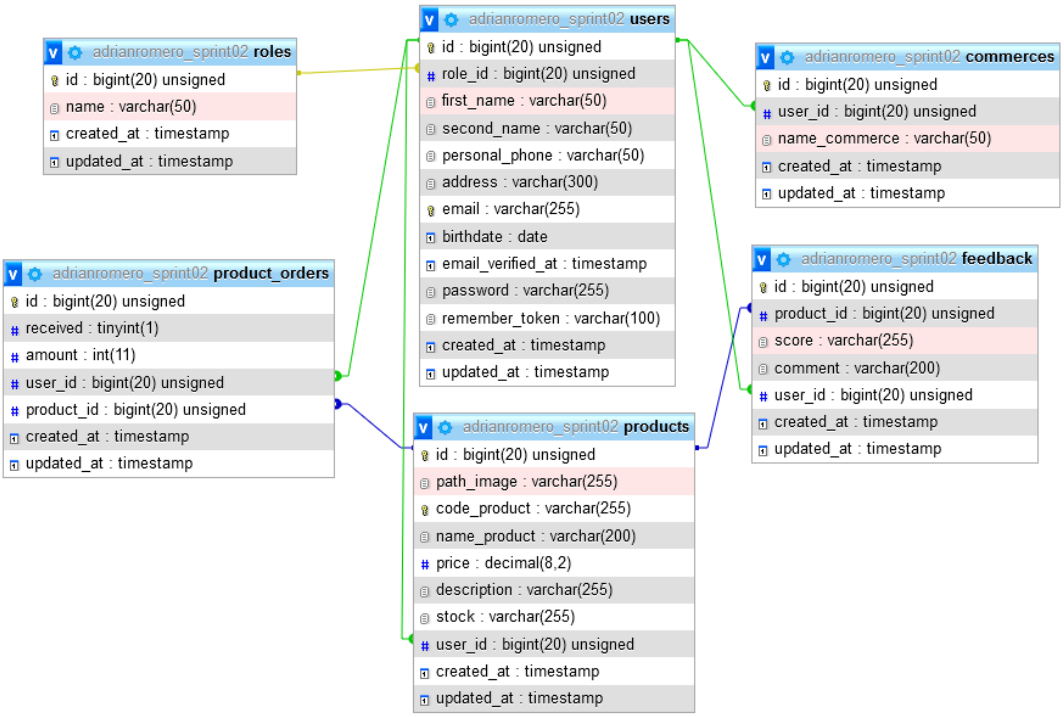


Fig. 8: Relaciones de las entidades

3.3 *Sprint* 2: Módulo de conexión con la base de datos

El administrador deberá contar con cada una de las tablas o modelos de la información registrada en el sistema *web*, por lo que para el desarrollo de este *Sprint* será necesario adaptar cada una de las tablas al proyecto de Laravel respectivamente.

Con ello mediante el lenguaje de programación *PHP* y *ORM* (Mapeo Objeto Relacional) podrán hacer uso de los datos alojados en las tablas. Asimismo, Laravel también posee su ORM, *Eloquent*, que lleva a cabo el patrón de arquitectura *Active Record*. Siendo así que ciertas clases de *PHP* conocidas como Modelos, estarán interactuando con cada una de las tablas alojadas en PhpMyAdmin y si existe alguna modificación o la ejecución de un *CRUD* en los atributos de alguna clase, entonces también se procederá a verse reflejado en la base de datos.

A causa de lo que antes se describió, es necesario que en el proyecto de *Laravel* se establezcan las siguientes acciones:

- Estructuración de cada una de las migraciones con sus respectivas relaciones de base de datos.
- Elaboración de Modelos y configuración de relaciones Eloquent.
- Creación de *Seeders* y *Factories*: con el fin de producir valores fortuitos.
- Prueba Unitaria *Sprint* 2.

Estructuración de cada una de las migraciones con sus respectivas relaciones de base de datos

Cada vez que se requiera la interacción del *framework* de *Laravel* con la base de datos se deberá primeramente realizar la configuración de cada una de las migraciones respectivas, por ejemplo, en la **Fig. *9***, correspondiente a la de productos, se desarrolla la creación de una de las tablas a poseer junto a las propiedades que contendrá la misma.

```php
Schema::create('products', function (Blueprint $table) {
    //ID for BDD
    $table->id();
    //Image for BDD
    $table->string('path_image')->nullable();
    //Product's Code
    $table->string('code_product')->unique();
    //Product's name
    $table->string('name_product',200);
    //Product's Price
    $table->decimal('price', 8, 2);
    //Product's Description
    $table->string('description')->nullable();
    //Product's Stock
    $table->string('stock');
    //Product's State
    $table->boolean('state')->default(true);
```

Fig. 9: Propiedades de los campos en archivo *PHP* de migración

De la misma manera se lo hará con cada una de las entidades identificadas para su respectiva migración hacia la base de datos.

En la **Fig. *10***, se muestra todas las migraciones existentes a efectuarse; usuarios, productos, negocios, roles, etc. En donde, la tabla *usuarios* posee por defecto acciones de autenticación junto a los cambios de contraseña.

Las 3 clases caracterizadas por ***add_to***, realizan la agregación de una columna hacia la tabla que se desee. Por ejemplo, en la última clase denominada *2022_12_13_052547_add_product_id_columnn_to_feedback*, realiza la adición de la columna *product_id* hacia la tabla *feedback*. Cada vez que se haga uso de esta denominación se debe hacerlo al final de la lista de migraciones.

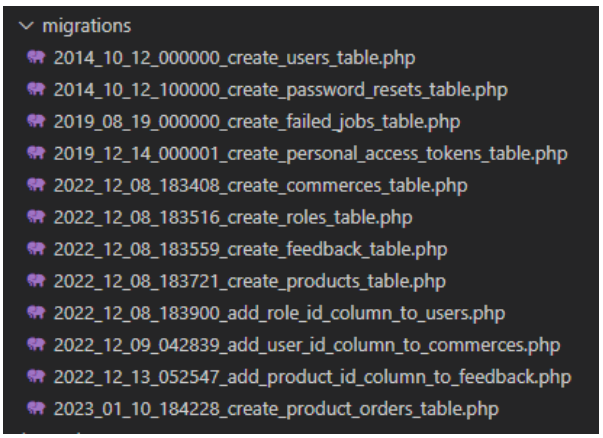


Fig. 10: Lista de migraciones

Una vez terminado con los nombres de una entidad (columnas), es necesario realizar el completado a nivel de bases de datos, detallando el nombre del campo del que proviene para que exista la relación. Esto se lo ejercerá para cada una de las tablas que sean

necesarias la implementación de campos foráneos, nótese la **Fig. *11***, se denota las configuraciones que poseerá a lo largo del proyecto de *software*; eliminación y actualización en cascada.

```
//Un producto puede estar varios usuarios y
//Un usuario puede tener muchos productos
$table->unsignedBigInteger('user_id');
$table->foreign('user_id')
      ->references('id')
      ->on('users')
      ->onDelete('cascade')
      ->onUpdate('cascade');
 $table->timestamps();
```

Fig. 11: Relación de producto a nivel de base de datos.

Elaboración de Modelos y configuración de relaciones Eloquent

A continuación, se muestra en la figura **Fig. 12**, la instrucción que sirve en la creación de algún modelo en *Laravel* seguido por la bandera *-mcrfs*, que especifica el establecimiento *CRUD* para el controlador de la clase Producto, junto a su *Seeder* y *Factory*.

```
PS C:\Users\MATR\Desktop\Ecommerce>
php artisan make:model Product -mcrfs
```

Fig. 12: Creación del modelo Producto

Es posible que en la mayoría de los proyectos ya exista un modelo con el nombre *User*, orientado a la autenticación y creado por defecto cuando se utiliza una plantilla con interfaz gráfica proveniente de *Laravel Breeze*. Sin embargo, en este proyecto orientado a *Backend* no se hará uso de dicha plantilla, sino del modelo *User* proporcionado.

En la **Fig. *13***, se muestra los atributos protegidos en un arreglo debido al término *$fillable,* que permitirá la asignación masiva, siempre y cuando los nombres de los campos existentes en la tabla de migraciones sean los mismos del arreglo. De lo contrario, no se podrá realizar métodos externos dirigidos a la base de datos, por ejemplo, la solicitud *HTTP* de método *POST*.

app > Models > Product.php

```php
<?php

namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Product extends Model
{
    use HasFactory;
    protected $fillable = [
        'user_id',
        'code_product',
        'name_product',
        'price',
        'description',
        'stock',
        'path_image'
    ];
```

Fig. 13: Modelo Producto

Por otra parte, las relaciones a nivel de *Eloquent* establecidas en la **Fig. *14***, muestran los nombres que provienen de otros modelos haciendo su relación con el modelo actual, así como *feedback*, *image* o *commerce*. Cada uno de los modelos creados (*User*, *Commerce*, *Feedback*, *Image*, *Product*) deberán ser consistentes con los archivos de migraciones creados anteriormente, de lo contrario se presentarán errores de relaciones a nivel de *Eloquent*.

```php
class Product extends Model
{
    use HasFactory;
    protected $fillable = [
        'user_id',
        'code_product',
        'name_product',
        'price',
        'description',
        'stock',
        'path_image'
    ];

    // Relación de uno a muchos
    //Un producto puede estar en varios negocios
    public function commerce()
    {
        return $this->belongsTo(Commerce::class);
    }

    // Relación de uno a muchos
    //Un producto puede tener muchos feedback
    public function feedback()
    {
        return $this->hasMany(Feedback::class);
    }

    //Relación de uno a muchos
    public function user()
    {
        return $this->belongsTo(User::class);
    }
```

Fig. 14: Relaciones a nivel ORM Eloquent

Creación de *Seeders* y *Factories*

La implementación de los *Seeders* y *Factories* son muy empleados en un proyecto de *Software*, debido a que favorecen el trabajo de una aplicación, así como la realización de pruebas en base a estos.

Los *Seeders*, son de utilidad para completar en una base de datos las tablas deseadas con valores de prueba. Mientras que los *Factories*, están encargados de especificar con qué tipo de valor debe llenarse una tabla, asimismo con el número de registros que se generarán [15].

Como ya se había mencionado antes, los *Seeders* y *Factories* ayudan al proyecto de software con valores ficticios para realizar pruebas. Por tal motivo, cada uno de estos valores deberán ser establecidos adecuadamente para que no haya conflictos después de que se haya realizado las migraciones. En la **Fig. *15***, se indica como ejemplo la manera con la que se completarán las columnas de la tabla Productos mediante un *Factory*.

```php
<?php

namespace Database\Factories;

use App\Models\Product;
use Illuminate\Database\Eloquent\Factories\Factory;
use App\Models\Commerce;
use App\Models\User;

class ProductFactory extends Factory
{
    protected $model = Product::class;

    public function definition()
    {
        return [
            'code_product' => $this->faker->iban(),
            'name_product' => $this->faker->userAgent(),
            'price' => $this->faker->randomFloat(2, 50, 100),
            'description' => $this->faker->sentence(),
            'stock' => $this->faker->numberBetween($min = 4, $max = 8),
            'path_image'=>'https://picsum.photos/id/'.$this
                    ->faker->unique()-> numberBetween($min = 1, $max = 6).'/200/300',
        ];
    }
}
```

Fig. 15: Factory de Producto

Asimismo, un *Seeder* debe ir de la mano con un *Factory*. Como ya se declaró los parámetros de las columnas, entonces se debe especificar cuántas repeticiones tendrá, tal y como se muestra en la **Fig. *16***, con 2 registros por cada negocio (*Commerce*).

```php
class ProductSeeder extends Seeder
{
    public function run()
    {
        $commerces_products = User::whereHas(
            'role', function($q){
                $q->where('name', 'owner');
            }
        )->get();

        $commerces_products->each(function($product)
        {
            Product::factory()->count(2)->for($product)->create();
        });
    }
}
```

Fig. 16: Seeder Producto

Cabe mencionar que existe un archivo llamado *DatabaseSeeder*, en el que se deberá poner el orden en el que se ejecutarán las clases tipo *Seeder*. De igual forma, la credencial de contraseña de inicio de sesión para cada uno de los usuarios será "*password*", debido a que gracias a los *Seeders* y *Factories* permiten agilizar el trabajo con la creación de la misma para todos los usuarios.

Por último, cuando ya se ejecuten cada una de estas clases junto al archivo de migración se obtendrán los registros de prueba que se revelan en la **Fig. *17***.

id	path_image	code_product	name_product	price	description	stock	state	user_id	created_at	updated_at
1	https://picsum.photos/id/5/200/300	MD78URND80304BCUE21MBCE6	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_0 rv:6...	91.23	Voluptates a accusamus quasi voluptate.	4	1	2	2023-01-24 18:00:25	2023-01-24 18:00:25
2	https://picsum.photos/id/3/200/300	LU44699FDA8N39I711AI	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 5.0;...	99.75	Perferendis nihil non atque autem.	6	1	2	2023-01-24 18:00:25	2023-01-24 18:00:25
3	https://picsum.photos/id/4/200/300	MK63830R2DT8OQA9P16	Opera/8.70 (Windows 98; nl-NL) Presto/2.12.308 Ver...	86.46	Quasi consequuntur tempora blanditiis enim accusam...	4	1	3	2023-01-24 18:00:26	2023-01-24 18:00:26
4	https://picsum.photos/id/2/200/300	MU54OELG247229608872231030 2IIC	Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_5_3) ...	67.30	Ullam asperiores dolorem inventore perferendis.	5	1	3	2023-01-24 18:00:26	2023-01-24 18:00:26
5	https://picsum.photos/id/6/200/300	MD931ZSAN9IP1MSR6HJBML04	Mozilla/5.0 (Windows CE) AppleWebKit/533.2 (KHTML,...	55.92	Est molestiae quo quos illo perferendis.	5	1	4	2023-01-24 18:00:26	2023-01-24 18:00:26
6	https://picsum.photos/id/1/200/300	CR02286846377988163428	Mozilla/5.0 (compatible; MSIE 11.0; Windows NT 6.0...	51.41	Autem id magnam consequatur vel ea nesciunt est se...	5	1	4	2023-01-24 18:00:27	2023-01-24 18:00:27

Fig. 17: Tabla Productos en la Base de datos

Prueba Unitaria *Sprint* 2

Dado que ya realizaron las respectivas migraciones y las creaciones de valores de prueba, se muestra en la **Fig. *18***, su creación exitosa de las mismas hacia la base de datos sin indicar error alguno en su ejecución. Cabe mencionar que para su ejecución se debe agregar el siguiente comando *php artisan migrate:refresh --seed*

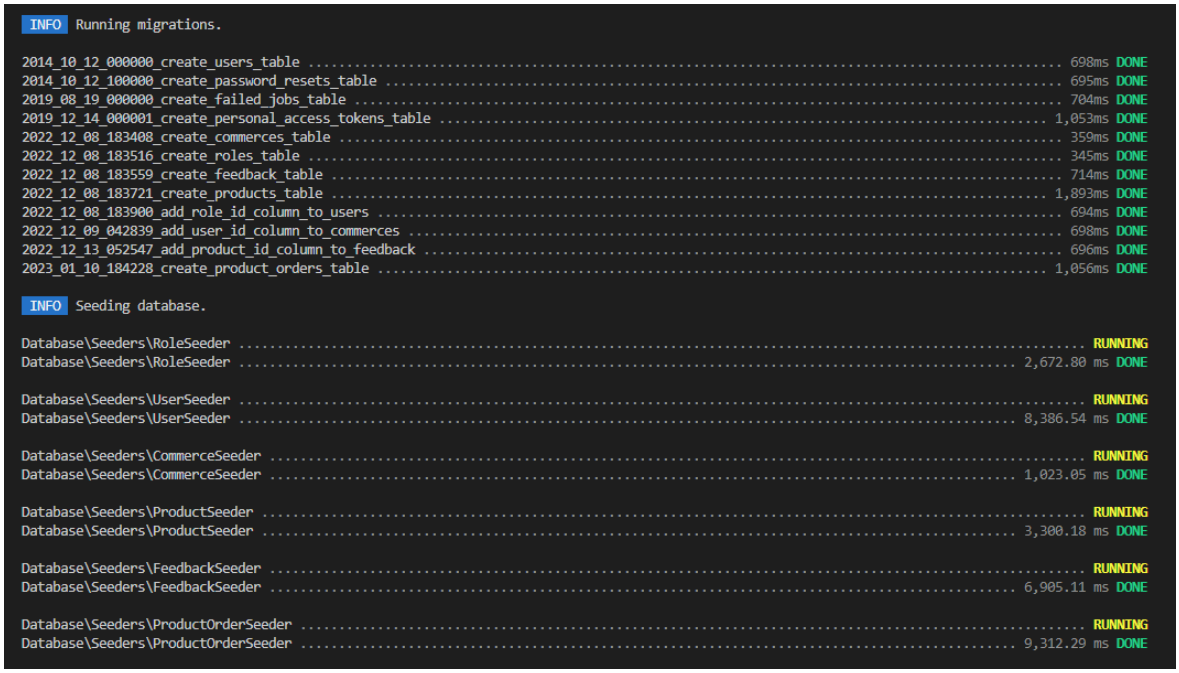


Fig. 18: Ejecución correcta de migraciones con valores de pruebas

3.4 *Sprint* 3: Desarrollo los módulos de autenticación y registro usuarios

En este punto es conveniente el ingreso de los usuarios al sistema de *Ecommerce*, por ello se establecen las siguientes tareas:

- Inicio de sesión acorde al perfil de cada usuario.
- Registro de usuarios y verificación de *email.*
- Actualización información de perfil.
- Recuperación de contraseña.
- Prueba unitaria *Sprint* 3.

Inicio de sesión acorde al perfil de cada usuario

En el desarrollo de este módulo de autenticación de *API* se hace uso del *token* JWT (*Json Web Token*), en el que se caracteriza por ser un sistema abierto orientado a la creación de *Tokens* momentáneos permitiendo en las aplicaciones el envío de información en formato *Json*.

La encriptación tiene cabida en cada solicitud a la *API* que se haga, lo cual genera mayor seguridad y utilidad hacia la arquitectura servidor-cliente. Así, al momento en el que los usuarios requieran acceder a sus recursos tendrán que realizar una solicitud a la *API*.

El trabajo que se realiza detrás de esta solicitud es que los datos de un usuario registrado deberán coincidir en la base de datos y ser válidos para que se cree un JWT para corroborar que la autenticación es exitosa. Tomando en cuenta que el token que se genera tiene un tiempo limitado para su uso y será suficiente para realizar cualquier petición que el usuario requiera. En la **Fig. 19**, se presenta las rutas de tipo *API* que estarán usando *JwtMiddleware.*

routes > api.php

```
12  use App\Http\Middleware\JwtMiddleware;
13
14  Route::post('/login', [AuthController::class, 'login']);
15  Route::post('/register', [RegisteredUserController::class, 'register']);
16  Route::get('/products', [ProductOwnerController::class, 'products']);
17  Route::get('/feedbacks', [FeedbackController::class, 'feedbacks']);
18
19
20  Route::get('/indexUsers', [AdminController::class, 'indexUsers']);
21  Route::get('/indexUsers/{id}', [AdminController::class, 'showUsers']);
```

Fig. 19: Rutas con JWT

Además, como ejemplo de la implementación de su uso se muestra la **Fig. 20**, el controlador *RegisteredUserController* de tipo *Api* que llama a *JWTAuth*, para generar la autenticación a un usuario registrado.

app > Http > Controllers > Api > RegisteredUserController.php

```
50
51      public function register(Request $request)
52      {
53
61          $user = User::create([
62              'first_name' => $request->first_name,
64              'second_name' => $request->second_name,
65              'email' => $request->email,
66              'password' => Hash::make($request->password),
67              'personal_phone' => $request->personal_phone,
68              'address' => $request->address,
69              'birthdate' => $request->birthdate,
70              'role_id' =>3
71          ]);
72
73          $token = JWTAuth::fromUser($user);
74
```

Fig. 20: Generación de token mediante *JWTAuth*

Cada proceso de autenticación será igual para cualquier rol de usuario, con la excepción de los recursos y alcances que tendrán cada uno de los roles.

Registro de usuarios y verificación de *email*

Cuando un nuevo usuario haga el registro en este *Sprint* requiere de un formulario, cuando este sea completado correctamente será enviado hacia el servidor para que apruebe las

credenciales ingresadas. De modo que, cuando se apruebe la solicitud por el servidor se enviará un correo electrónico de bienvenida al sistema.

Se muestra en la **Fig. 21**, la función *register* que se encarga de validar los datos ingresados, de la misma forma se creará un nuevo usuario en base al modelo *User*, lo que conduce a la creación de un *token* hacia dicho usuario. Tomar en cuenta que cada vez que se registre un nuevo usuario este tendrá por defecto el rol cliente (*role*_id=3).

Controllers > Api > RegisteredUserController.php

```php
public function register(Request $request)
{
    $validator = Validator::make($request->all(),[
        'first_name' => ['required', 'string', 'max:50'],
        'second_name' => ['required', 'string', 'max:50'],
        'email' => ['required', 'string', 'email', 'max:255', 'unique:'.User::class],
        'personal_phone' => ['required', 'string', 'max:200','unique:'.User::class],
        'password' => ['required', 'confirmed', Rules\Password::defaults()],
        'address' => ['required', 'string', 'max:200'],
        'birthdate' => ['required', 'string', 'max:10'],
    ]);
    if($validator->fails()){
        return response()->json($validator->errors()->toJson(),400);
    }

    $user = User::create([
        'first_name' => $request->first_name,
        'second_name' => $request->second_name,
        'email' => $request->email,
        'password' => Hash::make($request->password),
        'personal_phone' => $request->personal_phone,
        'address' => $request->address,
        'birthdate' => $request->birthdate,
        'role_id' =>3
    ]);

    $token = JWTAuth::fromUser($user);
    event(new Registered($user));

    return response()->json(compact('user','token'),201);
}
```

Fig. 21: Función de registro un nuevo usuario “Cliente”

Nótese en la figura **Fig. 22**, que mediante la herramienta de *Postman* se crea un nuevo usuario en la ruta de tipo *api* llamada *register*, recuerde que en la **Fig. 19**, se declaró previamente dicha ruta. Entonces, cuando se registra un nuevo usuario, llevará consigo la aparición de la misma en la base de datos, **Fig. 23**, así como el mensaje hacia el correo electrónico, **Fig. 24**.

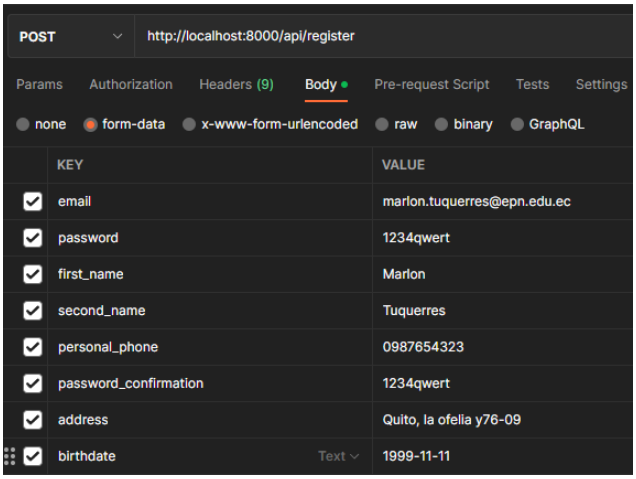


Fig. 22: Registro de un nuevo usuario en *Postman*

id	role_id	first_name	second_name	personal_phone	address	state	email	birthdate
30	3	Marlon	Tuquerres	0987654323	Quito, la ofelia y76-09	1	marlon.tuquerres@epn.edu.ec	1999-11-11

Fig. 23: Usuario registrado en la base de datos

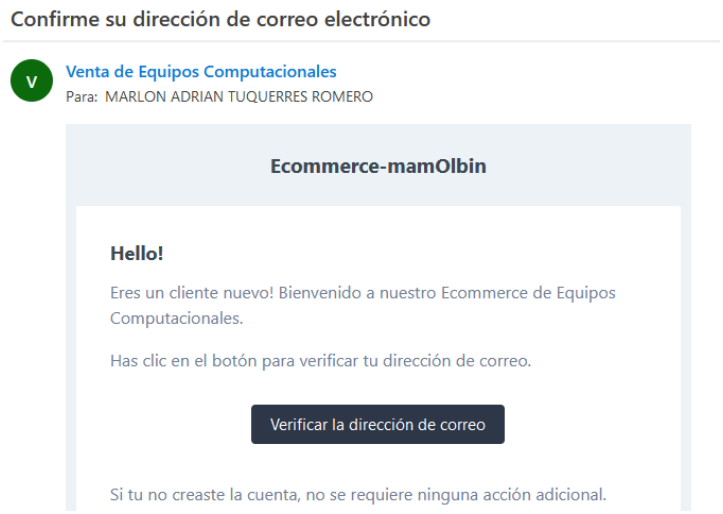


Fig. 24: Verificación del correo electrónico registrado por usuario

Actualización información de perfil

En cada sistema *web* existente es primordial que cada usuario tenga la posibilidad de actualizar su contraseña y/o información personal como lo es el número de celular o *email*, por si existe un caso de cambio de las mismas. Por tal motivo, en el desarrollo de este *Ecommerce* se implementará las tres actualizaciones mencionadas anteriormente; actualización de contraseña, número de celular y *email*.

En las figuras **Fig. *25*** y **Fig. 26**, se indican las configuraciones de cada una de las funciones que contendrán la respectiva estructura para que exista la actualización de la información del perfil.

app > Http > Controllers > Profile > ProfileInformationController.php

```php
public function update(Request $request)
{
    $request->validate([
        'personal_phone' => [ 'numeric', 'digits:10','unique:'.User::class],
        'address' => [ 'nullable','string', 'min:5', 'max:300'],
        'email' => [ 'nullable','string', 'email', 'max:255','unique:'.User::class]
    ]);

    $user = $request->user();
    $user->address = $request['address'];

    if($request->personal_phone!=null && $request->email!=null ){
        $user->personal_phone = $request['personal_phone'];
        $user->email = $request['email'];
    }
    else if($request->email!=null ){
        $user->personal_phone = Auth::user()->personal_phone;
        $user->email = $request['email'];
    }
    else if($request->personal_phone!=null ){
        $user->email = Auth::user()->email;
        $user->personal_phone = $request['personal_phone'];
    }

    $user->save();
    return with(
        ['msg' => 'Profile_information_updated']);
}
```

Fig. 25: Función para la actualización de *email* y número celular

app > Http > Controllers > Profile > PasswordController.php

```php
public function update(Request $request)
{
    $request->validate([
        'current_password' => ['required', 'string', 'max:255'],
        'password' => [
            'required','string','confirmed','max:255',
            Password::min(6)->mixedCase()->numbers()->symbols(),
        ],
    ]);

    $user = $request->user();

    //Se debe verificar si la contraseña coincide con la del usuario
    if(!$this->checkPassword($request->input('current_password'), $user->password)){
        throw ValidationException::withMessages([
            'current_password' => 'Is not your current password. Remember that you already updated it'
        ]);
    }

    $user->password = Hash::make($request->password);
    $user->save();
    return with(
        ['msg' => 'password_updated']);
}
```

Fig. 26: Función para la actualización de contraseña

Recuperación de contraseña

Es evidente que a cualquier persona se le olvide la contraseña de su cuenta y tenga que recuperarla mediante el envío de un correo electrónico para la recuperación de la misma. Por ende, a continuación, se desarrolla la implementación de esta labor.

Con el fin de llevar a cabo la recuperación de contraseña se hará uso de una plantilla proporcionada por *Laravel*, que fue implementada mediante *laravel/breeze*, con las siguientes dos instrucciones, **Fig. 27**.

Este fin requiere previamente la generación de credenciales de una aplicación de correo electrónico para que estas sean agregadas al archivo .env del proyecto.

```
composer require laravel/breeze --dev
php artisan breeze:install
```

Fig. 27: Comando para el uso de *laravel/breeze*

En la siguiente figura, **Fig. 28**, se muestra la configuración de la función del controlador *PasswordResetLinkController* para el restablecimiento de la contraseña.

app > Http > Controllers > Auth > PasswordResetLinkController.php

```php
public function store(Request $request)
{
    $request->validate([
        'email' => ['required', 'email'],
    ]);

    // We will send the password reset link to this user. Once we have attempted
    // to send the link, we will examine the response then see the message we
    // need to show to the user. Finally, we'll send out a proper response.
    $status = Password::sendResetLink(
        $request->only('email')
    );
    $status == Password::RESET_LINK_SENT
                ? back()->with('status', __($status))
                : back()->withInput($request->only('email'))
                        ->withErrors(['email' => __($status)]);
    return with([
        'msg'=>'Please check you email for reset your password']);
}
```

Fig. 28: Función para la recuperación de contraseña

Prueba unitaria *Sprint* 3

La mejor manera de verificar que un usuario esté dentro de un sistema es comprobando su aparición en sistema del administrador, tal y como se indica en la **Fig. 23**. De igual forma con el uso de la herramienta de *Postman* su inicio de sesión con respuesta de un *token* generado, **Fig. 29**, y esto es gracias a su registro que se lo realizó como indica la **Fig. 22**.

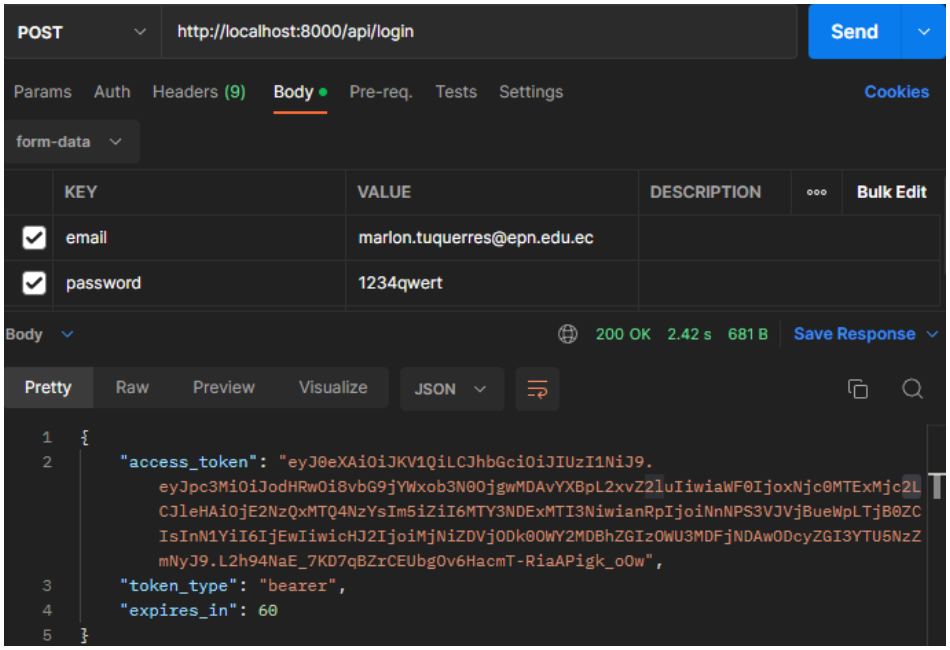


Fig. 29: Inicio de sesión de un nuevo usuario registrado

Con respecto a la recuperación de contraseña se efectuó su correcto restablecimiento, como lo indica en las siguientes figuras. Para el generar una nueva contraseña se puede ingresar mediante el sitio *web* o la herramienta de *Postman*, **Fig. *30*** y **Fig. *31***, cualquiera que fuese el método de ingreso generarán un enlace como se muestra en la **Fig. *32***.

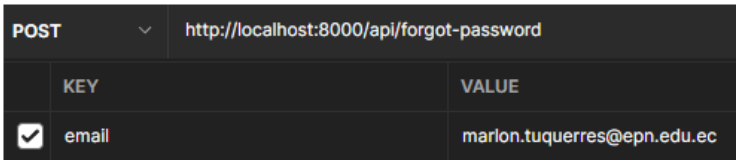


Fig. 30: Recuperación de contraseña mediante *Postman*

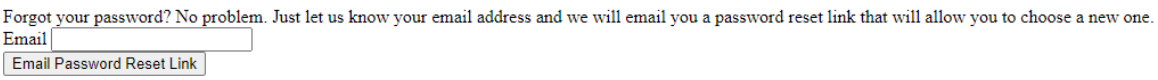


Fig. 31: Recuperación de contraseña mediante enlace

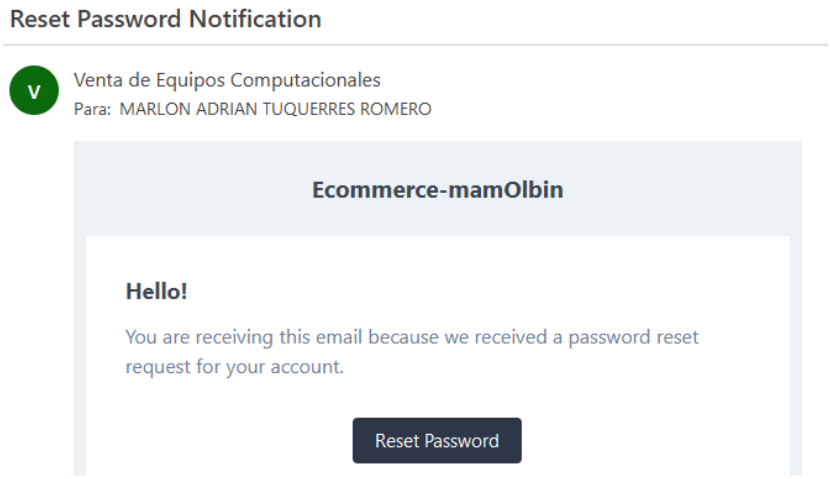


Fig. 32: Envío de *email* para restablecer contraseña

Seguido a ello se crea una nueva contraseña, **Fig. *33***, mostrando su correcto restablecimiento **Fig. *34***.

Fig. 33: Creación nueva contraseña

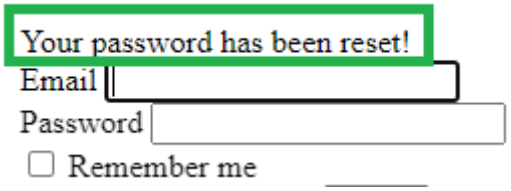


Fig. 34: Actualización de contraseña realizada

3.5 *Sprint* 4: Desarrollo de módulos *Endpoints* de Usuarios

En este punto se procede con la realización de tareas y actividades que corresponden a cada uno de los usuarios del sistema de *Ecommerce*. Sin embargo, como este proyecto está orientado al *Backend* se hará uso de controladores específicos de *API*, estos estarán creados mediante comandos proporcionados por *Laravel*. Esto se realizará con el fin de establecer los alcances y acciones que tendrán de cada uno de los roles de usuarios. Es por ello por lo que se plantean las acciones de a continuación:

- Creación de controladores *API.*
- Modificación de los métodos de los controladores *API.*

- Establecimiento de rutas *API.*
- Prueba unitaria *Sprint* 4.

Creación de controladores *API*

En base a los estándares de comandos que indica *Laravel*, se debe crear un controlador de tipo *API* con dicho término, tal y como se muestra a continuación, **Fig. 35**, con el objetivo de agrupar la lógica de peticiones *HTTP* y organizar de mejor manera a nuestro código [16].

```
PS C:\Users\MATR\php artisan make:controller Api/ProductOwnerController
```

Fig. 35: Creación de controlador de tipo *API*

De la misma manera se lo debe hacer con cada uno de los controladores a utilizarse a lo largo del proyecto, en la **Fig. 36**, se indica cada uno de ellos.

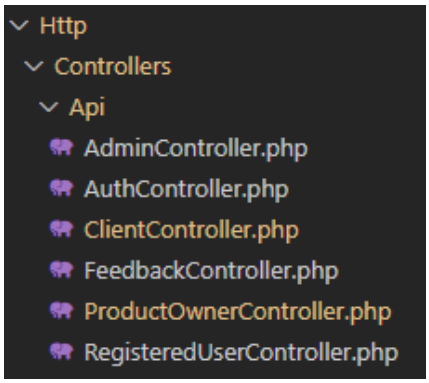


Fig. 36: Controladores *API*

Modificación de los métodos de los controladores *API*

Debido a que el proyecto se basa en un componente *Backend* no se manejan vistas. Por tal motivo, se utilizará respuestas en formato *JSON*, este formato se lo puede definir como un arreglo que contiene varios objetos caracterizados por clave y valor, y en esta ocasión serán utilizados como intercambio de información entre el servidor y cliente.

En la modificación de los métodos se tendrán que realizar un *CRUD* para cada uno de los usuarios. Por ejemplo, el propietario de un negocio tendrá la posibilidad de publicar un producto, editarlo, y/o eliminarlo. De la misma manera un usuario cliente con sus opiniones publicadas de los productos o en la adquisición de alguno. A continuación, se muestran los métodos a implementarse en el controlador *ProductOwnerController* (Propietario del Negocio), **Fig. *37***. Asimismo, del contenido de uno de ellos, ***Fig. 38***.

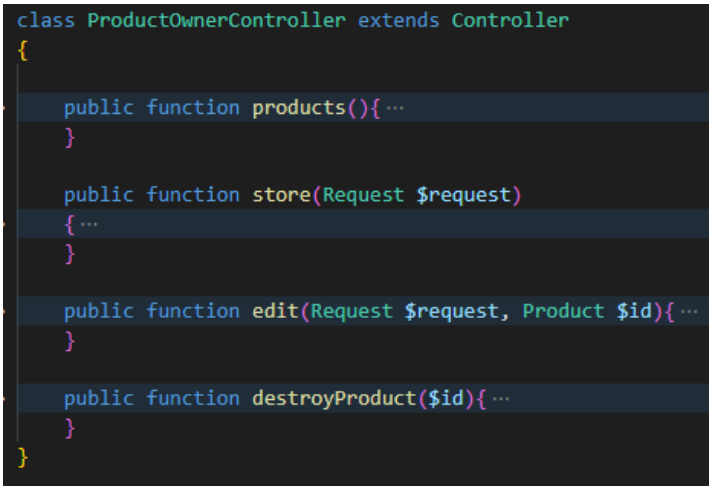


Fig. 37: Métodos *CRUD* del *ProductOwnerController*

```php
public function store(Request $request)
{
    $request->validate([
        'code_product' => ['required', 'string', 'min:3', 'max:35','unique:products'],
        'name_product' => ['required', 'string', 'min:3', 'max:35'],
        'price' => ['required', 'numeric'],
        'description' => ['required', 'string'],
        'stock' => ['required', 'numeric'],
        'path_image' => ['required', 'string'],
    ]);

    $owner = Auth::user();
    $product = Product::create([
        'code_product' => $request['code_product'],
        'name_product' => $request['name_product'],
        'price' => $request['price'],
        'description' => $request['description'],
        'stock' => $request['stock'],
        'path_image' =>$request['path_image'],
        'user_id'=>1
    ]);
    $owner->products()->save($product);
    return with(
        ['msg' => 'product_created']);
}
```

Fig. 38: Método para la creación de un producto

Establecimiento de rutas *API*

Las rutas de este sistema están sujetas a entregar datos en formato *JSON* para el consumo de *API* mediante solicitudes *HTTP*. En la siguiente figura, **Fig. 39**, se muestra la declaración de las mismas. Por ejemplo, en la sección de productos se está especificando el método con el que se está trabajando; *GET*, *POST*, *PUT*, *DELETE*. Y dependiendo de la acción a realizarse en el *ProductOwnerController* tendrá la posibilidad de hacer alguna de esas tareas. Además, *publishProduct*, *showProducts*, *destroyProduct*, *editProduct*, son los mismos nombres de los métodos que existen en el controlador *ProductOwnerController*.

Por otra parte, los términos mostrados en la **Fig. 40**, son las denominaciones que se deben colocar en la herramienta de *Postman*.

```php
/*----------------------ADMIN----------------------*/
/*Users*/
Route::get('/indexUsers', [AdminController::class, 'indexUsers']);
Route::get('/showUser/{id}', [AdminController::class, 'showUser']);
Route::delete('/destroyUser/{id}', [AdminController::class, 'destroyUser']);
/*Commerces*/
Route::get('/indexCommerces', [AdminController::class, 'indexCommerces']);
Route::get('/showCommerce/{id}', [AdminController::class, 'showCommerce']);
Route::delete('/destroyCommerce/{id}', [AdminController::class, 'destroyCommerce']);
/*Product */
Route::get('/indexProducts', [AdminController::class, 'indexProducts']);
Route::get('/showaProduct/{id}', [AdminController::class, 'showaProduct']);
/*Feedback */
Route::get('/showFeedback/{id}', [AdminController::class, 'showFeedback']);

/*------------------PRODUCT OWNER------------------*/
Route::get('/products', [ProductOwnerController::class, 'products']);
Route::post('/publishProduct', [ProductOwnerController::class, 'store']);
Route::get('/showProduct/{id}', [ProductOwnerController::class, 'showProduct']);
Route::delete('/destroyProduct/{id}', [ProductOwnerController::class, 'destroyProduct']);
Route::put('/editProduct/{id}', [ProductOwnerController::class, 'edit']);

/*----------------------CLIENT----------------------*/
Route::post('/postFeedback', [ClientController::class, 'postfeedback']);
Route::put('/editFeedback/{id}', [ClientController::class, 'editfeedback']);
Route::delete('/destroyFeedback/{id}', [ClientController::class, 'destroyfeedback']);
Route::post('/orderProduct', [ClientController::class, 'orderProduct']);
Route::put('/editOrderProduct/{id}', [ClientController::class, 'editOrderProduct']);
Route::get('/showfeedbacks', [ClientController::class, 'showfeedbacks']);
Route::get('/showMyProduct/{id}', [ClientController::class, 'showMyProduct']);
```

Fig. 39: Rutas *API*

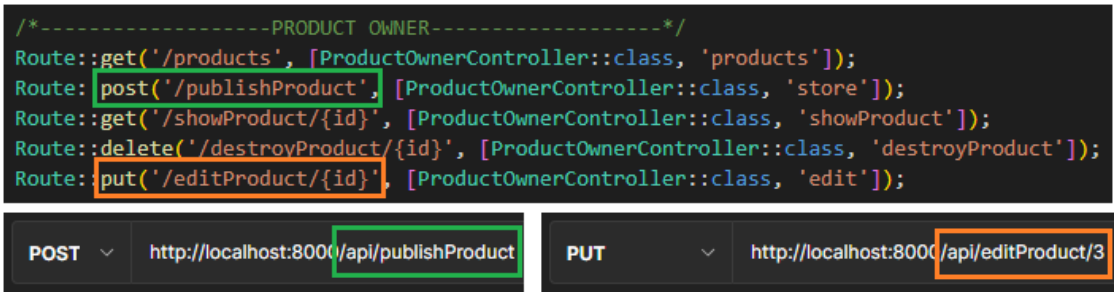


Fig. 40: Denominaciones *CRUD* para *API*

Prueba unitaria *Sprint* 4

A modo de ejemplo el usuario propietario ya puede publicar un nuevo producto a la venta, asimismo con su edición y/o eliminación.

En la **Fig. *41***, hay la existencia de 4 productos creados por distintos usuarios. Mientras que, en la **Fig. *42***, se está creando un nuevo producto a la venta, y se corrobora la creación de la misma en la ***Fig. 43***.

id	path_image	code_product	name_product	price	description	stock	state	user_id
1	https://picsum.photos/id/5/200/300	HU375017999123897305065459 53	Opera/9.99 (Windows NT 5.1; nl-NL) Presto/2.10.180...	83.06	Autem id culpa fugiat minus.	9	1	2
2	https://picsum.photos/id/6/200/300	HU206787168493990321173767 13	Mozilla/5.0 (iPad; CPU OS 8_1_1 like Mac OS X; en-...	57.05	Aut unde molestias laudantium.	4	1	2
3	https://picsum.photos/id/4/200/300	HU776721850892534035100792 32	Opera/8.35 (Windows NT 6.0; sl-SI) Presto/2.9.324 ...	69.38	Aut modi saepe incidunt.	5	1	3
4	https://picsum.photos/id/1/200/300	VG78VXFH5693417844490136	Mozilla/5.0 (Macintosh; PPC Mac OS X 10_7_0) Apple...	51.29	Praesentium numquam odio animi consequatur nisi.	2	1	3

ected: Edit Copy Delete Export

Fig. 41: Contenido tabla productos de la base de datos

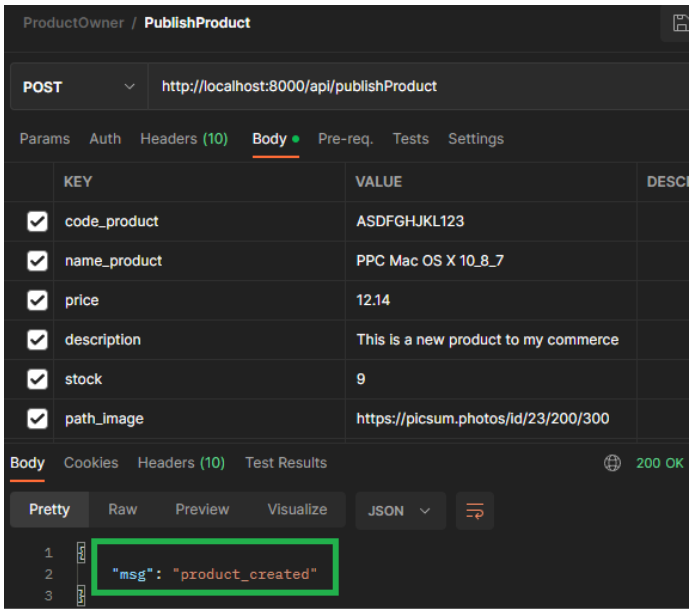


Fig. 42: Creación de un nuevo producto

id	path_image	code_product	name_product	price	description	stock	state	user_id
1	https://picsum.photos/id/5/200/300	HU375017999123897305065459 53	Opera/9.99 (Windows NT 5.1; nl-NL) Presto/2.10.180...	83.06	Autem id culpa fugiat minus.	9	1	2
2	https://picsum.photos/id/6/200/300	HU206787168493990321173767 13	Mozilla/5.0 (iPad; CPU OS 8_1_1 like Mac OS X; en-...	57.05	Aut unde molestias laudantium.	4	1	2
3	https://picsum.photos/id/4/200/300	HU776721850892534035100792 32	Opera/8.35 (Windows NT 6.0; sl-SI) Presto/2.9.324 ...	69.38	Aut modi saepe incidunt.	5	1	3
4	https://picsum.photos/id/1/200/300	VG78VXFH5693417844490136	Mozilla/5.0 (Macintosh; PPC Mac OS X 10_7_0) Apple...	51.29	Praesentium numquam odio animi consequatur nisi.	2	1	3
9	https://picsum.photos/id/23/200/300	ASDFGHJKL123	PPC Mac OS X 10_8_7	12.14	This is a new product to my	9	1	2

Fig. 43: Comprobación del producto creado

Por otra parte, si el usuario propietario desea eliminar alguno de productos se verá reflejado de la misma manera en la base de datos, **Fig. 44** y **Fig. 45**.

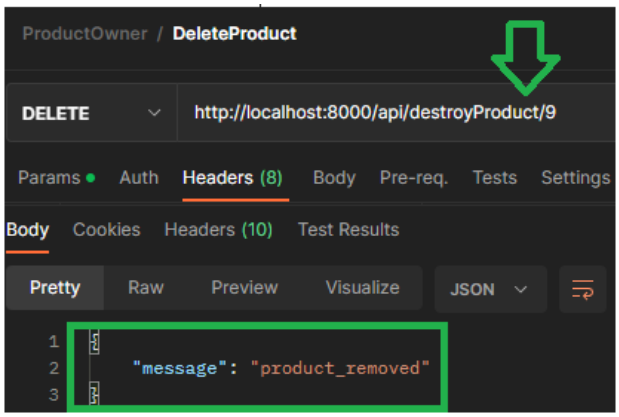


Fig. 44: Eliminación del producto creado

id	path_image	code_product	name_product	price	description	stock	state	user_id
1	https://picsum.photos/id/5/200/300	HU375017999123897305065459 53	Opera/9.99 (Windows NT 5.1; nl-NL) Presto/2.10.180...	83.06	Autem id culpa fugiat minus.	9	1	2
2	https://picsum.photos/id/6/200/300	HU206787168493990321173767 13	Mozilla/5.0 (iPad; CPU OS 8_1_1 like Mac OS X; en-...	57.05	Aut unde molestias laudantium.	4	1	2
3	https://picsum.photos/id/4/200/300	HU776721850892534035100792 32	Opera/8.35 (Windows NT 6.0; sl-SI) Presto/2.9.324 ...	69.38	Aut modi saepe incidunt.	5	1	3
4	https://picsum.photos/id/1/200/300	VG78VXFH5693417844490136	Mozilla/5.0 (Macintosh; PPC Mac OS X 10_7_0) Apple...	51.29	Praesentium numquam odio animi consequatur nisi.	2	1	3

ected: Edit Copy Delete Export

Fig. 45: Corroboración del producto eliminado

3.6 *Sprint* 5: Desarrollo de módulos *Gates*, integridad y control de acceso

En cada realización de las actividades y tareas convenientes a cada uno de los usuarios es necesario plantear las siguientes actividades que corresponderán en el sistema de *Ecommerce* que son:

- Elaboración de diagrama de casos de uso.
- Codificación de "*Gates*" para los distintos roles de usuario.
- Codificación de métodos para la integridad de datos con su respectiva autenticación y control de acceso.
- Prueba unitaria *Sprint* 5.

Elaboración de diagrama de casos de uso

La implementación de diagrama de casos de uso está dirigida con el objetivo de poseer una representación visual de las actividades y acciones que pueden tener los usuarios en un sistema, y esto servirá para la adaptación de la misma hacia el proyecto de *Laravel*. En las siguientes figuras **Fig. 46**, **Fig. *47*** y **Fig. 48**, se denotan cada una de las acciones existentes para el usuario cliente, administrador y propietario.

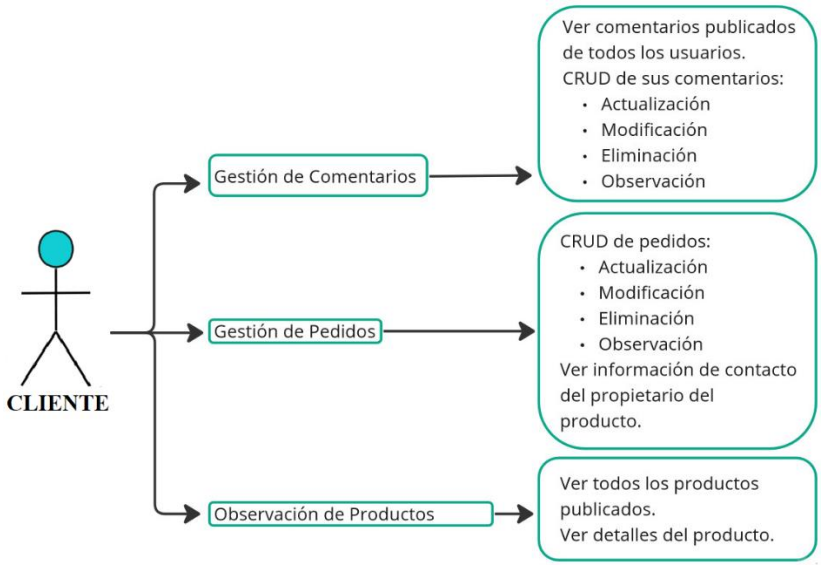


Fig. 46: Diagrama de Casos de uso de Rol "Cliente"

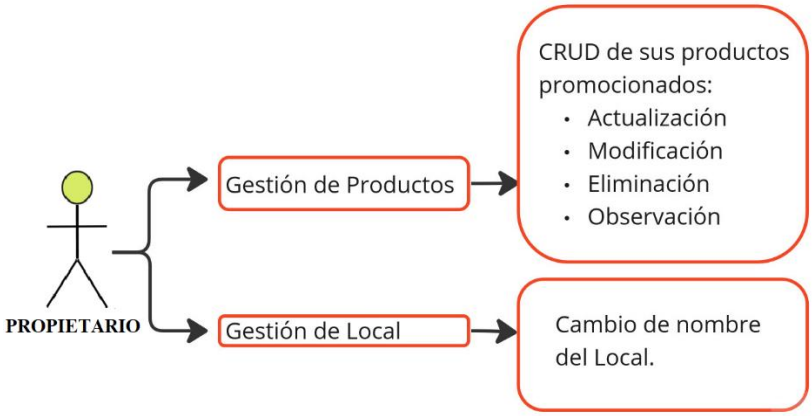


Fig. 47: Diagrama de Casos de uso de Rol "Propietario"

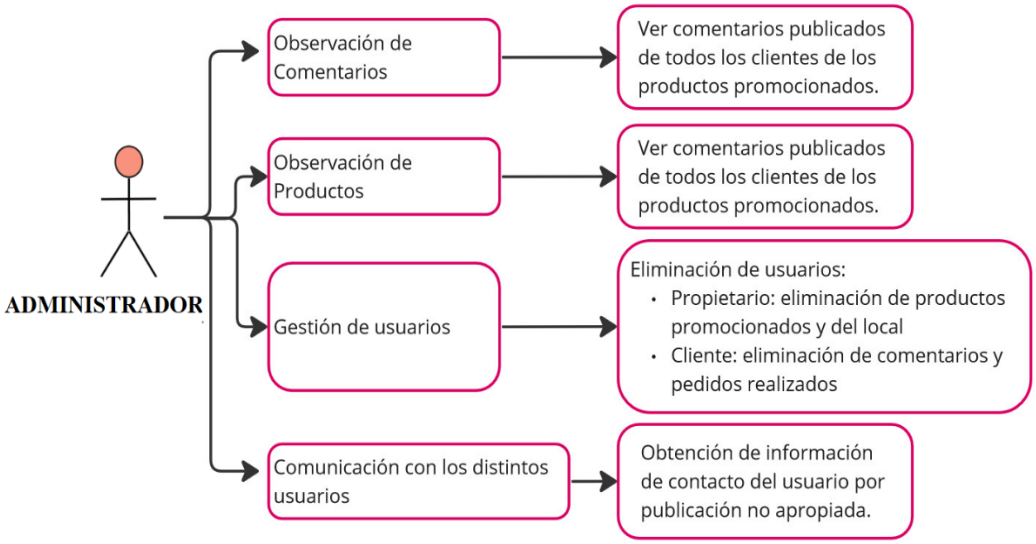


Fig. 48: Diagrama de Casos de uso de Rol "Administrador"

Codificación de "*Gates*" para los distintos roles de usuario

Dado que en el sistema de *Ecommerce* habrá varias actividades por parte de los usuarios, entonces es necesario la implementación de *Gates*, debido a que esto maneja la autorización óptima de los recursos en un sistema. Es decir, atribuir acciones de denegación o autorización ante los datos registrados, por ejemplo, cualquier usuario cliente no podrá ingresar a controlar los productos de un negocio, o que un usuario propietario pueda eliminar a un usuario del sistema.

En la **Fig. 49**, se muestra cada uno de los *Gates* que estarán trabajando en el proyecto de *Ecommerce*. Se indica que el usuario con *rol_id=1*, administrador, podrá manejar a todo el personal registrado en el sistema, asimismo con los negocios de los propietarios. Por otra parte, el usuario con *rol_id=2* podrá gestionar sus productos, mientras que el usuario con *rol_id=3* manejará sus pedidos.

```php
public function boot()
{
    $this->registerPolicies();

    Gate::define('manage-personal', function (User $user)
    {
        return $user->role_id == '1';
    });

    Gate::define('manage-commerces', function (User $user)
    {
        return $user->role_id == '1';
    });

    Gate::define('manage-products', function (User $user)
    {
        return $user->role_id == '2';
    });

    Gate::define('manage-orders', function (User $user)
    {
        return $user->role_id == '3';
    });
}
```

Fig. 49: Lista de *Gates*

Codificación de métodos para la integridad de datos con su respectiva autenticación y control de acceso

Si bien pueden estar los usuarios registrados en el sistema y posean sus roles respectivos, esto no los hace tener la autoridad de modificar la información de otros usuarios. Como modo de ejemplo se plantea el siguiente escenario, anteriormente se declaró que los usuarios propietarios deban modificar sus productos a conveniencia, sin embargo, no deben estar autorizados a modificar los productos de otro propietario que promocione otros productos. A esto se lo conoce como integridad de datos y recursos, y control de acceso.

En la **Fig. 50**, se está validando que el usuario propietario que acaba de ingresar al sistema deberá contener la misma identificación que le corresponde a su producto publicado si es que desea realizar alguna modificación de alguno de sus productos, de lo contrario no podrá hacerlo.

```php
public function edit(Request $request, Product $id){
    if(Auth::user()->id == $id->user_id){
    //input
    $request->validate([
        'code_product' => ['required', 'string', 'min:3', 'max:35','unique:products'],
        'name_product' => ['required', 'string', 'min:3', 'max:35'],
        'price' => ['required', 'numeric'],
        'description' => ['required', 'string'],
        'stock' => ['required', 'numeric'],
        'path_image' => ['required', 'string'],
    ]);

    $id->update(
        [
            'code_product' => $request['code_product'],
            'name_product' => $request['name_product'],
            'price' => $request['price'],
            'description' => $request['description'],
            'stock' => $request['stock'],
            'path_image' => $request['path_image'],
        ]);
        return Response::allow('Product updated');
    } else{
        return Response::deny('You do not own this product.');
    }
}
```

Fig. 50: Autorización para la modificación de productos

Prueba unitaria *Sprint* 5

El procedimiento para la corroboración de este *Sprint* resulta un poco extenso. En la **Fig. 51**, se muestra un *token* que fue agregado para la edición de un pedido de producto, sin embargo, no fue permitido la acción debido a que no es el propietario del mismo.

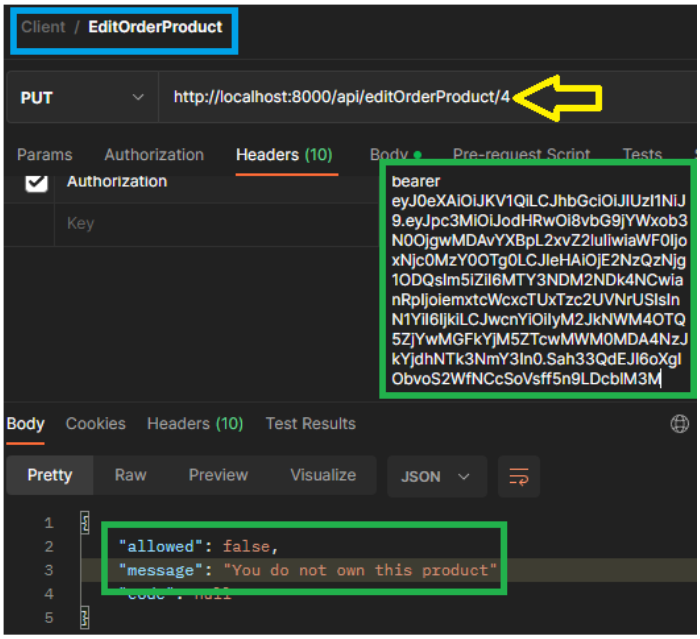


Fig. 51: Permiso denegado para editar un pedido

No obstante, se generará un *token* que pertenece al usuario que sí realizó dicho pedido, **Fig. 52**.

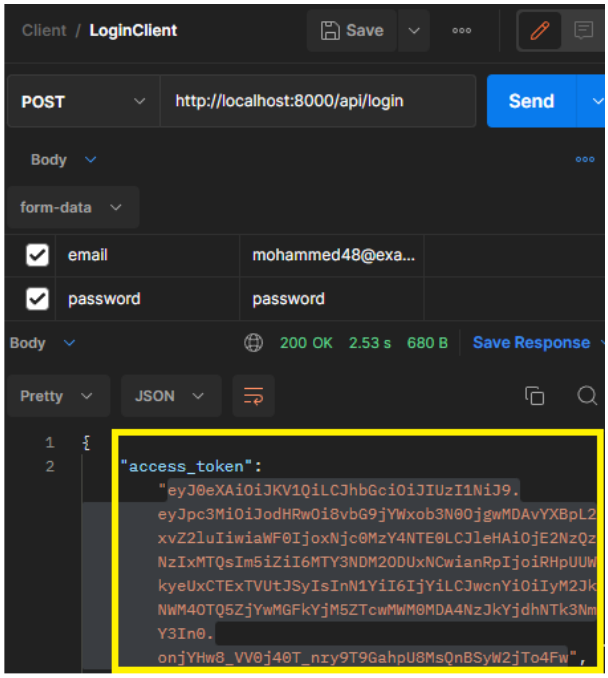


Fig. 52: Generación de *token* cliente

Posteriormente, se incluirá dicho *token* para la obtención de los productos que corresponden al usuario con *email*: *mohammed48@example.org,* **Fig. *53***. Obteniendo como resultado la realización de un producto con identificación igual a 4, y una cantidad de 1 producto, con su correspondiente nombre de producto.

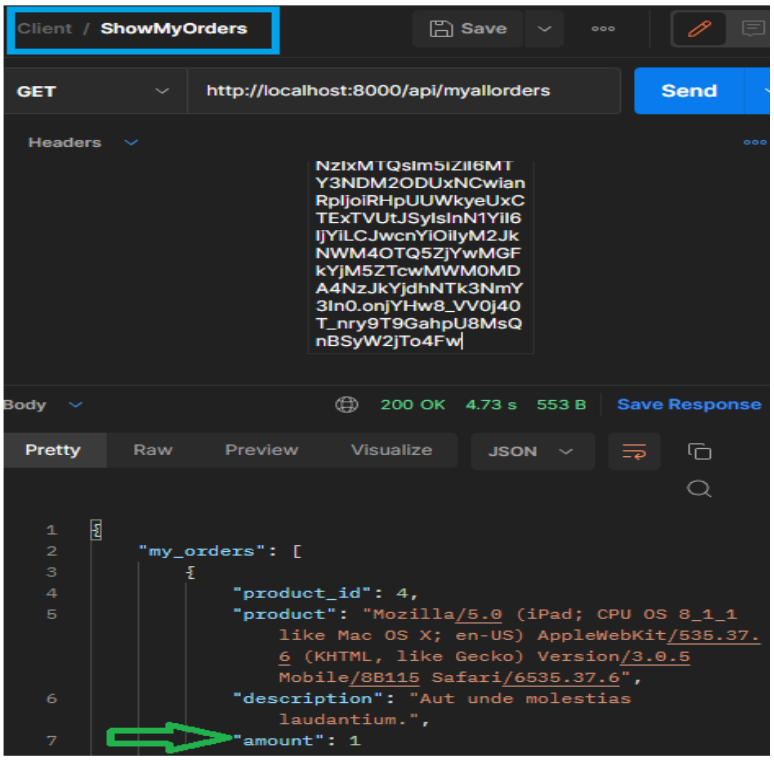


Fig. 53: Pedidos realizados por el cliente

Entonces, se procede a ingresar el cambio de cantidad a una mayor, **Fig. 54**. Tomando en cuenta la generación del *token* anterior, es oportuno agregarlo de la manera que se indica en la **Fig. 55**. Es así como se muestra el mensaje de confirmación del pedido actualizado. Finalmente se verifica su actualización, **Fig. 56**.

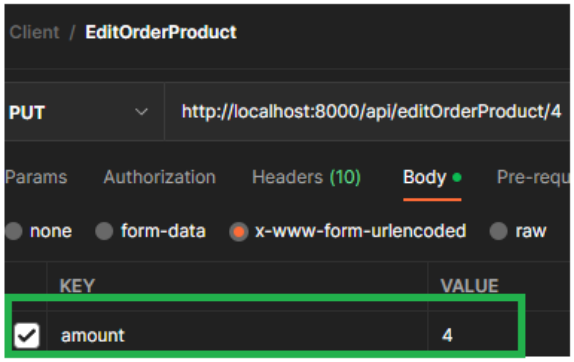


Fig. 54: Modificación del pedido

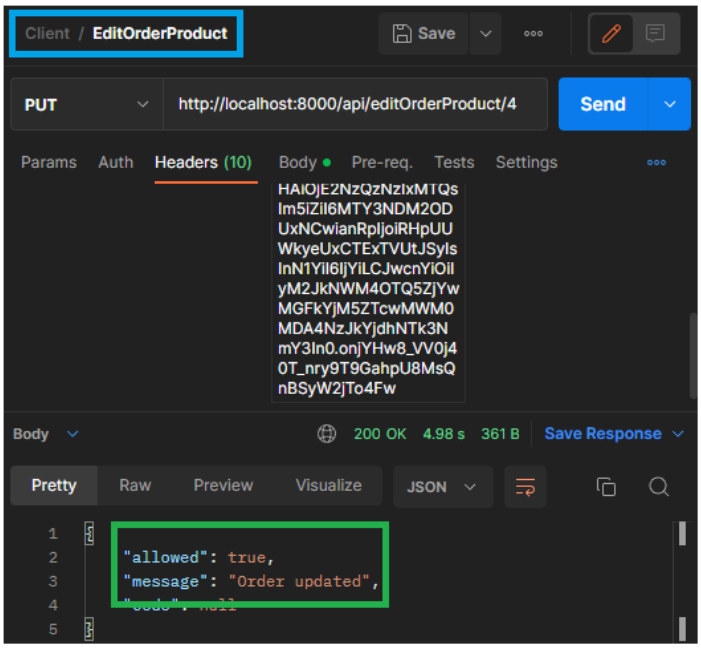


Fig. 55: Confirmación del pedido modificado

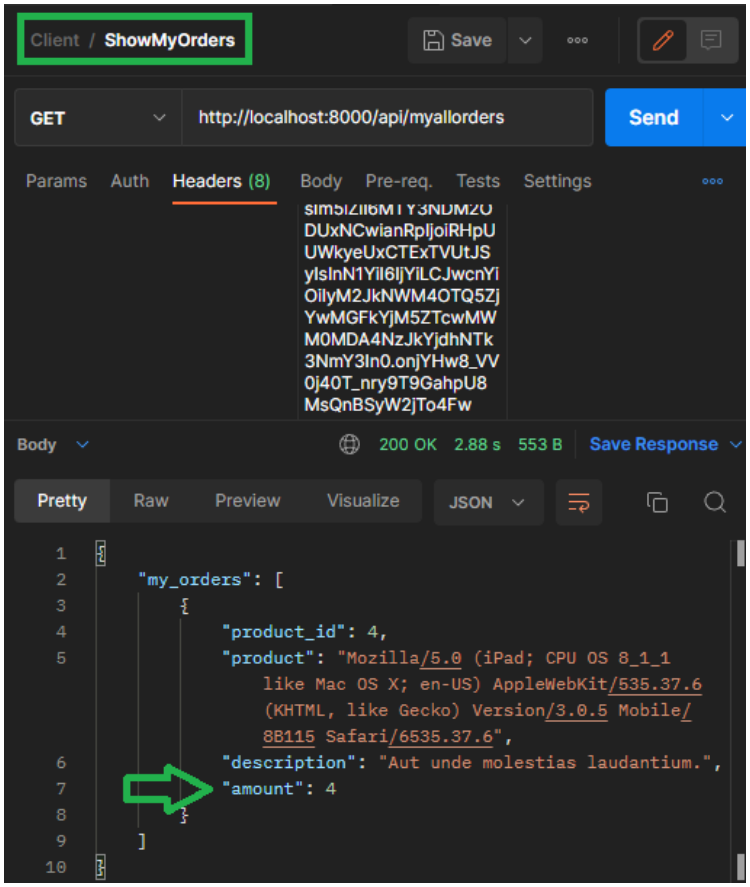


Fig. 56: Lista de productos realizados con actualización del producto

.

3.7 *Sprint* 6: Pruebas unitarias de la *API REST*

Antes de realizar el despliegue del sistema a producción es necesario realizar las pruebas unitarias de desarrollo de *API REST*, ya que permite comprender y asegurar la funcionalidad de un programa antes de una interacción con un usuario real [17]. Por ende, se propone las siguientes actividades:

- Pruebas unitarias a rutas.
- Comprobación de llamadas al servidor por medio de la *API* utilizando una aplicación cliente.
- Análisis de respuestas de cada componente del sistema que realice consulta.

Pruebas unitarias a rutas

Cuando se estableció las rutas *API*, se declaró que habría rutas públicas que obtienen información del sistema, como son los productos a la venta y los comentarios de dichos productos promocionados. La **Fig. 57**, muestra la declaración de las mismas.

routes > api.php

```
29
30    Route::get('/showproducts', [ClientController::class, 'showproducts']);
31    Route::get('/feedbacks', [FeedbackController::class, 'showfeedbacks']);
32
```

Fig. 57: Rutas públicas de los productos y comentarios

Esto llevará consigo a que se deba crear una función que se encargue de probar la petición *GET* de mostrar todos los productos, **Fig. 58**. De modo que, si se realiza una primera prueba unitaria a dichas ruta, el resultado será como se indica en la **Fig. 59**. Véase la sección de

ANEXOS en donde se evidencia otra prueba unitaria y una prueba de estrés.

Fig. 58: Función prueba unitaria a la ruta *Api* de productos

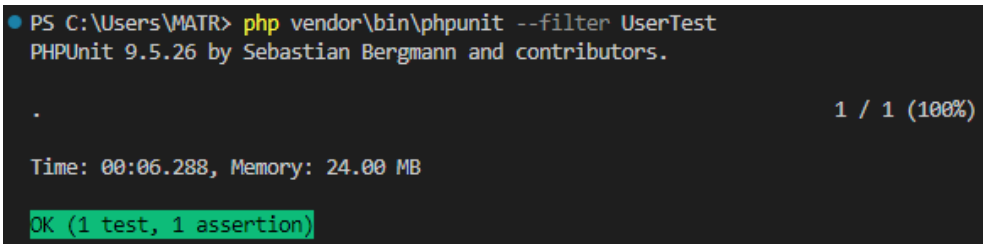


Fig. 59: Resultado prueba unitaria

Comprobación de llamadas al servidor por medio de la *API* utilizando una aplicación cliente

De la misma manera en que se pudo realizar la comprobación del apartado anterior, también se puede hacerlo mediante otras herramientas, tal es el caso de *Postman*. En la **Fig. *60***, se observa la obtención de la misma ruta *API* que muestra los productos ofertados por los propietarios.

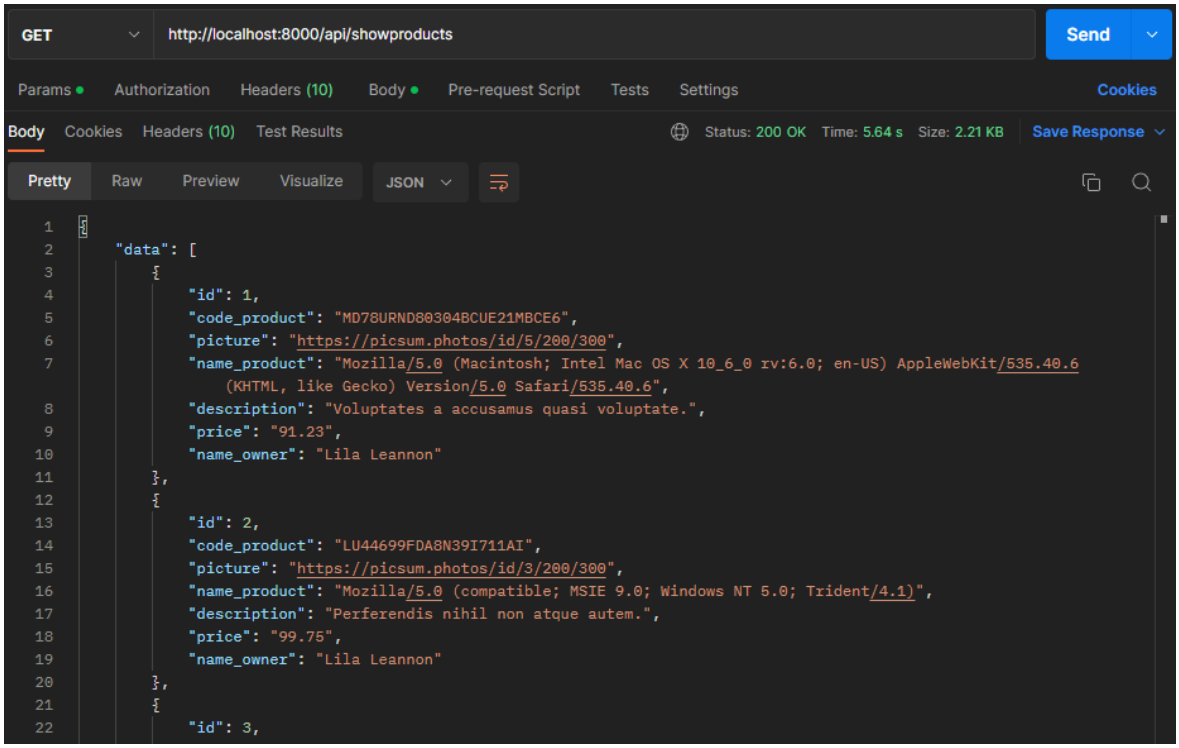


Fig. 60: Consumo de *API* con *Postman*

Por otra parte, si se desea obtener una *API* como respuesta de una petición *GET* para manipular los datos, estos deberán otorgarse a los usuarios que son dueños de los mismos. Nótese en la **Fig. 61**, en donde se indica la denegación de la misma.

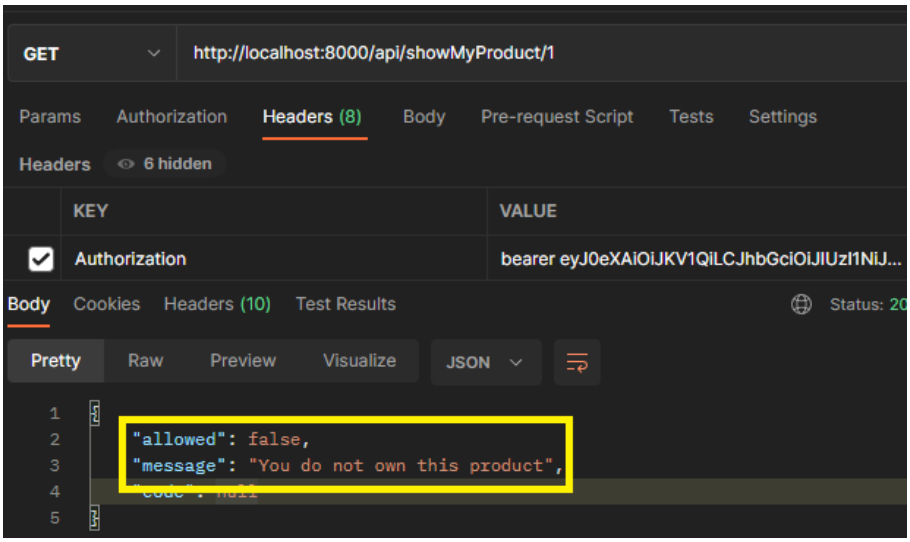


Fig. 61: Denegación de acceso a un recurso no propio del usuario

Análisis de respuestas de cada componente del sistema que realice consulta

Siempre que se realiza una consulta mediante una solicitud *http* se recibirá una respuesta por el lado de la *API,* existiendo métodos *GET*, *POST*, *PUT*, *DELETE*. Es así que al

momento en el que se realice una prueba de alguno de estos métodos se intenta ejecutar y comprobar todas las rutas existentes, y así establecer si poseen la respuesta esperada mediante un estado de solicitud, ya sea público o no. A continuación, se indica el estado de una solicitud realizada, **Fig. 62** .

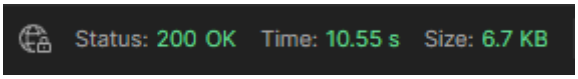


Fig. 62: Estado de solicitud GET

3.8 *Sprint* 7: Despliegue del proyecto de *API REST*

En el último paso a realizarse, es necesario llevar a cabo el despliegue del sistema *Backend* en un servidor público. Esto con el objetivo de que ya no se lo pueda ejecutar y probar localmente. Por ello, las tareas a realizarse son:

- Configuración del servidor *web*.
- Rendimiento del sistema.
- Comprobación de la seguridad del sistema.
- Prueba Unitaria *Sprint* 7.

Configuración del servidor *web*

Debido a que se requiere desplegar el proyecto de *Backend*, es necesario un *host* público encargado de contener dicho proyecto. *Railway* se acopla perfectamente a este trabajo, no solo por su alojamiento en la *web* pública, sino por otras implementaciones que se detallarán *más adelante*.

Por consiguiente, para que esta plataforma haga su trabajo se requiere agregar las variables de entorno del archivo .env del proyecto de *Laravel*. A modo de ejemplo se indica la **Fig. 63**, en donde existe la agregación de las mismas hacia *Railway*.

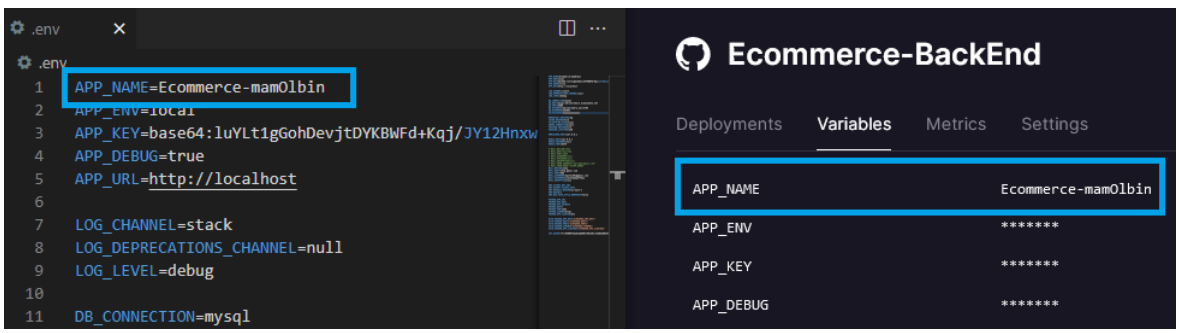


Fig. 63: Variables de entorno en *Railway*

Además, dada la creación con anterioridad del repositorio y la constante actualización del proyecto en *GitHub*, este agilizará los cambios existentes del proyecto en *Railway*. Por tal motivo, se recomienda tener enlazado el repositorio con dicho servidor.

Rendimiento del sistema

Como ya se mencionó antes, *Railway* trae consigo herramientas útiles en el sistema, por ejemplo, la actualización de la tabla productos mostrada por el *CPU* con un rango cercano a 50%, realizada en la fecha que se indica en la **Fig. *64***.

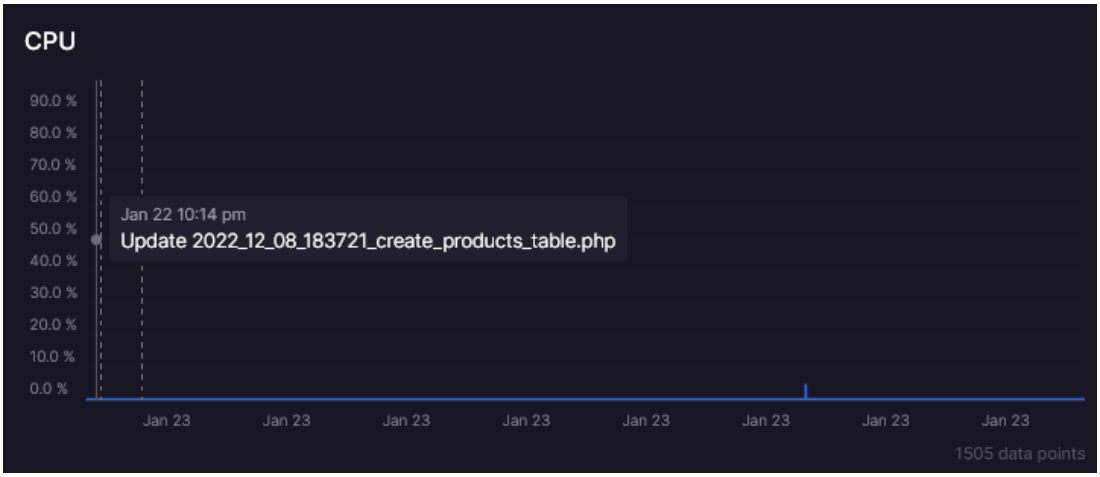


Fig. 64: Métrica *CPU*

Asimismo, con la cantidad de memoria en ejecución de 171 MB en un intervalo de tiempo, **Fig. *65***. O también con información de la red de entrada y salida, **Fig. *66*** y **Fig. *67***.

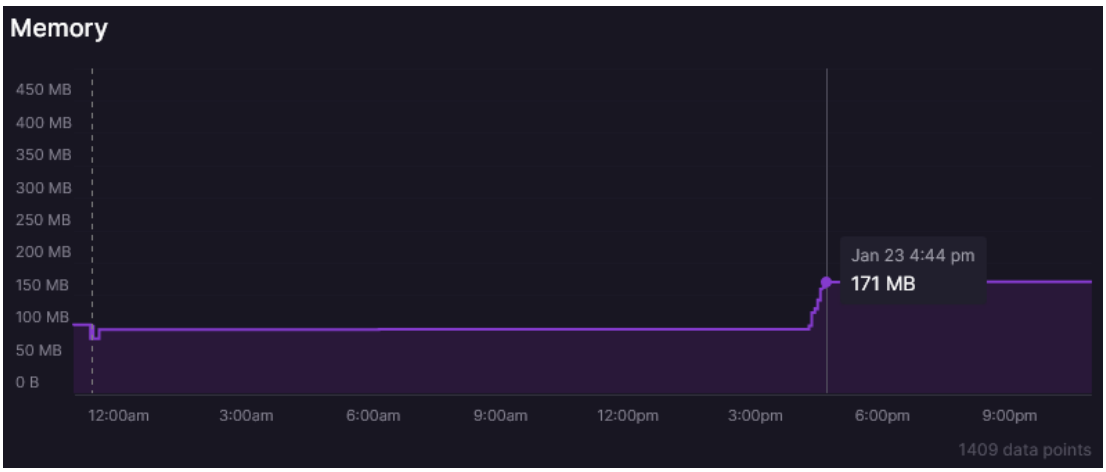


Fig. 65: Métrica Memoria

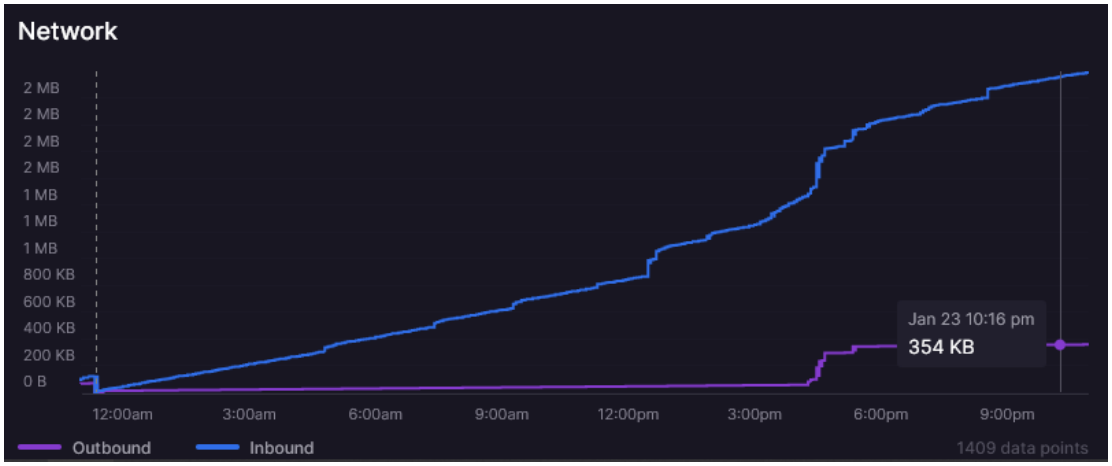


Fig. 66: Métrica red saliente

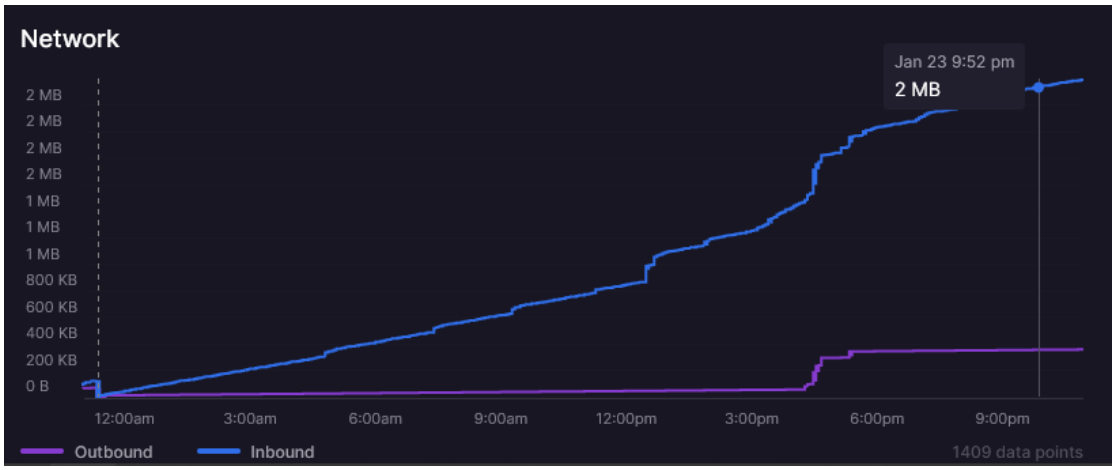


Fig. 67: Métrica red entrante

Comprobación de la seguridad del sistema

Formalmente, la implementación de la seguridad en un sistema es la práctica en la que se debe proteger a un sitio *web* y/o sistema para evitar el uso, acceso, modificación o eliminación, de entidades no autorizadas [18]. Es por ello por lo que la seguridad también está puesta en funcionamiento por parte de *Railway*. En las figuras **Fig. *68*** y **Fig. *69***, se corrobora la conexión cifrada existente por parte de la plataforma, de igual manera con la conexión segura al sitio.

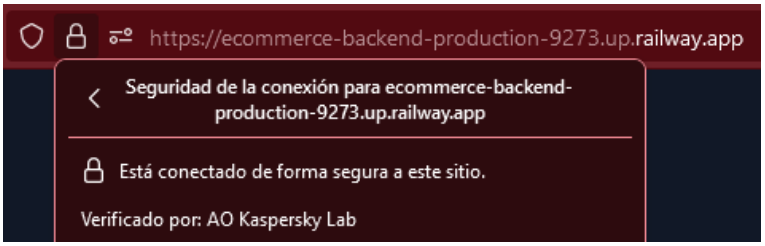


Fig. 68: Conexión segura al sitio *web*

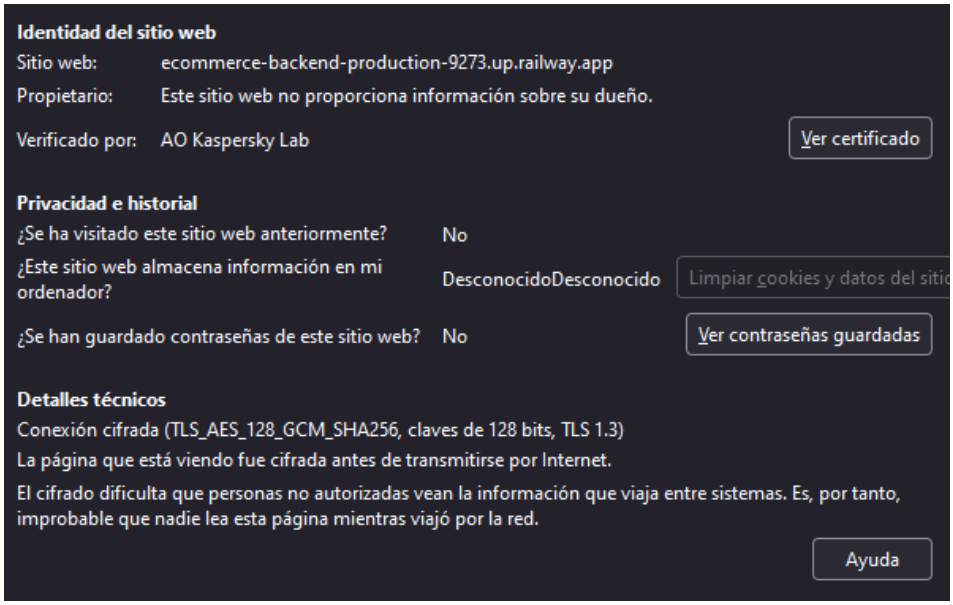


Fig. 69: Información del sitio *web*

Cabe recalcar que al mismo tiempo este sistema está regulado por la autenticación basada en *token* (JWT), lo que genera una seguridad adicional en el desarrollo de este *Backend*.

Prueba unitaria *Sprint* 7

Por último, para la prueba unitaria de este *Sprint* se realiza el consumo de la *API* pública en la herramienta de *Postman*, en donde se verifica el funcionamiento de la *url* proporcionada por *Railway*. De este modo, se obtiene los comentarios realizados por algunos de los clientes del sistema, **Fig. *70***, confirmando su correcta ejecución tanto local y públicamente.

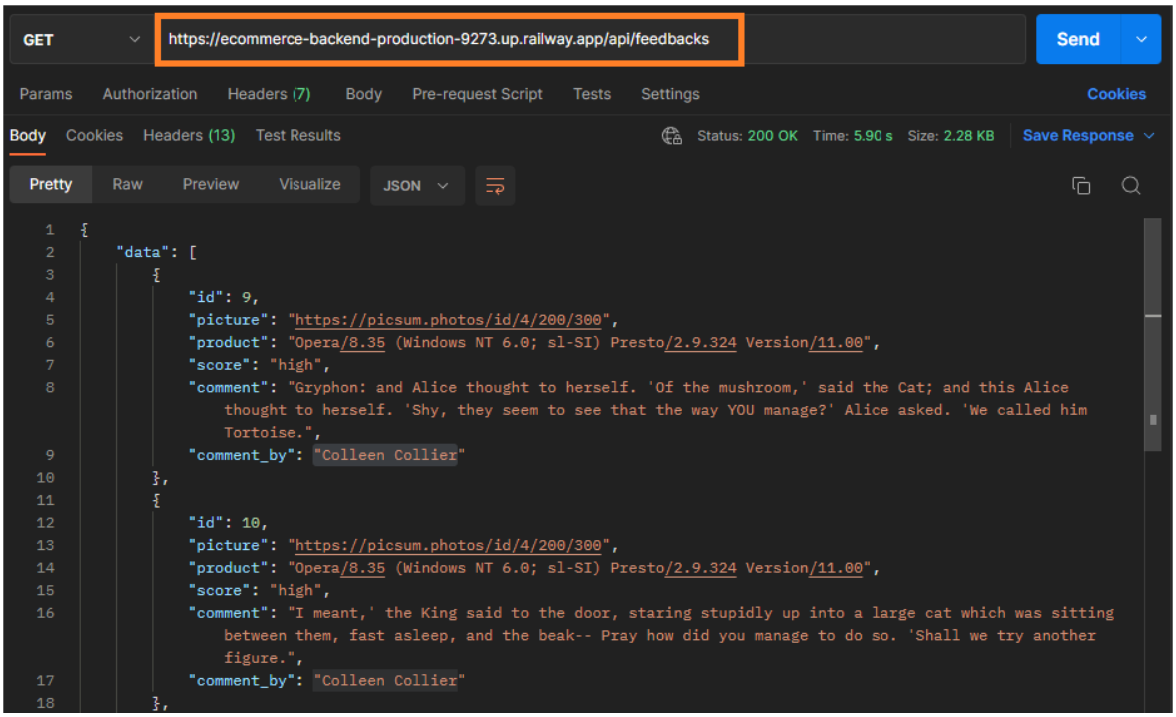


Fig. 70: Obtención de comentarios con *URL* en producción

4 CONCLUSIONES

- El desarrollo del componente Backend de *Ecommerce* para la venta de equipos computacionales cumple con los objetivos acorde a los requerimientos previstos y alcance propuesto. Por lo que, si existe una implementación con *Frontend* o aplicación móvil, el sistema está en la capacidad de solventar las necesidades para el consumo de los recursos por parte del *Backend*.
- Cada uno de los requerimientos logran cumplirse con las solicitudes de acuerdo con el perfil de los usuarios. Además de que estos requerimientos pueden irse alterando conforme se vayan desarrollando, debido a que existe la posibilidad de dotar una mejora para el usuario final, y en pocas ocasiones quitar alguna.
- Existe el logro de una notable esquematización del modelo, por lo que éste alcanza a desempeñar la correcta ejecución de las relaciones de las tablas de la base de datos al momento de realizar migraciones. Es decir, si se consigue un óptimo modelo para la base de datos relacional.
- La implementación de cada uno de los *Endpoints* logra el correcto funcionamiento del sistema de *Ecommerce*, para su verificación se realizan pruebas unitarias y pruebas de integración antes del despliegue a producción.
- *Railway* posee las herramientas necesarias para realizar el despliegue de un proyecto de *software*, lo que la convierte en una plataforma de infraestructura sólida. No obstante, posee la desventaja de un tiempo límite de 500 horas de un proyecto publicado. Por otra parte, algunos de sus puntos altos son el despliegue a producción con un protocolo HTPPS y de conexión cifrada, o herramientas útiles como la observación del rendimiento del sistema mediante gráficas o estadísticas.

5 RECOMENDACIONES

- En la futura implementación de un *Frontend* es relevante realizar una correcta relación a nivel de Eloquent y bases de datos, dado que si no existe una efectiva implementación de la misma cabe la posibilidad de un trabajo lento.
- La realización de migración hacia la base de datos, se ejecutan correctamente siempre y cuando haya un orden adecuado. Más aún si las tablas de migración poseen claves foráneas. Asimismo, el manejo correcto de los conceptos que engloban la elaboración de un proyecto de *Software*, en este caso *Backend.*
- La implementación correcta de *Eloquent Resources* en los controladores para hacer uso de los datos que pertenecen a otra entidad, de lo contrario no se obtiene la información deseada.
- Al momento de utilizar la herramienta de *Postman*, y se requiera hacer una actualización de un dato se recomienda hacerlo mediante *x-www-form-urlencoded*, de lo contrario no se puede realizar la actualización y envés de ello hace una petición *GET*.
- Si se desea realizar una publicación se recomienda hacerlo mediante *form-data*, cabe recalcar que no existen percances si se emplea *x-www-form-urlencoded*. Sin embargo, como es una publicación se lo hace mediante un formulario.
- Con respecto a la confirmación de correo electrónico, se aconseja poseer una cuenta de *Gmail* para generar las credenciales, que posteriormente son colocadas en el archivo .env, por lo que si no existe dichas credenciales se presentan errores como el de *SMTP*.
- Finalmente, se recomienda para un uso futuro la implementación de un *Frontend*, para que trabaje en conjunto al *Backend*. De igual forma realizar el despliegue en otro proveedor y/o herramienta, debido a las limitaciones que presenta *Railway*, tomando en cuenta que *Railway* es una excelente plataforma.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] J. Espinosa, «La forma de vender ha cambiado: Evolución o muerte de las marcas,» 18 Junio 2019. [En línea]. Available: https://n9.cl/opnvw. [Último acceso: 22 Noviembre 2022].

[2] RedHat, «¿Qué es la metodología ágil?,» 19 Julio 2022. [En línea]. Available: https://www.redhat.com/es/devops/what-is-agile-methodology. [Último acceso: 22 Noviembre 2022].

[3] OpenWebinars, «Qué es un lenguaje de programación,» 16 Julio 2020. [En línea]. Available: https://openwebinars.net/blog/que-es-un-lenguaje-de-programacion/. [Último acceso: 02 Diciembre 2022].

[4] freeCodeCamp, «What is PHP? The PHP Programming Language Meaning Explained,» 30 Agosto 2021. [En línea]. Available: https://www.freecodecamp.org/news/what-is-php-the-php-programming-language-meaning-explained/. [Último acceso: 22 Noviembre 2022].

[5] Vercel Inc., «Railway,» 22 Enero 2022. [En línea]. Available: https://vercel.com/integrations/railway. [Último acceso: 31 Enero 2023].

[6] Concepto, «Metodología,» 05 Agosto 2021. [En línea]. Available: https://concepto.de/metodologia/. [Último acceso: 30 Noviembre 2022].

[7] E. Maida y J. Pacienzia, «Metodologías de desarrollo de software,» Diciembre 2015. [En línea]. Available: https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf. [Último acceso: 30 Noviembre 2022].

[8] Northware, «Requerimientos en el desarrollo de software y aplicaciones,» 26 Mayo 2022. [En línea]. Available: https://www.northware.mx/blog/requerimientos-en-el-desarrollo-de-software-y-aplicaciones/. [Último acceso: 30 Noviembre 2022].

[9] M. Rehkopf, «Historias de usuario con ejemplos y plantilla,» Atlassian, 15 Marzo 2018. [En línea]. Available: https://www.atlassian.com/es/agile/project-management/user-stories. [Último acceso: 30 Noviembre 2022].

[10] Integra IT, «Sprint y Sprint Backlog: puntos esenciales de SCRUM,» 27 Marzo 2018. [En línea]. Available: https://integrait.com.mx/blog/sprint-y-sprint-backlog/. [Último acceso: 30 Noviembre 2022].

[11] Mozilla Corporation, «MVC,» 29 Noviembre 2022. [En línea]. Available: https://developer.mozilla.org/es/docs/Glossary/MVC. [Último acceso: 30 Noviembre 2022].

[12] Ubiqum, «11 herramientas de desarrollo de software que te harán más productivo,» 6 Junio 2019. [En línea]. Available: https://ubiqum.com/es/blog/20-herramientas-de-desarrollo-de-software-que-te-haran-mas-productivo/. [Último acceso: 03 Diciembre 2022].

[13] Gub.uy, «Arquitectura de Datos-Qué es la Arquitectura de Datos,» 2 Agosto 2019. [En línea]. Available: https://www.gub.uy/agencia-gobierno-electronico-sociedad-informacion-conocimiento/politicas-y-gestion/arquitectura-datos. [Último acceso: 17 Diciembre 2022].

[14] SiteGround, «Características de phpMyAdmin,» 19 Marzo 2021. [En línea]. Available: https://www.siteground.es/tutoriales/tutoriales-phpmyadmin/caracteristicas/. [Último acceso: 18 Diciembre 2022].

[15] V. Peña, «Seeders y Factories en Laravel,» Norvic Software, 27 Septiembre 2021. [En línea]. Available: https://norvicsoftware.com/seeders-y-factories-en-laravel/. [Último acceso: 30 Diciembre 2022].

[16] D. Palacios, «Controladores en Laravel,» Styde, 27 Octubre 2017. [En línea]. Available: https://styde.net/controladores-en-laravel/. [Último acceso: 18 Enero 2023].

[17] LoadView, «Tutorial de pruebas de automatización de API REST,» 04 Noviembre 2020. [En línea]. Available: https://www.loadview-testing.com/es/blog/tutorial-de-pruebas-de-automatizacion-de-api-rest/. [Último acceso: 24 Enero 2023].

[18] Mozilla Corporation's, «Seguridad de Sitios Web,» 29 Noviembre 2022. [En línea]. Available: https://developer.mozilla.org/es/docs/Learn/Server-side/First_steps/Website_security. [Último acceso: 24 Enero 2023].

7 ANEXOS

- **ANEXO I.** Certificación de originalidad.
- **ANEXO II.** Manual Técnico.
- **ANEXO III.** Manual de Usuario.
- **ANEXO IV.** Manual de Instalación.

ANEXO I

En el siguiente anexo en base a la herramienta *Turnitin* se evidencia el resultado de originalidad del documento desarrollado, además el certificado del director de tesis de acuerdo con dicho resultado.

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

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 27 de Febrero de 2023

De mi consideración:

Yo, MAYRA ISABEL ALVAREZ JIMÉNEZ, en calidad de directora del Trabajo de Integración Curricularcomponente **BACKEND**, titulado "**DESARROLLO DE ECOMMERCE PARA LA VENTA DE EQUIPOS COMPUTACIONALES**", elaborado por el estudiante **MARLON ADRIAN TUQUERRES ROMERO** de la carrera en DESARROLLO DE SOFTWARE, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo (excepto citas bibliográficas), producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 11%.

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,

Ing. Mayra Alvarez MSc.
Técnico docente EPN - ESFOT

ANEXO II

Con respecto a la Recopilación de los Requerimientos, *Sprint Backlog,* Historias de Usuario, *Product Backlog*, y el apartado de pruebas que completa la sección del *Sprint* 4, este anexo presenta a detalle cada una de estas realizaciones.

Recopilación de requerimientos

Se muestra en la **TABLA *X***, cada uno de los requerimientos recopilados en base a reuniones y charlas con el *Product Owner* durante la etapa de planificación

TABLA X: Recopilación de Requerimientos

Recopilación de Requerimientos			
ID	**Nombre**	**Descripción**	**Usuario**
RR-001	Ingresar al sistema	A fin de acceder al sistema, cada uno de los usuarios debe estar vigente con las apropiadas credenciales.	Todos los usuarios
RR-002	Consultar y eliminar usuarios	El Administrador global gestiona a los demás usuarios mediante la eliminación y observación para llevar un control de los mismos.	Administrador
RR-003	Consultar y eliminar publicaciones	El usuario puede observar y banear a una publicación de los productos ofertados por un propietario, con el objetivo de mantener respetabilidad en las publicaciones.	Administrador
RR-004	Comunicación con los usuarios	Al usuario Administrador se le permitirá comunicarse con cualquier usuario mediante correo o número de celular, con el objetivo de proporcionar un mejor servicio como sistema *web*.	Administrador
RR-005	Observación: Sección calificación/comentarios	El sistema permitirá al administrador ver la sección de calificación/comentarios, con el objetivo de estar atento hacia alguna de ellas, en caso necesario.	Administrador
RR-006	Gestionar información del negocio	El Propietario puede cambiar el nombre.	Propietario
RR-007	Gestionar productos del negocio	El Propietario puede gestionar cada uno de sus productos que	Propietario

		estén ofertándose en su negocio para la venta, mediante la observación, creación, modificación y/o eliminación.	
RR-008	Facilitar *stock* disponible de productos	El Propietario puede observar el número de sus productos disponibles de su negocio, a fin de tomar decisiones futuras para aumentar su *stock*.	Propietario
RR-009	Gestionar perfil del usuario	El sistema *web* permite que el usuario modifique su cuenta de perfil, con la intención de cambiar su número de celular, dirección y/o *email*.	Todos los usuarios
RR-010	Modificar y recuperar contraseña	Cualquier usuario puede modificar su contraseña por una nueva. Y en el caso de olvido entonces podrá restablecer una nueva.	Todos los usuarios
RR-011	Listar productos comprados	El sistema otorga al cliente la lista de los productos que haya comprado con el objetivo de observar cada uno de los productos que ha hecho.	Cliente
RR-012	Mostrar detalle de compra	El cliente podrá ver el detalle de cualquiera de sus pedidos.	Cliente
RR-013	Mostrar productos ofertados de los negocios	El sistema acepta que el usuario cliente vea las publicaciones existentes de los productos con la finalidad de un posible interés. El sistema permite la visualización al administrador de los productos publicados, para que no exista un producto inadecuado a la venta.	Cliente
RR-014	Solicitar pedido	El usuario podrá realizar un pedido de algún producto para satisfacer su necesidad de compra.	Cliente
RR-015	Modificar de pedido	El sistema debe posibilitar la modificación de pedidos con el aumento o disminución de cantidad deseada.	Cliente
RR-016	Comunicación con el dueño del producto	El aplicativo *web* posibilita la comunicación del cliente y propietario una vez realizado el pedido, con el objetivo de mejorar la experiencia de comunicación con el dueño del negocio	Cliente

RR-017	Listar de comentarios	El sistema acepta que el usuario cliente vea los comentarios existentes de los productos con la finalidad de un posible pedido. El sistema permite la visualización al administrador de los comentarios existentes, para que no exista un comentario inadecuado publicado.	Cliente
RR-018	*CRUD* comentarios	Al usuario cliente se le brindará la oportunidad de gestionar sus comentarios realizados a los productos ofertados por los negocios.	Cliente
RR-019	Probar y desplegar el sistema	A cualquier usuario que ingrese al sistema deberá ser capaz de realizar las acciones que se prevean y correspondan.	Todos los usuarios

Historias de Usuario

Una vez concluida la recopilación de cada uno de los requerimientos, es necesario la preparación de las "Historias de Usuario" para el aplicativo de *Backend*. Por ello, se muestran las elaboraciones de las mismas en base a la recopilación de requerimientos, comenzando desde la **TABLA XI** hasta la **TABLA XXIX**.

TABLA XI: Historia de usuario - Ingreso al sistema

Historia de Usuario	
Identificador (ID): HU001	**Usuario**: Todos los usuarios
Nombre de historia: Ingresar al sistema	
Prioridad: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: La ejecución de esta acción permite el ingreso al sistema de todos los usuarios.	
Observación: Cada uno de los respectivos usuarios podrá incorporarse al sistema según su interés.	

TABLA XII: Historia de usuario – Consulta/Eliminación de usuarios

Historia de Usuario	
Identificador (ID): HU002	**Usuario**: Administrador
Nombre de historia: Consultar y eliminar usuarios	
Prioridad: Media	**Riesgo en desarrollo:** Alto

Iteración Asignada: 1
Encargado: Marlon Adrian Tuquerres Romero
Descripción: Debo gestionar todos los usuarios que vayan ingresando al sistema.
Observación: El Administrador global podrá de gestionar a los demás usuarios mediante la eliminación y observación.

TABLA XIII: Historia de usuario – Consulta /Eliminación de publicaciones

Historia de Usuario	
Identificador (ID): HU003	**Usuario**: Administrador
Nombre de historia: Consultar y eliminar publicaciones	
Prioridad: Alta	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Necesito ver las publicaciones de los productos ofertados de los usuarios propietarios.	
Observación: El Administrador puede observar y banear a una publicación de los productos ofertados por un propietario.	

TABLA XIV: Historia de usuario - Comunicación con los usuarios

Historia de Usuario	
Identificador (ID): HU004	**Usuario**: Administrador
Nombre de historia: Faciltar la comunicación con los usuarios	
Prioridad: Media	**Riesgo en desarrollo:** Alto
Iteración Asignada: 1	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Cuando sea necesario debo comunicarme con los usuarios registrados en el sistema.	
Observación: El usuario Administrador podrá comunicarse con cualquier usuario mediante correo electrónico.	

TABLA XV: Historia de usuario - Consulta: calificación/comentarios

Historia de Usuario	
Identificador (ID): HU005	**Usuario**: Administrador
Nombre de historia: Mostrar sección calificación/comentarios	
Prioridad: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Requiero ver la sección de calificación/comentarios.	

Observación: El Administrador podrá ver la sección de calificación/comentarios y banearlo si es necesario por algo inapropiado.

TABLA XVI: Historia de usuario - Gestión de negocio

Historia de Usuario	
Identificador (ID): HU006	**Usuario**: Propietario
Nombre de historia: Gestionar información del negocio	
Prioridad: Alta	**Riesgo en desarrollo:** Media
Iteración Asignada: 2	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Necesito gestionar mi negocio, cambiando el nombre de mi negocio.	
Observación: El propietario puede modificar el nombre de su negocio.	

TABLA XVII: Historia de usuario - Gestión de productos del negocio

Historia de Usuario	
Identificador (ID): HU007	**Usuario**: Propietario
Nombre de historia: Gestionar productos del negocio	
Prioridad: Alta	**Riesgo en desarrollo:** Alto
Iteración Asignada: 2	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Necesito gestionar los productos de mi negocio.	
Observación: El Propietario podrá crear, modificar, observar o eliminar los productos que tenga en su negocio.	

TABLA XVIII: Historia de usuario - Facilitación de stock de productos

Historia de Usuario	
Identificador (ID): HU008	**Usuario**: Propietario
Nombre de historia: Facilitar de stock disponible de productos	
Prioridad: Alta	**Riesgo en desarrollo:** Alto
Iteración Asignada: 1	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Me gustaría ver el número disponible de mi stock conforme los clientes vayan haciendo pedidos.	
Observación: El usuario Propietario puede observar la cantidad de productos que aún tiene disponible en su negocio.	

TABLA XIX: Historia de usuario - Gestión de perfil del usuario

Historia de Usuario

Identificador (ID): HU009	**Usuario**: Todos los usuarios
Nombre de historia: Gestionar perfil del usuario	
Prioridad: Media	**Riesgo en desarrollo:** Baja
Iteración Asignada: 1	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Requiero modificar mi cuenta de perfil para actualizar mi número de celular, email, dirección.	
Observación: El usuario registrado puede cambiar la información personal.	

TABLA XX: Historia de usuario – Modificación/recuperación de contraseña

Historia de Usuario	
Identificador (ID): HU010	**Usuario**: Todos los usuarios
Nombre de historia: Modificar y recuperar contraseña	
Prioridad: Alta	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Requiero restablecer mi contraseña porque no recuerdo mi clave de ingreso al sistema o modificarla.	
Observación: Todos los usuarios pueden modificar su contraseña por otra. En el caso de pérdida pueden restablecer una nueva con el correo electrónico registrado.	

TABLA XXI: Historia de usuario - Lista productos comprados

Historia de Usuario	
Identificador (ID): HU011	**Usuario**: Cliente
Nombre de historia: Listar productos comprados.	
Prioridad: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Necesito observar mi lista de pedidos realizados.	
Observación: El usuario cliente tiene la posibilidad de ver la lista de todos sus pedidos realizados.	

TABLA XXII: Historia de usuario - Detalle de compra

Historia de Usuario	
Identificador (ID): HU012	**Usuario**: Cliente
Nombre de historia: Mostrar detalle de compra.	
Prioridad: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	

Encargado: Marlon Adrian Tuquerres Romero
Descripción: Necesito observar mi lista de pedidos realizados.
Observación: El usuario cliente tiene la posibilidad de ver la información de sus pedidos.

TABLA XXIII: Historia de usuario - Productos ofertados por un negocio

Historia de Usuario	
Identificador (ID): HU013	**Usuario**: Cliente
Nombre de historia: Mostrar productos ofertados de los negocios.	
Prioridad: Alta	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Quiero ver las publicaciones de los productos ofertados.	
Observación: El cliente puede ver los productos publicados con la finalidad de un posible interés.	

TABLA XXIV: Historia de usuario - Solicitar pedido

Historia de Usuario	
Identificador (ID): HU014	**Usuario**: Cliente
Nombre de historia: Solicitar pedido.	
Prioridad: Alta	**Riesgo en desarrollo:** Media
Iteración Asignada: 2	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Quiero ya hacer el pedido de mi producto.	
Observación: El usuario Cliente puede adquirir su producto deseado.	

TABLA XXV: Historia de usuario - Modificación de pedido

Historia de Usuario	
Identificador (ID): HU015	**Usuario**: Cliente
Nombre de historia: Modificar pedido.	
Prioridad: Alta	**Riesgo en desarrollo:** Alto
Iteración Asignada: 2	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Quiero reducir o aumentar la cantidad de mi pedido.	

Observación: El usuario Cliente puede modificar la cantidad de su pedido según lo que requiera.

TABLA XXVI: Historia de usuario - Comunicación con el dueño del producto

Historia de Usuario	
Identificador (ID): HU016	**Usuario**: Cliente
Nombre de historia: Facilitar la comunicación con el dueño del producto.	
Prioridad: Media	**Riesgo en desarrollo:** Alto
Iteración Asignada: 1	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Cuando sea necesario quiero contactarme con el propietario del negocio acerca de mi pedido realizado.	
Observación: El sistema debe posibilitar la comunicación del usuario cliente hacia el propietario.	

TABLA XXVII: Historia de usuario - Lista de comentarios

Historia de Usuario	
Identificador (ID): HU017	**Usuario**: Cliente
Nombre de historia: Listar de comentarios	
Prioridad: Alta	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Quiero ver las publicaciones de los comentarios realizados.	
Observación: El sistema debe posibilitar la observación de los comentarios existentes de los productos ofertados.	

TABLA XXVIII: Historia de usuario – *CRUD* comentarios

Historia de Usuario	
Identificador (ID): HU018	**Usuario**: Cliente
Nombre de historia: Implementar *CRUD* comentarios	
Prioridad: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Quiero modificar, eliminar el comentario que hice de un producto. Me gustaría realizar un comentario.	
Observación: El sistema debe posibilitar la creación, actualización, eliminación y observación del comentario realizado por un cliente.	

TABLA XXIX: Historia de Usuario - Probar y desplegar el sistema

Historia de Usuario	
Identificador (ID): HU019	**Usuario**: Todos los usuarios
Nombre de historia: Probar y desplegar el sistema	
Prioridad: Alta	**Riesgo en desarrollo:** Alta
Iteración Asignada: 1	
Encargado: Marlon Adrian Tuquerres Romero	
Descripción: Necesito ser capaz de realizar las acciones que se prevean y correspondan para interactuar correctamente en el sistema.	
Observación: El sistema debe posibilitar cada una de las acciones que correspondan a los distintos usuarios con sus respectivos roles.	

Product Backlog

En la siguiente tabla, ***TABLA XXX***, se la ha desarrollado de acuerdo con las Historias de Usuario, esta muestra una lista ordenada de los requerimientos del aplicativo *web*, con su respectiva identificación, historia de usuario, prioridad, iteración y estado.

TABLA XXX: *Product Backlog*

Product Backlog				
ID-HU	**Historia de Usuario**	**Prioridad**	**Iteración**	**Estado**
HU-001	Ingresar al sistema	Media	1	Finalizado
HU-002	Observación y eliminación de todos los usuarios	Media	1	Finalizado
HU-003	Observación y eliminación de publicaciones	Alta	1	Finalizado
HU-004	Comunicación con los usuarios	Media	1	Finalizado
HU-005	Observación: Sección calificación/comentarios	Media	1	Finalizado
HU-006	Gestión de negocio	Alta	2	Finalizado
HU-007	Gestión de productos del negocio	Alta	2	Finalizado
HU-008	Facilitación de stock disponible de productos	Alta	1	Finalizado
HU-009	Gestión de perfil del usuario	Media	1	Finalizado
HU-010	Modificación o recuperación de contraseña	Alta	1	Finalizado

HU-011	Lista productos comprados	Media	1	Finalizado
HU-012	Detalle de compra	Media	1	Finalizado
HU-013	Productos ofertados de los negocios	Alta	1	Finalizado
HU-014	Solicitar pedido	Alta	2	Finalizado
HU-015	Modificación de pedido	Alta	2	Finalizado
HU-016	Comunicación con el dueño del producto	Media	1	Finalizado
HU-017	Lista de comentarios	Alta	1	Finalizado
HU-018	CRUD comentarios	Media	1	Finalizado
HU-019	Pruebas y despliegue del sistema	Alta	1	Finalizado

Prueba Unitaria 2

Anteriormente se mencionó las **Pruebas unitarias a rutas**, en donde se realizó una primera prueba a los productos promocionados. En este apartado se hará una prueba adicional de los comentarios realizados en el sistema *web*, con el objetivo de corroborar su ejecución. La siguiente función, **Fig. 71**, da como respuesta el estado de la petición sobre la ruta *feedbaks* de tipo *API*.

Fig. 71: Función prueba unitaria a la ruta *Api* de comentarios

Para realizar la prueba se lo hace mediante la instrucción:

php vendor\bin\phpunit --filter UserTest

En dónde se obtiene la siguiente respuesta, **Fig. 72**, denotando que ejecuta correctamente la ruta de los comentarios publicados.

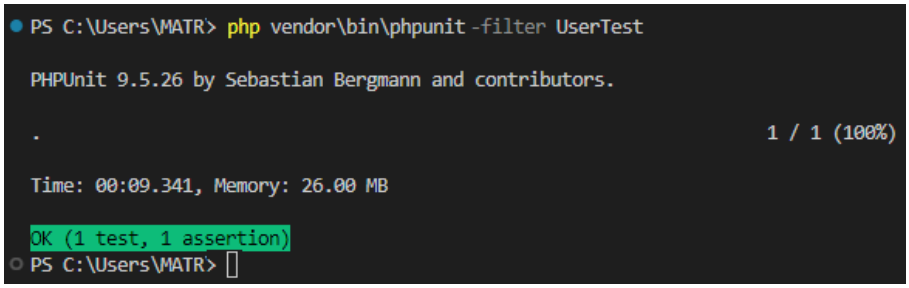


Fig. 72: Resultado prueba unitaria

Prueba de estrés

En cada proyecto de *software*, es necesario realizar distintas pruebas para verificar la funcionalidad de la misma, y es notorio que existen diversos tipos de pruebas. Una de ellas son las pruebas de estrés, esta se caracteriza por ser un tipo de prueba que determina los límites de un sistema poniéndolo en condiciones extremas, verificando su fiabilidad y estabilidad.

Una herramienta que posibilita hacer pruebas de estrés es *Apache JMeter*, y es la que se hará uso en este apartado.

A continuación, se denota la implementación de la prueba de estrés. En la figura, **Fig. 73**, se indica el estado inicial del sistema, en el que no llega ni a 10% del uso de su *CPU*.

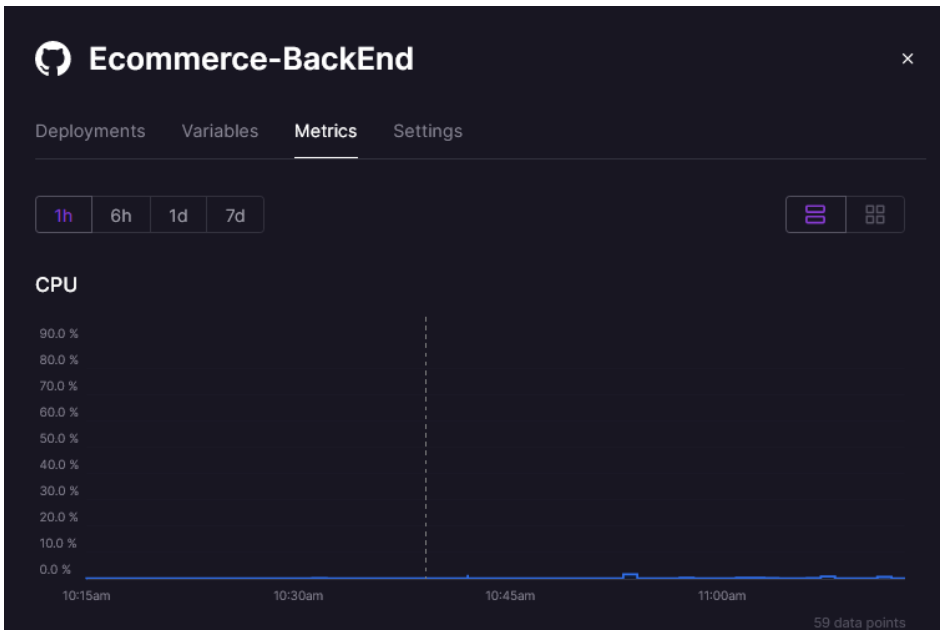


Fig. 73: Estado inicial *CPU*

Por lo tanto, en la herramienta de *Apache* se introduce los siguientes valores, **Fig. *74***. Estos indican que habrá 300 usuarios por 5 segundos y se estarán realizando 60 solicitudes por segundo (división de 300 entre 5). Así, ya se denota un cambio en el *CPU*, **Fig. *75***, trayendo

consigo cerca del 100% de su uso. A consecuencia de esto, en la figura **Fig. *76***, se muestran algunos errores de petición *GET* no obtenidos.

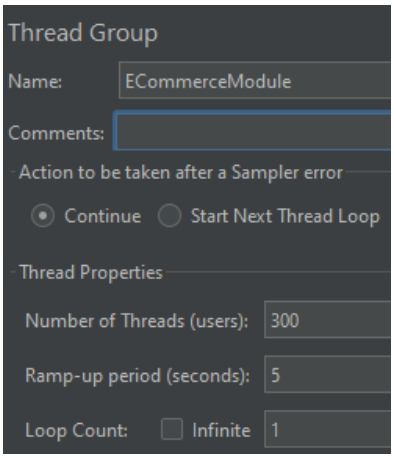


Fig. 74: *Test* con 300 solicitudes

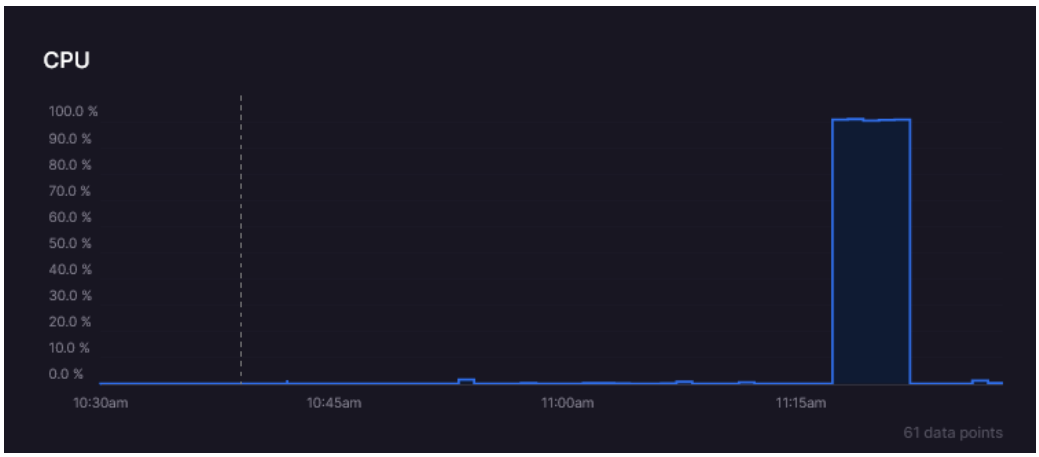


Fig. 75: Estado 2 del *CPU*

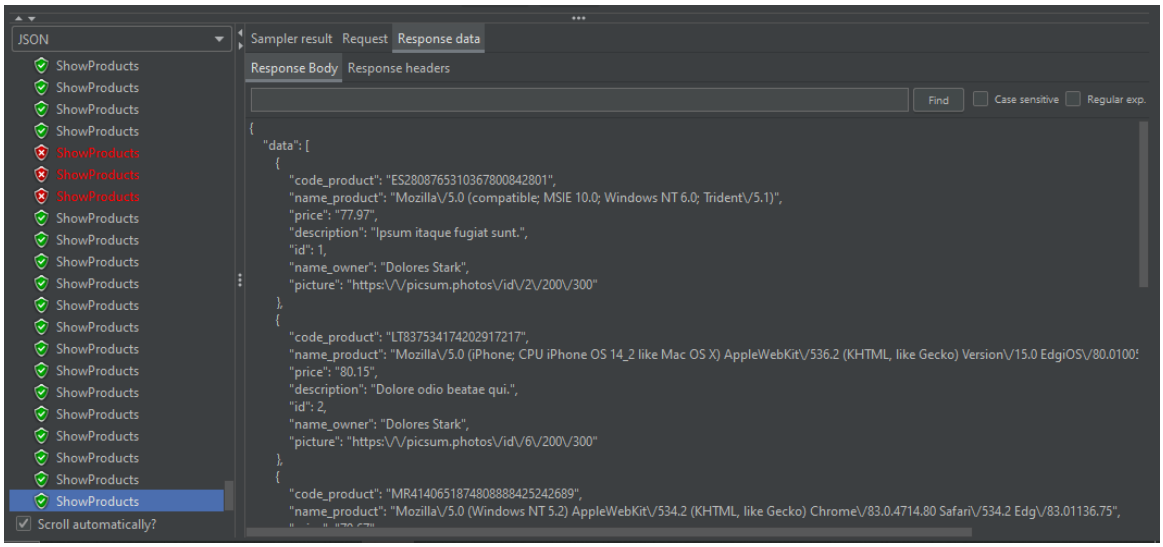


Fig. 76: Datos de respuesta de 300 solicitudes

Una prueba adicional en este apartado será mediante una condición extrema, con 200 solicitudes por segundo hasta cumplirse 1000 solicitudes, **Fig. *77***.

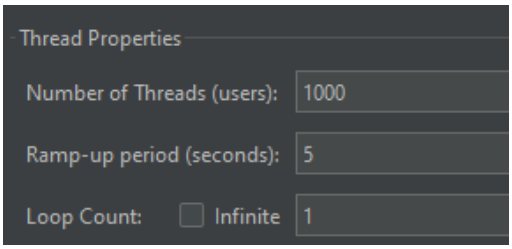


Fig. 77: *Test* con 1000 solicitudes

Es así como se denota la gran diferencia que existe en la prueba anterior con esta, ya que hace un uso de alrededor de 200% del *CPU*, **Fig. *78***. Además de presentar demasiados errores en la obtención de *ShowProducts*, **Fig. *79***.

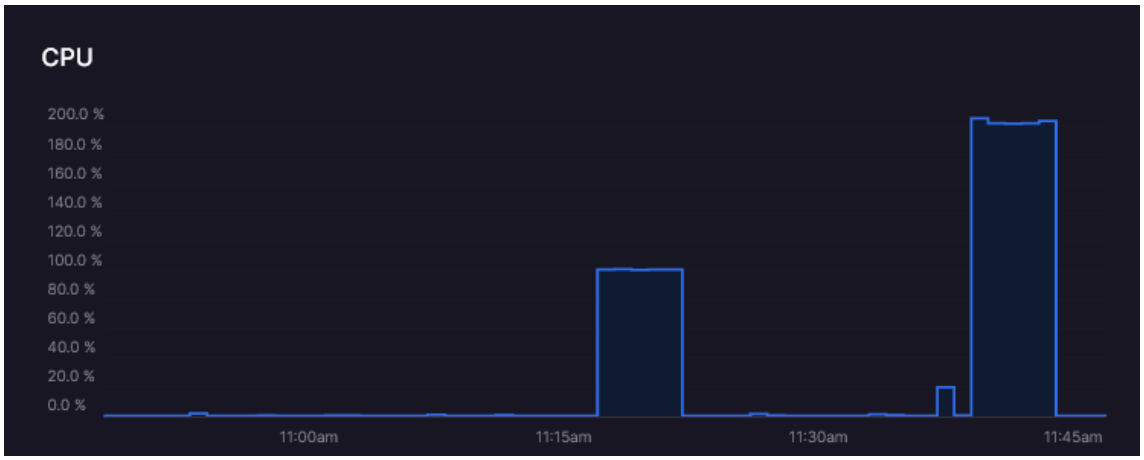


Fig. 78: Estado 3 del *CPU*

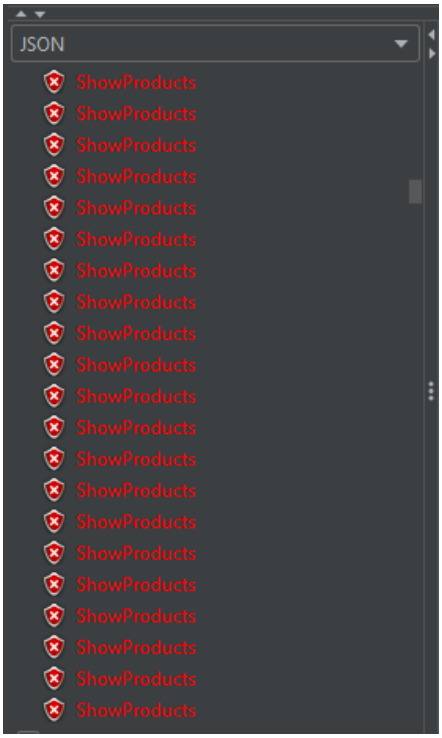


Fig. 79: Datos de respuesta de 1000 solicitudes

Casos de uso

A continuación, se muestran cada uno de los casos de uso que anteriormente se mencionó, en la **Elaboración de diagrama de casos de uso**, por lo que los siguientes diagramas son para los 3 roles existentes en el sistema; Administrador, Propietario y Cliente. **Fig. 80**, **Fig. 81**, **Fig. 82.**

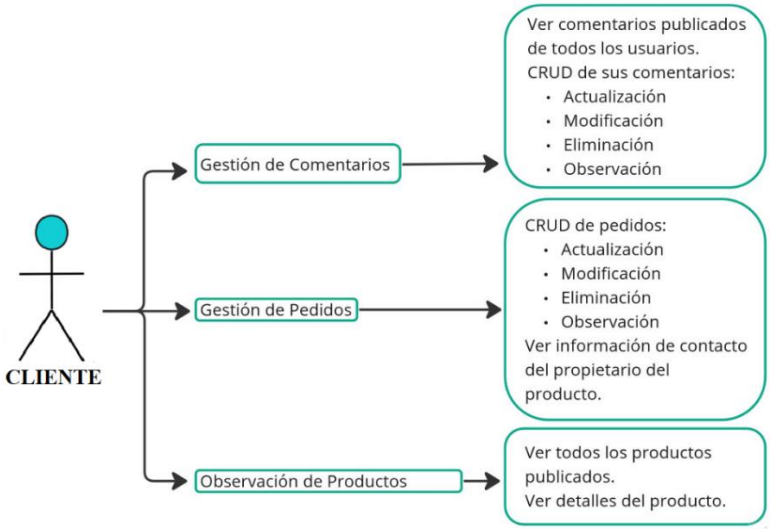


Fig. 80: Diagrama Caso de uso – Rol Cliente

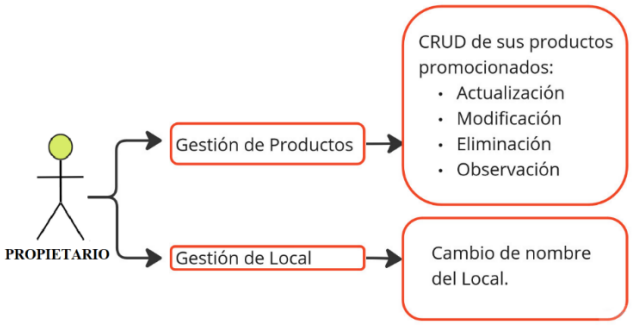


Fig. 81: Diagrama Caso de uso – Rol Propietario

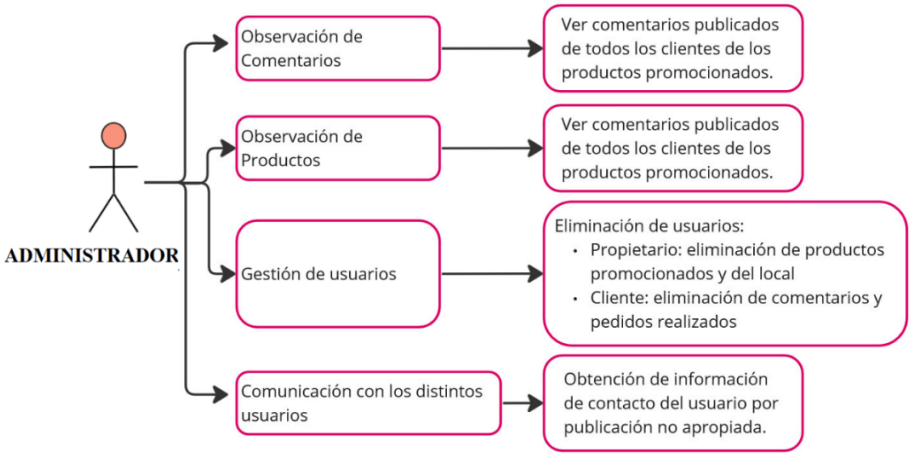


Fig. 82: Diagrama Caso de uso - Rol Administrador

Sprint Backlog

TABLA XXXI: Elaboración Sprint Backlog

Elaboración *Sprint BackLog*					
ID-SB	**Nombre**	**ID-HU**	**Historia de Usuario**	**Tarea**	**Tiempo estimado**
SB-000	Configuración del entorno de desarrollo	N/A	N/A	Puesta en marcha del proyecto en *Visual Studio Code*. Elaboración de la base de datos MySQL. Creación de flujo del sistema (diagrama).	20H
SB-001	Módulo de conexión entre base de base de datos y el ORM Eloquent de Laravel.	N/A	N/A	Migraciones, *Seeders* y *Factories*, Modelos Relaciones a nivel de base de datos y a nivel de Eloquent	30H
SB-002	Módulos de autenticación y registro usuarios.	HU001 HU002 HU009 HU-010	Ingreso al sistema. CRUD de todos los usuarios.	Inicio de sesión para los distintos roles de usuario. Registro de nuevos usuarios. Restablecimiento y actualización de información personal.	40H
SB-003	Módulos *Endpoints* de Usuarios, Productos, Pedidos, Negocios, Propietarios, Clientes, Roles, Comentarios.	HU-003 HU-005 HU-006 HU-007 HU-008 HU-011 HU-012 HU-013 HU-016 HU-017	Observación y eliminación de publicaciones. Sección calificación/comentarios. Gestión de productos del negocio. Facilitación de stock disponible de productos. Lista productos comprados. Productos ofertados por un negocio. Observación de publicaciones de los productos.	Creación controladores *API* Codificación de métodos: Controladores *API*. Creación rutas *API*. Modelos Propietario, Usuario, Productos, Negocios, Pedidos, Roles, Comentarios.	70H

		HU-018	Detalle de compra. Lista de comentarios. CRUD comentarios.		
SB-004	Módulo de *Gates*.	HU-014 HU-015	Solicitar pedido. Modificación de pedido	Elaboración de Diagrama de casos de uso. Codificación de "*Gates*" de los usuarios: Administrador, Propietario y Cliente. Codificación de controladores orientado al acceso de los datos por parte de ciertos usuarios	30H
SB-005	Módulo correo electrónico, comunicaciones	HU-004 HU-016	Comunicación con los usuarios. Comunicación con el dueño del producto.	Módulo de comunicaciones	20H

SB-006	Módulos de Pruebas	N/A	N/A	Pruebas unitarias a rutas. Validación de llamadas al servidor en base a la *API* por media de una aplicación cliente. Análisis de respuestas por cada componente que efectúe consultas del sistema.	20H
SB-007	Despliegue del proyecto de *API REST* en un servidor web público.	N/A	N/A	Configuración del servidor *web*. Validación del rendimiento del sistema en el servidor. Revisión de la seguridad del sistema. Prueba del consumo de la *API* por parte de otro sistema.	10H
TOTAL					**240H**

ANEXO III

El manual de usuario de este proyecto el cual posee la funcionalidad de la elaboración de este componente *Backend* se encuentra en YouTube.

URL del vídeo de funcionalidad:
https://www.youtube.com/watch?v=g2A9kMmOqKo&t=2s

ANEXO IV

El manual de instalación se encuentra en la URL del repositorio de GitHub, en donde está alojado todo el código fuente del proyecto que ha sido desarrollado, luego dirigirse en el README, allí se encuentra la descripción correspondiente.

La rama en que se encuentra el proyecto en base a los Sprint realizados y culminados se encuentra en la rama Sprint6. El enlace del despliegue del proyecto con la herramienta de *Postman* se adjunta a continuación, donde se deberá agregar al final de la URL */api/NombreDeRuta.*

Ejemplo: https://ecommerce-mamolbin-c632.up.railway.app/api/showproducts

Finalmente, como se mencionó en el apartado de Creación de Seeders y Factories, la contraseña de ingreso al sistema para el rol administrador, propietario y cliente será password. Mientras que, para el correo electrónico de ingreso al sistema del cliente y propietario puede ser cualquiera que se encuentre en la base de datos.

Enlace del repositorio de GitHub-Sprint6:
https://github.com/MarlonAdrian/Ecommerce-Backend/tree/Sprint6

Enlace del proyecto desplegado a producción:
https://ecommerce-mamolbin-c632.up.railway.app/

Email y contraseña para el ingreso al sistema:

- Administrador (único usuario de la base de datos)
  Email: connelly.junior@example.net
  Contraseña: password
- Propietario (aleatorio de la base de datos)
  Email: torp.lydia@example.org
  Contraseña: password
- Cliente (usuario aleatorio de la base de datos)
  Email: cjohns@example.net
  Contraseña: password