ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE SISTEMA WEB Y APLICACIÓN MÓVIL DE VENTA DE ROPA DE SEGUNDA MANO

DESARROLLO DEL BACKEND

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

ROBERTH FABIAN PINCHA HUANCA

roberth.pincha@epn.edu.ec

DIRECTOR: DR. RICHARD PAUL RIVERA GUEVARA

DMQ, septiembre 2022

CERTIFICACIONES

Yo, ROBERTH FABIAN PINCHA HUANCA 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.

ROBERTH FABIAN PINCHA HUANCA

roberth.pincha@epn.edu.ec

roberthlink@yahoo.es

Certifico que el presente trabajo de integración curricular fue desarrollado por ROBERTH FABIAN PINCHA HUANCA, bajo mi supervisión.

PhD. RICHARD PAUL RIVERA GUEVARA
DIRECTOR

richard.rivera01@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el producto resultante 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.

ROBERTH FABIAN PINCHA HUANCA

DEDICATORIA

Quiero dedicar el desarrollo del componente a mi familia, profesores y amigos que me han ayudado y brindado su apoyo durante el transcurso de la carrera.

ROBERTH FABIAN PINCHA HUANCA

AGRADECIMIENTO

Agradezco mucho haber tenido la oportunidad de estudiar en la Escuela Politécnica Nacional, porque he podido disfrutar y aprender mucho. Voy a llevarme muchos recuerdos de mi estancia aquí, porque he podido aprender y poner en práctica muchas cosas que mis maestros y compañeros han compartido conmigo.

Agradezco mucho a mis padres, a mi hermana y a mi familia, por siempre estar a mi lado, y siempre darme su apoyo. Cada día con ellos son los mejores, y es por eso que los quiero un montón por que aun en los momentos difíciles, ellos siempre se mantienen firmes y demuestran que pueden superar todo, es por eso que también los admiro y es lo que me ha motivado a ser quien soy.

A mis maestros, les agradezco el tiempo, el esfuerzo, la paciencia y los conocimientos que han compartido conmigo y los demás; guardare y atesorare cada consejo que me han brindado. Todas las clases siempre han sido interesantes, cada practica o proyecto ha sido un gran desafío y todo por querer sacar lo mejor de nosotros y poder así enfrentarnos al ambiente laboral, y es por eso que les agradezco cada momento.

ROBERTH FABIAN PINCHA HUANCA

ÍNDICE DE CONTENIDO

RESUMEN

Desde la aparición del COVID-19, el mundo ha cambiado totalmente puesto que la enfermedad ha sido el mayor problema para la sociedad en los últimos años. Es así como muchos negocios tuvieron que cerrar y algunos otros tuvieron que adaptarse a las medidas de bioseguridad, esto provocó una reducción en el aforo, y el comercio se sintió afectado totalmente en el país. A medida que esto sucedía muchos de los negocios optaron por hacer uso de diferentes medios, y es así que el uso de plataformas digitales permitió el avance de estos negocios, al ofrecer sus servicios de una manera más segura y agilizando la compra. Por lo que implementar una tienda virtual ayudaría a manejar el comercio de una manera más segura para la compra y venta de artículos.

De esta forma, se ha desarrollado un componente de *Backend* que implementa la tecnología *API REST* la cual proporciona un acceso, obtención, edición y retorno de la información que es compartida con alguna aplicación cliente para que pueda beneficiarse al implantar los métodos y medidas de seguridad necesarias a través de consultas robustas, y así agilizar el trabajo de este tipo de aplicaciones de *Frontend*.

El proyecto está realizado a través del uso de diferente tecnologías y técnicas de desarrollo, las cuales otorgan las guías y herramientas necesarias para producir un software de calidad. El proyecto se ha realizado en conjunto al *framework* de *Laravel,* el cual ofrece la interacción y recursos necesarios para generar la lógica de la aplicación y establecer la comunicación con la base de datos de tipo SQL. *Scrum* es usado en el desarrollo de este proyecto puesto que sus técnicas para gestionar el desarrollo de software ofrecen la flexibilidad y adaptación a los cambios, lo cual fomenta la productividad y genera valor al desarrollar un producto de software de manera responsable y cumpliendo con la agenda establecida.

En el presente documento se expone el desarrollo del componente *Backend*, donde los procedimientos para generar la *API REST* que en conjunto a los resultados obtenidos son parte de la explicación del desarrollo de este proyecto.

PALABRAS CLAVE: *Backend, Frontend, API REST, Laravel, Scrum*.

ABSTRACT

Since the appearance of COVID-19, the world has completely changed since the disease has been the biggest problem for society in recent years. Thus, many businesses had to close and some others had to adapt to biosecurity measures, this rejected a reduction in capacity, and trade felt totally affected in the country. As this happened, many of the businesses chose to make use of different means, and it is thus that the use of digital platforms allowed these businesses to advance, by offering their services in a safer way and speeding up the purchase. Therefore, implementing a virtual store would help manage commerce in a safer way for the purchase and sale of items.

In this way, a *Backend* component has been developed that implements the *API REST* technology which provides access, obtaining, editing and return of the information that is shared with a client application so that it can deteriorate when implementing the security methods and measures. needs through robust queries, and thus speed up the work of this type of *Frontend* applications.

The project is carried out using different technologies and development techniques, which provide the necessary guides and tools to produce quality software. The project has been carried out in conjunction with the *Laravel framework*, which offers the necessary interaction and resources to generate the application logic and establish communication with the SQL-type database. *Scrum* is used in the development of this project since its techniques to manage software development offer flexibility and adaptation to changes, which promotes productivity and generates value by developing a software product responsibly and complying with the agenda. established.

In this document, the development of the *Backend* component is exposed, where the procedures to generate the *API REST* that together with the results obtained are part of the explanation of the development of this project.

KEYWORDS: *Backend, Frontend, API REST, Laravel*, Scrum.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

La tendencia de las compras en plataformas E-commerce no reaccionó diferente al resto de países en el mundo. El comercio electrónico en Ecuador alcanzó un volumen de negocio de USD 2.3 mil millones, lo que supone un crecimiento de USD 700 millones (43,75 %) frente al 2019. En el año 2021 se estima un crecimiento no menor a un doble dígito. El año 2020 marcó un antes y un después en el comercio online [1].

Según La Cámara Ecuatoriana de Comercio Electrónico las compras por canales digitales o sitios web, han incrementado al menos en 15 veces desde el inicio del distanciamiento social. Estas cifras indican que pandemia ha impulsado a ese 34% los usuarios que usaban plataformas digitales como medios de compra secundario, a usar estas plataformas constantemente. [2]

En el caso de las personas que usaban esta plataforma rara vez, o nunca ahora también son usuarios de este medio para evitar salir de casa. La pandemia superó uno de los desafíos dentro de la industria en Ecuador, la adopción de este método de compra. Este incremento demuestra que los ecuatorianos tienen la apertura a usar estas plataformas, y la pandemia puede ser el impulso necesario para que esta industria crezca en el país. Es importante tomar esta oportunidad y manejar el resto de los desafíos como la seguridad, el manejo y protección de datos, y la experiencia del consumidor, para que, al terminar esta emergencia, el uso de estos medios continúe. [3] Con el crecimiento del E-commerce, también se han desarrollado los métodos de pago digitales. Hay varias maneras de realizar pagos en línea, algunos negocios usan pasarelas de pago y otros procesadores de pago.

Esto, solo ha crecido debido a la crisis sanitaria del COVID-19. Las restricciones que incluyen el distanciamiento social y el cierre total o parcial de varias industrias y negocios han causado un incremento en la actividad comercial en línea. También, debido a la crisis existe el miedo de contagio al exponerse en un lugar público, incluyendo tiendas y supermercados. Esto ha llevado a que los usuarios que realizaban sus compras en persona empiecen a usar las plataformas digitales para ordenar alimentos y víveres, medicinas, artículos de limpieza, entre otros necesarios. Frente a esta situación, ante la cual nadie tiene un conocimiento claro del impacto o duración, nos hemos ido adaptado a las medidas impuestas por el gobierno, y a hacer uso de las alternativas como el E-commerce, para no exponernos.

En la ciudad de Quito, desde principios de marzo del año 2020 se ha venido trabajando en medidas de bioseguridad, pero cuando el problema se fue agravando junto a las resoluciones del COE Nacional [4], se ha dejado a un lado a los negocios, mercados y las tiendas que fueron las más afectadas y con el pasar del tiempo limitaría las oportunidades por el hecho de implementar las nuevas medidas. Los mercados de pulgas pertenecen al sector comercial, donde se puede comercializar todo tipo de artículos y donde la mayor parte de gente acude por las facilidades de adquisición. El mercado de San Roque es un ejemplo claro de comercialización de productos de segunda mano y donde alberga una gran cantidad de comerciantes [5]. Con la llegada del virus este mercado tuvo que cerrar sus puertas por un tiempo para poder ajustarse a las restricciones de bioseguridad y así poder albergar a un número limitado de personas. Los beneficios tras esta nueva modalidad no son más que ayudar a la salud pública [6], sin embargo, existe la perdida de trabajo para algunos comerciantes por el espacio o la poca venta. Es por esta razón muchos de estos negocios necesitan una alternativa diferente para la atención y que promuevan una oportunidad nuevamente a los comerciantes sin el riesgo de contagio.

Ante una problemática de tal magnitud y la necesidad de empleo para la comercialización de productos de segunda mano, se plantea el desarrollo de una aplicación web que permita la comercialización sin la necesidad de salir de casa [7]. El desarrollo de esta aplicación ayudara a muchos mercados de pulgas, comerciantes o cualquier persona a tener un espacio óptimo para ofrecer sus artículos de manera ágil y sin gastos secundarios (alquiler del espacio del mercado, carpas, el auto donde se movilizan, entre otros) con información actualizada que ayudara al sector comercial a mejorar el eje de negocios. Por ello, se implementará la tecnología *API* que permitirá manejar la información de las prendas de vestir, así como también almacenar nueva información y poder otorgar el acceso a esta información a otras aplicaciones clientes.

1.1 Objetivo general

Desarrollar una interfaz de programación de aplicaciones (*API REST)* para automatizar el sistema de venta de segunda mano.

1.2 Objetivos específicos

1. Analizar los requerimientos para el sistema.
2. Diseñar la arquitectura y base de datos del sistema.
3. Programar los módulos y tecnología *API REST* que permita el acceso de su información en otras aplicaciones clientes.
4. Probar la tecnología *API REST*.
5. Desplegar a producción el *Backend* en un servidor web público.

1.3 Alcance

El uso de aplicaciones web dentro del comercio durante los acontecimientos de la pandemia, han resultado ser un pilar importante para resguardar la salud y fomentar una relación social segura sin salir de casa. Esto genera bastante beneficio como la comodidad, la relación social, la optimización del tiempo y asegurarse de evitar el contagio del Covid-19.

El alcance del proyecto busca implementar la tecnología *API REST* con el propósito de otorga un alcance y base de datos más simple a otros sistemas. La aplicación está enfocada en los usuarios y comerciantes, donde a través de *API REST* podrán realizar peticiones para visualizar, guardar, editar o eliminar los datos que se encuentran almacenados en las tablas de la base de datos, la propuesta también garantiza la integridad, armonía y seguridad de las rutas para mantener el orden y acceso a los datos.

El usuario con rol Administrador en la *API REST* puede:

- Visualizar las ultimas publicaciones.
- Visualizar los usuarios registrados.
- Registrar nuevas categorías.
- Registrar nuevas etiquetas.
- Dar de baja a los perfiles o publicaciones que no cumplan con los estándares establecidos por la aplicación.

El usuario con rol Comprador/Vendedor en la *API REST* puede:

- Acceder al inicio de sesión.
- Acceder al registro por medio de un formulario.
- Acceder a los datos de su usuario o el de otros.
- Visualizar los productos publicados en orden de la última publicación.
- Acceder a la información de un producto.
- Listar los productos por categoría o etiquetas.
- Subir una publicación.
- Asignar la categoría o etiquetas a sus publicaciones.
- Guardar las publicaciones en el carrito de compras.
- Guardar las publicaciones en un listado de compras.

1.4 Marco Teórico

Metodología

Al momento de desarrollar un software de calidad, este puede seguir dos distintos marcos de trabajo donde se estructura, planifica y controla el proceso de desarrollo. Las metodologías se definen por elementos, los cuales son las fases, métodos, técnicas, herramientas, documentación, control y evaluación donde en cada actividad por realizar debe seguir un estándar propio de estos elementos, y de esta forma mantener el control de cada actividad realizada a través de una comunicación con el equipo de desarrollo [8]. Las metodologías pueden ser tradicionales o agiles que dependen de la naturaleza del negocio y sus requerimientos, porque la primera puede enfocarse en el desarrollo secuencial mientras que la otra puede ser flexible.

Metodología ágil

La metodología ágil tuvo su origen en el manifiesto del año 2001 que estaba enfocado al desarrollo de software de manera ágil y flexible, donde se mostraron los principios que se deben tomar en cuenta para la construcción de un proyecto y de esta forma satisfacer las necesidades de los clientes y fortalecer los equipos de trabajo [9]. En el manifiesto ágil se establecieron doce principios siguiendo las premisas principales y de las cuales las metodologías han establecido dentro de su marco de trabajo.

Los ideales que siguen estas metodologías son:

- Priorización de individuos e iteraciones del equipo sobre los procesos y herramientas.
- Software funcional sobre documentación.
- Colaboración con el cliente sobre negociación contractual.
- Flexibilidad ante el cambio sobre el seguimiento del plan.

Las metodologías agiles otorgan a las organizaciones o equipos de desarrollo la flexibilidad para integrar en sus marcos de trabajo, principios y practicas útiles para el crecimiento sostenible de un sistema y que también es base de los valores, cultura y los tipos de sistemas que desarrolla la organización.

Backend

La tecnología de *Backend* es muy crítica dentro del desarrollo de software, por ser la más importante puesto que se encarga del funcionamiento de la lógica del sistema. En él se desarrolla un sin número de funciones en la cual trabajan para el desempeño del sistema y procesos que se ejecutan para la comunicación con el servidor y el *Frontend*.

El desarrollo del *Backend* es de mucha importancia porque es vital para el *Frontend* y sin duda enlazar estas dos tecnologías requiere de un proyecto simétrico, donde se pueda transmitir abiertamente la lógica puesta el desarrollo de *Backend* es más racional, lógico en la parte de funcionalidad [10]. Las características de la gestión de un *Backend* se resumen en las siguientes: la simplificación de procesos, acciones Lógicas, la conexión a la base de datos, uso de librerías y pruebas sobre el control de calidad.

Base de datos relacional

Las bases de datos han sido parte de nosotros desde tiempos inmemorables en la cual hemos organizado nuestra información de forma diaria, y esto ha sido un precedente para entender el comportamiento o predecir situaciones. Las bases de datos son grandes cantidades de información registrada que almacena datos importantes y que se maneja a través de registros en los cuales se puede ingresar, buscar, actualizar o eliminar la información, pero dependiendo de la interrelación de esta información evitaremos la duplicidad y así mejorando la organización [11].

Una base de datos relacional es un tipo de almacenamiento de datos que proporciona acceso a secciones de datos relacionados entre sí. Este tipo de base de datos se maneja a través del modelo relacional, que representa muy bien la organización de datos en tablas que están definidas por filas y columnas las cuales los registros se identifican con un ID único y sus atributos.

API REST

La tecnología *API REST* es un tipo de arquitectura para el desarrollo web que utiliza el estándar HTTP, que permite realizar peticiones con cualquier dispositivo cliente y siendo la parte más fundamental para la comunicación entre el *Backend* y *Frontend*. En el desarrollo de una *API REST* conocer el significado de HTTP es fundamental para poder enfrentar con regularidad los métodos permitidos, códigos de estado y aceptación de contenido.

Esta tecnología se maneja a través de métodos que manipulan los diferentes recursos que son conformados por el *Backend*. Los métodos más utilizados para la gestión de datos e información son *POST*, *PUT*, *GET*, *DELETE* y *PATCH*; estos métodos son utilizados para manejar los registros de la base de datos y que se manejan a través del dominio [12], que nos otorga una interfaz uniforma para la transferencia de datos en el sistema de *API*.

2 METODOLOGÍA

En la creación de un Software, determinar el desarrollo de una aplicación debe empezar con una seria de tareas, empezando con una reunión para establecer lineamientos y orden en el trabajo, por ejemplo: determinar los objetivos y elegir la metodología de desarrollo de software. Posteriormente, se hace un análisis de los requerimientos para establecer concretamente algunos elementos importantes para la aplicación. El diseño del sistema sería el siguiente paso, donde el desarrollo del prototipado ayuda la creación de un modelo para nuestros *stackholders*. Luego de su aprobación podremos diseñar la arquitectura del sistema y modelo de base datos que serán piezas claves para el desarrollo del *Backend*. Una vez terminado ese paso se establece la integración donde se realizará la conexión entre el *Frontend* y *Backend* con todas sus funcionalidades. Finalmente se realizan pruebas para comprobar las funcionalidades del sistema y que todas esas funciones actúen como es debido.

2.1 Metodología de Desarrollo

La metodología ágil Scrum es un conjunto de buenas practica para trabajar colaborativamente, en equipo y obtener el mejor resultado en un proyecto que está orientada a la creación de software en el que los requerimientos son cambiantes o están poco definidos, donde la flexibilidad, productividad, innovación y la competitividad son fundamentales. Por ello, usar la metodología Scrum nos otorga un marco de gestión de proyectos incremental en la que los trabajos se recibe a manera de iteraciones entre tiempo cortos que suceden dentro de dos o cuatro semanas como mínimo que establecen el desarrollo sostenible de un software [13].

Roles

Scrum: define roles y responsabilidades del equipo permitiendo el éxito del proyecto. El equipo para el presente proyecto está formado por:

Product Owner: es la persona responsable de comunicar la visión del producto y las características que debe poseer, siendo el medio de representación de necesidades y comentarios de las partes interesadas en el producto para generar incrementos más valorizados en cada *Sprint* [14], Este rol lo desempeña el Dr. Richard Rivera puesto que su gran conocimiento y experiencia que puede compartir con el grupo de desarrollo.

Scrum Master: es la persona que dirige al equipo de desarrollo, garantiza el avance y cumplimiento de los objetivos establecidos, acatando los procesos de la metodología [14], este rol lo cumple el Dr. Richard Rivera como director del proyecto.

Development Team: son los profesionales encargados del cumplimiento de cada *Sprint*, con capacidad de autogestión del trabajo y cumplimiento de los objetivos de cada iteración [14].

En la **TABLA I** se presenta las asignaciones del equipo Scrum para el presente proyecto.

TABLA I: Equipo Scrum

Rol	Nombre
Product Owner	Dr. Richard Rivera
Scrum Master	Dr. Richard Rivera
Development Team	Sr. Luis Jacome Sr. Roberth Pincha

Artefactos

La metodología Scrum al ser aplicada esta genera elementos que pueden ser llamados Artefactos, que están diseñados para organizar adecuadamente el registro de la información y que a su vez garantiza la transparencia del desarrollo.

Recopilación de Requerimientos

En el desarrollo de software al usar alguna metodología ágil, es importante tener un registro de información como entrevistas y reuniones, en la cual pueda generar los requerimientos del sistema. Este es un elemento importante para el éxito de un proyecto porque al realizar el análisis correspondiente este puede otorgar las tareas necesarias que satisfagan las necesidades del cliente. La lista de los requerimientos obtenidos para el proyecto se presenta dentro de la sección Recopilación de Requerimientos dentro del **ANEXO II.**

Historias de Usuario

Son una serie de tarjetas que plasman de forma detallada las funcionalidades que el sistema posee en función de las necesidades de nuestros clientes. A continuación, la **TABLA II** muestra un ejemplo de las historias de usuario que se han utilizado. Las historias de usuario restantes pueden observarse en el **ANEXO II**.

TABLA II: Ejemplo Historia de Usuario 1 –Crear modelo y tablas de la BD

HISTORIA DE USUARIO	
Identificador (ID): HU001	**Usuario:** Administrador
Nombre Historia: Autenticar usuarios	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Iteración Asignada: 1	
Responsable: Roberth Pincha	
Descripción: Todos los usuarios finales registrados, deben iniciar sesión colocando su email/nombre de usuario y contraseña para de esta manera ingresar al sistema e interactuar en este, de igual forma pueden cerrar sesión cuando lo deseen.	
Observación: El inicio de sesión constará con validaciones que se visualizarán en el caso que se ingresen credenciales inválidas.	

Product Backlog

Es un listado de todas las tareas que se quieren realizar a partir de la previa priorización de los requerimientos durante el desarrollo del proyecto. Este listado se va actualizando mientras se va realizando el producto, con el propósito de ayudar a tener un mejor control y manejo del estado de las actividades y cambios durante el desarrollo de este.

El listado muestra todos los requerimientos funcionales con descripciones breves que fueron definidas anteriormente para el desarrollo del sistema web, y que están ordenadas según la prioridad que tienen en la complejidad para el desarrollo y en el negocio. El detalle del *Product Backlog* se encuentra en el **ANEXO II.**

Sprint Backlog

Es un listado de tareas que han sido seleccionadas previamente del *Product Backlog*, los que se desarrollan día a día en diferentes *Sprint's* del desarrollo del proyecto. En el listado el equipo de desarrollo deberá priorizar las funcionalidades que serán entregadas en cada *Sprint*.

El *Sprint Backlog* puede ser representado mediante un tablero de tareas, para que de esa forma todo el equipo conozca que actividades tienen asignadas**.** El *Sprint Backlog* definido para el desarrollo de este proyecto se encuentra en el **ANEXO II.**

2.2 Diseño de la arquitectura

El patón de arquitectura para el desarrollo de software es fundamental puesto que al crear un software se debe planificar y conocer totalmente los requerimientos del producto, por ende, utilizar un esquema organizativo estructural es la mejor solución para llevar a cabo de manera eficaz los aportes más significativos para dar un valor muy alto al producto final. A continuación, se describe el modelo arquitectónico usado para el desarrollo del sistema web.

Arquitectura de Datos

La implementación de *Laravel Eloquent*, es parte de las herramientas que integra el proyecto para el establecimiento de una fácil interacción con la base de datos, que a través de funcionalidades importantes como la inserción, actualización y eliminación de registros por medio de comandos propios de la herramienta [15] la vuelven única. Integra funciones que ayudan a realizar complejas consultas y la definición de componentes denominados modelos, los cuales son parte fundamental para comunicarse directamente con la base de datos.

En la **Fig. 1** se muestra el patrón arquitectónico y las herramientas que intervienen para establecer el esquema que se usa en el proyecto propuesto a través del flujo Modelo-Vista-Controlador.

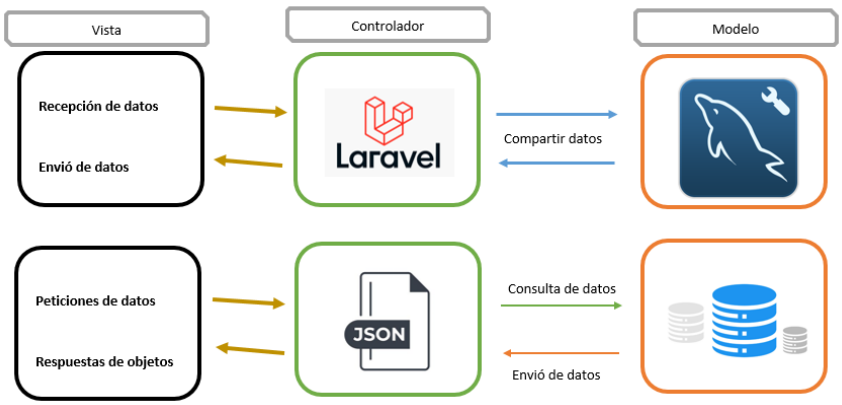


Fig. 1: Patrón arquitectónico de la *API REST*.

Patrón arquitectónico

En el desarrollo del sistema web hacer uso del patrón arquitectónico Modelo Vista Controlador (MVC), permite el manejo de una correcta organización y esquematización de los elementos de un componente de software, lo que beneficia bastante a la hora de gestionar las relaciones entre los elementos. En este modelo se describen las capas que maneja este paradigma de software para el diseño de la arquitectura del proyecto.

Modelo: Es la capa donde se trabaja en la definición de las propiedades más importantes en la cual se establecen los mecanismos de relación y gestión de datos [16].

Vista: Es la parte visual del sistema que también es denominado interfaz de usuario, y en ella se puede interactuar, visualizar y realizar acciones para el tratamiento de los datos [16].

Controlador: Es la capa de unión entre la vista y el modelo, y en la cual se trabaja con la lógica de la aplicación donde responde con peticiones para el establecimiento del flujo interacción de la vista y el modelo [16].

2.3 Herramientas de desarrollo

La **TABLA III** expone las herramientas de desarrollo junto a su definición y breve justificación para definir el alcance y el desarrollo del componente *Backend*.

TABLA III Herramientas y Librerías para el desarrollo del *Backend*.

Herramienta	Descripción	Justificación
Laravel	*Laravel* es un *framework* PHP de código abierto para el desarrollo de aplicaciones web. Posee una sintaxis fácil de entender y usar que junto al uso de un sistema de mapeo relacional y un sistema de plantillas definen el desarrollo corto y eficiente de los proyectos web [15].	El diseño está bajo el patrón MVC que separa una aplicación en tres capas, lo que permite proporcionar adaptación y desarrollo en toda la lógica de la aplicación de manera clara y organizada.
Composer	Composer es un sistema de gestión de dependencias de PHP que proporciona un estándar para administrar librerías que son necesarias para integrar una gran variedad de funcionalidad y complementos para el desarrollo de *API's* [17].	Recursos de *Laravel* como Composer beneficia en gran manera en la implementación, ejecución e integración de dependencias y librerías actualizadas.

XAMPP	Es una distribución de Apache que incluye un conjunto de varios softwares libres como Apache, Mysql, PHP y Perl [18].	La utilización de XAMPP en el desarrollo de aplicaciones web es de vital importancia, puesto que integra los recursos necesarios como la implementación de un servidor web, el lenguaje de programación y el sistema gestor de base de datos.
MySQL	Es un sistema gestor de base de datos relacional que implementa la gestión de tablas y que almacena la información de manera organizada [19].	Utilizar el gestor de datos MySQL ayuda a la administración de los registros generados en las diferentes tablas, y que permite conectarse al ambiente de desarrollo y establecer un canal en el cual se verifique la inserción de tablas, relaciones y datos.
Postman	Herramienta que integra la funcionalidad de realizar consultas a una *API REST* mediante peticiones HTTP, lo que permite realizar pruebas, simulaciones y monitoreos [20].	Al usar la herramienta de Postman se puede comprobar el funcionamiento de los diferentes *endpoints* generados, lo que permite una ayuda sobre la recuperación, envió y almacenamiento de datos a través de la tecnología *API*.
Heroku	Es una plataforma como servicio que se encarga del alojamiento de aplicaciones de software donde puede soportar lenguajes como Python o PHP. Heroku implementa contenedores virtuales que son los encargados de mantener y ejecutar las aplicaciones [21].	Heroku es una plataforma como servicio, que ayuda en el despliegue a producción del componente de *Backend* para la generación de *API REST*.
Git	Es un sistema avanzado de control de versiones que permite realizar capturas de los avances y trabajar paralelamente en el proyecto [22].	*Git* permite el control de versiones sobre un proyecto, a través de la generación de ramas que guardan los avances de un proyecto y que se pueden ir actualizando.
GitHub	*GitHub* es un servidor de alojamiento en línea que permite albergar elementos de un proyecto, que a través de GIT puede implementar los registros necesarios y que mejora la organización y control de versiones [22].	Usar la plataforma ayuda a organizar el conjunto de tareas planificadas, a través de ramas y que permite la gestión del proyecto para el desarrollo y despliegue a producción.

3 RESULTADOS

A continuación, se expone la realización de las actividades junto a los resultados conseguidos en cada desarrollo del *Sprint*.

Sprint 0. Configuración del ambiente de desarrollo

De acuerdo con lo establecido en el *Sprint Backlog*, el *Sprint* 0 presenta las actividades de acuerdo a la configuración del ambiente de desarrollo, lo cual es fundamental para desarrollar y tener compatibilidad de las herramientas y librerías. A continuación, las tareas que se establecieron en el *Sprint* 0 son:

- Recopilación de los requerimientos.
- Diseño de la base de datos.
- Instalación y actualización de herramientas y *frameworks*.
- Producir las entidades y relaciones a nivel de base de datos y *Eloquent* en el proyecto.
- Llenar con datos *Fakes* la base de datos.

Recopilación de los requerimientos.

El levantamiento de los Requerimientos de la *API REST* para el consumo de un sistema web, se realiza a través de un análisis y conversaciones con el cliente para determinar las funcionalidades de la aplicación. De esta forma, al ser establecido los requerimientos se puede identificar los actores, funcionalidades y requisitos importantes que manejará la aplicación y establecerá el ambiente adecuado para el cliente.

Diseño de la Base de datos.

Desarrollar la base de datos implica entender el alcance y modelo de negocio a realizar con seguridad. Por ello, es esencial identificar entidades, atributos y relaciones que posee la aplicación con el fin de cumplir las expectativas y los requerimientos propuestos por el cliente. Se presenta el modelo de la base de datos en la **Fig. 2**.

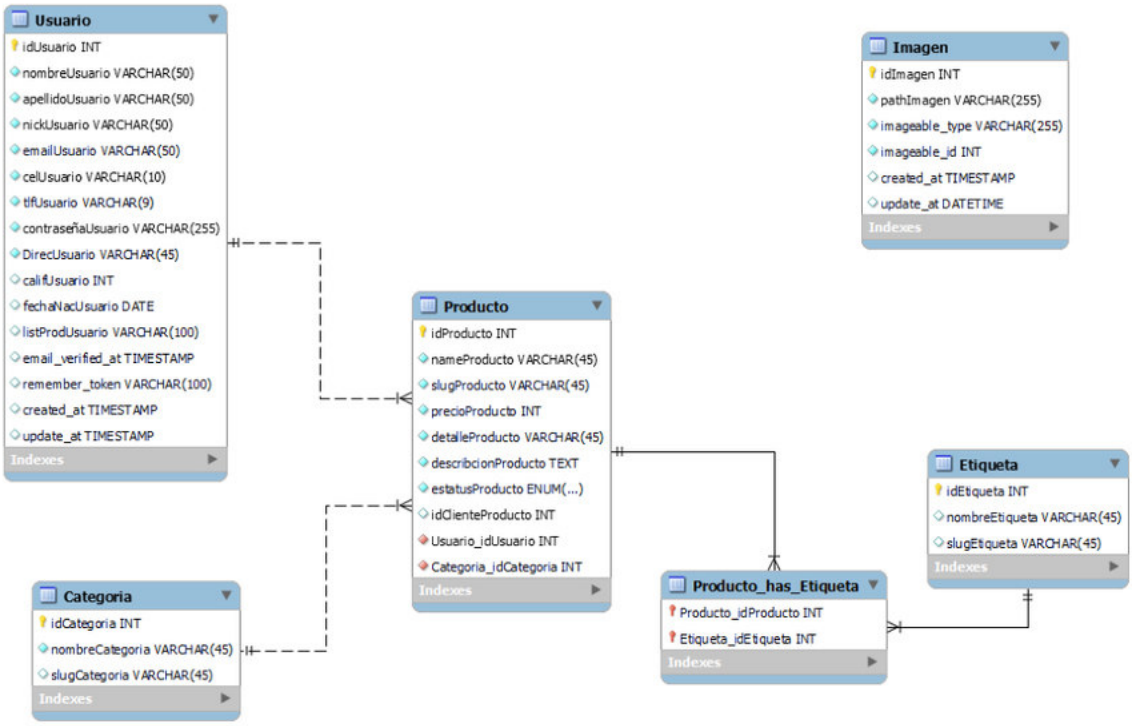


Fig. 2: Base de datos del sistema.

Instalación y actualización de herramientas y frameworks.

La configuración de la *API* y los componentes necesarios para el desarrollo del proyecto, se debe realizar la instalación de herramientas como Node.js, npm, PHP Storm, Composer, Postman y XAMMP. Son importantes para establecer la estructura del proyecto y el dominio necesario para establecer el control del *Backend*.

También, es necesario comprobar el versionamiento de las herramientas puesto que pueden no ser compatibles unas a otras, por ello se es necesario comprobarlas a través de una consulta en la interfaz de línea de comandos (CLI). En la **Fig. 3** se muestra las versiones instaladas en el equipo.

Además, los editores de código como Visual Studio Code necesita necesariamente extensiones que permitan agilizar el llamado de palabras claves y referencias con respecto a esas sentencias, en la **Fig. 4** se muestran las extensiones utilizadas.

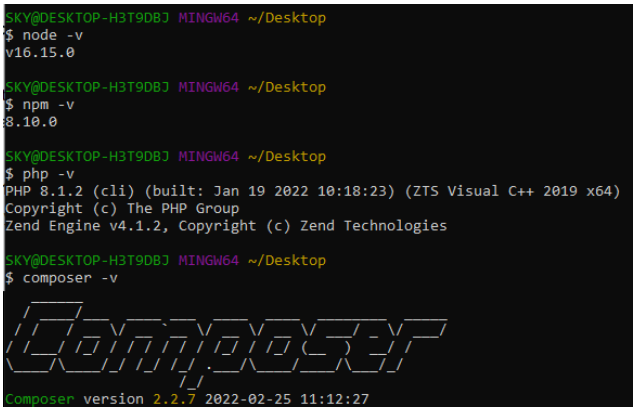


Fig. 3: Verificación de las herramientas instaladas

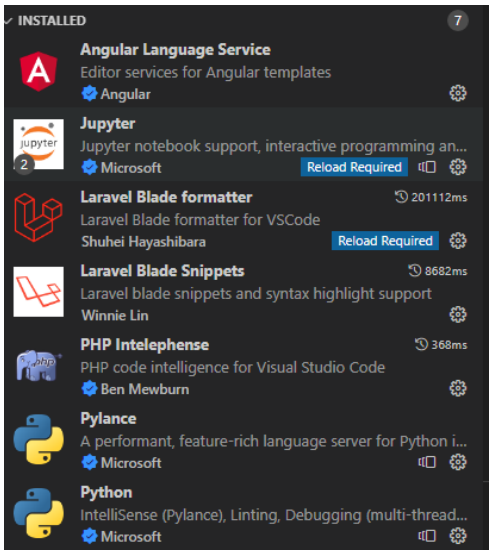


Fig. 4: Extensiones implementadas en Visual Estudio Code.

El desarrollo del componente *Backend* para el sistema, es desarrollado a través del *framework* de *Laravel*, que establece el conjunto de elementos necesarios, donde el patrón arquitectónico es de vital importancia para su implementación. A continuación, en la **Fig. 5** se muestra la estructura generada.

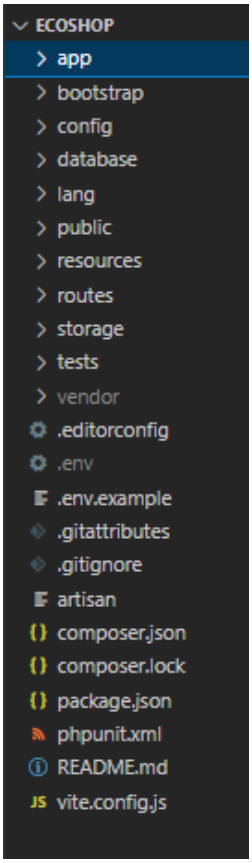


Fig. 5: Estructura del proyecto de *Laravel*.

Producir las entidades y relaciones a nivel de base de datos y *Eloquent* en el proyecto.

En función a la planificación y siguiendo el modelo de base de datos se establecen dentro del proyecto de *Laravel*, los requerimientos necesarios para la creación de las entidades y sus relaciones son a través de *Eloquent* y tablas. El desarrollo de este modelo primeramente *Laravel* trabaja siguiendo el modelo MVC, el cual establece que necesitamos de modelos los cuales son representaciones a las entidades y donde se pueden establecer las características, en la **Fig. 6** se muestra los atributos correspondientes al modelo de productos.

```php
public function up()
{
    Schema::create('products', function (Blueprint $table) {
        $table->id();
        $table->string('name')->unique();
        $table->string('slug')->unique();
        $table->string('details')->nullable();
        $table->integer('price');
        $table->text('description');
        $table -> enum('status', [0,1,2,3]) -> default(0);

        $table ->integer('idClient') -> nullable();

        $table->unsignedBigInteger('user_id');
        $table->unsignedBigInteger('category_id');

        $table->foreign('user_id')->references('id')->on('users')->onDelete('cascade');
        $table->foreign('category_id')->references('id')->on('categories')->onDelete('cascade');

        $table->timestamps();
    });
}
```

Fig. 6: Definir las características de una entidad a través de Tablas

A nivel de *Eloquent* se establen las relaciones dependiendo de los modelos que necesitan ser relacionados a través de opciones que integra *Laravel*. En la **Fig. 7** se establece la relación a nivel de *Eloquent* de la entidad usuario con la entidad productos e imágenes.

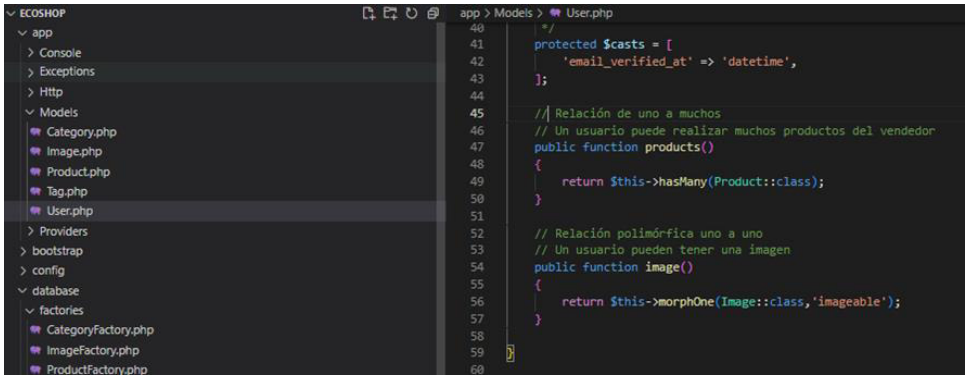


Fig. 7: Relación a nivel de *Eloquent* del modelo Usuario.

Llenar con datos Fakes la base de datos.

Las relaciones a nivel de *Eloquent* como a nivel de tablas, se puede comprobar las implementaciones al utilizar la generación de datos falsos. *Laravel* integra la opción de establecer estos registros, usando *Factories* y *Seeders*, que se acomodan a las relaciones predispuestas en el modelo de base de datos. En la **Fig. 8** y **Fig. 9** se establece el ejemplo de aplicación de un *Factory* y *Seeder* correspondientemente.

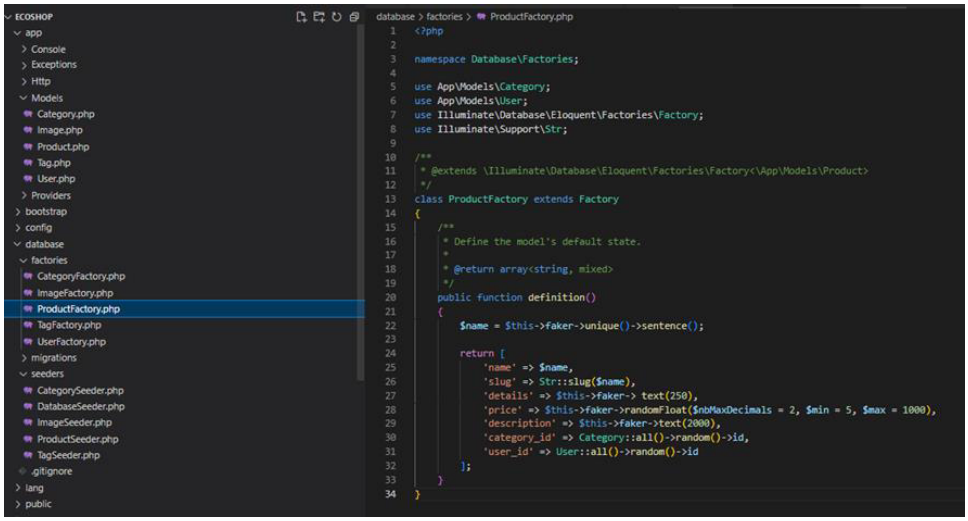


Fig. 8: Determinar los campos necesarios para realizar el registro de un Usuario con datos *Fakes*.

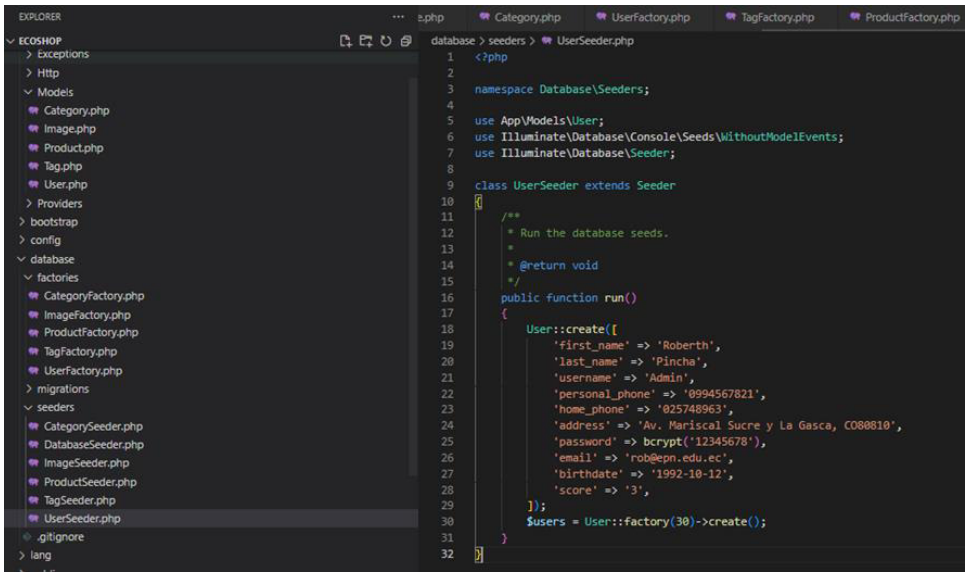


Fig. 9: Determinar el total de registros en base al *Factory*.

Al establecer las sentencias necesarias y campos a rellenar por *Laravel*, se puede posteriormente insertar los registros hacia un Administrador de base de datos como MySQL y comprobar que los atributos y relaciones estén correctamente establecidas. En la **Fig. 10** se muestran los registros generados por *Laravel*.

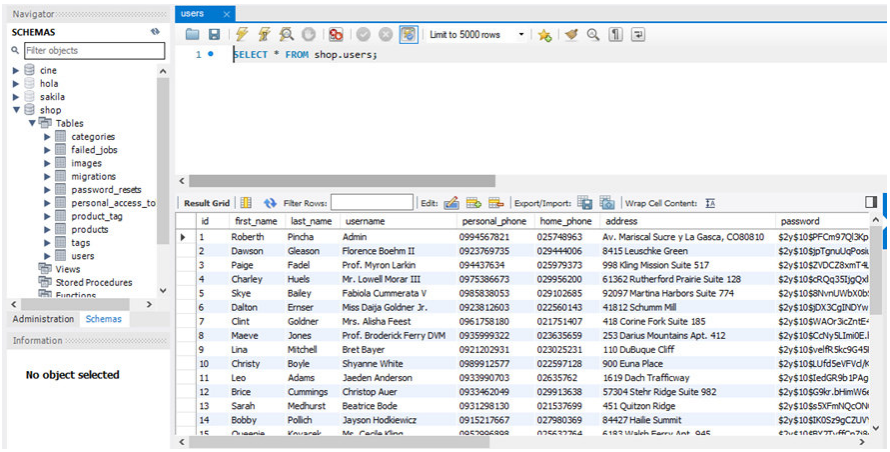


Fig. 10: Registros dentro de la base de datos MySQL.

Sprint 1. Diseño de la lógica del Usuario, Categorías, Etiquetas y Productos.

El *Sprint* 1 se ha establecido el desarrollo de la lógica de la funcionalidad del programa donde se implementarán los controladores, *traits*, modelos y rutas que caracterizarán a cada entidad para darles la funcionalidad determinada. A continuación, se presenta la planificación y los resultados obtenidos tras realizar las tareas establecidas dentro del *Sprint*.

- Autenticar Usuarios.
- Registrar Usuarios.
- Actualizar datos del usuario.
- Implementación del CRUD de las Categorías y Etiquetas.
- Implementación de la lógica de comercio electrónico que realizan los consumidores entre ellos (C2C)

Autenticar Usuarios.

En el desarrollo del Inicio de sesión de los usuarios, el proyecto puede contar con paquetes que implementan componentes y vistas necesarias con respecto a la autenticación de usuarios. El paquete de *Laravel* llamado Breeze integra los elementos y vistas necesarias para integrar la lógica de autenticación y registro. Las plantillas han sido mejoradas y pueden observarse dentro la sección Interfaces del **ANEXO II**.

Implementado el paquete de *Laravel* se hace uso de los nuevos componentes que se generaron automáticamente para el inicio de sesión, por lo cual a través de los controladores se trabaja para organizar las funciones del controlador en la creación y verificación de los datos del usuario, que a través de reglas se determina si el usuario pertenece al registro como se muestra en la **Fig. 11** y **Fig. 12**, las rutas de inicio de sesión serán parte de la implementación con las vistas necesarias como se muestran en la **Fig. 13.**

```php
public function store(LoginRequest $request)
{
    $request->authenticate();
    $request->session()->regenerate();
    return redirect()->intended(RouteServiceProvider::HOME);
}

public function destroy(Request $request)
{

    Auth::guard('web')->logout();
    $request->session()->invalidate();
    $request->session()->regenerateToken();
    return redirect('/');
}
```

Fig. 11: Controlador de Autenticación

```php
public function rules()
{
    return [
        'login_field' => ['required', 'string'],
        'password' => ['required', 'string'],
    ];
}

public function authenticate()
{
    $this->ensureIsNotRateLimited();

    $email_exist = Auth::attempt(['email' => $this->input('login_field'), 'pass

    $username_exist = Auth::attempt(['username' => $this->input('login_field'),

    if (!$email_exist && !$username_exist)
    {
        RateLimiter::hit($this->throttleKey());
        throw ValidationException::withMessages([
            'email' => __('auth.failed'),
        ]);
    }
```

Fig. 12: Reglas de autenticación.

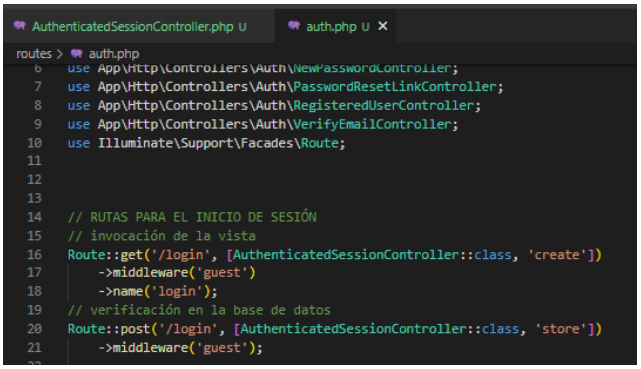


Fig. 13: Rutas de Inicio de sesión

Registrar Usuarios

En la **Fig. 14** se presenta la implementación de la Lógica de registro en la cual se solicita al usuario que, en un formulario ingrese un nombre y apellido para identificarse, y de igual manera un nombre de usuario que lo caracterizara, un correo valido y que no esté registrado previamente, también se solicita que ingrese información oportuna que pueda ayudar a los usuarios a contactarse con el vendedor tras una compra, y una contraseña con su confirmación.

Dentro de la lógica se establece que cada campo sea establecido correctamente, de esta forma se implementan las validaciones al momento de tratar de enviar el formulario. Una vez implementada la funcionalidad se procede a crear una ruta para el registro como se muestra en la **Fig. 15**.

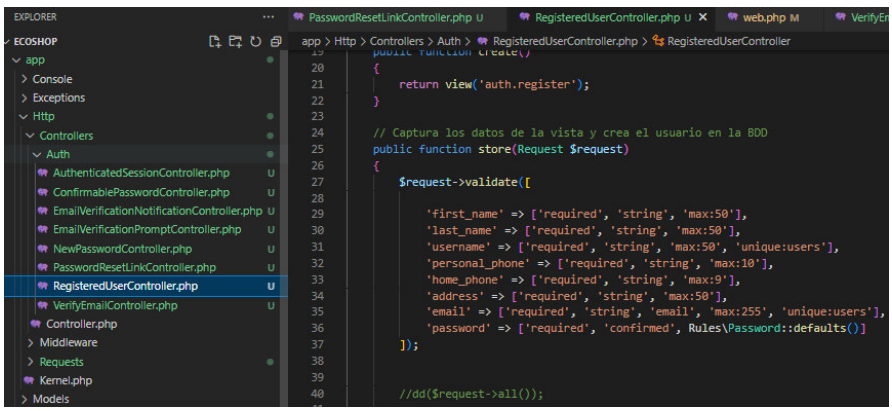


Fig. 14: Lógica y validaciones de los campos requeridos de la funcionalidad del Registro.

```php
// RUTAS PARA EL REGISTRO DEL USUARIO
// invocación de la vista
Route::get('/register', [RegisteredUserController::class, 'create'])
    ->middleware('guest')
    ->name('register');
// verificación en la base de datos
Route::post('/register', [RegisteredUserController::class, 'store'])
    ->middleware('guest');
```

Fig. 15: Rutas del Registro.

Actualizar datos del usuario.

Según lo establecido en la planificación la actualización de los datos es de suma importancia para una aplicación de comercio electrónico, puesto que los datos son la base importante para la comunicación entre usuarios tras la compra y venta de productos. Por ello, la aplicación establece funciones que extraigan las acciones necesarias para la actualización de datos. La implementación dentro del proyecto se establece la creación de cuatro controladores, donde dividen las funciones importantes para la actualización de datos, foto de perfil, Nick y contraseña, lo cual influye mucho para tener organizado la lógica del programa.

La **Fig. 16** establece los métodos necesarios para la implementación del cambio de datos del usuario, y a su vez se establecen validaciones que ayudan a controlar las entradas de los usuarios, donde al verificar un campo si está en blanco o es completado de manera incorrecta, se mostrará un mensaje de error en el campo.

```php
public function update(Request $request)
{

    $request->validate([
        'first_name' => ['required', 'string', 'alpha','min:3', 'max:35'],
        'last_name' => ['required', 'string', 'min:3', 'max:35'],
        'birthdate' => ['nullable', 'string', 'date_format:d/m/Y',
        'after_or_equal:' . date('Y-m-d', strtotime('-70 years')),
        'before_or_equal:' . date('Y-m-d', strtotime('-18 years'))],
        'personal_phone' => ['required', 'numeric', 'digits:10'],
        'home_phone' => ['required', 'numeric', 'digits:9'],
        'address' => ['required', 'string', 'min:5', 'max:50'],
    ]);

    $user = $request->user();

    /*Update the model using Eloquent*/
    $user->first_name = $request['first_name'];
    $user->last_name = $request['last_name'];
    $user->birthdate = $this->verifyDateFormat($request['birthdate']);
    $user->personal_phone = $request['personal_phone'];
    $user->home_phone = $request['home_phone'];
    $user->address = $request['address'];
    $user->save();

    $this->updateUIAvatar($user);
```

Fig. 16: Algoritmo para la validación y actualización de datos.

La **Fig. 17** establece el algoritmo para la actualización del avatar del usuario y a su vez se establece las validaciones correspondientes para determinar el tamaño y tipo de imagen que se puede implementar dentro del proyecto.

```php
class ProfileAvatarController extends Controller
{
    public function update(Request $request)
    {
        $request->validate([
            'image' => ['required', 'image', 'mimes:jpg,png,jpeg', 'max:512']
        ]);

        $user = $request->user();

        $user->updateImage($request['image'], 'avatars');
```

Fig. 17: Algoritmo para la validación y actualización del avatar del usuario.

Implementación del CRUD de las Categorías y Etiquetas.

La implementación de Categorías y Etiquetas dentro del programa, sin duda es necesaria para establecer una organización con respecto al tipo de producto que es fundamental para los usuarios para la búsqueda o definición. Por ello, las categorías y etiquetas serán administradas dentro del programa por un administrador. En la **Fig. 18** se establecen las funcionalidades importantes para administrar las Categorías y Etiquetas.

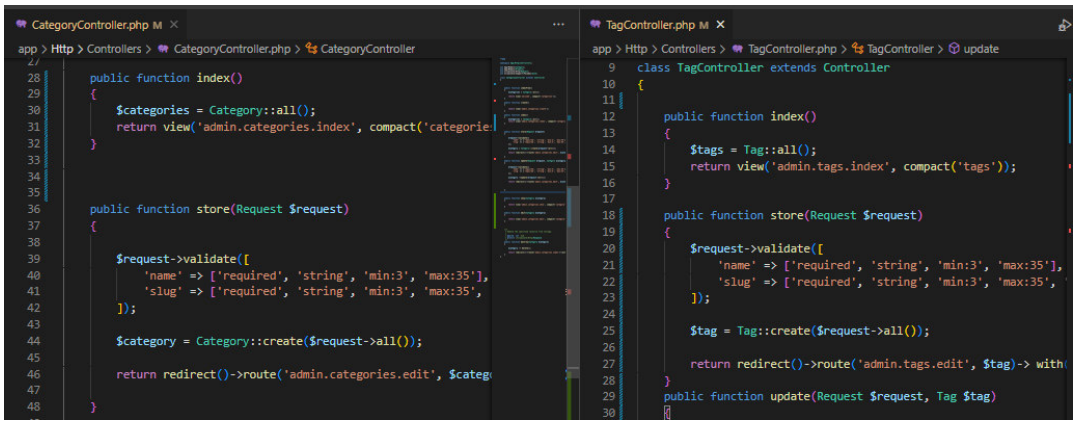


Fig. 18: Métodos para mostrar y crear Categorías y Etiquetas.

Implementación de la lógica de marketing C2C.

En el desarrollo de esta aplicación, los Productos son de suma importancia para establecer las reglas y funcionalidades necesarias que necesita. Por ello, se establecen varias tareas que han sido planificadas de acuerdo con el modelo C2C el cual atribuye el CRUD de productos y a usuarios los roles de clientes y vendedores.

Los productos en el proyecto se establecen en el desarrollo de controladores, que son protegidos a través de políticas, donde solamente los usuarios autenticados pueden crear productos y que a su vez los productos que le pertenecen a dicho usuario podrán ser administrados. Si estas normativas no se cumplen el programa regresara mensajes de error que establecen acciones no autorizadas. En la **Fig. 19** se muestra un ejemplo de los métodos utilizados para la creación del CRUD de Productos.

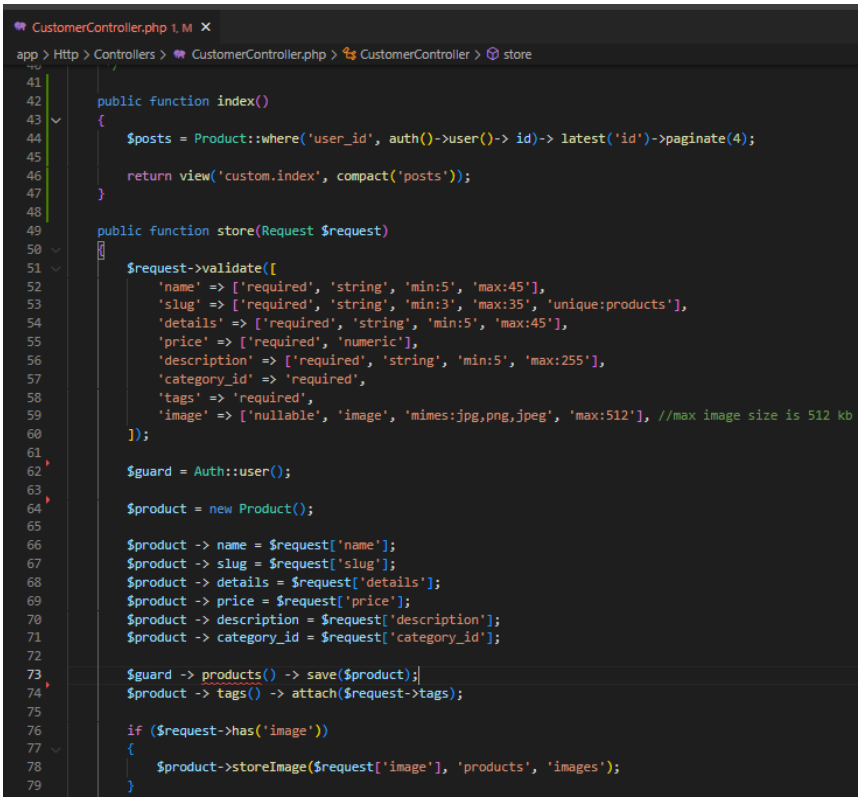


Fig. 19: Funciones para la Lectura y creación de Productos.

Estas funciones implementadas para el usuario final deben estar protegidas por el usuario que no comparte relación con los productos. Por ello, en el proyecto para evitar el acceso no autorizado a usuarios para modificar algún producto se establece la creación de políticas para resguardar los datos. En la **Fig. 20** se establece una política correspondiente al CRUD de Productos donde en cada función del controlador se tiene que validar si el usuario autenticado es o no el propietario de dicho producto.

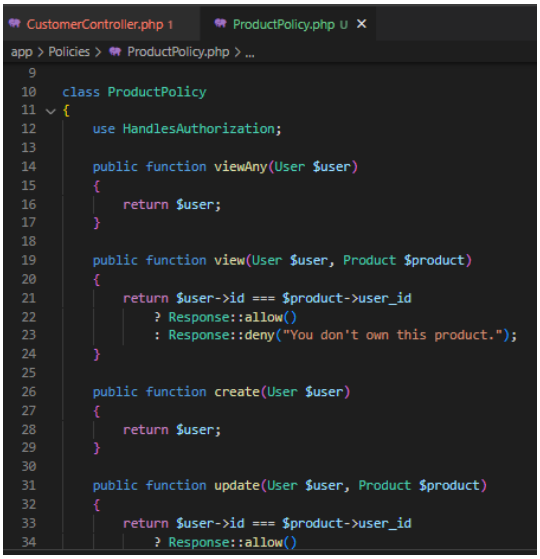
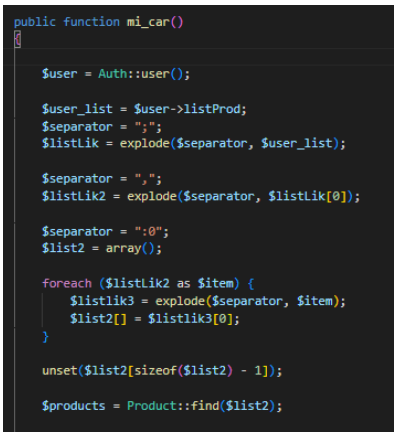


Fig. 20: Política de los Productos.

Los productos en la plataforma estarán en el alcance de cualquier usuario de forma de lectura, en donde podrán buscar e interactuar únicamente con funciones de agregar al carrito de compras o reservar un producto si se lo desea comprar, de esta forma los usuarios podrán realizar las gestiones necesarias para obtener el producto. El sistema con respecto a la presentación de los productos ofrece funciones de búsqueda a través de características esenciales de un producto, a esto se realiza la implementación de funciones que comparen las Categorías o Etiquetas relacionadas a los productos con la búsqueda que los usuarios desean, en la **Fig. 21** se muestra la implementación de esta lógica.

Fig. 21: Funciones de recuperación de datos a través de categorías y etiquetas.

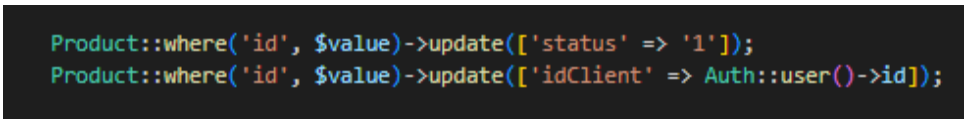
Los usuarios en la aplicación podrán interactuar con los productos en la forma en que pueden revisar la información y la del usuario que lo posteo. Los posts que son implementados dentro de la aplicación contaran con botones que están al alcance de los usuarios para poder guardar el producto en favoritos o cómpralos. Dentro de la lógica del programa se realiza la implementación de una lista que almacena los id's de los productos junto al estado que puede ser favorito, reservado o comprado. De esta forma, su implementación agiliza la obtención y almacenado de los productos obtenidos de un cliente que se lo realiza a través de una lista que le pertenece al usuario. En la **Fig. 22** se muestra la obtención de los productos que se almacenan en la lista de favoritos.



```
public function mi_car()
{

    $user = Auth::user();

    $user_list = $user->listProd;
    $separator = ";";
    $listLik = explode($separator, $user_list);

    $separator = ",";
    $listLik2 = explode($separator, $listLik[0]);

    $separator = ":0";
    $list2 = array();

    foreach ($listLik2 as $item) {
        $listlik3 = explode($separator, $item);
        $list2[] = $listlik3[0];
    }

    unset($list2[sizeof($list2) - 1]);

    $products = Product::find($list2);
```

Fig. 22: Lógica del carrito de compras.

Los usuarios también contaran con un apartado de notificaciones que les avisara si uno de los productos que postearon han sido reservados o comprados, lo que ayuda a mantener la comunicación con el usuario que reservo el producto a través de la obtención de la información del contacto. En la lógica del proyecto se realiza la inyección del id del usuario cliente dentro del registro del producto del usuario vendedor y también se hace un cambio al estado de la publicación lo que permite no ser mostrado dentro de la lista de productos publicados. En la **Fig. 23** se muestra la implementación de este algoritmo.



```
Product::where('id', $value)->update(['status' => '1']);
Product::where('id', $value)->update(['idClient' => Auth::user()->id]);
```

Fig. 23: Lógica de la reserva de un producto.

Sprint 2. Módulos de presentación e interacción con la lógica del proyecto.

El *Sprint* 2 según lo planificado trata del desarrollo de un mini *Frontend* en el cual se puede testear cada una de las funcionalidades impuestas por el *Sprint* 1. Lo cual ayuda en la verificación y simulación del funcionamiento de la aplicación que puede servir de guía para las aplicaciones clientes. A continuación, se presenta la planificación y los resultados obtenidos tras realizar las tareas establecidas dentro del *Sprint*.

- Visualización del perfil de usuario.
- Visualización de otros perfiles.
- Visualización de los productos.
- Visualización del CRUD de Productos a través de módulos.
- Visualización de gestión de productos del carrito.
- Visualización del historial de compras
- Visualización del historial de ventas

Visualización del perfil de usuario.

Tras la realización de la lógica del programa, los controladores junto a sus funciones serán implementadas a través de rutas y vistas las cuales deberán recoger la información y presentarla al usuario a través de interfaces. El perfil de usuario para comprobar su funcionalidad se realiza la implementación de vistas que son creadas con Blade para tener modelos de plantillas y de esta forma presentar todo la información o formularios. En la **Fig. 24** se muestra la vista del perfil de usuario en la cual se presenta a través de un formulario su información más importante y en la cual podrán actualizar los datos si lo desean.

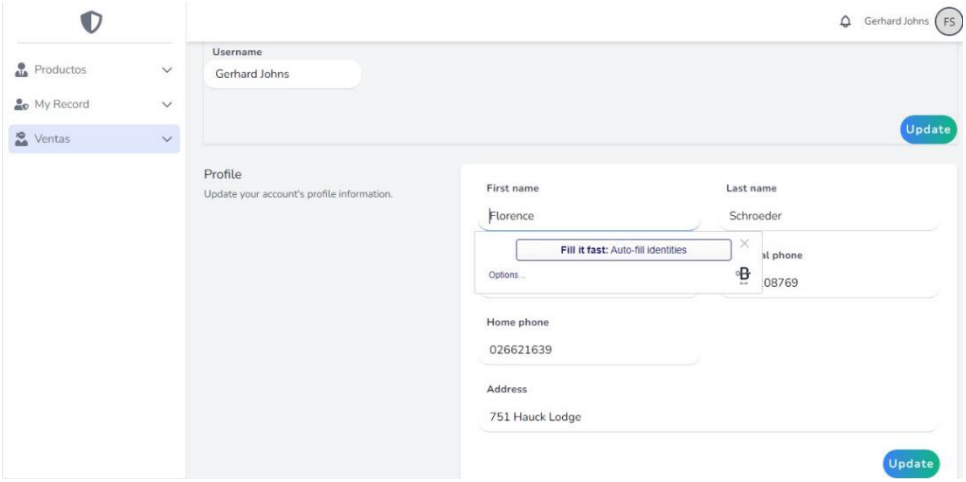


Fig. 24 Modulo del Perfil de usuario autenticado.

En la **Fig. 25** se muestra otra vista que recoge la información de los usuarios que a diferencia del perfil de usuario esta vista es de forma pública para todos los usuarios, lo que quiere decir que cualquier usuario puede revisar la información de contacto o su reputación con respecto a las ventas y junto a ello los productos que son vendidos.

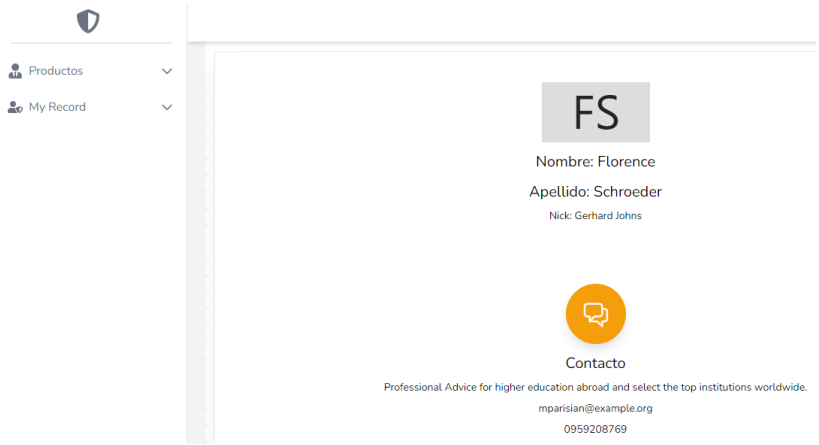


Fig. 25: Modulo de información del perfil de usuarios.

Visualización de los productos.

Dentro la aplicación, los productos se presentarán de forma pública a todo usuario donde la información puede estar presente en todo momento. La aplicación organiza los productos a través de diferentes vistas, la vista presenta todos los productos o puede presentarlos a través de categorías o etiquetas. En la **Fig. 26** se muestra la presentación de todos los productos, la presentación de los productos filtrados por categorías y etiquetas se mostrarán dentro la sección Interfaces del **ANEXO II**.

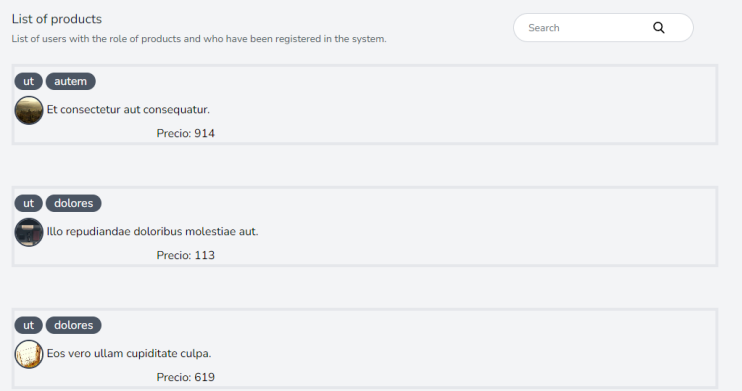


Fig. 26: Presentación de la lista de todos los productos.

Visualización del CRUD de Productos a través de módulos.

La aplicación ofrecerá a los usuarios la posibilidad de realizar publicaciones sobre sus prendas de vestir que quieran vender, a través de la **Fig. 27** se muestra la plantilla de creación de un producto. De esta forma, el usuario tendrá un apartado para gestionar sus publicaciones como se muestra en la **Fig. 28** en la cual, al momento de crear una nueva publicación, esta pueda presentarse dentro del listado de publicaciones y así mismo poder cambiar su estado de publicación dependiendo si el producto ha sido reservado o haya sido inhabilitado.

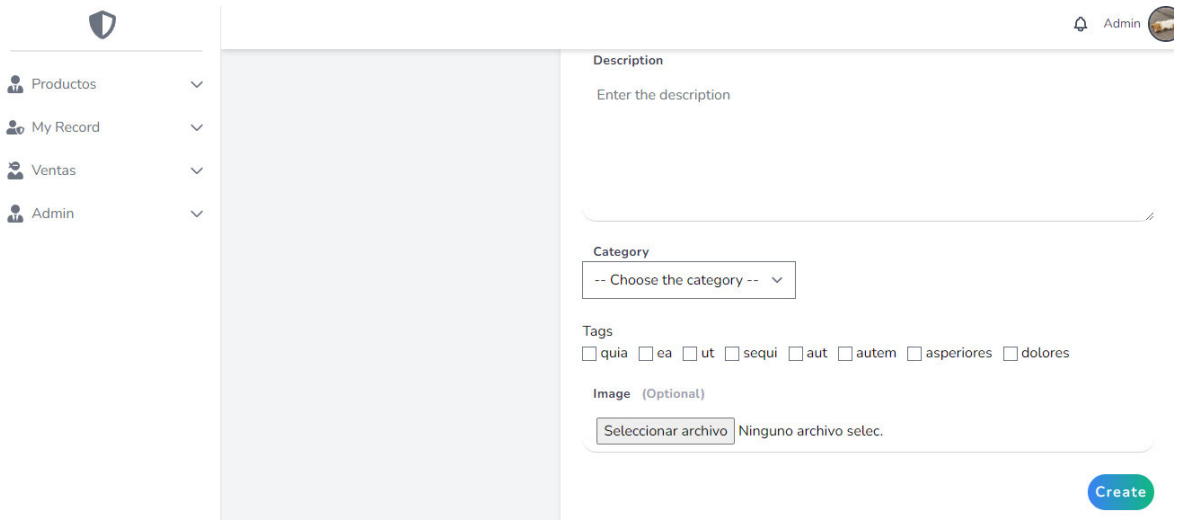


Fig. 27: Formulario de creación de una publicación

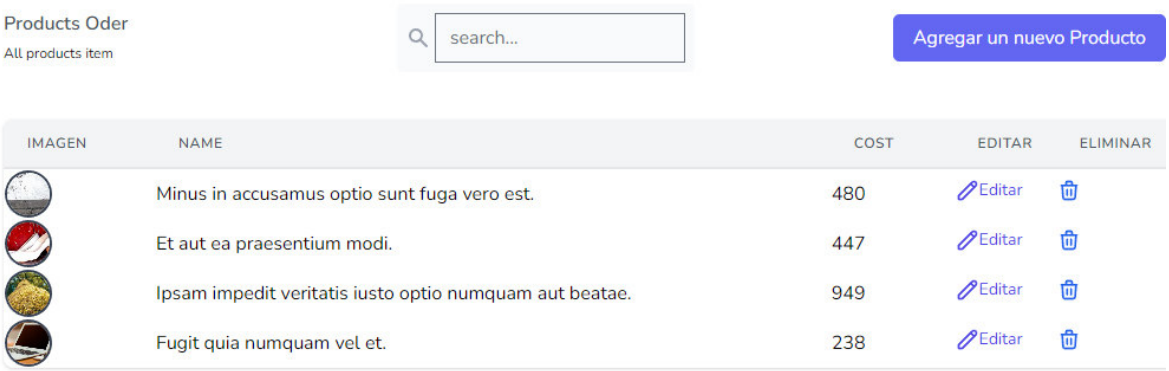


Fig. 28: Lista de productos publicados del usuario.

Visualización de gestión de productos del carrito.

En la aplicación se podrá ver los datos de un producto que el usuario seleccione, pero a su vez este usuario puede guardar esa publicación dentro de su listado llamado carrito de compras. Un usuario al realizar dicha acción de guardar su producto podrá recibir un mensaje como notificación como en la **Fig. 29** que le mostrara si se hizo correctamente la acción y que podrá ser comprobado a través de la visualización del carrito de compras como se muestra en la **Fig. 30**, además contara con validación por si el usuario ya había añadido el producto o si le pertenece el producto.

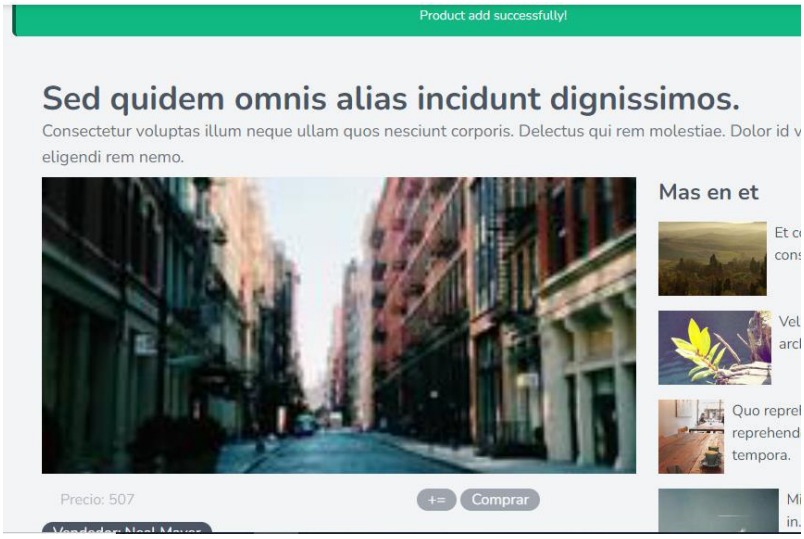


Fig. 29: Producto añadido al carrito de compras.

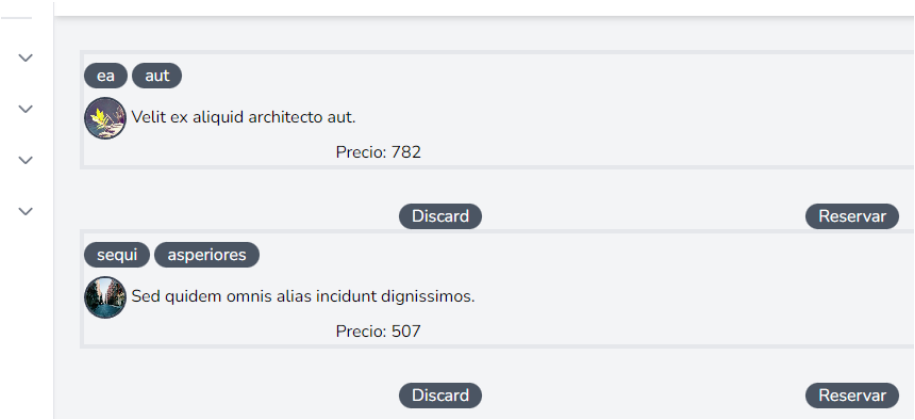


Fig. 30: Lista de los productos añadidos al carrito de compras.

Visualización del historial de compras.

El usuario al momento de almacenar en su carrito de compras un producto puede realizar la reserva del mismo por ese medio o también puede reservar el producto desde la publicación, lo que ocasionara que el producto cambie su estado e implemente la inserción de nuevos campos que reflejaran la comunicación con el usuario que haya reservado de esta manera el usuario que realizo la publicación podrá contactarse con el comprador, en la **Fig. 31** se puede visualizar el listado de los producto que hayan sido reservados y de la misma forma los productos comprados junto a los datos del vendedor.

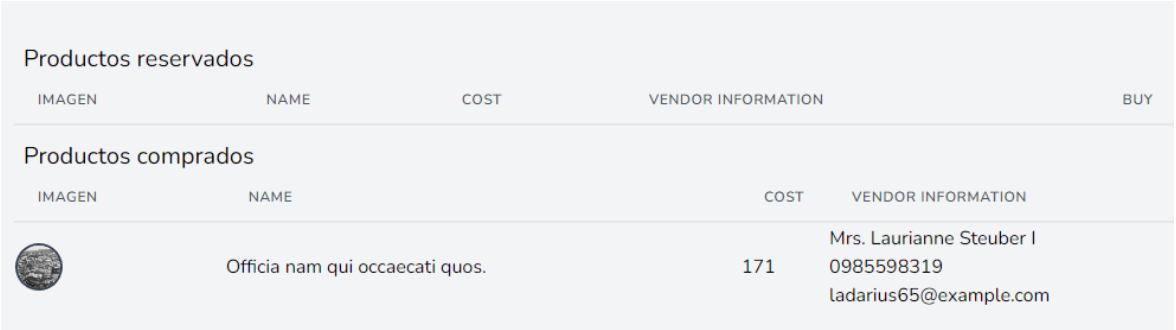


Fig. 31: Listado de los productos reservados y comprados.

Visualización del historial de ventas.

El usuario vendedor tendrá un apartado aparte que mostrará notificaciones de los usuarios que hayan reservado sus productos y de la misma forma los productos vendidos que estarán junto a los datos del comprador y a su vez podremos visualizar la calificación de ventas como se muestra en la **Fig. 32**.

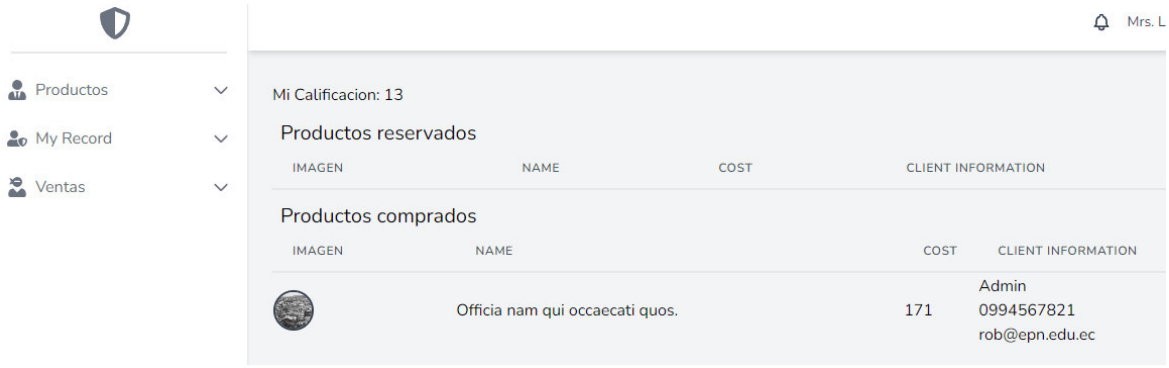


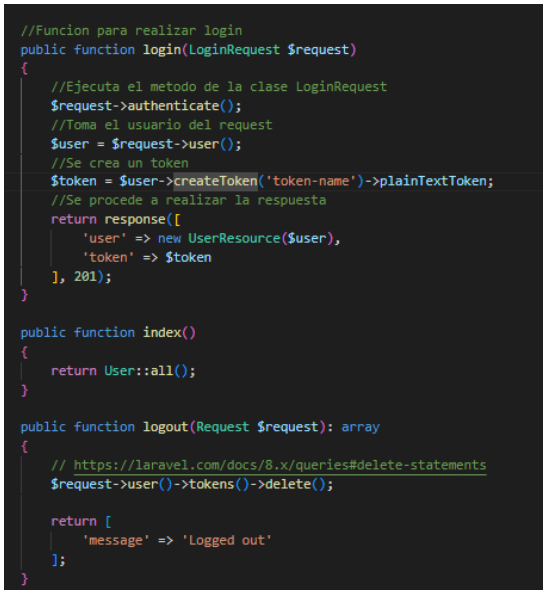
Fig. 32: Listado de los productos reservados y vendidos del usuario.

Sprint 3. Interfaces externas.

Acorde a la planificación del Sprint Backlog, el *Sprint 3* corresponde a la creación de *API's REST*. *Laravel* nos proporciona otro ámbito para no solamente generar rutas de tipo web sino de tipo *API REST*, que en base a la lógica de negocio y desarrollo de métodos realizados en los anteriores Sprints se podrá generar las rutas necesarias con facilidad y correcto funcionamiento, y de esta forma cumpliendo el propósito de conectar dos sistemas y compartir información usando la arquitectura de Transferencia de Estado Representacional (*REST*) que implementa el protocolo HTTP para la comunicación entre cliente y servidor.

Las rutas *API's* en el proyecto necesitaran de controladores únicos que implementen la lógica correspondiente que necesitaran las rutas, para realizar diferentes acciones como la visualización, creación, actualización y eliminación de un determinado elemento que puede ser comprobado a través de mensajes de retorno que validan si la acción es correcta o el retorno de los datos a través de un formato tipo JSON.

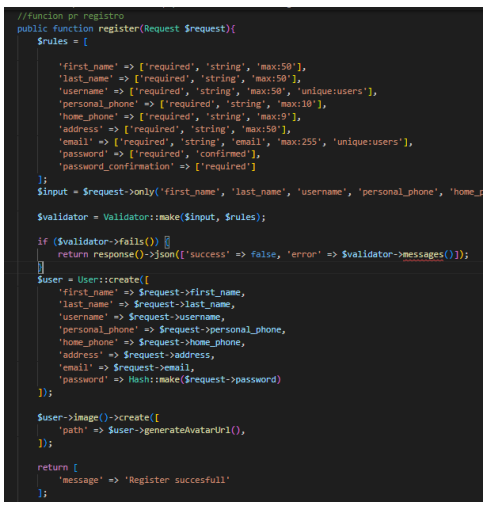
En la **Fig. 33** se muestra la implementación de los métodos para el inicio de sesión y cierre de sesión que necesita para generar las rutas API's respectivas.



```php
//Funcion para realizar login
public function login(LoginRequest $request)
{
    //Ejecuta el metodo de la clase LoginRequest
    $request->authenticate();
    //Toma el usuario del request
    $user = $request->user();
    //Se crea un token
    $token = $user->createToken('token-name')->plainTextToken;
    //Se procede a realizar la respuesta
    return response([
        'user' => new UserResource($user),
        'token' => $token
    ], 201);
}

public function index()
{
    return User::all();
}

public function logout(Request $request): array
{
    // https://laravel.com/docs/8.x/queries#delete-statements
    $request->user()->tokens()->delete();

    return [
        'message' => 'Logged out'
    ];
}
```

Fig. 33: Lógica de autenticación de usuario para rutas API's

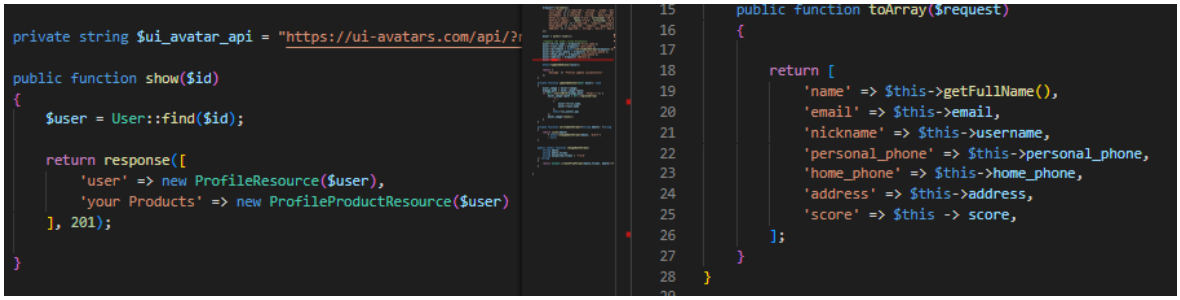
En la **Fig.34** se muestra la lógica necesaria para el registro de usuario que se hace a través de obtención de entradas que realiza el usuario y son comprobadas a través de validaciones, de esta forma se crean los usuarios respectivos para que se pueda insertar dentro de la base de datos.



```php
//funcion pr registro
public function register(Request $request){
    $rules = [

        'first_name' => ['required', 'string', 'max:50'],
        'last_name' => ['required', 'string', 'max:50'],
        'username' => ['required', 'string', 'max:50', 'unique:users'],
        'personal_phone' => ['required', 'string', 'max:10'],
        'home_phone' => ['required', 'string', 'max:9'],
        'address' => ['required', 'string', 'max:50'],
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'password' => ['required', 'confirmed'],
        'password_confirmation' => ['required']
    ];
    $input = $request->only('first_name', 'last_name', 'username', 'personal_phone', 'home_p

    $validator = Validator::make($input, $rules);

    if ($validator->fails()) {
        return response()->json(['success' => false, 'error' => $validator->messages()]);
    }
    $user = User::create([
        'first_name' => $request->first_name,
        'last_name' => $request->last_name,
        'username' => $request->username,
        'personal_phone' => $request->personal_phone,
        'home_phone' => $request->home_phone,
        'address' => $request->address,
        'email' => $request->email,
        'password' => Hash::make($request->password)
    ]);

    $user->image()->create([
        'path' => $user->generateAvatarUrl(),
    ]);

    return [
        'message' => 'Register succesful1'
    ];
```

Fig. 34: Lógica correspondiente al registro de usuario para la ruta *API*

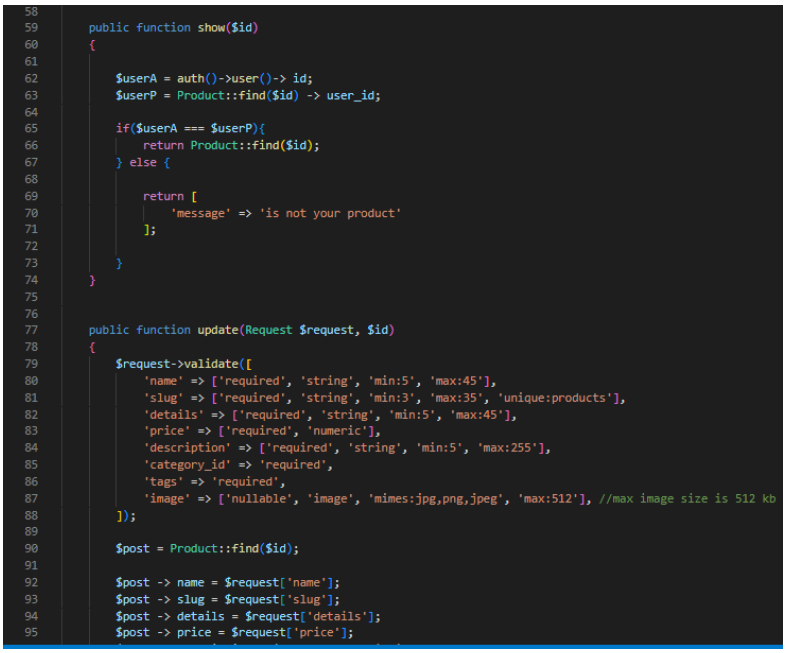
De esta forma, la mayor parte de *API's* se generan a través de métodos ya realizados en los anteriores *Sprints*, y cumpliendo las características *REST* como puede ser la representación de los datos a través de un formato de tipo JSON como se muestra en la **Fig. 35,** y la utilización de verbos HTTP que integran las opciones CRUD que son representadas a través de métodos como se muestra en la **Fig. 36** y los cuales serán identificados dependiendo de la acción que realice al momento de definir la rutas que pueden ser POST o INDEX.



```
private string $ui_avatar_api = "https://ui-avatars.com/api/?

public function show($id)
{
    $user = User::find($id);

    return response([
        'user' => new ProfileResource($user),
        'your Products' => new ProfileProductResource($user)
    ], 201);

}
```

```
    public function toArray($request)
    {

        return [
            'name' => $this->getFullName(),
            'email' => $this->email,
            'nickname' => $this->username,
            'personal_phone' => $this->personal_phone,
            'home_phone' => $this->home_phone,
            'address' => $this->address,
            'score' => $this -> score,
        ];
    }
}
```

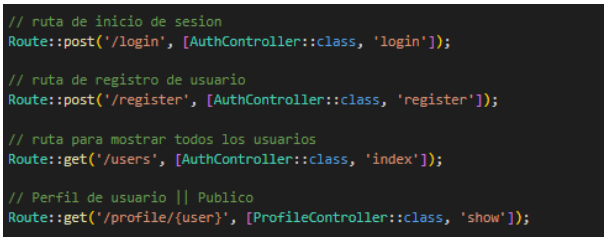
Fig. 38: Organización de los datos de un usuario



```
public function show($id)
{

    $userA = auth()->user()-> id;
    $userP = Product::find($id) -> user_id;

    if($userA === $userP){
        return Product::find($id);
    } else {

        return [
            'message' => 'is not your product'
        ];

    }
}


public function update(Request $request, $id)
{
    $request->validate([
        'name' => ['required', 'string', 'min:5', 'max:45'],
        'slug' => ['required', 'string', 'min:3', 'max:35', 'unique:products'],
        'details' => ['required', 'string', 'min:5', 'max:45'],
        'price' => ['required', 'numeric'],
        'description' => ['required', 'string', 'min:5', 'max:255'],
        'category_id' => 'required',
        'tags' => 'required',
        'image' => ['nullable', 'image', 'mimes:jpg,png,jpeg', 'max:512'], //max image size is 512 kb
    ]);

    $post = Product::find($id);

    $post -> name = $request['name'];
    $post -> slug = $request['slug'];
    $post -> details = $request['details'];
    $post -> price = $request['price'];
```

Fig. 39: Métodos que representan los verbos HTTP

El sistema web dispone de una serie de *API's REST* que son divididas en acceso público y privado, estas rutas privadas se acceden al contenido y sus funcionalidades a partir de la autenticación del usuario. Al definir las rutas también se cumple con los estándares *REST* de acceder al tipo de información a través de un identificador y que cada petición es totalmente independiente. En la **Fig. 40** se muestra un ejemplo de *API's* públicas y en la **Fig. 41** se encuentra otro ejemplo de *API's* privadas.



```
// ruta de inicio de sesion
Route::post('/login', [AuthController::class, 'login']);

// ruta de registro de usuario
Route::post('/register', [AuthController::class, 'register']);

// ruta para mostrar todos los usuarios
Route::get('/users', [AuthController::class, 'index']);

// Perfil de usuario || Publico
Route::get('/profile/{user}', [ProfileController::class, 'show']);
```

Fig.40: Rutas *API's* Publicas

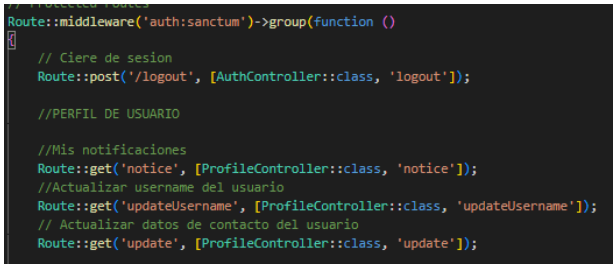


Fig.41: Rutas *API's* Protegidas

Sprint 4. Pruebas unitarias del Backend.

De acuerdo con la planificación, el Sprint 4 conforma la realización de las pruebas unitarias, las cuales comprobaran el funcionamiento de los algoritmos impuestos para la aplicación. En metodologías agiles es fundamental realizar este tipo de testeos antes de realizar el levantamiento a producción de la aplicación puesto que es un paso para la autenticidad del buen rendimiento y un correcto funcionamiento.

La realización de la verificación del buen funcionamiento de las rutas API's se hacen a través de peticiones HTTP, de esta forma al realizar la verificación con rutas públicas se obtiene la recuperación de los datos, aunque si se realiza en rutas privadas se obtiene una restricción que ayuda a mantener la confidencialidad de determinados datos en este caso serían las actualizaciones de los datos del usuario o productos.

La verificación para rutas privadas se hace a través de la obtención de un token de autenticación, el cual podrá avisar al servidor que un usuario determinado puede realizar acciones que tiene a su disposición únicamente para la lectura y modificación de sus datos. En la **Fig. 42** se realiza la petición POST para realizar el inicio de sesión. Las demás pruebas unitarias pueden ser visualizadas dentro la sección Pruebas Unitarias que se encuentran dentro del **ANEXO II**.

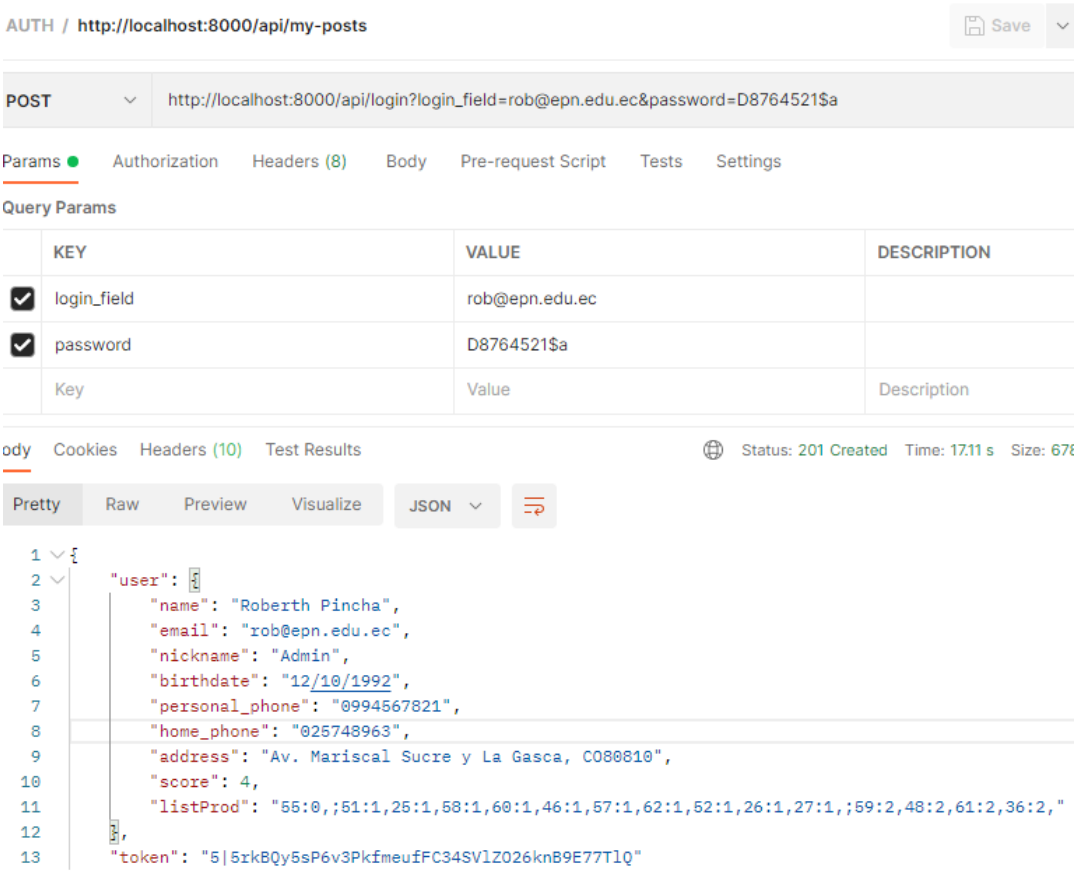


Fig.42: Petición HTTP – Inicio de Sesión.

En la **Fig. 43** se realiza el testeo sobre implementación de una petición que compruebe el resultado de la información obtenida tras realizar el inicio de sesión, este tipo de testo se realiza para comprobar y verificar que el usuario tenga un campo de Token, un identificador único, email y el estado de la petición sea 200.

Fig.43: Pruebas unitarias para petición HTTP – Inicio de Sesión

En la **Fig. 44** se muestra la respuesta obtenida tras la implementación de pruebas unitarias con respecto al inicio de sesión, esto comprueba que el usuario pertenece a la base de datos, que tiene elementos necesarios para la comunicación entre usuarios y sobre todo le otorga el token necesario para realizar las acciones que se encuentran restringidas sin antes iniciar sesión.

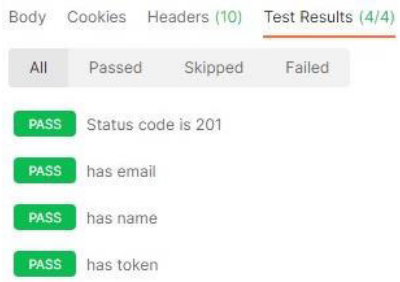


Fig.44: Respuesta de pruebas unitarias para petición HTTP – Inicio de Sesión

Sprint 5. Despliegue del componente Backend.

El objetivo de la aplicación es generar un entorno servidor para que aplicaciones clientes puedan comunicarse y compartir información. De esta forma, el Sprint 5 trata de generar el ambiente de producción de la aplicación usando herramientas de plataformas como servicio y base de datos como servicio, de esta forma la aplicación podrá obtener un espacio de alojamiento dentro de la Web. A continuación, se presenta la planificación y los resultados obtenidos tras realizar las tareas establecidas dentro del Sprint.

- Usar un modelo de base de datos como servicio (DBaaS).
- Configuración de Heroku.
- Desplegar el componente a producción.

Usar un modelo de servicio DBaaS

El proyecto en producción se compone de diferentes servicios que definen el modelo de la arquitectura, estos servicios son un gestor de base de datos y un almacenamiento multimedia lo cual estos servicios pueden ser implementados usando servicios en la nube. AlwaysData es una base de datos como servicio de manera gratuita la cual ofrece un gestor de base de datos lo que ayuda a mantener activos los registros en todo momento. Dropbox es un servicio de alojamiento de archivos multimedia, sin duda es servicio único para mantener también activos las imágenes como los productos y avatar activos en todo el momento. Estas plataformas ofrecen opciones para desarrolladores en los cuales nos ayudan con el alojamiento y un limitado almacenamiento pero que sin duda mantiene la arquitectura de la aplicación en todo momento.

Configuración de Heroku.

Heroku es una alternativa única para el despliegue de software, y permite usar herramientas como *Git* para manejar el contenido del proyecto. En primera instancia se necesita la interfaz de línea de comandos (CLI) de la plataforma puesto que introduce los mecanismos y herramienta necesarias, a su vez el proyecto debe contar con la generación de un archivo que tenga la referencia de la configuración del servidor de Apache como se muestra en la **Fig. 45**.

Fig. 45: Archivo *Procfile*

La implementación de estos elementos ayudara a continuar el proceso de despliegue sin problemas. Heroku permite crear aplicaciones las cuales son como repositorios que podemos gestionar y donde se alojaran las aplicaciones. En la **Fig. 46** se muestra la aplicación generada para el despliegue de la aplicación y junto a las indicaciones necesarias.

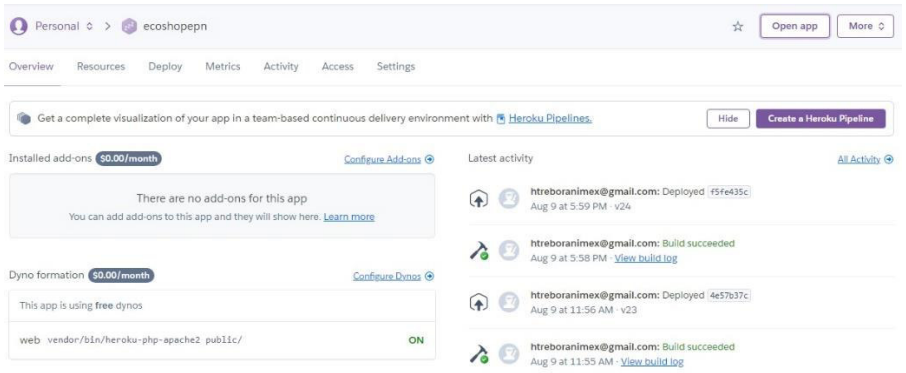


Fig. 46: Aplicación generada en Heroku.

Desplegar el componente a producción

La plataforma como servicio Heroku es una plataforma que ofrece el despliegue de componentes de software de manera gratuita, aunque estos servicios son limitados como el almacenamiento. Como solución a estas limitantes, Heroku posee configuraciones que permiten hacer uso de otros servicios de software, como el almacenamiento o un servidor SMTP a través de la inyección de las variables de entorno correspondientes. En la **Fig. 47** se muestra las variables de entorno que necesita el proyecto para funcionar una vez que esté en producción.

Fig.47: Implementación de variables de entorno.

Al momento de inyectar las variables de entorno, el proyecto puede ser desplegado a producción en un entorno de la nube, pero antes es necesario que las migraciones junto a los *Seeders,* se ejecuten en los determinados servicios correspondientes de esta forma la aplicación podrá empezar a funcionar con todos sus elementos y registros necesarios. En la **Fig. 48** se muestra el proyecto desplegado junto a su dominio.

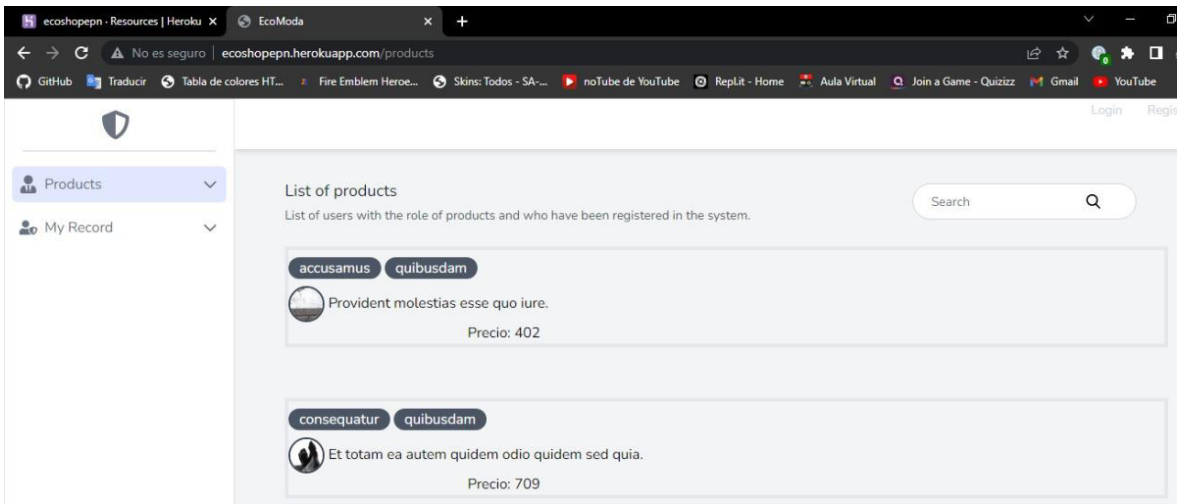


Fig.48: Despliegue a producción del componente *Backend*.

4 CONCLUSIONES

- Hacer uso de metodologías Agiles como Scrum mantienen los requerimientos y objetivos organizados, a través de la utilización de normativas que otorgan la facilidad de un desarrollo simultaneo o la realización de cambios, lo que beneficia a la identificación de los procesos a realizar de manera que se encuentre un balance en el desarrollo y tiempo, lo cual da valor agregado al proyecto y fomenta la calidad de este.

- Mediante el uso de *Laravel* que es un *framework* totalmente novedoso que su principal característica, es el integra varios servicios y paquetes, y junto a sus constantes actualizaciones hacen que este *framework* sea realmente importante para el desarrollo web, puesto que permite acoplarse de mejor manera con los objetivos de nuestros *stackholders* y el desarrollo con elementos únicos.

- El componente *Backend* es sin duda un pilar importante debido a que integra funcionalidades únicas, con el fin, de ser capaces de cumplir con los requerimientos de manera oportuna y eficaz, dándole así el valor y experiencia que todo cliente necesita. El proyecto desarrollado cumple satisfactoriamente los requerimientos definidos puesto que al generar una *API REST*, se está cumpliendo con los objetivos de crear un ambiente único, en el cual compartir las funcionalidades y registros con aplicaciones clientes que desarrollan el *Frontend*, muestran la importancia del desarrollo web y otorga a los usuarios la experiencia única.

- El componente *Backend* implementa un mini *Frontend* que muestra la utilización de la aplicación, esto beneficia mucho al momento de explicar su uso y mostrar al cliente como utilizar dichas *API's*.

- Al realizar la aplicación junto al uso de metodologías agiles, es importantes al momento de terminar las funcionalidades del proyecto realizar pruebas, que verifiquen el correcto funcionamiento de los métodos desarrollados y también comprobar el acceso de los usuarios.

- El despliegue de la aplicación se ha realizado a través de la utilización de una plataforma como servicio llamado Heroku, la cual ofrece las herramientas y servicios necesarios, para que el proyecto pueda estar disponible en la web y que puedan las aplicaciones clientes hacer uso de las API's generadas por la aplicación.

5 RECOMENDACIONES

- Implementar metodologías agiles fomenta el trabajo continuo, desarrollo sostenible y el contacto con el cliente, lo que beneficia enormemente para que el trabajo pueda cumplir con todas las expectativas y entrega a tiempo a los clientes.

- El desarrollo del *Backend* es sin duda importante y por ello, es necesario conocer el modelo de negocio que establecen las entidades anfitrionas, las cuales ayudaran a realizar la base de datos, y que se debe tener un control continuo con el registro de los datos, que ayudan a verificar que los modelos y relaciones estén cumpliendo con los objetivos de una aplicación.

- Las funcionalidades deben cumplir con la confidencialidad y privacidad a los datos, puesto que al generar *API's* que son consumibles por otras aplicaciones clientes, puedan ofrecer la protección necesaria ante la lectura o cambio de datos de usuarios que no hayan sido autenticados o no posean esa información.

- Establecer las *API's* dentro del componente *Backend*, determinan los principios claves de comunicación entre cliente y servidor, por ello se implementan estándares que son mostrados a través de la documentación oficial que es una guía única para establecer los verbos HTTP y que también son una solución ante los errores que puedan aparecer en la aplicación.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] T. Burki, «COVID-19 IN LATIN AMERICA,» The Lancet, 17 Abril 2020. [En línea]. Available: https://www.thelancet.com/journals/laninf/article/PIIS1473-3099(20)30303-0/fulltext. [Último acceso: 15 Mayo 2022].

[2] J. Jacome, «CONTAGIOS DIARIOS EN ECUADOR CAEN A MENOS DE LA MITAD, EL LUNES,» La Republica, 18 Enero 2022. [En línea]. Available: https://www.larepublica.ec/blog/2022/01/18/ecuador-registra-4-448-nuevos-casos-de-covid-y-acumula-629-507-positivos/. [Último acceso: 18 Mayo 2022].

[3] J. G. Vera Ortiz, A. X. Vera Barzola, y M. L. Parrales Poveda, «CRISIS ECONÓMICA DEL ECUADOR: UNA MIRADA AL SECTOR MICROEMPRESARIAL POST COVID-19: CRISIS ECONÓMICA DEL ECUADOR», UNESUM-Ciencias, vol. 4, n.º 4, pp. 1-14, nov. 2020.

[4] M. d. Quito, «COMERCIANTES DE LOS MERCADOS CONTINÚAN CAPACITÁNDOSE PARA PREVENIR EL COVID 19,» Quito Informa, 09 Marzo 2020. [En línea]. Available: http://www.quitoinforma.gob.ec/2020/03/09/comerciantes-de-los-mercados-continuancapacitandose-para-prevenir-el-covid-19/. [Último acceso: 18 Mayo 2022].

[5] R. Villarroel, y F. Alexander, «CAMBIOS POST COVID-19 EN EL SECTOR DEL MERCADO SAN ROQUE: COMERCIALIZACIÓN Y SOBREVIVENCIA,» [En línea]. Available: http://hdl.handle.net/10644/8285. [Último acceso: 19 Mayo 2022].

[6] M. Alcaraz, «CORONAVIRUS: CÓMO EVITAR CONTAGIOS AL HACER LA COMPRA,» ABC, 27 Marzo 2020. [En línea]. Available: https://www.abc.es/bienestar/alimentacion/abci-covid-19-como-evitar-contagioshacer-compra-202003231108_noticia.html?ref=https:%2F%2Fwww.google.com%2F. [Último acceso: 19 Mayo 2022].

[7] D. Monasterio y M. Briceño, «EDUCACIÓN MEDIADA POR LAS TECNOLOGÍAS: UN DESAFÍO ANTE LA COYUNTURA DEL COVID-19, » rev_ODC, vol. 5, n.º 1 enero-ab, pp. 100-108, oct. 2020.

[8] G. Velásquez, «METODOLOGÍA DE DESARROLLO DE SOFTWARE,» Universidad Católica de los Ángeles Chimbote, 19 Diciemnre 2017. [En línea]. Available: https://www.uladech.edu.pe/images/stories/universidad/documentos/2018/metodologia-desarrollo-software-v001.pdf. [Último acceso: 20 Mayo 2022].

[9] A. Ramos-Blanco, H. G. Pérez-González, S. E. Nava-Muñoz & F. E. Martínez-Pérez, «MIC-AGILE: METODOLOGÍA ÁGIL PARA MICRO-EQUIPOS DE DESARROLLO DE SOFTWARE, » Revista Ingenio, vol. 19(1), pp. 1-8, 2022 [Último acceso: 21 Mayo 2022].

[10] L. Fitzgibbons, «FRONT END AND BACK END, » Whatls, [En línea]. Avaliable: https://whatis.techtarget.com/definition/front-end. [Último acceso: 21 Mayo 2022].

[11] V. Valverde, N. Portalanza y P. Mora, «ANÁLISIS DESCRIPTIVO DE BASE DE DATOS RELACIONAL Y NO RELACIONAL,» Revista Atlante: Cuadernos de Educación y Desarrollo junio 2019. En línea: https://www.eumed.net/rev/atlante/2019/06/base-datos-relacional.html. [Último acceso: 21 Mayo 2022].

[12] «FRONT END AND BACK END, » Red Hat, 8 de mayo 2020. [En línea]. Avaliable: https://www.redhat.com/es/topics/api/what-is-a-rest-api. [Último acceso: 21 Mayo 2022].

[13] M. T. Gallego, «METODOLOGÍA SCRUM,» [En línea]. Available: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf. [Último acceso: 23 Mayo 2022].

[14] O. Garcia, « LOS TRES PRINCIPALES ROLES EN SCRUM» projectum, 19 Octubre 2016. [En línea]. Available: https://www.proyectum.com/sistema/blog/los-tres-principales-roles-en-scrum/ [Último acceso: 23 Mayo 2022].

[15] Laravel, «LARAVEL DOCUMENTATION,» Laravel, [En línea]. Available: https://laravel.com/docs/8.x.. [Último acceso: 26 Mayo 2022].

[16] J. M. Aguilar, «¿QUÉ ES EL PATRÓN MVC EN PROGRAMACIÓN Y POR QUÉ ES ÚTIL?,» campusMVP, 15 Octubre 2019. [En línea]. Available: https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx. [Último acceso: 30 Mayo 2022].

[17] R. Patiño, «¿QUÉ ES COMPOSER Y CÓMO SE USA EN PHP?» ANEXSOFT, [En línea]. Available: https://anexsoft.com/que-es-composer-y-como-se-usa-en-php. [Último acceso: 10 Junio 2022].

[18] «WINDOWS PREGUNTAS FRECUENTES,» Apache Friends, [En línea]. Available: https://www.apachefriends.org/es/faq_windows.html. [Último acceso: 10 Junio 2022].

[19] A. M. Robledano, «QUÉ ES MYSQL: CARACTERÍSTICAS Y VENTAJAS,» OpenWebinars, 24 Septiembre 2019. [En línea]. Available: https://openwebinars.net/blog/que-esmysql/. [Último acceso: 14 Junio 2022].

[20] A. López, «DESARROLLO WEB - QUE ES POSTMAN Y PARA QUE SIRVE,» Open Webinars, 03 Junio 2019. [En línea]. Available: https://openwebinars.net/blog/que-espostman/. [Último acceso: 25 Junio 2022].

[21] «DOCUMENTATION,» Heroku Dev Center, [En línea]. Available: https://devcenter.heroku.com/categories/reference. [Último acceso: 25 Junio 2022].

[22] J. Astigarraga y V. Cruz-Alonso, « ¡ SE PUEDE ENTENDER CÓMO FUNCIONAN GIT Y GITHUB! ,» Ecosistemas, 17 Diciembre 2021. [Último acceso: 28 Junio 2022]

7 ANEXOS

ANEXO I. Certificado de Originalidad.

ANEXO II. Manual técnico.

ANEXO III. Manual de usuario.

ANEXO IV. Manual de instalación.

ANEXO I Certificado de originalidad

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

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 12 de septiembre de 2022

De mi consideración:

Yo, Richard Paúl Rivera Guevara, en calidad de Director del Trabajo de Integración Curricular titulado DESARROLLO DEL BACKEND asociado al DESARROLLO DE SISTEMA WEB Y APLICACIÓN MÓVIL DE VENTA DE ROPA DE SEGUNDA MANO elaborado por el estudiante ROBERTH FABIAN PINCHA HUANCA de la carrera en TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito desde las Sección II en adelante, 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,

RICHARD PAUL RIVERA GUEVARA
Firmado digitalmente por RICHARD PAUL RIVERA GUEVARA
Fecha: 2022.09.12 00:44:12 -05'00'

Richard Rivera
Profesor Ocasional a Tiempo Completo
ESFOT

ANEXO II Manual técnico

1. RECOPILACION DE LOS REQUERIMIENTOS.

En esta sección se muestran los requerimientos obtenidos para el desarrollo del componente *Backend*. A continuación, se presenta la **TABLA IV** con los requerimientos.

TABLA IV: Recopilación de requerimientos

RECOPILACION DE REQUERIMIENTOS	
ID - RR	**ENUNCIADO DEL ÍTEM**
RR001	Como usuario final y administrador, debo iniciar sesión/cerrar sesión del sistema web.
RR002	Como usuario final, debo registrar usuarios a través de un formulario.
RR003	Como usuario final, puedo editar mi perfil (cambiar nombre, foto).
RR004	Como usuario final, debo visualizar, agregar y descartar productos dentro de las diferentes opciones que tiene un comprador o vendedor.
RR005	Como usuario final, puedo usar un método de búsqueda a partir de alguna característica del producto.
RR006	Como usuario final, puedo visualizar mi perfil.
RR007	Como usuario final, puedo visualizar la información de otros usuarios (Información del usuario y reputación).
RR008	Como usuario final, debo visualizar los productos y listarlos a través de una alguna característica.
RR009	Como usuario final, puedo realizar la publicación de mis productos.
RR010	Como usuario final, puedo gestionar el carrito de compras.
RR011	Como usuario final, puedo gestionar el historial de compras.
RR012	Como usuario final, puedo gestionar el historial de ventas.
RR013	El sistema de tipo *Backend*, debe generar rutas de tipo *API REST*.
RR014	La aplicación debe realizar pruebas que certifiquen el funcionamiento del programa.
RR015	Subir la base de datos a AlwaysData y el *Backend* a Heroku.

2. HISTORIAS DE USUARIO

A continuación, se presentan el listado de las historias de usuario del proyecto desde la **TABLA V** hasta la **TABLA XIII.**

TABLA V: Historia de Usuario - HU002

HISTORIA DE USUARIO	
Identificador (ID): HU002	**Usuario:** Usuario final
Nombre Historia: Registrar usuarios	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Iteración Asignada: 1	
Responsable: Roberth Pincha	
Descripción: El usuario final pueda ingresar al sistema web debe crearse una cuenta llenando los campos requeridos: • Nombre • Apellido • Nombre de usuario • Teléfono • Celular • Correo electrónico	
Observación: El registro constará con validaciones que se visualizarán en el caso de no completar los campos requeridos.	

TABLA I: Historia de Usuario HU003

HISTORIA DE USUARIO	
Identificador (ID): HU003	**Usuario:** Usuario final
Nombre Historia: Actualizar datos del usuario	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 1	
Responsable: Roberth Pincha	
Descripción: El usuario final podrá realizar una modificación de sus datos más importantes dentro de las configuraciones de perfil.	
Observación: El usuario final cuando inicio sesión puede gestionar su perfil de usuario y poder realizar la edición de sus datos importantes.	

TABLA III: Historia de Usuario HU004

HISTORIA DE USUARIO	
Identificador (ID): HU004	**Usuario:** Usuario final
Nombre Historia: Implementación del CRUD de las Categorías y Etiquetas.	
Prioridad en Negocio: Medio	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 1	
Responsable: Roberth Pincha	
Descripción: El usuario puede realizar la búsqueda de productos mediante la elección de una categoría o etiqueta, la cual mostrara una página con un listado de productos mayormente relacionados con la búsqueda. El Administrador puede contar con funcionalidades exclusivas para cambiar la información de una categoría o etiqueta.	
Observación: El usuario final podrá acceder a una vista donde se muestran las categorías o etiquetas de la aplicación web, la cual permitirá a los usuarios buscar productos, simplemente seleccionando alguna característica propia del producto a través de opciones. Se muestran los resultados mayormente relacionados con la búsqueda. Además, el administrador tendrá el alcance de cambiar la información de las categorías o etiquetas.	

TABLA IIIII: Historia de Usuario HU005

HISTORIA DE USUARIO	
Identificador (ID): HU005	**Usuario:** Usuario final comprador/vendedor
Nombre Historia: Implementación de la lógica de marketing C2C.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Iteración Asignada: 1	
Responsable: Roberth Pincha	
Descripción: El usuario final puede listar todos los productos publicados en base a las categorías o etiquetas, y añadirlos a su lista de preferencias o historial de compras. Además, el usuario puede publicar sus productos, editar su información, recibir notificaciones del estado del producto e inactivar productos.	
Observación: Se debe desarrollar los controladores y clases importantes para integrar la lógica del desarrollo del CRUD de los productos e implementación de la lógica de adición, reserva y compra de productos.	

TABLA IV: Historia de Usuario HU006

HISTORIA DE USUARIO	
Identificador (ID): HU006	**Usuario:** Usuario final
Nombre Historia: Visualización del perfil de usuario	
Prioridad en Negocio: Medio	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 2	
Responsable: Roberth Pincha	
Descripción: El usuario final podrá ver y realizar una modificación de sus datos más importantes dentro de las configuraciones de perfil.	
Observación: El usuario final cuando inicio sesión puede gestionar su perfil de usuario a través de un interfaz de visualización de su perfil y de otra interfaz para la edición de sus datos importantes.	

TABLA V: Historia de Usuario HU007

HISTORIA DE USUARIO	
Identificador (ID): HU007	**Usuario:** Usuario final
Nombre Historia: Visualización de otros perfiles de usuario	
Prioridad en Negocio: Media	**Riesgo en Desarrollo: Bajo**
Iteración Asignada: 2	
Responsable: Roberth Pincha	
Descripción: El usuario final, puede visualizar los datos más importantes de otros usuarios registrados dentro de la aplicación web.	
Observación: El usuario final podrá visualizar la información de otro usuario, a través de una vista donde se exponen los datos personales y datos de su reputación como vendedor.	

TABLA VI: Historia de Usuario HU008

HISTORIA DE USUARIO	
Identificador (ID): HU008	**Usuario:** Usuario final.
Nombre Historia: Visualización de los productos.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 2	
Responsable: Roberth Pincha	
Descripción: Como usuario final, puedo visualizar la información de cada producto mediante una interfaz gráfica. • Información del producto a comprar. • Información del producto a vender.	
Observación: El usuario podrá observar toda la información necesaria para realizar la compra o venta de un producto en específico.	

TABLA VII: Historia de Usuario HU009

HISTORIA DE USUARIO	
Identificador (ID): HU009	**Usuario:** Usuario final.
Nombre Historia: Visualización del CRUD de Productos a través de módulos.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 2	
Responsable: Roberth Pincha	
Descripción: Como usuario final vendedor, puedo subir productos para la venta completando un formulario: • Imagen del producto. • Título • Descripción • Detalle. • Precio • Categoría. A su vez también puedo realizar gestión de mis productos a través de módulos que muestran información y acciones de edición o cambiar el estado de mi publicación.	
Observación: Todos los campos deberán ser llenados de manera obligatoria, caso contrario el usuario no podrá completar la publicación del producto. El usuario contara con interfaces diferentes para administrar el contenido y estado de una de sus publicaciones.	

TABLA VIII: Historia de Usuario HU010

HISTORIA DE USUARIO	
Identificador (ID): HU010	**Usuario:** Usuario final.
Nombre Historia: Visualización de gestión de productos del carrito.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 2	
Responsable: Roberth Pincha	
Descripción: Como usuario final, puedo agregar y descartar productos del carrito de compras.	
Observación: En la interfaz del carrito de compras se podrá observar la información de cada producto como: nombre, precio y junto a botones para comprar o descartar productos.	

TABLA IX: Historia de Usuario HU011

HISTORIA DE USUARIO	
Identificador (ID): HU011	**Usuario:** Usuario final.
Nombre Historia: Visualización del historial de compras	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 2	
Responsable: Roberth Pincha	
Descripción: Como usuario final, puedo revisar los productos del historial de compras.	
Observación: En la interfaz del historial de compras se podrá observar la información de cada producto como: precio, información del vendedor y detalle.	

TABLA X: Historia de Usuario HU012

HISTORIA DE USUARIO	
Identificador (ID): HU012	**Usuario:** Usuario final.
Nombre Historia: Visualización del historial de ventas	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 2	
Responsable: Roberth Pincha	
Descripción: Como usuario final comprador y vendedor, puedo revisar los productos del historial de ventas.	
Observación: En la interfaz del historial de ventas se podrá observar la información de cada producto como: precio, detalle e información del cliente que compro.	

TABLA XI: Historia de Usuario HU013

HISTORIA DE USUARIO	
Identificador (ID): HU013	**Usuario:** Usuario final
Nombre Historia: Interfaces externas	
Prioridad en Negocio: Alto	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 3	
Responsable: Roberth Pincha	
Descripción: El sistema de tipo *Backend*, debe generar rutas de tipo *API REST*.	
Observación: El componente *Backend* deberá generar rutas de tipo *API REST* para poder compartir los datos, y de esta forma el componente *Frontend* podrá recuperar los datos sin mayor problema.	

TABLA XII: Historia de Usuario HU014

HISTORIA DE USUARIO	
Identificador (ID): HU014	**Usuario:** Administrador
Nombre Historia: Pruebas unitarias del *Backend*	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Iteración Asignada: 4	
Responsable: Roberth Pincha	
Descripción: Se realizará pruebas unitarias de todos los *resolvers* del proyecto usando herramientas que permiten crear peticiones sobre rutas API's.	
Observación: Se realizará pruebas unitarias asíncronas de los *resolvers,* usando la herramienta de Postman.	

TABLA XIII: Historia de Usuario HU015

HISTORIA DE USUARIO	
Identificador (ID): HU015	**Usuario:** Administrador
Nombre Historia: Despliegue a producción del *Backend*.	
Prioridad en Negocio: Alto	**Riesgo en Desarrollo:** Alto
Iteración Asignada: 5	
Responsable: Roberth Pincha	
Descripción: Subir los datos a AlwaysData y el *Backend* a Heroku.	
Observación: La base de datos que se usará es una DBaaS de esta forma los servicios de nuestro proveedor serán los mejores puesto que nos ofrece seguridad, mantebilidad, accesibilidad y contabilidad. Se desplegará el proyecto a producción para que esté disponible en internet y pueda estar disponibles las *API REST* para el componente *Frontend*.	

3. PRODUCT BACKLOG

En esta sección, la **TABLA XIX** se mostrará el *Product Backlog* del componente *Backend*.

TABLA XIV: *Product Backlog*

ELABORACIÓN DEL PRODUCT BACKLOG				
ID –HU	**HISTORIA DE USUARIO**	**ITERACIÓN**	**ESTADO**	**PRIORIDAD**
HU000	Configuraciones Iniciales.	0	**Finalizado**	**Alto**
HU001	Autenticar usuarios.	1	**Finalizado**	**Alto**
HU002	Registrar usuarios.	1	**Finalizado**	**Medio**
HU003	Actualizar datos del usuario.	1	**Finalizado**	**Medio**
HU004	Implementación del CRUD de las Categorías y Etiquetas.	1	**Finalizado**	**Medio**
HU005	Implementación de la lógica de marketing C2C.	1	**Finalizado**	**Alto**
HU006	Visualización del perfil de usuario.	2	**Finalizado**	**Bajo**
HU07	Visualización de otros perfiles.	2	**Finalizado**	**Bajo**
HU08	Visualización de los productos.	2	**Finalizado**	**Medio**
HU09	Visualización del CRUD de Productos a través de módulos.	2	**Finalizado**	**Medio**
HU010	Visualización de gestión de productos del carrito.	2	**Finalizado**	**Medio**
HU011	Visualización del historial de compras	2	**Finalizado**	**Medio**
HU012	Visualización del historial de ventas	2	**Finalizado**	**Medio**
HU013	Interfaces externas	3	**Finalizado**	**Alto**
HU014	Pruebas unitarias del *Backend*	4	**Finalizado**	**Alto**
HU015	Despliegue a producción del *Backend*	5	**Finalizado**	**Alto**

4. SPRINT BACKLOG

En esta sección, la **TABLA XX** se mostrará el *Sprint Backlog* del componente *Backend.*

TABLA XV: *Sprint Backlog*

ELABORACIÓN DEL SPRINT BACKLOG					
ID-SB	**NOMBRE**	**ID-HU**	**HISTORIA DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
SB00	Configuraciones Iniciales	HU000	Configuraciones Iniciales	• Recopilación de los requerimientos. • Diseño y creación de la base de datos. • Instalación y actualización de herramientas y *frameworks* • Producir las entidades y relaciones a nivel de base de datos y *Eloquent* en el proyecto. • Llenar con datos *Fakes* la base de datos.	60 H
		HU001	Autenticar usuarios	• Elaboración de la lógica de autenticación usando controladores para el modelo de Usuarios. • Elaboración de ruta de inicio de sesión de usuarios. • Validación del usuario en la base de datos	60 H
		HU002	Registrar usuarios	• Implementación función de registro en controlador para modelo de Usuario. • Validación de campos requeridos. • Elaboración de ruta de registro de usuarios.	
		HU003	Actualizar datos del usuario	• Implementación de las funcionalidades para la edición de los datos en el controlador respectivo para actualizar el perfil del usuario.	

SB01	Diseño de la lógica del Usuario, Categorías, Etiquetas y Productos			• Validación de campos para la actualización de un usuario. • Elaboración de la función para el cambio de Imagen de perfil.	
		HU004	Implementación del CRUD de las Categorías y Etiquetas	• Implementar las funcionalidades de gestión de los datos de categorías y etiquetas al administrador.	
		HU005	Implementación de la lógica de marketing C2C.	• Implementación del CRUD de productos en su respectivo controlador. • Implementación de validación de campos para la actualización y/o creación de un producto. • Implementación de función para la recuperación de productos a través de categorías o etiquetas. • Implementación de la lógica de añadir a favoritos un producto • Implementación para reservar o comprar un producto. • Implementar la lógica de notificación tras la reserva o compra de los productos que le pertenecen a un usuario.	70 H
		HU006	Visualización del perfil de usuario	• Mostrar la información del usuario y opciones de personalización de datos.	25 H
		HU007	Visualización de otros perfiles	• Implementar un módulo para recuperar los datos y mostrar los datos de cualquier usuario.	
		HU008	Visualización de los productos	• Implementación de una interfaz gráfica para mostrar la información de los productos.	50 H

SB02	Módulos de presentación e interacción con la lógica del proyecto			• Implementación de la funcionalidad que permita visualizar los productos por categorías. • Implementación de filtración de productos por etiquetas.	
		HU009	Visualización del CRUD de Productos a través de módulos.	• Implementación de una interfaz para mostrar el proceso de creación de productos. • Implementar los módulos junto a las funciones de administración correspondientes para el producto.	
		HU010	Visualización de gestión de productos del carrito.	• Implementar una interfaz para inyectar las funcionalidades de un carrito de compras. • Implementar la administración de productos agregados al carrito.	
		HU011	Visualización del historial de compras	• Implementación de una interfaz y funcionalidades para presentar la lista de los productos comprados	
		HU012	Visualización del historial de ventas	• Implementación de una interfaz y funcionalidades para presentar la lista de los productos vendidos	
SB03	Interfaces externas	HU013	Interfaces externas	• El componente *Backend* deberá generar rutas de tipo *API REST*	30 H
SB04	Pruebas unitarias del *Backend*	HU014	Pruebas unitarias del *Backend*	• Pruebas Unitarias	30 H
SB05	Despliegue del componente *Backend.*	HU015	Despliegue a producción del *Backend*	• Usar el modelo DBaaS de AlwaysData. • Configuración de Heroku. • Desplegar el componente a producción.	20 H

5. INTERFACES

A continuación, se presentarán los prototipos de interfaces que recrean los módulos necesarios para trabajar con las funcionalidades únicas del sistema cuyo propósito es guiar a las aplicaciones clientes en la utilización de sus datos y métodos desarrollados. Las interfaces se presentan desde la **Fig. 1** hasta la **Fig. 8**.

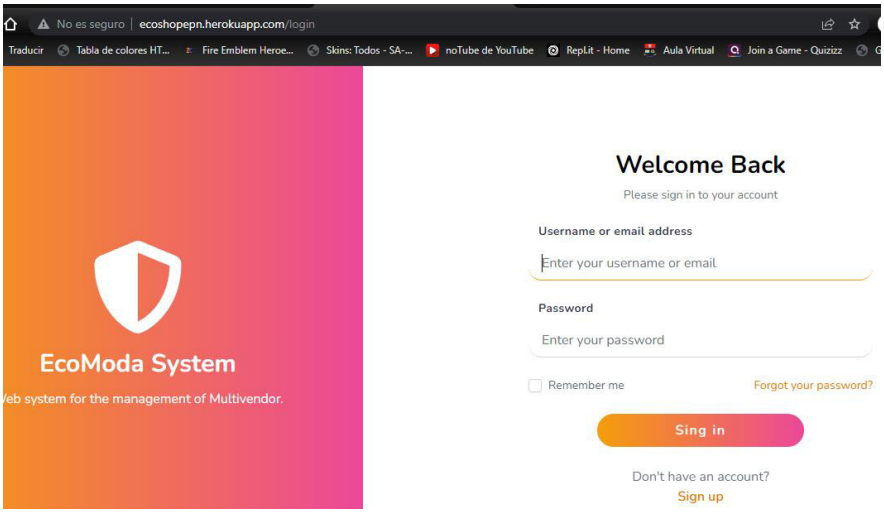


Fig. 1: Plantilla de inicio de sesión.

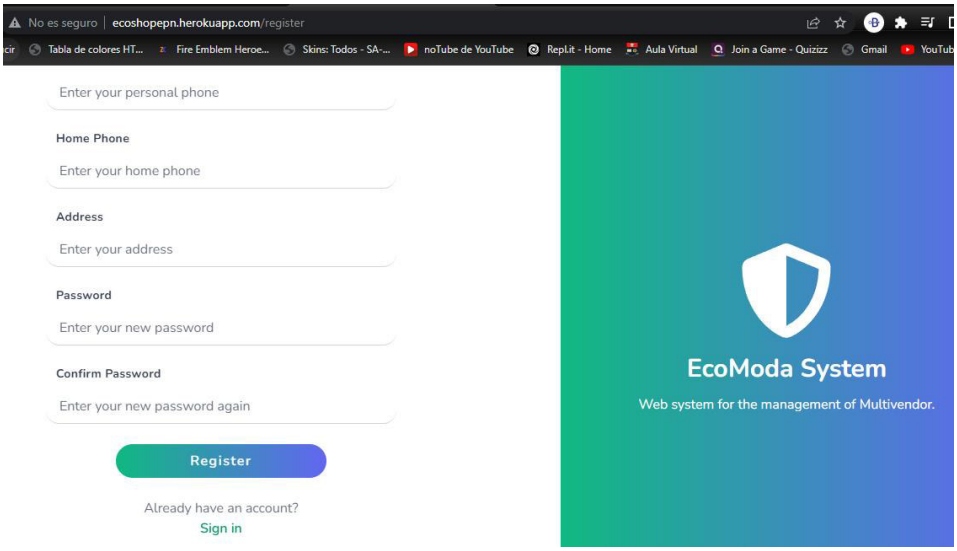


Fig. 2: Plantilla de Registro.

Fig. 3: Presentación de los productos organizados por Categorías.

Fig. 4: Presentación de los productos organizados por Etiquetas

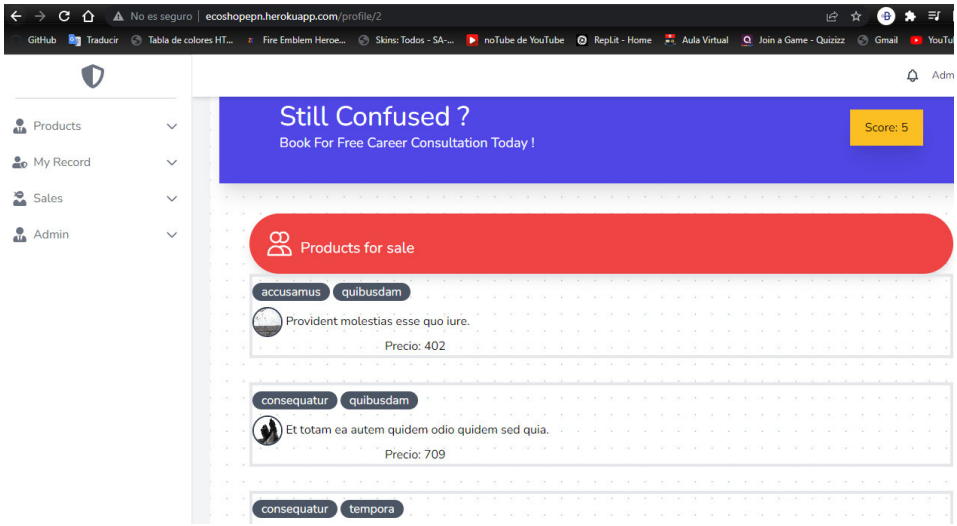


Fig. 5: Presentación del perfil de usuario junto a los productos que ha publicado.

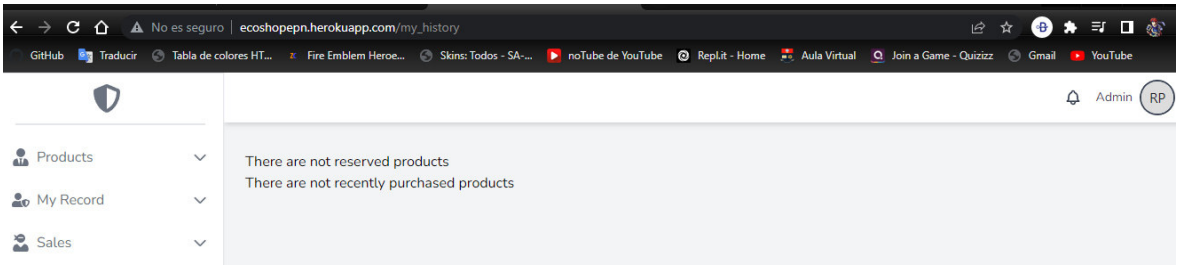


Fig. 6: Listado vacío de los productos reservados o comprados.

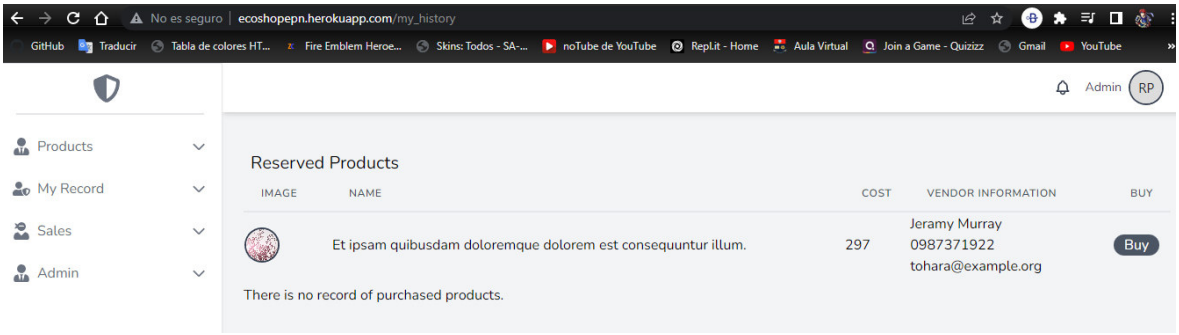


Fig. 7: Listado de los productos reservados junto al botón para finalizar la compra.

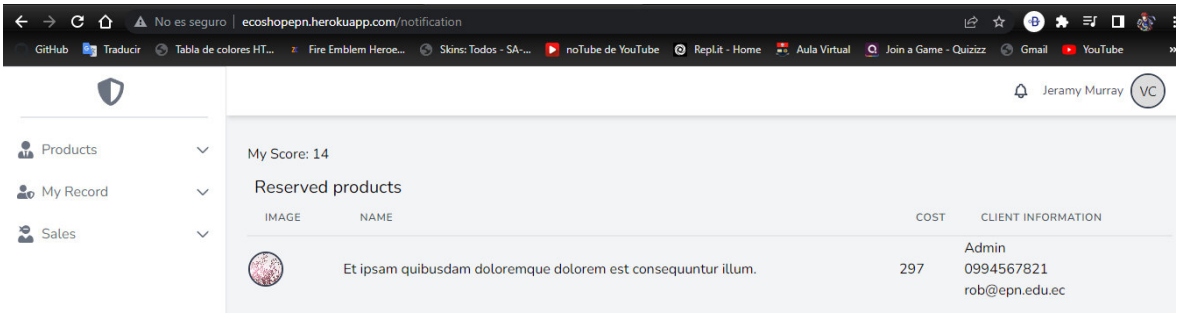


Fig. 8: Modulo de notificación con respecto al usuario que reservo el producto publicado.

6. PRUEBAS UNITARIAS

Con la finalidad de comprobar el funcionamiento de la lógica generada en el desarrollo del *Backend*, se toman algunas rutas generadas por el componente para verificar si el servicio *REST* funciona y cumple con los requerimientos. A continuación, se muestran las pruebas unitarias realizadas en el *Backend* de la aplicación.

Prueba 1:

Uso del token generado al iniciar sesión, en actividades como actualización de datos personales y notificaciones del usuario.

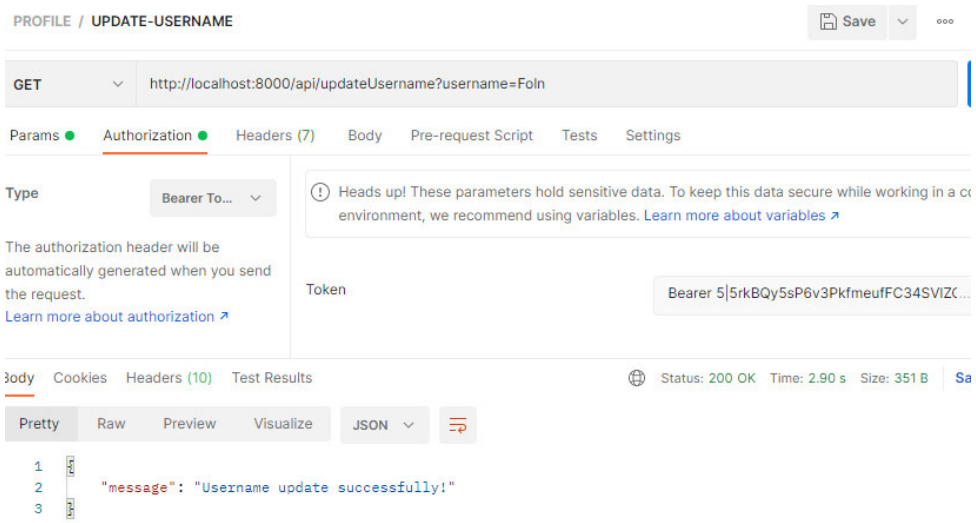


Fig. 9: Petición HTTP – Actualización del nombre de usuario.

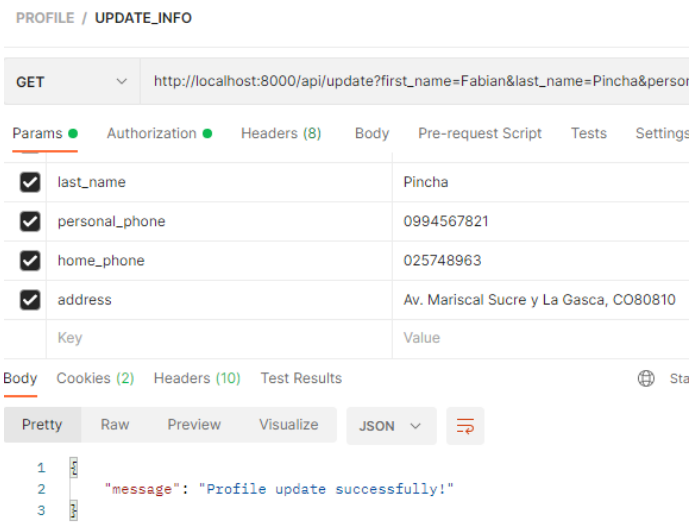


Fig. 10: Petición HTTP – Actualización de datos personales

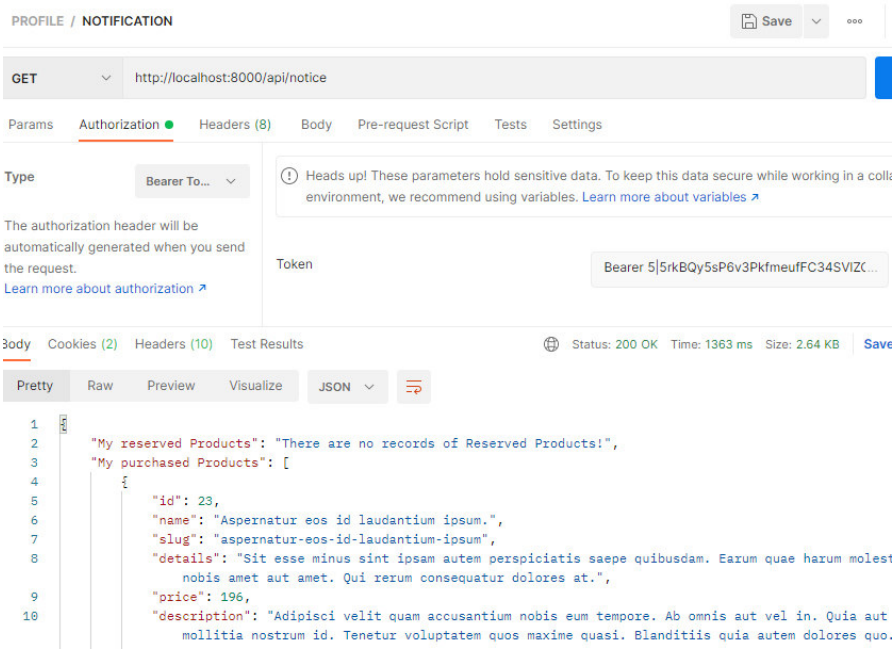


Fig. 11: Petición HTTP – Notificaciones del usuario con respecto a la situación de sus publicaciones.

Prueba 2:

Obtención de registros publicados y organizados por categorías y etiquetas.

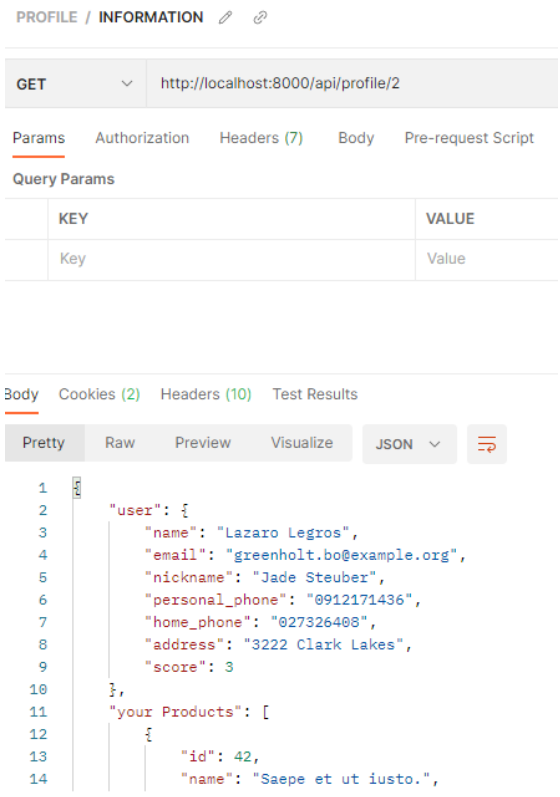


Fig. 12: Petición HTTP – Datos de un perfil de usuario determinado.

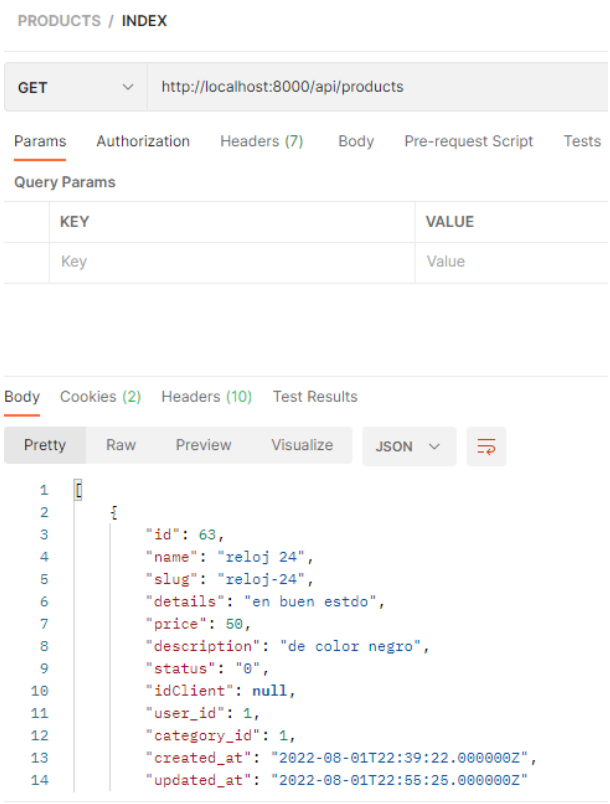


Fig. 13: Petición HTTP – Lista de las publicaciones.

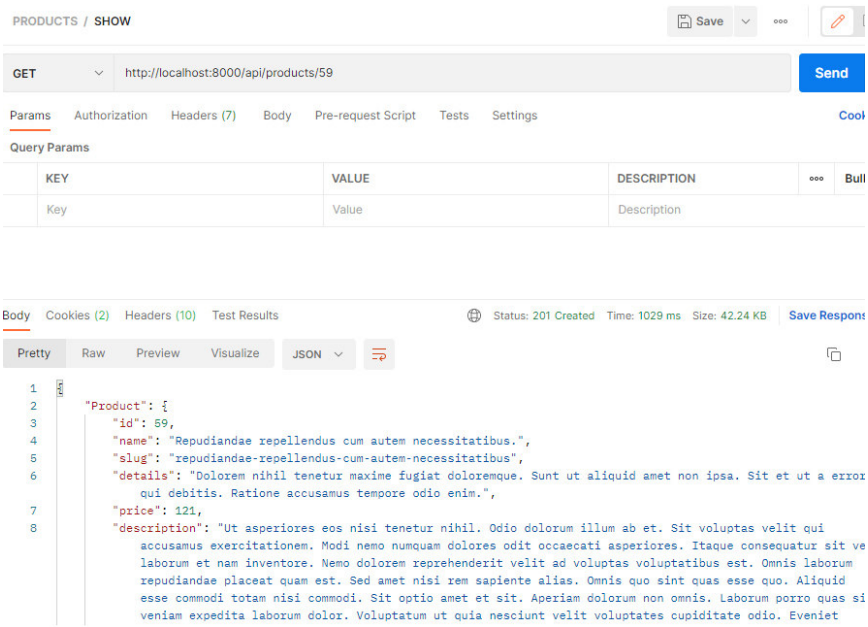


Fig. 14: Petición HTTP – Datos de una publicación.

PRODUCTS / CATEGORY-R

GET http://localhost:8000/api/category/3

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Body Cookies (2) Headers (10) Test Results Status: 201 C

Pretty Raw Preview Visualize JSON


```
{
    "Category": {
        "id": 3,
        "name": "quidem",
        "slug": "quidem",
        "created_at": "2022-07-25T00:01:36.000000Z",
        "updated_at": "2022-07-25T00:01:36.000000Z"
    },
    "Products": [
        {
            "id": 56,
            "name": "Amet nam vero minus accusantium.",
            "slug": "amet-nam-vero-minus-accusantium",
            "details": "Deserunt reprehenderit nostrum debitis qui aut. Mo
                quis. Fugiat dolorem laborum aut earum quia ex sit.",
            "price": 342,
            "description": "Aut omnis illum earum consequatur illum cum. F
                Animi et voluptates aut consequuntur impedit est. Ab moles
```


Fig. 15: Petición HTTP – Lista de productos por categoría.

PRODUCTS / TAG-R

GET http://localhost:8000/api/tag/1

Params Authorization Headers (7) Body Pre-request Script Tests Setting

Query Params

KEY	VALUE
Key	Value

Body Cookies (2) Headers (10) Test Results Status

Pretty Raw Preview Visualize JSON


```
{
    "Tag": {
        "id": 1,
        "name": "animi",
        "slug": "animi",
        "created_at": "2022-07-25T00:01:36.000000Z",
        "updated_at": "2022-07-25T00:01:36.000000Z"
    },
    "Products": [
        {
            "id": 50,
            "name": "Optio ad accusamus ipsum.",
            "slug": "optio-ad-accusamus-ipsum",
            "details": "Animi ut cumque ullam consequuntur. Qui quis
                sed aut ullam. Atque et quidem velit non neque.",
            "price": 889,
```


Fig. 16: Petición HTTP – Lista de los productos ordenados por etiquetas.

Prueba 3:

Uso del token generado al iniciar sesión, en actividades como añadir o descartar del carrito de compras y mostrar la lista.

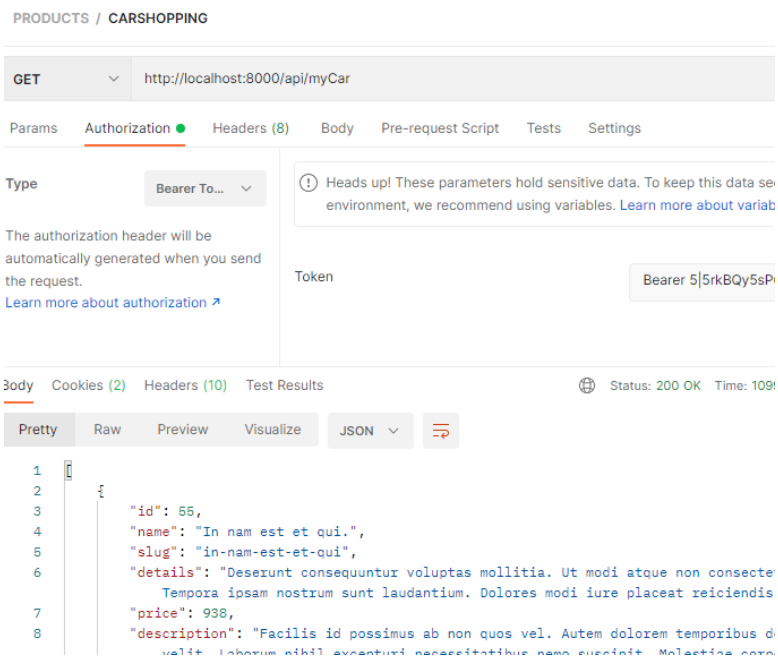


Fig. 17: Petición HTTP – Lista de los productos añadidos al carrito de compras.

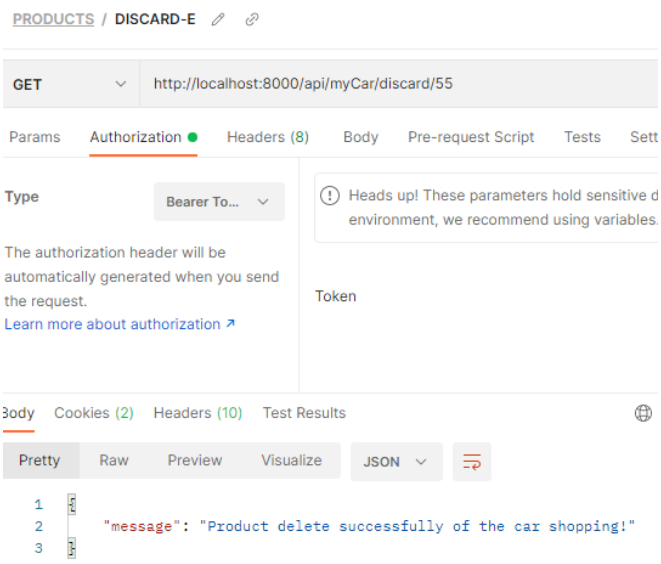


Fig. 18: Petición HTTP – Realización del descarte de algún producto añadido a favoritos.

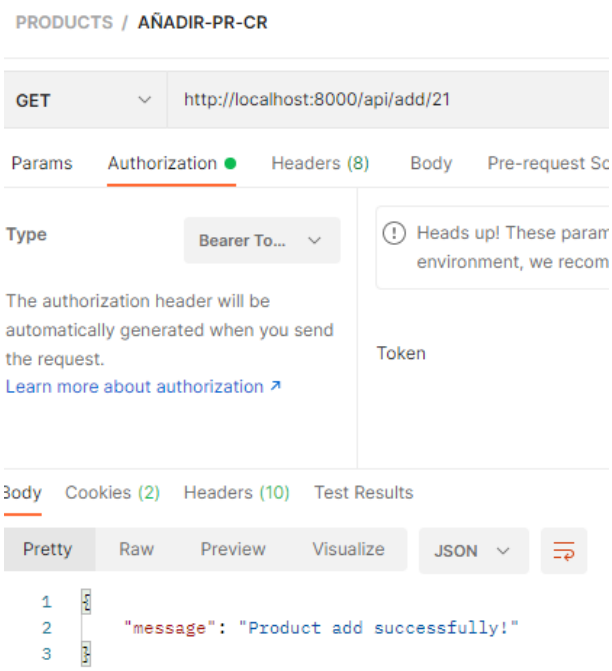


Fig. 19: Petición HTTP – Añadir un producto al listado de favoritos.

Prueba 4:

Uso del token generado al iniciar sesión, en actividades como reservar un producto, mostrar el historial de compras y calificar al usuario.

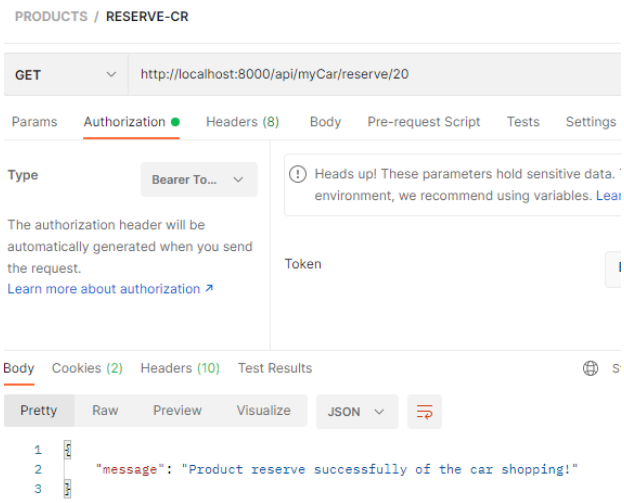


Fig. 20: Petición HTTP – Realización de la reserva de un producto.

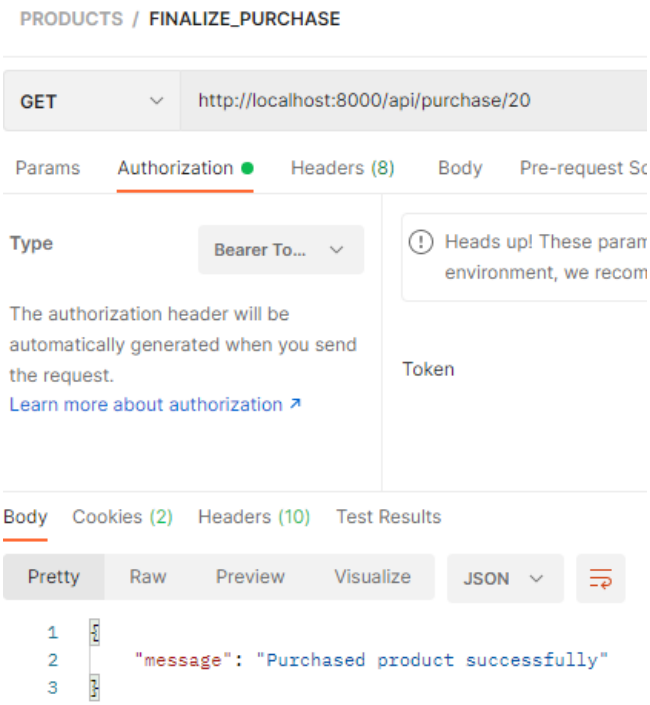


Fig. 21: Petición HTTP – Confirmación de la compra de un producto.

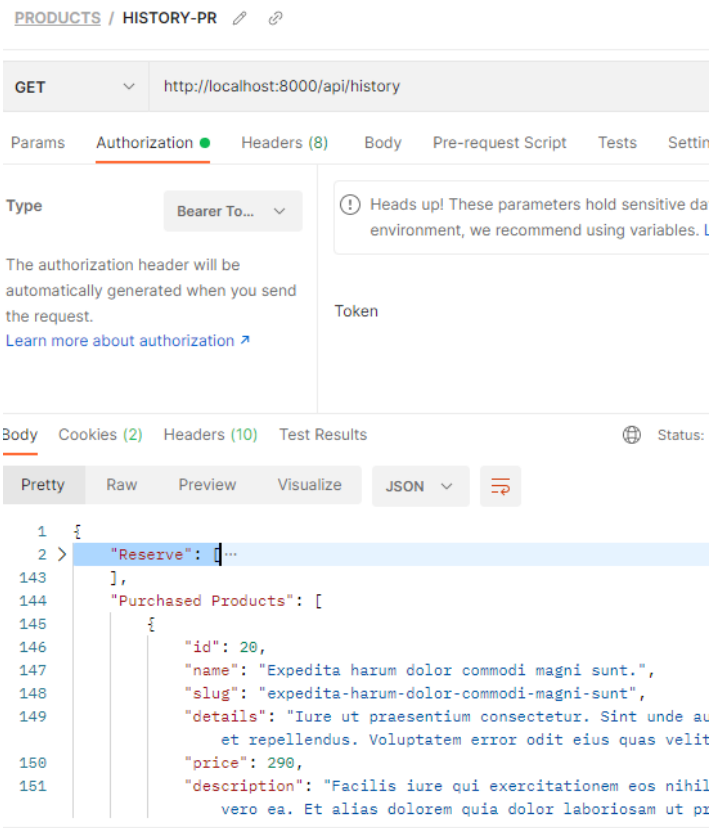


Fig. 22: Petición HTTP – Historial de compras y reservas de productos.

Prueba 5:

Uso del token generado al iniciar sesión, en actividades como el cierre de sesión.

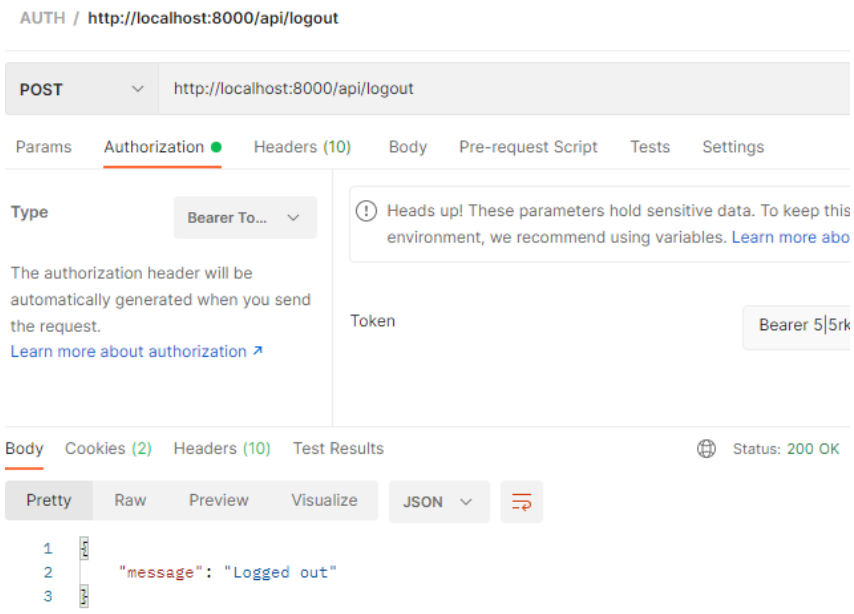


Fig. 23: Petición HTTP – Cierre de sesión.

Prueba 6:

Uso del token generado al iniciar sesión con el sistema en produccion, en actividades como crear una publicacion.

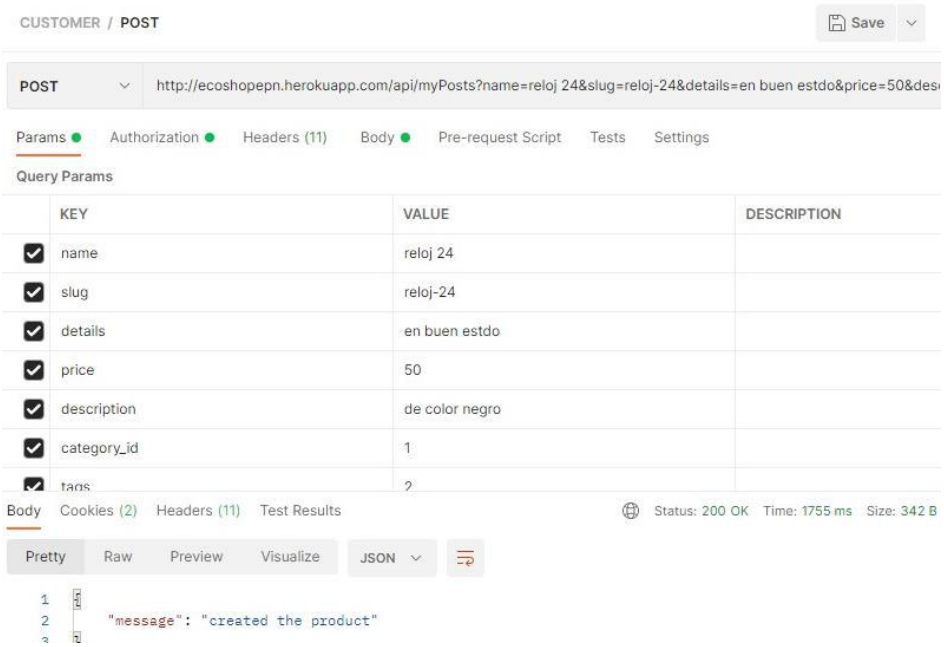


Fig. 24: Petición HTTP – Creación de una publicación con api en producción.

ANEXO III Manual de Usuario

El siguiente video corresponde al manual de usuario del sistema que puede ser accedido mediante el siguiente enlace: https://www.youtube.com/watch?v=pjMmb1DQ2i0

ANEXO IV Manual de instalación

El componente *Backend* generado durante el desarrollo de este proyecto, se encuentra alojado en la plataforma de *Github* junto a los pasos necesarios para replicar este proyecto. A continuación, se comparte el enlace donde se encuentra el código fuente del componente *Backend*.

https://github.com/Roblink-21/Aplicacion_2daMano_EcoModa/tree/sprint3