

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

**DESARROLLO DE SISTEMA DE GESTIÓN DE VENTA DE
LICORES PARA LA LICORERÍA "LA NENITA"**

DESARROLLO DE UN BACKEND

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

JOSELYN GABRIELA CORREA FIGUEROA
joselyn.correa@epn.edu.ec

DIRECTOR: BYRON GUSTAVO LOARTE CAJAMARCA
byron.loarteb@epn.edu.ec

Quito, agosto 2023

CERTIFICACIONES

Yo, **JOSELYN GABRIELA CORREA FIGUEROA** 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.

JOSELYN GABRIELA CORREA FIGUEROA

joselyn.correa@epn.edu.ec

Certifico que el presente trabajo de integración curricular fue desarrollado por **JOSELYN GABRIELA CORREA FIGUEROA**, bajo mi supervisión.

Ing. BYRON LOARTE, MSc.
DIRECTOR

byron.loarteb@epn.edu.ec

DECLARACIÓN DE AUTORÍA

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

JOSELYN GABRIELA CORREA FIGUEROA

DEDICATORIA

El presente proyecto y el esfuerzo invertido durante la carrera se lo dedico en primera instancia a mi madre y hermanos, quienes han sido las personas que me han motivado a cumplir mis sueños y lograr cada una de mis metas. A mi amado Abuelito Oswaldo Correa, quien desde el cielo me guía y me llena de fuerzas para no rendirme y continuar luchando cada día por ser una excelente profesional. A mi compañero de vida Geovanny, por su guía, paciencia y por compartir su conocimiento y experiencia en este arduo camino, siendo él mi fuente de inspiración y fortaleza para alcanzar este objetivo. A mis amigos y compañeros de estudio, por su amistad y por demostrar que este camino no se hubiese podido lograr sin su amistad. Cada uno de ustedes ha sido fundamental en mi crecimiento y éxito, y les estoy profundamente agradecido.

JOSELYN GABRIELA CORREA FIGUEROA

AGRADECIMIENTO

Deseo expresar mi más sincero agradecimiento a todas las personas que han desempeñado un papel fundamental en este arduo camino. En primer lugar, quiero extender mi más profundo reconocimiento a mi familia y amigos por su constante apoyo y motivación. A mi compañero, quien ha estado a mi lado día tras día, brindándome apoyo inquebrantable hasta alcanzar nuestro objetivo. También, quiero mencionar a mis profesores, cuya guía y respaldo han sido invaluableles en diferentes etapas de mi formación.

JOSELYN GABRIELA CORREA FIGUEROA

CONTENIDO

CERTIFICACIONES.....	I
DECLARACIÓN DE AUTORÍA.....	II
DEDICATORIA.....	III
AGRADECIMIENTO.....	IV
CONTENIDO.....	V
RESUMEN	VII
ABSTRACT	VIII
1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO	1
1.1 Objetivo general	2
1.2 Objetivos específicos.....	2
1.3 Alcance.....	2
1.4 Marco Teórico.....	4
2 METODOLOGÍA	7
2.1 Metodología de Desarrollo.....	7
Roles	8
Artefactos	9
2.2 Diseño de la arquitectura	12
Patrón arquitectónico.....	12
2.3 Herramientas de desarrollo.....	13
Librerías	14
3 RESULTADOS	16
3.1 <i>Sprint 0</i> . Configuración del ambiente de desarrollo.....	16
Definición de requerimientos y limitaciones	16
Creación de la estructura del proyecto <i>backend</i>	19
Diseño y desarrollo de la Base de datos SQL.....	19
Definición y asignación de roles de usuario	20
3.2 <i>Sprint 1</i> . Resultados del desarrollo de <i>endpoints</i> para el perfil administrador.	21
Generar <i>endpoints</i> que permiten iniciar sesión, cerrar sesión y modificar contraseña.....	22
Generar <i>endpoints</i> que permiten visualizar y modificar el perfil de usuario	22
Generar <i>endpoints</i> que permiten gestionar productos.....	23
Generar <i>endpoints</i> que permiten gestionar categorías.....	24
Generar <i>endpoints</i> que permiten gestionar subcategorías.....	25

Generar <i>endpoints</i> que permiten gestionar pedidos.....	26
Generar <i>endpoints</i> que permiten visualizar estado de pedidos	27
Generar <i>endpoints</i> que permiten gestionar comentarios y/o sugerencias	28
Generar <i>endpoints</i> que permiten visualizar detalle de pedido	29
3.3 <i>Sprint 2. Resultados del desarrollo de endpoints para el perfil empleado.</i>	30
Generar <i>endpoints</i> para seleccionar el pedido como entregado	30
3.4 <i>Sprint 3. Resultados del Desarrollo de endpoints para el perfil cliente.</i>	31
Gestionar <i>endpoints</i> que permiten registrarse	31
Gestionar <i>endpoints</i> que permiten gestionar carrito de compras	33
Gestionar <i>endpoints</i> que permiten gestionar pedido	34
Gestionar <i>endpoints</i> que permiten visualizar el historial de pedidos	35
Gestionar <i>endpoints</i> que permiten enviar comentarios y/o sugerencias.....	36
3.5 <i>Sprint 4. Pruebas para el componente backend</i>	37
Ejecución de pruebas unitarias	37
Ejecución de pruebas de compatibilidad	38
Ejecución de pruebas de estrés.....	39
Ejecución de pruebas de aceptación	39
3.6 <i>Sprint 5. Despliegue del componente backend.....</i>	40
Despliegue del <i>backend</i> en Linode.	40
4 CONCLUSIONES	42
5 RECOMENDACIONES.....	43
6 REFERENCIAS BIBLIOGRÁFICAS	44

RESUMEN

Licorería “La Nenita”, es un establecimiento físico con más de tres años de experiencia en la venta de bebidas alcohólicas ubicado en Santa Rosa de Cuzubamba. En el cual se identificó que el proceso de pedidos se llevaba a cabo a través de redes sociales, llamadas telefónicas y anotaciones de forma manual lo que resultaba ineficiente debido a la limitada disponibilidad del administrador y el riesgo de pérdida y duplicación de información. Además, se destacó la ausencia de un catálogo virtual para mostrar su amplia variedad de productos hacia sus clientes.

Para dar una solución tecnológica a la problemática mencionada anteriormente, el presente Trabajo de Integración Curricular se centra en el desarrollo y despliegue de un *backend* para la gestión de ventas de licores para la licorería "La Nenita". Esta solución del lado del servidor se conecta con un *frontend* ya existente, brindando de esta manera una sólida infraestructura que les permita a los clientes navegar y realizar compras a través de un catálogo virtual y un carrito de compras. Gracias a esta integración, el administrador obtiene un mejor control del inventario y una gestión más eficiente de los pedidos, lo que aumenta la competitividad y el posicionamiento del negocio en el mercado. Además, se mejora la experiencia del cliente, ya que pueden realizar sus pedidos desde la comodidad de su hogar, brindando una experiencia de compra más agradable.

La estructura de este documento se organiza de la siguiente manera: en primer lugar, se identifica la problemática relacionada con el negocio y, en función de ello, se establecen los objetivos, alcance y el marco teórico correspondiente. A continuación, se detalla la implementación de la metodología que se ha utilizado, en este caso, *Scrum*, junto con el diseño de la base de datos y su interacción con el *Framework FastAPI* y la Base de datos PostgreSQL. Además, se describe el uso de herramientas y librerías relevantes para el desarrollo del *backend*. Posteriormente, se proporciona una descripción detallada de cada una de las tareas que se han desarrollado en cada uno de los módulos, así como los resultados que se han obtenido en cada iteración (*Sprint*). Finalmente, se presentan las conclusiones y recomendaciones que se han obtenido a lo largo del proceso de desarrollo y despliegue a producción del presente proyecto.

PALABRAS CLAVE: Backend, Frontend, Scrum, Python, FastAPI, Framework, PostgreSQL, Sprint.

ABSTRACT

Liquor Store 'La Nenita' is a physical establishment with over three years of experience in the sale of alcoholic beverages located in Santa Rosa de Cuzubamba. It was identified that the order process was carried out through social media, phone calls, and manual note-taking, resulting in inefficiency due to limited administrator availability and the risk of information loss and duplication. Additionally, the absence of a virtual catalog to showcase their wide range of products to customers was highlighted.

To provide a technological solution to the aforementioned issue, this Curricular Integration Work focuses on the development and deployment of a backend for liquor sales management for 'La Nenita' liquor store. This server-side solution interfaces with an existing frontend, thus providing a robust infrastructure that enables customers to browse and make purchases through a virtual catalog and a shopping cart. Through this integration, the administrator gains better inventory control and more efficient order management, enhancing business competitiveness and market positioning. Furthermore, the customer experience is improved, as they can place orders from the comfort of their homes, providing a more enjoyable shopping experience.

The structure of this document is organized as follows: first, the business-related issue is identified, and based on that, the objectives, scope, and relevant theoretical framework are established. Next, the implementation of the methodology used, in this case, Scrum, is detailed along with the design of the database and its interaction with the FastAPI Framework and PostgreSQL database. Moreover, the use of relevant tools and libraries for backend development is described. Subsequently, a detailed description of each task developed in each module is provided, along with the results obtained in each iteration (Sprint). Finally, the conclusions and recommendations drawn throughout the development and production deployment process of this project are presented.

Keywords: Backend, Frontend, Scrum, Python, FastAPI, Framework, PostgreSQL, Sprint.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

La licorería "La Nenita" es un negocio físico ubicado en Santa Rosa de Cuzubamba, el cual cuenta con más de 3 años de experiencia especializada en la venta de diversas categorías de bebidas alcohólicas a clientes adultos de la zona [1]. Actualmente, los pedidos de la licorería se gestionan de varias maneras: los clientes pueden ingresar al local, utilizar aplicaciones de mensajería instantánea o realizar llamadas telefónicas para luego recibir la entrega en el lugar acordado. Sin embargo, esta forma de comunicación puede resultar ineficiente debido a que la mayoría de los clientes tienen que acercarse personalmente al establecimiento y a la disponibilidad limitada del administrador para atender llamadas. Esto impacta negativamente en la reputación del negocio y en la experiencia de sus futuros clientes. Además, todos los pedidos y el inventario se registran en un cuaderno, lo que conlleva a una serie de problemas, como pérdida y duplicidad de información, falta de un catálogo virtual, pérdida de clientes debido a la falta de atención telefónica, entre otras.

En la actualidad, la falta de una plataforma virtual puede ser una desventaja importante para cualquier negocio, ya que limita su alcance y capacidad de llegar a nuevos clientes. Especialmente en esta nueva era digital y con el auge de las compras en línea, contar con una plataforma virtual se vuelve cada vez más necesario para mantenerse competitivo y adaptarse a las necesidades cambiantes de los consumidores [2]. Así mismo, es de suma importancia disponer de una plataforma virtual, ya que ofrece de una amplia variedad de productos disponibles de manera digital, lo que facilita su adquisición de forma sencilla, rápida y eficiente todo desde la comodidad del hogar.

Tanto propietarios como clientes se benefician cada día más de estas plataformas virtuales y de las experiencias memorables que ofrecen, ya que las mismas están disponibles las 24 horas del día y los 365 días del año, lo que permite realizar otras actividades mientras se atiende virtualmente [3]. Además, estas plataformas facilitan el análisis de datos para comprender los hábitos de consumo de los clientes y permiten llegar a un público más amplio y diverso, expandiendo el alcance geográfico y permitiendo el crecimiento escalable del negocio [4].

Por este motivo, el presente Trabajo de Integración Curricular considera el desarrollo, pruebas e implementación en producción de un *backend* para la gestión de ventas de licores para la licorería "La Nenita". Adicional a ello, este componente cuenta con varios *endpoints* que aseguran que la información sea fácilmente accesible por una aplicación del lado del cliente, móvil o la integración de otras plataformas externas. Además, se han establecido diferentes roles para garantizar la seguridad y el acceso adecuado a la

información del negocio, garantizando de esta manera un mayor número de ventas y competitividad en el mercado actual gracias a la utilización de tecnologías modernas.

1.1 Objetivo general

Desarrollar un *backend* para la gestión de venta de licores para licorería "LA NENITA".

1.2 Objetivos específicos

1. Levantar los requisitos fundamentales para el desarrollo del *backend*.
2. Diseñar la arquitectura del *backend* según la recopilación de requerimientos.
3. Diseñar el modelo de Base de datos relacional según la recopilación de requerimientos.
4. Codificar los *endpoints* y módulos para el *backend* según los roles.
5. Verificar la correcta funcionalidad de los *endpoints* en base a una serie de pruebas.
6. Desplegar el *backend* a un ambiente en la nube.

1.3 Alcance

En la actualidad, gracias al comercio electrónico es posible realizar todo tipo de transacciones desde la comodidad del hogar. Las compañías tienen la capacidad de establecer tiendas en línea que proporcionan una gran variedad de productos para que los clientes elijan los que más se adapten a sus necesidades, realicen pagos de manera segura y reciban la entrega de sus pedidos en la puerta de su domicilio [5]. Contar con una tienda virtual brinda a la licorería "La Nenita" la oportunidad de expandirse, atraer nuevos clientes y mejorar su posicionamiento en el mercado [6].

En las diversas áreas de la programación, el desarrollo de un *backend* es fundamental en la construcción de aplicaciones y sistemas *web*, ya que se encarga de implementar toda la lógica y funcionalidad de un sistema *software*. Además, se encarga de la administración, gestión y almacenamiento de la información, establecimiento de reglas del negocio, protocolos de comunicación, reglas de seguridad, entre otras tareas. Por último, existen diversos lenguajes de programación, librerías y *Frameworks* que permiten el desarrollo de un *backend* entre los que destacan Python, Java, Ruby, PHP, etc. [7].

El presente proyecto propone el desarrollo, pruebas e implementación a producción de un *backend* para la gestión de ventas de licores para la licorería "La Nenita". Esto permitirá a los clientes realizar compras de manera segura, rápida y sencilla sin salir de su hogar u

oficina, con entregas en un plazo de tiempo razonable. Mientras que, el administrador va a poder gestionar de forma efectiva cada uno de los pedidos, inventarios e información estratégica relevante al negocio. En cuanto a la etapa de desarrollo, se utiliza una metodología de desarrollo de *software* ágil, un patrón arquitectónico, una base de datos relacional, herramientas y bibliotecas para la codificación, una serie de pruebas y la fase de implementación en producción, respectivamente. Adicional, existen 3 perfiles de usuario, cada uno con permisos diferentes para visualizar los distintos módulos como se presenta a continuación:

Perfiles que se establecen: Administrador, Empleado y Cliente.

Para el perfil administrador se generan:

- *Endpoints* que permiten iniciar sesión, cerrar sesión y modificar contraseña.
- *Endpoints* que permiten visualizar y modificar el perfil de usuario.
- *Endpoints* que permiten gestionar categorías.
- *Endpoints* que permiten gestionar subcategorías.
- *Endpoints* que permiten gestionar productos.
- *Endpoints* que permiten gestionar pedidos.
- *Endpoints* que permiten visualizar estado pedidos.
- *Endpoints* que permiten gestionar comentarios y/o sugerencias.

Para el perfil empleado se generan:

- *Endpoints* que permiten iniciar sesión, cerrar sesión y modificar contraseña.
- *Endpoints* que permiten visualizar y modificar el perfil de usuario.
- *Endpoints* que permiten gestionar pedidos.
- *Endpoints* que permiten seleccionar el pedido como entregado.
- *Endpoints* que permiten visualizar estado de pedidos.

Para el perfil cliente se generan:

- *Endpoints* que permiten iniciar sesión, cerrar sesión y modificar contraseña.
- *Endpoints* que permiten visualizar y modificar el perfil de usuario.

- *Endpoints* que permiten visualizar favoritos.
- *Endpoints* que permiten carrito de compras.
- *Endpoints* que permiten gestionar pedido.
- *Endpoints* que permiten visualizar detalle de pedido.
- *Endpoints* que permiten visualizar el historial de pedidos.
- *Endpoints* que permiten enviar comentarios y/o sugerencias.
- *Endpoints* que permiten visualizar estado de pedidos.

1.4 Marco Teórico

Dentro del campo del *software* se engloban diversas actividades relacionadas con la informática, que consisten en una secuencia de instrucciones destinadas a guiar a una computadora en la ejecución de tareas específicas. Las cuales se dividen en: *software* de sistema, de aplicación y lenguajes de programación [8].

Desde el análisis de los requerimientos hasta el mantenimiento del producto final, el proceso de creación de *software* se fundamenta en diversas etapas. Estas incluyen una fase de análisis, en la cual se estudian y comprenden los requisitos del *software*, seguida del diseño, en la cual se define la estructura y la arquitectura del sistema *software*. A continuación, se lleva a cabo el desarrollo propiamente dicho, donde se codifican las instrucciones necesarias. Posteriormente, se realizan pruebas exhaustivas para verificar su correcto funcionamiento. Por último, se procede a la implementación del *software* y su posterior mantenimiento, asegurando su óptimo rendimiento a lo largo del tiempo [9].

Python es ampliamente utilizado como un lenguaje de programación en alta demanda, es conocido por su eficacia y facilidad de aprendizaje. Además, es un lenguaje de propósito general, multiplataforma e interpretado, lo que significa que puede ejecutarse en diferentes sistemas operativos sin necesidad de compilar el código. También es interactivo, lo que facilita la prueba y depuración del código. Por otra parte, es orientado a objetos, lo que permite una estructuración eficaz del código ya que ofrece una extensa variedad de funciones, librerías y gran flexibilidad en el desarrollo de programas complejos. Su sintaxis clara y legible ayuda a mejorar la comprensión y el mantenimiento del código [10].

FastAPI es un marco de trabajo moderno y altamente eficiente para desarrollar interfaces de programación de aplicaciones (API) utilizando Python 3.6+ como base. Se destaca por su facilidad de aprendizaje, lo que lo hace accesible incluso para aquellos que están

comenzando en el desarrollo de APIs. Además, su enfoque en el rendimiento lo convierte en una opción rápida y eficiente para implementar soluciones de producción de manera ágil. Además, ofrece una combinación de características que lo hacen un *Framework* potente y listo para ser utilizado en la creación y despliegue de APIs de tipo RESTful [11].

JSON al igual que XML, son estándares que se emplea en la programación *web* para el intercambio de datos entre diferentes aplicaciones, se caracteriza por su simplicidad y ligereza. Es fácil de leer y escribir para los seres humanos, al mismo tiempo que las máquinas pueden interpretarlo y generar datos de forma sencilla [12].

JSON Web Token (JWT) proporciona un método seguro y eficiente para transmitir información entre diferentes partes en formato de objetos JSON. Tanto para los seres humanos como para las máquinas, leer y escribir JWT resulta sencillo. Además, la información contenida en un JWT puede ser verificada y es confiable, ya que está firmada digitalmente, lo que asegura su integridad y autenticidad [13].

Una API RESTful se compone de reglas y protocolos que posibilitan la interacción y comunicación entre distintas aplicaciones y componentes de *software*. Se basa en los principios de REST, que promueven un enfoque uniforme y orientado a recursos para la transferencia de información a través de internet. Permite a los sistemas comunicarse y compartir información de manera eficiente y escalable. Utiliza los métodos HTTP como GET, POST, PUT, DELETE, HEAD, PATCH, para realizar operaciones en los recursos lo que facilita la creación, actualización, lectura y eliminación de datos [14].

Un ORM (Mapeo Objeto-Relacional) es un estilo de programación para bases de datos relacionales, que mapea las tablas de la base de datos en una serie de clases ofreciendo un enfoque eficiente para almacenar y gestionar datos [15]. Además, las tablas se relacionan entre sí mediante claves primarias y externas. Este enfoque ofrece estructura y coherencia en la manipulación de datos, permitiendo consultas y recuperación eficiente de información. Por otra parte, proporciona un alto nivel de integridad, consistencia y flexibilidad en la manipulación de datos ya que permite consultas complejas y eficientes a través del lenguaje de consulta estructurado (SQL) [16].

Linode es una destacada empresa que ofrece servicios de alojamiento en la nube. Proporciona servidores virtuales escalables y confiables en diversas ubicaciones globales. Con su enfoque en la simplicidad, seguridad y rendimiento, Linode ofrece una solución confiable para implementar y gestionar aplicaciones y sitios *web* en la nube. Además, cuenta con un panel de control intuitivo, opciones de configuración sumamente flexibles y

un sólido soporte técnico, lo que la convierte en una elección popular para empresas de diferentes tamaños [17].

Las pruebas en el desarrollo de un *backend* son cruciales para asegurar la calidad y el adecuado funcionamiento de un producto *software*. Los diferentes tipos de pruebas, como las unitarias, de integración, de rendimiento, de seguridad y de regresión, abordan aspectos clave del *backend*, como la funcionalidad, interacción entre componentes, el rendimiento, la protección contra vulnerabilidades y la estabilidad a lo largo del tiempo. Estas pruebas aseguran la confiabilidad del *backend* y brindan una base estable y sólida para la creación y desarrollo de aplicaciones y sistemas escalables con el tiempo [18].

2 METODOLOGÍA

El estudio de casos se utiliza para examinar de manera detallada un fenómeno o situación particular en su contexto real [19]. Consiste en un análisis profundo y exhaustivo de un caso específico, recolectando información a través de diversas fuentes de datos, como entrevistas, observaciones y documentos. Esto permite comprender, describir, identificar patrones, analizar relaciones causa-efecto y extraer lecciones prácticas. El estudio de casos es ampliamente utilizado en diversas disciplinas, como tecnología, psicología, educación, sociología y la administración, para comprender fenómenos complejos y establecer un cimiento robusto para la toma de decisiones y formulación de teorías [20].

Basándose en lo mencionado anteriormente, este proyecto de Integración Curricular adopta el estudio de casos, debido a que se realiza una investigación exhaustiva de las problemáticas que presentan las tiendas físicas en la venta de sus productos y su posicionamiento en el mercado. Como resultado, se lleva a cabo el desarrollo del *backend* para la gestión de pedidos en línea, lo cual mejora las ventas de la licorería y facilita el acceso de los clientes mediante el uso de tecnologías emergentes en el ámbito del desarrollo de *software*.

2.1 Metodología de Desarrollo

Se refiere a un conjunto de principios, enfoques, técnicas y prácticas que se utilizan para abordar un problema, llevar a cabo un proyecto o realizar una actividad. Además, proporciona un marco estructurado y sistemático que incluye los pasos secuenciales, herramientas, recursos recomendados, roles y responsabilidades específicos, así como criterios para evaluar el progreso de los resultados. El objetivo de esta metodología es minimizar el riesgo de fracaso y lograr un desarrollo de *software* de alta calidad. Se busca mejorar la organización, incrementar la eficiencia y asegurar la adopción de las mejores prácticas para alcanzar los resultados planificados. [21].

La metodología ágil es un enfoque flexible y colaborativo para el desarrollo de proyectos, donde se prioriza la adaptabilidad, comunicación y la entrega incremental. Se trabaja en ciclos cortos, conocidos como iteraciones o *Sprints* para responder rápidamente a los cambios y obtener retroalimentación temprana. Este enfoque permite una mayor flexibilidad, adaptabilidad y trabajo en equipo entre los integrantes del equipo y los *stakeholders*, mejorando la entrega de valor en cada etapa del proyecto [22].

Considerando lo anteriormente citado, el desarrollo de este componente utiliza la metodología *Scrum*, que forma parte del conjunto de metodologías ágiles. Además, la

elección de esta metodología ágil se basa en su enfoque colaborativo y su capacidad para fomentar un trabajo en equipo eficiente y efectivo. Con *Scrum*, se puede realizar entregas periódicas y parciales del trabajo final, lo que permite una mayor visibilidad del progreso y la capacidad de realizar ajustes en base a la retroalimentación recibida. Por otra parte, se establecen reuniones regulares con el cliente para evaluar el avance del proyecto y asegurar una alineación constante con los requisitos y expectativas [23]. A continuación, se detalla la ejecución de cada fase a lo largo del desarrollo del proyecto.

Roles

Scrum fundamenta el trabajo basado en roles, cada rol debe ser desempeñado con seriedad y compromiso con el fin de producir *software* de alta calidad [23]. A continuación, se describen los roles que forman parte del desarrollo del *backend*.

Product Owner

Esta persona tiene la responsabilidad de gestionar el *Product Backlog* y maximizar su valor de manera óptima. Además, su labor consiste en dar prioridad a las actividades basándose en la importancia y el riesgo que representa cada una de estas [24]. Por otra parte, la **Tabla 2.1** especifica la asignación de la persona con su respectivo rol en el desarrollo del *backend*.

Scrum Master

Esta persona tiene una habilidad para ser altamente organizado en un proyecto de *software*, ya que ayuda a recordar a los miembros del equipo sus tareas asignadas garantizando el cumplimiento de las reglas y prácticas de *Scrum*. Además, fomenta la colaboración y la mejora constante en el equipo de trabajo. [24]. En resumen, es una pieza clave en el funcionamiento efectivo del equipo, es por esta razón que, en la **Tabla 2.1** se especifica la asignación de personas con sus respectivos roles en el desarrollo del *backend*.

Development Team

Es responsabilidad del equipo de desarrollo elegir las tecnologías apropiadas con las que se va a llevar a cabo el proyecto. Su trabajo consiste en efectuar las iteraciones de manera efectiva y eficiente, con el fin de cumplir los objetivos establecidos para cada *Sprint*. Además, si surge algún problema, es responsabilidad del equipo informar al líder del proyecto para poder solventarlo y alcanzar las metas propuestas [24]. Es por esta razón que en la **Tabla 2.1** se especifica la asignación de la persona con su respectivo rol en el desarrollo del *backend*.

Tabla 2.1: Designación de roles para el proyecto.

ROL	INTEGRANTE
<i>Product Owner</i>	Sra. María Irene Jiménez Flores.
<i>Scrum Master</i>	Ing. Byron Loarte, MSc.
<i>Development Team</i>	Joselyn Gabriela Correa Figueroa.

Artefactos

Estos artefactos son los recursos que *Scrum* ofrece para recoger y gestionar de manera ordenada y efectiva todos los datos pertinentes para llevar a cabo la ejecución del proyecto *software* y evitar tomar decisiones equivocadas que puedan provocar retrasos innecesarios durante su transcurso [25]. Dentro del ámbito del *backend*, se han implementado los siguientes artefactos.

Recopilación de Requerimientos

La fase de recopilación de requisitos es fundamental para definir las necesidades que el proyecto debe cumplir. Estos requisitos se recopilan minuciosamente, se analizan y se documentan con un alto nivel de detalle, lo que facilita la fase de codificación del proyecto [26]. A continuación, en la **Tabla 2.2** se puede visualizar el formato que se ha utilizado para la recopilación de requerimientos del presente proyecto y en el **ANEXO II** se puede encontrar la lista completa.

Tabla 2.2: Recopilación de requerimientos.

RECOPIACIÓN DE REQUERIMIENTOS		
TIPO DE SISTEMA	ID – RR	ENUNCIADO DEL ITEM
<i>Backend</i>	RR- 008	Como usuario administrador necesita generar varios <i>endpoints</i> para: <ul style="list-style-type: none">• Gestionar subcategorías.

Historias de Usuario

Es una breve, sencilla y formal descripción de una función del proyecto de *software*, redactada desde la perspectiva del usuario que la requiere. Estas descripciones se redactan utilizando un enfoque natural, y se utilizan para comunicar y documentar de

manera concisa los requerimientos del proyecto. [27]. A continuación, en la **Tabla 2.3** se puede observar el formato que se ha establecido para las Historias de usuario del presente proyecto y en el **ANEXO II** se puede encontrar la lista completa.

Tabla 2.3: Formato Historias de Usuario.

HISTORIA DE USUARIO	
Identificador: HU-006	Usuario: Administrador
Nombre Historia: Gestionar subcategorías	
Prioridad en el negocio: Alta	Riesgo en el desarrollo: Media
Iteración asignada: 1	
Responsable: Joselyn Correa	
Descripción: El <i>backend</i> permite generar varios <i>endpoints</i> para que el perfil administrador pueda: <ul style="list-style-type: none"> • Gestionar subcategorías. 	
Observación: El usuario administrador es el único que tiene acceso a los <i>endpoints</i> antes mencionados, para poder hacerlo se requiere inicio de sesión.	

Product Backlog

Es una lista organizada que resume de manera estructurada las necesidades identificadas durante la Recopilación de requisitos. Es flexible, ya que puede ser modificado conforme avanza el proyecto. Además, cada elemento del *backlog* incluye una descripción detallada de las tareas que deben ser realizadas, y éstas se desarrollan en función de las prioridades establecidas [28]. A continuación, en **Tabla 2.4** se puede observar el formato que se ha establecido para el *Product Backlog* del presente proyecto y en el **ANEXO II** se puede encontrar la lista completa.

Tabla 2.4: Formato para el *Product Backlog*.

ELABORACION DEL <i>PRODUCT BACKLOG</i>				
ID – HU	HISTORIA DE USUARIO	ITERACIÓN	ESTADO	PROIRIDAD

HU-005	Gestionar categorías	1	Finalizado	Alta
HU-006	Gestionar subcategorías	1	Finalizado	Alta

Sprint Backlog

Después de haber desarrollado el *Product Backlog*, se genera una lista más detallada de las tareas que deben ser realizadas y ejecutadas. Estas tareas son organizadas y clasificadas mediante *Sprints* o Iteraciones [28]. Por otra parte, cada una de las tareas tiene asignado un plazo específico para ser completada. A continuación, en la **Tabla 2.5** se puede observar el formato que se ha establecido para el *Sprint Backlog* del presente proyecto y en el **ANEXO II** se puede encontrar la lista completa.

Tabla 2.5: Formato de *Sprint Backlog*.

ELABORACIÓN DE SPRINT BACKLOG						
ID – SB	NOMBRE	MÓDULO	ID-HU	HISTORIA DE USUARIO	TAREA	TIEMPO ESTIMADO
SB001	Diseño e implementación de <i>endpoints</i> para el usuario administrador.	Módulo - Gestión de categorías	HU004	Gestionar categorías	<ul style="list-style-type: none"> • Diseño e implementación de endpoints para crear, visualizar, eliminar y actualizar las categorías. • Consulta en la base de datos. • Validación de los datos requeridos. 	10H
		Módulo - Gestión de subcategorías	HU005	Gestionar subcategorías	<ul style="list-style-type: none"> • Diseño e implementación de endpoints para crear, visualizar, 	

					eliminar y actualizar las subcategorías. <ul style="list-style-type: none"> • Consulta en la base de datos. • Validación de los datos requeridos. 	
--	--	--	--	--	---	--

2.2 Diseño de la arquitectura

La arquitectura en el desarrollo de *software* se refiere al diseño y estructura fundamental de un sistema o aplicación. Es la base sobre la cual se construye todo el *software*, definiendo la manera en que los distintos componentes se relacionan e interactúan entre sí y cómo se organizan los datos. La arquitectura tiene como objetivo principal garantizar la calidad, el rendimiento, la mantenibilidad y la escalabilidad del *software*. Esto implica tomar decisiones clave sobre los patrones de diseño, selección de tecnologías, gestión de datos y la distribución de responsabilidades entre los diferentes módulos. Una arquitectura sólida y bien diseñada es esencial para el logro exitoso del proyecto, debido a que proporciona una estructura clara y coherente que facilita el desarrollo, evolución y el mantenimiento del *software* a lo largo del tiempo [29]. A continuación, se puede observar una descripción sobre la arquitectura que se ha empleado en el *backend*.

Patrón arquitectónico

MVC es una estructura de diseño arquitectónico de *software* que divide las responsabilidades en los siguientes componentes Modelo, Vista y Controlador. En base a lo citado, el modelo maneja los datos, la vista se encarga de la presentación y el controlador gestiona la lógica del negocio. Esta separación facilita la creación de módulos independientes y reutilizables, así como, la colaboración entre desarrolladores, mejorando así el mantenimiento del sistema. Además, MVC permite realizar cambios en la interfaz sin afectar la lógica subyacente, lo que ofrece flexibilidad y escalabilidad en el desarrollo de *software* [30]. En ese sentido, en la **Figura 2.1** se puede evidenciar el patrón de arquitectura que se ha utilizado para la codificación del *backend*, juntamente con la integración de librerías externas y una serie de herramientas de desarrollo asociadas.

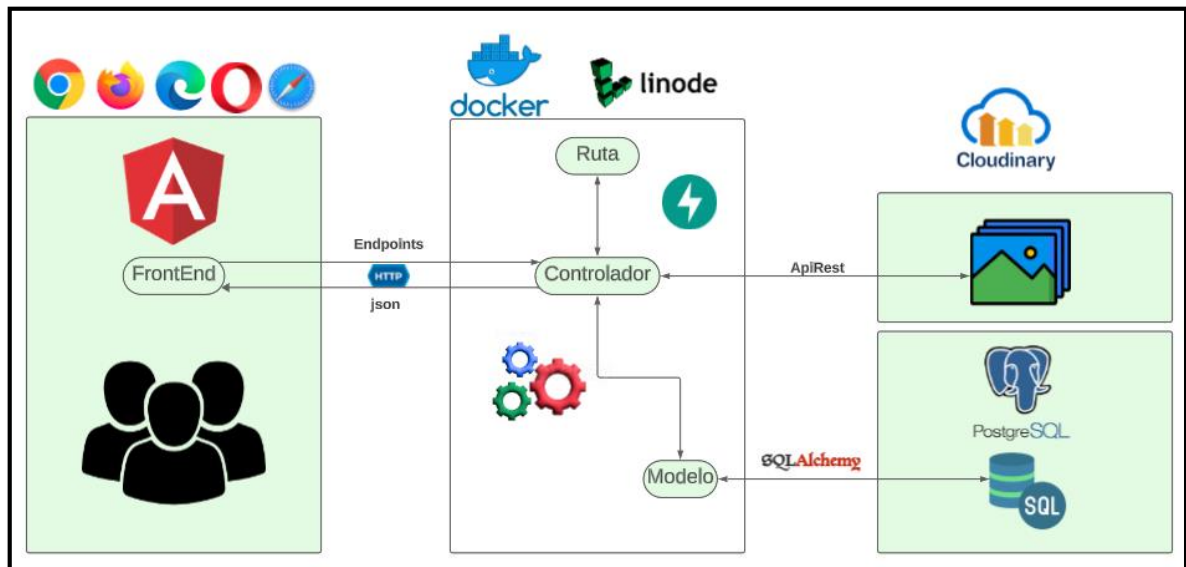


Figura 2.1: Patrón Arquitectónico – *backend*.

2.3 Herramientas de desarrollo

Reconocidas por su eficiencia y rapidez, las herramientas de desarrollo de *software* simplifican y agilizan el proceso de escritura de código. Estas herramientas ofrecen funcionalidades avanzadas e interfaces intuitivas, permitiendo a los desarrolladores enfocarse en la lógica del negocio y minimizar el tiempo empleado en tareas repetitivas. Por eso al elegir las mejores herramientas, se optimiza la productividad y se facilita la creación de código de mejor calidad en menor tiempo [31]. En la **Tabla 2.6** se detalla el conjunto de herramientas que se han seleccionado, así como su contribución al desarrollo del proyecto *backend*.

Tabla 2.6: Herramientas de *backend*.

HERRAMIENTA	JUSTIFICACIÓN
PyCharm	IDE de Python que se usa para el desarrollo del proyecto <i>backend</i> [32].
Gitlab y Git	Permite gestionar de manera eficiente el control de versiones del código del proyecto <i>backend</i> [33].
Linode	Proporciona servicios de alojamiento confiables y escalables, lo que ha permitido desplegar el <i>backend</i> de manera eficiente y garantizar su disponibilidad y rendimiento [17].

PostgreSQL	Brinda una Base de datos confiable y eficiente para almacenar y gestionar los datos de manera segura que son consumidos desde el <i>backend</i> [34].
Docker	Permite crear y desplegar de forma rápida y eficiente contenedores para proyectos <i>software</i> , lo que ha simplificado la configuración del entorno de desarrollo y facilitar la portabilidad del proyecto <i>backend</i> [35].

Librerías

Las librerías son archivos que se han desarrollado en diferentes lenguajes de programación y que ofrecen funcionalidades extra para simplificar y mejorar los procesos dentro del código fuente. Debido a esta razón, en la **Tabla 2.7** se muestran las librerías que han sido utilizadas para complementar la creación de cada uno de los *endpoints* en la codificación del *backend*.

Tabla 2.7: Librerías en el *backend*.

LIBRERÍA	DESCRIPCIÓN
FastApi	Permite desarrollar una API rápida, robusta y bien documentada, mejorando la eficiencia y la calidad del código [11].
Uvicorn	Brinda un servidor de desarrollo rápido y eficiente para la ejecución de la API que se ha desarrollado con FastAPI [36].
psycopg2-binary	Facilita la interacción eficiente con la Base de datos PostgreSQL [37].
SQLAlchemy	Permite simplificar y agilizar las operaciones relacionadas con el acceso y manipulación de datos [38].
python-dotenv	Permite cargar fácilmente variables de entorno desde archivos <i>.env</i> , asegurando la seguridad y flexibilidad en la gestión de configuraciones sensibles [39].
alembic	Facilita la realización de migraciones de Base de datos en Python, posibilitando cambios estructurales controlados y sin complicaciones en la Base de datos. [40].
passlib[bcrypt]	Garantiza la seguridad y eficiencia en el almacenamiento y verificación de contraseñas encriptadas [41].

python-jose[<code>cryptography</code>]	Proporciona funcionalidades de criptografía y seguridad para la manipulación de tokens y firmas digitales [42].
python-multipart	Ayuda a simplificar el manejo de datos en la carga de archivos [43].
pydantic	Sirve para definir y validar modelos de datos de forma sencilla y segura [44].
regex	Ayuda a realizar búsquedas y manipulaciones avanzadas de patrones de texto [45].
nameparser	Sirve para analizar y descomponer nombres completos de manera eficiente [46].

3 RESULTADOS

En este apartado del documento, se presentan de forma detallada los notables avances que se han logrado tras la implementación y desarrollo de los *endpoints* que definen la lógica del *backend*. Por otra parte, se exponen los resultados que se han alcanzado en cada prueba, así como su respectivo despliegue en producción. Además, se describen minuciosamente los logros en los *Sprints*, brindando un enfoque claro y progresivo a cada uno de ellos.

3.1 *Sprint 0. Configuración del ambiente de desarrollo.*

El *Sprint Backlog* para este *Sprint* incluye las siguientes tareas:

- Establecimiento de los requerimientos y limitaciones.
- Creación de la estructura del proyecto *backend*.
- Diseño y desarrollo de la Base de datos SQL.
- Definición y asignación de los roles de usuario.

Definición de requerimientos y limitaciones

Implementación de *endpoints* para el registro de usuarios

El *backend* incorpora un *endpoint* que posibilita el registro de usuarios con perfil cliente utilizando los campos predefinidos.

Implementación de *endpoints* para inicio de sesión, cierre de sesión y modificar contraseña

El *backend* incorpora un *endpoint* que posibilita a los usuarios con perfil administrador, empleado y cliente iniciar sesión mediante los campos predefinidos, además de otros *endpoints* que posibilitan el cierre de sesión y la modificación de contraseña.

Implementación de *endpoints* para modificar el perfil de usuario

El *backend* incorpora un *endpoint* que posibilita a los usuarios con perfil administrador, empleado y cliente visualizar y modificar su perfil personal de usuario.

Implementación de *endpoints* para gestionar productos

El *backend* incorpora un *endpoint* que posibilita al usuario con perfil administrador, gestionar productos. De esta forma, el usuario puede ver, registrar, modificar y eliminar el listado de productos.

Implementación de *endpoints* para gestionar categorías

El *backend* incorpora un *endpoint* que posibilita al usuario con perfil administrador, gestionar categorías. De esta forma, el usuario puede ver, registrar, modificar y eliminar categorías.

Implementación de *endpoints* para gestionar subcategorías

El *backend* incorpora un *endpoint* que posibilita al usuario con perfil administrador, gestionar subcategorías. De esta forma, el usuario puede ver, registrar, modificar y eliminar subcategorías.

Implementación de *endpoints* para gestionar pedidos

El *backend* incorpora un *endpoint* que posibilita al usuario con perfil administrador, gestionar pedidos. De este modo, el usuario puede visualizar, aceptar o rechazar pedidos.

Implementación de *endpoints* para gestionar comentarios y/o sugerencias

El *backend* incorpora un *endpoint* que posibilita al usuario con perfil administrador, gestionar comentarios y/o sugerencias que han sido realizados por los usuarios con perfil cliente. Esto brinda la capacidad de leer y responder los comentarios y/o sugerencias.

Implementación de *endpoints* para seleccionar el pedido como entregado

El *backend* incorpora un *endpoint* que posibilita al usuario con perfil empleado y seleccionar el pedido como entregado. De este modo, el usuario puede notificar que el pedido ha sido entregado.

Implementación de *endpoints* para visualizar favoritos

El *backend* incorpora un *endpoint* que posibilita al usuario con perfil cliente, visualizar sus productos favoritos. De esta forma el usuario puede visualizar de una manera más rápida los productos que adquiere con mayor frecuencia.

Implementación de *endpoints* para gestionar carrito de compras

El *backend* incorpora un *endpoint* que posibilita al usuario con perfil cliente, gestionar el carrito de compras. De este modo, el usuario puede crear, eliminar, editar o rechazar los pedidos.

Implementación de *endpoints* para gestionar pedidos

El *backend* incorpora un *endpoint* que posibilita al usuario con perfil cliente, gestionar los pedidos que va a realizar. De este modo, el usuario puede crear, eliminar, editar o rechazar

pedidos. Adicional a ello, el usuario tiene que subir un comprobante de pago si ha seleccionado el método por transferencia.

Implementación de *endpoints* para visualizar detalle de pedido

El *backend* incorpora un *endpoint* que posibilita al usuario con perfil cliente, visualizar el estado del pedido. De este modo, el usuario puede observar información sobre el estado del pedido que desea realizar, ya sea que este haya sido aceptado o rechazado. En caso de que el pedido sea aceptado, también puede verificar si ha sido enviado y si está en camino para su entrega.

Implementación de *endpoints* para gestionar historial de pedidos

El *backend* incorpora un *endpoint* que posibilita al usuario con perfil cliente, gestionar el historial de pedidos. De este modo, el usuario tiene la posibilidad de revisar todos los pedidos que ha realizado y ver el detalle de cada uno.

Implementación de *endpoints* para enviar comentarios y/o sugerencias

El *backend* incorpora un *endpoint* que posibilita al usuario con perfil cliente, enviar comentarios y/o sugerencias. Esto otorga al usuario la capacidad de reportar cualquier contratiempo ocurrido.

Por último, la **Figura 3.1** presenta un gráfico que ilustra las distintas características disponibles para los usuarios con roles de administrador, empleado y cliente. El objetivo es facilitar la comprensión y el conocimiento de las funcionalidades que están a su disposición.

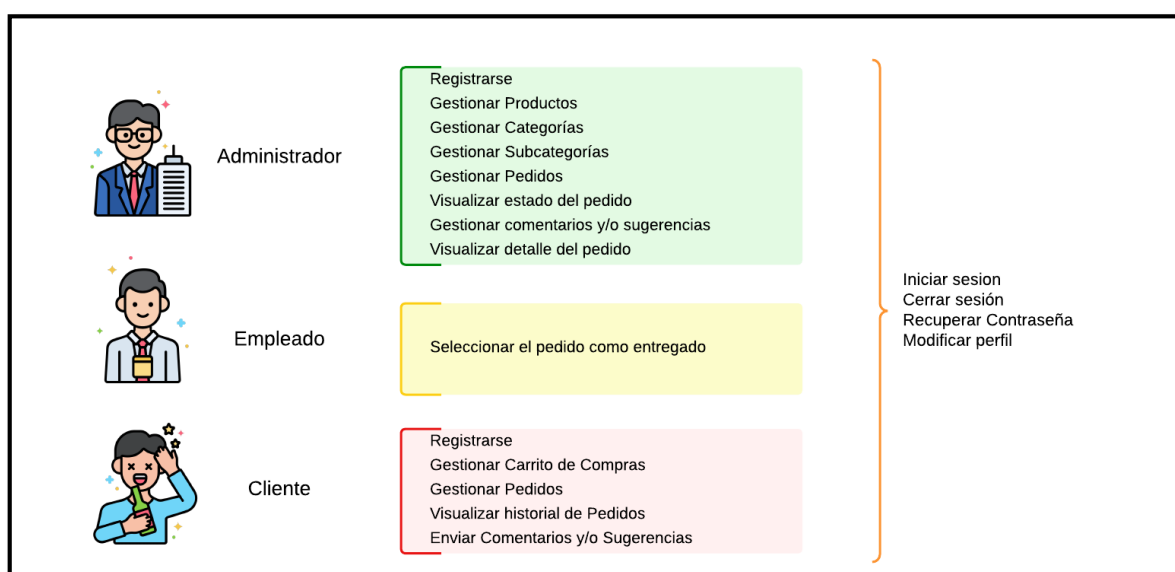


Figura 3.1: Perfiles de usuario y funcionalidades dentro del *backend*.

Creación de la estructura del proyecto *backend*

Se ha utilizado el editor de código PyCharm creado específicamente para trabajar con Python y de esta manera agilizar la implementación de los distintos *endpoints* del *backend*. Además, ofrece una estructura inicial basada en el patrón arquitectónico que se ha establecido anteriormente, junto con los archivos de configuración y directorios necesarios, lo que facilita la organización del proyecto como se puede observar en la **Figura 3.2**

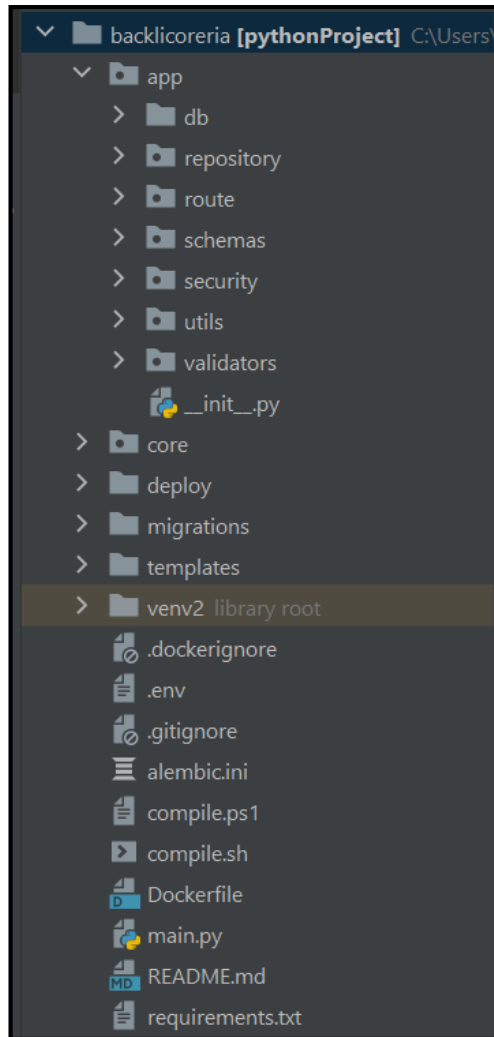


Figura 3.2: Estructura del proyecto.

Diseño y desarrollo de la Base de datos SQL

En la licorería, se ha decidido emplear PostgreSQL como sistema gestor de bases de datos para implementar un enfoque relacional, garantizando así la integridad de la información. Al basarse en relaciones entre tablas, PostgreSQL permite almacenar los datos de manera coherente y eficiente, lo que facilita la gestión y el acceso efectivo a la información como

se puede observar en la **Figura 3.3** a cada una de las tablas, mientras que en el **ANEXO II** se puede apreciar el diseño completo de manera detallada.

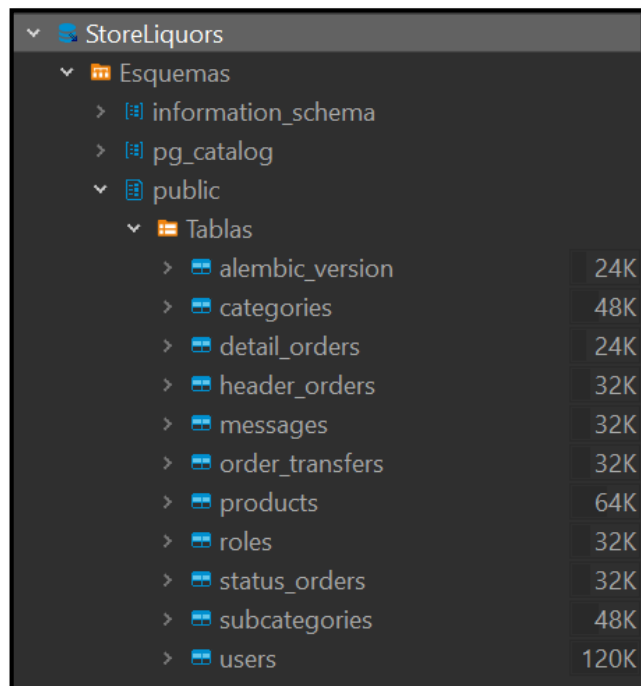


Tabla	Tamaño
alembic_version	24K
categories	48K
detail_orders	24K
header_orders	32K
messages	32K
order_transfers	32K
products	64K
roles	32K
status_orders	32K
subcategories	48K
users	120K

Figura 3.3: Diseño de la Base de Datos.

Definición y asignación de roles de usuario

Como parte del desarrollo del *backend*, se han definido 3 roles de usuario. Estos roles permiten asignar funciones y permisos específicos, asegurando un control de acceso adecuado y una gestión eficiente dentro de los módulos del *backend*. Por último, los roles de usuario se encuentran representados en la **Figura 3.4**.

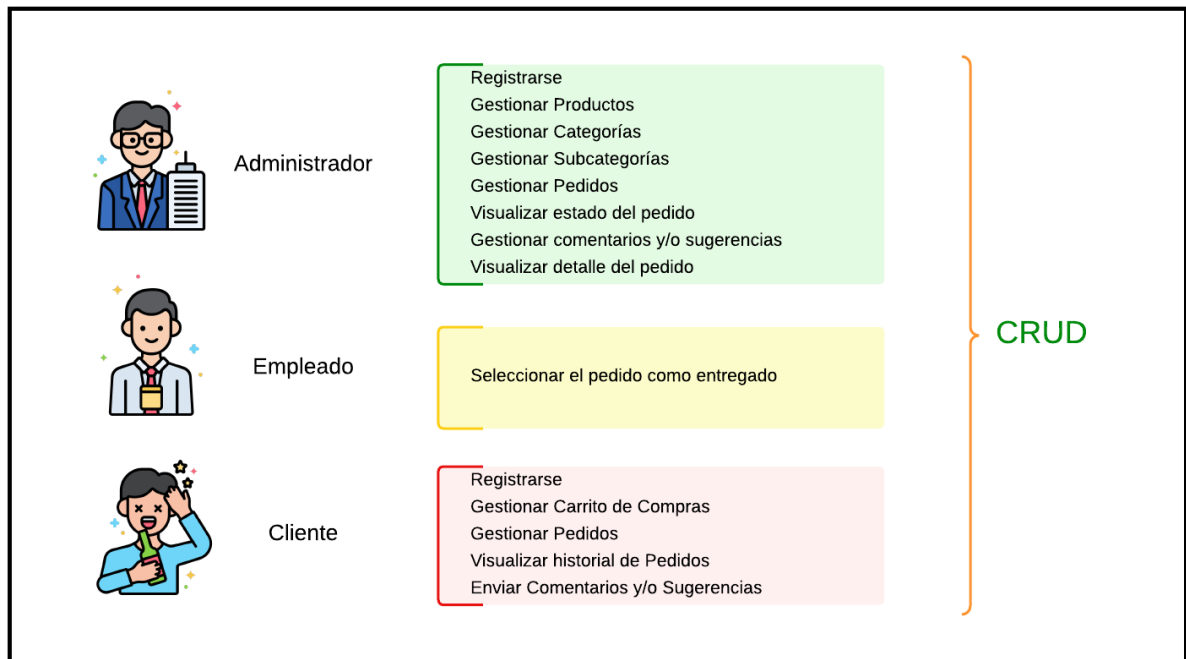


Figura 3.4: Roles de usuario del proyecto *backend*.

3.2 *Sprint* 1. Resultados del desarrollo de *endpoints* para el perfil administrador.

En el *Sprint* actual, se han definido las siguientes tareas:

- Generar *endpoints* que permiten iniciar sesión, cerrar sesión y modificar contraseña.
- Generar *endpoints* que permiten visualizar y modificar el perfil de usuario.
- Generar *endpoints* que permiten gestionar productos.
- Generar *endpoints* que permiten gestionar categorías.
- Generar *endpoints* que permiten gestionar subcategorías.
- Generar *endpoints* que permiten gestionar pedidos.
- Generar *endpoints* que permiten visualizar estado de pedidos.
- Generar *endpoints* que permiten gestionar comentarios y/o sugerencias.
- Generar *endpoints* que permiten visualizar detalle de pedido.

Generar *endpoints* que permiten iniciar sesión, cerrar sesión y modificar contraseña

Se han implementado diversos *endpoints* con sus respectivas rutas, permitiendo al usuario con perfil administrador, empleado y cliente realizar lo siguiente implementar un *endpoint* de tipo POST para el inicio de sesión, donde se autentica al usuario y se le otorga acceso a los recursos según su rol de usuario, como se puede observar en la **Figura 3.5** Además, la **Figura 3.6** muestra el resultado de la prueba unitaria correspondiente. Por otra parte, se implementan varios *endpoints* de tipo POST para modificar la contraseña en caso de que un usuario la olvide. Cabe destacar que, el consumo y almacenamiento de registros, junto con las validaciones de cada campo, están claramente detallados en el **ANEXO III**.

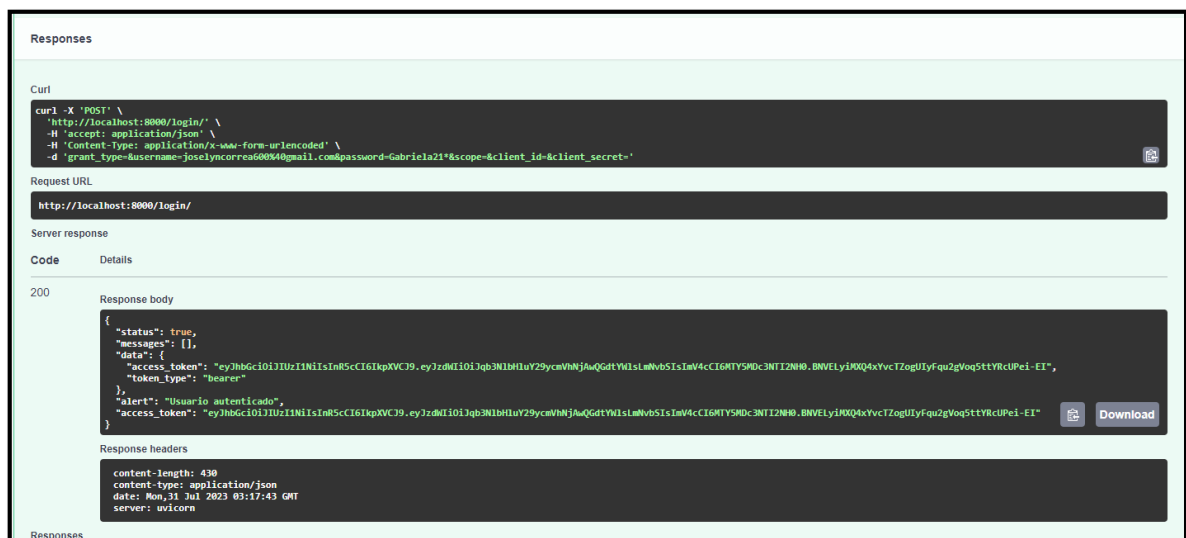


Figura 3.5: *Endpoint* de *login*.

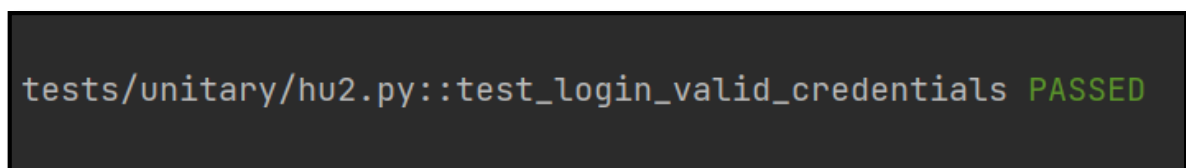


Figura 3.6 Resultado de la prueba unitaria de *login*.

Generar *endpoints* que permiten visualizar y modificar el perfil de usuario

Para visualizar y modificar el perfil de usuario, se han implementado diversos *endpoints* con sus respectivas rutas permitiendo al usuario con perfil administrador, empleado y cliente modificar la información de su perfil de forma completa. Entre estos *endpoints*, se ha implementado un *endpoint* privado de tipo POST para la actualización de los datos. También, se encuentra uno de tipo GET con una ruta privada que permite obtener todos

los datos del usuario autenticado previamente, como se puede observar en la **Figura 3.7** y el resultado de la prueba unitaria correspondiente se encuentra detallado en la **Figura 3.8**. Es importante resaltar que, el consumo y almacenamiento de registros, junto con las validaciones de cada campo, están claramente detallados en el **ANEXO III**.

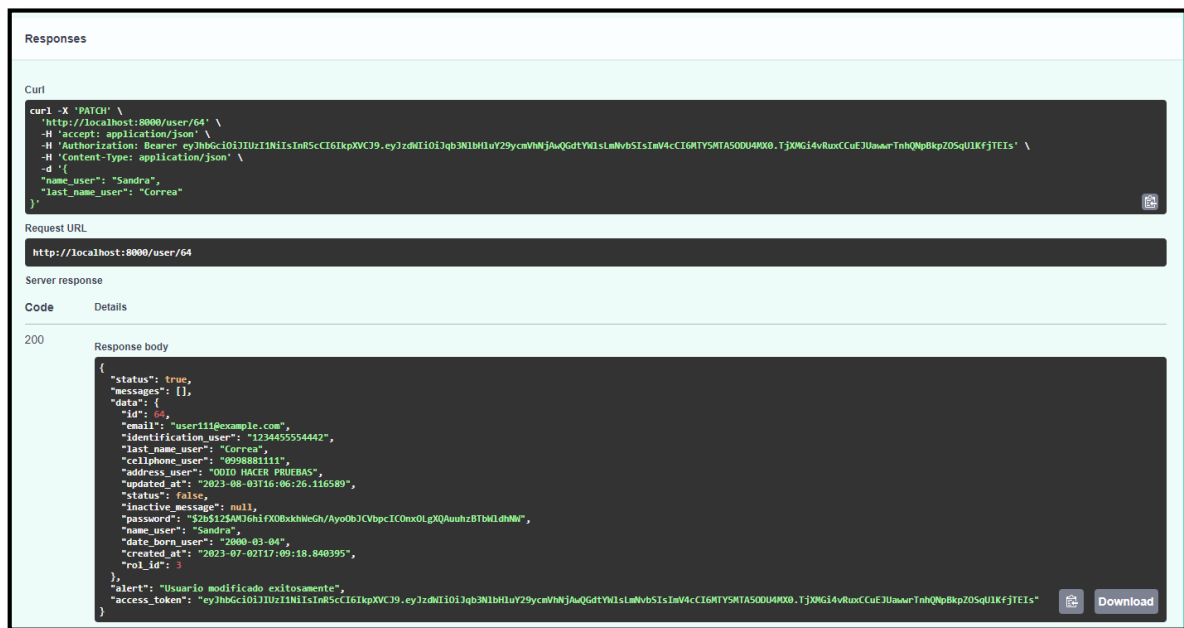


Figura 3.7: modificar perfil de usuario – Administrador.

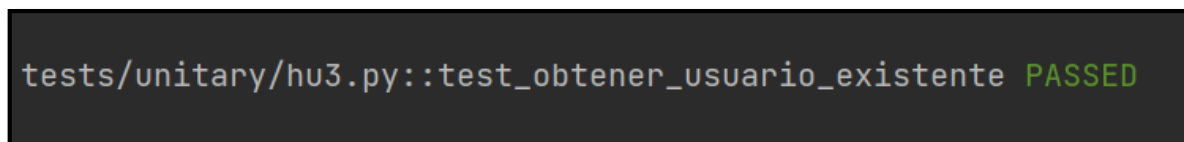


Figura 3.8: Resultado de la prueba modificar perfil de usuario.

Generar *endpoints* que permiten gestionar productos

Con el fin de garantizar una gestión integral de los productos de la licorería, se han implementado varios *endpoints* con sus respectivas rutas. Estos *endpoints* permiten al usuario con perfil administrador gestionar los productos de manera efectiva. En este sentido, se ha incluido un *endpoint* privado de tipo POST para registrar un producto, así como dos *endpoints* públicos de tipo GET. Permitiendo obtener el registro de un producto según su ID y como se puede observar en la **Figura 3.9** y el resultado de la prueba unitaria correspondientes se encuentra detallado en la **Figura 3.10**. Es importante resaltar que, el

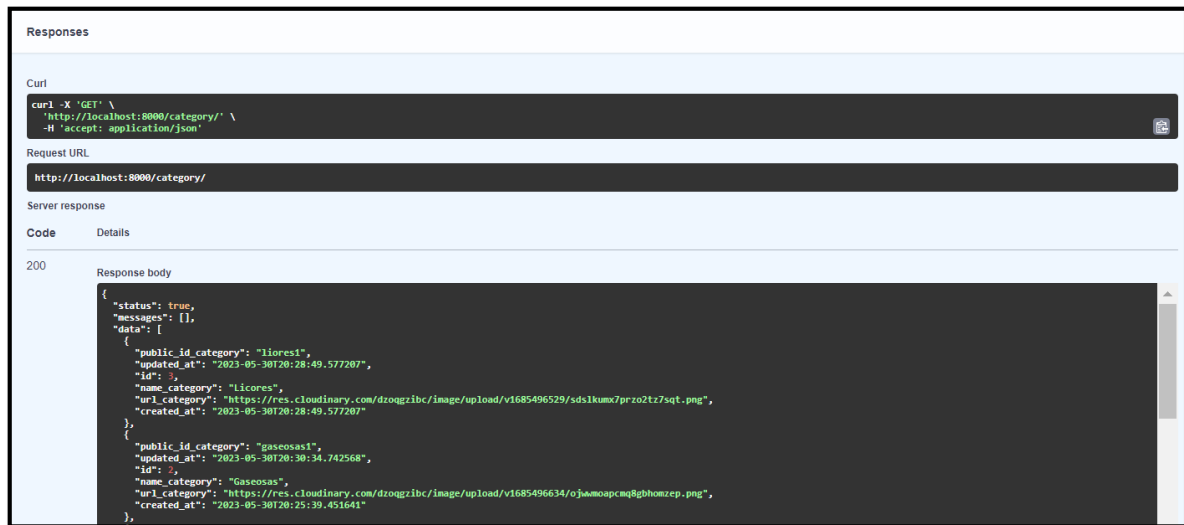


Figura 3.11: Listar categorías.

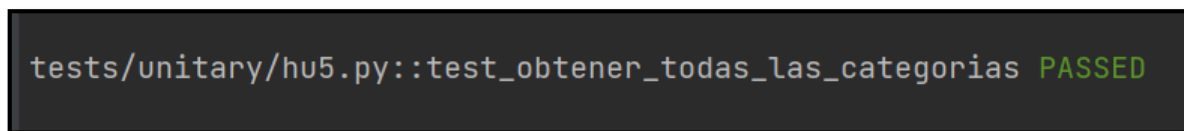


Figura 3.12: Resultado de la prueba listar categorías.

Generar *endpoints* que permiten gestionar subcategorías

Para la gestión de subcategorías de productos, se han implementado diversos *endpoints* con sus respectivas rutas permitiendo al usuario con perfil administrador gestionar las categorías de forma completa. Se ha desarrollado un *endpoint* privado de tipo POST que permite registrar una subcategoría, y dos *endpoints* públicos de tipo GET para obtener información específica sobre una subcategoría mediante su ID, así como la lista completa de todas las subcategorías que se han registrado. Además, se ha incluido un *endpoint* privado de tipo DELETE para eliminar la información de una subcategoría existente, utilizando su ID como se puede ver en la **Figura 3.13** y en **Figura 3.14** se encuentra el resultado de la prueba unitaria. Es importante resaltar que, el consumo y almacenamiento de registros, junto con las validaciones de cada campo, están claramente detallados en el **ANEXO III**.

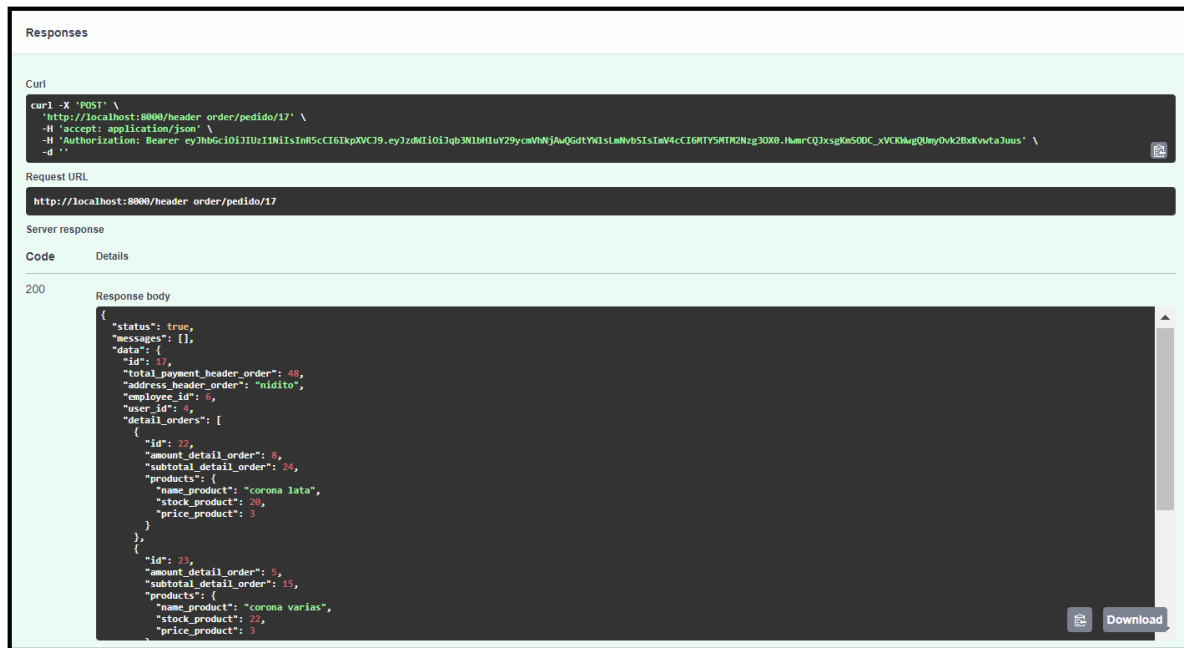


Figura 3.21: Detalle de un pedido.

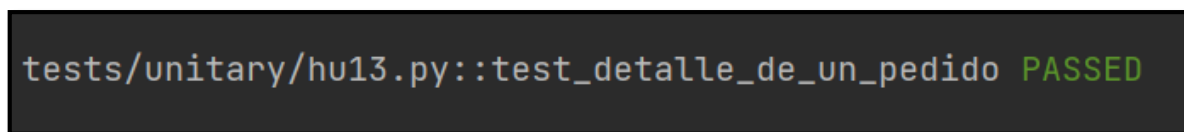


Figura 3.22: Resultado de la prueba detalle de un pedido.

3.3 *Sprint 2. Resultados del desarrollo de endpoints para el perfil empleado.*

En el *Sprint* actual, se ha definido la siguiente tarea:

Generar endpoints para seleccionar el pedido como entregado

Para permitir la selección del estado de un pedido, se han implementado varios *endpoints* con sus respectivas rutas. Estos *endpoints*, de tipo PATCH y privados, brindan la posibilidad al usuario con perfil empleado actualizar el estado de un pedido específico. Por ejemplo, se puede marcar un pedido como "Pendiente", indicando que está en curso su preparación e indicando que ha sido despachado y está en camino hacia su destino. Por último, se puede marcar un pedido como "Entregado", cuando este ha sido recibido por el cliente como se puede observar en la **Figura 3.23** y en la **Figura 3.24** se encuentra el resultado de la prueba unitaria. Es importante resaltar que, el consumo y almacenamiento de registros, junto con las validaciones de cada campo, están claramente detallados en el **ANEXO III**.

información necesaria para registrarse, como nombre de usuario, correo electrónico contraseña entre otros como se puede observar en la **Figura 3.25** y en la **Figura 3.26** se encuentra el resultado de la prueba unitaria. Es importante resaltar que, el consumo y almacenamiento de registros, junto con las validaciones de cada campo, están claramente detallados en el **ANEXO III**.

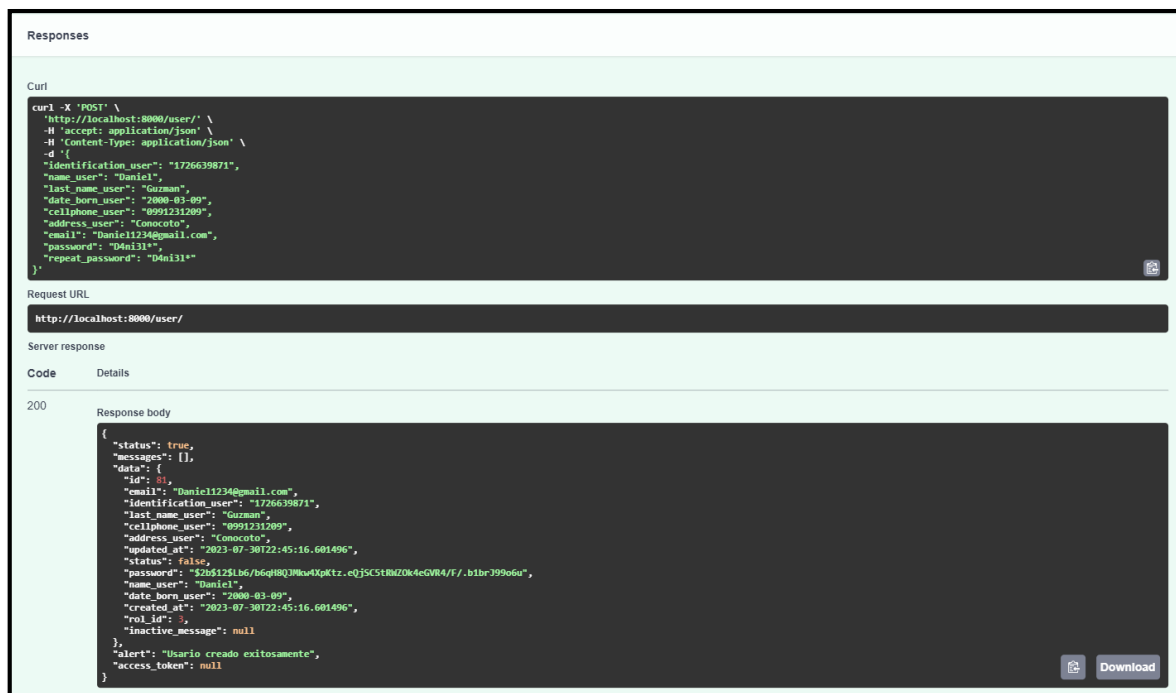


Figura 3.25: Proceso para registrarse.

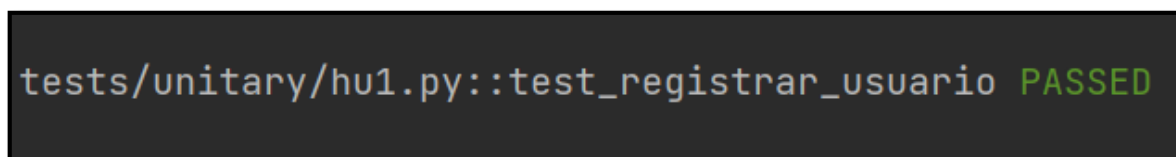


Figura 3.26: Resultado de la prueba unitaria para registrarse.

Gestionar *endpoints* que permiten visualizar favoritos

Para facilitar la visualización de los elementos marcados como favoritos, se han implementado varios *endpoints* con sus respectivas rutas. Estos *endpoints*, de tipo GET y públicos, proporcionan información detallada sobre cada elemento favorito, como su nombre, descripción, precio entre otros como se puede observar en la **Figura 3.27** y en la **Figura 3.28** se encuentra el resultado de la prueba unitaria. Es importante resaltar que, el consumo y almacenamiento de registros, junto con las validaciones de cada campo, están claramente detallados en el **ANEXO III**.

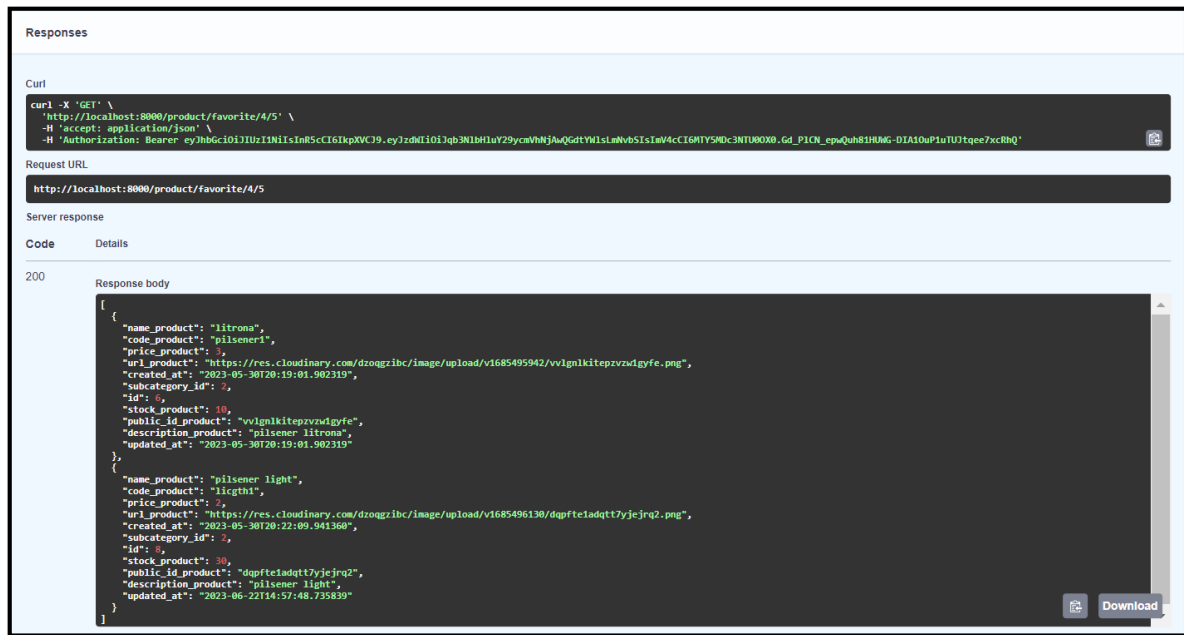


Figura 3.27: Obtener productos favoritos.

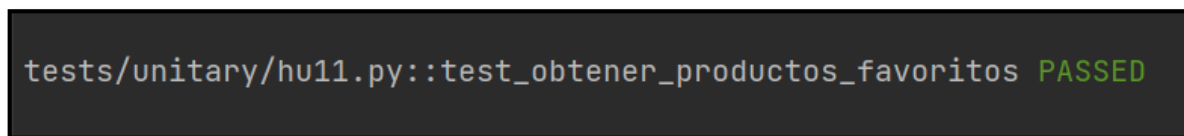


Figura 3.28: Resultado de la prueba obtener favoritos.

Gestionar endpoints que permiten gestionar carrito de compras

Para facilitar la gestión del carrito de compras, se han implementado varios endpoints con sus respectivas rutas que permiten al usuario con perfil cliente gestionar de manera eficiente los productos seleccionados para su compra. Estos endpoints privados de tipo GET, POST, PATCH y DELETE, brindan funcionalidades clave para ver, agregar, actualizar y eliminar productos del carrito. Los usuarios pueden obtener una vista detallada de los productos en su carrito mediante un endpoint de tipo GET, agregar nuevos productos con un endpoint de tipo POST, actualizar la cantidad de productos con un endpoint de tipo PATCH, y eliminar productos específicos con un endpoint de tipo DELETE como se puede observar en la Figura 3.29 y en la Figura 3.30 se encuentra el resultado de la prueba unitaria. Es importante resaltar que, el consumo y almacenamiento de registros, junto con las validaciones de cada campo, están claramente detallados en el ANEXO III.

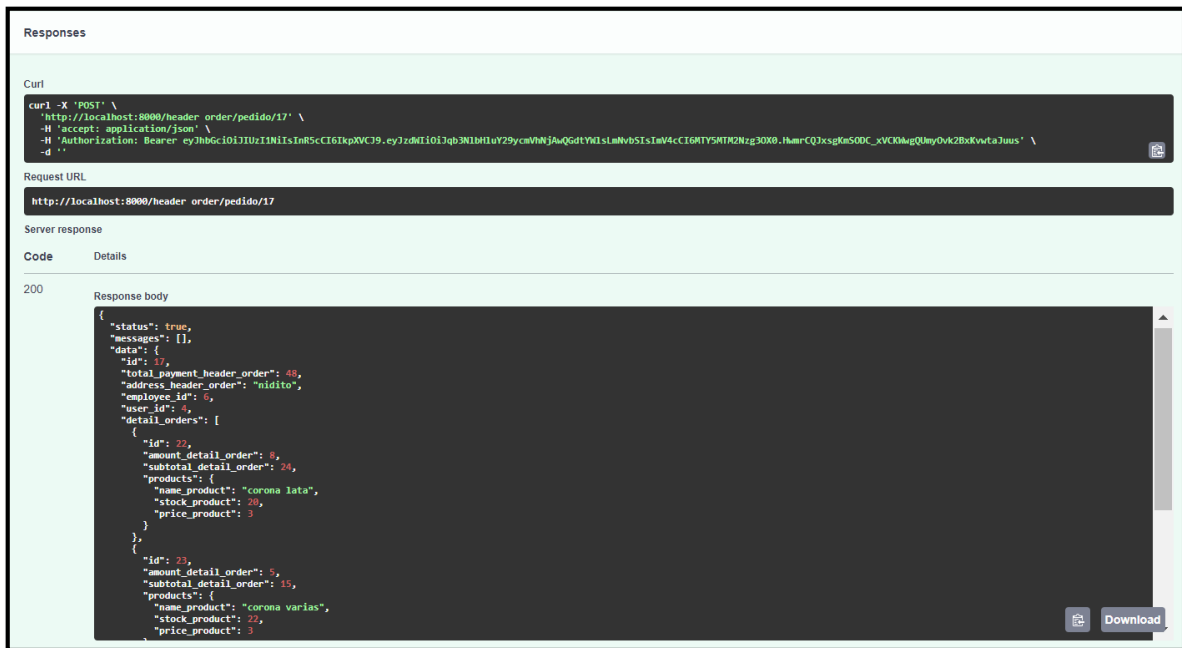


Figura 3.29: Carrito de compras.

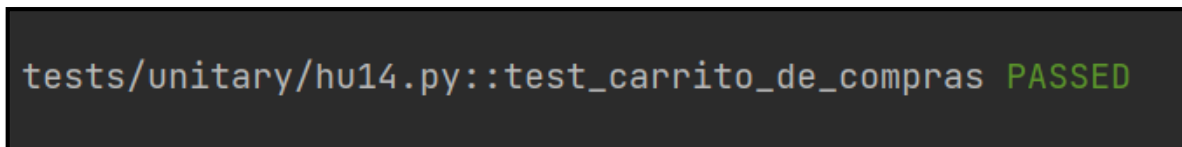


Figura 3.30: Resultado de la prueba carrito de compras.

Gestionar *endpoints* que permiten gestionar pedido

Para simplificar la gestión de pedidos, se han implementado varios *endpoints* con sus respectivas rutas que permiten al usuario con perfil cliente gestionar de manera eficiente los pedidos que se han realizado. Estos *endpoints* privados de tipo GET, POST, PUT y DELETE, ofrecen funcionalidades clave para visualizar, crear, actualizar y eliminar pedidos. Estas rutas han sido diseñadas para proporcionar una experiencia fluida en la gestión de pedidos como se muestra en la **Figura 3.31** Además, puede elegir el método de pago al realizar un pedido si su elección es mediante transferencia el usuario empleado tiene que subir una foto del comprobante de pago. En la **Figura 3.32** se muestra el resultado de la prueba unitaria. Es importante resaltar que, el consumo y almacenamiento de registros, junto con las validaciones de cada campo, están claramente detallados en el **ANEXO III**.

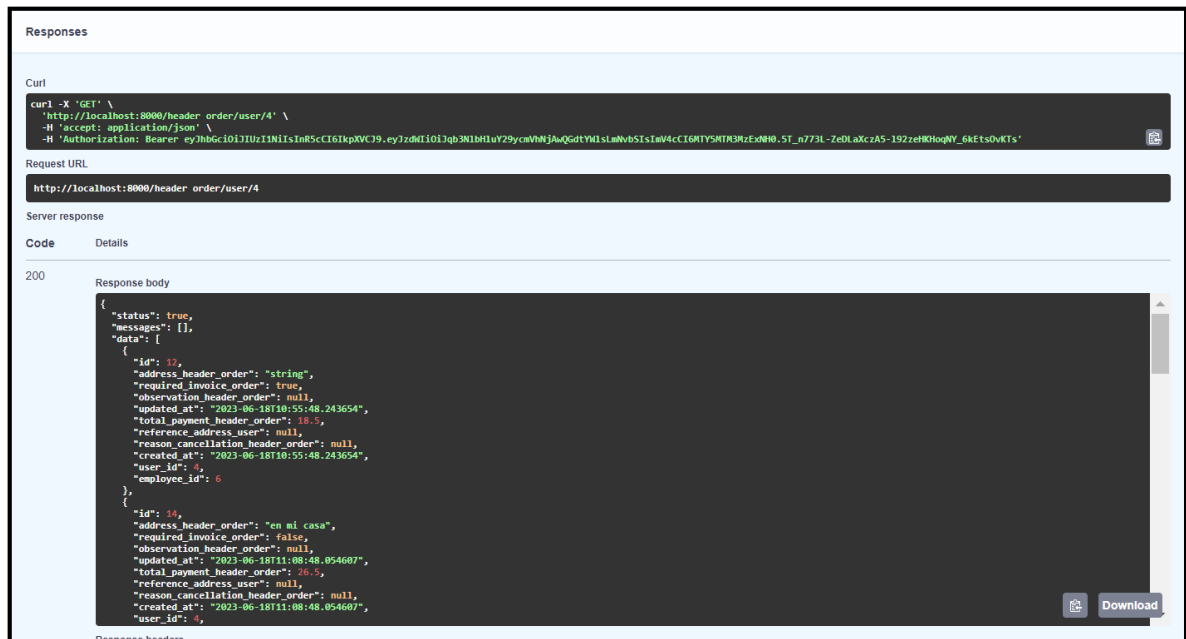


Figura 3.33: Historial de pedidos.

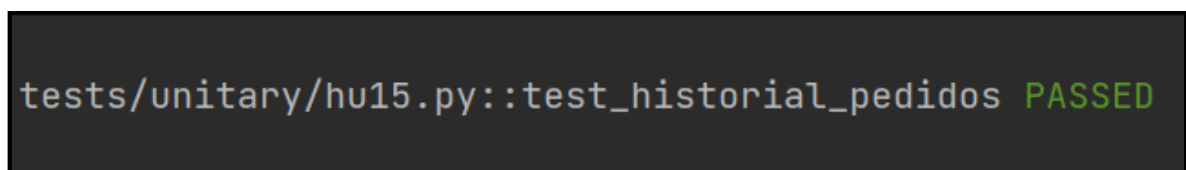


Figura 3.34: Resultado de la prueba historial de pedidos.

Gestionar *endpoints* que permiten enviar comentarios y/o sugerencias

Para permitir a los usuarios enviar comentarios y/o sugerencias, se han implementado varios *endpoints* con sus respectivas rutas que permiten al usuario con perfil cliente enviar comentarios y/o sugerencias, Mediante un *endpoint* de tipo POST, los usuarios pueden enviar sus comentarios y/o sugerencias, proporcionando información detallada y relevante. Esta función mejora continuamente el negocio mediante valiosos comentarios y/o sugerencias de los clientes, impulsando su desarrollo y mejora constante. El uso de estos *endpoints* se puede observar en la **Figura 3.35** y en la **Figura 3.36** se muestra el resultado de la prueba unitaria. Es importante resaltar que, el consumo y almacenamiento de registros, junto con las validaciones de cada campo, están claramente detallados en el **ANEXO III**.

En este caso, se ha utilizado la biblioteca de pruebas pytest, que proporciona una manera sencilla de realizar pruebas unitarias en Python. En la **Figura 3.37** se muestra una sección del código para obtener un usuario por un id, y en la **Figura 3.38** se muestra el resultado detallado de la implementación de esa prueba en particular. Además, todas las pruebas restantes y sus resultados correspondientes se encuentran detallados en el **ANEXO II**.

```
Gautzelin *
def test_obtener_usuario_por_id():

    response_login = client.post("/login/",
                                data={"username": "joselyncorrea600@gmail.com", "password": "Gabriela21*"})
    assert response_login.status_code == 200
    token = response_login.json()
    assert "access_token" in token
    headers = {"Authorization": f"Bearer {token['access_token']}"}
    response = client.post(f"/user/4", headers=headers)

    assert response.status_code == 200
    print(response.json())
```

Figura 3.37: Código de la prueba obtener usuario.

```
tests/unitary/hu1.py::test_obtener_usuario_por_id PASSED
```

Figura 3.38: Resultado de la prueba obtener usuario.

Los resultados de las pruebas unitarias confirman la adecuada operatividad de los módulos, demostrando la implementación exitosa de todas las validaciones requeridas.

Ejecución de pruebas de compatibilidad

Estas pruebas se enfocan en garantizar que el *backend* sea compatible con diferentes sistemas operativos, versiones de *software*, componentes y entornos de ejecución. El objetivo es asegurar que el sistema se comporte de manera consistente y sin problemas en una variedad de configuraciones, evitando incompatibilidades y problemas de funcionamiento que puedan surgir al interactuar con diferentes elementos del entorno tecnológico [48]. En este caso, se emplearon los siguientes clientes HTTP Swagger, Imsomnia y Postman que proporcionan una manera sencilla de realizar pruebas unitarias en Python. Como resultado, se muestra en la **Tabla 3.1** el desglose de los clientes HTTP que se han utilizado anteriormente, junto con los resultados de las pruebas correspondientes, los cuales se pueden visualizar en el **ANEXO II**.

Tabla 3.1: Pruebas de compatibilidad

NOMBRE	VERSIÓN
Swagger	3.0
Insomnia	2023.4.0
Postman	10.16.0

Las pruebas que se han efectuado en los distintos *endpoints* han confirmado su correcto funcionamiento en términos de tiempo de respuesta y presentación de información.

Ejecución de pruebas de estrés

Estas pruebas evalúan el rendimiento y la capacidad de un sistema bajo cargas extremas. Estas pruebas someten al *backend* a altos volúmenes de solicitudes o cargas de datos para observar su respuesta, tiempo de respuesta y estabilidad. Su propósito es asegurar que el sistema maneje situaciones de alta demanda sin problemas, detectar cuellos de botella, identificar fallos y optimizar la escalabilidad y capacidad de respuesta [49]. En este caso, se ha utilizado la plataforma de pruebas locust, que proporciona una manera sencilla de realizar pruebas unitarias en Python. Como resultado, en la **Figura 3.39** se puede observar a detalle la prueba de estrés y su respectivo resultado, el detalle de las pruebas faltantes se presenta en el **ANEXO II**.

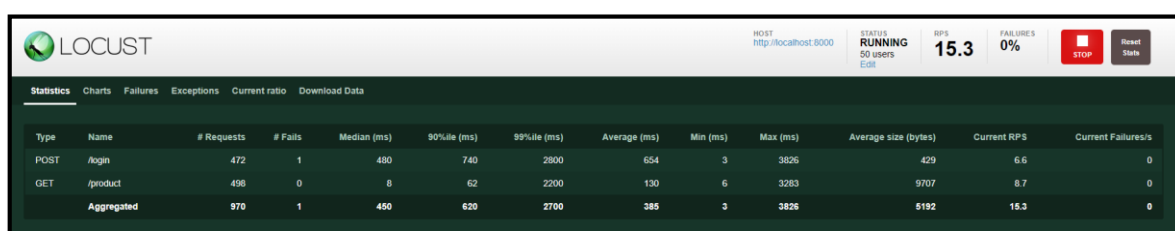


Figura 3.39: Prueba estrés *login*.

Se llevó a cabo una prueba para simular 50 usuarios, con una tasa de inicio de sesión de 5 usuarios por segundo. Los resultados obtenidos de la prueba de *login* indican que los usuarios están satisfechos con las respuestas proporcionadas por el *backend*.

Ejecución de pruebas de aceptación

Estas evaluaciones se enfocan en determinar si el sistema de *software* satisface los requisitos y expectativas establecidos por el propietario del producto final y, al mismo

tiempo, aseguran que ha sido desarrollado conforme a los requerimientos establecidos al inicio del proyecto [50]. Como resultado, en la **Tabla 3.2** se puede observar el detalle de una prueba de aceptación y su respectivo resultado, el detalle de las pruebas faltantes se presenta en el **ANEXO II**.

Tabla 3.2 Prueba de aceptación - Gestionar categorías

PRUEBA DE ACEPTACIÓN	
Identificador (ID): PA-005	Identificador de historia de Usuario: HU005
Nombre: Gestionar categorías	
Descripción: El usuario administrador en el <i>Backend</i> deben generar varios <i>endpoints</i> para: <ul style="list-style-type: none"> • Registrar categorías • Modificar categorías • Eliminar categorías • Visualizar lista de categorías 	
Pasos de ejecución: Para gestionar las categorías: <ul style="list-style-type: none"> • Inicio de sesión como usuario de tipo administrador. • Visualizar el listado de las categorías. • Crear nuevas categorías. • Editar categorías existentes. • Eliminar categorías existentes. 	
Resultado deseado: El <i>backend</i> permite crear, actualizar, eliminar y listar categorías.	
Evaluación de la prueba: El cliente se muestra satisfecho con el resultado al 100%.	

Tras realizar la prueba mencionada, se ha verificado que todos los módulos cumplen con los requisitos recopilados al comienzo del proyecto, y, además, han sido aprobados por el *Product Owner*.

3.6 Sprint 5. Despliegue del componente *backend*

En el *Sprint* actual, se ha definido la siguiente tarea:

Despliegue del *backend* en Linode.

Una vez concluida la programación de cada módulo, se lleva a cabo la implementación del *backend* en el entorno de producción en Linode y la URL para acceder al mismo es la siguiente:

<https://api-licoreria-la-nenita.kuntur.pro/docs>

Finalmente, el administrador de la Licorería "La Nenita" ha emitido un certificado que valida la satisfacción de todos los requisitos y características del *backend* que fueron solicitados al inicio del proyecto. Dicho certificado se lo puede observar en el **ANEXO II**.

4 CONCLUSIONES

En esta sección del documento, se presentan todas las conclusiones surgidas durante el desarrollo del *backend* como parte de este Trabajo de Integración Curricular:

- Es fundamental basar el desarrollo únicamente en los requerimientos que se han recopilado inicialmente. Esta práctica asegura una comprensión clara de las tareas a realizar y evita la inclusión de funcionalidades no solicitadas.
- Diseñar cuidadosamente la estructura de la Base de Datos, asegura notablemente la compatibilidad entre los datos y las necesidades del sistema. Esto facilita la gestión eficiente de la información y promueve la integridad de estos.
- Mediante la implementación del patrón arquitectónico Modelo-Vista-Controlador (MVC), se ha logrado organizar el *backend* de manera efectiva. Este enfoque ha permitido mantener el código de forma más sencilla mediante la separación de las distintas funcionalidades en bloques de código y una estructura de archivos clara.
- Codificar los *endpoints* y módulos del *backend* según los roles es esencial para garantizar la seguridad y control en el sistema, permitiendo una mejor administración de recursos y una experiencia de usuario personalizada.
- Las pruebas de los endpoints se llevaron a cabo de manera efectiva una vez que se comprendieron los procedimientos adecuados. No se encontraron obstáculos significativos durante el proceso, destacando la importancia de una planificación sólida y una documentación adecuada. Estos resultados positivos inspiran confianza en la calidad y el rendimiento de los endpoints en el sistema, fundamentales para el éxito del proyecto.
- Desplegar el *backend* en un entorno en la nube, como Linode, brinda diversos beneficios, entre ellos escalabilidad, flexibilidad y disponibilidad. Al aprovechar los recursos avanzados que ofrece la nube, se facilita la gestión y el monitoreo del *backend*, lo cual resulta en una solución robusta y escalable.

5 RECOMENDACIONES

En esta sección del documento, se presentan todas las recomendaciones que han surgido durante el desarrollo del *backend* como parte de este Trabajo de Integración Curricular:

- Se recomienda tener en cuenta la importancia de adoptar una estructura modular en el proyecto para mejorar la comprensión y gestión del código. Dividir el código en módulos coherentes y organizar los directorios de manera clara fomentará la reutilización de componentes y, en última instancia, contribuirá a una mayor facilidad de mantenimiento del proyecto.
- Se recomienda dedicar tiempo y esfuerzo al diseño del modelo de base de datos relacional desde el inicio del proyecto. Esto implica un análisis profundo de los requisitos recopilados para identificar entidades, atributos y relaciones relevantes. Un diseño sólido de la base de datos es fundamental para una gestión eficaz de la información.
- Se recomienda encarecidamente definir con claridad los roles y permisos en el sistema. Además, se subraya la importancia de implementar de manera adecuada la autenticación y autorización, así como las funcionalidades de seguridad necesarias. Estas medidas son esenciales para garantizar una implementación exitosa y segura del proyecto.
- Se recomienda realizar copias de seguridad de la base de datos de forma regular, ya sea de manera semanal o mensual. Esto se debe a que la información de los clientes es de vital importancia, y contar con copias de seguridad programadas garantiza la posibilidad de recuperarla de manera eficiente en caso de pérdida.

6 REFERENCIAS BIBLIOGRÁFICAS

- [1] I. Jiménez. "Licorería la nenita". Facebook. <https://www.facebook.com/profile.php?id=100063883096253&mibextid=ZbWKwL>
- [2] Agora Servicios Web Tecnologicos. "Las tiendas online y su auge". Posicionamiento web. [https://posicionamientoweb-seo.net/las-tiendas-online-y-su-auge/#:~:text=¿Por%20qué%20el%20auge%20la%20tiendas%20online?%20Como,virtuales%20están%20en%20un%20excelente%20momento%20y%20creciendo:\(accedido el 19 de mayo de 2023\).](https://posicionamientoweb-seo.net/las-tiendas-online-y-su-auge/#:~:text=¿Por%20qué%20el%20auge%20la%20tiendas%20online?%20Como,virtuales%20están%20en%20un%20excelente%20momento%20y%20creciendo:(accedido%20el%2019%20de%20mayo%20de%202023).)
- [3] M. Berenstein. "Auge de las tiendas en línea ¿por qué son tan importantes? - Emprendedores News". Emprendedores News. <https://emprendedoresnews.com/ecommerce/auge-de-las-tiendas-en-linea-por-que-son-tan-importantes.html> (accedido el 16 de mayo de 2023).
- [4] M. A. "La digitalización de las tiendas físicas ante el auge del 'e-commerce'". El País. <https://elpais.com/economia/estar-donde-estes/2020-09-08/la-digitalizacion-de-las-tiendas-fisicas-ante-el-auge-del-e-commerce.html> (accedido el 16 de mayo de 2023).
- [5] M. Mulford, L. Vergara y D. Plata de Plata, "Tienda virtual: social market Colombia", Multiciencias, vol. 14, n.º 3, p. 273, 2014.
- [6] M. de la Maza. "Bsale Chile". Bsale Chile. <https://www.bsale.cl/article/que-es-el-ecommerce-beneficios-de-tener-una-tienda-en-linea> (accedido el 19 de mayo de 2023).
- [7] F. Machuca. "🖥️ ¿Qué es Backend y para qué sirve en programación?" <https://www.crehana.com>. <https://www.crehana.com/blog/transformacion-digital/que-es-el-Backend-y-como-usarlo/> (accedido el 18 de mayo de 2023).
- [8] "¿Qué es el desarrollo de software?" Desarrollo de software. <https://desarrollodesoftware.dev/> (accedido el 17 de mayo de 2023).
- [9] Certus. "Descubre en qué es el desarrollo de software | Certus". Certus Blog | Carreras Técnicas Profesionales. <https://www.certus.edu.pe/blog/consiste-desarrollo-software/> (accedido el 17 de mayo de 2023).
- [10] "Qué es python". Home de DesarrolloWeb.com. <https://desarrolloweb.com/articulos/1325.php> (accedido el 18 de mayo de 2023).
- [11] "FastAPI". FastAPI. <https://fastapi.tiangolo.com/es/> (accedido el 18 de mayo de 2023).

- [12] "Json". JSON. <https://www.json.org/json-es.html> (accedido el 18 de mayo de 2023).
- [13] "JWT.IO - JSON web tokens introduction". JSON Web Tokens - jwt.io. <https://jwt.io/introduction> (accedido el 18 de mayo de 2023).
- [14] "¿Qué es una API de REST?" Red Hat - We make open source technologies for the enterprise. <https://www.redhat.com/es/topics/api/what-is-a-rest-api> (accedido el 18 de mayo de 2023).
- [15] C. D. Alcolea. "Qué es un ORM". OpenWebinars.net. <https://openwebinars.net/blog/que-es-un-orm/> (accedido el 19 de mayo de 2023).
- [16] "¿Qué es una base de datos relacional?" Oracle | Cloud Applications and Cloud Platform. <https://www.oracle.com/mx/database/what-is-a-relational-database/> (accedido el 19 de mayo de 2023).
- [17] "Que es linode: Eficiencia y accesibilidad en la nube". Expertos en Hostings. <https://expertosenhostings.one/linode/> (accedido el 19 de mayo de 2023).
- [18] C. Singureanu. "¡Backend testing - tipos, proceso, herramientas y más!" ZAPTEST. <https://www.zaptest.com/es/pruebas-de-Backend-profunda-inmersion-en-lo-que-es-sus-tipos-procesos-enfoques-herramientas-y-mas> (accedido el 19 de mayo de 2023).
- [19] "Estudio de caso: Concepto, características, cómo hacerlo, ejemplos". Lifeder. <https://www.lifeder.com/estudio-caso/> (accedido el 26 de mayo de 2023).
- [20] I. R. Salvador. "Estudio de caso: Características, objetivos y metodología". Psicología y Mente. <https://psicologiymente.com/psicologia/estudio-de-caso> (accedido el 26 de mayo de 2023).
- [21] VGS. Tech solutions. "Metodologías de desarrollo de software: ¿En qué consisten?" Creamos productos digitales para un mundo en constante cambio. <https://vgst.net/blog/development/metodologias-de-desarrollo-de-software> (accedido el 19 de mayo de 2023).
- [22] Atlassian. "¿Qué es ágil? | Atlassian". Atlassian. <https://www.atlassian.com/es/agile> (accedido el 19 de mayo de 2023).
- [23] Miguel Ángel De Dios. "Scrum: Qué es y cómo funciona este marco de trabajo". Digital Marketing Agency and Consultancy Online | WAM. <https://www.waemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html> (accedido el 19 de mayo de 2023).

- [24] "Los tres principales roles en Scrum - Proyectum". Proyectum. <https://www.proyectum.com/sistema/blog/los-tres-principales-roles-en-scrum/> (accedido el 21 de febrero de 2023).
- [25] "Artefactos Scrum ¿Qué son y para qué sirven? - Viewnext". Viewnext. <https://www.viewnext.com/artefactos-scrum/> (accedido el 21 de febrero de 2023).
- [26] "Recopilación de requisitos - Proyectum". Proyectum. <https://www.proyectum.com/sistema/blog/recopilacion-de-requisitos/> (accedido el 21 de febrero de 2023).
- [27] M. Rehkopf. "Historias de usuario | Ejemplos y plantilla | Atlassian". Atlassian. <https://www.atlassian.com/es/agile/project-management/user-stories> (accedido el 21 de febrero de 2023).
- [28] Ealde. "Que es el product backlog y el sprint backlog en scrum". EALDE Business School. <https://www.ealde.es/product-backlog-sprint-backlog/> (accedido el 21 de febrero de 2023).
- [29] Apiumhub. "Importancia de la arquitectura de software | Apiumhub". Apiumhub. <https://apiumhub.com/es/tech-blog-barcelona/arquitectura-de-software/> (accedido el 19 de mayo de 2023).
- [30] R. D. Hernandez. "El patrón modelo-vista-controlador: Arquitectura y frameworks explicados". freeCodeCamp.org. <https://www.freecodecamp.org/espanol/news/el-modelo-de-arquitectura-view-controller-pattern/> (accedido el 19 de mayo de 2023).
- [31] "¿Qué son herramientas de desarrollo? | El Mundo Infinito". El Mundo Infinito. <https://bit.ly/42iHExy> (accedido el 19 de mayo de 2023).
- [32] "PyCharm: The python IDE for professional developers by jetbrains". JetBrains. <https://www.jetbrains.com/pycharm/> (accedido el 19 de mayo de 2023).
- [33] "¿QUE ES GITLAB y GITHUB?" Stack Overflow en español. <https://bit.ly/45AXXZ6> (accedido el 19 de mayo de 2023).
- [34] "DataGrip: PostgreSQL GUI tool". JetBrains. <https://bit.ly/3qhPdqb> (accedido el 19 de mayo de 2023).
- [35] "Docker: Accelerated, containerized application development". Docker. <https://www.docker.com/> (accedido el 19 de mayo de 2023).
- [36] "Uvicorn". Uvicorn. <https://www.uvicorn.org/> (accedido el 27 de mayo de 2023).

- [37] "Psycopg2-binary". PyPI. <https://pypi.org/project/psycopg2-binary/> (accedido el 27 de mayo de 2023).
- [38] "SQLAlchemy". SQLAlchemy - The Database Toolkit for Python. <https://www.sqlalchemy.org/> (accedido el 27 de mayo de 2023).
- [39] "¿Qué es Python dotenv? | KeepCoding Bootcamps". KeepCoding Bootcamps. <https://bit.ly/45zdA3d> (accedido el 27 de mayo de 2023).
- [40] "Welcome to Alembic's documentation! — Alembic 1.11.1 documentation". Welcome to Alembic's documentation! — Alembic 1.11.1 documentation. <https://alembic.sqlalchemy.org/en/latest/> (accedido el 27 de mayo de 2023).
- [41] "Passlib.hash.bcrypt - bcrypt — passlib v1.7.4 documentation". Passlib 1.7.4 documentation — Passlib v1.7.4 Documentation. <https://passlib.readthedocs.io/en/stable/lib/passlib.hash.bcrypt.html> (accedido el 27 de mayo de 2023).
- [42] "Python-jose". PyPI. <https://pypi.org/project/python-jose/> (accedido el 27 de mayo de 2023).
- [43] "Python-multipart". PyPI. <https://pypi.org/project/python-multipart/> (accedido el 27 de mayo de 2023).
- [44] "Pydantic". Pydantic. <https://pydantic.dev/> (accedido el 27 de mayo de 2023).
- [45] "Regex o expresiones regulares: La manera más sencilla de describir secuencias de caracteres". IONOS Digital Guide. <https://bit.ly/3MGDiKC> (accedido el 27 de mayo de 2023).
- [46] "Nameparser". PyPI. <https://pypi.org/project/nameparser/> (accedido el 27 de mayo de 2023).
- [47] Tamushi. "Pruebas unitarias de software: Definición, características y ventajas". Testing IT. <https://www.testingit.com.mx/blog/pruebas-unitarias-de-software> (accedido el 15 de julio de 2023).
- [48] Tamushi. "¿Deberías implementar pruebas de compatibilidad en tus aplicaciones móviles?" Testing IT. <https://www.testingit.com.mx/blog/pruebas-de-compatibilidad#:~:text=Las%20pruebas%20de%20compatibilidad%20de%20software%20son%20una%20herramienta%20para,operativo%20y%20el%20navegador%20utilizado.> (accedido el 15 de julio de 2023).

[49] Tamushi. "Pruebas de estrés de software: ¿qué son y para qué sirven?" Testing IT. <https://www.testingit.com.mx/blog/pruebas-de-estres-de-software> (accedido el 15 de julio de 2023).

[50] Tamushi. "Pruebas de aceptación de software, ¿Cuándo y por qué son necesarias?" Empresa de pruebas de software | Testing IT. <https://www.testingit.com.mx/blog/pruebas-de-aceptacion-de-software> (accedido el 3 de agosto de 2023).