

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

**DESARROLLO DE SISTEMA DE GESTIÓN DE MUDANZAS EN
QUITO**

DESARROLLO DE UN BACKEND

**TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO
SUPERIOR EN DESARROLLO DE SOFTWARE**

JHAEL DARIO NICOLALDE MOLINA
jhael.nicolalde@epn.edu.ec

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

DMQ, agosto 2023

CERTIFICACIONES

Yo, **NICOLALDE MOLINA JHAEL DARIO** 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.

NICOLALDE MOLINA JHAEL DARIO

jhael.nicolalde@epn.edu.ec

Certifico que el presente trabajo de integración curricular fue desarrollado por **NICOLALDE MOLINA JHAEL DARIO**, 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.

NICOLALDE MOLINA JHAEL DARIO

DEDICATORIA

Deseo expresar mi más sincero reconocimiento y gratitud a mi familia, cuyo apoyo ha sido el pilar fundamental en la realización de este proyecto. Su confianza en mi capacidad y su incesante aliento han sido un factor determinante en la consecución de mis aspiraciones.

También cabe destacar a mis apreciados amigos y compañeros de estudio, cuya amistad, constante motivación y apoyo han sido un factor esencial que ha enriquecido tanto mi trayectoria académica como mi vida social.

Asimismo, deseo extender mi agradecimiento a mi tutor de tesis, cuya orientación experta, paciencia y generosidad al compartir su vasto conocimiento y experiencia ha sido esencial para encaminar este proyecto hacia el éxito y enriquecer mi formación académica. Sin su respaldo incansable, este logro no habría sido posible.

JHAEL DARIO NICOLALDE MOLINA

AGRADECIMIENTO

Deseo expresar mi profundo agradecimiento a todas aquellas personas que me han ofrecido su colaboración a lo largo de este desafiante proceso. A mi familia, amigos e instructores, cuya orientación y respaldo en diferentes fases de mi formación académica han sido invaluableles.

JHAEL DARIO NICOLALDE MOLINA

ÍNDICE DE CONTENIDO

CERTIFICACIONES.....	I
DECLARACIÓN DE AUTORÍA.....	II
DEDICATORIA.....	III
AGRADECIMIENTO.....	IV
ÍNDICE DE 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.....	3
2 METODOLOGÍA.....	4
2.1 Metodología de Desarrollo.....	7
Roles.....	8
Artefactos.....	9
2.2 Diseño de arquitectura.....	12
Arquitectura Modelo Vista Controlador.....	12
2.3 Herramientas de desarrollo.....	13
3 RESULTADOS.....	15
3.1 Sprint 0. Configuración del ambiente de desarrollo.....	15
Definición de requerimientos y restricciones para el <i>backend</i>	15
Diseño e implementación de tablas y relaciones para la Base de datos SQL.....	18
Definición de archivos, directorios y configuraciones para el proyecto <i>backend</i>	19
Definición de módulos por cada rol de usuario.....	20
3.2 Sprint 1. Resultados del desarrollo de endpoints para el perfil administrador...	21
Inicio de sesión, cierre de sesión y recuperación de contraseña.....	22
Modificación de perfil de usuario.....	23
Gestión de categorías.....	23
Gestión de servicios.....	23
Gestión de pedidos.....	25
Asignación de pedidos.....	27

Visualización de estado del pedido	28
Gestion de comentarios	29
3.3 Sprint 2. Resultados del desarrollo de endpoints para el perfil empleado	30
Revisar asignación del pedido	31
Finalizar entrega del pedido.....	32
3.4 Sprint 3. Resultados del desarrollo de endpoints para el perfil cliente	33
Visualizar servicios	34
Contratar servicios.....	35
Envío de comentarios y/o sugerencias	37
3.5 Sprint 4 Ejecución de pruebas para el componente <i>backend</i>	38
Resultados de pruebas unitarias del <i>backend</i>	38
Resultados de pruebas de compatibilidad del <i>backend</i>	40
Resultados de pruebas de carga del <i>backend</i>	40
Resultados de pruebas de aceptación del <i>backend</i>	41
Despliegue del <i>backend</i>	42
4 CONCLUSIONES	44
5 RECOMENDACIONES.....	46
6 REFERENCIAS BIBLIOGRÁFICAS	47

RESUMEN

Hoy en día, se evidencia una carencia de aplicaciones que faciliten a los residentes de Quito encontrar servicios especializados en el ámbito de mudanzas. Además, no existe una plataforma que presente un catálogo de los diferentes servicios que son ofertados por profesionales especializados en este sector. Sin embargo, surge otro desafío la necesidad de acudir personalmente a lugares físicos para coordinar las mudanzas, lo que puede conllevar inconvenientes como pérdida de tiempo, posibilidad de robos, tarifas excesivas y trato con personal inexperto, generando preocupaciones sobre la calidad del servicio ofertado.

La conjunción de sistemas web que están contruidos por un backend y frontend brindan múltiples ventajas para impulsar el crecimiento de los servicios en línea en el contexto de mudanzas en Quito. Esta transformación digital puede impactar positivamente en la sociedad, otorgando nuevas fuentes de empleo a profesionales en el área y proporcionar herramientas como catálogos digitales, acceso conveniente en cualquier lugar y momento, pagos seguros y la facilidad en la búsqueda de servicios. En ese sentido y en pro de brindar asistencia a la población de Quito, se ha desarrollado un proyecto de Integración Curricular que presenta el desarrollo de un backend de un sistema de gestión de mudanzas. Proporcionando varios *endpoints* que permiten a las empresas a promocionar sus servicios a través de cualquier aplicación del lado del cliente. Además, el backend asegura que los clientes puedan contratar servicios de manera segura, estableciendo las condiciones y garantías adecuadas para garantizar la satisfacción del cliente.

Por último, este documento se encuentra estructurado en cinco secciones, cada una detallando secciones clave. En la primera sección, se presenta la problemática a resolver, objetivos específicos, alcance y el marco teórico correspondiente. En la segunda sección se presenta la metodología que se ha utilizado en el desarrollo del proyecto, herramientas, patrón de arquitectura y roles. A continuación, en la tercera sección, se presentan cada uno de los logros y resultados que se han obtenido en cada una de las iteraciones. Finalmente, se presenta las respectivas conclusiones y recomendaciones derivadas de todo el trabajo de Integración Curricular.

PALABRAS CLAVE: Laravel, Mudanzas, Endpoint, Backend, Transporte.

ABSTRACT

Today, there is a lack of applications that make it easier for Quito residents to find specialized services in the field of removals. In addition, no platform presents a catalog of the different services that are offered by professionals specialized in this sector. However, another challenge arises: the need to personally go to physical locations to coordinate removals, which can lead to inconveniences such as loss of time, the possibility of theft, excessive fees, and dealing with inexperienced personnel, raising concerns about the quality of the service offered.

The conjunction of web systems that are built by a backend and frontend provide multiple advantages to boost the growth of online services in the context of removals in Quito. This digital transformation can have a positive impact on society, granting new sources of employment to professionals in the area and providing tools such as digital catalogs, convenient access anywhere and at any time, secure payments, and the ease of searching for services. In this sense and in favor of assisting the population of Quito, a Curricular Integration project has been developed that presents the development of a backend of a removal management system. Providing various endpoints that allow companies to promote their services through any client-side application. In addition, the backend ensures that customers can contract services safely, establishing the appropriate conditions and guarantees to guarantee customer satisfaction.

Finally, this document is structured into five sections, each detailing key sections. In the first section, the problem to be solved, specific objectives, scope, and the corresponding theoretical framework are presented. The second section presents the methodology that has been used in the development of the project, tools, architecture pattern, and roles. Next, in the third section, each of the achievements and results that have been obtained in each of the iterations are presented. Finally, the respective conclusions and recommendations derived from all the Curricular Integration work are presented.

KEYWORDS: Laravel, Removals, Endpoint, Backend, Transportation

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

En el Distrito Metropolitano de Quito, según datos proporcionados por el Municipio residen aproximadamente 2.644.145 personas. Es importante destacar que entre estas personas la mayoría se enfrentan al proceso de cambio de domicilio al menos una vez en su vida, ya sea al mudarse de una casa o departamento. Este proceso de reubicación suele conllevar diversos desafíos y dificultades que es necesario afrontar [1]. En ese sentido, se ha identificado una necesidad en la comunidad, ya que la mayoría de los ciudadanos no cuentan con un medio tecnológico seguro y confiable que les brinde información sobre los servicios de transporte de mudanza, precios, tipo de transporte, ubicación, personal, guía para el traslado de bienes muebles, entre otros. Lo que ha ocasionado que los usuarios contraten servicios de manera informal o en ciertas ocasiones hagan uso de sus propios vehículos para llevar los bienes inmuebles exponiéndose a multas ya que no disponen la guía para el traslado de bienes muebles, el cual debe ser tramitado en la Unidad de Policía Comunitaria (UPC) [2].

En la actualidad, el sector de las mudanzas ha experimentado una evolución significativa debido al crecimiento del comercio electrónico y la globalización. Estos factores han generado una mayor demanda de servicios de mudanzas, lo que ha impulsado la necesidad de soluciones tecnológicas eficientes y escalables [3]. Además, el brindar servicios a través de plataformas digitales en Internet aumenta las ventas al ofrecer una mayor accesibilidad hacia los clientes, quienes pueden ver toda la información del negocio en cualquier momento desde diferentes dispositivos tecnológicos. Los sistemas web son esenciales en el día a día ya que proporcionan diferente tipo de información, permiten realizar compra y venta de productos o servicios las 24 horas. Por otra parte, el período de vida de cualquier sistema *software* aproximadamente es de 5 años, por esta razón la velocidad con la que cada año se desarrollan nuevas y mejores tecnologías vuelve al mundo del desarrollo de *software* extremadamente evolutivo y cambiante [4].

Por otra parte, según información obtenida de la Agencia de Regulación y Control de las Telecomunicaciones (ARCOTEL) en la actualidad, el uso de la tecnología es crucial ya que permite acceder a información en tiempo real, automatizar procedimientos y procesos. En un mundo donde los usuarios demandan rapidez, simplicidad y conexión constante, las aplicaciones móviles y sistemas web son capaces de satisfacer estas necesidades [5].

Por lo citado anteriormente, el propósito del presente Trabajo de Integración Curricular es dotar a la ciudadanía el desarrollo de un *backend* como parte de un sistema de gestión, el cual les permita contratar servicios de transporte para mudanzas y ofrecer soluciones

tecnológicas para facilitar y optimizar el proceso de traslado de bienes y objetos de forma segura. Adicional a ello, la gestión de la información se realiza desde un sistema web, el cual se encuentra dividido en dos componentes: *backend* y *frontend*, permitiendo de esta manera una mejor organización, seguridad de la información y escalabilidad gracias al uso de nuevas tecnologías emergentes de desarrollo de *software*.

1.1 Objetivo general

Desarrollar un *backend* de sistema de gestión de mudanzas en Quito.

1.2 Objetivos específicos

1. Identificar los requerimientos para el *backend*.
2. Diseñar e implementar una estructura de Base de datos relacional para que el *backend* cumpla con los requerimientos que se han obtenido.
3. Diseñar la arquitectura para la interfaz de programación de aplicaciones (*APIs*) *RESTful* según los requerimientos que se han planteado.
4. Codificar los recursos del *backend*.
5. Verificar el funcionamiento sometiendo a una serie de pruebas.
6. Desplegar el *backend* para su utilización.

1.3 Alcance

Este Trabajo de Integración Curricular propone desarrollar un *backend* que le permita al usuario con perfil administrador gestionar (crear, listar, actualizar y eliminar) y atender (aceptar y rechazar) los servicios de transporte de mudanza que va a ofertar, mientras que los clientes desde una aplicación del lado del cliente pueden visualizar y acceder a los diferentes tipos de servicios y contratar el que mejor se adapte a sus necesidades. Ahora bien, para lograr lo antes mencionado se utilizan herramientas modernas de desarrollo de *software* que garanticen la robustez y escalabilidad del *backend* y sus futuras integraciones. Además, se implementa la metodología ágil *Scrum* para cumplir con los objetivos, se utiliza un patrón de arquitectura para la codificación de cada una de los recursos que son accedidos a través de *endpoints*, para luego realizar una serie de pruebas para asegurar el correcto funcionamiento de todos los recursos del *backend* y, por último, se realiza el despliegue a producción. Además, se prioriza la seguridad y privacidad de la información mediante diversas medidas de protección y el cumplimiento de normativas

vigentes. A continuación, se presenta cada uno de los perfiles que intervienen en todo el desarrollo del *backend*:

Perfiles que se establecen:

- Administrador.
- Empleado.
- Cliente.

Para el perfil administrador se generan:

- *Endpoints* que permiten iniciar sesión, cerrar sesión y modificar contraseña.
- *Endpoints* para registrar empleados.
- *Endpoints* para gestionar las categorías.
- *Endpoints* para gestionar servicios.
- *Endpoints* para gestionar pedidos.
- *Endpoints* para asignación de pedidos a empleado.
- *Endpoints* para visualización del estado de los pedidos.
- *Endpoints* para gestión de comentarios y/o sugerencias.

Para el perfil empleado se generan:

- *Endpoints* que permiten iniciar sesión, cerrar sesión y modificar contraseña.
- *Endpoints* para revisar la asignación del pedido.
- *Endpoints* para la finalización del pedido.

Para el perfil cliente se generan:

- *Endpoints* que permiten registrarse, iniciar sesión, cerrar sesión y modificar contraseña.
- *Endpoints* de visualización de servicios.
- *Endpoints* para contratación de un servicio.
- *Endpoints* para visualización el detalle del servicio.
- *Endpoints* para envío de comentarios y/o sugerencias.

- *Endpoints* para solicitar cotización.

1.4 Marco teórico

En el ámbito del *software* el desarrollo es un proceso mediante el cual se crea, diseña y programa varias instrucciones y algoritmos para permitir que un sistema informático realice diferentes tareas específicas y cumpla con determinadas funcionalidades. Este proceso se inicia desde la concepción inicial de la idea y la asignación de requisitos al proyecto, para luego realizar la respectiva arquitectura del *software* planificando como se realizará y que diseño se implementará, para luego realizar pruebas y mantenimiento al mismo. El objetivo primordial del desarrollo de *software* es generar aplicaciones o sistemas informáticos eficientes, confiables y que cumplan con las necesidades del usuario [6].

Para administrar toda la información y gestionar la lógica de un sistema web se desarrolla el *backend*, también conocido como aplicación del lado del servidor, el cual se encarga de procesar las solicitudes del usuario, lógica empresarial y manejar los datos subyacentes. Además, este componente es responsable de la autenticación, autorización de usuarios, el intercambio de comunicación, así como de proporcionar las respuestas adecuadas al *frontend*. Por otra parte, el *backend* se desarrolla utilizando una variedad de lenguajes de programación, *Frameworks* y tecnologías que permiten su implementación y garantizan un rendimiento eficiente [7].

El *Framework* Laravel es utilizado ampliamente en el desarrollo de sistemas de tipo web, basado en el lenguaje PHP y posee una estructura fácil de usar y robusta. Además, se considera uno de los *Frameworks* más usados en la actualidad [5]. Utiliza una elegante sintaxis y amplia gama de características, además, simplifica el proceso de desarrollo y promueve una programación limpia y eficiente. En el caso de la aplicación de mudanzas, Laravel se utiliza como el marco principal para gestionar las rutas, vistas y controladores necesarios para manejar las solicitudes del usuario y proporcionar respuestas adecuadas [8].

Una API RESTful, acrónimo de Interfaz de Programación de Aplicaciones Representacional de Estado Transferencia, es un estilo de arquitectura para el desarrollo de servicios web que se basa en los principios del protocolo HTTP. Permite la creación, modificación y consulta de recursos a través de operaciones estándar como GET, POST, PUT y DELETE. Además, utiliza la estructura de URL y los códigos de estado HTTP para comunicarse de manera eficiente con otras aplicaciones y sistemas. Proporcionando una interfaz clara y flexible que permite a los desarrolladores interactuar con los recursos de manera uniforme y escalable [9].

Las Bases de datos se pueden dividir en dos categorías distintas, en el caso del presente proyecto se hace uso del tipo relacional, puesto que por medio de esta estructura toda la información se organiza de forma de tablas interrelacionadas entre sí, por otra parte, los datos se encuentran de manera que se estructuran en filas y columnas, además, las conexiones entre las tablas se fundamentan a través de claves externas y primarias. Este enfoque permite establecer y mantener relaciones consistentes entre los datos, lo que facilita la consulta, administración y modificación de los datos guardados [10].

Un gestor es un sistema diseñado con el objetivo de manejar y manipular de manera eficiente, sencilla y de forma rápida las Bases de datos, en este caso el gestor MySQL se encarga de manejar datos de tipo relacional, es utilizado ampliamente en aplicaciones web y ofrece una alta confiabilidad, rendimiento y escalabilidad [9], por lo que se posiciona como una de las alternativas más destacadas para almacenar y gestionar datos en una aplicación *web* [11].

Eloquent es el ORM (Mapeo Objeto-Relacional) incorporado en Laravel, el cual proporciona una forma elegante y sencilla de manipular la Base de datos utilizando modelos y relaciones entre ellos. Además, simplifica las operaciones de escritura y lectura con la respectiva Base de datos [12].

Las migraciones son una herramienta que permite gestionar y mantener una estructura controlada y eficiente en la información de una aplicación *software*. Con las migraciones, se puede realizar una manipulación programática de las estructuras de tablas en los datos de la aplicación, en vez de efectuar modificaciones directas en los datos del sistema. Esto proporciona un enfoque versionado y flexible para gestionar el progreso y desarrollo a medida que la aplicación avanza [13].

Los *seeders* y *factories* en Laravel son herramientas que permiten la creación y población de datos de prueba en los datos de un sistema. Los *seeders* son clases utilizadas para insertar datos estáticos en la base de datos, como registros iniciales, datos de configuración o datos de prueba predefinidos. Por otro lado, los *factories* son clases que generan datos aleatorios y realistas para poblar la Base de datos con registros de prueba [14].

Postman es una herramienta de colaboración que permite probar, documentar y realizar solicitudes a APIs (Interfaz de Programación de Aplicaciones) de manera eficiente. Con su interfaz intuitiva facilita la validación utilizando solicitudes HTTP y verificar las respuestas proporcionadas por un *backend*, asegurando así la correcta funcionalidad de los servicios *web* implementados [15].

PHP es un lenguaje utilizado en gran parte en el desarrollo de aplicaciones *web*. Su capacidad para integrarse con HTML y su amplia comunidad de desarrollo hacen de PHP una elección popular para el desarrollo de aplicaciones *web* [16].

Las pruebas en el ámbito del *software* conllevan una serie de actividades y técnicas utilizadas para verificar y validar el correcto funcionamiento de una aplicación o sistema. Además, estas pruebas se llevan a cabo con el objetivo de identificar errores, fallas y comportamientos inesperados, garantizando así la calidad y confiabilidad del *software*. Por otra parte, existen diferentes tipos de pruebas, como las pruebas de integración, unitarias, de sistema y aceptación, cada una enfocada en validar aspectos específicos del *software* [17].

2 METODOLOGÍA

El enfoque de investigación se fundamenta basándose principalmente en el análisis y la recopilación de información relevante para comprender los requisitos, objetivos y restricciones del proyecto. En este contexto, el enfoque se centra en investigar y comprender a fondo las tecnologías, herramientas y prácticas relacionadas con el desarrollo. Esto implica investigar las mejores prácticas de programación, patrones de diseño de *software*, tecnologías de seguridad, Bases de datos y la optimización del rendimiento, entre otros aspectos. Además, es importante investigar las preferencias y necesidades que tiene el usuario al momento de interactuar con el sistema, así como también las tendencias y avances actuales en el desarrollo *backend*, para garantizar un desarrollo efectivo [18].

Por lo antes expuesto, se utiliza un estudio de casos de tipo sistemático y estructurado para planificar, diseñar e implementar el *backend* como parte del presente proyecto. Esto implica la aplicación de metodologías de desarrollo ágil, para iterar de manera incremental y colaborativa en el desarrollo del *software*. Además, técnicas de pruebas y validación para asegurar la calidad y el correcto funcionamiento del *backend*.

2.1 Metodología de desarrollo

Las metodologías que se reconocen como enfoques de desarrollo ágiles son aquellas cuando utilizan enfoques iterativos e incrementales utilizados en el desarrollo de *software*, los cuales se caracterizan por poner énfasis en la colaboración, la adaptabilidad y la entrega temprana de valor al cliente. Estas metodologías fomentan la flexibilidad y adaptación a cambios en los requerimientos del proyecto, basándose en equipos multidisciplinarios y autoorganizados que trabajan en ciclos cortos de desarrollo, donde se entregan incrementos funcionales del *software* de manera regular [19].

Una de tantas metodologías ágiles que es enfocada principalmente en el desarrollo de aplicaciones *software* es la metodología *Scrum*, la cual está enfocada en la entrega de valor de forma incremental e iterativa. Además, se basa en equipos autoorganizados y multidisciplinarios que trabajan en *Sprints*, que son ciclos de desarrollo de corta duración. Por otra parte, emplea elementos como el listado de tareas pendientes del producto, el listado de tareas pendientes del *Sprint* y el resultado del producto, además de actividades como reuniones diarias de seguimiento y revisiones para supervisar el *Sprint* garantizando de esta manera la transparencia y una comunicación continua [20].

Por lo descrito anteriormente, se ha realizado la implementación en el presente proyecto la metodología *Scrum*, ya que, como enfoque ágil, promueve la flexibilidad, adaptabilidad y la entrega temprana de los respectivos avances. Adicionalmente, *Scrum* promueve una colaboración estrecha entre los integrantes del equipo y los *stakeholders*, lo cual facilita una comunicación continua y asegura que el desarrollo del *backend* se alinee adecuadamente con las necesidades reales del proyecto. Por último, se presenta la implementación de cada una de las fases en el desarrollo del proyecto.

Roles

En la metodología *Scrum* que se utiliza para el presente proyecto, intervienen tres roles principales que desempeñan funciones específicas dentro del equipo de desarrollo, estos roles son.

Product Owner

Es la persona con la responsabilidad de gestionar el *backlog* del producto y optimizar al máximo su valor. Su rol implica definir y priorizar los requisitos, establecer la visión del producto y tomar decisiones sobre qué funcionalidades se desarrollarán [21]. En la **Tabla 2.1** se indica la asignación de la persona con su rol respectivo en el desarrollo del *backend*.

Scrum Master

Es quien tiene la responsabilidad de garantizar la correcta implementación de *Scrum* y facilitar el proceso. Su rol principal es servir como líder del equipo, asegurándose de que se sigan las reglas y prácticas de *Scrum*, eliminando obstáculos y fomentando la colaboración y la mejora continua. Además, es responsable de organizar y facilitar las reuniones y ceremonias de *Scrum* [22]. En la **Tabla 2.1** se indica la asignación de la persona con su rol respectivo en el desarrollo del *backend*.

Development Team

Está compuesto por diferentes personas multidisciplinarias que son responsables de diseñar, desarrollar, probar y entregar incrementos de *software* funcionales en cada *Sprint*. Su objetivo es colaborar estrechamente con el *Product Owner* para comprender y satisfacer los requisitos del producto y trabajar en incrementos de valor de manera regular [23]. Por esto, en la **Tabla 2.1** se indica la asignación de la persona con su rol respectivo en el desarrollo del *backend*.

Tabla 2.1 Asignación de roles.

ROL	INTEGRANTES
<i>Product Owner</i>	Ing. Byron Loarte, MSc.
<i>Scrum Master</i>	Ing. Byron Loarte, MSc.
<i>Development Team</i>	Jhael Nicolalde

Artefactos

En la metodología *Scrum* son los elementos tangibles que se utilizan para planificar, monitorear y gestionar el desarrollo del proyecto [24]. En el contexto del *backend* los principales artefactos que se han implementado son los siguientes.

Recopilación de requerimientos

Teniendo de base a la metodología *Scrum* la recopilación de requerimientos es el proceso de identificar, documentar y comprender lo que requieren los usuarios. Además, en el contexto del presente proyecto implica identificar los elementos y funcionalidades clave que deben ser implementados para satisfacer las necesidades específicas del negocio y los usuarios, siendo un elemento con mayor importancia para el desarrollo *software* [25]. En la **Tabla 2.2** se indica el respectivo formato que se ha establecido para obtener los requerimientos respectivos al presente proyecto, y en el **ANEXO II** se puede visualizar la recopilación completa.

Tabla 2.2 Tabla de recopilación de requerimientos.

RECOPIACIÓN DE REQUERIMIENTOS		
Tipo de sistema	ID-RR	Enunciado del Ítem
<i>Backend</i>	RR010	Como usuario empleado necesita generar varios <i>endpoints</i> para: <ul style="list-style-type: none"> • Revisar asignación de pedido
	RR011	Como usuario empleado necesita generar varios <i>endpoints</i> para: <ul style="list-style-type: none"> • Finalizar entrega de pedido

Historias de Usuario

Consiste en detallar mediante descripciones concisas y breves las funcionalidades o características de un sistema o aplicación desde la perspectiva de los usuarios con el objetivo de detallar los requerimientos preestablecidos. Estas descripciones se redactan en un lenguaje natural y son utilizadas para comunicar y documentar los requerimientos del proyecto de manera concisa y comprensible [26]. En la **Tabla 2.3** se indica el respectivo formato que se ha establecido para redactar las Historias de usuario respectivas al presente proyecto, y en el **ANEXO II** se puede visualizar todas de manera completa.

Tabla 2.3 Historia de usuario 10 – Revisar asignación de pedido.

HISTORIA DE USUARIO	
Identificador: HU010	Usuario: Empleado
Nombre historia: Revisar asignación de pedido	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Iteración asignada: 4	
Responsable (es): Jhael Nicolalde	
Descripción: El <i>backend</i> permite generar varios <i>endpoints</i> para que el perfil empleado pueda: <ul style="list-style-type: none">• Revisar la asignación de un nuevo pedido	
Observación: El usuario empleado puede acceder a los <i>endpoints</i> mencionados. Además, para poder ver la asignación de un nuevo pedido tiene que iniciar sesión.	

Product Backlog

Este artefacto consiste en una serie de funcionalidades que conciernen al proyecto, así como características, mejoras y correcciones que se desean implementar en un producto en base a las Historias de usuario predefinidas. Además, representa los requisitos del proyecto y sirve como base para el desarrollo del producto *software* [27]. Para esto, en la **Tabla 2.4** se indica el formato que se ha establecido para obtener el *Product Backlog* respectivo al presente proyecto, y en el **ANEXO II** se puede visualizar la tabla completa.

Tabla 2.4: Formato para presentar el *Product Backlog*.

ID – HU	HISTORIA DE USUARIO	ITERACIÓN	ESTADO	PRIORIDAD
HU010	Revisar asignación de pedido	4	Finalizado	Alta
HU011	Finalizar entrega de pedido	4	Finalizado	Media

Sprint Backlog

Se basa en la compilación de una serie de tareas particulares elegidas por parte del *Product Backlog* para ser abordadas durante un *Sprint*. Además, es una representación del trabajo planificado para cada iteración y es generado mediante la colaboración entre las partes involucradas por parte de desarrolladores y el *Product Owner*. Por otra parte, contiene las tareas y elementos que están estrechamente relacionados específicamente con el desarrollo del producto *software* [28]. Para esto, en la **Tabla 2.5** se indica el formato que se ha establecido para obtener el *Sprint Backlog* respectivo al presente proyecto, y en el **ANEXO II** se puede visualizar la tabla completa.

Tabla 2.5 Formato para presentar el *Sprint Backlog*.

ELABORACIÓN DE <i>SPRINT BACKLOG</i>						
ID – SB	NOMBRE	MÓDULO	ID-HU	HISTORIA DE USUARIO	TAREA	TIEMPO ESTIMADO
SB003	Diseño e implementación de <i>endpoints</i> para el usuario administrador	Módulo - Visualización del estado de los pedidos	HU008	Visualizar estado del pedido	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para visualizar el estado del pedido Consulta en la base de datos. 	30H

2.2 Diseño de arquitectura

El diseño de la arquitectura de una aplicación o sistema *software* es fundamental para garantizar un funcionamiento eficiente, escalable y seguro. La arquitectura debe contemplar los requisitos y necesidades específicas, permitiendo la gestión de usuarios, la integración con servicios externos, entre otros aspectos clave. Al definir la arquitectura, se deben considerar elementos como la estructura de capas, selección de tecnologías, implementación de patrones de diseño y la escalabilidad horizontal. Además, un diseño bien pensado y robusto es esencial para asegurar la calidad, rendimiento y mantenibilidad de un sistema *software* [29]. Por lo expuesto anteriormente, se presenta la arquitectura para el desarrollo del *backend*.

Arquitectura Modelo Vista Controlador

La arquitectura MVC es considerada un diseño en el que se manipula todo el *software* por medio de componentes que son Modelo, Vista y Controlador, permitiendo así una mejor organización y modularidad de un sistema o aplicación *software*. En ese sentido, el modelo es responsable de manejar los datos y configurar la lógica que posee la aplicación, mientras que la vista se ocupa de la visualización de la información requerida al usuario y el controlador maneja las interacciones del usuario, coordina la comunicación entre el modelo y la vista [30]. La **Fig. 2.1** muestra el diseño que se ha implementado para el *backend*.

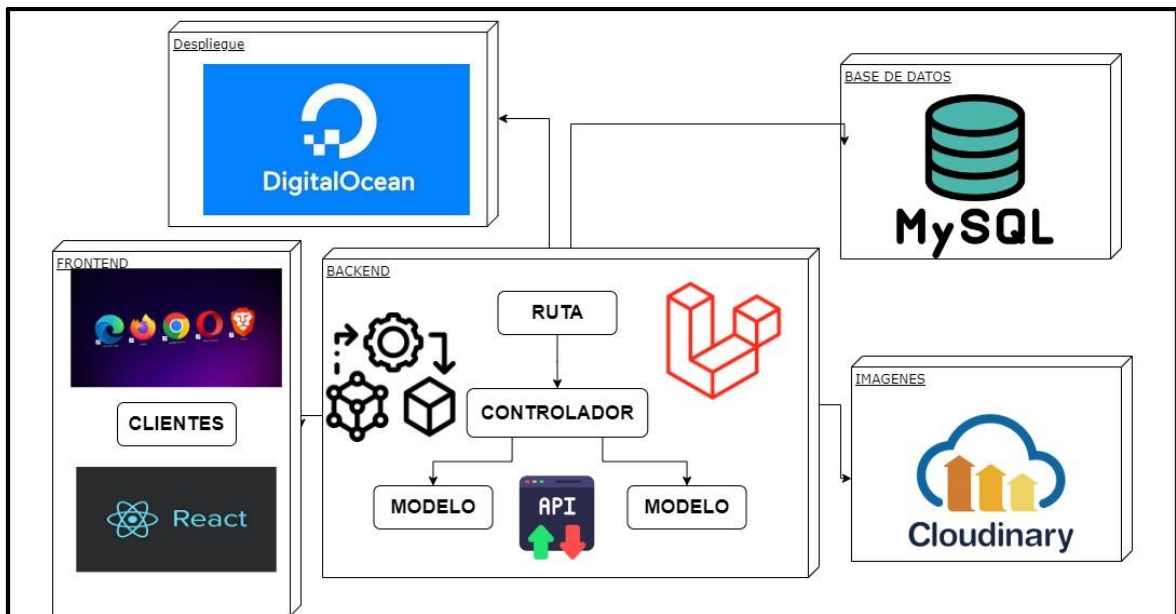


Fig. 2.1: Arquitectura para el *backend*.

2.3 Herramientas de desarrollo

Las herramientas que han sido seleccionadas para la codificación del *backend* permiten el desarrollo de cada uno de los *endpoints* los cuales son accesibles gracias al conjunto de herramientas que son exclusivas para el despliegue de aplicaciones *software*, los cuales permiten gestionar datos mediante consultas e implementando sistemas de contenedores [31]. Por ello, la **Tabla 2.6** presenta el listado de herramientas que se han seleccionado y de qué forma han aportado en el desarrollo del proyecto.

Tabla 2.6: Serie de herramientas que se han empleado para el desarrollo del *backend*.

HERRAMIENTA	JUSTIFICACIÓN
Xampp	Proporciona un entorno de desarrollo completo para crear la aplicación <i>backend</i> en un entorno local y facilita la instalación y configuración de un servidor <i>web</i> local de php [32].
MySQL	Posibilita el almacenamiento estructurado de información y agiliza la ejecución de comandos SQL para facilitar la administración y búsqueda de datos en el <i>backend</i> de la aplicación [33].
Visual Studio Code	Posee una gran utilidad en el progreso del proyecto ya que cuenta con la capacidad de resaltar la sintaxis de php y Laravel y ofrecer sugerencias inteligentes, Visual Studio Code ha permitido escribir un código limpio y libre de errores [34].
GitHub	GitHub desempeña una función fundamental en el <i>backend</i> que se ha desarrollado. Esta plataforma basada en la nube ha proporcionado un control de versiones y subida de cambios de forma sencilla, lo que facilita realizar cambios en el código fuente del <i>backend</i> [35].
Laravel	La inclusión de Laravel en el proyecto ha resultado fundamental para su éxito y funcionalidad. Laravel, como <i>Framework</i> de desarrollo web, ha permitido agilizar y simplificar la creación del <i>backend</i> , brindando una estructura sólida y eficiente para la gestión de

	servicios de mudanzas en la ciudad de Quito [36].
--	---

Librerías

La **Tabla 2.7** presenta el listado de librerías que se han seleccionado y de qué forma han aportado en el proyecto.

Tabla 2.7: Librerías que se han empleado para el *backend*.

LIBRERÍA	DESCRIPCIÓN
Laravel Sanctum	Librería que ofrece autenticación basada en <i>tokens</i> para salvaguardar las rutas de la API [37].
Validator	Librería con reglas de validación predefinidas y adaptable [38].
Request	Librería es esencial para acceder y gestionar los datos de las solicitudes HTTP entrantes [39].
Cloudinary	Esta librería permite acceder a una variedad de funcionalidades y características ofrecidas por Cloudinary, como la carga, almacenamiento y manipulación de imágenes [40].
Faker	La librería "Faker" es una herramienta esencial para la generación de datos de prueba ficticios y generar una amplia variedad de información simulada [41].

3 RESULTADOS

En esta sección, se presentan los objetivos que se han alcanzado durante la fase de desarrollo del *backend*. Además, se describe el diseño, codificación y la implementación de los *endpoints* esenciales para cada tipo de usuario, junto con las pruebas correspondientes y, por último, la puesta en marcha a producción. Es crucial destacar que en cada *Sprint* se enumeran las tareas clave que han sido fundamentales para completar cada iteración.

3.1 *Sprint 0*. Configuración del ambiente de desarrollo

El *Sprint 0* consta de tareas esenciales para preparar el entorno en el que se ha desarrollado, las cuales son las siguientes.

- Establecimiento de requerimientos y limitaciones para el *backend*.
- Organización y diseño de tablas en la Base de datos SQL.
- Configuraciones necesarias en los directorios para el desarrollo del *backend*.
- Definición de los módulos particulares para cada categoría de usuario.

Establecimiento de requerimientos y limitaciones para el *backend*

Diseño e implementación de *endpoints* para el registro

El *backend* incorpora múltiples *endpoints* que posibilitan el registro de usuarios con perfil cliente y empleado. Es importante destacar que el registro del perfil empleado debe ser realizado por un usuario con rol administrador.

Diseño e implementación de *endpoints* de inicio de sesión, cierre de sesión, modificación de perfil y contraseña

Los usuarios que cuentan con los roles de administrador, empleado y cliente pueden acceder al módulo de autenticación a través del componente *backend*. En este sentido, se han habilitado diversos *endpoints* que permiten realizar el inicio de sesión, cierre de sesión, actualización de la información del perfil, incluyendo la imagen de perfil del usuario y el cambio de contraseña. Además, es importante mencionar que el administrador únicamente posee la capacidad de modificar los siguientes datos de perfil: nombre, correo electrónico, contraseña e imagen de perfil.

Diseño e implementación de *endpoints* para la gestión de categorías

Mediante el componente *backend*, se brinda al usuario con perfil administrador la capacidad de acceder al módulo de gestión de las diferentes categorías en las que se clasifican los servicios a ofertar. Para este propósito, se han habilitado diversos *endpoints* que permiten al administrador ver, crear, modificar y eliminar categorías. Cabe destacar que, de forma predeterminada, ya se encuentran 3 categorías creadas previamente.

Diseño e implementación de *endpoints* para gestionar servicios

Mediante el componente *backend*, se brinda al usuario con perfil administrador la capacidad de acceder al módulo de gestión de servicios. Para este propósito, se han habilitado diversos *endpoints* que permiten al administrador ver, crear, modificar y eliminar servicios. Cabe destacar que, de forma predeterminada, ya se encuentran 6 servicios creados previamente.

Diseño e implementación de *endpoints* para gestionar pedidos

Mediante el componente *backend*, se brinda al usuario con perfil administrador la capacidad de acceder al módulo de gestión de pedidos. Para este propósito, se han habilitado diversos *endpoints* que permiten al administrador ver, aceptar o rechazar pedidos.

Diseño e implementación de *endpoints* para asignar pedidos a empleados

Mediante el componente *backend*, se brinda al usuario con perfil administrador la capacidad de acceder al módulo de asignación de pedidos. Para este propósito, se han habilitado diversos *endpoints* que permiten al administrador asignar uno o más pedidos a los empleados.

Diseño e implementación de *endpoints* para visualizar estado del pedido

Mediante el componente *backend*, se brinda a los usuarios con rol administrador, empleado y cliente la capacidad de acceder al módulo para visualizar el estado del pedido. Para este propósito, se han habilitado diversos *endpoints* que permiten al usuario ver los detalles del pedido al que está relacionado.

Diseño e implementación de *endpoints* para gestionar comentarios y/o sugerencias

Mediante el componente *backend*, se brinda al usuario con rol administrador el acceso al módulo de gestión de comentarios y/o sugerencias que se han realizado por el cliente. Para

este propósito, se han habilitado diversos *endpoints* que permiten al administrador ver los comentarios, así como de responder a cada uno de estos.

Diseño e implementación de *endpoints* para revisar asignación del pedido

Mediante el componente *backend*, se brinda al usuario con perfil empleado la capacidad de acceder al módulo de asignación de pedido. Para este propósito, se han habilitado diversos *endpoints* que permiten al empleado ver los pedidos que tiene asignado, así como los detalles de cada uno de ellos.

Diseño e implementación de *endpoints* para finalizar entrega del pedido

Mediante el componente *backend*, se brinda a los usuarios con perfil empleado la capacidad de acceder al módulo de finalizar entrega del pedido. Para este propósito, se han habilitado diversos *endpoints* que permiten al empleado cambiar el estado de los pedidos a completado en el momento de ofertar el servicio.

Diseño e implementación de *endpoints* para visualizar servicios

Mediante el componente *backend*, se brinda a los usuarios con perfil cliente la capacidad de acceder al módulo de visualizar servicios. Para este propósito, se han habilitado diversos *endpoints* que permiten al cliente visualizar los servicios mediante sus categorías.

Diseño e implementación de *endpoints* para contratar servicios

Mediante el componente *backend*, se brinda a los usuarios con perfil cliente la capacidad de acceder al módulo de contratar servicios. Para este propósito, se han habilitado diversos *endpoints* que permiten al cliente crear un pedido para contratar uno o varios servicios.

Diseño e implementación de *endpoints* de visualización de servicios

Mediante el componente *backend*, se brinda a los usuarios con perfil cliente la capacidad de acceder al módulo de visualización de detalles de servicios. Para este propósito, se han habilitado diversos *endpoints* que permiten al cliente visualizar los servicios con sus detalles respectivos.

Diseño e implementación de *endpoints* para enviar comentarios y/o sugerencias

Mediante el componente *backend*, se brinda a los usuarios con perfil cliente la capacidad de acceder al módulo de envío de comentarios y/o sugerencias. Para este propósito, se han habilitado diversos *endpoints* que permiten al cliente enviar comentarios después de haber contratado algún servicio.

A continuación, se muestran en la **Figura 3.1**, **Figura 3.2** y **Figura 3.3** los roles de los usuarios, conjuntamente con los *endpoints* a los que tienen privilegios.

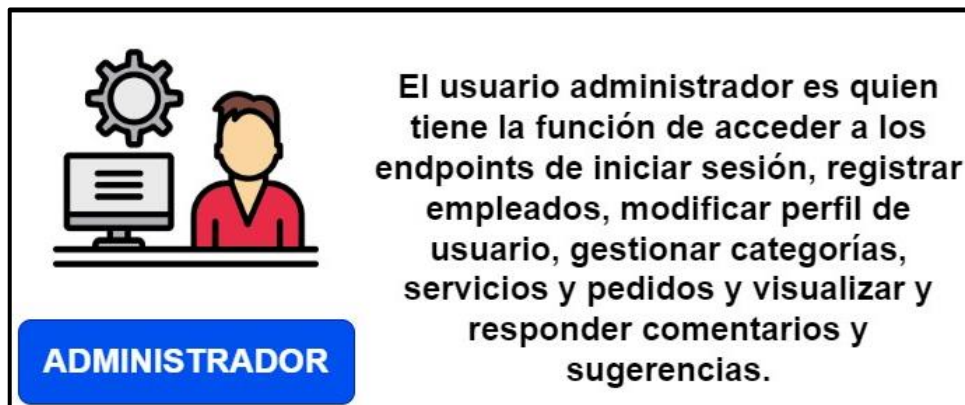


Figura 3.1 Usuario Administrador.

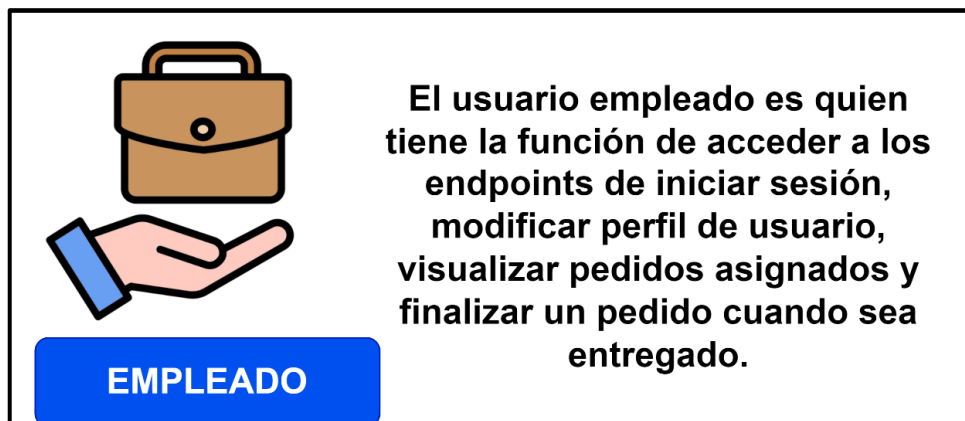


Figura 3.2 Usuario Empleado.

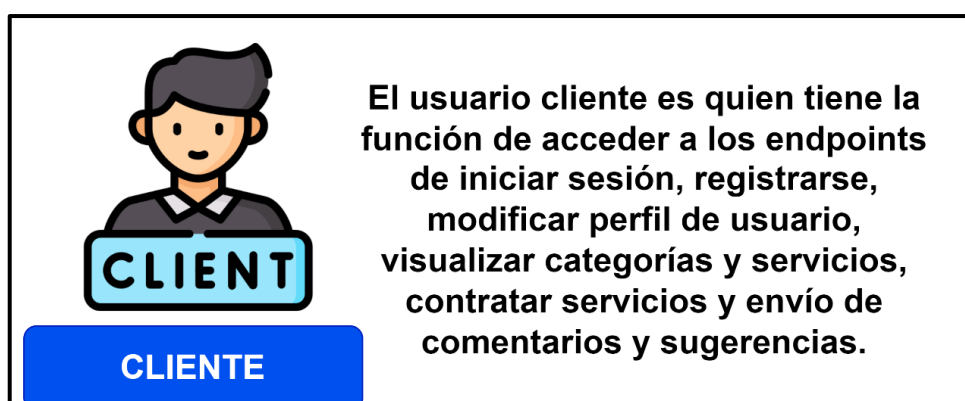


Figura 3.3 Usuario Cliente.

Organización y diseño de tablas en la Base de datos SQL.

Se ha optado por emplear MySQL como plataforma para la ejecución de este módulo, lo cual posibilita el resguardo de los datos relativos a los usuarios, servicios, pedidos,

comentarios, etc. En la **Figura 3.4**, se exhiben de manera global todas las entidades identificadas, al tiempo que en el **ANEXO II** se aprecia de forma más clara la interrelación entre dichas entidades.

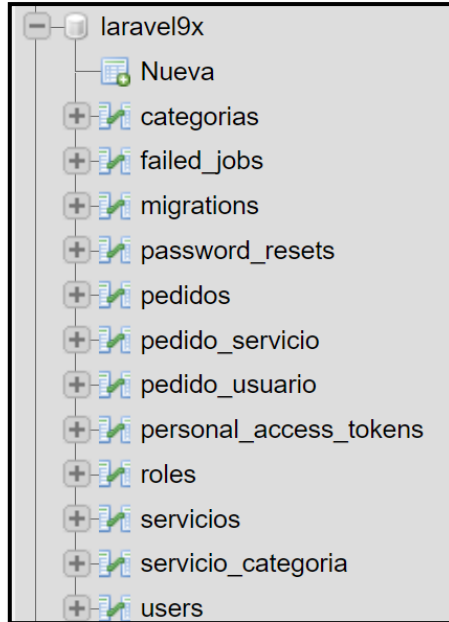


Figura 3.4 Entidades de la Base de datos.

Configuraciones necesarias en los directorios para el desarrollo del *backend*

La herramienta principal que se ha empleado para la codificación de los *endpoints* es *Visual Studio Code*. Además, junto con las librerías y directorios proporcionados por el *Framework* Laravel, esta herramienta ofrece una organización estructurada y de seguimiento sencillo. Un ejemplo de esto es la presentación completa del proyecto en la **Figura 3.5**.

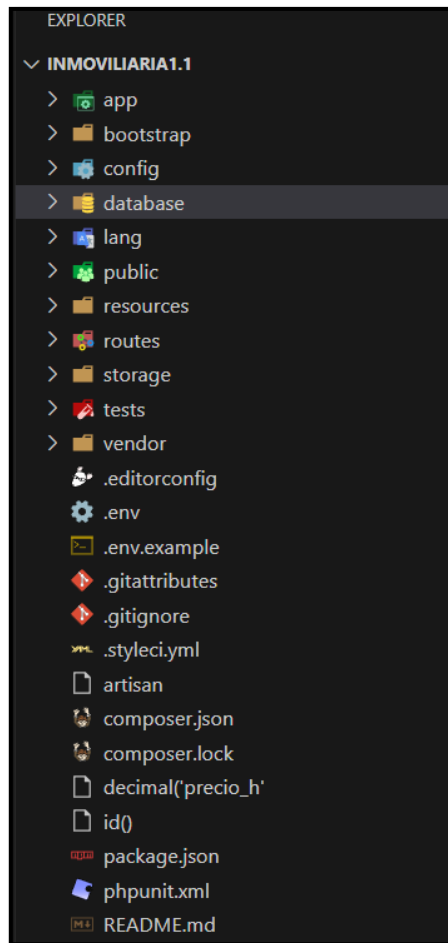


Figura 3.5 Estructura del proyecto.

Definición de los módulos particulares para cada categoría de usuario

Después de iniciar sesión en el *backend*, los usuarios con perfil administrador, empleado y cliente pueden acceder a los diferentes módulos como se presenta en la **Figura 3.6** cada uno según su respectivo rol.

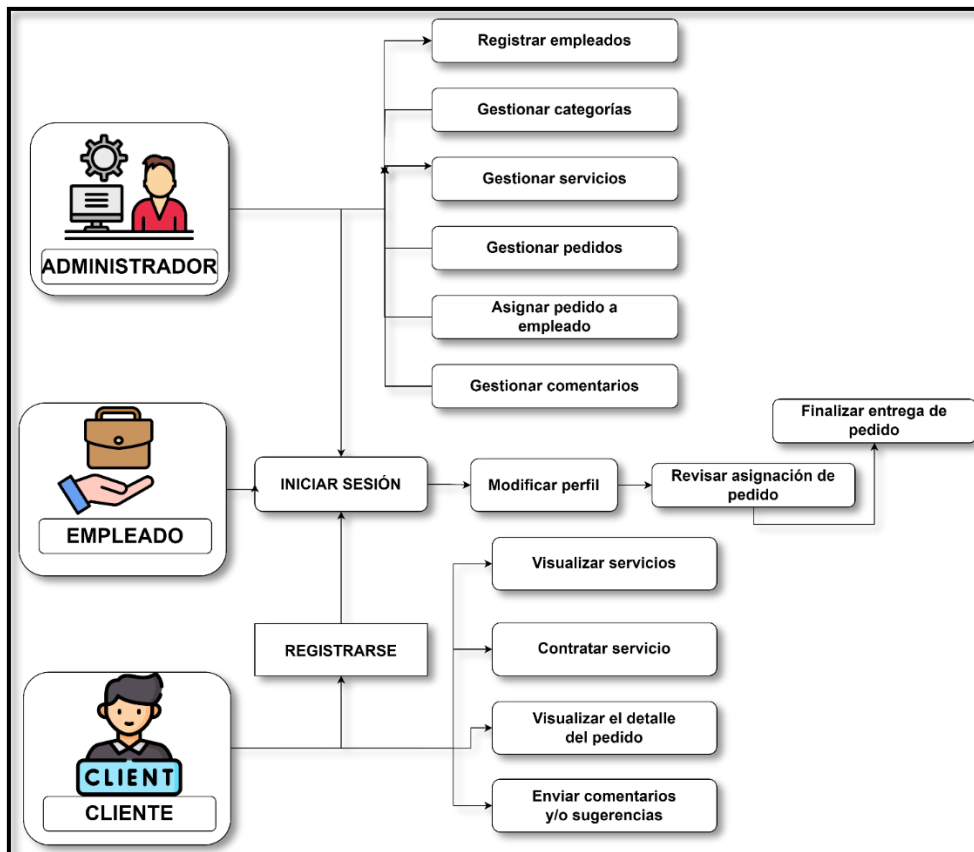


Figura 3.6 Definición de módulos para los usuarios.

3.2 Sprint 1. Endpoints para el perfil administrador.

Este *Sprint* está formado por tareas esenciales para la implementación de *endpoints* correspondientes al rol administrador, los cuales abarcan:

- Inicio de sesión, cierre de sesión y modificación de contraseña.
- Modificación de perfil de usuario.
- Gestión de categorías.
- Gestión de servicios.
- Gestión de pedidos.
- Asignar pedido a empleado.
- Visualización de estado del pedido.
- Gestión de comentarios y/o sugerencias.

Inicio de sesión, cierre de sesión y modificación de contraseña

El *backend* se encarga de gestionar múltiples *endpoints* para el respectivo inicio de sesión, modificación de contraseña y cierre de sesión. Estos *endpoints* están definidos en rutas privadas por lo que requiere iniciar sesión previamente. También se encuentra el *endpoint* de inicio de sesión, el cual es accesible a través de una ruta pública utilizando el método POST. Por otro lado, el cierre de sesión permite a los usuarios salir del *backend* en cualquier momento, según se evidencia en la **Figura 3.7**, junto con su correspondiente prueba unitaria en la **Figura 3.8**. Cabe destacar que todos estos *endpoints* antes mencionados están dirigidos a los roles de administrador, empleado y cliente. Adicional a ello, se encuentra disponible una descripción más detallada de todas estas funcionalidades en el **ANEXO III**.

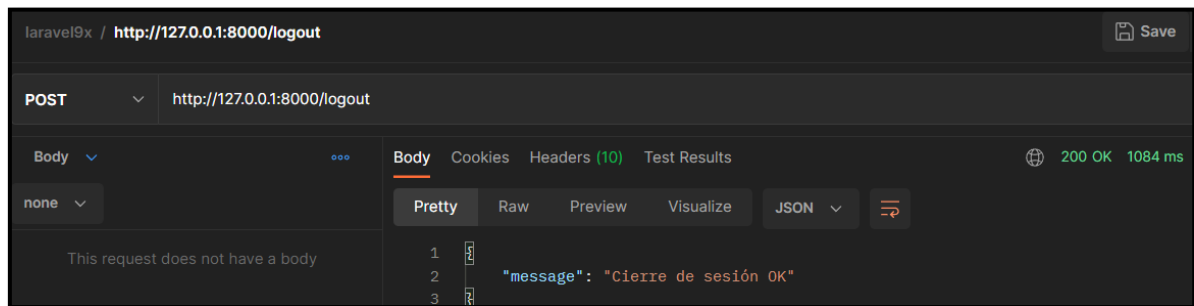


Figura 3.7 Logout.



Figura 3.8 Test de Logout.

Modificación de perfil de usuario

El *backend* ejecuta un *endpoint* asignado a la modificación de datos del perfil de usuario, el cual está dirigido a todos los usuarios. Este *endpoint* está definido en una ruta privada por lo que requiere iniciar sesión previamente. Además, el *endpoint* de modificar perfil para el usuario administrador solo puede acceder al cambio de los campos *email* y nombre, según se evidencia en la **Figura 3.9** junto con su correspondiente prueba unitaria en **Figura 3.10**. Adicional a ello, se encuentra disponible una descripción más detallada de todas estas funcionalidades en el **ANEXO III**.

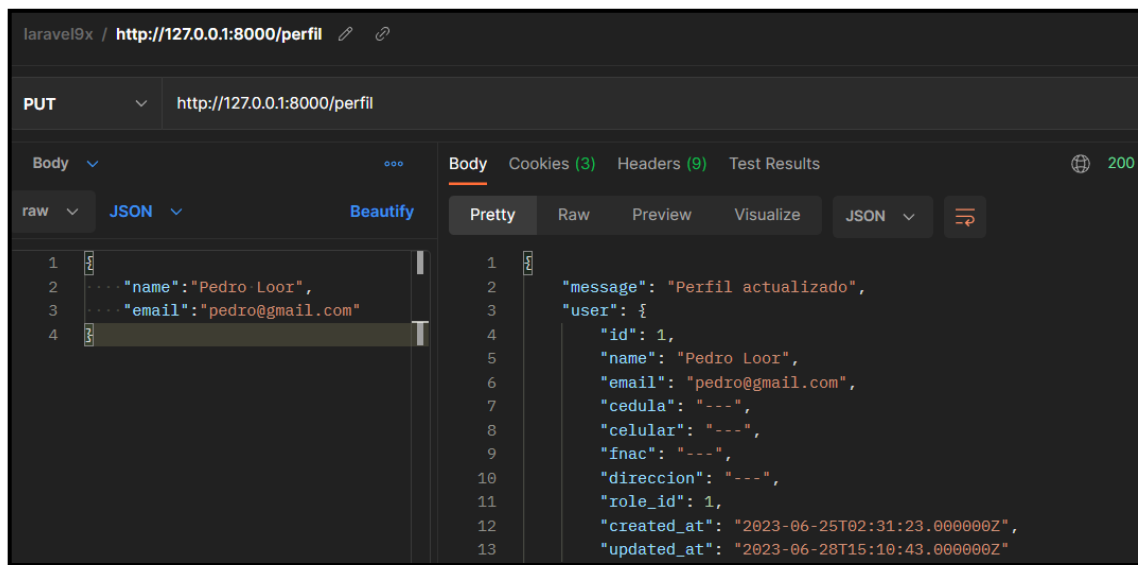


Figura 3.9 Modificación del Perfil.



Figura 3.10 Test de modificación del perfil.

Gestión de categorías

Mediante el componente *backend*, se brinda al usuario administrador la capacidad de acceder al módulo de gestión de categorías en las que se clasifican los servicios a ofertar. Para este propósito, se han habilitado diversos *endpoints* definidos en rutas privadas que permiten al administrador ver, crear, modificar y eliminar categorías. Cabe destacar que, de forma predeterminada, ya se encuentran creadas 3 categorías. Además, para eliminar una categoría es un requisito que no se encuentren servicios relacionado a dicha categoría, como se muestra en la **Figura 3.11** junto con su correspondiente prueba unitaria en la **Figura 3.12**. Adicional a ello, se encuentra disponible una descripción más detallada de todas estas funcionalidades en el **ANEXO III**.

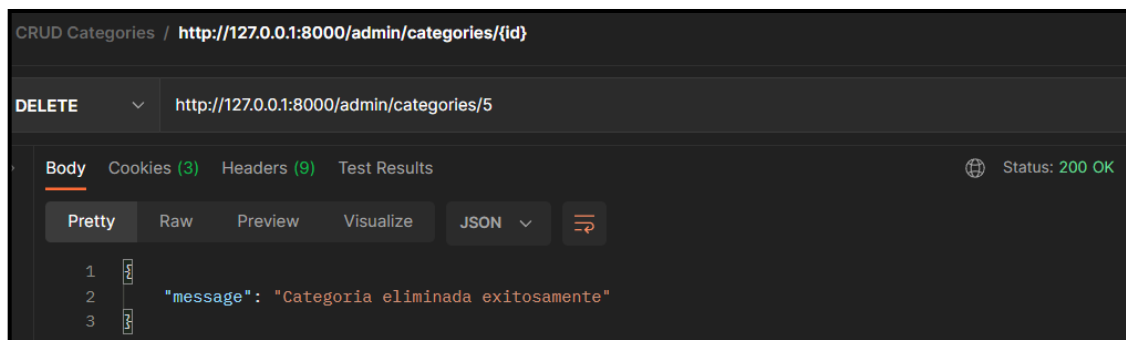


Figura 3.11 Eliminar categoría.



Figura 3.12 Test de eliminar categoría.

Gestión de servicios

Mediante el componente *backend*, se brinda al usuario administrador la capacidad de acceder al módulo de gestión de servicios. Para este propósito, se han habilitado diversos *endpoints* definidos en rutas privadas que permiten al administrador ver, crear, modificar y

eliminar servicios. Cabe destacar que, de forma predeterminada, ya se encuentran creados 6 servicios. Además, para añadir un nuevo servicio es necesario agregar a las categorías que está relacionado dicho servicio, como se muestra en la **Figura 3.13** junto con su correspondiente prueba unitaria en la **Figura 3.14**. Adicional a ello, se encuentra disponible una descripción más detallada de todas estas funcionalidades en el **ANEXO III**.

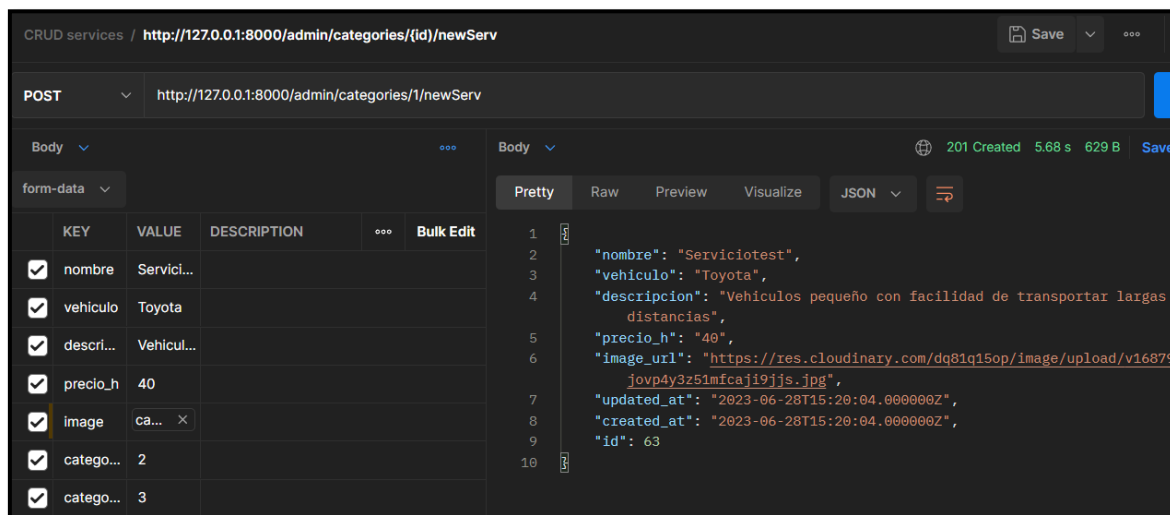


Figura 3.13 Creación de servicio.



Figura 3.14 Test de creación de servicio.

Gestión de pedidos

Mediante el componente *backend*, se brinda al usuario administrador la capacidad de acceder al módulo de gestión de pedidos. Para este propósito, se han habilitado diversos *endpoints* definidos en rutas privadas que permiten al administrador ver, aceptar y rechazar los pedidos que se han generado por el cliente. En este contexto cabe destacar que, el estado del pedido puede tener 4 tipos: pendiente, aceptado, rechazado y completado.

Además, en caso de rechazar un pedido el administrador puede añadir una observación que puede ver el cliente, como se muestra en la **Figura 3.15** junto con su correspondiente prueba unitaria en la **Figura 3.16**. Adicional a ello, se encuentra disponible una descripción más detallada de todas estas funcionalidades en el **ANEXO III**.

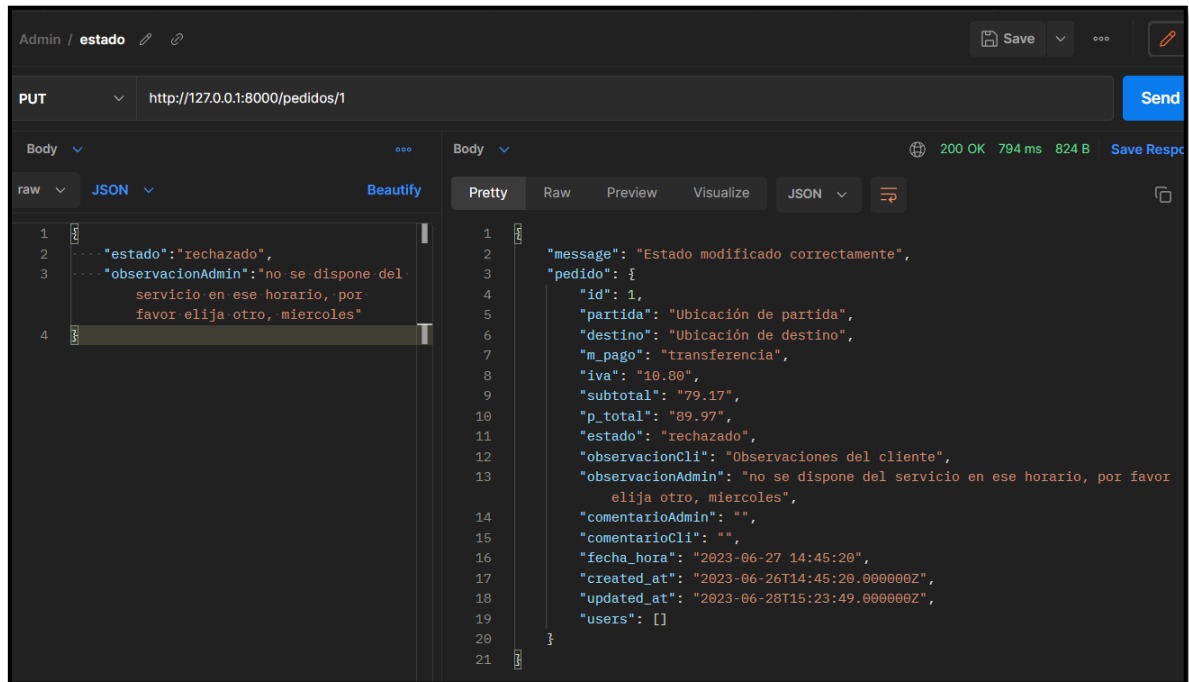


Figura 3.15 Aprobar y rechazar pedidos.

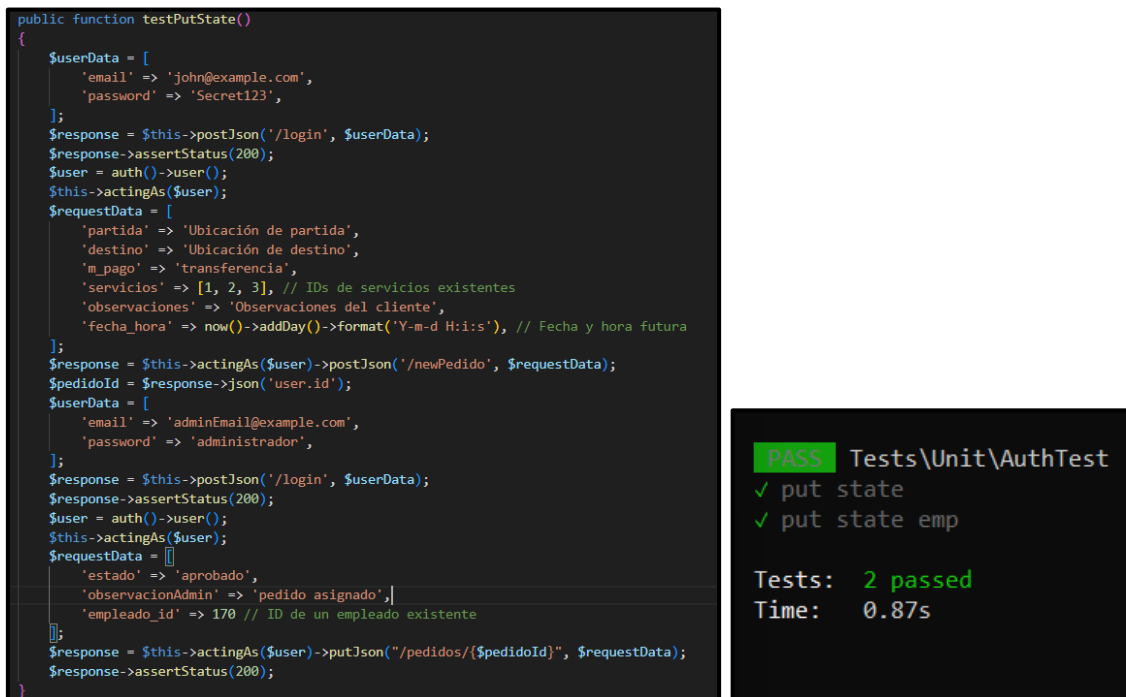
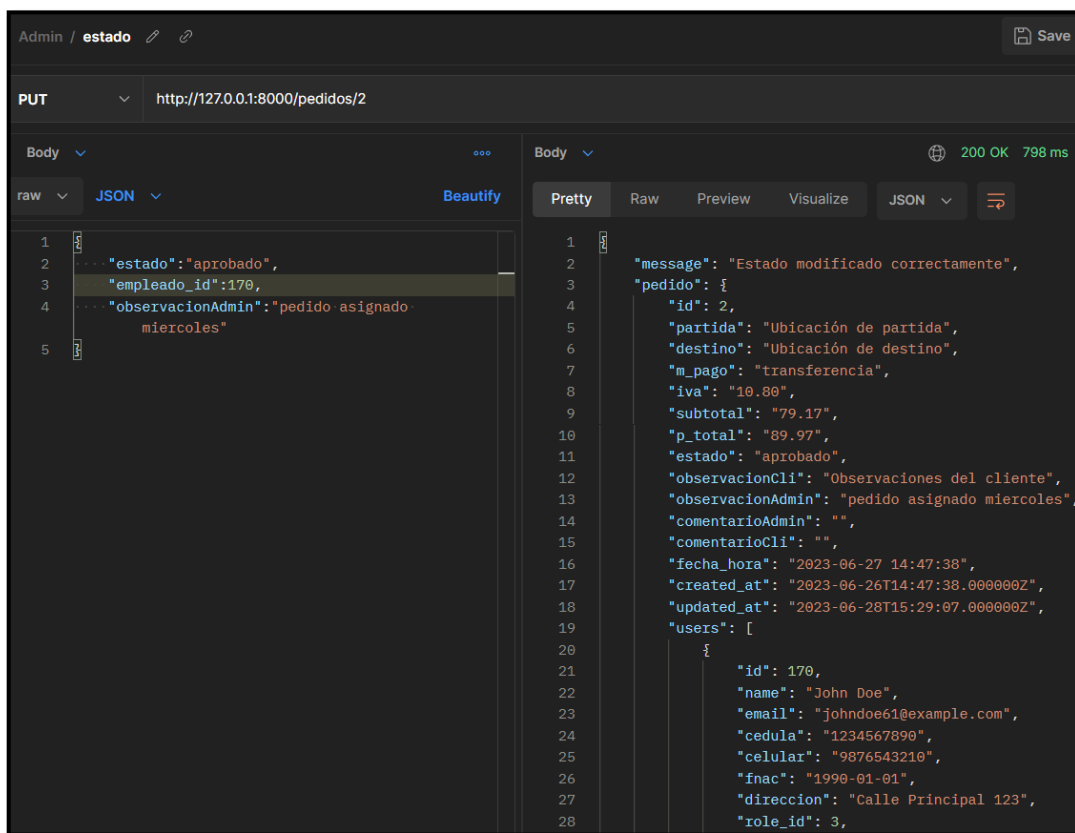


Figura 3.16 Test de aprobar y rechazar pedidos.

Asignar pedido a empleado

Mediante el componente *backend*, se brinda al usuario administrador la capacidad de asignar al empleado uno o varios pedidos. Para este propósito, se han habilitado diversos *endpoints* definidos en rutas privadas que permiten al administrador crear una relación entre el pedido y el empleado. En este contexto, cabe destacar que, para asignar un empleado es necesario primero aceptar un pedido, como se muestra en la **Figura 3.17** junto con su correspondiente prueba unitaria en la **Figura 3.18**. Adicional a ello, se encuentra disponible una descripción más detallada de todas estas funcionalidades en el **ANEXO III**.



The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** http://127.0.0.1:8000/pedidos/2
- Request Body (JSON):**

```
1 {
2   "estado": "aprobado",
3   "empleado_id": 170,
4   "observacionAdmin": "pedido asignado
5     miercoles"
```
- Response (JSON):**

```
1 {
2   "message": "Estado modificado correctamente",
3   "pedido": {
4     "id": 2,
5     "partida": "Ubicación de partida",
6     "destino": "Ubicación de destino",
7     "m_pago": "transferencia",
8     "iva": "10.80",
9     "subtotal": "79.17",
10    "p_total": "89.97",
11    "estado": "aprobado",
12    "observacionCli": "Observaciones del cliente",
13    "observacionAdmin": "pedido asignado miercoles",
14    "comentarioAdmin": "",
15    "comentarioCli": "",
16    "fecha_hora": "2023-06-27 14:47:38",
17    "created_at": "2023-06-26T14:47:38.000000Z",
18    "updated_at": "2023-06-28T15:29:07.000000Z",
19    "users": [
20      {
21        "id": 170,
22        "name": "John Doe",
23        "email": "johndoe61@example.com",
24        "cedula": "1234567890",
25        "celular": "9876543210",
26        "fnac": "1990-01-01",
27        "direccion": "Calle Principal 123",
28        "role_id": 3,
```

Figura 3.17 Asignación de pedido a empleado.

```

public function testPutState()
{
    $userData = [
        'email' => 'john@example.com',
        'password' => 'Secret123',
    ];
    $response = $this->postJson('/login', $userData);
    $response->assertStatus(200);
    $user = auth()->user();
    $this->actingAs($user);
    $requestData = [
        'partida' => 'Ubicación de partida',
        'destino' => 'Ubicación de destino',
        'm_pago' => 'transferencia',
        'servicios' => [1, 2, 3], // IDs de servicios existentes
        'observaciones' => 'Observaciones del cliente',
        'fecha_hora' => now()->addDay()->format('Y-m-d H:i:s'), // Fecha y hora futura
    ];
    $response = $this->actingAs($user)->postJson('/newPedido', $requestData);
    $pedidoId = $response->json('user.id');
    $userData = [
        'email' => 'adminEmail@example.com',
        'password' => 'administrador',
    ];
    $response = $this->postJson('/login', $userData);
    $response->assertStatus(200);
    $user = auth()->user();
    $this->actingAs($user);
    $requestData = [
        'estado' => 'rechazado',
        'observacionAdmin' => 'No se dispone',
        'empleado_id' => 170 // ID de un empleado existente
    ];
    $response = $this->actingAs($user)->putJson("/pedidos/{$pedidoId}", $requestData);
    $response->assertStatus(200);
}

```

```

PASS Tests\Unit\AuthTest
✓ put state
✓ put state emp

Tests: 2 passed
Time: 0.87s

```

Figura 3.18 Test de asignación de pedido a empleado.

Visualización de estado del pedido

Mediante el componente *backend*, se brinda a todos los usuarios la capacidad de ver el estado y el detalle del pedido. Para este propósito, se han habilitado diversos *endpoints* definidos en rutas privadas que permiten al administrador, empleado y cliente visualizar el estado del pedido, así como el detalle. En este contexto, cabe destacar que, los usuarios empleado y cliente solo pueden ver los pedidos a los que están relacionados, como se muestra en la **Figura 3.19** junto con su correspondiente prueba unitaria en la **Figura 3.20**. Adicional a ello, se encuentra disponible una descripción más detallada de todas estas funcionalidades en el **ANEXO III**.

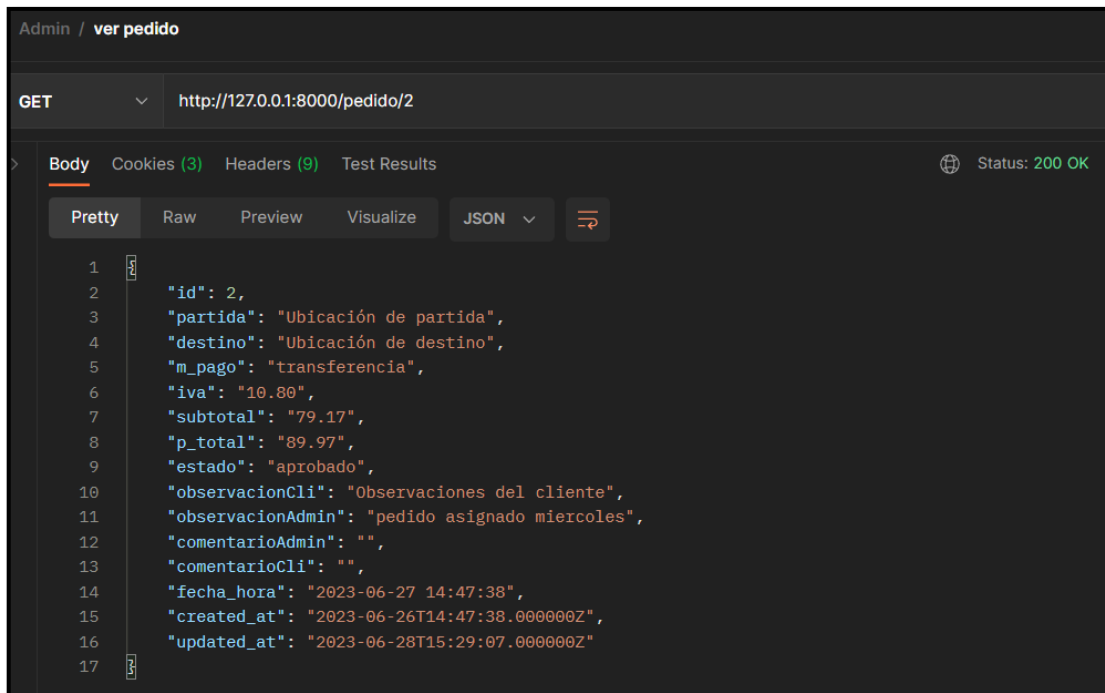


Figura 3.19 Visualización de estado de pedido.



Figura 3.20 Test de visualización de pedido.

Gestión de comentarios y/o sugerencias

Mediante el componente *backend*, se brinda al usuario administrador la capacidad de gestionar los comentarios y/o sugerencias que se han generado por el cliente. Para este propósito, se han habilitado diversos *endpoints* definidos en rutas privadas que permiten al administrador ver todos los comentarios y responder a ellos si el administrador cree necesario, como se muestra en la **Figura 3.21** junto con su correspondiente prueba unitaria en la **Figura 3.22**. Adicional a ello, se encuentra disponible una descripción más detallada de todas estas funcionalidades en el **ANEXO III**.

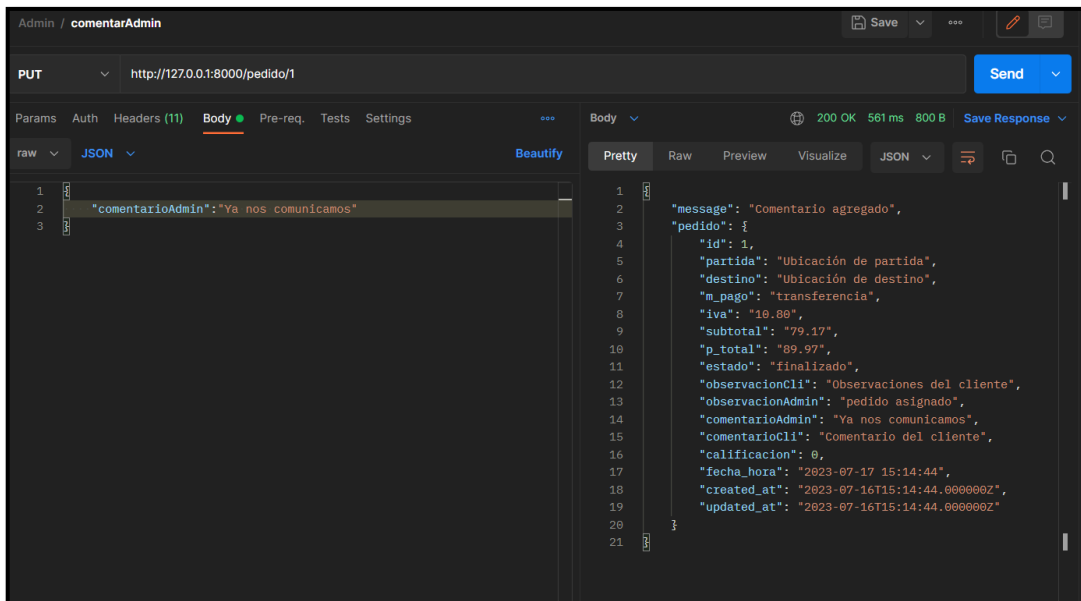


Figura 3.21 Comentar un pedido.



Figura 3.22 Test de comentar un pedido.

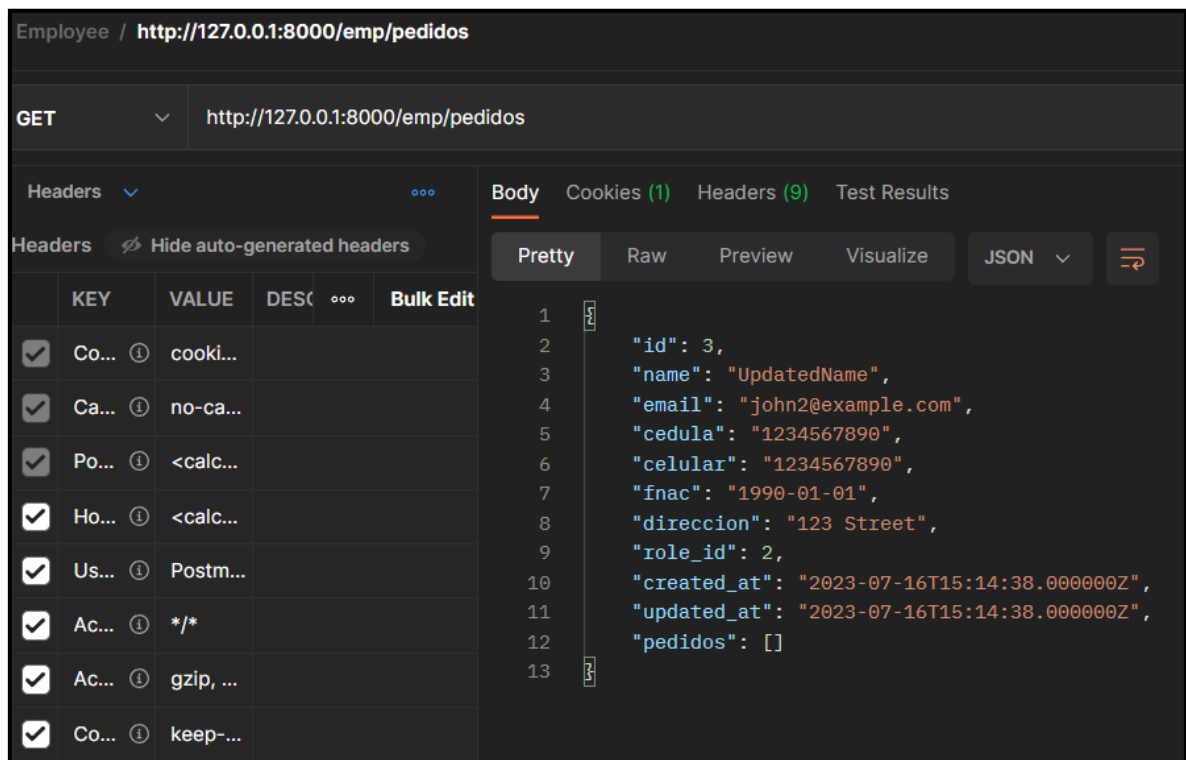
3.3 Sprint 2. Endpoints para el perfil empleado.

Este *Sprint* est\u00e1 formado por tareas esenciales para la implementaci\u00f3n de *endpoints* correspondientes al rol empleado, los cuales abarcan:

- Revisar asignaci\u00f3n del pedido.
- Finalizar entrega del pedido.

Revisar asignación del pedido

Mediante el componente *backend*, se brinda al usuario empleado la capacidad de visualizar los pedidos a los que fue asignado previamente por el administrador. Para este propósito, se han habilitado diversos *endpoints* definidos en rutas privadas que permiten al empleado ver lo siguiente: el detalle del pedido, datos del cliente y los servicios relacionados a dicho pedido, como se muestra en la **Figura 3.23** junto con su correspondiente prueba unitaria en la **Figura 3.24**. Adicional a ello, se encuentra disponible una descripción más detallada de todas estas funcionalidades en el **ANEXO III**.



The screenshot shows a web browser's developer tools interface. At the top, it indicates the user is an 'Employee' and the URL is 'http://127.0.0.1:8000/emp/pedidos'. The request method is 'GET'. The response body is displayed in a 'Pretty' JSON format, showing the following data:

```
1  {
2    "id": 3,
3    "name": "UpdatedName",
4    "email": "john2@example.com",
5    "cedula": "1234567890",
6    "celular": "1234567890",
7    "fnac": "1990-01-01",
8    "direccion": "123 Street",
9    "role_id": 2,
10   "created_at": "2023-07-16T15:14:38.000000Z",
11   "updated_at": "2023-07-16T15:14:38.000000Z",
12   "pedidos": []
13 }
```

Figura 3.23 Visualizar pedido que ha sido asignado.


```

public function testPedidosEmp()
{
    // Creamos un usuario con el rol de empleado
    $userData = [
        'email' => 'johndoe6@example.com',
        'password' => 'password123',
    ];

    $response = $this->postJson('/login', $userData);

    // Verificamos que el inicio de sesión sea exitoso
    $response->assertStatus(200);

    // Obtenemos el usuario autenticado
    $user = auth()->user();

    // Simulamos la autenticación del usuario
    $this->actingAs($user);

    $response = $this->actingAs($user)->get("/emp/pedidos");
    $response->assertStatus(200);
}

```

PASS Tests\Unit\AuthTest
 ✓ pedidos emp

Tests: 1 passed
 Time: 0.52s

Figura 3.24 Test de visualizar pedido que ha sido asignado.

Finalizar entrega del pedido

Mediante el componente *backend*, se brinda al usuario empleado la capacidad de finalizar la entrega de un pedido asignado una vez que se termine de ofertar todos los servicios asociados al pedido. Para este propósito, se han habilitado diversos *endpoints* definidos en rutas privadas que permiten al empleado cambiar el estado del pedido a “completado”, como se muestra en la **Figura 3.25** junto con su correspondiente prueba unitaria en la **Figura 3.26**. Adicional a ello, se encuentra disponible una descripción más detallada de todas estas funcionalidades en el **ANEXO III**.

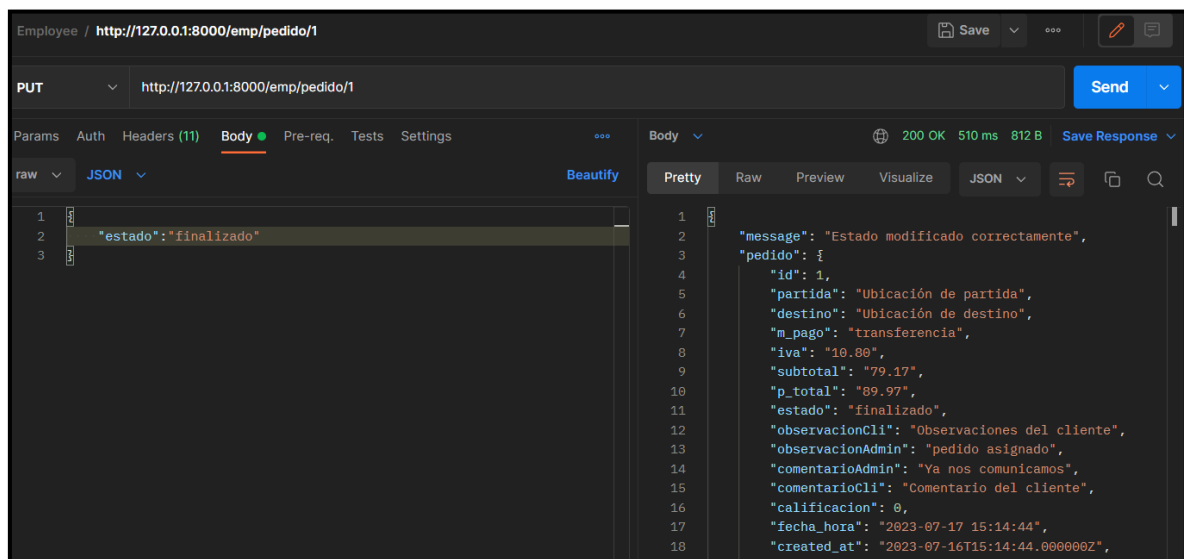


Figura 3.25 Finalizar pedido.

```
public function testPutStateEmp()
{
    // Creamos un usuario con el rol de cliente
    $userData = [
        'email' => 'johndoe@example.com',
        'password' => 'password123',
    ];

    $response = $this->postJson('/login', $userData);

    // Verificamos que el inicio de sesión sea exitoso
    $response->assertStatus(200);

    // Obtenemos el usuario autenticado
    $user = auth()->user();

    // Simulamos la autenticación del usuario
    $this->actingAs($user);

    $requestData = [
        'estado' => 'finalizado',
    ];
    $response = $this->actingAs($user)->putJson("/emp/pedido/1", $requestData);
    $response->assertStatus(200);
}
```

```
PASS Tests\Unit\AuthTest
✓ put state
✓ put state emp

Tests: 2 passed
Time: 0.87s
```

Figura 3.26 Test de finalizar pedido.

3.4 Sprint 3. Endpoints para el perfil cliente.

Este *Sprint* está formado por tareas esenciales para la implementación de los *endpoints* correspondientes al rol cliente, los cuales abarcan:

- Visualizar servicios.
- Contratar servicios.
- Envío de comentarios y/o sugerencias.

Visualizar servicios

Mediante el componente *backend*, se brinda al usuario cliente la capacidad de visualizar los servicios que el usuario con perfil administrador ha creado previamente. Para este propósito, se han habilitado diversos *endpoints* definidos en rutas privadas que permiten al cliente ver las categorías, así como los servicios asociados y los detalles correspondientes, como se muestra en la **Figura 3.27** junto con su correspondiente prueba unitaria en la **Figura 3.28**. Adicional a ello, se encuentra disponible una descripción más detallada de todas estas funcionalidades en el **ANEXO III**.

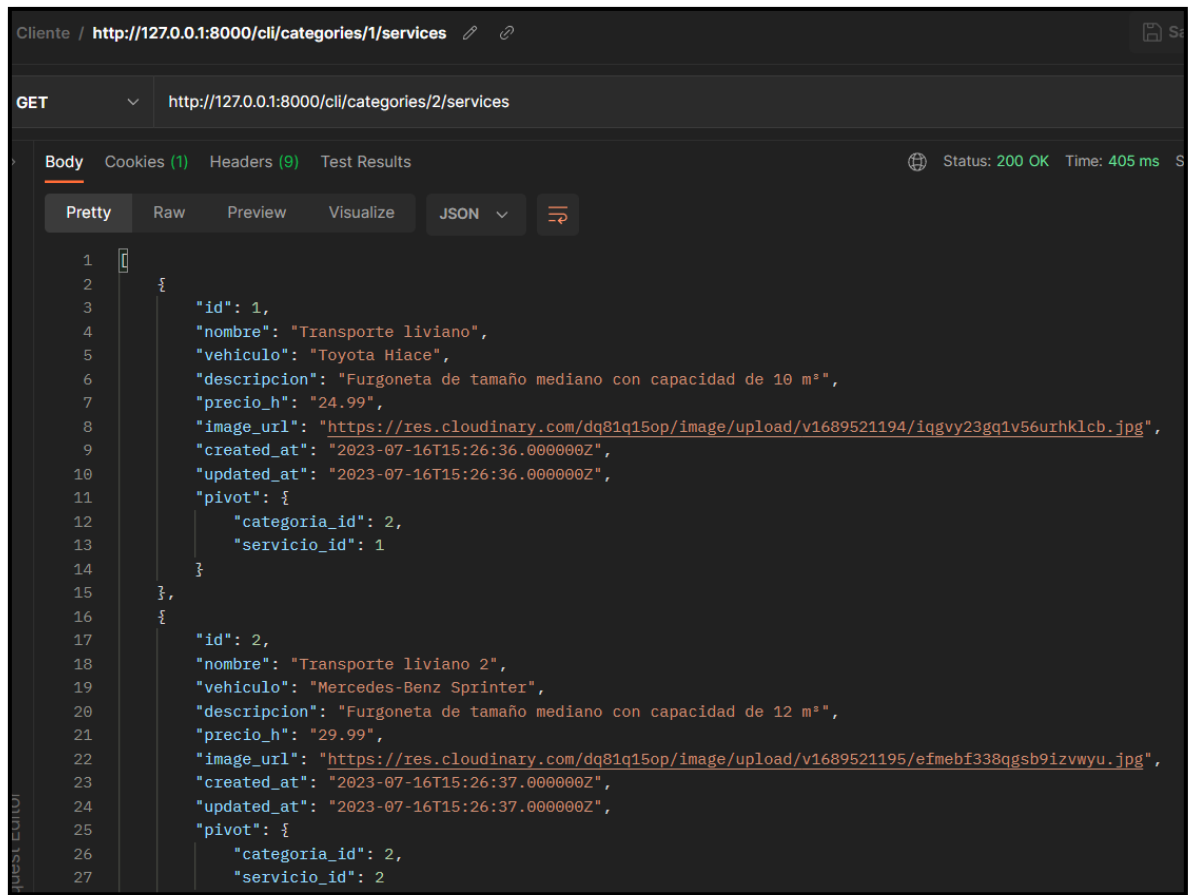


Figura 3.27 Mostrar diversos servicios.

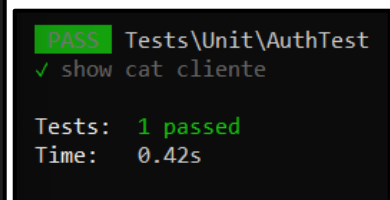
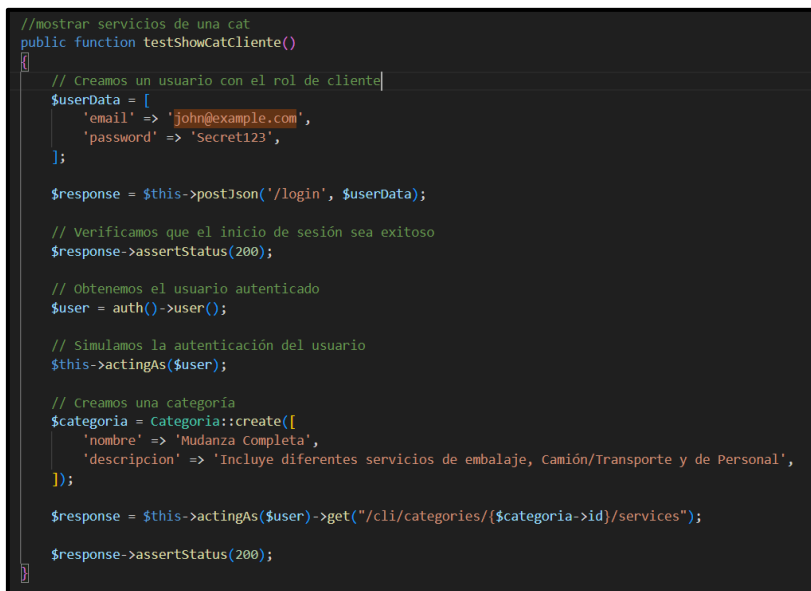
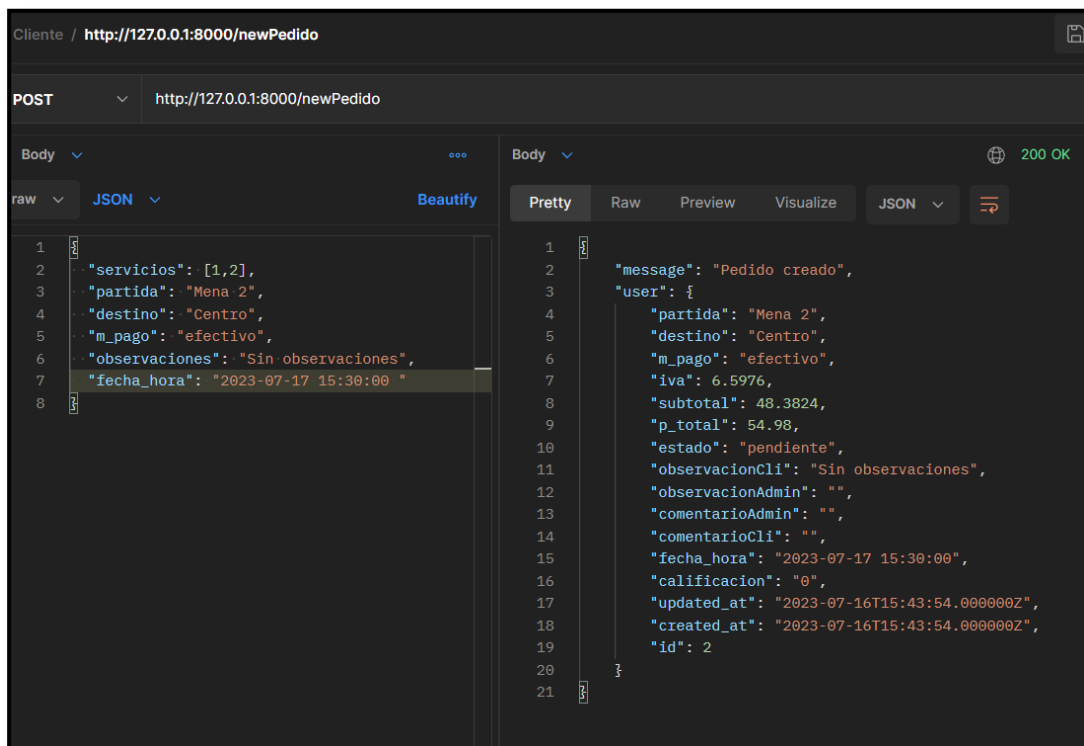


Figura 3.28 Test para mostrar diversos servicios.

Contratar servicios

Mediante el componente *backend*, se brinda al usuario cliente la capacidad de contratar diversos servicios. Para este propósito, se han habilitado diversos *endpoints* definidos en rutas privadas que permiten al cliente crear un nuevo pedido y que al mismo se le puede ir añadiendo uno o varios servicios, así como visualizar el detalle de cada pedido, como se muestra en la **Figura 3.29** junto con su correspondiente prueba unitaria en la **Figura 3.30**. Adicional a ello, se encuentra disponible una descripción más detallada de todas estas funcionalidades en el **ANEXO III**.



The screenshot shows a REST client interface with the following details:

- Client: `http://127.0.0.1:8000/newPedido`
- Method: `POST`
- URL: `http://127.0.0.1:8000/newPedido`
- Body (Request):

```
1  [1,2],
2  "servicios": [1,2],
3  "partida": "Mena 2",
4  "destino": "Centro",
5  "m_pago": "efectivo",
6  "observaciones": "Sin observaciones",
7  "fecha_hora": "2023-07-17 15:30:00 "
```
- Body (Response):

```
1  "message": "Pedido creado",
2  "user": {
3    "partida": "Mena 2",
4    "destino": "Centro",
5    "m_pago": "efectivo",
6    "iva": 6.5976,
7    "subtotal": 48.3824,
8    "p_total": 54.98,
9    "estado": "pendiente",
10   "observacionCli": "Sin observaciones",
11   "observacionAdmin": "",
12   "comentarioAdmin": "",
13   "comentarioCli": "",
14   "fecha_hora": "2023-07-17 15:30:00",
15   "calificacion": "0",
16   "updated_at": "2023-07-16T15:43:54.000000Z",
17   "created_at": "2023-07-16T15:43:54.000000Z",
18   "id": 2
19 }
20
21
```
- Status: `200 OK`

Figura 3.29 Crear un nuevo pedido.

```
public function testNewPedido()
{
    // Creamos un usuario con el rol de cliente
    $userData = [
        'email' => 'john@example.com',
        'password' => 'Secret123',
    ];

    $response = $this->postJson('/login', $userData);

    // Verificamos que el inicio de sesión sea exitoso
    $response->assertStatus(200);

    // Obtenemos el usuario autenticado
    $user = auth()->user();

    // Simulamos la autenticación del usuario
    $this->actingAs($user);

    // Datos para la solicitud POST
    $requestData = [
        'partida' => 'Ubicación de partida',
        'destino' => 'Ubicación de destino',
        'm_pago' => 'transferencia',
        'servicios' => [1, 2, 3], // IDs de servicios existentes
        'observaciones' => 'Observaciones del cliente',
        'fecha_hora' => now()->addDay()->format('Y-m-d H:i:s'), // Fecha y hora futura
    ];

    // Enviamos la solicitud POST a la ruta /newPedido
    $response = $this->actingAs($user)->postJson('/newPedido', $requestData);
    $response->assertStatus(200);
}

PASS Tests\Unit\AuthTest
✓ new pedido

Tests: 1 passed
Time: 0.53s
```

Figura 3.30 Test de crear un nuevo pedido.

Envío de comentarios y/o sugerencias

Mediante el componente *backend*, se brinda al usuario cliente la capacidad de enviar comentarios y/o sugerencias. Para este propósito, se han habilitado diversos *endpoints* definidos en rutas privadas que permiten al cliente dejar un comentario y/o sugerencia a un pedido que se ha contratado y el cual se haya completado previamente, como se muestra en la **Figura 3.31** junto con su correspondiente prueba unitaria en la **Figura 3.32**. Adicional a ello, se encuentra disponible una descripción más detallada de todas estas funcionalidades en el **ANEXO III**.

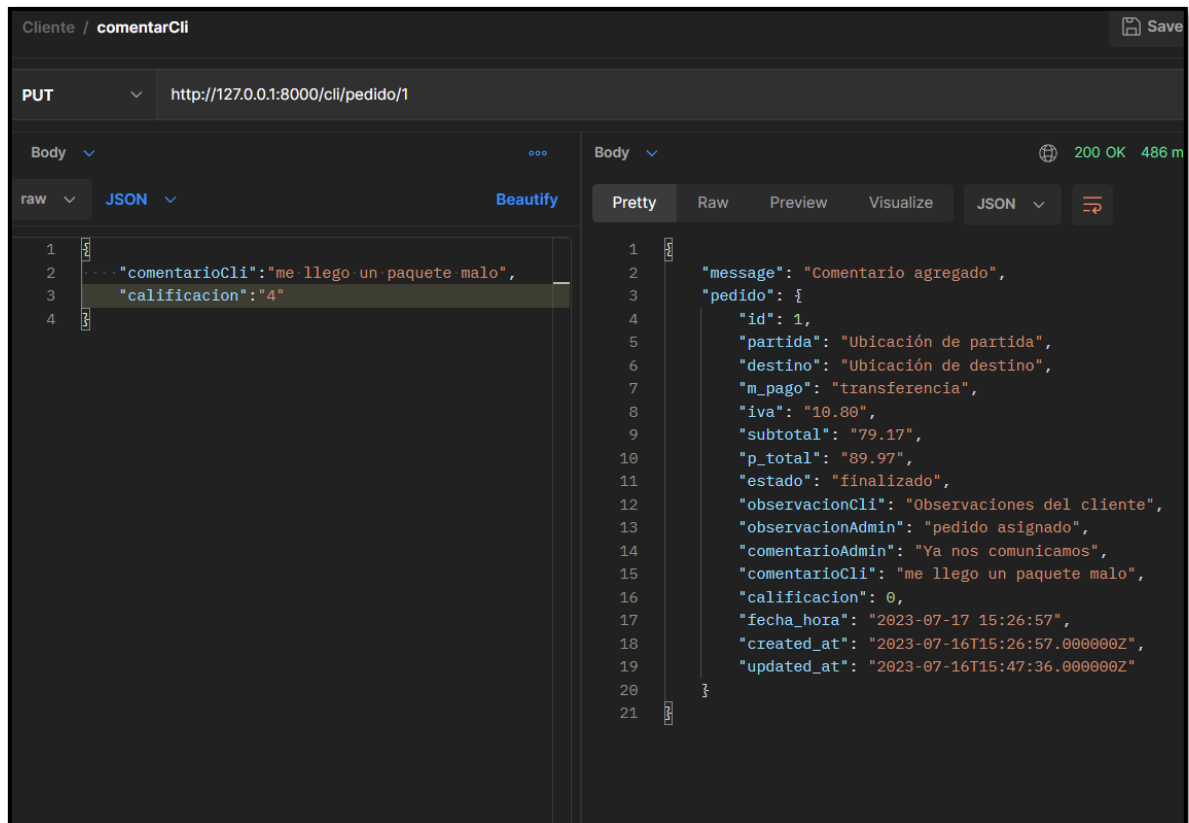


Figura 3.31 Comentar pedido por parte del cliente.

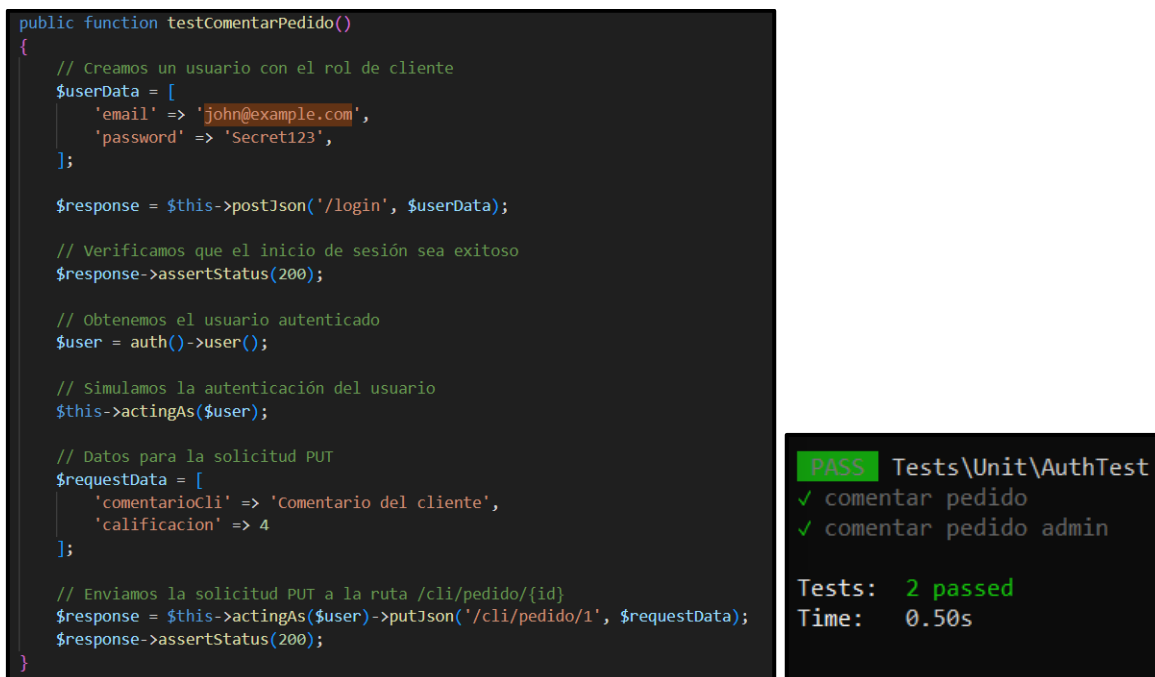


Figura 3.32 Test de comentar un pedido por parte del cliente.

3.5 *Sprint* 4. Ejecución de pruebas para el componente *backend*.

Durante este *Sprint*, se han llevado a cabo tareas fundamentales para realizar pruebas después de completar la fase de codificación de los *endpoints*. Estas tareas incluyen la elaboración de pruebas unitarias, pruebas de compatibilidad, pruebas de carga, pruebas de aceptación en el *backend*.

- Resultados de pruebas unitarias del *backend*.
- Resultados de pruebas de compatibilidad del *backend*.
- Resultados de pruebas de carga del *backend*.
- Resultados de pruebas de aceptación del *backend*.
- Despliegue del *backend*.

Resultados de pruebas unitarias del *backend*

Estas pruebas se basan en descomponer el *software* en componentes más pequeños y realizar pruebas individuales en cada uno de ellos. El objetivo es verificar el correcto funcionamiento de cada componente por separado, identificar posibles errores y garantizar la calidad y la integridad del sistema en su conjunto. Mediante las pruebas unitarias, se busca asegurar que cada unidad de código cumpla con sus funcionalidades específicas y se integre correctamente con las demás partes del sistema *software*. Esto contribuye a la detección temprana de problemas y a la mejora de la eficiencia y la estabilidad del sistema, lo que resulta en un *software* más confiable y de mayor calidad [42]. Muestra de ello, en la **Figura 3.33** se muestra una función que sirve para la ejecución de la prueba unitaria de inicio de sesión. Por otro lado, en la **Figura 3.34** se presentan el resultado que se ha obtenido tras llevar a cabo la respectiva prueba. Adicional a ello, se encuentra disponible las demás funciones completas, así como el resultado de cada una de ellas en el **ANEXO II**.

```

public function testLogin()
{
    $userData = [
        'email' => 'adminEmail@example.com',
        'password' => 'administrador',
    ];

    $response = $this->postJson('/login', $userData);

    $response->assertStatus(200)
        ->assertJson([
            'message' => 'Inicio de sesión exitoso',
            'user' => [
                'email' => $userData['email'],
            ],
        ]);
}

```

Figura 3.33 Test de Login.

```

PASS Tests\Unit\AuthTest
✓ register
✓ login
✓ perfil
✓ logout
✓ update password
✓ register employee
✓ index cat
✓ store cat
✓ update cat
✓ destroy cat
✓ show cat
✓ store serv
✓ show serv
✓ update serv
✓ destroy serv
✓ show pedidos
✓ index cat cliente
✓ show cat cliente
✓ new pedido
✓ show pedido
✓ put state
✓ pedidos emp
✓ put state emp
✓ comentar pedido
✓ comentar pedido admin

PASS Tests\Unit\ExampleTest
✓ that true is true

PASS Tests\Feature\ExampleTest
✓ the application returns a successful response

Tests: 27 passed
Time: 13.55s

```

Figura 3.34 Resultado de las pruebas unitarias.

Tras la conclusión exitosa de las pruebas unitarias, se han logrado obtener resultados altamente satisfactorios para las 27 APIs restantes que han sido implementadas en los 16 *endpoints* principales. Estos logros han sido alcanzados con un tiempo de respuesta de 7.57 segundos.

Resultados de pruebas de compatibilidad del *backend*

Las pruebas de compatibilidad consisten en evaluar el rendimiento del *software* en diversos entornos, que pueden incluir diferentes navegadores, dispositivos, sistemas operativos y otros factores relevantes. El propósito principal de estas pruebas es identificar y minimizar posibles errores de ejecución que puedan surgir en los diferentes entornos en los que se ejecuta el sistema *software*. Al realizar pruebas exhaustivas de compatibilidad, se busca asegurar que el *backend* pueda interactuar de manera adecuada y eficiente con diversos entornos, brindando una experiencia consistente y sin problemas para los usuarios finales [42]. Muestra de ello, en la **Tabla 3.1** se lista los clientes HTTP que se han empleado para llevar a cabo la evaluación de la compatibilidad de las APIs generadas por el *backend*. Adicional a ello, se encuentra disponible todos los resultados completos de esta prueba en el **ANEXO II**.

Tabla 3.1: Clientes HTTP.

NOMBRE	VERSIÓN
<i>Postman</i>	V9.28.3
<i>Thunder Client</i>	V2.0.2

Tras finalizar las pruebas de compatibilidad, se han logrado obtener respuestas similares durante la ejecución de las solicitudes, obteniendo así resultados positivos en la utilización de las API mediante clientes HTTP,

Resultados de pruebas de carga del *backend*

Estas pruebas consisten en evaluar el desempeño del sistema, como una API REST, bajo condiciones simuladas de alta demanda. Estas pruebas generan una carga de trabajo intensiva mediante un gran número de solicitudes simultáneas o una cantidad considerable de datos a procesar. El propósito principal es determinar cómo responde el sistema en situaciones normales y extremas, identificando posibles limitaciones o problemas de escalabilidad. Estas pruebas permiten evaluar la capacidad del *backend* para manejar un alto volumen de tráfico y garantizar un rendimiento óptimo, brindando una experiencia fluida a los usuarios incluso en momentos de carga máxima [43]. Muestra de ello, en la **Figura**

3.35 se muestra una prueba de carga que se ha ejecutado conjuntamente con su resultado utilizando la herramienta *apache jmeter*. Adicional a ello, se encuentra disponible todos los resultados completos de esta prueba en el **ANEXO II**.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
Petición HTTP	101	3814	0	5337	1495.74	40.59%	2.1/sec	6.64	0.52	3315
TOTAL	101	3814	0	5337	1495.74	40.59%	2.1/sec	6.64	0.52	3315

Figura 3.35 Resultado de las pruebas de carga de *login* de usuario.

Tras finalizar las pruebas de carga, se han logrado obtener resultados altamente satisfactorios para las APIs que fueron implementadas en los 16 *endpoints* principales. Se logró un rendimiento de carga de 2,1 segundos al ejecutar un rango de entre 50 a 100 solicitudes.

Resultado de pruebas de aceptación del *backend*

Estas pruebas buscan garantizar que la aplicación cumpla con los requisitos y expectativas del usuario final. Estas pruebas van más allá de verificar la funcionalidad, abordando la experiencia global del usuario y la capacidad del sistema para operar en diversas situaciones. Para ilustrar este enfoque, se presenta una prueba de aceptación en la **Tabla 3.2**. Además, todos los resultados se encuentran disponibles en el **ANEXO II**.

Tabla 3.2 Prueba de aceptación – Gestión de categorías.

PRUEBA DE ACEPTACIÓN	
Identificador (ID): PA004	Identificador de historia de Usuario: HU004
Nombre: Gestionar categorías	
Descripción: El <i>backend</i> permite generar varios <i>endpoints</i> para que el perfil administrador pueda:	
<ul style="list-style-type: none"> Gestionar las categorías, en lo que respecta a mudanzas básicas, mudanzas completas y mudanzas Interprovinciales 	

Pasos de ejecución:

Para gestionar las categorías:

- 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 *backend* admite crear, actualizar, eliminar y listar categorías.

Evaluación de la prueba:

Resultado y conformidad del cliente al 100%.

Tras la realización de la prueba previamente mencionada, se ha verificado la conformidad de todos los componentes con los requisitos que se han establecido al comienzo del proyecto. Asimismo, cada uno de estos elementos ha recibido la aprobación por parte del *Product Owner*.

Despliegue del *backend* a producción

En esta sección se aborda la implementación del componente de *backend* en un entorno de producción a través de DigitalOcean y se obtiene un subdominio a través de duckdns, el cual permite llevar toda la lógica directamente a producción, haciendo accesible desde cualquier ubicación, como se muestra en la **Figura 3.36**. Por último, los detalles del proceso de despliegue se encuentran en el **ANEXO IV**.

<http://mudanzapp.duckdns.org/>

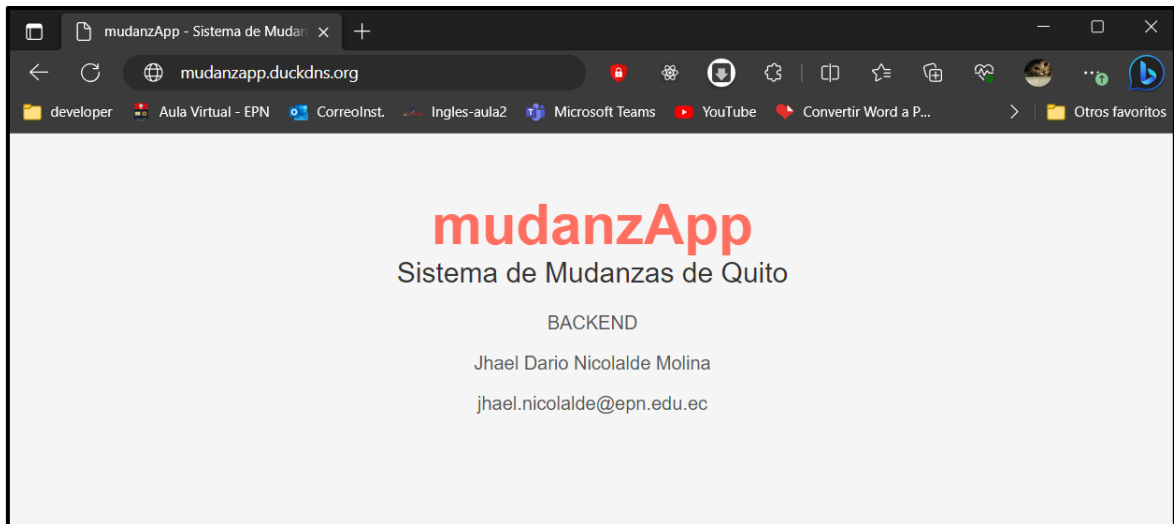


Figura 3.36 Despliegue del *backend* a producción.

4 CONCLUSIONES

Una vez que se ha finalizado el proceso de desarrollo, pruebas y despliegue del *backend* para la gestión de mudanzas, es crucial resaltar las conclusiones que se han obtenido a lo largo de este proyecto. Estas conclusiones representan los hallazgos y resultados que se han alcanzado, así como las lecciones que se han aprendido y los aspectos destacados que han surgido durante todo el proceso. A continuación, se presentan las principales conclusiones derivadas de este desarrollo, que proporcionan una visión integral de los beneficios y logros que se han obtenido con la implementación de esta solución tecnológica para la gestión de mudanzas en la ciudad de Quito.

- El *backend* se basa en satisfacer las demandas esenciales para el funcionamiento óptimo de la aplicación de mudanzas en Quito. Además, este componente facilita la gestión de servicios de mudanzas, permitiendo así a los residentes de Quito explorar y contratar servicios de mudanza de forma segura y confiable.
- *Scrum* ha permitido una mayor flexibilidad y adaptabilidad durante el proceso de desarrollo del *backend* para la gestión de mudanzas, así mismo, ha facilitado la colaboración entre el equipo de desarrollo y los *stakeholders*; y la entrega de incrementos funcionales de manera iterativa y continua.
- La implementación de Scrum y sus artefactos ha influido positivamente en el desarrollo del *backend*, al proporcionar un marco de trabajo que favorece la adaptabilidad, la colaboración efectiva y la entrega progresiva de características esenciales para la presente aplicación de mudanzas en la ciudad de Quito.
- La arquitectura Modelo-Vista-Controlador, el *Framework* Laravel y el desarrollo de APIs han sido factores clave para el desarrollo del *backend* ya que las mismas han permitido la creación de una solución tecnológica confiable, segura y escalable cumpliendo con las necesidades de los usuarios y proporcionando una experiencia óptima durante el proceso de gestión de mudanza.
- La base de datos juega un papel crucial en el funcionamiento central del proyecto, ya que una Base de datos relacional hospedada en MySQL y gestionada mediante Eloquent, ha simplificado tanto el almacenamiento como la recuperación de datos en el proyecto.
- Mediante la realización de diversas pruebas en los múltiples *endpoints*, se ha conseguido alcanzar resultados positivos en todas las características del *backend*,

comprobando su compatibilidad con las diversas consultas asegurando la satisfacción de los usuarios finales.

- En última instancia, la implementación del *backend* en el entorno de producción permite un acceso satisfactorio a la información, la cual puede ser accedida por cualquier aplicación móvil o de lado del cliente.

5 RECOMENDACIONES

A continuación, se presentan las siguientes recomendaciones:

- Es recomendable llevar a cabo un monitoreo regular del rendimiento de la Base de datos MySQL y realizar ajustes según sea necesario. Esto incluye optimizar consultas, índices y configuraciones para garantizar un rendimiento óptimo del *backend*.
- Documentar adecuadamente las APIs son un componente clave en la comunicación entre diferentes sistemas y aplicaciones. Por esta razón, se recomienda documentar de manera exhaustiva y clara todas las APIs que se han desarrollado, incluyendo detalles sobre los puntos de entrada, parámetros aceptados, respuestas esperadas y ejemplos de uso. Lo que facilita la integración con terceros y futuros desarrollos.
- Se recomienda mantener la modularidad y estructura clara del patrón MVC, para así determinar de mejor manera la separación de responsabilidades de cada componente para garantizar la escalabilidad y extensibilidad del *backend* en el futuro.

6 REFERENCIAS BIBLIOGRÁFICAS

- [1] "Diagnóstico estratégico del distrito metropolitano de Quito". QuitoHonesto. https://www.quitohonesto.gob.ec/images/biblioteca/RDC-CMLCC-2021/Anexo_2.pdf, 2023
- [2] "Comercio electrónico. Ideas fundamentales". gestiopolis. <https://www.gestiopolis.com/comercio-electronico-ideas-fundamentales/>, 2023
- [3] "Evolución del comercio electrónico: fases y futuro". DispatchTrack | Beetrack. <https://www.beetrack.com/es/blog/evolucion-del-comercio-electronico>
- [4] "¿Qué es el Marketing Digital, cómo se hace y para qué sirve ✓". RD Station. <https://www.rdstation.com/es/marketing-digital/>
- [5] "Agencia de Regulación y Control de las Telecomunicaciones". Agencia de Regulación y Control de las Telecomunicaciones. <https://www.arcotel.gob.ec/>
- [6] Sommerville, I. (2016). Ingeniería de software. Pearson Educación.
- [7] "¿Qué significa lado del cliente y lado del servidor? | Lado del cliente vs. Lado del servidor". Cloudflare. <https://www.cloudflare.com/es-es/learning/serverless/glossary/client-side-vs-server-side/>
- [8] "Todo lo que ofrece el framework Laravel para el desarrollo web". Gunka Studios. <https://gunkastudios.com/desarrollo-con-el-framework-laravel/>
- [9] Taylor Otwell. (2021). Laravel - The PHP Framework For Web Artisans. Recuperado de <https://laravel.com/>
- [10] Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures. Doctoral dissertation, University of California, Irvine.
- [11] Elmasri, R., & Navathe, S. B. (2016). Fundamentals of Database Systems. Pearson Education
- [12] "Aprende a usar eloquent el ORM de laravel". Styde.net. <https://styde.net/aprende-a-usar-eloquent-el-orm-de-laravel/>

- [13] "¿Cómo crear migraciones en Laravel? | LARAVEL". Cursos de Programación Web y Desarrollo de Aplicaciones Móviles Online.
<https://codea.app/blog/migraciones-en-laravel>

- [14] shani singh. "Laravel 8 factories, seeder". DEV Community.
<https://dev.to/shanisinh03/generate-dummy-laravel-data-with-model-factories-seeder-gg4>

- [15] "API documentation tool | postman". Postman API Platform.
<https://www.postman.com/api-documentation-tool/>

- [16] "PHP: ¿qué es, para qué sirve y cuáles son sus características?" Rock Content - ES. <https://rockcontent.com/es/blog/php/>

- [17] "¿Qué es la prueba de software y cómo funciona? | IBM". IBM - Deutschland | IBM. <https://www.ibm.com/es-es/topics/software-testing>

- [18] "La metodología de la investigación". gestiopolis.
<https://www.gestiopolis.com/la-metodologia-de-la-investigacion/>

- [19] "Las metodologías ágiles más utilizadas y sus ventajas dentro de la empresa". Thinking for Innovation. <https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/>

- [20] "Metodología Scrum para el desarrollo de software ágil". Eniun.
<https://www.eniun.com/metodologia-scrum-desarrollo-software-agil/>

- [21] "Product Owner - Scaled Agile Framework". Scaled Agile Framework.
[https://scaledagileframework.com/product-owner/#:~:text=The%20Product%20Owner%20\(PO\)%20is,with%20customer%20and%20stakeholder%20needs.](https://scaledagileframework.com/product-owner/#:~:text=The%20Product%20Owner%20(PO)%20is,with%20customer%20and%20stakeholder%20needs.)

- [22] "Scrum masters: What are they and what do they do? [2023]". Asana.
<https://asana.com/es/resources/scrum-master#:~:text=Un%20Scrum%20Master%20es%20el,equipo%20a%20crecer%20y%20mejorar.>

- [23] "Equipo de desarrollo (development team)". Proyectos Ágiles.
<https://proyectosagiles.org/equipo->

- [33] ¿Qué es MySQL? (s.f.). Desarrollo tecnológico para empresas :: Consultora tecnológica y de desarrollo. <https://www.esepestudio.com/noticias/que-es-mysql#:~:text=MySQL%20es%20un%20sistema%20de%20administraci%C3%B3n%20de%20bases,simple%20archivo%20hasta%20sistemas%20relacionales%20orientados%20a%20objetos.>
- [34] Flores, F. (2022, 22 de julio). Qué es visual studio code y qué ventajas ofrece. OpenWebinars.net. <https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/>
- [35] Colaboradores de los proyectos Wikimedia. (2010, 1 de septiembre). GitHub - Wikipedia, la enciclopedia libre. Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/GitHub#:~:text=GitHub%20es%20una%20forja%20\(plataforma%20de%20desarrollo%20colaborativo\),opera%20GitHub%20fue%20escrito%20en%20Ruby%20on%20Rails.](https://es.wikipedia.org/wiki/GitHub#:~:text=GitHub%20es%20una%20forja%20(plataforma%20de%20desarrollo%20colaborativo),opera%20GitHub%20fue%20escrito%20en%20Ruby%20on%20Rails.)
- [36] Desarrollo de software web con laravel: Beneficios e inconvenientes. (s.f.). Novadevs. <https://novadevs.com/publicaciones/por-que-usar-laravel-en-el-desarrollo-de-software/>
- [36] Configurar laravel sanctum para la autenticación SPA y por API tokens. (s.f.). Página principal - Desarrollolibre. <https://www.desarrollolibre.net/blog/laravel/configurar-laravel-sanctum-para-la-autenticacion-spa-y-por-api-tokens>
- [37] Laravel - the PHP framework for web artisans. (s.f.). Laravel - The PHP Framework For Web Artisans. <https://laravel.com/docs/10.x/validation>
- [39] Laravel - the PHP framework for web artisans. (s.f.-a). Laravel - The PHP Framework For Web Artisans. <https://laravel.com/docs/10.x/requests#main-content>
- [40] Node.js SDK node.js upload + image, video transformations | cloudinary. (s.f.). Cloudinary. https://cloudinary.com/documentation/node_integration
- [41] Laravel - the PHP framework for web artisans. (s.f.-a). Laravel - The PHP Framework For Web Artisans. <https://laravel.com/docs/10.x/helpers#method-fake>

- [42] Los distintos tipos de pruebas en software | Atlassian. (s.f.). Atlassian.
<https://www.atlassian.com/es/continuous-delivery/software-testing/types-of-software-testing>
- [43] Pruebas de carga: ¿Qué es y cómo funciona? - Atentus: Servicio de Monitoreo. (s.f.). Atentus: Servicio de Monitoreo.
<https://atentus.com/2022/11/10/pruebas-de-carga/>