ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE SISTEMA WEB DE GESTIÓN DE PEDIDOS EN MINIMARKETS

DESARROLLO DE UN BACKEND

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

PEDRO PABLO PÁEZ CORDERO

DIRECTOR: JUAN PABLO ZALDUMBIDE PROAÑO

DMQ, agosto 2023

CERTIFICACIONES

Yo, Pedro Pablo Páez Cordero 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.

PEDRO PABLO PÁEZ CORDERO

pedro.paez@epn.edu.ec

pedro_ppc@hotmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Pedro Pablo Páez Cordero, bajo mi supervisión.

JUAN PABLO ZALDUMBIDE PROAÑO

DIRECTOR

juan.zaldumbide@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.

Pedro Pablo Páez Cordero

DEDICATORIA

Este trabajo se lo dedico a mi mamá, que siempre hizo todo lo posible para guiarme y permitirme terminar mis estudios.

Pedro Pablo Páez Cordero

AGRADECIMIENTO

Agradezco a mi mamá, mi papá, mi hermana, mis sobrinos, mi cuñado, mi tía, mis primos y mi amigo Sebas; por siempre brindar todo su apoyo. Ellos estuvieron presentes durante momentos muy duros en mi vida, en los cuales nunca hubiera creído poder llegar a este punto y por eso siempre tendrán mi cariño y respeto.

Pedro Pablo Páez Cordero

ÍNDICE DE CONTENIDO

RESUMEN

El comercio *online* ha tenido un notable crecimiento durante los últimos años, gracias al avance del Internet. Esto ha dado paso al surgimiento de una gran cantidad de tiendas *online*, las cuales abarcan un amplio rango de productos, desde alimentos hasta vehículos y viviendas. Tanto pequeños como grandes negocios comercializan una gran variedad de productos y, por lo tanto, requieren de cierto tipo de organización y fluidez al tratar con los pedidos generados en sus tiendas virtuales. Es por esto que el objetivo de este proyecto consiste en desarrollar una página *web* que permita a los clientes realizar sus compras de forma conveniente, y a los negocios gestionar sus pedidos de forma eficaz y organizada.

En el presente proyecto se ha desarrollado e implementado un *backend* que permita la gestión de datos, como productos, cuentas de usuario, carritos de compras y pedidos, con el fin de brindar información al *frontend*. De esta manera, se ha logrado desarrollar un sistema gestor de pedidos que se adapte a las necesidades de diferentes tipos de negocios, tales como los *minimarkets*, al momento de tratar con pedidos generados en sus tiendas virtuales.

La estructura de este documento ha sido organizada de forma clara. En la primera sección se encuentra la descripción del problema, los objetivos, el alcance y marco teórico. En la segunda sección se encuentra el marco metodológico, el diseño de la arquitectura y las herramientas de desarrollo. En la tercera sección, se encuentra detallado cada paso seguido al momento de desarrollar la aplicación. Por último, la cuarta sección contiene las conclusiones obtenidas y recomendaciones para este proyecto.

PALABRAS CLAVE: backend, gestor, pedidos, compras

ABSTRACT

Online commerce has had a remarkable growth in recent years, thanks to the advancement of the Internet. This has given way to the emergence of a huge number of online stores, which cover a wide range of products, from food to cars and homes. Both small and large businesses sell a wide variety of products and, therefore, require a certain type of organization and fluidity when dealing with the orders generated in their virtual stores. That is why the objective of this project is to develop a web page that allows customers to make their purchases conveniently, and businesses to manage their orders in an efficient and organized manner.

In this project, a backend has been developed and implemented that allows data such as products, user accounts, shopping carts and orders, to be managed in order to provide information to the frontend. In this way, it has been possible to develop an order management system that adapts to the needs of different types of businesses, such as minimarkets, when dealing with orders generated in their virtual stores.

This project's structure has been clearly organized. In the first section you can find the description of the problem, the objectives, the scope and theoretical framework. The second section contains the methodological framework, architecture design, and development tools. In the third section, each step followed when developing the application, detailed. Finally, the fourth section contains the conclusions obtained and recommendations for this project.

KEYWORDS: backend, management, orders, shopping

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El comercio electrónico, también conocido como *e-commerce*, ha revolucionado el mundo del comercio [1]. Según A. Fonseca [2], se refiere a las ventas generadas por internet, cable o TV inteligente, que se realizan mediante pagos en línea.

Las tiendas virtuales representan al concepto de *e-commerce,* puesto que ofrecen la opción de realizar transacciones online de forma sencilla. Estas tiendas virtuales pueden ser encontradas en sitios *web* que ofrecen productos y servicios a los consumidores [2].

La gestión de pedidos es un elemento fundamental de una tienda virtual. Es importante contar con un sistema eficiente para recibir, procesar y entregar pedidos de manera rápida y segura. Los sistemas de gestión de pedidos ayudan a agilizar todo lo que implica un pedido, desde que es generado hasta su envío y seguimiento. El objetivo es brindar una experiencia satisfactoria al cliente [3].

Por otro lado, los *minimarkets* también se benefician de los sistemas de gestión de pedidos. Al contar con locales y bodegas de reducido espacio comparado con un supermercado, es de alta importancia una adecuada y eficiente organización de sus productos y pedidos [4].

Por tal motivo, se ha planteado el desarrollo del *backend* para el sistema web de gestión de pedidos en *minimarkets*, el cual les permite tener un control más preciso de su inventario, minimizar errores y agilizar los procesos de entrega al facilitar el envío de datos sobre productos, clientes y pedidos al componente de *frontend* mediante una API.

1.1 Objetivo general

- Desarrollar el *backend* para sistema *web* de gestión de pedidos en *minimarkets*

1.2 Objetivos específicos

- Levantar los requerimientos para el *backend*
- Diseñar una base de datos basada en los requerimientos levantados
- Codificar la lógica de los *endpoints* que provee el *backend*
- Verificar el funcionamiento del *backend*
- Realizar el despliegue

1.3 Alcance

El presente proyecto consiste en el desarrollo de una API que permite al frontend añadir, editar o eliminar información sobre productos, clientes y usuarios. El cliente puede buscar productos del catálogo en categorías como alimentos, productos de aseo personal y productos electrónicos y añadirlos a un carrito de compras. Si el cliente decide generar la compra, puede acceder a una página donde tendrá toda la información disponible para realizar su pago. Una vez hecho el pago, el cliente puede ver toda la información relacionada con su pedido en su perfil de usuario.

Los administradores son capaces de gestionar productos y cuentas de empleados. Los empleados tienen acceso a un historial de pedidos y una lista de pedidos activos, los cuales pueden categorizar como entregados o pendientes.

Existen 4 perfiles de usuario en el sistema, cada uno tiene permisos diferentes:

Superadministrador:

- Ingresar al sistema
- Administrar cuentas de administradores

Administrador:

- Ingresar al sistema
- Administrar cuentas de empleados
- Ver historial de pedidos
- Ver lista de pedidos activos
- Cambiar estado de pedidos
- Ver información sobre inventario
- Gestionar productos
- Gestionar inventario

Empleado:

- Ingresar al sistema
- Ver historial de pedidos

- Ver lista de pedidos activos
- Cambiar estado de pedidos

Cliente:

- Registrarse como usuario
- Ingresar y salir de su cuenta de cliente
- Gestionar su información personal
- Ver historial de sus pedidos
- Navegar por el catálogo de productos
- Filtrar productos del catálogo
- Añadir o remover productos en su carrito de compras
- Revisar el estado de su pedido

1.4 Marco teórico

Backend

Un *backend*, es el término utilizado en el desarrollo de *software* para referirse a todo lo que maneja la lógica de una aplicación de forma transparente para el usuario. El *backend* es el encargado de gestionar varias funciones como la conexión con la base de datos y su posterior uso para el almacenamiento de información y asegurarse de la optimización y seguridad de la aplicación. [5].

La aplicación web de este proyecto se desarrolla utilizando PHP como lenguaje de programación para el *backend.*

Framework

Un *framework* puede ser descrito como una plantilla la cual contiene un conjunto de librerías, clases y herramientas que pueden ser utilizadas para un desarrollo sólido de una aplicación. Existen *frameworks* para distintos lenguajes de programación que proporcionan funcionalidades ya implementadas que permiten al desarrollador enfocarse únicamente en el código de su aplicación [6].

Laravel

Laravel es un *framework*, el cual permite el desarrollo de aplicaciones utilizando el lenguaje PHP. Laravel proporciona una gran cantidad de herramientas y características como soporte para bases de datos, autenticación de usuarios, enrutamiento y plantillas para la creación de vistas [7].

Utilizar este *framework* es una elección estratégica para garantizar un desarrollo eficiente y escalable del proyecto.

Integrated Development Environment

Un *Integrated Development Environment* (IDE) es un programa utilizado por desarrolladores de *software* para la creación de aplicaciones. Esta herramienta ofrece varias funciones como un editor de código y un compilador. Los IDEs proporcionan comodidad y garantizan eficiencia a los desarrolladores gracias a su gran variedad de características [8].

Visual Studio Code se destaca como el entorno de desarrollo preferido para este proyecto debido a su amplia lista de características y su versatilidad.

2 METODOLOGÍA

Las metodologías de desarrollo son un conjunto de métodos que contribuyen al desarrollo de *software*. Trabajar con una metodología es vital para evitar la desorganización de un proyecto y eventualmente un producto final que no cumpla con las expectativas del cliente y del equipo de desarrollo. Uno de sus objetivos siempre es la continua mejora del proyecto, con la finalidad de alcanzar un nivel de calidad apropiado [9].

Este proyecto busca utilizar métodos de desarrollo de *software* que sean consistentes con sus objetivos. En este caso, el método Scrum es el más adecuado.

Scrum es una metodología ágil de desarrollo de software en el cual se hacen entregas parciales del producto final. Estas entregas se las realiza de forma iterativa, eventualmente creando un producto de forma incremental. Scrum es utilizado para gestionar proyectos complejos con cambios constantes [10].

Roles

Dentro de la metodología Scrum existen tres roles principales: *Product Owner, Scrum Master y Development Team.* Cada rol tiene su participación en asegurarse que el proyecto tenga motivo de existir y que además fluya de la mejor manera [11].

Product Owner

El *Product Owner* es el representante del cliente, es quien contrata al equipo de desarrollo y es quien se encarga de priorizar requerimientos. Debe tener una idea clara de los requerimientos de la empresa a la que representa y también de los usuarios [11].

Scrum *Master*

El Scrum *Master* es el líder del equipo de Scrum y es el encargado de dirigir al equipo en las mejores prácticas y lineamientos de la metodología Scrum. El Scrum Master trabaja constantemente con el *Product Owner* y trata de crear un ambiente colaborativo entre el equipo de trabajo [11].

Development Team

El equipo de desarrollo es el encargado de implementar y entregar el producto. El equipo de desarrollo es un grupo que selecciona las tareas del *Product Backlog* que completará durante el *Sprint* siguiente. El equipo de desarrollo es un grupo que incluye programadores, ingenieros de pruebas, diseñadores, etc. [11].

Tabla I Asignación de roles

ROLES	NOMBRES
Product Owner	Ing. Juan Pablo Zaldumbide
Scrum Master	Ing. Juan Pablo Zaldumbide
Development Team	Pedro Páez

Historia de Usuario

Es una descripción corta y detallada de una necesidad del usuario. Al crear una historia de usuario es importante describir el rol, funcionalidad y el resultado [13]. Las historias de usuario de este proyecto pueden ser encontradas en el ANEXO II Manual técnico.

Tabla II Historia de usuario HU001

HISTORIA DE USUARIO	
Identificador (ID): HU001	**Usuario:** Cliente
Nombre Historia: Registrar Usuario	
Prioridad: Alta	**Riesgo en Desarrollo:** Alta
Iteración: 3	
Responsable: Pedro Páez	
Descripción: El backend permite a los roles de Cliente y Administrador crear una cuenta de usuario y así registrarse en el sistema.	
Observación: El cliente debe estar registrado para poder realizar transacciones en el sistema.	

Product Backlog

Es una lista organizada de todas las tareas a ser desarrolladas durante un proyecto. En un *Product Backlog* se puede organizar y dar prioridad a los elementos de mayor importancia, lo cual es clave en Scrum. Inicialmente, no todos los requerimientos deben ser definidos y pueden ser añadidos conforme sea necesario. Un *Product Backlog* requiere de mantenimiento y actualizaciones, la persona encargada de esto es el Product Owner [12].

Tabla III Formato del Product Backlog

PRODUCT BACKLOG				
ID - HU	**HISTORIA DE USUARIO**	**SPRINT**	**ESTADO**	**PRIORIDAD**
HU001	Registro de usuario	1	Finalizado	Alta
HU002	Inicio y cierre de sesión	1	Finalizado	Alta

Sprint Backlog

Es una lista de tareas tomadas del *Product Backlog* que serán desarrolladas durante un *sprint*. Cada tarea está compuesta por "bloques" que se combinan en una entrega. El *Sprint Backlog* puede ser ajustado en cualquier momento mientras el *sprint* está siendo desarrollado. [12].

Tabla IV Formato del Sprint Backlog

SPRINT BACKLOG						
ID - SB	**NOMBRE**	**MÓDULO**	**ID - HU**	**HISTORIA DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
SB000	Configuración del ambiente de desarrollo	-----	-----	-----	Recopilación y definición de requerimientos. Elaboración del Modelo de Datos. Roles de los usuarios. Elaboración de la Base de datos	15H

2.1 Diseño de la Arquitectura

Una arquitectura es de suma importancia en un proyecto como este, el funcionamiento y mantenibilidad del mismo se verán afectados por el correcto uso de dicha arquitectura.

Patrón arquitectónico

El patrón arquitectónico a ser usado es el de Modelo-Vista-Controlador, este patrón ofrece mayor flexibilidad y escalabilidad al desarrollar aplicaciones ya que separa la lógica de la vista y de los datos, así clasificándolos. Lo que se consigue con su aplicación es mejor reutilización de código, mejor mantenibilidad y hasta facilidad de lectura del código [14].

- **Modelo:** Es el componente que representa los datos y la lógica de negocios de la aplicación. Se encarga de almacenar, modificar, eliminar y acceder a la información que el controlador requiera. También es responsable de realizar cálculos y validaciones de datos antes de ser mostrados en la vista [14].
- **Vista:** Es el componente que se encarga de presentar los datos al usuario y de gestionar su interacción con la aplicación. La vista es la interfaz gráfica que el usuario utiliza para interactuar con la aplicación. Se actualiza según los datos provenientes del modelo [14].
- **Controlador:** Es el componente que actúa como intermediario entre el modelo y la vista. Se encarga de recibir las acciones del usuario, procesarlas y actualizar el modelo y la vista según sea necesario [14].

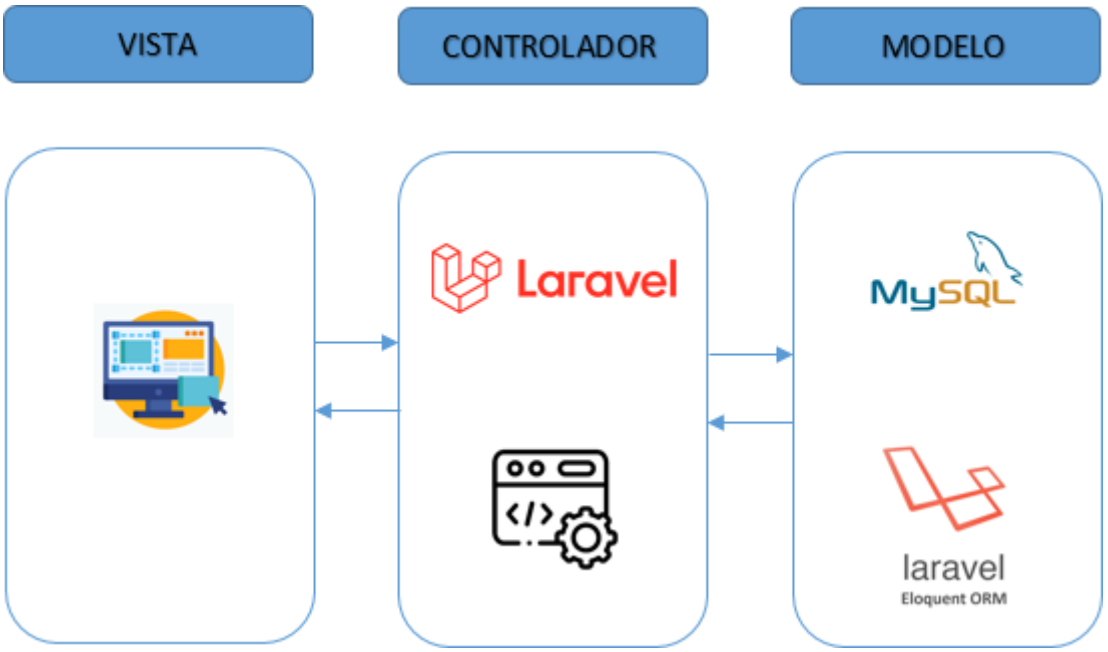


Fig. 1: Patrón arquitectónico API REST

2.2 Herramientas de desarrollo

Tabla V Herramientas para el desarrollo del backend

Herramienta	Justificación
Laravel	Laravel como *framework* para este proyecto proporciona una base sólida para el desarrollo y garantiza la creación de una aplicación web de alta calidad y rendimiento [7].
Composer	Composer mejora significativamente el proceso de desarrollo de Laravel al simplificar la gestión de dependencias [15].
MySQL	Dado el volumen de los datos que este proyecto debe gestionar, MySQL se presenta como una solución robusta y confiable para la administración de bases de datos relacionales [16].
Visual Studio Code	Visual Studio Code es una poderosa herramienta para la escritura y ejecución de código [17], con ella los desarrolladores pueden aumentar su productividad y obtener resultados de alta calidad en sus tareas de desarrollo de software.
Xampp	Para este proyecto, es fundamental contar con un gestor de bases de datos confiable. Xampp se considera la herramienta preferida, ya que proporciona una solución que incluye phpMyAdmin para la gestión de bases de datos SQL [18].
Cloudinary	Cloudinary es una plataforma de almacenamiento de imágenes en la nube [19]. Esta plataforma permite ahorrar espacio físico ya que las imágenes se

	almacenan de forma segura en la nube de Cloudinary.
Thunder Client	Es una extensión para Visual Studio Code que ofrece la capacidad de probar llamadas a una API. Esta funcionalidad es especialmente útil para probar y validar las rutas de tipo API que ofrece este backend [20].

3 RESULTADOS

A continuación, se detallan los resultados de cada *sprint*.

Sprint 0. Acondicionamiento del ambiente de trabajo

Basándose en lo especificado en el *Sprint Backlog*, el sprint 0 contiene las siguientes tareas:

- Instalar herramientas para el desarrollo
- Realizar modelo de base de datos

Instalar herramientas para el desarrollo

Visual Studio Code

El entorno de desarrollo proporciona una gran ayuda para el desarrollo en PHP, ofreciendo un amplio soporte de extensiones que mejoran la experiencia del usuario.

Fig. 2: Logo de Visual Studio Code

Xampp

Este programa ofrece la capacidad de crear y utilizar una base de datos local para realizar pruebas con MySQL. Además, permite la ejecución de Apache para visualizar la base de datos desde un navegador y ofrece una interfaz intuitiva como phpMyAdmin.

Fig. 3: Panel de control de Xampp

Realizar el modelo de la base de datos

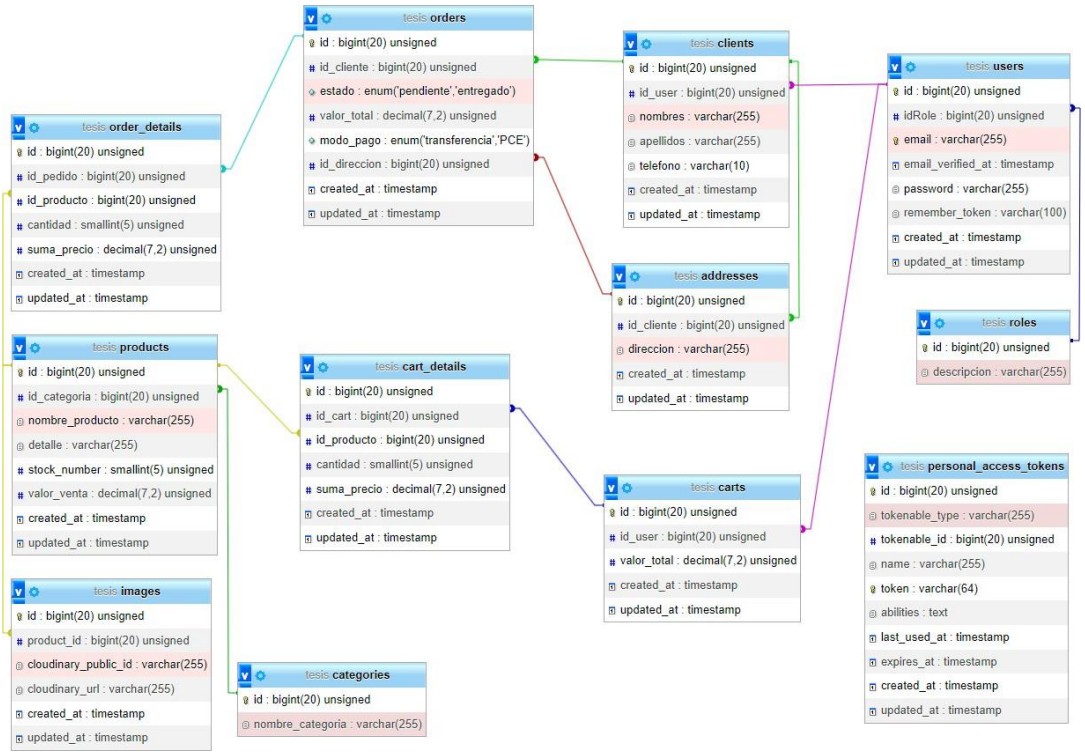


Fig. 4: Modelo de la base de datos en phpMyAdmin

Sprint 1. *Login*, registro y gestión de cuentas de administrador

Basándose en lo especificado en el *Sprint Backlog*, el *sprint* 1 contiene las siguientes tareas:

- Implementación de librería de Laravel para la autenticación de usuarios
- Ingreso de un usuario superadmin en la base de datos de forma manual
- Ingreso al sistema a través de endpoint
- Creación de cuenta de cliente por medio de datos recibidos a través de un *endpoint*
- Implementación de rutas para la creación y eliminación de cuentas de administrador
- Implementación de rutas para la creación y eliminación de cuentas de empleado

Implementación de librería de Laravel para la autenticación de usuarios

La herramienta a utilizar para la autenticación de usuarios es Laravel Sanctum, la cual utiliza muchas características propias de Laravel para lograr un sistema seguro.

Ingreso de un usuario superadmin en la base de datos de forma manual

Para lograr esto, se utiliza un *seeder* el cual creará al único usuario con rol de superadministrador del sistema. Se ingresan los datos del tipo de rol, email y contraseña como se observa en la Fig. 5.

```php
public function run(): void
{
    $data = [
        ['idRole' => 0, 'email' => 'pedro@gmail.com', 'password' => Hash::make(12345678)]
    ];

    User::insert($data);
}
```

Fig. 5: Función seeder

Ingreso al sistema a través de *endpoint*

El *login* utiliza una función que primero valida los datos de entrada, en caso de pasar la validación intenta ingresar al usuario al sistema, crea un *bearer token* y envía un mensaje de respuesta en formato JSON con este *token* como se observa en la Fig. 6.

```php
$request->validate([
    'email' => 'required|email',
    'password' => 'required',
]);

try{
    $user = User::where('email', $request['email'])->first();
    $credentials = $request->only('email', 'password');

    if($user == NULL){
        return response()->json(['error' => 'Invalid email or password'], 401);
    }

    if (Auth::attempt($credentials)) {
        $user = Auth::user();
        $token = $user->createToken('Personal Access Token')->plainTextToken;

        return response()->json([
            'message' => 'Successfully logged in',
            'access_token' => $token,
            'token_type' => 'Bearer',
            'user_role' => $user->idRole,
            'user_id' => $user->id
        ]);
    } else {
        return response()->json(['error' => 'Invalid email or password'], 401);
    }
} catch (\Exception $e){
    Log::error("Error loggging in: " . $e->getMessage());
    return response()->json(['error' => 'Failed to log in'], 500);
}
```

Fig. 6: Función para iniciar sesión

El *logout* es sencillo, se elimina el *token* del usuario y se envía un mensaje de respuesta con la confirmación tal como se observa en la Fig. 7.

```php
public function logout(Request $request){
    $user = $request->user();
    $user->tokens()->delete();
    return response()->json(['message' => 'Logged out successfully']);
}
```

Fig. 7: Función para cerrar sesión

Creación de cuenta de cliente por medio de datos recibidos a través de un *endpoint*

Al momento de registrar un cliente es necesario tomar en cuenta que en la base de datos los datos del cliente se almacenan en dos tablas, una contiene los nombres, apellidos del cliente y la otra contiene su información de *login* como *email* y contraseña.

El método implementado para lograr esta función valida primero los datos de entrada, luego los ingresa en las tablas de cliente y usuario y por último crea un *token*, el cual lo incluye en un mensaje de respuesta como se observa en la Fig. 8.

```php
public function register(Request $request){
    $validator = Validator::make($request->all(), [
        'nombres' => ['required', 'string', 'max:255'],
        'apellidos' => ['required', 'string', 'max:255'],
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'password' => ['required', 'string', 'min:8', 'confirmed'],
        //password_confirmation
    ]);

    if ($validator->fails()) {
        return response()->json(['errors' => $validator->errors()], 422);
    }

    $user = User::create([
        'email' => $request->input('email'),
        'password' => bcrypt($request->input('password')),
    ]);

    $client = Client::create([
        'id_user' => $user->id,
        'nombres' => $request->input('nombres'),
        'apellidos' => $request->input('apellidos'),
    ]);

    $token = $user->createToken('API Token')->plainTextToken;

    return response()->json([
        'client' => $client,
        'user' => $user,
        'token' => $token,
    ], 201);
}
```

Fig. 8: Función para registrar cliente

Implementación de rutas para la creación y eliminación de cuentas de administrador

La creación de cuentas de administrador es realizada únicamente por el superadministrador, el cual debe estar autenticado al momento de necesitar crear una cuenta.

La función implementada para esto es similar a la función que registra clientes, simplemente no registra al administrador como un cliente, solamente ingresa sus datos en la tabla de usuarios como se observa en la Fig. 9.

```php
public function registerAdmin(Request $request){
    $validator = Validator::make($request->all(), [
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'password' => ['required', 'string', 'min:8', 'confirmed'],
        //password_confirmation
    ]);

    if ($validator->fails()) {
        return response()->json(['errors' => $validator->errors()], 422);
    }

    $user = User::create([
        'idRole' => 1,
        'email' => $request->input('email'),
        'password' => bcrypt($request->input('password')),
    ]);

    $token = $user->createToken('API Token')->plainTextToken;

    return response()->json([
        'user' => $user,
        'token' => $token,
    ], 201);
}
```

Fig. 9: Función para registrar administrador

Para la eliminación de cuentas de administrador y empleado se utiliza un método que recibe como parámetro el ID del usuario registrado, luego verifica si existe. Los únicos roles que pueden eliminar cuentas son el superadministrador y el administrador. Los administradores no pueden eliminar sus propias cuentas y la cuenta de superadministrador no puede ser eliminada por nadie como se observa en la Fig. 10.

```php
public function delete($id){
    $user = User::find($id);

    $idRole = Auth::user()->idRole;

    if($idRole == 0 || $idRole == 1){
        if($user == null){
            return response()->json(['response' => 'User does not exist']);
        } else if($user->idRole == 0){
            return response()->json(['response' => 'Cannot delete that user']);
        } else if($user->idRole == 1 && $idRole == 1){
            return response()->json(['response' => 'Admins cannot delete their own accounts']);
        } else {
            //Cambia el valor de id_user a null antes de borrar al user para que no haya problemas de constraints
            DB::table('clients')->where('id_user', $id)->update(['id_user' => null]);
            $user->tokens()->delete();
            $user ->delete();
            return response()->json(['response' => 'User deleted']);
        }
    } else {
        return response()->json(['response' => 'Unauthorized Role']);
    }
}
```

Fig. 10: Función para borrar cuentas de usuario

Implementación de rutas para la creación y eliminación de cuentas de empleado

El proceso de creación de cuentas de empleado es similar al de creación de cuentas de administrador como se puede observar en la Fig. 11, este proceso solo puede ser ejecutado por cuentas de administrador.

```php
public function registerEmployee(Request $request){
    $validator = Validator::make($request->all(), [
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'password' => ['required', 'string', 'min:8', 'confirmed'],
        //password_confirmation
    ]);

    if ($validator->fails()) {
        return response()->json(['errors' => $validator->errors()], 422);
    }

    $user = User::create([
        'idRole' => 2,
        'email' => $request->input('email'),
        'password' => bcrypt($request->input('password')),
    ]);

    return response()->json([
        'user' => $user,
        'message' => 'Employee account created',
    ], 201);
}
```

Fig. 11: Función para registrar empleados

La eliminación de cuentas de empleado la puede realizar el superadministrador o cualquier administrador y su metodología se la observa en la Fig. 10.

Sprint 2. Gestión de productos, navegación por catálogo y carrito de compras

Basándose en lo especificado en el *Sprint Backlog*, el sprint 2 contiene las siguientes tareas:

- Implementación de rutas para la creación, modificación y eliminación de productos del catálogo
- Implementación de almacenamiento en la nube para imágenes
- Implementación de ruta para mostrar lista de productos
- Implementación de método para filtrado de productos
- Implementación de rutas para crear, mostrar y eliminar carrito de compras
- Implementación de ruta para agregar y eliminar productos de un carrito
- Implementación de método para calcular precio total de un carrito

Implementación de rutas para la creación, modificación y eliminación de productos del catálogo

El primer paso de esta implementación requiere una tabla de productos en la base de datos, la cual se crea por medio de una migración en donde se especifican todos los campos necesarios para la lógica de negocio como se indica en la Fig. 12.

```php
public function up(): void
{
    Schema::create('products', function (Blueprint $table) {
        $table->id()->index();
        $table->unsignedBigInteger('id_categoria');
        $table->foreign('id_categoria')
                ->references('id')
                ->on('categories')
                ->onUpdate('cascade');
        $table->string('nombre_producto');
        $table->string('detalle')->nullable();
        $table->smallInteger('stock_number')->unsigned();
        $table->decimal('valor_venta', 7, 2)->unsigned();
        $table->timestamps();
    });
}
```

Fig. 12: Migración para tabla productos

Para la creación de un producto en la tabla se utiliza un método que primero valide los datos de entrada, como el stock y valor, ya que estos no pueden ser negativos, entre otras validaciones. Esto se puede observar en la Fig. 13.

Luego, se usa el método *create()* para ingresar los datos del producto a la base de datos, en caso de haberse provisto de una imagen, se llama al método correspondiente para almacenarla en la nube y asociarla con el producto en la base de datos como lo indica la Fig. 14.

```
$request->validate([
    'id_categoria' => 'required|integer|numeric|between:1,3',
    'nombre' => 'required|string',
    'detalle' => 'string',
    'stock' => 'required|integer|numeric|gte:0',
    'valor' => 'required|decimal:2|numeric|gte:0',
    'file' => 'image|max:10240'
]);
```

Fig. 13: Validación de datos

```
try{
    $product = Product::create([
        'id_categoria' => $request->input('id_categoria'),
        'nombre_producto' => $request->input('nombre'),
        'detalle' => $request->input('detalle'),
        'stock_number' => $request->input('stock'),
        'valor_venta' => $request->input('valor')
    ]);

    //Se llama al metodo para ingresar la imagen
    if ($request->hasFile('file') && !is_null($product->id)) {
        $imageController = app(ImageController::class);
        $imageController->store($request, $product->id);
    }

    return response()->json(['message' => "Product saved successfully"], 201);

} catch (\Exception $e){
    Log::error("Error saving product: " . $e->getMessage());
    return response()->json(['error' => 'Failed to save product'], 500);
}
```

Fig. 14: Función para insertar producto

Para la modificación de productos, lo primero es validar y luego buscar el producto a modificar en la base de datos utilizando el ID que se recibe como parámetro. Una vez encontrado, se lo modifica con los datos validados como se indica en la Fig. 15.

```php
try{
    $product = Product::find($request->input('id'));

    $product->update([
        'id_categoria' => $request->input('id_categoria'),
        'nombre_producto' => $request->input('nombre_producto'),
        'detalle' => $request->input('detalle'),
        'stock_number' => $request->input('stock_number'),
        'valor_venta' => $request->input('valor_venta')
    ]);

    return response()->json(['message' => "Product updated successfully"]);

} catch (\Exception $e){
    Log::error("Error updating product: " . $e->getMessage());
    return response()->json(['error' => 'Failed to update product'], 500);
}
```

Fig. 15: Función para modificar producto

Para la eliminación de productos, solamente es necesario validar el ID que se recibe como parámetro. La lógica es similar a la de modificación; primero se busca el producto y luego se lo elimina con el método *delete(),*el único cambio necesario es eliminar las imágenes asociadas al producto junto con el producto como se indica en la Fig. 16.

```php
$product = Product::find($request->input('id'));

if ($product) {
    //Se llama al metodo para destruir las imagenes antes de borrar el producto
    $imageController = app(ImageController::class);
    $imageController->destroyAll($product->id);

    $product->delete();
    return response()->json(['message' => 'Product deleted successfully'], 204);
} else {
    return response()->json(['message' => 'Product not found'], 404);
}
```

Fig. 16: Función para eliminar producto

Implementación de almacenamiento en la nube para imágenes

Al momento de recibir una imagen del frontend, es necesario verificar que en realidad se trate de una imagen y no de otro tipo de archivo. Una vez hecho esto se utiliza la *facade* de Cloudinary para subir la imagen, esto toma una sola línea de código. Al subir la imagen, se obtiene el ID público y junto con el ID del producto y la URL de la imagen se crea un registro en la base de datos como se observa en la Fig. 17.

```php
$uploadedFileUrl = Cloudinary::upload($request->file('file')->getRealPath())->getSecurePath();
$publicId = Cloudinary::getPublicId($uploadedFileUrl);

try{
    Image::create([
        'product_id' => $request->input('id_producto'),
        'cloudinary_public_id' => $publicId,
        'cloudinary_url' => $uploadedFileUrl
    ]);

    return response()->json(['message' => "Image saved successfully"], 201);
```

Fig. 17: Función para insertar imágenes

Para visualizar una o más imágenes asociadas a un producto, se busca en la base de datos utilizando el ID del producto como lo muestra la Fig. 18.

```php
$imagenes = DB::table('images')
        ->where('product_id', $request->input('id_producto'))
        ->get();

return response()->json([$imagenes], 201);
```

Fig. 18: Función para mostrar imagen

Al eliminar una imagen se espera que el frontend envíe el ID público de la imagen, con lo cual se realiza una búsqueda en la base de datos. Si se encuentra un registro se lo elimina de la tabla con la función *delete()* y también se lo elimina de Cloudinary como se observa en la Fig. 19.

```php
$imagen = Image::where('cloudinary_public_id', $request->input('cloudinary_id'))->first();

if ($imagen) {
    $imagen->delete();
    Cloudinary::destroy($request->input('cloudinary_id'));
    return response()->json(['message' => 'Image deleted successfully']);
} else {
    return response()->json(['message' => 'Image not found'], 404);
}
```

Fig. 19: Función para eliminar imágenes

Implementación de ruta para mostrar lista de productos

Esta implementación resulta sencilla ya que lo único que la función realiza es retornar todos los productos encontrados en la base de datos como se muestra en la Fig. 20.

```php
try{
    $products = Product::all();

    if ($products->isEmpty()) {
        return response()->json(['message' => 'No products available'], 404);
    }

    return $products;

} catch (\Exception $e){
    Log::error("Error loading products: " . $e->getMessage());
    return response()->json(['error' => 'Failed to load products'], 500);
}
```

Fig. 20: Función para mostrar lista de productos

Implementación de método para filtrado de productos

Este es el primer caso en el que se necesita implementar un *query* SQL más específico. La función recibe una palabra y procede a buscar productos o detalles de productos en la base de datos que sean similares a dicha palabra, como se lo puede observar en la Fig. 21. Por ejemplo: buscar “pap” retornaría productos como “papas” y “papel higiénico”, entre otros.

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

    $keyword = $request->input('keyword');

    try{
        $products = DB::table('products')
            ->where('nombre_producto', 'like', '%'.$keyword.'%')
            ->orWhere('detalle', 'like', '%'.$keyword.'%')
            ->get();

        if($products->isEmpty() || $products == "" || $products == null){
        return response()->json(['message' => "Cannot find product"], 404);
        }

        return response()->json($products);

    } catch (\Exception $e){
        Log::error("Error searching product: " . $e->getMessage());
        return response()->json(['error' => 'Failed to search product'], 500);
    }
}
```

Fig. 21: Función para buscar productos

Implementación de rutas para crear, mostrar y eliminar carrito de compras

Cabe mencionar que este proyecto toma en cuenta al carrito y su contenido como dos entidades diferentes pero atadas una a la otra por medio del uso de llaves foráneas en la base de datos. Se observan las migraciones para estas dos tablas en las Fig. 22 y Fig. 23.

```
Schema::create('carts', function (Blueprint $table) {
    $table->id()->index();
    $table->unsignedBigInteger('id_user');
    $table->foreign('id_user')
            ->references('id')
            ->on('users')
            ->onUpdate('cascade');
    $table->decimal('valor_total', 7, 2)->unsigned()->nullable();
    $table->timestamps();
});
```

Fig. 22: Migración para tabla carts

```
Schema::create('cart_details', function (Blueprint $table) {
    $table->id()->index();
    $table->unsignedBigInteger('id_cart');
    $table->foreign('id_cart')
            ->references('id')
            ->on('carts')
            ->onUpdate('cascade');
    $table->unsignedBigInteger('id_producto');
    $table->foreign('id_producto')
            ->references('id')
            ->on('products')
            ->onUpdate('cascade')
            ->onDelete('cascade');
    $table->smallInteger('cantidad')->unsigned();
    $table->decimal('suma_precio', 7, 2)->unsigned();
    $table->timestamps();
});
```

Fig. 23: Migración para tabla cart_details

Para crear un carrito, se verifica que el usuario no tenga ningún carrito activo, si no lo tiene se llama a la función *create()* y se inserta el valor de *id_user*, es decir el ID del usuario que tenga sesión activa en ese momento para crear un nuevo carrito como se observa en la Fig. 24.

```php
$id = Auth::id();

try{
    $cartExists = Cart::where('id_user', $id)->exists();

    if($cartExists){
        return response()->json(['message' => "Only one cart per account can be active at a time"], 409);
    } else {
        $cart = Cart::create([
            'id_user' => $id
        ]);
    }

    return response()->json(['message' => "Cart saved successfully", 'cart id' => $cart->id], 201);
```

Fig. 24: Función para crear un carrito

Para borrar un carrito, se verifica si existe un carrito asociado con el ID del usuario, si ese es el caso, se borra todos los registros de la tabla *cart_details* relacionados con el carrito. Por último, se elimina el carrito en sí como se muestra en la Fig. 25.

```php
$id = Auth::id();

try{
    $cart = DB::table('carts')
        ->where('id_user', $id)
        ->first();

    if ($cart) {
        DB::table('cart_details')
            ->where('id_cart', $cart->id)
            ->delete();

        DB::table('carts')
            ->where('id', $cart->id)
            ->delete();
    } else {
        return response()->json(['message' => "Cart does not exist"], 404);
    }

    return response()->json(['message' => "Cart deleted successfully"]);
```

Fig. 25: Función para eliminar un carrito

Mostrar todos los productos de un carrito tiene un cierto nivel de complejidad por la necesidad de un *query* SQL más detallado. Se realiza un *select* de tres tablas y se obtiene: el ID del producto, nombre del producto, detalle del producto, valor unitario, cantidad y precio total como se indica en la Fig. 26.

```php
$id = $request->input('id');

$results = DB::select("
    SELECT cd.id_producto, pr.nombre_producto, pr.detalle, pr.valor_venta, cd.cantidad, cd.suma_precio
    FROM cart_details cd
    JOIN products pr ON cd.id_producto = pr.id
    JOIN carts ca ON ca.id = cd.id_cart
    WHERE ca.id = :id
", ['id' => $id]);

if (empty($results)) {
    return response()->json(['message' => 'Cart is empty']);
}

return response()->json($results, 200);
```

Fig. 26: Función para mostrar los productos de un carrito

Implementación de ruta para agregar y eliminar productos de un carrito

Para añadir un producto al carrito se necesita primero verificar que el producto exista en la base de datos, lo cual se hace con la función *find().* Luego, se verifica que exista suficiente inventario basándose en la cantidad del producto que se quiere agregar. Si existe suficiente inventario se calcula el valor total multiplicando la cantidad por el precio unitario.

Una vez hecho esto, se procede a verificar si el producto a agregar ya existe en el carrito, en el caso que ya exista la cantidad especificada solo se suma a la existente, en caso que no exista se agrega el producto al carrito.

También es necesario actualizar el campo de valor total de la tabla *carts* para el carrito con el que se está tratando, una vez encontrado el carrito se ingresa el valor total si es que el campo se encontraba en *null*, caso contrario se lo actualiza con el nuevo valor total.

Cada vez que un producto se agrega al carrito, el inventario se actualiza para reflejar este cambio como se lo puede observar en la Fig. 27.

```php
$product = Product::find($request->input('id_producto'));

if($product){

    $quantity = $request->input('cantidad');

    if($product->stock_number < $quantity){
        return response()->json(['message' => 'Not enough stock'], 404);
    } else if ($product->stock_number == 0){
        return response()->json(['message' => 'Product out of stock'], 404);
    }

    $price = $product->valor_venta;
    $total = $quantity * $price;

    $productExists = CartDetail::where('id_cart', $request->input('id_cart'))
       ->where('id_producto', $request->input('id_producto'))
       ->first();

    $cart = Cart::where('id', $request->input('id_cart'))->first();

    if($productExists){
        $productExists->increment('cantidad', $quantity);
        $productExists->increment('suma_precio', $total);

        $product->decrement('stock_number', $quantity);

    } else {
        CartDetail::create([
            'id_cart' => $request->input('id_cart'),
            'id_producto' => $request->input('id_producto'),
            'cantidad' => $request->input('cantidad'),
            'suma_precio' => $total
        ]);

        $product->decrement('stock_number', $quantity);
    }

    if($cart->valor_total == null){
        $cart->update([
            'valor_total' => $total
        ]);
    } else {
        $cart->increment('valor_total', $total);
    }

    return response()->json(['message' => "Product added to cart successfully"], 201);

} else {
    return response()->json(['message' => 'Product not found'], 404);
}
```

Fig. 27: Función para agregar productos a un carrito

Cuando se elimina un producto del carrito se utiliza un proceso similar al de agregar, pero esta vez se reduce la cantidad especificada del producto y, por lo tanto, el precio total.

Cabe mencionar que valor total del carrito debe ser modificado también, por lo cual, dependiendo del valor total actual se decidirá lo que sucede que con ese valor como se puede observar en la Fig. 28.

Al eliminar un producto del carrito, el inventario se actualiza para reflejar este cambio.

```php
if($productExists){
    $productExists->decrement('cantidad', $quantity);
    $productExists->decrement('suma_precio', $total);

    //Si el valor total del carrito es null, solo se actualiza con el nuevo valor
    if($cart->valor_total == null){
        $cart->update([
            'valor_total' => $total
        ]);
    //Si al restar el total, el valor total se hace 0 o menos de 0, no resta, solo actualiza
    } else if ($cart->valor_total - $total <= 0){
        $cart->update([
            'valor_total' => 0.00
        ]);
    } else {
        $cart->decrement('valor_total', $total);
    }

    if ($productExists->cantidad <= 0) {
        $productExists->delete();
    }

    $product->increment('stock_number', $quantity);

    return response()->json(['message' => "Product removed from cart successfully"], 204);
} else {
    return response()->json(['message' => 'Product not found in cart'], 404);
}
```

Fig. 28: Función para eliminar un producto del carrito

Implementación de método para calcular precio total de un carrito

Anteriormente, se pudo apreciar cómo funciona el cálculo del precio total. Se utiliza el valor de cantidad de producto que el cliente quiera agregar y se lo multiplica por el valor unitario del producto como lo muestra la Fig. 29.

```php
$quantity = $request->input('cantidad');

$price = $product->valor_venta;
$total = $quantity * $price;
```

Fig. 29: Cálculo del precio total

Sprint 3. Gestión de pedidos, inventario y gestión de perfil de usuario

Basándose en lo especificado en el *Sprint Backlog*, el sprint 3 contiene las siguientes tareas:

- Implementación de rutas para modificar cantidad de inventario, además de visualizar el inventario
- Implementación de ruta para modificar datos personales del perfil del usuario
- Implementación de rutas para la creación y eliminación de pedidos

Implementación de rutas para modificar cantidad de inventario, además de visualizar el inventario

En este caso, se puede utilizar la ruta para actualizar un producto y obtener los resultados deseados al recibir un numero de ID y un nuevo valor para el inventario como lo muestra la Fig. 30.

```php
$product = Product::find($request->input('id'));

if ($product) {
    $product->update([
        'id_categoria' => $request->input('id_categoria'),
        'nombre_producto' => $request->input('nombre_producto'),
        'detalle' => $request->input('detalle'),
        'stock_number' => $request->input('stock_number'),
        'valor_venta' => $request->input('valor_venta')
    ]);
    return response()->json(['message' => "Product updated successfully"], 202);
} else {
    return response()->json(['message' => 'Product not found'], 404);
}
```

Fig. 30: Función para modificar inventario

Implementación de ruta para modificar datos personales del perfil del usuario

Para poder modificar un perfil es necesario obtener el ID del usuario autenticado y luego buscar los registros de cliente y direcciones que estén relacionados con ese usuario. Luego se modifica con la función *update()* como se muestra en la Fig. 31.

```php
$id = Auth::id();
$client = Client::find($id);
$address = Address::where('id_cliente', $id)->first();

if ($client) {
    $client->update([
        'nombres' => $request->input('nombres'),
        'apellidos' => $request->input('apellidos'),
        'telefono' => $request->input('telefono')
    ]);

    if($address){
        $address->update([
            'direccion' => $request->input('direccion'),
        ]);
    } else {
        Address::create([
            'id_cliente' => $id,
            'direccion' => $request->input('direccion')
        ]);
    }

    return response()->json(['message' => "Profile updated successfully"], 202);
} else {
    return response()->json(['message' => 'Profile not found'], 404);
}
```

Fig. 31: Función para modificar datos personales

Implementación de rutas para la creación y eliminación de pedidos

Para el módulo de creación se utiliza el ID del usuario autenticado para obtener el ID del cliente, también para obtener el valor total del carrito de compras. El ID del cliente sirve para obtener la dirección del cliente.

Una vez obtenidos estos datos se crea el pedido con la función estática *create()* como se muestra en la Fig. 32.

```php
$client = Client::where('id_user', $id)->first();
$valor = Cart::where('id_user', $id)->first();
$address = Address::where('id_cliente', $client->id)->first();

if($valor == null){
    return response()->json(['message' => 'User does not have a cart'], 404);
}

$order = Order::create([
    'id_cliente' => $client->id,
    'estado' => 'pendiente',
    'valor_total' => $valor->valor_total,
    'modo_pago' => $request->input('modo_pago'),
    'id_direccion' => $address->id
]);
```

Fig. 32: Función para crear un pedido

Luego, se obtienen todos los objetos añadidos al carrito y se los pasa a la tabla que contiene los detalles del pedido, una vez hecho esto se elimina el carrito de compras junto con todo su contenido como se observa en la Fig. 33.

```php
$cartDetails = DB::table('cart_details')
    ->where('id_cart', $valor->id)
    ->get();

foreach ($cartDetails as $cartDetail) {
    DB::table('order_details')->insert([
        'id_pedido' => $order->id,
        'id_producto' => $cartDetail->id_producto,
        'cantidad' => $cartDetail->cantidad,
        'suma_precio' => $cartDetail->suma_precio
    ]);
}

//Se llama al metodo para destruir el carrito despues de hecho el pedido
$cartController = app(CartController::class);
$cartController->destroy();
```

Fig. 33: Traspaso de productos y eliminación de carrito

Para eliminar una orden se utiliza el ID del usuario autenticado para obtener los datos del cliente y, con esos datos, obtener el pedido a destruir. Si el pedido existe, se eliminan todos los productos pertenecientes a ese pedido, luego se elimina el pedido en sí como lo indica la Fig. 34.

```php
$client = Client::where('id_user', $id)->first();
$order = Order::where('id_cliente', $client->id)->first();

if ($order) {
    OrderDetail::where('id_pedido', $order->id)->delete();
    $order->delete();
} else {
    return response()->json(['message' => "Order does not exist"], 404);
}

return response()->json(['message' => "Order deleted successfully"]);
```

Fig. 34: Función para eliminar pedido

Sprint 4. Historial y estado de pedidos

Basándose en lo especificado en el *Sprint Backlog*, el sprint 4 contiene las siguientes tareas:

- Implementación de ruta para mostrar el historial de pedidos de un cliente además sus estados
- Implementación de rutas para listar pedidos ya entregados o pendientes
- Implementación de ruta para modificar estado de pedido

Implementación de ruta para mostrar el historial de pedidos de un cliente además sus estados

Utilizando un SQL *query* con dos *JOIN* se obtienen los datos más pertinentes sobre los pedidos (nombres, apellidos, dirección, modo de pago, estado de pedido, etc.) basándose en el ID del cliente autenticado como lo indica la Fig. 35.

```php
$results = DB::select("
    SELECT os.id, cl.nombres, cl.apellidos, ad.direccion, os.valor_total, os.modo_pago, os.estado
    FROM orders os
    JOIN addresses ad ON os.id_cliente = ad.id_cliente
    JOIN clients cl ON os.id_cliente = cl.id
    WHERE cl.id_user = :id
", ['id' => $id]);

if (empty($results)) {
    return response()->json(['message' => 'No orders available']);
}

return response()->json($results, 200);
```

Fig. 35: Función para mostrar historial de pedidos

Implementación de rutas para listar pedidos ya entregados o pendientes

Esta implementación es similar a la vista anteriormente, con un pequeño cambio: no se utiliza el ID del usuario autenticado para realizar la búsqueda, lo que resulta en la lista completa de pedidos con sus respectivos estados como se muestra en la Fig. 36.

```
$results = DB::select("
    SELECT os.id, cl.nombres, cl.apellidos, ad.direccion, os.valor_total, os.modo_pago, os.estado
    FROM orders os
    JOIN addresses ad ON os.id_cliente = ad.id_cliente
    JOIN clients cl ON os.id_cliente = cl.id
    ");

if (empty($results)) {
    return response()->json(['message' => 'No orders available']);
}

return response()->json($results, 200);
```

Fig. 36: Función para listar pedidos

Implementación de ruta para modificar estado de pedido

Para modificar el estado de un pedido es necesario obtener el ID del pedido y su nuevo estado como parámetros. Se encuentra el pedido usando el ID y con la función *update()* se procede a modificar su estado como lo indica la Fig. 37.

```
$order = Order::find($id);

if ($order) {
    $order->update([
        'estado' => $request->input('estado')
    ]);
    return response()->json(['message' => "Order state updated successfully"], 202);
} else {
    return response()->json(['message' => 'Order not found'], 404);
}
```

Fig. 37: Función para modificar estado de pedidos

Sprint 5. Pruebas y despliegue

Basándose en lo especificado en el *Sprint Backlog*, el sprint 5 contiene las siguientes tareas:

- Realizar pruebas de integración
- Realizar pruebas de carga
- Realizar pruebas de estrés

- Desplegar

Realizar pruebas de integración

Para la realización de este tipo de pruebas se utiliza la herramienta Thunder Client [20].

Para verificar el correcto funcionamiento de la autenticación se ingresan datos de un usuario ya registrado en la base de datos como lo muestra la Fig. 38.

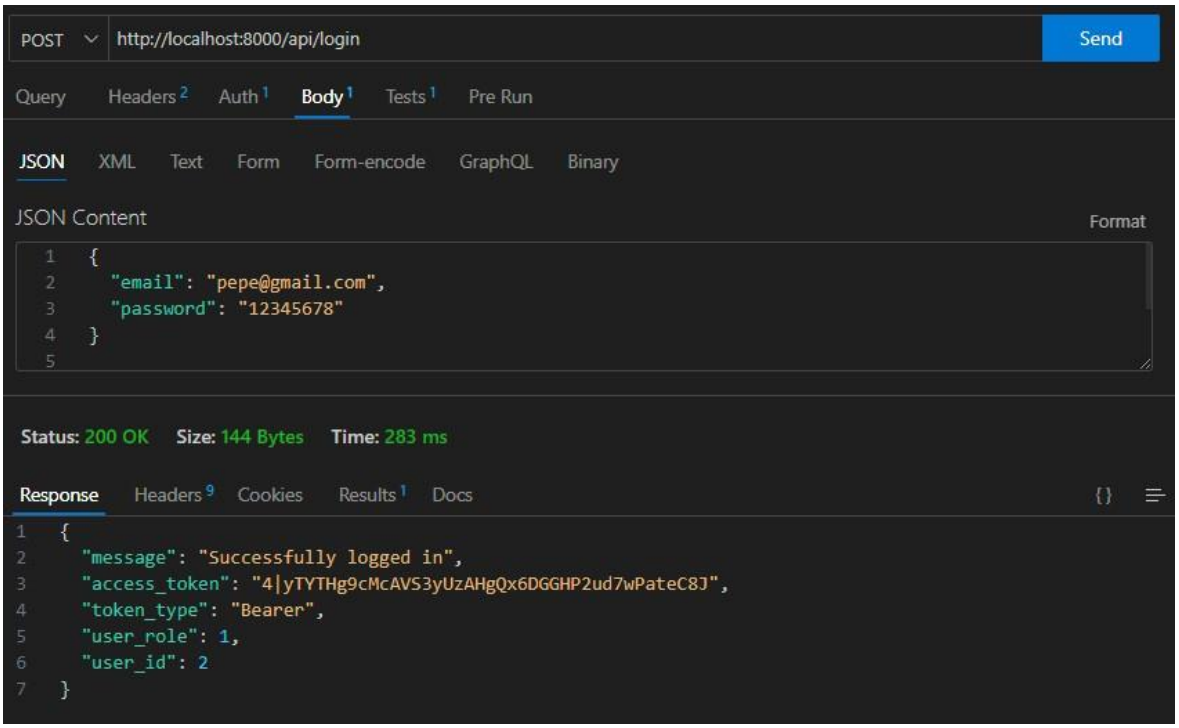


Fig. 38: Cuerpo de petición

Luego, se procede a verificar que se retorne un código de respuesta exitoso como lo indica la Fig. 39.

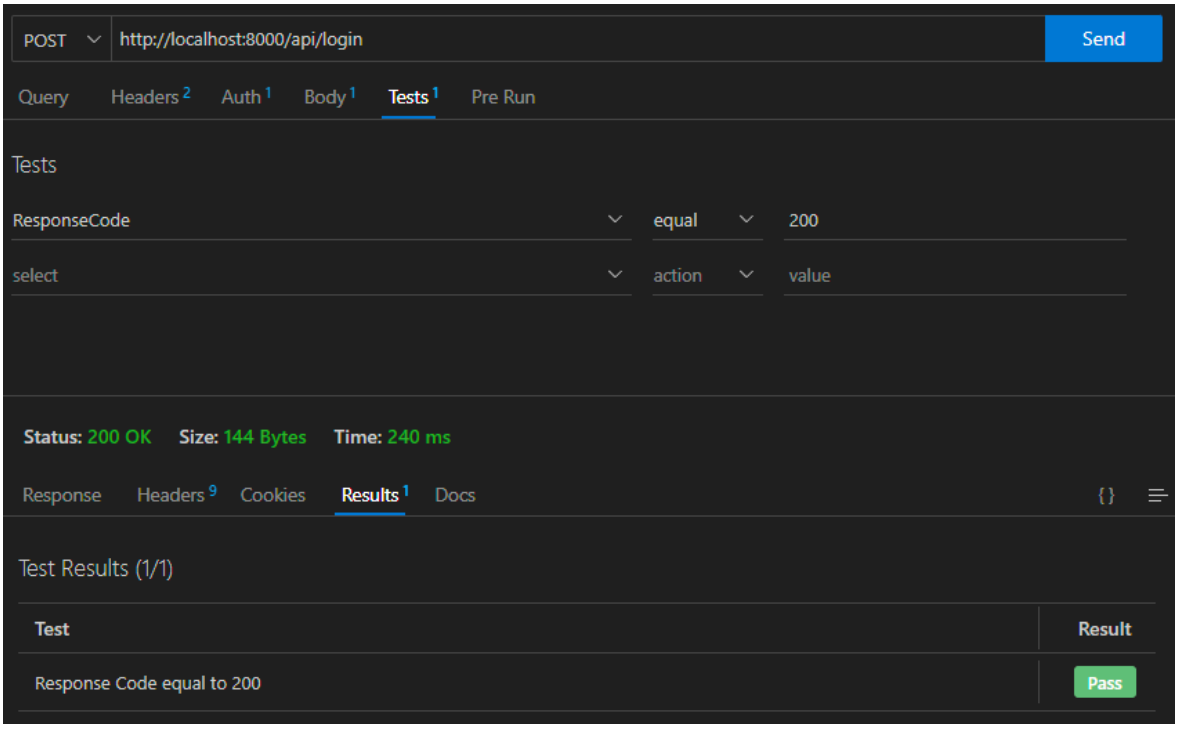


Fig. 39: Resultado de prueba

A continuación, se muestra el cuerpo de la petición para insertar un producto en la Fig. 40 y su respectivo código de respuesta en la Fig. 41.

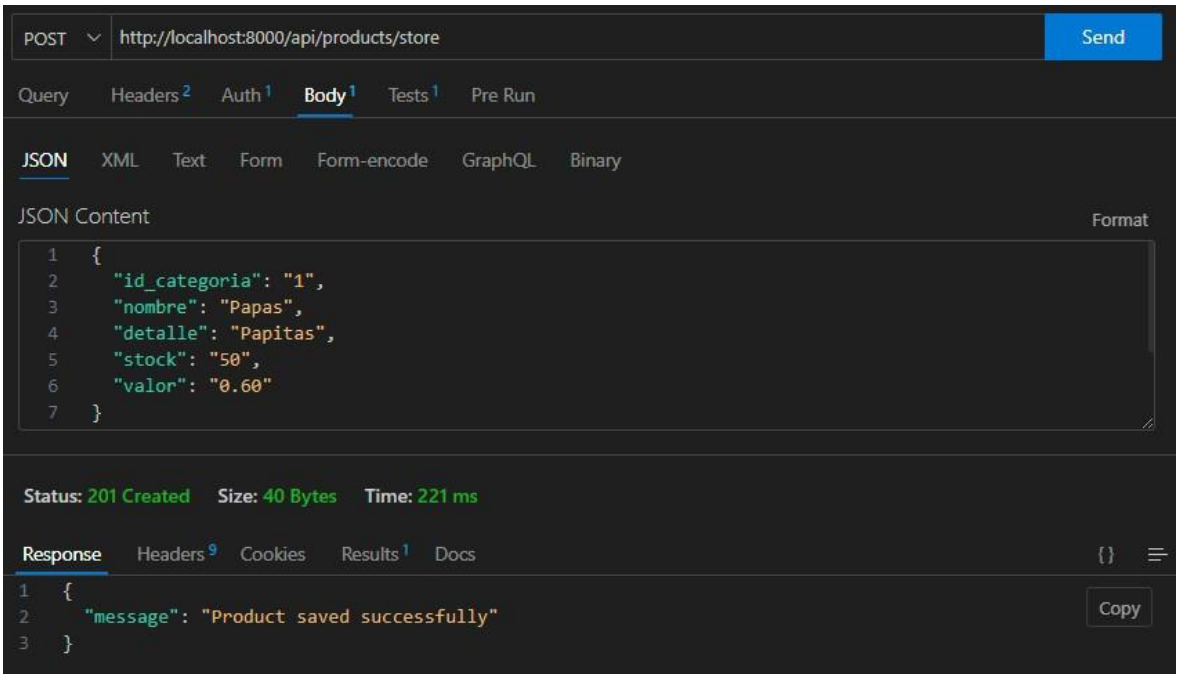


Fig. 40: Cuerpo de petición

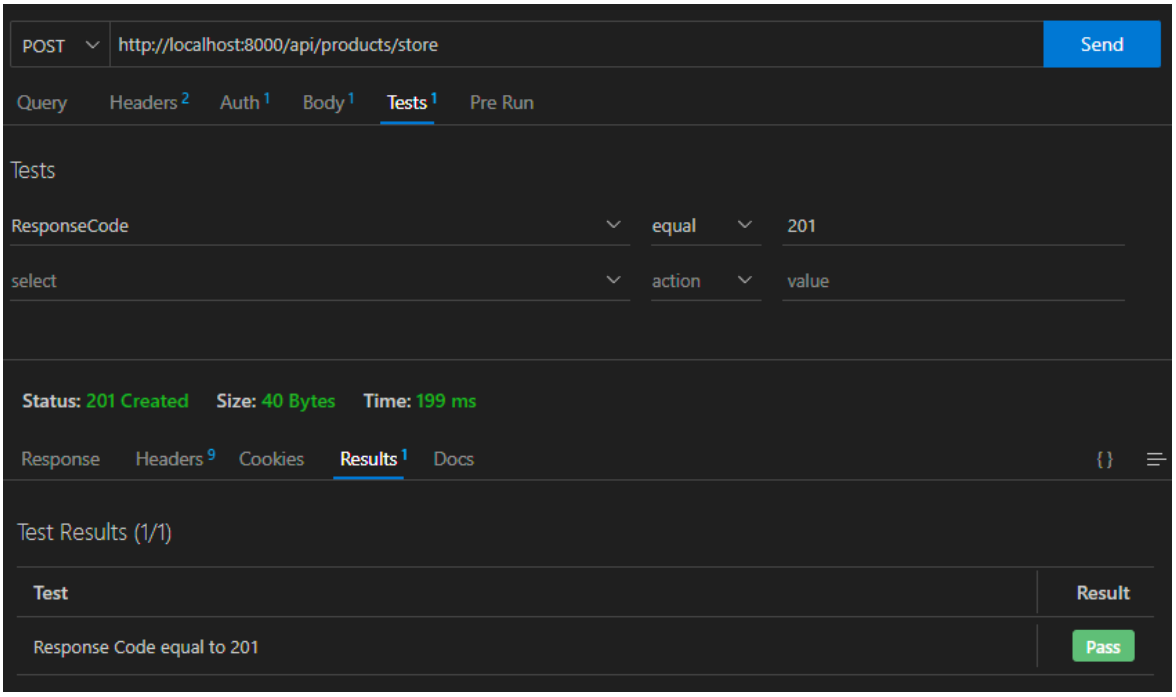


Fig. 41: Resultado de prueba

Por último, se verifica que la creación de pedidos funcione debidamente, se hace una petición a la ruta respectiva como se muestra en la Fig. 42, y se observan los resultados en la Fig. 43.

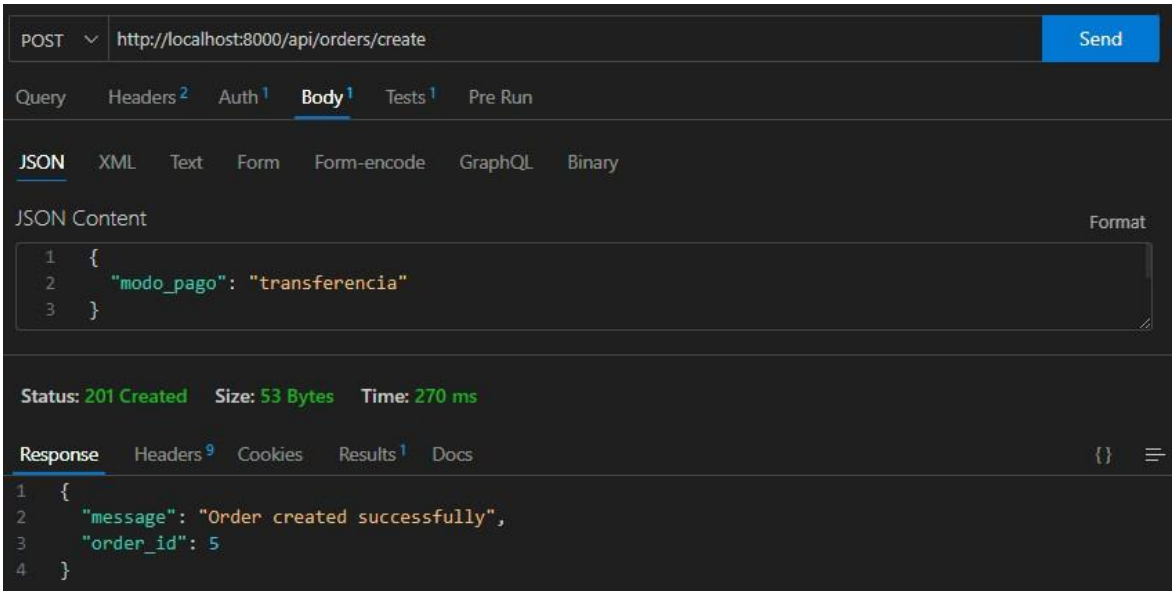


Fig. 42: Cuerpo de petición

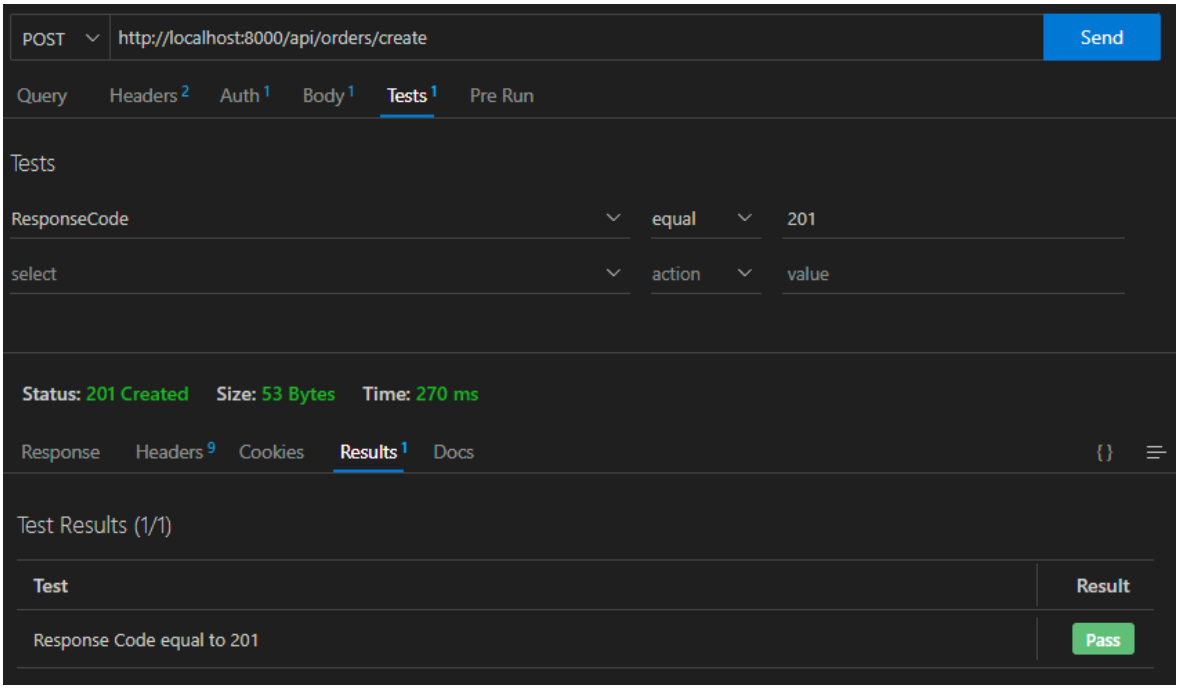


Fig. 43: Resultados de prueba

Como el usuario ya ha creado su pedido, su carrito de compras ha sido eliminado, por lo tanto, ya no puede crear otro pedido hasta que agregue algún producto a su nuevo carrito. Debido a esto, si se llama de nuevo a la misma ruta con el mismo cuerpo desde la misma sesión de usuario antes que esto ocurra, el código de respuesta debe ser no exitoso como lo muestra la Fig. 44 y la Fig. 45.

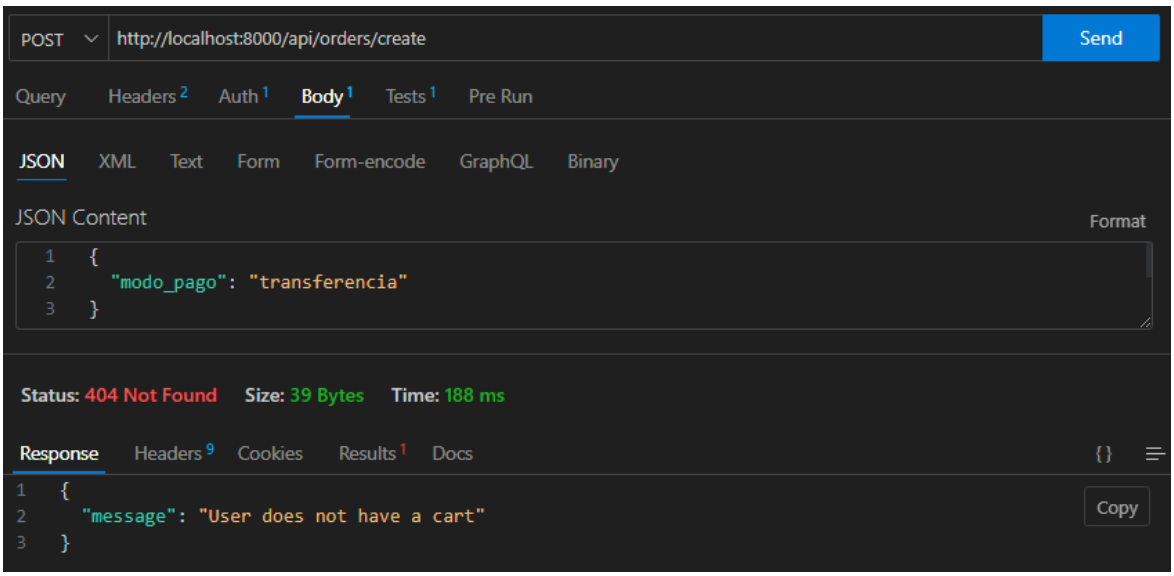


Fig. 44: Cuerpo de petición

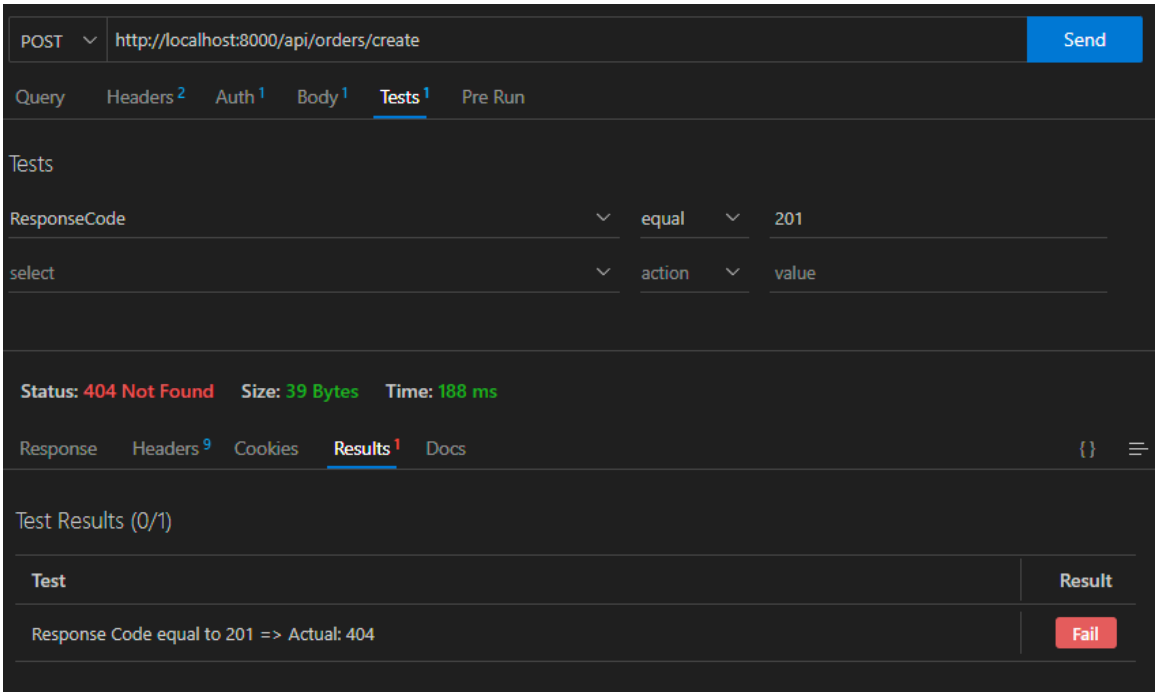


Fig. 45: Resultados de prueba

Realizar pruebas de carga

Por medio del uso de la herramienta JMeter, se realizaron pruebas de carga a varios módulos, con el fin de medir el tiempo de respuesta para 50 usuarios concurrentes como lo muestra la Fig. 46.

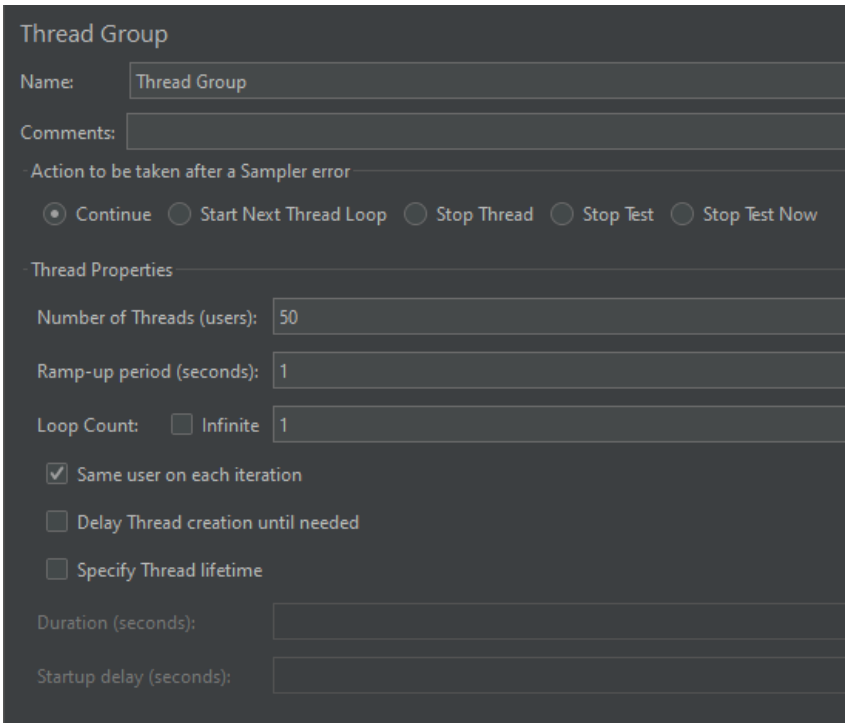
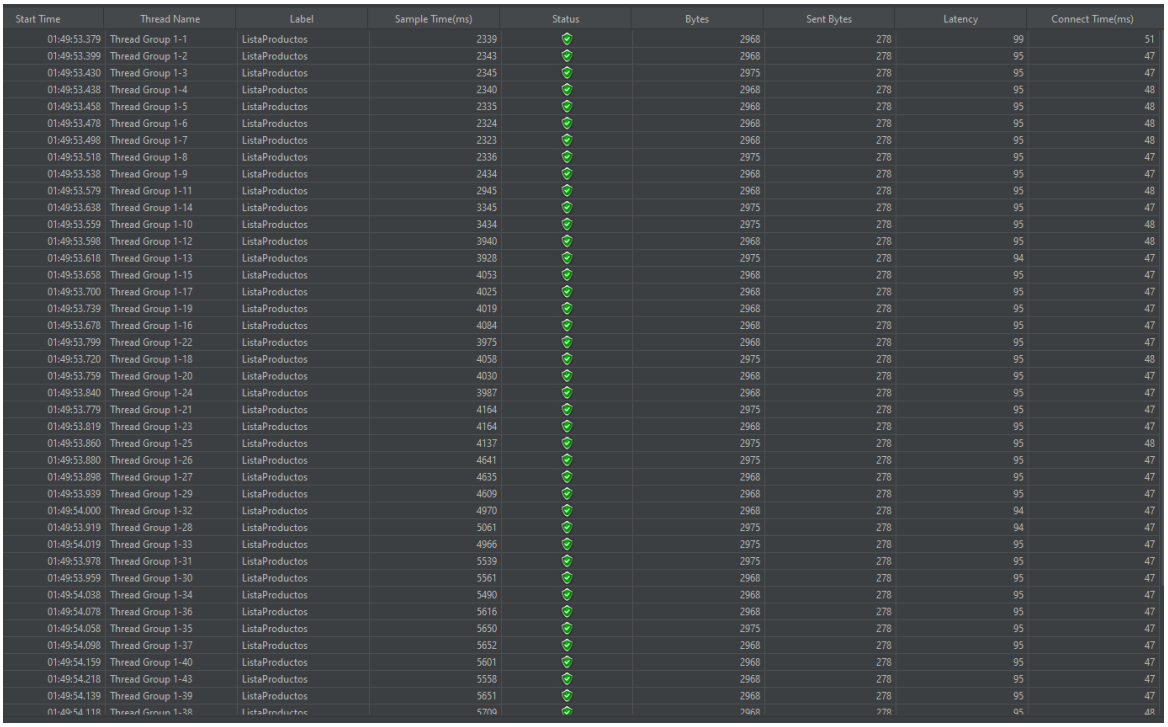


Fig. 46: Configuraciones Jmeter

Una vez ejecutado el test, se observa que los resultados son favorables ya que no existen errores al momento de hacer peticiones a la ruta que retorna la lista completa de productos como lo muestra la Fig. 47. Los tiempos de respuesta son aceptables, aunque un poco altos; esto puede ser dado que la base de datos está ubicada en un servicio de alojamiento externo al sitio donde se encuentra alojado el *backend* en sí.



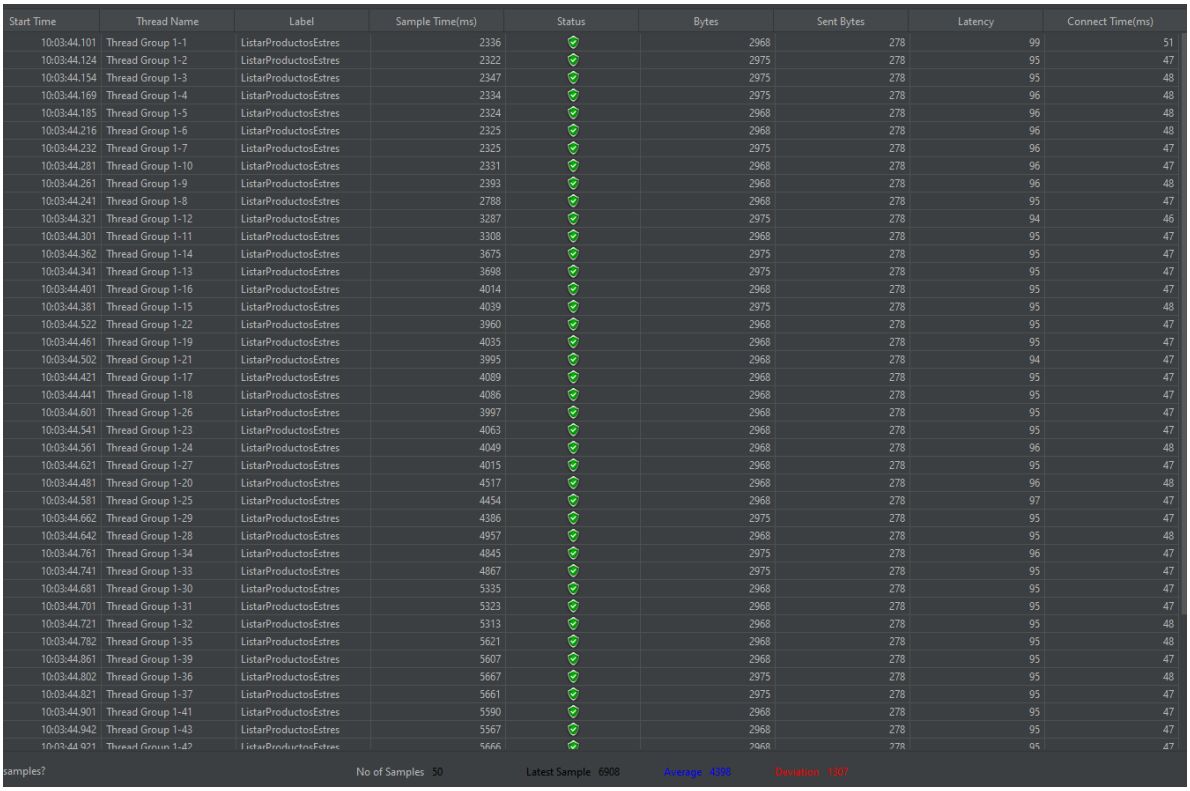
Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
01:49:53.379	Thread Group 1-1	ListaProductos	2339		2968	278	99	51
01:49:53.399	Thread Group 1-2	ListaProductos	2343		2968	278	95	47
01:49:53.430	Thread Group 1-3	ListaProductos	2345		2975	278	95	47
01:49:53.438	Thread Group 1-4	ListaProductos	2340		2968	278	95	48
01:49:53.458	Thread Group 1-5	ListaProductos	2335		2968	278	95	48
01:49:53.478	Thread Group 1-6	ListaProductos	2324		2968	278	95	48
01:49:53.498	Thread Group 1-7	ListaProductos	2323		2968	278	95	48
01:49:53.518	Thread Group 1-8	ListaProductos	2336		2975	278	95	47
01:49:53.538	Thread Group 1-9	ListaProductos	2434		2968	278	95	47
01:49:53.579	Thread Group 1-11	ListaProductos	2945		2968	278	95	48
01:49:53.638	Thread Group 1-14	ListaProductos	3345		2975	278	95	47
01:49:53.559	Thread Group 1-10	ListaProductos	3434		2975	278	95	48
01:49:53.598	Thread Group 1-12	ListaProductos	3940		2968	278	95	48
01:49:53.618	Thread Group 1-13	ListaProductos	3928		2975	278	94	47
01:49:53.658	Thread Group 1-15	ListaProductos	4053		2968	278	95	47
01:49:53.700	Thread Group 1-17	ListaProductos	4025		2968	278	95	47
01:49:53.739	Thread Group 1-19	ListaProductos	4019		2968	278	95	47
01:49:53.678	Thread Group 1-16	ListaProductos	4084		2968	278	95	47
01:49:53.799	Thread Group 1-22	ListaProductos	3975		2968	278	95	47
01:49:53.720	Thread Group 1-18	ListaProductos	4058		2975	278	95	48
01:49:53.759	Thread Group 1-20	ListaProductos	4030		2968	278	95	47
01:49:53.840	Thread Group 1-24	ListaProductos	3987		2968	278	95	47
01:49:53.779	Thread Group 1-21	ListaProductos	4164		2975	278	95	47
01:49:53.819	Thread Group 1-23	ListaProductos	4164		2968	278	95	47
01:49:53.860	Thread Group 1-25	ListaProductos	4137		2975	278	95	48
01:49:53.880	Thread Group 1-26	ListaProductos	4641		2975	278	95	47
01:49:53.898	Thread Group 1-27	ListaProductos	4635		2968	278	95	47
01:49:53.939	Thread Group 1-29	ListaProductos	4609		2968	278	95	47
01:49:54.000	Thread Group 1-32	ListaProductos	4970		2968	278	94	47
01:49:53.919	Thread Group 1-28	ListaProductos	5061		2975	278	94	47
01:49:54.019	Thread Group 1-33	ListaProductos	4966		2975	278	95	47
01:49:53.978	Thread Group 1-31	ListaProductos	5539		2975	278	95	47
01:49:53.959	Thread Group 1-30	ListaProductos	5561		2968	278	95	47
01:49:54.038	Thread Group 1-34	ListaProductos	5490		2968	278	95	47
01:49:54.078	Thread Group 1-36	ListaProductos	5616		2968	278	95	47
01:49:54.058	Thread Group 1-35	ListaProductos	5650		2975	278	95	47
01:49:54.098	Thread Group 1-37	ListaProductos	5652		2968	278	95	47
01:49:54.159	Thread Group 1-40	ListaProductos	5601		2968	278	95	47
01:49:54.218	Thread Group 1-43	ListaProductos	5558		2968	278	95	47
01:49:54.139	Thread Group 1-39	ListaProductos	5651		2968	278	95	47
01:49:54.118	Thread Group 1-38	ListaProductos	5709		2968	278	95	48

Fig. 47: Resultados test de carga

Las demás pruebas de carga realizadas se encuentran dentro del ANEXO II.

Realizar pruebas de estrés

Para estas pruebas se realizaron 50 peticiones al *endpoint* que retorna la lista de productos completa dentro del lapso de 1 segundo. Los resultados son positivos dado que el *backend* pudo soportar la carga sin ningún error y con un tiempo promedio de respuesta de 4398ms como lo indica la Fig. 48.

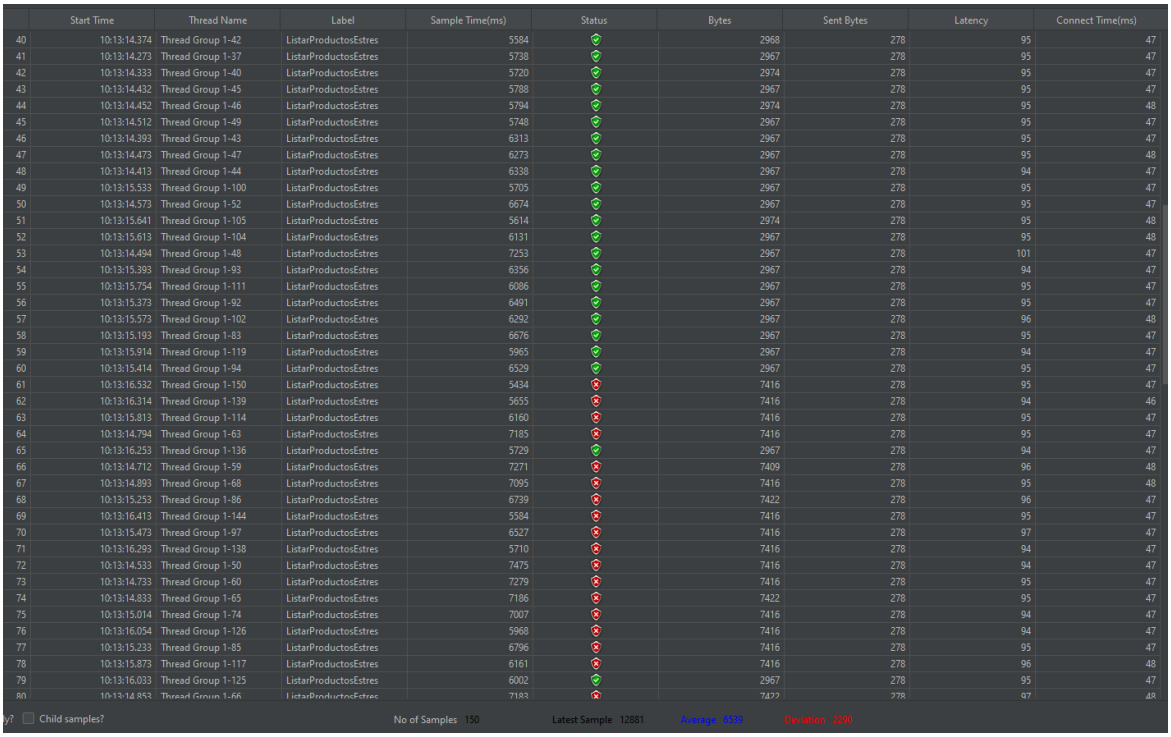


Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
10:03:44.101	Thread Group 1-1	ListarProductosEstres	2336		2968	278	99	51
10:03:44.124	Thread Group 1-2	ListarProductosEstres	2322		2975	278	95	47
10:03:44.154	Thread Group 1-3	ListarProductosEstres	2347		2975	278	95	48
10:03:44.169	Thread Group 1-4	ListarProductosEstres	2334		2975	278	96	48
10:03:44.185	Thread Group 1-5	ListarProductosEstres	2324		2968	278	96	48
10:03:44.216	Thread Group 1-6	ListarProductosEstres	2325		2968	278	96	48
10:03:44.232	Thread Group 1-7	ListarProductosEstres	2325		2975	278	96	47
10:03:44.281	Thread Group 1-10	ListarProductosEstres	2331		2968	278	96	47
10:03:44.261	Thread Group 1-9	ListarProductosEstres	2393		2968	278	96	48
10:03:44.241	Thread Group 1-8	ListarProductosEstres	2788		2968	278	95	47
10:03:44.321	Thread Group 1-12	ListarProductosEstres	3287		2975	278	94	46
10:03:44.301	Thread Group 1-11	ListarProductosEstres	3308		2968	278	95	47
10:03:44.362	Thread Group 1-14	ListarProductosEstres	3675		2975	278	95	47
10:03:44.341	Thread Group 1-13	ListarProductosEstres	3698		2975	278	95	47
10:03:44.401	Thread Group 1-16	ListarProductosEstres	4014		2968	278	95	47
10:03:44.381	Thread Group 1-15	ListarProductosEstres	4039		2975	278	95	48
10:03:44.522	Thread Group 1-22	ListarProductosEstres	3960		2968	278	95	47
10:03:44.461	Thread Group 1-19	ListarProductosEstres	4035		2968	278	95	47
10:03:44.502	Thread Group 1-21	ListarProductosEstres	3995		2968	278	94	47
10:03:44.421	Thread Group 1-17	ListarProductosEstres	4089		2968	278	95	47
10:03:44.441	Thread Group 1-18	ListarProductosEstres	4086		2968	278	95	47
10:03:44.601	Thread Group 1-26	ListarProductosEstres	3997		2968	278	95	47
10:03:44.541	Thread Group 1-23	ListarProductosEstres	4063		2968	278	95	47
10:03:44.561	Thread Group 1-24	ListarProductosEstres	4049		2968	278	96	48
10:03:44.621	Thread Group 1-27	ListarProductosEstres	4015		2968	278	95	47
10:03:44.481	Thread Group 1-20	ListarProductosEstres	4517		2968	278	96	48
10:03:44.581	Thread Group 1-25	ListarProductosEstres	4454		2968	278	97	47
10:03:44.662	Thread Group 1-29	ListarProductosEstres	4386		2975	278	95	47
10:03:44.642	Thread Group 1-28	ListarProductosEstres	4957		2968	278	95	48
10:03:44.761	Thread Group 1-34	ListarProductosEstres	4845		2975	278	96	47
10:03:44.741	Thread Group 1-33	ListarProductosEstres	4867		2975	278	95	47
10:03:44.681	Thread Group 1-30	ListarProductosEstres	5335		2968	278	95	47
10:03:44.701	Thread Group 1-31	ListarProductosEstres	5323		2968	278	95	47
10:03:44.721	Thread Group 1-32	ListarProductosEstres	5313		2968	278	95	48
10:03:44.782	Thread Group 1-35	ListarProductosEstres	5621		2968	278	95	48
10:03:44.861	Thread Group 1-39	ListarProductosEstres	5607		2968	278	95	47
10:03:44.802	Thread Group 1-36	ListarProductosEstres	5667		2975	278	95	48
10:03:44.821	Thread Group 1-37	ListarProductosEstres	5661		2975	278	95	47
10:03:44.901	Thread Group 1-41	ListarProductosEstres	5590		2968	278	95	47
10:03:44.942	Thread Group 1-43	ListarProductosEstres	5567		2968	278	95	47
10:03:44.921	Thread Group 1-42	ListarProductosEstres	5666		2968	278	95	47

samples? No of Samples 50 Latest Sample 6908 Average 4398 Deviation 1307

Fig. 48: Resultados prueba de estrés

Dado que el objetivo de las pruebas es encontrar el punto de quiebre del *backend*, se aumenta el número de peticiones a 150 en un lapso de 3 segundos al mismo *endpoint* utilizado anteriormente. Los resultados comienzan a indicar errores al momento de ejecutar la petición número 61, la cual retorna un código 429 que quiere decir que se han realizado demasiadas peticiones como lo indica la Fig. 49.



	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
40	10:13:14.374	Thread Group 1-42	ListarProductosEstres	5584		2968	278	95	47
41	10:13:14.273	Thread Group 1-37	ListarProductosEstres	5738		2967	278	95	47
42	10:13:14.333	Thread Group 1-40	ListarProductosEstres	5720		2974	278	95	47
43	10:13:14.432	Thread Group 1-45	ListarProductosEstres	5788		2967	278	95	47
44	10:13:14.452	Thread Group 1-46	ListarProductosEstres	5794		2974	278	95	48
45	10:13:14.512	Thread Group 1-49	ListarProductosEstres	5748		2967	278	95	47
46	10:13:14.393	Thread Group 1-43	ListarProductosEstres	6313		2967	278	95	47
47	10:13:14.473	Thread Group 1-47	ListarProductosEstres	6273		2967	278	95	48
48	10:13:14.413	Thread Group 1-44	ListarProductosEstres	6338		2967	278	94	47
49	10:13:15.533	Thread Group 1-100	ListarProductosEstres	5705		2967	278	95	47
50	10:13:14.573	Thread Group 1-52	ListarProductosEstres	6674		2967	278	95	47
51	10:13:15.641	Thread Group 1-105	ListarProductosEstres	5614		2974	278	95	48
52	10:13:15.613	Thread Group 1-104	ListarProductosEstres	6131		2967	278	95	48
53	10:13:14.494	Thread Group 1-48	ListarProductosEstres	7253		2967	278	101	47
54	10:13:15.393	Thread Group 1-93	ListarProductosEstres	6356		2967	278	94	47
55	10:13:15.754	Thread Group 1-111	ListarProductosEstres	6086		2967	278	95	47
56	10:13:15.373	Thread Group 1-92	ListarProductosEstres	6491		2967	278	95	47
57	10:13:15.573	Thread Group 1-102	ListarProductosEstres	6292		2967	278	96	48
58	10:13:15.193	Thread Group 1-83	ListarProductosEstres	6676		2967	278	95	47
59	10:13:15.914	Thread Group 1-119	ListarProductosEstres	5965		2967	278	94	47
60	10:13:15.414	Thread Group 1-94	ListarProductosEstres	6529		2967	278	95	47
61	10:13:16.532	Thread Group 1-150	ListarProductosEstres	5434		7416	278	95	47
62	10:13:16.314	Thread Group 1-139	ListarProductosEstres	5655		7416	278	94	46
63	10:13:15.813	Thread Group 1-114	ListarProductosEstres	6160		7416	278	95	47
64	10:13:14.794	Thread Group 1-63	ListarProductosEstres	7185		7416	278	95	47
65	10:13:16.253	Thread Group 1-136	ListarProductosEstres	5729		2967	278	94	47
66	10:13:14.712	Thread Group 1-59	ListarProductosEstres	7271		7409	278	96	48
67	10:13:14.893	Thread Group 1-68	ListarProductosEstres	7095		7416	278	95	48
68	10:13:15.253	Thread Group 1-86	ListarProductosEstres	6739		7422	278	96	47
69	10:13:16.413	Thread Group 1-144	ListarProductosEstres	5584		7416	278	95	47
70	10:13:15.473	Thread Group 1-97	ListarProductosEstres	6527		7416	278	97	47
71	10:13:16.293	Thread Group 1-138	ListarProductosEstres	5710		7416	278	94	47
72	10:13:14.533	Thread Group 1-50	ListarProductosEstres	7475		7416	278	94	47
73	10:13:14.733	Thread Group 1-60	ListarProductosEstres	7279		7416	278	95	47
74	10:13:14.833	Thread Group 1-65	ListarProductosEstres	7186		7422	278	95	47
75	10:13:15.014	Thread Group 1-74	ListarProductosEstres	7007		7416	278	94	47
76	10:13:16.054	Thread Group 1-126	ListarProductosEstres	5968		7416	278	94	47
77	10:13:15.233	Thread Group 1-85	ListarProductosEstres	6796		7416	278	95	47
78	10:13:15.873	Thread Group 1-117	ListarProductosEstres	6161		7416	278	96	48
79	10:13:16.033	Thread Group 1-125	ListarProductosEstres	6002		2967	278	95	47
80	10:13:14.853	Thread Group 1-66	ListarProductosEstres	7183		7422	278	97	48

ly? Child samples? No of Samples 150 Latest Sample 12881 Average 6339 Deviation 2290

Fig. 49: Resultados prueba de estrés

Las demás pruebas de estrés realizadas se encuentran dentro del ANEXO II.

Desplegar

El despliegue de este proyecto es realizado en la plataforma Fly.io; el primer paso es instalar flyctl, el cual es el centro de comandos requerido, por medio del comando indicado en la Fig. 50.

```
$ pwsh -Command "iwr https://fly.io/install.ps1 -useb | iex"
```

Fig. 50: Comando para la instalación de flyctl

El siguiente paso es crear los archivos que Fly requiere dentro del directorio del proyecto, esto se logra por medio del comando indicado en la Fig. 51.

```
$ fly launch
```

Fig. 51: Comando para crear archivos de Fly

Una vez creados los archivos, se procede a ingresar las variables de entorno necesarias, como las de la base de datos y la API *KEY* de Cloudinary, esto se hace por medio del comando indicado en la Fig. 52.

```
$ fly secrets set SOME_SECRET_KEY=<the-value-from-your-env-file>
```

Fig. 52: Comando para ingresar variables de entorno

Por último, se corre el comando indicado en la Fig. 53, el cual realiza el despliegue final.

```
$ fly deploy
```

Fig. 53: Comando para realizar el despliegue final

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Al identificar y analizar los requerimientos en las etapas iniciales del proyecto, se obtuvo una visión clara de la metodología adecuada y se adquirió una comprensión de la magnitud y alcance del mismo.
- Se ha logrado diseñar una base de datos que cumple con todos los requisitos organizacionales del proyecto, ofreciendo una experiencia fluida como parte importante del patrón arquitectónico aplicado.
- La utilización de Laravel ha simplificado la creación de *endpoints*, gracias a sus herramientas como Eloquent y Sanctum, las cuales han reducido significativamente la complejidad del desarrollo del proyecto. Además, la escalabilidad que Laravel proporciona al proyecto es bastante prometedora.
- Se han obtenido resultados positivos de las pruebas realizadas a los *endpoints*, los tiempos de respuesta y el rendimiento en general han resultado favorecedores, aunque existe cierto tiempo de espera dada la ubicación geográfica de los servidores de la base de datos.
- El despliegue queda realizado exitosamente, la plataforma Fly.io ha provisto al proyecto de suficientes recursos para su correcto funcionamiento. La *API* queda lista para ser utilizada por el *frontend*.

4.2 Recomendaciones

- Es recomendable siempre verificar los permisos asignados a cada rol en relación con los diferentes *endpoints* de la aplicación. Además, es importante restringir el acceso a ciertos *endpoints* por parte de terceros que lleguen a obtener las *URLs*.
- Es importante escoger la metodología de desarrollo adecuada para cada proyecto y estar siempre atento a las necesidades del cliente para poder realizar ajustes o cambios oportunos.
- Se recomienda escribir código que sea escalable para facilitar nuevas implementaciones y optimizaciones, y así el proyecto pueda crecer.
- Se podría aumentar un módulo de facturación digital, dependiendo de la necesidad del cliente.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] Peña-López, I. «La revolución del comercio electrónico,» 2014

[2] A. Fonseca, «Fundamentos del eCommerce,» 2014. [En línea]. Available: https://www.scribd.com/book/216428537/Fundamentos-del-eCommerce. [Último acceso: 26 Mayo 2023].

[3] F. Asorey, «Gestión de pedidos en la tienda online,» 15 Febrero 2014. [En línea]. Available: https://www.federicoasorey.es/2014/02/gestion-de-pedidos-en-la-tienda-on-line/ [Último acceso: 26 Mayo 2023].

[4] Equipo Redacción apser, «Ventajas de un sistema de gestión de pedidos para una empresa,» 31 Agosto 2015. [En línea]. Available: https://apser.es/ventajas-de-un-sistema-de-gestion-de-pedidos-para-una-empresa/ [Último acceso: 26 Mayo 2023].

[5] Agencia Marketing y Desarrollo Web Nestrategia, «¿Qué es Back End, Front End y Back Office y por qué es importante para tu web?,» [En línea]. Available: https://nestrategia.com/desarrollo-web-back-end-front-end/ [Último acceso: 26 Mayo 2023].

[6] F. Cristancho, «¿Qué es un framework en programación?,» 08 Febrero 2022. [En línea]. Available: https://talently.tech/blog/que-es-un-framework-en-programacion/ [Último acceso: 26 Mayo 2023].

[7] T. Otwell, «Laravel - The PHP Framework For Web Artisans,». [En línea]. Available en: https://laravel.com/. [Último acceso: 26 Mayo 2023].

[8] Revista UNIR, «¿Qué es un IDE en programación?,» 06 Julio 2021. [En línea]. Available: https://www.unir.net/ingenieria/revista/ide-programacion. [Último acceso: 26 Mayo 2023].

[9] Maida, EG, Pacienzia, J., «Metodologías de desarrollo de software,» 2015. [En línea]. Available: https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf. [Último acceso: 27 Mayo 2023].

[10] D. da Silva, «¿Qué es scrum, cuál es su finalidad y sus principales ventajas?,» 31 Mayo 2021. [En línea]. Available: https://www.zendesk.com.mx/blog/que-es-scrum/. [Último acceso: 27 Mayo 2023].

[11] S. Monroy. «¿Cuáles son los roles de la metodología Scrum?,» 14 Diciembre 2021. [En línea]. Available: https://www.apd.es/roles-metodologia-scrum/. [Último acceso: 27 Mayo 2023].

[12] Kaizenia. «¿Qué son los Artefactos de Scrum?,» 04 Marzo 2020. [En línea]. Available: https://kzi.mx/que-son-los-artefactos-de-scrum/. [Último acceso: 27 Mayo 2023].

[13] Team Asana. «Historias de usuario: 3 ejemplos para generar valor para el usuario,» 21 Enero 2022. [En línea]. Available: https://asana.com/es/resources/user-stories. [Último acceso: 27 Mayo 2023].

[14] M. A. Álvarez. «Qué es MVC,» 28 Julio 2020. [En línea]. Available: https://desarrolloweb.com/articulos/que-es-mvc.html. [Último acceso: 28 Mayo 2023].

[15] Yair. «Qué es Composer y cómo usarlo,» 23 Diciembre 2019. [En línea]. Available: https://styde.net/que-es-composer-y-como-usarlo/. [Último acceso: 28 Mayo 2023].

[16] A. Robledano. «Qué es MySQL: Características y ventajas,» 24 Septiembre 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-mysql/. [Último acceso: 28 Mayo 2023].
[17] F. Flores. «Qué es Visual Studio Code y qué ventajas ofrece,» 22 Julio 2022. [En línea]. Available: https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/. [Último acceso: 28 Mayo 2023].
[18] Jesús. «Conoce qué es Xampp y por qué deberías usarlo en tus proyectos,» 25 Abril 2022. [En línea]. Available: https://www.dongee.com/tutoriales/que-es-xampp/. [Último acceso: 28 Mayo 2023].
[19] A. Fernández. «OPTIMIZA LA CARGA DE IMÁGENES DE TU APLICACIÓN WEB, APP O PÁGINA CON CLOUDINARY,» 09 Junio 2016. [En línea]. Available: https://antoniofernandez.com/optimizar-carga-de-imagenes-cloudinary/. [Último acceso: 28 Mayo 2023].
[20] R. Vadhineni. «Thunder Client — lightweight alternative to Postman,» 30 Marzo 2021. [En línea]. Available: https://rangav.medium.com/thunder-client-alternative-to-postman-68ee0c9486d6/. [Último acceso: 28 Mayo 2023].

6 ANEXOS

En la siguiente sección se presentan los ANEXOS relacionados con el desarrollo e implementación de este proyecto:

- ANEXO I. Certificado Turnitin
- ANEXO II. Manual Técnico
- ANEXO III. Manual de Usuario
- ANEXO IV. Manual de Instalación

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

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 28 de agosto de 2023

De mi consideración:

Yo, Juan Pablo Zaldumbide Proaño, en calidad de Director del Trabajo de Integración Curricular titulado Desarrollo de un backend asociado al DESARROLLO DE SISTEMA WEB DE GESTIÓN DE PEDIDOS EN MINIMARKETS elaborado por el estudiante PAEZ CORDERO PEDRO PABLO de la carrera en TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE , certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 10%.

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

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

Atentamente,

Firmado electrónicamente por:
JUAN PABLO ZALDUMBIDE PROANO

Juan Pablo Zaldumbide Proaño
Docente Ocasional a Tiempo Completo
ESFOT

6.1 ANEXO II Manual Técnico

Recopilación de Requerimientos

En la TABLA VI se presenta la recopilación de requerimientos del proyecto.

Tabla VI Recopilación de requerimientos

RECOPILACIÓN DE REQUERIMIENTOS	
ID - RR	**ENUNCIADO DEL ITEM**
RR01	Como usuarios superadministrador, administrador, usuario y cliente; se necesita un módulo para: • Inicio y cierre de sesión
RR02	Como usuario cliente necesito un módulo para: • Registrarse en el sistema
RR03	Como usuario superadministrador necesito: • Gestionar cuentas de administrador
RR04	Como usuario administrador necesito: • Gestionar cuentas de empleados
RR05	Como usuario administrador necesito: • Gestionar productos
RR06	Como usuario administrador y empleado, se necesita: • Ver historial de pedidos
RR07	Como usuario administrador y empleado, se necesita: • Ver lista de pedidos activos
RR08	Como usuario administrador y empleado, se necesita: • Cambiar estado de pedidos
RR09	Como usuario administrador necesito: • Ver información y gestionar inventario
RR10	Como usuario cliente necesito: • Gestionar información personal
RR11	Como usuario cliente necesito: • Ver historial de mis pedidos
RR12	Como usuario cliente necesito: • Navegar por el catálogo de productos
RR13	Como usuario cliente necesito: • Buscar productos del catálogo

RR14	Como usuario cliente necesito: • Añadir o remover productos del carrito de compras
RR15	Como usuario cliente necesito: • Crear un pedido
RR16	Como usuario cliente necesito: • Revisar el estado de mi pedido

Historias de Usuario

En las siguientes tablas se puede observar las historias de usuario correspondientes a este backend.

Tabla VII Historia de Usuario HU001

HISTORIA DE USUARIO	
Identificador (ID): HU001	**Usuario:** Superadministrador, administrador, empleado, cliente
Nombre Historia: Iniciar sesión	
Prioridad: Media	**Riesgo en Desarrollo:** Medio
Iteración: 1	
Responsable: Pedro Páez	
Descripción: El backend debe permitir el inicio y cierre de sesión por medio de un endpoint	
Observación: El usuario debe estar registrado para poder iniciar sesión	

Tabla VIII Historia de Usuario HU002

HISTORIA DE USUARIO	
Identificador (ID): HU002	**Usuario:** Cliente
Nombre Historia: Registro en el sistema	
Prioridad: Media	**Riesgo en Desarrollo:** Medio
Iteración: 1	
Responsable: Pedro Páez	
Descripción: El backend debe permitir el registro de un cliente al sistema	
Observación: El usuario registrado solamente contará con el rol de cliente	

Tabla IX Historia de Usuario HU003

HISTORIA DE USUARIO	
Identificador (ID): HU003	**Usuario:** Superadministrador
Nombre Historia: Gestionar cuentas administrador	
Prioridad: Media	**Riesgo en Desarrollo:** Medio
Iteración: 1	
Responsable: Pedro Páez	
Descripción: El backend debe permitir la creación y eliminación de cuentas de administrador	
Observación: El usuario puede crear o eliminar cualquier tipo de cuenta excepto de superadmin	

Tabla X Historia de Usuario HU004

HISTORIA DE USUARIO	
Identificador (ID): HU004	**Usuario:** Administrador
Nombre Historia: Gestionar cuentas empleados	
Prioridad: Media	**Riesgo en Desarrollo:** Medio
Iteración: 1	
Responsable: Pedro Páez	
Descripción: El backend debe permitir la creación y eliminación de cuentas de empleados	
Observación: El usuario únicamente puede crear o eliminar cuentas de empleado	

Tabla XI Historia de Usuario HU005

HISTORIA DE USUARIO	
Identificador (ID): HU005	**Usuario:** Administrador
Nombre Historia: Gestionar productos	
Prioridad: Media	**Riesgo en Desarrollo:** Medio
Iteración: 2	
Responsable: Pedro Páez	
Descripción: El backend debe permitir la inserción, modificación y eliminación de productos al catálogo, también se deben almacenar las imágenes de los productos en la nube	

Observación: Los productos únicamente pueden pertenecer a las categorías de alimentos, productos de aseo personal y productos electrónicos

Tabla XII Historia de Usuario HU006

HISTORIA DE USUARIO	
Identificador (ID): HU006	**Usuario:** Cliente
Nombre Historia: Navegación por catálogo	
Prioridad: Media	**Riesgo en Desarrollo:** Bajo
Iteración: 2	
Responsable: Pedro Páez	
Descripción: El backend debe permitir al usuario navegar por el catálogo de productos y también filtrar productos	
Observación: La búsqueda de productos es en tiempo real	

Tabla XIII Historia de Usuario HU007

HISTORIA DE USUARIO	
Identificador (ID): HU007	**Usuario:** Cliente
Nombre Historia: Carrito de compras	
Prioridad: Alta	**Riesgo en Desarrollo:** Medio
Iteración: 2	
Responsable: Pedro Páez	
Descripción: El backend debe permitir al usuario agregar o eliminar productos de un carrito de compras	
Observación: El carrito de compras es permanente, es decir hasta que el cliente decida modificarlo el carrito siempre va a estar disponible	

Tabla XIV Historia de Usuario HU008

HISTORIA DE USUARIO	
Identificador (ID): HU008	**Usuario:** Administrador
Nombre Historia: Gestión de inventario	
Prioridad: Alta	**Riesgo en Desarrollo:** Alta
Iteración: 3	
Responsable: Pedro Páez	

Descripción: El backend debe permitir al usuario ver la cantidad de inventario disponible y también modificar dicha cantidad
Observación:

Tabla XV Historia de Usuario HU009

HISTORIA DE USUARIO	
Identificador (ID): HU009	**Usuario:** Cliente
Nombre Historia: Gestión de perfil	
Prioridad: Baja	**Riesgo en Desarrollo:** Bajo
Iteración: 3	
Responsable: Pedro Páez	
Descripción: El backend debe permitir al usuario insertar, modificar o eliminar información personal de su perfil de usuario	
Observación: La información que el usuario puede gestionar es: nombres, apellidos, teléfono y direcciones	

Tabla XVI Historia de Usuario HU010

HISTORIA DE USUARIO	
Identificador (ID): HU010	**Usuario:** Administrador, empleado, cliente
Nombre Historia: Gestión de pedidos	
Prioridad: Media	**Riesgo en Desarrollo:** Bajo
Iteración: 3	
Responsable: Pedro Páez	
Descripción: El backend debe permitir al cliente crear un pedido y al administrador o empleado eliminar un pedido	
Observación:	

Tabla XVII Historia de Usuario HU011

HISTORIA DE USUARIO	
Identificador (ID): HU011	**Usuario:** Administrador, empleado
Nombre Historia: Historial de pedidos	
Prioridad: Media	**Riesgo en Desarrollo:** Bajo
Iteración: 4	
Responsable: Pedro Páez	

Descripción: El backend debe permitir al usuario ver el historial de los pedidos generados hasta el momento que ya hayan sido entregados y ver la lista de los pedidos generados hasta el momento que no hayan sido entregados
Observación: Los pedidos pueden estar en estado pendiente o entregado

Tabla XVIII Historia de Usuario HU012

HISTORIA DE USUARIO	
Identificador (ID): HU012	**Usuario:** Administrador, cliente
Nombre Historia: Estado de pedidos	
Prioridad: Media	**Riesgo en Desarrollo:** Bajo
Iteración: 4	
Responsable: Pedro Páez	
Descripción: El backend debe permitir al usuario cambiar el estado de un pedido	
Observación: Los pedidos pueden estar en estado pendiente o entregado	

Product Backlog

La TABLA XIX se muestra las prioridades de cada requerimiento basado en las necesidades del Product Owner.

Tabla XIX Product Backlog

PRODUCT BACKLOG				
ID - HU	**HISTORIA DE USUARIO**	**SPRINT**	**ESTADO**	**PRIORIDAD**
HU001	Iniciar sesión	1	Finalizado	Media
HU002	Registro en el sistema	1	Finalizado	Media
HU003	Gestionar cuentas administrador	1	Finalizado	Media
HU004	Gestionar cuentas empleados	1	Finalizado	Media
HU005	Gestionar productos	2	Finalizado	Media
HU006	Navegación por catálogo	2	Finalizado	Media
HU007	Carrito de compras	2	Finalizado	Alta
HU008	Gestión de inventario	3	Finalizado	Alta

HU009	Gestión de perfil	3	Finalizado	Baja
HU010	Gestión de pedidos	3	Finalizado	Media
HU011	Historial de pedidos	4	Finalizado	Media
HU012	Estado de pedidos	4	Finalizado	Alta

Sprint Backlog

En la TABLA XX se observa el sprint Backlog completo y detallado del proyecto.

Tabla XX Sprint Backlog

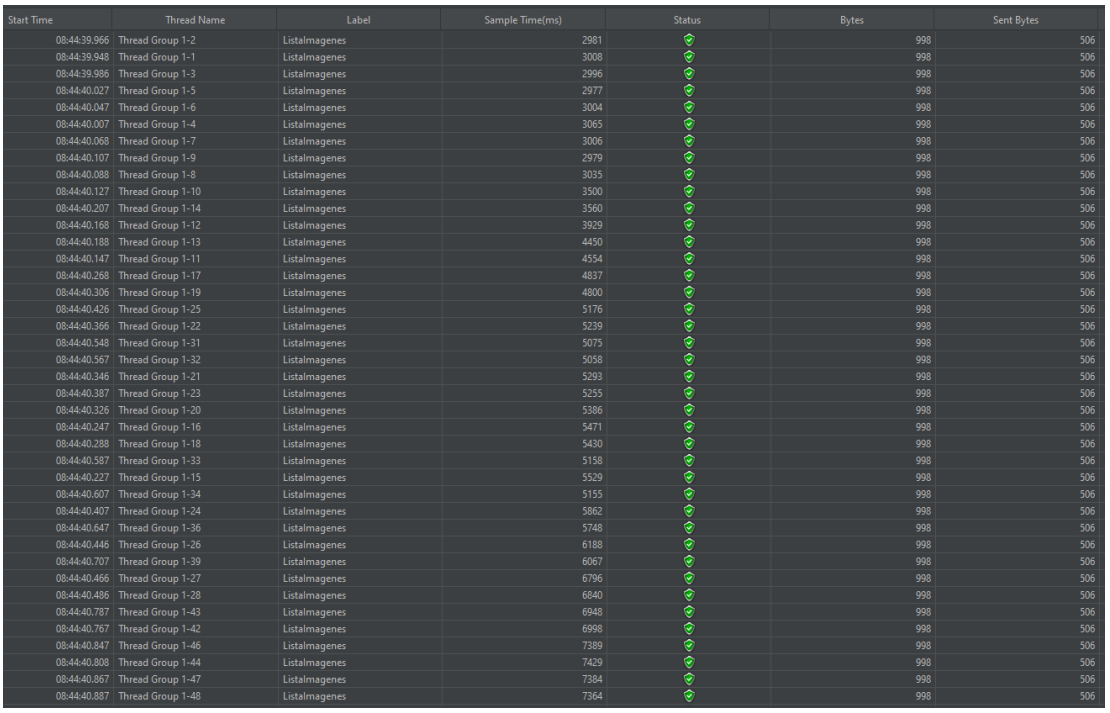
SPRINT BACKLOG						
ID - SB	**NOMBRE**	**MÓDULO**	**ID - HU**	**HISTORIA DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
SB000	Acondicionamiento del ambiente de trabajo				• Instalar herramientas para el desarrollo • Realizar modelo de base de datos	20H
SB001	Login, registro y gestión de cuentas	Inicio de sesión	HU001	Iniciar sesión	• Implementación de librería de Laravel para la autenticación de usuarios • Ingreso de un usuario superadmin en la base de datos de forma manual • Ingreso al sistema a través de endpoint	40H
		Creación de cuentas de clientes	HU002	Registro en el sistema	• Creación de cuenta de cliente por medio de datos recibidos a través de un *endpoint*	
		Creación y eliminación de cuentas de administrador	HU003	Gestionar cuentas administrador	• Implementación de rutas para la creación y eliminación de cuentas de administrador	

		Creación y eliminación de cuentas de empleados	HU004	Gestionar cuentas empleados	• Implementación de rutas para la creación y eliminación de cuentas de empleados	
SB002	Gestión de productos, navegación por catálogo y carrito de compras	Creación, modificación y eliminación de productos del catálogo	HU005	Gestionar productos	• Implementación de rutas para la creación, modificación y eliminación de productos del catálogo • Implementación de almacenamiento en la nube para imágenes	50H
		Navegar todos los productos del catálogo	HU006	Navegación por catálogo	• Implementación de ruta para mostrar lista de productos Implementación de método para filtrado de productos	
		Agregar, modificar o eliminar productos del carrito de compras	HU007	Carrito de compras	• Implementación de rutas para crear, mostrar y eliminar carrito de compras • Implementación de ruta para agregar y eliminar productos de un carrito • Implementación de método para calcular precio total de un carrito	
SB003	Gestión de pedidos, inventario	Ver y modificar cantidad de inventario	HU008	Gestión de inventario	• Implementación de rutas para modificar cantidad de inventario, además de visualizar el inventario de cada producto	40H

	y gestión de perfil de usuario	Agregar, modificar y eliminar información personal	HU009	Gestión de perfil	• Implementación de ruta para modificar datos personales del perfil del usuario	
		Creación y eliminación de pedidos	HU010	Gestión de pedidos	• Implementación de rutas para la creación y eliminación de pedidos	
SB004	Historial y estado de pedidos	Ver historial de pedidos	HU011	Historial de pedidos	• Implementación de ruta para mostrar el historial de pedidos de un cliente además de sus estados • Implementación de rutas para listar pedidos ya entregados o pendientes	50H
		Modificación de estado de pedidos	HU012	Estado de pedidos	• Implementación de ruta para modificar estado de pedido	
SB005	Pruebas y despliegue	• Realizar pruebas de integración • Realizar pruebas de carga • Realizar pruebas de estrés • Desplegar				40H
	Documentación					

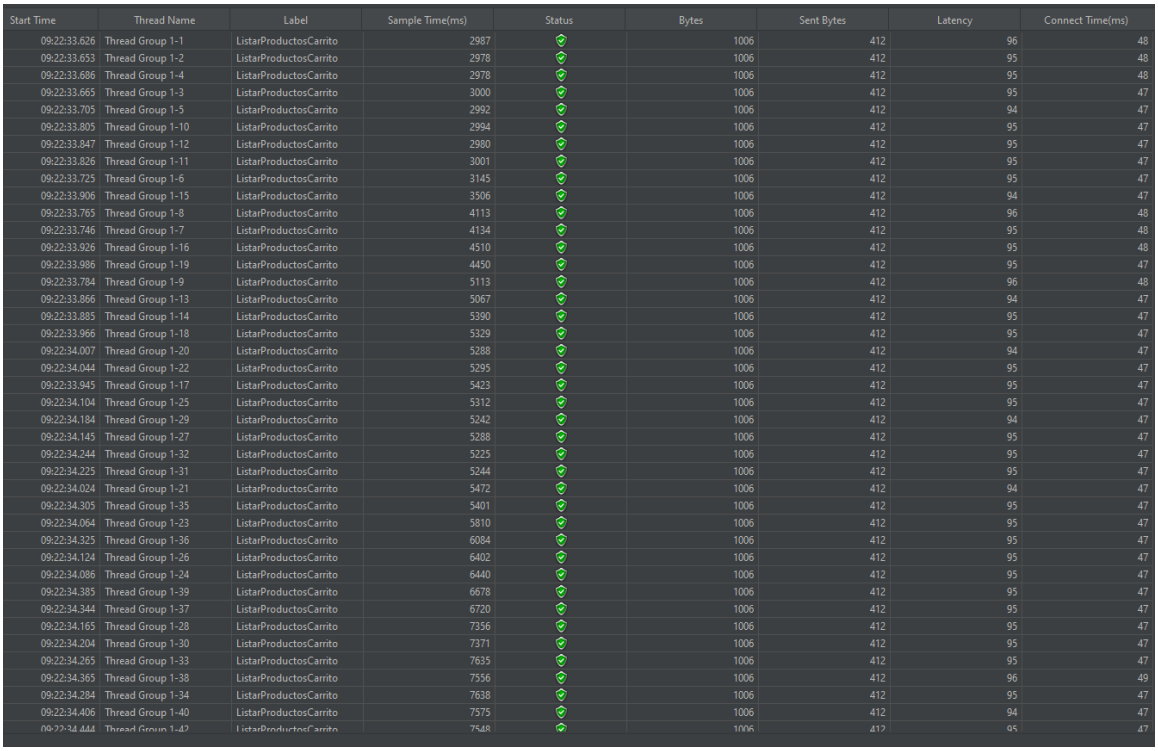
Pruebas de carga

A continuación, se observan los resultados de varias pruebas de carga realizadas a otros módulos del backend.



Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes
08:44:39.966	Thread Group 1-2	ListaImagenes	2981		998	506
08:44:39.948	Thread Group 1-1	ListaImagenes	3008		998	506
08:44:39.986	Thread Group 1-3	ListaImagenes	2996		998	506
08:44:40.027	Thread Group 1-5	ListaImagenes	2977		998	506
08:44:40.047	Thread Group 1-6	ListaImagenes	3004		998	506
08:44:40.007	Thread Group 1-4	ListaImagenes	3065		998	506
08:44:40.068	Thread Group 1-7	ListaImagenes	3006		998	506
08:44:40.107	Thread Group 1-9	ListaImagenes	2979		998	506
08:44:40.088	Thread Group 1-8	ListaImagenes	3035		998	506
08:44:40.127	Thread Group 1-10	ListaImagenes	3500		998	506
08:44:40.207	Thread Group 1-14	ListaImagenes	3560		998	506
08:44:40.168	Thread Group 1-12	ListaImagenes	3929		998	506
08:44:40.188	Thread Group 1-13	ListaImagenes	4450		998	506
08:44:40.147	Thread Group 1-11	ListaImagenes	4554		998	506
08:44:40.268	Thread Group 1-17	ListaImagenes	4837		998	506
08:44:40.306	Thread Group 1-19	ListaImagenes	4800		998	506
08:44:40.426	Thread Group 1-25	ListaImagenes	5176		998	506
08:44:40.366	Thread Group 1-22	ListaImagenes	5239		998	506
08:44:40.548	Thread Group 1-31	ListaImagenes	5075		998	506
08:44:40.567	Thread Group 1-32	ListaImagenes	5058		998	506
08:44:40.346	Thread Group 1-21	ListaImagenes	5293		998	506
08:44:40.387	Thread Group 1-23	ListaImagenes	5255		998	506
08:44:40.326	Thread Group 1-20	ListaImagenes	5386		998	506
08:44:40.247	Thread Group 1-16	ListaImagenes	5471		998	506
08:44:40.288	Thread Group 1-18	ListaImagenes	5430		998	506
08:44:40.587	Thread Group 1-33	ListaImagenes	5158		998	506
08:44:40.227	Thread Group 1-15	ListaImagenes	5529		998	506
08:44:40.607	Thread Group 1-34	ListaImagenes	5155		998	506
08:44:40.407	Thread Group 1-24	ListaImagenes	5862		998	506
08:44:40.647	Thread Group 1-36	ListaImagenes	5748		998	506
08:44:40.446	Thread Group 1-26	ListaImagenes	6188		998	506
08:44:40.707	Thread Group 1-39	ListaImagenes	6067		998	506
08:44:40.466	Thread Group 1-27	ListaImagenes	6796		998	506
08:44:40.486	Thread Group 1-28	ListaImagenes	6840		998	506
08:44:40.787	Thread Group 1-43	ListaImagenes	6948		998	506
08:44:40.767	Thread Group 1-42	ListaImagenes	6998		998	506
08:44:40.847	Thread Group 1-46	ListaImagenes	7389		998	506
08:44:40.808	Thread Group 1-44	ListaImagenes	7429		998	506
08:44:40.867	Thread Group 1-47	ListaImagenes	7384		998	506
08:44:40.887	Thread Group 1-48	ListaImagenes	7364		998	506

Fig. 54: Resultados prueba de carga a endpoint para mostrar imágenes



Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
09:22:33.626	Thread Group 1-1	ListarProductosCarrito	2987		1006	412	96	48
09:22:33.653	Thread Group 1-2	ListarProductosCarrito	2978		1006	412	95	48
09:22:33.686	Thread Group 1-4	ListarProductosCarrito	2978		1006	412	95	48
09:22:33.665	Thread Group 1-3	ListarProductosCarrito	3000		1006	412	95	47
09:22:33.705	Thread Group 1-5	ListarProductosCarrito	2992		1006	412	94	47
09:22:33.805	Thread Group 1-10	ListarProductosCarrito	2994		1006	412	95	47
09:22:33.847	Thread Group 1-12	ListarProductosCarrito	2980		1006	412	95	47
09:22:33.826	Thread Group 1-11	ListarProductosCarrito	3001		1006	412	95	47
09:22:33.725	Thread Group 1-6	ListarProductosCarrito	3145		1006	412	95	47
09:22:33.906	Thread Group 1-15	ListarProductosCarrito	3506		1006	412	94	47
09:22:33.765	Thread Group 1-8	ListarProductosCarrito	4113		1006	412	96	48
09:22:33.746	Thread Group 1-7	ListarProductosCarrito	4134		1006	412	95	48
09:22:33.926	Thread Group 1-16	ListarProductosCarrito	4510		1006	412	95	48
09:22:33.986	Thread Group 1-19	ListarProductosCarrito	4450		1006	412	95	47
09:22:33.784	Thread Group 1-9	ListarProductosCarrito	5113		1006	412	96	48
09:22:33.866	Thread Group 1-13	ListarProductosCarrito	5067		1006	412	94	47
09:22:33.885	Thread Group 1-14	ListarProductosCarrito	5390		1006	412	95	47
09:22:33.966	Thread Group 1-18	ListarProductosCarrito	5329		1006	412	95	47
09:22:34.007	Thread Group 1-20	ListarProductosCarrito	5288		1006	412	94	47
09:22:34.044	Thread Group 1-22	ListarProductosCarrito	5295		1006	412	95	47
09:22:33.945	Thread Group 1-17	ListarProductosCarrito	5423		1006	412	95	47
09:22:34.104	Thread Group 1-25	ListarProductosCarrito	5312		1006	412	95	47
09:22:34.184	Thread Group 1-29	ListarProductosCarrito	5242		1006	412	94	47
09:22:34.145	Thread Group 1-27	ListarProductosCarrito	5288		1006	412	95	47
09:22:34.244	Thread Group 1-32	ListarProductosCarrito	5225		1006	412	95	47
09:22:34.225	Thread Group 1-31	ListarProductosCarrito	5244		1006	412	95	47
09:22:34.024	Thread Group 1-21	ListarProductosCarrito	5472		1006	412	94	47
09:22:34.305	Thread Group 1-35	ListarProductosCarrito	5401		1006	412	95	47
09:22:34.064	Thread Group 1-23	ListarProductosCarrito	5810		1006	412	95	47
09:22:34.325	Thread Group 1-36	ListarProductosCarrito	6084		1006	412	95	47
09:22:34.124	Thread Group 1-26	ListarProductosCarrito	6402		1006	412	95	47
09:22:34.086	Thread Group 1-24	ListarProductosCarrito	6440		1006	412	95	47
09:22:34.385	Thread Group 1-39	ListarProductosCarrito	6678		1006	412	95	47
09:22:34.344	Thread Group 1-37	ListarProductosCarrito	6720		1006	412	95	47
09:22:34.165	Thread Group 1-28	ListarProductosCarrito	7356		1006	412	95	47
09:22:34.204	Thread Group 1-30	ListarProductosCarrito	7371		1006	412	95	47
09:22:34.265	Thread Group 1-33	ListarProductosCarrito	7635		1006	412	95	47
09:22:34.365	Thread Group 1-38	ListarProductosCarrito	7556		1006	412	96	49
09:22:34.284	Thread Group 1-34	ListarProductosCarrito	7638		1006	412	95	47
09:22:34.406	Thread Group 1-40	ListarProductosCarrito	7575		1006	412	94	47
09:22:34.444	Thread Group 1-42	ListarProductosCarrito	7548		1006	412	95	47

Fig. 55: Resultados prueba de carga a endpoint para mostrar productos de un carrito

Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
09:30:10.995	Thread Group 1-2	ListarPedidos	2982		1329	426	96	48
09:30:10.976	Thread Group 1-1	ListarPedidos	3005		1329	426	96	48
09:30:11.016	Thread Group 1-3	ListarPedidos	2982		1329	426	95	47
09:30:11.036	Thread Group 1-4	ListarPedidos	2982		1329	426	96	48
09:30:11.595	Thread Group 1-32	ListarPedidos	2894		760	426	96	48
09:30:11.056	Thread Group 1-5	ListarPedidos	3838		1329	426	95	47
09:30:11.075	Thread Group 1-6	ListarPedidos	4830		1329	426	94	47
09:30:11.095	Thread Group 1-7	ListarPedidos	4825		1329	426	96	48
09:30:11.136	Thread Group 1-9	ListarPedidos	5476		1329	426	95	48
09:30:11.175	Thread Group 1-11	ListarPedidos	5467		1329	426	94	47
09:30:11.155	Thread Group 1-10	ListarPedidos	5487		1329	426	95	47
09:30:11.116	Thread Group 1-8	ListarPedidos	5530		1329	426	95	47
09:30:11.195	Thread Group 1-12	ListarPedidos	5727		1329	426	95	48
09:30:11.216	Thread Group 1-13	ListarPedidos	5723		1329	426	96	48
09:30:11.237	Thread Group 1-14	ListarPedidos	6329		1329	426	95	47
09:30:11.255	Thread Group 1-15	ListarPedidos	6640		1329	426	95	48
09:30:11.855	Thread Group 1-45	ListarPedidos	6123		1329	426	94	47
09:30:11.495	Thread Group 1-27	ListarPedidos	6485		1329	426	96	48
09:30:11.695	Thread Group 1-37	ListarPedidos	6285		1329	426	96	47
09:30:11.796	Thread Group 1-42	ListarPedidos	6382		1329	426	96	48
09:30:11.275	Thread Group 1-16	ListarPedidos	7257		1329	426	95	47
09:30:11.295	Thread Group 1-17	ListarPedidos	7248		1329	426	95	48
09:30:11.535	Thread Group 1-29	ListarPedidos	7620		1329	426	95	48
09:30:11.876	Thread Group 1-46	ListarPedidos	7280		1329	426	96	48
09:30:11.375	Thread Group 1-21	ListarPedidos	7900		1329	426	96	48
09:30:11.355	Thread Group 1-20	ListarPedidos	7921		1329	426	95	47
09:30:11.336	Thread Group 1-19	ListarPedidos	7940		1329	426	95	47
09:30:11.316	Thread Group 1-18	ListarPedidos	7963		1329	426	95	48
09:30:11.397	Thread Group 1-22	ListarPedidos	8150		1329	426	95	47
09:30:11.416	Thread Group 1-23	ListarPedidos	8147		1329	426	96	48
09:30:11.576	Thread Group 1-31	ListarPedidos	8583		1329	426	95	48
09:30:11.816	Thread Group 1-43	ListarPedidos	8346		1329	426	96	48
09:30:11.435	Thread Group 1-24	ListarPedidos	8760		1329	426	96	48
09:30:11.456	Thread Group 1-25	ListarPedidos	9068		1329	426	96	48
09:30:11.775	Thread Group 1-41	ListarPedidos	8837		1329	426	95	47
09:30:11.915	Thread Group 1-48	ListarPedidos	8698		1329	426	95	48
09:30:11.556	Thread Group 1-30	ListarPedidos	9070		1329	426	95	47
09:30:11.636	Thread Group 1-34	ListarPedidos	9165		1329	426	95	47
09:30:11.835	Thread Group 1-44	ListarPedidos	9316		1329	426	95	48
09:30:11.955	Thread Group 1-50	ListarPedidos	9197		1329	426	95	47
09:30:11.716	Thread Group 1-38	ListarPedidos	10069		1329	426	95	47

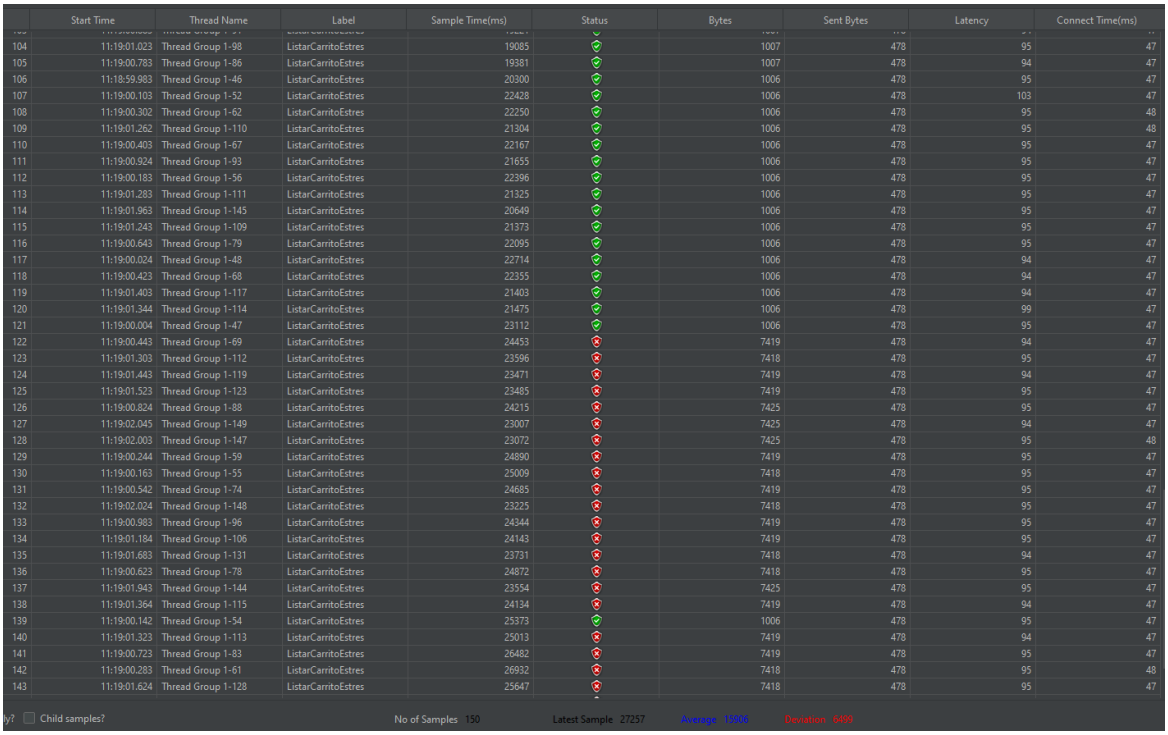
Fig. 56: Resultados prueba de carga a endpoint para listar pedidos

Pruebas de estrés

	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
103	11:02:16.359	Thread Group 1-67	ListarImagenesEstres	18521		998	506	95	48
104	11:02:17.160	Thread Group 1-107	ListarImagenesEstres	17777		998	506	96	47
105	11:02:16.440	Thread Group 1-71	ListarImagenesEstres	18547		998	506	94	47
106	11:02:16.321	Thread Group 1-65	ListarImagenesEstres	18696		998	506	95	47
107	11:02:16.280	Thread Group 1-63	ListarImagenesEstres	18747		998	506	95	47
108	11:02:16.561	Thread Group 1-77	ListarImagenesEstres	18495		998	506	95	47
109	11:02:17.721	Thread Group 1-135	ListarImagenesEstres	17343		998	506	95	48
110	11:02:17.520	Thread Group 1-125	ListarImagenesEstres	17829		998	506	95	47
111	11:02:16.640	Thread Group 1-81	ListarImagenesEstres	18711		998	506	95	47
112	11:02:17.961	Thread Group 1-147	ListarImagenesEstres	17408		998	506	94	47
113	11:02:17.919	Thread Group 1-145	ListarImagenesEstres	17474		998	506	95	48
114	11:02:17.939	Thread Group 1-146	ListarImagenesEstres	17463		998	506	95	47
115	11:02:16.940	Thread Group 1-96	ListarImagenesEstres	18490		998	506	95	47
116	11:02:16.119	Thread Group 1-55	ListarImagenesEstres	19313		998	506	96	48
117	11:02:17.020	Thread Group 1-100	ListarImagenesEstres	18453		998	506	95	47
118	11:02:16.739	Thread Group 1-86	ListarImagenesEstres	18757		998	506	95	48
119	11:02:16.681	Thread Group 1-83	ListarImagenesEstres	18848		998	506	96	48
120	11:02:17.139	Thread Group 1-106	ListarImagenesEstres	18391		998	506	95	48
121	11:02:17.379	Thread Group 1-118	ListarImagenesEstres	19785		7433	506	95	48
122	11:02:16.800	Thread Group 1-89	ListarImagenesEstres	20380		7433	506	95	47
123	11:02:17.760	Thread Group 1-137	ListarImagenesEstres	19475		7439	506	94	47
124	11:02:17.340	Thread Group 1-116	ListarImagenesEstres	19949		7433	506	95	47
125	11:02:17.741	Thread Group 1-136	ListarImagenesEstres	19578		7433	506	95	47
126	11:02:17.559	Thread Group 1-127	ListarImagenesEstres	19764		7440	506	95	47
127	11:02:17.619	Thread Group 1-130	ListarImagenesEstres	19752		7433	506	96	48
128	11:02:16.880	Thread Group 1-93	ListarImagenesEstres	20484		7432	506	95	48
129	11:02:17.260	Thread Group 1-112	ListarImagenesEstres	20216		998	506	95	47
130	11:02:17.201	Thread Group 1-109	ListarImagenesEstres	20459		7433	506	95	47
131	11:02:16.701	Thread Group 1-84	ListarImagenesEstres	20961		7426	506	95	48
132	11:02:16.239	Thread Group 1-61	ListarImagenesEstres	21424		7432	506	95	47
133	11:02:16.059	Thread Group 1-52	ListarImagenesEstres	21632		7440	506	95	48
134	11:02:17.301	Thread Group 1-114	ListarImagenesEstres	20418		7433	506	95	47
135	11:02:18.020	Thread Group 1-150	ListarImagenesEstres	19710		7433	506	94	47
136	11:02:16.139	Thread Group 1-56	ListarImagenesEstres	21609		7426	506	95	48
137	11:02:16.900	Thread Group 1-94	ListarImagenesEstres	20887		7433	506	95	48
138	11:02:17.119	Thread Group 1-105	ListarImagenesEstres	20676		7433	506	95	48
139	11:02:17.061	Thread Group 1-102	ListarImagenesEstres	20769		7432	506	95	48
140	11:02:17.280	Thread Group 1-113	ListarImagenesEstres	20552		7439	506	95	48
141	11:02:16.981	Thread Group 1-98	ListarImagenesEstres	22476		7432	506	94	47
142	11:02:17.820	Thread Group 1-140	ListarImagenesEstres	21669		7439	506	94	47
143	11:02:16.659	Thread Group 1-82	ListarImagenesEstres	22887		7432	506	95	48

lly? Child samples? No of Samples 150 Latest Sample 22641 Average 12891 Deviation 5923

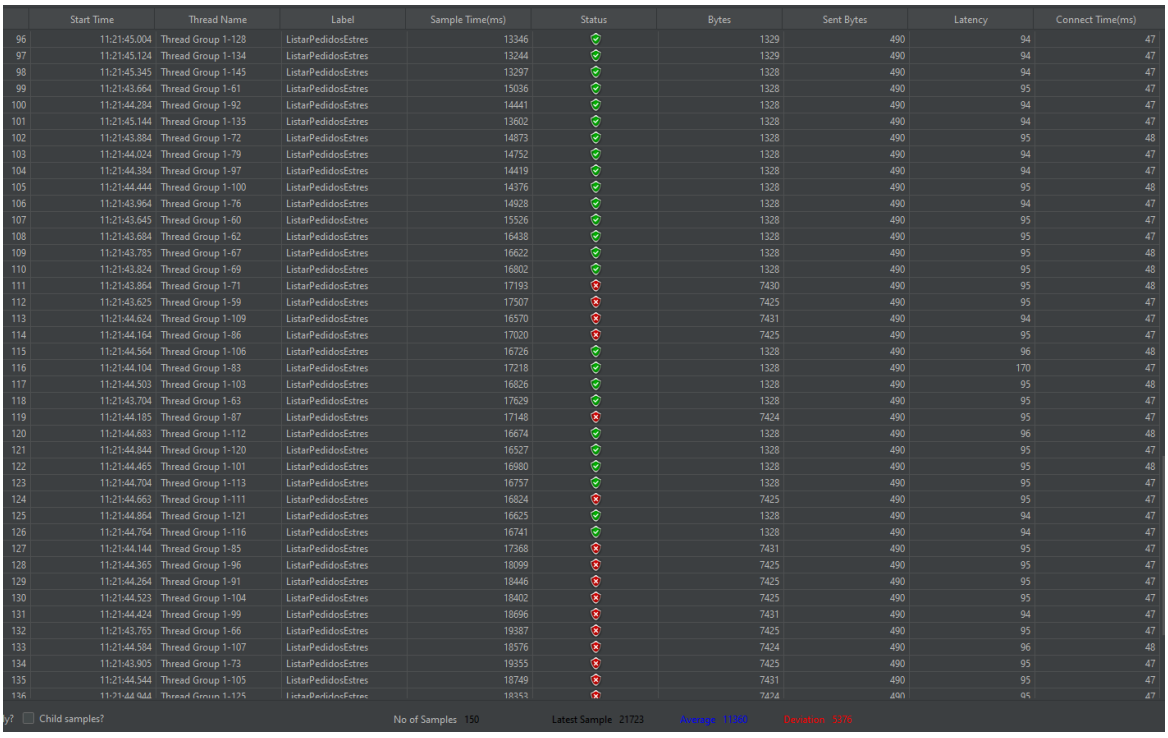
Fig. 57: Resultados pruebas de estrés a endpoint para mostrar imágenes



	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
104	11:19:01.023	Thread Group 1-98	ListarCarritoEstres	19085		1007	478	95	47
105	11:19:00.783	Thread Group 1-86	ListarCarritoEstres	19381		1007	478	94	47
106	11:18:59.983	Thread Group 1-46	ListarCarritoEstres	20300		1006	478	95	47
107	11:19:00.103	Thread Group 1-52	ListarCarritoEstres	22428		1006	478	103	47
108	11:19:00.302	Thread Group 1-62	ListarCarritoEstres	22250		1006	478	95	48
109	11:19:01.262	Thread Group 1-110	ListarCarritoEstres	21304		1006	478	95	48
110	11:19:00.403	Thread Group 1-67	ListarCarritoEstres	22167		1006	478	95	47
111	11:19:00.924	Thread Group 1-93	ListarCarritoEstres	21655		1006	478	95	47
112	11:19:00.183	Thread Group 1-56	ListarCarritoEstres	22396		1006	478	95	47
113	11:19:01.283	Thread Group 1-111	ListarCarritoEstres	21325		1006	478	95	47
114	11:19:01.963	Thread Group 1-145	ListarCarritoEstres	20649		1006	478	95	47
115	11:19:01.243	Thread Group 1-109	ListarCarritoEstres	21373		1006	478	95	47
116	11:19:00.643	Thread Group 1-79	ListarCarritoEstres	22095		1006	478	95	47
117	11:19:00.024	Thread Group 1-48	ListarCarritoEstres	22714		1006	478	94	47
118	11:19:00.423	Thread Group 1-68	ListarCarritoEstres	22355		1006	478	94	47
119	11:19:01.403	Thread Group 1-117	ListarCarritoEstres	21403		1006	478	94	47
120	11:19:01.344	Thread Group 1-114	ListarCarritoEstres	21475		1006	478	99	47
121	11:19:00.004	Thread Group 1-47	ListarCarritoEstres	23112		1006	478	95	47
122	11:19:00.443	Thread Group 1-69	ListarCarritoEstres	24453		7419	478	94	47
123	11:19:01.303	Thread Group 1-112	ListarCarritoEstres	23596		7418	478	95	47
124	11:19:01.443	Thread Group 1-119	ListarCarritoEstres	23471		7419	478	94	47
125	11:19:01.523	Thread Group 1-123	ListarCarritoEstres	23485		7419	478	95	47
126	11:19:00.824	Thread Group 1-88	ListarCarritoEstres	24215		7425	478	95	47
127	11:19:02.045	Thread Group 1-149	ListarCarritoEstres	23007		7425	478	94	47
128	11:19:02.003	Thread Group 1-147	ListarCarritoEstres	23072		7425	478	95	48
129	11:19:00.244	Thread Group 1-59	ListarCarritoEstres	24890		7419	478	95	47
130	11:19:00.163	Thread Group 1-55	ListarCarritoEstres	25009		7418	478	95	47
131	11:19:00.542	Thread Group 1-74	ListarCarritoEstres	24685		7419	478	95	47
132	11:19:02.024	Thread Group 1-148	ListarCarritoEstres	23225		7418	478	95	47
133	11:19:00.983	Thread Group 1-96	ListarCarritoEstres	24344		7419	478	95	47
134	11:19:01.184	Thread Group 1-106	ListarCarritoEstres	24143		7419	478	95	47
135	11:19:01.683	Thread Group 1-131	ListarCarritoEstres	23731		7418	478	94	47
136	11:19:00.623	Thread Group 1-78	ListarCarritoEstres	24872		7418	478	95	47
137	11:19:01.943	Thread Group 1-144	ListarCarritoEstres	23554		7425	478	95	47
138	11:19:01.364	Thread Group 1-115	ListarCarritoEstres	24134		7419	478	94	47
139	11:19:00.142	Thread Group 1-54	ListarCarritoEstres	25373		1006	478	95	47
140	11:19:01.323	Thread Group 1-113	ListarCarritoEstres	25013		7419	478	94	47
141	11:19:00.723	Thread Group 1-83	ListarCarritoEstres	26482		7419	478	95	47
142	11:19:00.283	Thread Group 1-61	ListarCarritoEstres	26932		7418	478	95	48
143	11:19:01.624	Thread Group 1-128	ListarCarritoEstres	25647		7418	478	95	47

ly? Child samples? No of Samples 150 Latest Sample 27257 Average 19806 Deviation 6499

Fig. 58: Resultados pruebas de estrés a endpoint para mostrar productos de un carrito



	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
96	11:21:45.004	Thread Group 1-128	ListarPedidosEstres	13346		1329	490	94	47
97	11:21:45.124	Thread Group 1-134	ListarPedidosEstres	13244		1329	490	94	47
98	11:21:45.345	Thread Group 1-145	ListarPedidosEstres	13297		1328	490	94	47
99	11:21:43.664	Thread Group 1-61	ListarPedidosEstres	15036		1328	490	95	47
100	11:21:44.284	Thread Group 1-92	ListarPedidosEstres	14441		1328	490	94	47
101	11:21:45.144	Thread Group 1-135	ListarPedidosEstres	13602		1328	490	94	47
102	11:21:43.884	Thread Group 1-72	ListarPedidosEstres	14873		1328	490	95	48
103	11:21:44.024	Thread Group 1-79	ListarPedidosEstres	14752		1328	490	94	47
104	11:21:44.384	Thread Group 1-97	ListarPedidosEstres	14419		1328	490	94	47
105	11:21:44.444	Thread Group 1-100	ListarPedidosEstres	14376		1328	490	95	48
106	11:21:43.964	Thread Group 1-76	ListarPedidosEstres	14928		1328	490	94	47
107	11:21:43.645	Thread Group 1-60	ListarPedidosEstres	15526		1328	490	95	47
108	11:21:43.684	Thread Group 1-62	ListarPedidosEstres	16438		1328	490	95	47
109	11:21:43.785	Thread Group 1-67	ListarPedidosEstres	16622		1328	490	95	48
110	11:21:43.824	Thread Group 1-69	ListarPedidosEstres	16802		1328	490	95	48
111	11:21:43.864	Thread Group 1-71	ListarPedidosEstres	17193		7430	490	95	48
112	11:21:43.625	Thread Group 1-59	ListarPedidosEstres	17507		7425	490	95	47
113	11:21:44.624	Thread Group 1-109	ListarPedidosEstres	16570		7431	490	94	47
114	11:21:44.164	Thread Group 1-86	ListarPedidosEstres	17020		7425	490	95	47
115	11:21:44.564	Thread Group 1-106	ListarPedidosEstres	16726		1328	490	96	48
116	11:21:44.104	Thread Group 1-83	ListarPedidosEstres	17218		1328	490	170	47
117	11:21:44.503	Thread Group 1-103	ListarPedidosEstres	16826		1328	490	95	48
118	11:21:43.704	Thread Group 1-63	ListarPedidosEstres	17629		1328	490	95	47
119	11:21:44.185	Thread Group 1-87	ListarPedidosEstres	17148		7424	490	95	47
120	11:21:44.683	Thread Group 1-112	ListarPedidosEstres	16674		1328	490	96	48
121	11:21:44.844	Thread Group 1-120	ListarPedidosEstres	16527		1328	490	95	47
122	11:21:44.465	Thread Group 1-101	ListarPedidosEstres	16980		1328	490	95	48
123	11:21:44.704	Thread Group 1-113	ListarPedidosEstres	16757		1328	490	95	47
124	11:21:44.663	Thread Group 1-111	ListarPedidosEstres	16824		7425	490	95	47
125	11:21:44.864	Thread Group 1-121	ListarPedidosEstres	16625		1328	490	94	47
126	11:21:44.764	Thread Group 1-116	ListarPedidosEstres	16741		1328	490	94	47
127	11:21:44.144	Thread Group 1-85	ListarPedidosEstres	17368		7431	490	95	47
128	11:21:44.365	Thread Group 1-96	ListarPedidosEstres	18099		7425	490	95	47
129	11:21:44.264	Thread Group 1-91	ListarPedidosEstres	18446		7425	490	95	47
130	11:21:44.523	Thread Group 1-104	ListarPedidosEstres	18402		7425	490	95	47
131	11:21:44.424	Thread Group 1-99	ListarPedidosEstres	18696		7431	490	94	47
132	11:21:43.765	Thread Group 1-66	ListarPedidosEstres	19387		7425	490	95	47
133	11:21:44.584	Thread Group 1-107	ListarPedidosEstres	18576		7424	490	96	48
134	11:21:43.905	Thread Group 1-73	ListarPedidosEstres	19355		7425	490	95	47
135	11:21:44.544	Thread Group 1-105	ListarPedidosEstres	18749		7431	490	95	47
136	11:21:44.944	Thread Group 1-125	ListarPedidosEstres	18353		7424	490	95	47

ly? Child samples? No of Samples 150 Latest Sample 21723 Average 11360 Deviation 5376

Fig. 59: Resultados pruebas de estrés a endpoint para listar pedidos

6.2 ANEXO III Manual de usuario

Dentro de este anexo se puede encontrar el enlace del manual de usuario, en el cual se explica de manera clara la implementación de métodos para la funcionalidad del backend.

https://youtu.be/45Pgv7Z5WbA

6.3 ANEXO IV Manual de instalación

En el siguiente enlace se encuentra el manual de instalación para este proyecto, junto con el código fuente.

https://github.com/PPC80/tesis