

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

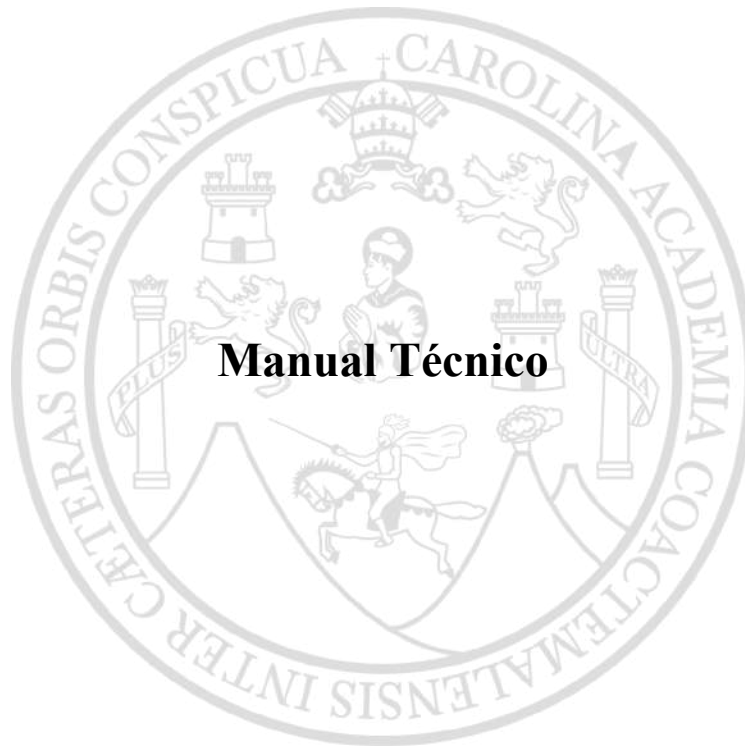
Centro Universitario de Occidente
División de Ciencias de la Ingeniería

Curso: Laboratorio de Manejo e Implementación de Archivos

Sección: A

Docente: Pedro Luis Domingo Vásquez

Auxiliar: Fernando Ocaña



Carné:

202030799

Nombre:

Manuel Antonio Rojas Paxtor

Quetzaltenango 11 de mayo de 2023

INTRODUCCIÓN

El presente documento es el Manual Técnico del sistema web "eCommerce GT". Diseñado para brindar información detallada a aquellos interesados en comprender y contribuir al desarrollo y mantenimiento de esta plataforma, este manual proporciona una visión completa de la arquitectura, tecnologías utilizadas y los procesos que sustentan su funcionamiento.

"eCommerce GT" es una plataforma en línea enfocada en facilitar la compra y venta de productos exclusivamente en Guatemala. Con un enfoque centrado en la eficiencia y seguridad, esta aplicación ha sido desarrollada utilizando una amplia gama de tecnologías y herramientas modernas para garantizar una experiencia fluida y segura para los usuarios.

En este manual, te adentrarás en los componentes clave de "eCommerce GT". Desde la estructura de la base de datos y la arquitectura del sistema, hasta los aspectos de seguridad y rendimiento, cada aspecto técnico se analiza de manera exhaustiva para proporcionar una comprensión completa del funcionamiento interno de la plataforma.

A medida que avances en la lectura, descubrirás cómo se gestionan los productos en el catálogo, cómo se realizan las transacciones seguras utilizando tarjetas de crédito ficticias, cómo se controlan los pedidos y cómo se generan informes para respaldar la toma de decisiones empresariales.

Este manual también ofrece orientación sobre el desarrollo y mantenimiento continuo de "eCommerce GT". Aprenderás sobre las mejores prácticas de codificación, la implementación de nuevas funcionalidades y la solución de problemas comunes que puedan surgir.

Ya sea que estés interesado en contribuir al desarrollo de "eCommerce GT" como programador, administrador de sistemas o cualquier otro rol técnico, este manual te proporcionará los conocimientos necesarios para comprender y colaborar efectivamente en el desarrollo y mejora constante de esta plataforma.

Objetivos

Objetivo General

Aplicar los conocimientos adquiridos durante el laboratorio para crear software que permita automatizar procesos y almacenar información de forma segura y eficiente.

Objetivos Específicos

- Diseñar y desarrollar una aplicación web de comercio electrónico que permita a los usuarios comprar y vender productos en línea.
- Aplicar conocimientos sobre herramientas NoSQL, específicamente MongoDB para la gestión de una base de datos.
- Realizar una REST API con NodeJS que reciba peticiones HTTP, permita la recolección de datos a una base de datos de MongoDB y devuelva alguna respuesta al cliente.
- Crear una aplicación gráfica que permita al usuario interactuar con la base de datos en base a su rol en el sistema y de forma amigable.
- Aplicar buenas prácticas tanto en el desarrollo de una aplicación en JavaScript y una base de datos construida en MongoDB.

Tecnologías utilizadas

Al ser una aplicación web, se solicitó realizar la aplicación con herramientas de javascript, por lo que a continuación se describen las tecnologías utilizadas para el cliente y el servidor

Backend

Tecnologías utilizadas en el backend de la aplicación:

1. Node.js:

Node.js es un entorno de ejecución de JavaScript en el lado del servidor que permite construir aplicaciones web escalables y de alto rendimiento. Proporciona un modelo de E/S sin bloqueo y basado en eventos, lo que lo hace ideal para aplicaciones de tiempo real y con gran cantidad de solicitudes.

2. MongoDB:

MongoDB es una base de datos NoSQL orientada a documentos. Es altamente escalable y flexible, lo que lo hace adecuado para almacenar y recuperar grandes cantidades de datos estructurados y no estructurados. MongoDB utiliza documentos JSON-like para almacenar datos, lo que permite una fácil integración con aplicaciones JavaScript y Node.js.

Dependencias utilizadas en el proyecto:

- **bcrypt**: Librería para el hashing de contraseñas, proporcionando un cifrado seguro para almacenar contraseñas de forma segura en la base de datos.
- **cookie-parser**: Middleware para analizar las cookies enviadas por el navegador y facilitar su manipulación en Node.js.
- **cors**: Middleware que permite habilitar el acceso a recursos en el servidor desde dominios diferentes, lo que facilita la comunicación entre el frontend y el backend de la aplicación.
- **date-fns**: Biblioteca para manipulación y formateo de fechas en JavaScript, proporcionando funciones y utilidades para trabajar con fechas de manera sencilla y eficiente.
- **dotenv**: Módulo que carga variables de entorno desde un archivo .env, lo que permite almacenar configuraciones sensibles de forma segura y separada del código fuente.
- **express**: Framework de aplicaciones web para Node.js que simplifica la creación de rutas, el manejo de peticiones y respuestas HTTP, y la gestión de middleware.
- **fs-extra**: Módulo que extiende las funcionalidades del módulo fs estándar de Node.js, proporcionando métodos adicionales para trabajar con el sistema de archivos de forma más conveniente.

- **jsonwebtoken:** Biblioteca para la creación y verificación de tokens de JSON Web, utilizada para la autenticación y autorización en aplicaciones web.
- **mongoose:** Biblioteca de modelado de objetos para Node.js que facilita la interacción con MongoDB al proporcionar una capa de abstracción sobre las operaciones de la base de datos y la definición de esquemas.
- **multer:** Middleware para el manejo de la carga de archivos en aplicaciones web. Permite recibir archivos enviados desde formularios HTML y almacenarlos en el servidor de forma sencilla.
- **uuid:** Biblioteca para generar identificadores únicos universales (UUID). Se utiliza para asignar identificadores únicos a entidades en la aplicación, como usuarios o productos, garantizando su unicidad y evitando colisiones.
- **nodemon:** Herramienta de desarrollo que supervisa los cambios en los archivos del proyecto y reinicia automáticamente el servidor Node.js cuando se detectan cambios. Esto facilita el proceso de desarrollo al eliminar la necesidad de reiniciar manualmente el servidor después de cada cambio realizado en el código.
- **express:** Framework de aplicaciones web para Node.js que simplifica la creación de rutas, el manejo de peticiones y respuestas HTTP, y la gestión de middleware. Express es una dependencia fundamental para el desarrollo del backend de la aplicación, ya que proporciona una estructura y funciones útiles para construir y organizar una aplicación web.

Frontend

Tecnologías utilizadas en el frontend:

- **Vite:** Es un sistema de compilación y desarrollo ultrarrápido para aplicaciones web. Vite se utiliza para construir y empaquetar el código del frontend, optimizando el rendimiento y facilitando el desarrollo. Proporciona una experiencia de desarrollo en tiempo real y recarga rápida del navegador, lo que agiliza el proceso de desarrollo.
- **React:** Es una biblioteca de JavaScript utilizada para construir interfaces de usuario interactivas y reutilizables. React permite crear componentes UI modularizados que se actualizan de manera eficiente cuando cambia el estado de la aplicación. Es una tecnología ampliamente utilizada en el desarrollo frontend debido a su eficiencia y facilidad de uso.
- **Tailwind CSS:** Es un framework CSS utilitario que facilita la creación de interfaces de usuario personalizadas. Tailwind CSS utiliza clases CSS predefinidas para estilizar los componentes y

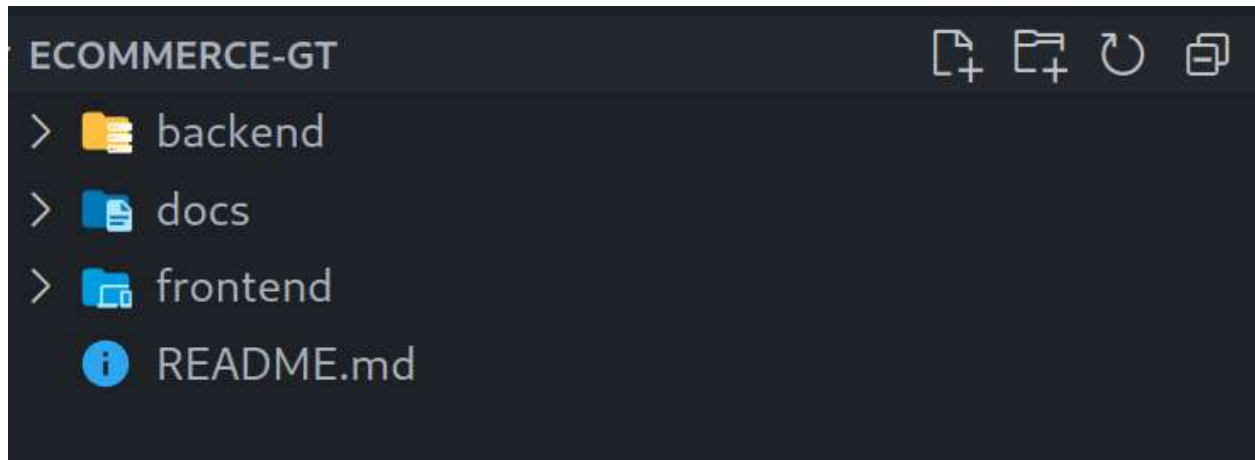
elementos de la interfaz, lo que agiliza el proceso de diseño y facilita la mantenibilidad del código.

Dependencias relevantes utilizadas en el proyecto:

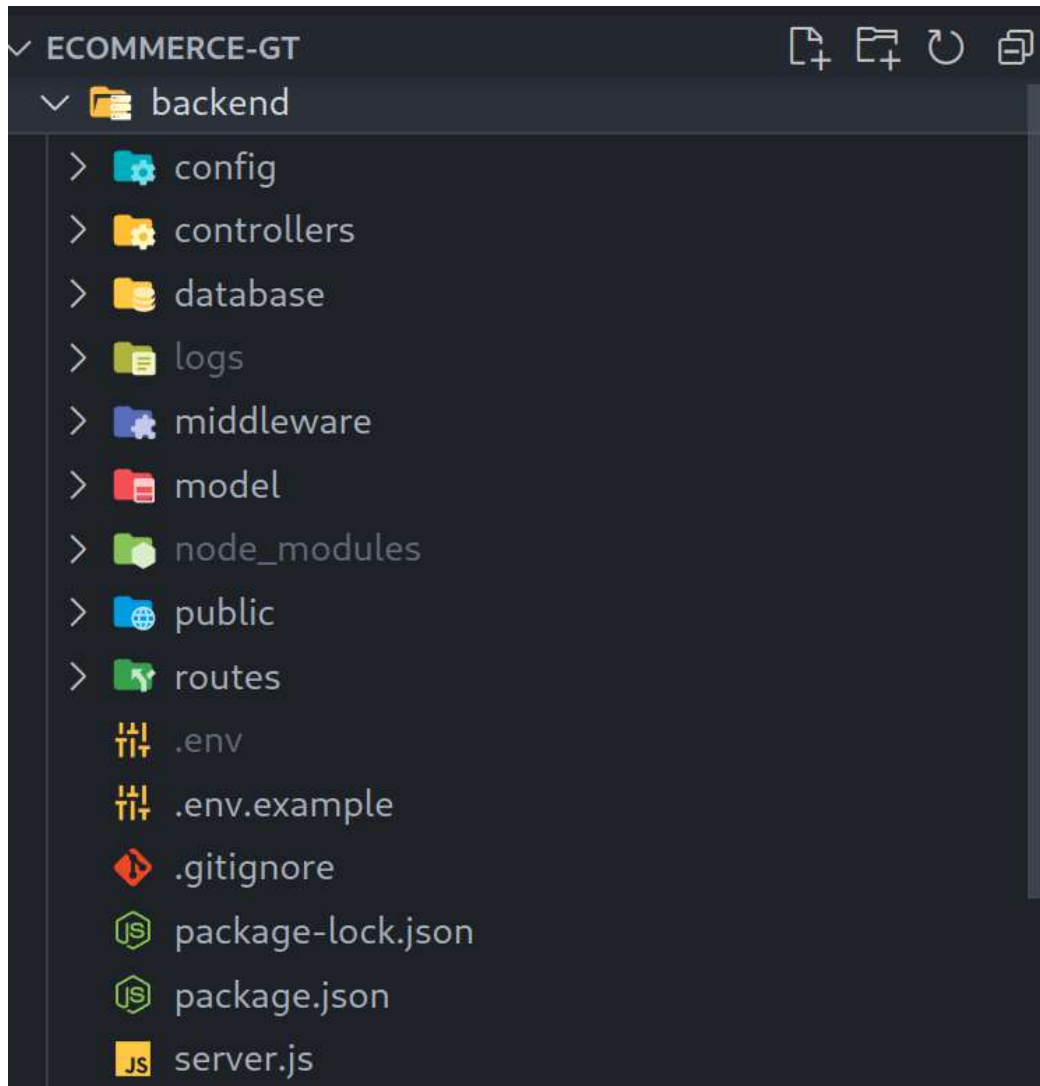
- **react-router-dom**: Proporciona enrutamiento y navegación para aplicaciones de React. Permite definir rutas y enlaces entre diferentes componentes, lo que facilita la navegación dentro de la aplicación.
- **axios**: Es una biblioteca de JavaScript utilizada para realizar solicitudes HTTP desde el cliente. Axios simplifica la comunicación con el backend, permitiendo realizar peticiones GET, POST, PUT, DELETE, entre otras, y manejar las respuestas de manera eficiente.
- **react-hook-form**: Es una biblioteca que facilita la creación de formularios en React. Proporciona funciones y ganchos personalizados que simplifican la validación y el manejo de datos de formulario.
- **tailwindcss/forms**: Un complemento de Tailwind CSS que proporciona estilos prediseñados para formularios, facilitando la creación de formularios con un aspecto coherente y moderno.
- **date-fns**: Es una biblioteca de JavaScript utilizada para manipular, formatear y mostrar fechas y horas. Proporciona una amplia gama de funciones para realizar operaciones comunes en fechas, como cálculos, formateo y ajuste de zonas horarias.

Estructura del proyecto

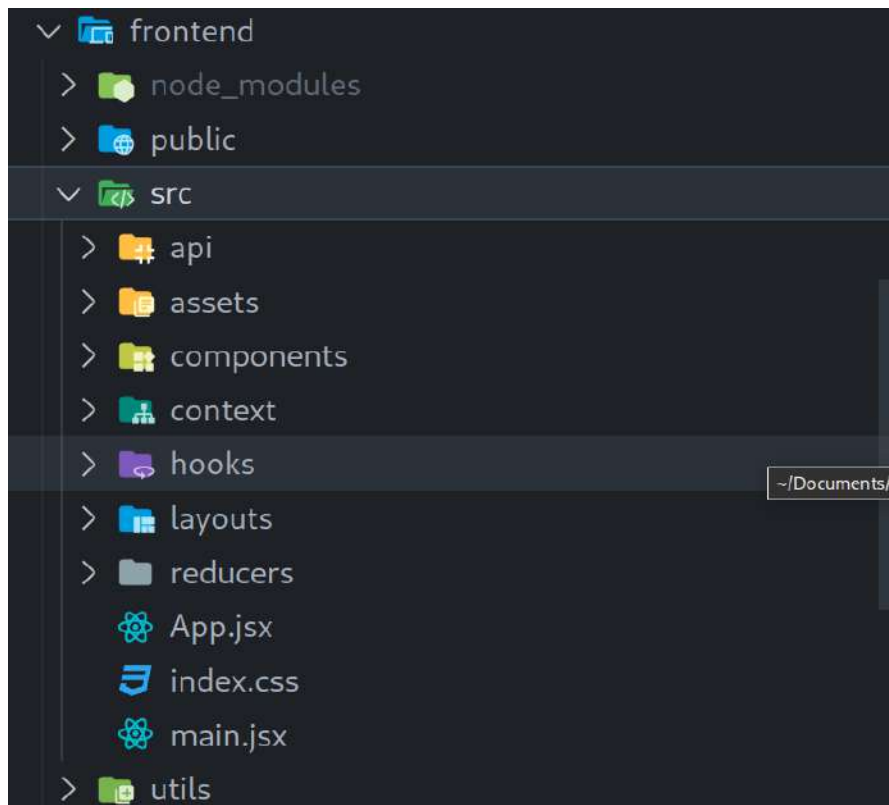
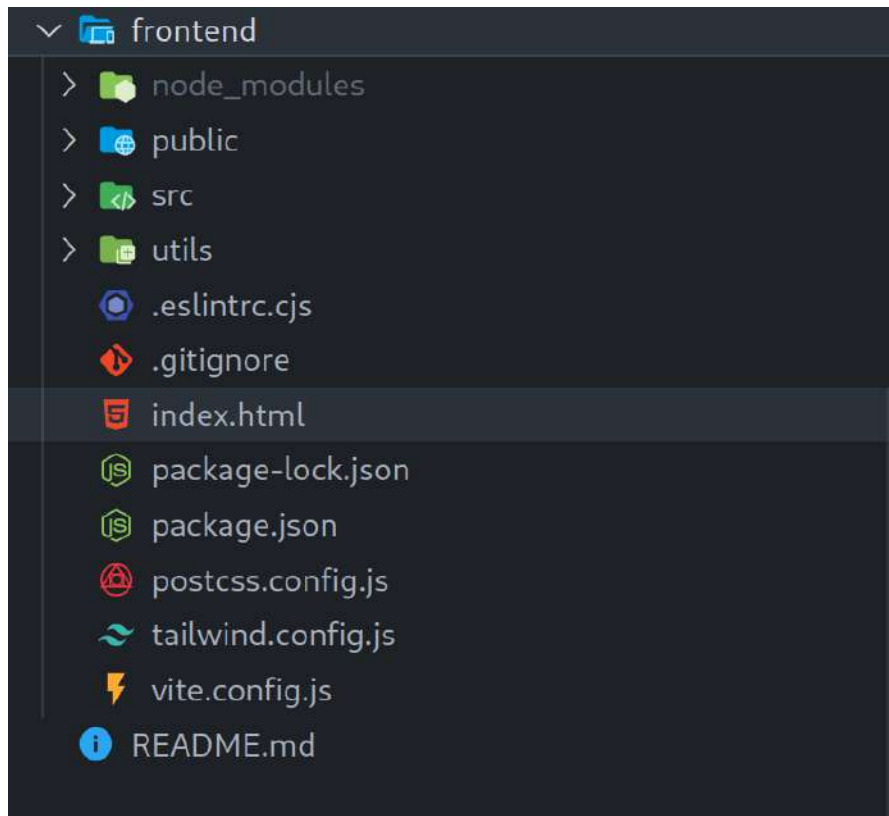
Al descargarse el repositorio se contará con las siguientes carpetas



La estructura del backend es el siguiente



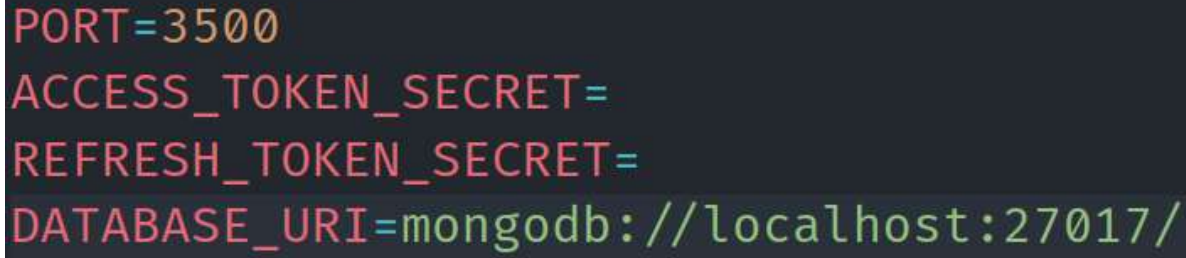
La estructura del frontend es la siguiente:



Configuración del entorno de desarrollo

Se debe tener instalado MongoDB y Node JS en su versión 18.16.0 LTS

Se debe crear un archivo llamado “.env” donde se deben colocar las variables de entorno de la aplicación. Existe un ejemplo en el repositorio en la carpeta de backend. Los token secret deben ser una cadena encriptada en base 64, por el momento es ficticia así que puede colocarse la que sea ya que no funciona en producción.



```
PORT=3500
ACCESS_TOKEN_SECRET=
REFRESH_TOKEN_SECRET=
DATABASE_URI=mongodb://localhost:27017/
```

Para levantar el proyecto es necesario descargarse el repositorio del proyecto <https://github.com/mromer08/eCommerce-GT> y una vez descargado instalar las dependencias tanto para el backend como el frontend. Situados en la raíz del proyecto se pueden realizar los siguientes pasos para instalar las dependencias:

Instalación de dependencias para el backend

```
cd backend/
npm install
```

Instalación de dependencias para el frontend

```
cd frontend/
npm install
```

Luego se debe de crear la base de datos de la aplicación con el nombre de “ecommerce_gt”. Para ello se debe tener instalado [MongoDB](#). Luego en la raíz del proyecto puede abrirse una terminal y cargar el archivo de la base de datos

Creación de la base de datos y carga del archivo inicial

Utilizaremos mongosh para la creación de la base de datos y la carga del archivo inicial, por lo que en la raíz del proyecto se debe hacer lo siguiente:

```
cd ./backend/database
mongosh
use ecommerce_gt
load('./db-ecommerce_gt.js')
exit
```

Backend

Principales endpoints

- `/api/register`: Este endpoint se utiliza para registrar nuevos usuarios en la aplicación. Realiza una sola tarea de crear una cuenta de usuario.
- `/api/auth`: Este endpoint se utiliza para autenticar a los usuarios en la aplicación. Realiza una sola tarea de verificar las credenciales de inicio de sesión y generar un token de acceso.
- `/api/refresh`: Este endpoint se utiliza para solicitar un nuevo token de acceso válido. Permite refrescar el token de autenticación sin necesidad de volver a iniciar sesión.
- `/api/logout`: Este endpoint se utiliza para cerrar la sesión de un usuario. Invalida el token de acceso y evita que se pueda utilizar para acceder a recursos protegidos.
- `/api/categories`: Este endpoint se utiliza para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre las categorías de productos. Permite agregar nuevas categorías, obtener información de categorías existentes, actualizar categorías y eliminar categorías.
- `/api/products`: Este endpoint se utiliza para realizar operaciones CRUD sobre los productos. Permite agregar nuevos productos, obtener información de productos existentes, actualizar productos y eliminar productos.
- `/api/users`: Este endpoint se utiliza para realizar operaciones CRUD sobre los usuarios registrados en la aplicación. Permite agregar nuevos usuarios, obtener información de usuarios existentes, actualizar usuarios y eliminar usuarios. Solo está disponible para usuarios autenticados con roles específicos.
- `/api/cards`: Este endpoint se utiliza para realizar operaciones CRUD sobre las tarjetas de crédito registradas por los usuarios. Permite agregar nuevas tarjetas, obtener información de tarjetas existentes, actualizar tarjetas y eliminar tarjetas. Solo está disponible para usuarios autenticados.
- `/api/sales`: Este endpoint se utiliza para realizar operaciones CRUD sobre las ventas realizadas en la aplicación. Permite agregar nuevas ventas, obtener información de ventas existentes, actualizar ventas y eliminar ventas. Solo está disponible para usuarios autenticados.
- `/api/orders`: Este endpoint se utiliza para realizar operaciones CRUD sobre los pedidos realizados en la aplicación. Permite agregar nuevos pedidos, obtener información de pedidos existentes, actualizar pedidos y eliminar pedidos. Solo está disponible para usuarios autenticados.
- `/api/reports/products-count`: Este subendpoint se utiliza para obtener el recuento de productos en la aplicación. Requiere que el usuario tenga el rol de administrador para acceder a esta información.

- `/api/reports/orders-count`: Este subendpoint se utiliza para obtener los principales clientes basados en la cantidad de pedidos realizados. Requiere que el usuario tenga el rol de administrador para acceder a esta información.
- `/api/reports/sales-count`: Este subendpoint se utiliza para obtener los productos más vendidos. Requiere que el usuario tenga el rol de administrador para acceder a esta información.
- `/api/reports/profits-count`: Este subendpoint se utiliza para obtener los principales clientes basados en las ganancias generadas. Requiere que el usuario tenga el rol de administrador para acceder a esta información.
- `/api/reports/customers-count`: Este subendpoint se utiliza para obtener los principales clientes basados en la cantidad de productos vendidos. Requiere que el usuario tenga el rol de administrador para acceder a esta información.

Frontend

Principales componentes de la aplicación

- Register: Componente utilizado para el registro de nuevos usuarios en la aplicación.
- Login: Componente utilizado para el inicio de sesión de los usuarios.
- Layout: Componente de diseño principal que envuelve la estructura de la aplicación.
- Admin: Componente utilizado para la gestión de tareas administrativas.
- Missing: Componente que se muestra cuando se intenta acceder a una ruta no existente.
- Unauthorized: Componente que se muestra cuando un usuario no autorizado intenta acceder a una ruta protegida.
- RequireAuth: Componente de alto orden utilizado para proteger las rutas y requerir autenticación.
- ProductOverview: Componente utilizado para mostrar una vista detallada de un producto específico.
- ProductsList: Componente utilizado para mostrar una lista de productos.
- NavBarLayout: Componente de diseño que muestra una barra de navegación en la parte superior.
- ProductForm: Componente utilizado para agregar o editar un producto.
- CreditCardList: Componente utilizado para mostrar una lista de tarjetas de crédito asociadas a un usuario.
- OrderList: Componente utilizado para mostrar una lista de pedidos realizados.
- Users: Componente utilizado para mostrar y gestionar usuarios registrados.
- Reports: Componente utilizado para generar y mostrar informes basados en diferentes tipos de reportes.