PRUEBA MAGICLOG

DOCUMENTACIÓN TÉCNICA

1.Introducción

1. 1. Descripción del Proyecto

Propósito: Este proyecto es una aplicación de gestión de productos para un e-commerce que permite a los usuarios (administradores, vendedores y clientes) interactuar con productos y gestionar su inventario. El propósito principal es proporcionar una plataforma donde los usuarios puedan realizar operaciones (Crear, Leer) en productos y realizar un seguimiento de su disponibilidad y detalles.

Objetivos:

- **Administrar Productos:** Permitir a los vendedores agregar, y listar productos en el catálogo.
- **Visualizar Productos:** Facilitar a los clientes y administradores la visualización de productos con filtros y buscador.
- **Gestión de Inventario:** Mantener un registro de la cantidad de productos disponibles y sus detalles.
- **Autenticación y Autorización:** Implementar un sistema seguro de autenticación para diferentes roles de usuario (administradores, vendedores).

1.2. Alcance del Proyecto

Incluye:

- **Frontend:** Desarrollo de una interfaz de usuario en React que permita a los usuarios interactuar con el sistema de productos.

- **Backend:** Creación de una API RESTful en Node.js con Express para gestionar las operaciones CRUD de productos y la autenticación de usuarios.

- **Base de Datos:** Configuración y gestión de una base de datos PostgreSQL para almacenar la información de los productos y usuarios.

- **Autenticación y Autorización:** Implementación de mecanismo de autenticación de usuarios y autorización basada en roles por medio de Token JWT.

- **Documentación de la API:** Provisión de documentación interactiva para la API utilizando la herramienta como Swagger.

1.3. Tecnologías Utilizadas

Frontend:

- **React:** Biblioteca de JavaScript para construir interfaces de usuario interactivas.
- **Axios:** Biblioteca para realizar solicitudes HTTP desde el frontend.
- **React Router:** Biblioteca para manejar la navegación en la aplicación de una sola página (SPA).
- **CSS:** Lenguajes para el diseño y la estilización de la interfaz de usuario.
- **Redux:** Biblioteca para gestionar el estado global de aplicaciones JavaScript, facilitando la sincronización de datos y el manejo predecible de estados.

Backend:

- **Node.js:** Entorno de ejecución para JavaScript en el servidor.
- **Express:** Framework para construir aplicaciones web y API en Node.js.
- **Sequelize:** ORM (Object-Relational Mapping) para gestionar la interacción con la base de datos PostgreSQL.
- **jsonwebtoken (JWT):** Biblioteca para la gestión de tokens de autenticación y autorización.

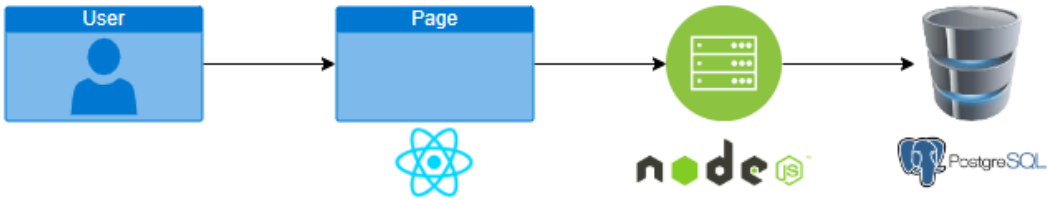
Base de Datos:

- **PostgreSQL:** Sistema de gestión de bases de datos relacional para almacenar la información de productos y usuarios.

Herramientas de Desarrollo:

- **Git:** Sistema de control de versiones para gestionar el código fuente.
- **Insomnia:** Herramientas para probar y documentar las API RESTful.

2. Arquitectura del sistema



Descripción de Componentes

1. Frontend (React):

- **Responsabilidad:** Proporcionar una interfaz de usuario interactiva. Utiliza componentes React para construir la UI, maneja el estado de la aplicación y envía solicitudes HTTP al backend.
- **Tecnologías:** React, Axios, React Router, CSS.

2. Backend (Node.js / Express):

- **Responsabilidad:** Gestionar la lógica del servidor, autenticar usuarios, procesar solicitudes de API, interactuar con la base de datos y enviar respuestas al frontend.
- **Tecnologías:** Node.js, Express, Sequelize (ORM).

3. Base de Datos (PostgreSQL):

- **Responsabilidad:** Almacenar datos de la aplicación, como información de productos y usuarios. Proporciona una interfaz para realizar consultas, actualizaciones y eliminaciones.

- **Tecnologías:** PostgreSQL.

Flujo de Datos:

1. **Solicitud del Usuario:** El usuario realiza una acción en la interfaz (por ejemplo, listado de productos en vista Home).

2. **Petición al Frontend:** La acción del usuario es capturada por React, que realiza una solicitud HTTP al backend para obtener datos (por ejemplo, productos).

3. **Solicitud al Backend:** El backend (Node.js / Express) recibe la solicitud, procesa la lógica requerida y realiza consultas a la base de datos para obtener la información solicitada.

4. **Consulta a la Base de Datos:** Sequelize interactúa con la base de datos PostgreSQL para recuperar la información solicitada (por ejemplo, datos de productos).

5. **Respuesta del Backend:** El backend procesa los datos recibidos de la base de datos y envía una respuesta al frontend.

6. **Actualización del Frontend:** React recibe la respuesta del backend y actualiza la interfaz de usuario con la nueva información (por ejemplo, mostrando una lista de productos).

3. Frontend

Estructura del Proyecto

La estructura de carpetas y archivos del frontend sigue una organización clara para facilitar el mantenimiento y escalabilidad del proyecto. A continuación se detalla la estructura principal:

src/

components/ → **Componentes usados en las páginas**

- Button/
- Header/
- List/
- Login/
- Menu/
- Modal/
- PrivateRoute/
- Search/
- Table/

pages/ → **Páginas principales del proyecto**
- Admin/
- Home/
- Login/
- Vendor/

redux/ → **Estado global de la aplicación (usuario y productos)**
- Actions/
- Reducers/
- store.js

services/ → **Configuración de API y consumo de las mismas**
- adminServices.js
- authService.js
- axios.js
- productService.js

styles/
- color.css

app.js
index.js

Dependencias y Librerías

Las principales dependencias y librerías utilizadas en el proyecto incluyen:

- **React**: Biblioteca principal para construir la interfaz de usuario.
- **Redux**: Para la gestión del estado global de la aplicación.
- **React-Router-Dom**: Para la gestión de rutas y navegación.
- **Axios**: Para realizar solicitudes HTTP a la API.
- **React-Redux**: Biblioteca para conectar Redux con React.
- **Prop-Types**: Para la validación de tipos de props en los componentes.

Estado y Gestión de Datos

La gestión del estado en la aplicación se realiza mediante **Redux**. La configuración incluye:

- **Actions**: Se definen en "*/redux/actions"* y se utilizan para despachar cambios al estado global.
- **Reducers**: Se encuentran en "*/redux/reducers"* y se encargan de actualizar el estado en función de las acciones.

- **Store**: Configurado en "*/redux/store"*, combina los reducers y aplica middlewares necesarios como thunk para manejar operaciones asincrónicas.

El estado global maneja información clave como la lista de productos y detalles del usuario. Se emplea el hook *useSelector* para acceder al estado y *useDispatch* para despachar acciones desde los componentes.

Rutas y Navegación

La navegación en la aplicación se gestiona utilizando **React Router**. La configuración de rutas se realiza en el componente principal *App.js* y permite el enrutamiento entre las diferentes páginas:

- **/login**: Página de inicio de sesión.
- /**register**: Página para registro de usuarios
- **/admin**: Página principal para administradores.
- **/vendor**: Página principal para vendedores.

Se utilizan componentes *Route* y *Routes* de React Router para definir y manejar las rutas.

Estilos y Diseño

La metodología de diseño sigue el enfoque **mobile-first**. Se utilizan estilos definidos en archivos CSS para asegurar la compatibilidad y la capacidad de respuesta en dispositivos móviles y de escritorio.

- **CSS Variables**: Se emplean para definir una paleta de colores profesional y mantener la coherencia en toda la aplicación.

4. Backend

4.1. Estructura del Proyecto

La estructura del proyecto backend está organizada de la siguiente manera:

Backend/

config/
config.js
database.js

controllers/
adminController.js

authController.js
productController.js

middleware/
authMiddleware.js
authorizeRole.js
errorMiddleware.js

migrations/

models/
product.js
user.js

routes/
adminRoutes.js
authRoutes.js
productRoutes.js

validators/
.env
index.js

4.2. Autenticación y Autorización

- **Sistema de Autenticación:** Utiliza JSON Web Tokens (JWT) para autenticar a los usuarios. El token se genera durante el inicio de sesión y se incluye en los encabezados de las solicitudes protegidas por medio de cookies.
- **Roles de Usuario:** Se implementa un sistema de roles que distingue entre *admin* y user. Los roles se verifican para determinar el acceso a ciertas rutas o acciones.
- **Manejo de Tokens:** Los tokens se envían en los encabezados *Authorization* de las solicitudes. La verificación del token se realiza mediante middleware antes de procesar las solicitudes.

4.3. Dependencias y Librerías

- **Express:** Framework para la creación de servidores y manejo de rutas.
- **Sequelize:** ORM para gestionar la base de datos y los modelos.
- **dotenv:** Gestión de variables de entorno.
- **cors:** Middleware para habilitar CORS.
- **helmet:** Middleware para mejorar la seguridad.
- **body-parser:** Middleware para analizar el cuerpo de las solicitudes.

- **swagger-ui-express:** Herramienta para documentar la API con Swagger.
- **cookie-parser:** Middleware para analizar cookies.

5. Bases de datos

5.1. Esquema de la Base de Datos

Diagrama de Entidad-Relación (ERD)

Un diagrama ERD visualiza cómo se relacionan las tablas dentro de la base de datos. Aquí está una descripción de las tablas principales y sus relaciones:

- **Usuarios (*Users*)**
  - **Campos:**
    - *id*: Identificador único (clave primaria).
    - *email*: Correo de usuario único.
    - *password*: Contraseña del usuario (almacenada de manera segura).
    - *role*: Rol del usuario (*admin* o *user*).

- **Productos (*Products*)**
  - **Campos:**
    - *id*: Identificador único (clave primaria).
    - *name*: Nombre del producto.
    - *price*: Precio del producto.
    - *quantity*: Cantidad en inventario.
    - *SKU*: Código único del producto.
    - *vendorId*: ID del usuario que vende el producto (clave foránea que referencia a *Users*).

Relaciones:

- **Usuarios a Productos**: Un usuario puede tener múltiples productos (relación uno a muchos).
  - *Users.id* (clave primaria) → *Products*.*vendorId* (clave foránea).


productos
id int8
nombre text
descripcion text
precio numeric
cantidad_disponible int4
id_vendedor int8
fecha_creacion timestamptz
usuarios
id int8
nombre text
email text
contraseña text
rol text
fecha_creacion timestamptz