Documentación Marketplace Challenge

Objetivo

El propósito de esta aplicación es desarrollar un **Marketplace** funcional que permita a vendedores y compradores interactuar a través de una interfaz de usuario (UI) intuitiva y una API robusta.

La aplicación incluye funcionalidades como creación de cuentas, registro y administración de productos, búsqueda de productos, y funcionalidades exclusivas para administradores, cumpliendo con los principios SOLID y buenas prácticas de desarrollo.

Tecnologías Utilizadas

- **Frontend**:
    - **Framework**: Next.js
    - **Estilizado**: NextUI y Tailwind CSS
    - **Gestión de Estado**: Redux
    - **Lenguaje**: TypeScript
- **Backend**:
    - **Framework**: Nest.js
    - **Base de Datos**: MongoDB
    - **Autenticación**: JWT (JSON Web Tokens)
- **Infraestructura**:
    - **Almacenamiento de Imágenes**: AWS S3 Bucket
    - **Despliegue**: Aplicación alojada en un servidor público.

Historias de Usuario y Criterios de Aceptación

1. Creación de Cuenta (Vendedor)

- **Historia**:
    - Como vendedor, quiero crear una cuenta usando mi email para acceder al Marketplace.
- **Criterios de Aceptación**:
    - Modal para ingresar correo, contraseña y validación de contraseña.
    - Error si la contraseña y su validación no coinciden.
    - Error si el correo ya está registrado.
    - Cierre del modal si la creación de la cuenta es exitosa.

2. Registro de Productos

- **Historia**:
    - Como vendedor, quiero registrar mis productos y asignarles precios para que los clientes puedan visualizarlos.
- **Criterios de Aceptación**:
    - Formulario para agregar nombre, SKU, cantidad y precio.
    - Error si falta algún atributo obligatorio.
    - Solo usuarios autenticados pueden registrar productos.

3. Administración de Productos

- **Historia**:
    - Como vendedor, quiero ver toda la lista de productos registrados para poder administrarlos.
- **Criterios de Aceptación**:
    - Lista accesible solo para vendedores autenticados.
    - Cada vendedor solo ve sus propios productos.

4. Búsqueda de Productos (Comprador)

- **Historia**:
    - Como comprador, quiero buscar productos para agregarlos a mi carrito de compras.
- **Criterios de Aceptación**:
    - Filtros por nombre, SKU y/o rango de precios.

5. Visualización de Productos (Administrador)

- **Historia**:
    - Como administrador, quiero ver todos los productos registrados en el Marketplace.
- **Criterios de Aceptación**:
    - Visualización de todos los productos con filtro por vendedor.
    - Acceso exclusivo para administradores autenticados.

Diseño del Sistema

1. Arquitectura

- **Frontend**:
    - App Router de Next.js con manejo de rutas protegidas.
    - Interfaz desarrollada con NextUI y Tailwind CSS.
    - Redux para manejo del estado global (autenticación y carrito de compras).

- **Backend**:
    - Arquitectura basada en Nest.js con principios SOLID.
    - Autenticación basada en JWT con diferentes roles (vendedor, comprador, administrador).
    - Controladores y servicios modulares para cada entidad (usuarios, productos, pedidos).
    - Conexión a MongoDB para almacenamiento de datos.

2. Base de Datos (MongoDB)

- **Colecciones**:

Users:

```
{
  "_id": "ObjectId",
  "email": "string",
  "password": "string",
  "imageUrl": "string",
  "roles": "Array<string>",
  "createdAt": "Date",
  "updatedAt": "Date"
}
```

    -

Products:

```
{
  "_id": "ObjectId",
  "name": "string",
  "sku": "string",
  "price": "number",
  "stock": "number",
  "vendorId": "ObjectId",
  "imageUrl": "string",
  "cont": "number",
  "createdAt": "Date",
  "updatedAt": "Date"
}
```

    -

Flujos Principales

1. Autenticación

- **Registro**:
  - POST `/auth/register`: Crea un nuevo usuario.
- **Login**:
  - POST `/auth/login`: Genera un token JWT.
- **Middleware**:
  - `AuthGuard` para proteger rutas basadas en roles.

2. Productos

- **Registro**:
  - POST `/products`: Permite a vendedores registrar productos.
- **Lista**:
  - GET `/products`: Devuelve productos según filtros (autenticado como comprador/vendedor/administrador).

3. Subida de Imágenes

- Imágenes subidas a AWS S3 mediante un servicio en el backend.

UI/UX

- **Responsividad**: Diseño optimizado para dispositivos móviles y de escritorio.
- **UX Mejorada**:
  - Indicaciones claras de errores en formularios.
  - Feedback visual para operaciones exitosas o fallidas.

Requisitos de Evaluación

1. **Cumplimiento de Historias de Usuario**:
   - Validación de todos los criterios de aceptación.
2. **Calidad del Código**:
   - Código limpio, modular y reutilizable.
3. **Patrones de Diseño**:
   - Uso de principios SOLID y separación de responsabilidades.
4. **Performance**:

    - Optimización en consultas y uso eficiente del almacenamiento en AWS S3.
5. **Buenas Prácticas**:
    - Uso de TypeScript para tipado fuerte.
    - Despliegue en un entorno público.

Despliegue

- **Frontend**: Desplegado en **Vercel**.
- **Backend**: Desplegado en **Railway**.
- **Imágenes**: Almacenadas en **AWS S3 Bucket**.