Documentación Técnica API REST

Miguel Antonio Amaya Hernández

Introducción:

La aplicación “Gestión de Productos” permite a los usuarios autenticados realizar operaciones básicas (Crear, Listar y Eliminar) sobre productos. Las principales características son:

- **Gestión de productos**: Los usuarios pueden agregar, listar y eliminar productos. Cada producto incluye un nombre, descripción, precio y categoría.
- **Autenticación**: Incluye inicio de sesión y registro de usuarios. Solo los usuarios autenticados pueden acceder a las funcionalidades.
- **Búsqueda y filtrado**: Los productos se pueden buscar por nombre o categoría. Además, se organizan automáticamente según su categoría para facilitar su visualización.
- **Interfaz moderna y responsiva**: La aplicación presenta una interfaz de usuario intuitiva que permite una navegación eficiente y agradable, con diseño basado en componentes reutilizables.
- **Notificaciones visuales**: Utiliza alertas visuales (con SweetAlert2) para informar al usuario sobre el éxito o error de sus acciones.
- **Card de productos**: Cada producto se visualiza en una tarjeta (card) que incluye su información clave, con un botón para eliminarlo.

Tecnologías utilizadas

- **Frontend**:
  - ✓ **React**: Biblioteca JavaScript para crear interfaces de usuario dinámicas y responsivas.
  - ✓ **TailwindCSS**: Framework CSS para estilizar los componentes de manera eficiente.
  - ✓ **SweetAlert2**: Biblioteca para mostrar alertas personalizables y atractivas.
  - ✓ **Axios**: Cliente HTTP para interactuar con el backend.
- **Backend**:
  - ✓ **Spring Boot**: Framework para el desarrollo rápido de aplicaciones web en Java.
  - ✓ **Hibernate**: Framework ORM para la gestión de la base de datos.
  - ✓ **Spring Security**: Módulo para la autenticación y autorización de usuarios.
- **Base de Datos**:
  - ✓ **MySQL**: Sistema de gestión de bases de datos relacional para almacenar usuarios y productos.
- **Integración y despliegue**:
  - ✓ **Maven**: Para la gestión de dependencias del proyecto backend.
  - ✓ **Node.js y npm**: Para la instalación y gestión de dependencias del proyecto frontend.
  - ✓ **GitHub**: Repositorio para el control de versiones y colaboración.

Arquitectura:

Modelo Cliente-Servidor

La aplicación sigue el modelo cliente-servidor, donde el backend (servidor) se encarga de gestionar la lógica del negocio, la validación de datos y las operaciones con la base de datos. El frontend (cliente) consume los servicios del backend mediante APIs REST para mostrar la información y proporcionar interactividad al usuario.

Flujo de Comunicación

1. El cliente (React) realiza solicitudes HTTP hacia el servidor (Spring Boot) a través de endpoints REST.
2. El backend valida las solicitudes, interactúa con la base de datos MySQL y responde con datos estructurados en formato JSON.
3. El frontend interpreta las respuestas y actualiza la interfaz de usuario.

Estructura del Backend con Spring Boot

El backend está construido utilizando "Spring Boot", una tecnología que simplifica el desarrollo de aplicaciones basadas en Java. Su estructura modular sigue los principios de "MVC" (Modelo-Vista-Controlador), organizando el código de la siguiente manera:

1. **Controladores (Controllers)**:
   - ✓ Gestionan las solicitudes HTTP (GET, POST, PUT, DELETE) provenientes del frontend.
   - ✓ Exponen los endpoints REST necesarios para las operaciones con productos y usuarios (e.g., /api/auth/register, /api/productos).
2. **Servicios (Services)**:
   - ✓ Contienen la lógica del negocio.
   - ✓ Manejan operaciones complejas y validaciones antes de interactuar con la base de datos.
3. **Repositorios (Repositories)**:
   - ✓ Interactúan directamente con la base de datos MySQL utilizando JPA (Java Persistence API).
   - ✓ Realizan operaciones CRUD sobre las tablas.

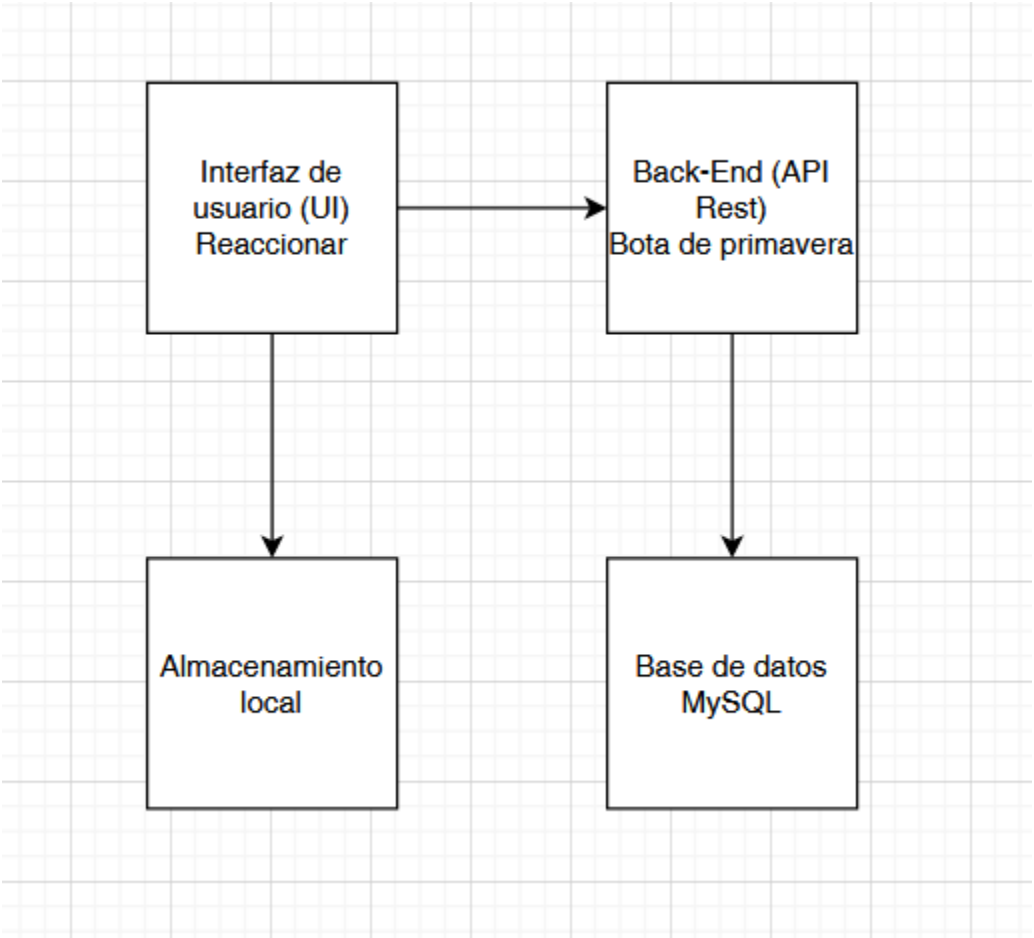
4. **Modelos (Entities)**:
    - ✓ Representan las tablas de la base de datos en forma de clases Java.
    - ✓ Incluyen anotaciones como @Entity para mapear los atributos de las clases a las columnas de la base de datos.
5. **Seguridad**:
    - ✓ Utiliza “Spring Security” para la autenticación y autorización.
    - ✓ Implementa autenticación basada en “tokens JWT “para proteger los endpoints.

Estructura del Frontend con React

El frontend está construido con **React**, aprovechando su flexibilidad para crear componentes reutilizables y una interfaz interactiva.

1. **Componentes**:
    - ✓ Componentes individuales como Login, Registro, ProductosList, ProductoCard.
    - ✓ Cada componente se enfoca en una tarea específica, promoviendo la reutilización.
2. **Estado Global**:
    - ✓ Utiliza “React Hooks” (useState, useEffect, useCallback) para manejar el estado y los ciclos de vida de los componentes.
3. **Comunicación con el Backend**:
    - ✓ Usa “Axios” para realizar solicitudes HTTP a las APIs del backend.
    - ✓ Implementa un archivo api.js para configurar Axios, centralizando la lógica de conexión.
4. **Estilización**:
    - ✓ La interfaz está estilizada con “TailwindCSS”, logrando una experiencia visual moderna y responsiva.

Diagrama de componentes.



Decisiones de diseño:

Elección de Tecnologías

Spring Boot

- **Simplicidad**: Ideal para construir APIs REST rápidas y robustas.
- **Ecosistema**: Ofrece integración nativa con JPA, Spring Security y otras librerías.
- **Escalabilidad**: Soporte para arquitecturas microservicio si el proyecto crece.

React

- **Interactividad**: Perfecto para desarrollar una interfaz dinámica y moderna.
- **Componentes Reutilizables**: Mejora la modularidad y la mantenibilidad.
- **Ecosistema Amplio**: Compatibilidad con herramientas como Axios y TailwindCSS.

MySQL

- **Estabilidad**: Base de datos relacional confiable y ampliamente utilizada.
- **Compatibilidad**: Integración sencilla con JPA/Hibernate.

JWT para Autenticación

- **Seguridad**: Genera tokens únicos para cada sesión, protegiendo los endpoints.
- **Estandarización**: Es un estándar ampliamente soportado.

Criterios de Validaciones y Seguridad

Validaciones

- Uso de **Bean Validation** en Spring Boot para validar campos como email, nombre de usuario, y contraseña.
- Validaciones en el frontend para mejorar la experiencia del usuario (e.g., longitud mínima de contraseñas, formato de correo).

Seguridad

- Autenticación basada en JWT:
  - ❖ Cada usuario recibe un token al iniciar sesión.
  - ❖ Los tokens protegen los endpoints sensibles.
- **Protección contra CORS**:
  - ❖ Configurado en el backend para aceptar solicitudes solo desde el dominio del frontend.

- **Validaciones en el Backend**:
    - ❖ Verifica campos obligatorios y formatos.
    - ❖ Asegura que las operaciones como eliminación o edición sean realizadas por usuarios autorizados.

Guía de instalación y ejecución:

- **Requisitos previos**:
  - JDK 11+.
  - Node.js y npm.
  - MySQL 8.1 o superior.
  - Maven (si no se incluye en el entorno).
- **Instrucciones**:
  - Clonar el repositorio.
  - Configurar la base de datos MySQL y cambiar el archivo application.properties o application.yml con las credenciales correctas.
  - Construir el backend con mvn clean install.
  - Iniciar el backend con mvn spring-boot:run.
  - En el directorio del frontend, ejecutar npm install y luego npm start.