DOCUMENTACION TECNICA

William Alexander Ventura Gonz□lez

Introducción

Se desarrolló una aplicación de gestión de productos, combinando herramientas modernas y prácticas para construir un sistema funcional. El backend fue implementado con Spring Boot, ofreciendo una API REST que permite realizar operaciones básicas como crear, leer, actualizar y eliminar productos, mientras que el frontend se construyó con React, proporcionando una interfaz sencilla y amigable para los usuarios.

La autenticación y autorización fueron aspectos clave en el desarrollo, utilizando JSON Web Tokens (JWT) para garantizar que solo usuarios autenticados puedan realizar ciertas acciones dentro del sistema. Este enfoque ayuda a mantener la seguridad en las operaciones y protege las rutas sensibles.

El desarrollo se llevó a cabo dividiendo el backend y el frontend como módulos separados, lo que simplifica el mantenimiento y facilita futuras actualizaciones. Librerías como Axios fueron utilizadas para gestionar las solicitudes al servidor desde el cliente, y SweetAlert2 se implementó para proporcionar mensajes interactivos que mejoren la experiencia del usuario.

En este documento se describen aspectos relevantes de la arquitectura del sistema, las decisiones técnicas tomadas durante el desarrollo, y las instrucciones necesarias para instalar y poner en marcha la aplicación.

Arquitectura de la Aplicación

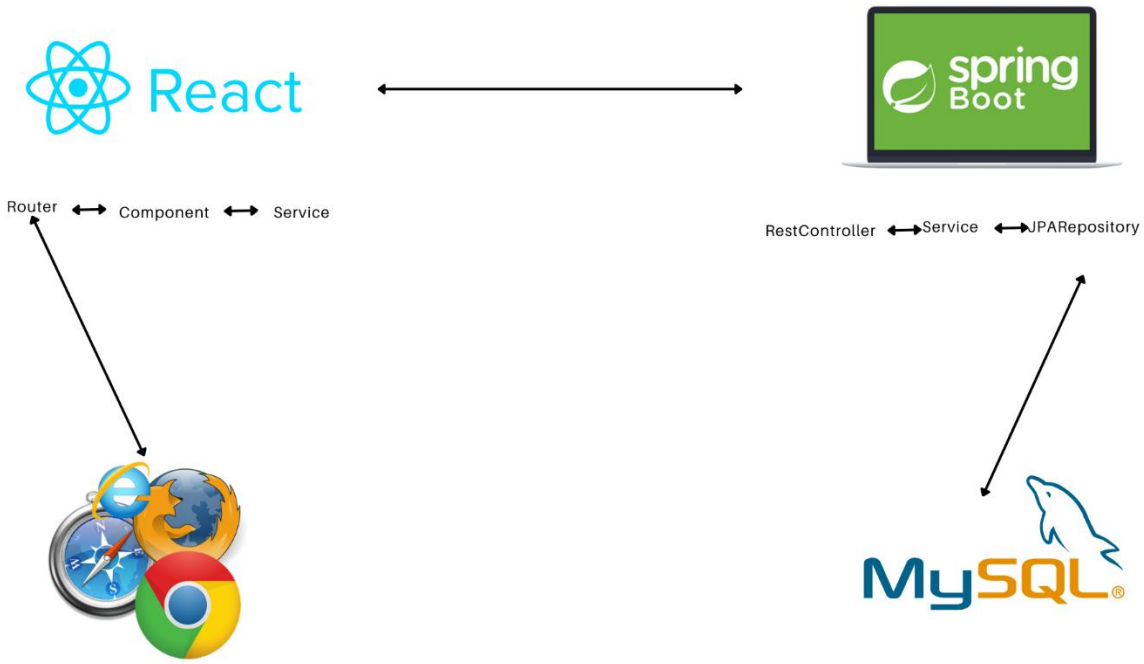
La aplicación se construyó siguiendo el patrón cliente-servidor, con una separación clara entre el frontend, el backend y la base de datos:

- **Frontend (React):**
    - Es responsable de la interfaz de usuario.
    - Permite a los usuarios interactuar con el sistema mediante formularios para el inicio de sesión, el registro de usuarios y productos, y realizar operaciones CRUD en los productos.
    - Utiliza Axios para realizar peticiones al backend y SweetAlert2 para mostrar mensajes interactivos.
- **Backend (Spring Boot):**
    - Maneja la lógica del servidor, como la autenticación de usuarios, validación de datos y procesamiento de las solicitudes del cliente.

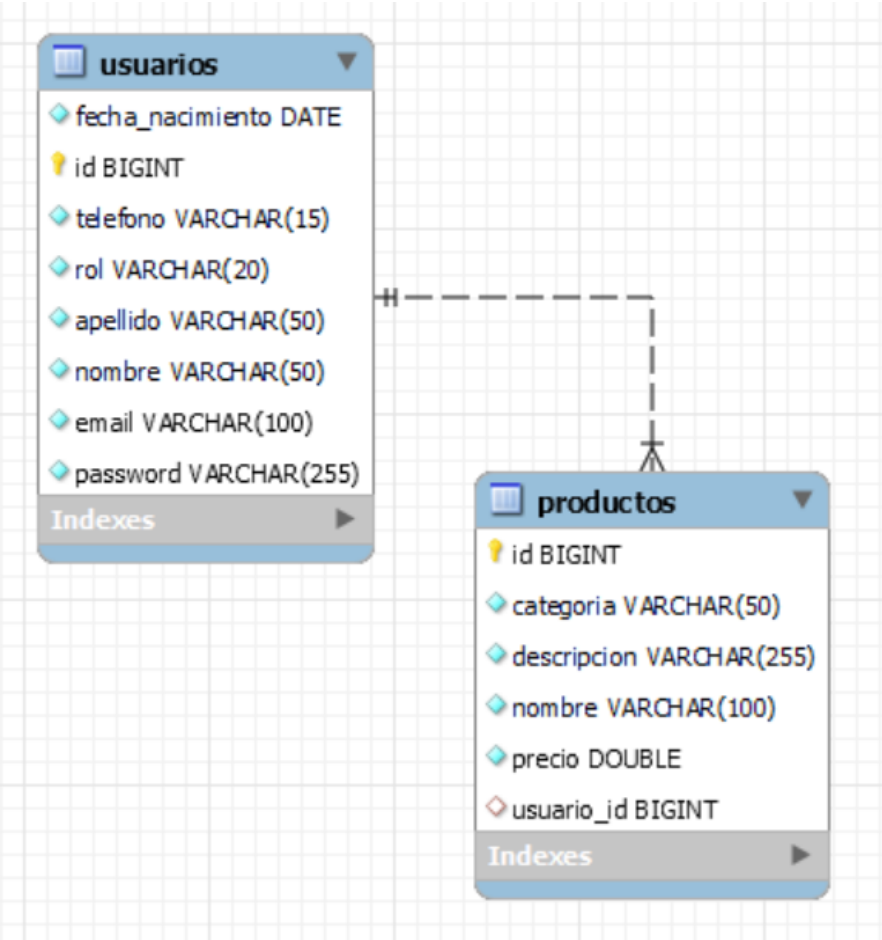
- Implementa JWT (JSON Web Tokens) para proteger rutas y garantizar un acceso seguro.
- Expone una API REST para que el frontend interactúe con el sistema.

- **Base de Datos (MySQL):**
  - Almacena toda la información relacionada con los usuarios y productos.
  - Sigue un esquema relacional, con tablas estructuradas para facilitar la gestión y consulta de datos.

Diagrama de Arquitectura



Esquema de la base de datos



Decisiones de Diseño

Backend (Spring Boot)

1. **Controladores RESTful:**
    - Se implementaron controladores para manejar los endpoints del sistema, organizados por módulos (usuarios, productos).
    - Cada controlador utiliza anotaciones como @RestController, @PostMapping, @GetMapping, y @DeleteMapping para definir rutas claras y específicas.
    - Las rutas se estructuraron de manera que representen operaciones específicas, por ejemplo:
        - /api/usuarios/login para iniciar sesión.
        - /api/productos para gestionar los productos.

2. **Herencia y Abstracción:**
    - Se diseñó una clase abstracta Persona, que incluye atributos comunes como nombre, apellido y correo.
    - Las clases concretas, como Usuario, heredan de Persona, lo que permitió reducir la duplicación de código y mantener un diseño limpio.
3. **Servicios, Repositorios y DTO:**
    - La lógica de negocio se encapsuló en servicios (@Service), separando el acceso a los datos y la lógica de las operaciones de los controladores.
    - Se utilizaron repositorios (@Repository) que extienden de JpaRepository para realizar las operaciones con la base de datos de manera eficiente.
    - Los DTOs (Data Transfer Objects) fueron utilizados para manejar la transferencia de datos entre el frontend y el backend, garantizando seguridad y claridad.
4. **Roles y Enum:**
    - Se definió un enumerador (Enum) para los roles de los usuarios (ADMIN, USER), lo que facilita la gestión de permisos y asegura consistencia en el sistema.
5. **JWT para Autenticación:**
    - Se utilizó JSON Web Tokens (JWT) para autenticar y autorizar a los usuarios:
        - Al iniciar sesión, se genera un token único que contiene la información del usuario.
        - Este token es enviado en cada solicitud protegida a través del encabezado Authorization.
    - El middleware de seguridad valida el token antes de procesar la solicitud.
6. **Validaciones:**
    - Las entradas del usuario se validaron utilizando anotaciones de Hibernate Validator como:
        - @NotNull para campos obligatorios.
        - @Email para correos electrónicos válidos.
        - @Size para límites de caracteres.
    - Esto asegura la integridad de los datos antes de procesarlos.

7. **Gestión de Errores:**
    - Se implementó un manejo centralizado de excepciones utilizando la anotación @ControllerAdvice.
    - Esto permite capturar y gestionar errores comunes de manera uniforme, enviando respuestas claras al frontend.

Frontend (React)

1. **Rutas Protegidas:**
    - Se implementaron rutas protegidas verificando si el usuario está autenticado mediante el estado de autenticación (isLoggedIn) y el token JWT almacenado en localStorage.
    - Si el token no es válido, se redirige al usuario a la página de inicio de sesión.
2. **Componentes Reutilizables:**
    - Se diseñaron componentes genéricos como:
        - Formularios reutilizables para el registro e inicio de sesión.
        - Mensajes de confirmación y alerta utilizando SweetAlert2.
3. **Interfaz de Usuario:**
    - Diseño intuitivo y responsivo utilizando CSS modular, asegurando que la aplicación se visualice correctamente tanto en dispositivos móviles como en escritorio.
    - Los formularios incluyen validaciones visuales que indican al usuario si un campo es obligatorio o contiene datos inválidos.
4. **Alertas Interactivas:**
    - Se utilizó SweetAlert2 para mostrar mensajes claros de éxito, error o advertencia, mejorando la experiencia del usuario.
5. **Conexión con el Backend:**
    - La comunicación con el backend se realizó a través de Axios.
    - Se incluyeron encabezados de autorización en las peticiones protegidas para enviar el token JWT.

Seguridad

1. **Middleware de Seguridad:**

- Se implementaron filtros de seguridad que interceptan solicitudes protegidas, validando el token JWT.
- Solo los usuarios con roles válidos tienen acceso a ciertas rutas del backend.

2. **Encriptación de Contraseñas:**
   - Las contraseñas se almacenan en la base de datos de forma segura utilizando el algoritmo BCrypt.
3. **Roles Basados en Enum:**
   - Los roles definidos en el backend permiten restringir funcionalidades dependiendo del nivel de acceso del usuario.

Guía de Instalación Rápida

Requisitos Previos

Asegúrate de tener instalados los siguientes programas en tu máquina:

1. **Java Development Kit (JDK)** - Versión 11 o superior.
2. **Node.js y npm** - Instalar la versión recomendada de Node.js.
3. **MySQL** - Para la base de datos.
4. **Maven** - Para manejar las dependencias del backend.

1. Clonar el Repositorio

Clonar el repositorio en tu máquina local usando el siguiente comando:

git clone https://github.com/WilliamV10/PruebaTecnica/

Luego, acceder al directorio clonado:

cd PRUEBATECNICA

2. Configuración del Backend (Spring Boot)

Paso 2.1: Configurar la Base de Datos

1. **Crear una base de datos en MySQL**:
   - Abrir cliente de MySQL o usar la terminal y ejecuta:

CREATE DATABASE productos;

- Crear un usuario si es necesario:
- CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'contraseña';
- GRANT ALL PRIVILEGES ON productos_db.* TO 'usuario'@'localhost';

FLUSH PRIVILEGES;

2. **Configurar application.properties**:
    - Ve al directorio productos/src/main/resources/.
    - Crea un archivo application.properties (o edita si ya existe).
    - Configura las credenciales de la base de datos:
    - spring.datasource.url=jdbc:mysql://localhost:3306/productos
    - spring.datasource.username=usuario
    - spring.datasource.password=contraseña
    - spring.jpa.hibernate.ddl-auto=update
    - spring.jpa.show-sql=true

**cambiar usuario y contraseña por las tuyas.

Paso 2.2: Instalar Dependencias

Entrar al directorio del backend:

cd productos

Ejecutar Maven Wrapper para descargar las dependencias:

./mvnw install

Paso 2.3: Iniciar el Servidor

Levanta el servidor backend ejecutando:

./mvnw spring-boot:run

El servidor estará disponible en http://localhost:8080.

3. Configuración del Frontend (React)

Paso 3.1: Instalar Dependencias

Entra al directorio del frontend:

cd front-react

Instala las dependencias necesarias con:

npm install

Paso 3.3: Iniciar el Servidor de Desarrollo

Levantar el servidor del frontend ejecutando:

npm start

Esto abrirá la aplicación React en el navegador en http://localhost:3000.

4. Verificar la Aplicación

1. Abre http://localhost:3000 en tu navegador.
2. Prueba iniciar sesión y realiza operaciones CRUD.

Biblioteca Calculadora

Ejemplos de como importarla y usarla (se debe de copiar el .jar en el proyecto para que maven tenga acceso).

```
<dependency>
<groupId>org.prueba</groupId>
<artifactId>Calculadora</artifactId>
<version>1.0-SNAPSHOT</version>
</dependency>
```

```
import org.prueba.Calculadora;
```

```
System.out.println("Suma"+ Calculadora.suma( a: 2, b: 3));
```