

Documentación de la API RESTful para Gestión de Productos

Introducción

Este documento explica el desarrollo de una API RESTful en Java, utilizando Spring Boot para gestionar productos. Incluye las decisiones de diseño tomadas, cómo se aplicaron los principios y buenas prácticas, y las instrucciones para ejecutar el proyecto.

Decisiones Tomadas

Tecnología

Se eligió Java con Spring Boot como framework para el desarrollo de la API REST debido a su robustez y amplia documentación, lo que facilita la creación de aplicaciones web y APIs. Spring Data JPA se utilizó para gestionar la comunicación con la base de datos, permitiendo realizar operaciones CRUD con un mínimo de código.

Base de Datos

Se decidió usar H2 Database como base de datos en memoria para facilitar las pruebas y el desarrollo. En un entorno de producción, se podría reemplazar H2 por una base de datos como MySQL o PostgreSQL.

Principios y Buenas Prácticas

Principio de Separación de Responsabilidades

Se aplicó el principio de separación de responsabilidades (SoC) dividiendo la lógica del programa en capas: la entidad `Producto`, el repositorio `ProductoRepository` para la lógica de persistencia, y el controlador `ProductoController` para gestionar las peticiones HTTP. Esto mejora la mantenibilidad y escalabilidad del código.

CRUD y Convenciones RESTful

Se implementaron las operaciones CRUD siguiendo las convenciones RESTful: se utilizaron métodos HTTP adecuados como GET para obtener datos, POST para crear nuevos registros, PUT para actualizaciones y DELETE para eliminaciones. Cada operación responde con los códigos de estado HTTP correspondientes (200, 201, 404, etc.).

Validación y Manejo de Errores

Para asegurar la integridad de los datos y mejorar la experiencia de usuario, se implementó un manejo básico de errores. Las respuestas a solicitudes inválidas devuelven un mensaje de error claro y el código de estado HTTP adecuado.

Instrucciones para Ejecutar el Proyecto

Configuración

El proyecto requiere Java instalado. La base de datos H2 se configura automáticamente en memoria, por lo que no se necesitan pasos adicionales para la base de datos.

Ejecución del Proyecto

Instrucciones adaptadas para Visual Studio Code, IntelliJ IDEA y NetBeans

Visual Studio Code

Instalar extensiones:


Asegúrate de tener instaladas las extensiones Java Extension Pack y Spring Boot Extension Pack en Visual Studio Code para soporte de Java y Spring Boot.

Abrir el proyecto:

Abre Visual Studio Code y selecciona File > Open Folder. Navega hasta la carpeta raíz del proyecto y ábrela.

Ejecutar el proyecto:

En la paleta de comandos (Ctrl + Shift + P en Windows/Linux o Cmd + Shift + P en macOS), escribe y selecciona Spring Boot Dashboard.

En el panel de Spring Boot Dashboard, selecciona el proyecto y haz clic en el botón de reproducción () para ejecutarlo.

Alternativamente, abre una terminal en VS Code y ejecuta el comando:

```
./mvnw spring-boot:run
```

o en Windows:

```
mvnw.cmd spring-boot:run
```

Acceder a la API:

La API estará disponible en <http://localhost:8080/productos>.


IntelliJ IDEA

Abrir el proyecto:

En IntelliJ IDEA, selecciona File > Open, y abre la carpeta raíz del proyecto.

Configurar y ejecutar el proyecto:

Si es la primera vez que abres el proyecto, IntelliJ detectará automáticamente que es un proyecto Maven y configurará las dependencias. Acepta las sugerencias que aparecen en la parte inferior de la ventana.

Una vez cargado el proyecto, ve al archivo principal de tu aplicación (CrudApplication) y haz clic en el botón de reproducción () en la esquina superior derecha para ejecutar la aplicación.

Acceder a la API:

La API estará disponible en <http://localhost:8080/productos>.

Pruebas

Para probar la API, se pueden utilizar herramientas como Postman.

- Crear un producto (POST):

URL: `http://localhost:8080/productos`

Body:

```
{  
  "nombre": "Laptop",  
  "descripción": "Laptop de alta gama",  
  "precio": 1500.0,  
  "stock": 10  
}
```

- Obtener todos los productos (GET):

URL: `http://localhost:8080/productos`

- Actualizar un producto (PUT):

URL: `http://localhost:8080/productos/1`

Body:

```
{  
  "precio": 1400.0  
}
```

- Eliminar un producto (DELETE):

URL: `http://localhost:8080/productos/1`