# Proyecto Spring Boot REST Spring Data API CRUD

**Explicación de qué se hizo en el código:**

- **Objetivo:** Crear una aplicación Spring Boot que implemente una API REST para realizar operaciones CRUD (Crear, Leer, Actualizar, Borrar) en una entidad `User`.
- **Componentes Principales:**
  - **Entidad `User`:** Representa la estructura de datos de un usuario y se mapea a una tabla en la base de datos.
  - **Repositorio `UserRepository`:** Interfaz que extiende `JpaRepository`, proporcionando métodos para operaciones CRUD básicas.
  - **Servicio `UserService`:** Contiene la lógica de negocio para manejar usuarios, usando `UserRepository`.
  - **Controlador `UserController`:** Exponiendo los endpoints REST que permiten a los clientes interactuar con la entidad `User`.

**Código más importante:**

**`pom.xml`:**

xml

Copiar código

```
<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>



    <groupId>com.example</groupId>

    <artifactId>SpringRESTAPISpringDataJPACRUD</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <packaging>jar</packaging>
```

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.2.2</version>
</parent>

<properties>
    <java.version>17</java.version>
</properties>

<dependencies>
    <!-- Spring Boot JPA for ORM -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <!-- Spring Boot Web for REST API -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- H2 Database (in-memory) -->
```

```
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>

        <!-- Spring Boot DevTools (optional, for hot reload) -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
        </dependency>

        <!-- Testing Dependencies -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
```

```
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

**Motivo:** Este archivo configura las dependencias necesarias para un proyecto Spring Boot que utiliza JPA para la persistencia de datos y expone una API REST para realizar operaciones CRUD. Incluye la dependencia para Spring Data JPA, Spring Web, y una base de datos en memoria H2 para desarrollo y pruebas.

**Anotaciones extra:**

- **`@Entity`** en la clase `User`: Marca la clase como una entidad JPA, que se mapea a una tabla en la base de datos.
- **`@Table`** en la clase `User` (opcional): Especifica el nombre de la tabla en la base de datos.
- **`@Id`** y **`@GeneratedValue`** en el campo `id` de `User`: Indica que este campo es la clave primaria y su valor se genera automáticamente.
- **`@Repository`** en `UserRepository`: Indica que esta interfaz es un repositorio Spring Data JPA, proporcionando métodos para realizar operaciones de base de datos.
- **`@Service`** en `UserService`: Marca la clase como un servicio que contiene la lógica de negocio.
- **`@RestController`** en `UserController`: Marca la clase como un controlador REST que maneja solicitudes HTTP y produce respuestas en formato JSON.
- **`@RequestMapping`**, **`@GetMapping`**, **`@PostMapping`**, **`@PutMapping`**, **`@DeleteMapping`** en `UserController`: Define los endpoints REST para las operaciones CRUD.