Paso 1: Crear el Proyecto Spring Boot

1. **Crear el proyecto:**
   - Ve a [Spring Initializr](https://start.spring.io/).
   - Configura el proyecto con los siguientes detalles:
     - **Project:** Maven Project
     - **Language:** Java
     - **Spring Boot:** 3.x (la versión más reciente)
     - **Group:** com.example
     - **Artifact:** api-restful
     - **Packaging:** Jar
     - **Java:** 17 (o la versión que prefieras)
   - Agrega las siguientes dependencias:
     - **Spring Web** (para crear APIs RESTful)
     - **Spring Reactive Web** (para WebFlux en el Ejercicio 4)
     - **Spring Boot DevTools** (opcional, para desarrollo)
     - **Lombok** (para simplificar el código)
     - **Spring Boot Starter Test** (para pruebas unitarias)
   - Haz clic en **Generate** para descargar el proyecto.

2. **Importar el proyecto:**
   - Extrae el archivo ZIP descargado.
   - Importa el proyecto en tu IDE favorito (IntelliJ IDEA, Eclipse, etc.).

Paso 2: Configurar el Proyecto

1. **Estructura del proyecto:**

   - Asegúrate de que la estructura del proyecto sea similar a esta:

   ```
   src
   ├── main
   │   ├── java
   │   │   └── com
   │   │       └── example
   │   │           └── apirestful
   │   │               ├── controller
   │   │               ├── model
   │   │               └── ApiRestfulApplication.java
   │   └── resources
   │       ├── application.properties
   │       └── messages.properties
   └── test
       └── java
           └── com
               └── example
                   └── apirestful
                       └── controller
   ```

2. **Configurar `application.properties`:**

- Abre el archivo `src/main/resources/application.properties` y agrega:

  ```properties
  server.port=8080
  spring.messages.basename=messages
  ```

Paso 3: Ejercicio 1 - Crear un Endpoint Simple

1. **Crear el controlador:**
   - Crea una clase `SaludoController` en el paquete `controller`:

     ```java
     package com.example.apirestful.controller;

     import org.springframework.web.bind.annotation.GetMapping;
     import org.springframework.web.bind.annotation.RequestMapping;
     import org.springframework.web.bind.annotation.RestController;

     @RestController
     @RequestMapping("/api")
     public class SaludoController {

         @GetMapping("/saludo")
         public String obtenerSaludo() {
             return "Hola, API RESTful!";
     ```

```
    }
  }
```

2. **Probar el endpoint:**
   - Ejecuta la aplicación (`ApiRestfulApplication.java`).
   - Abre tu navegador o Postman y visita:
     ```
     GET http://localhost:8080/api/saludo
     ```
   - Deberías ver la salida: `"Hola, API RESTful!"`.

Paso 4: Ejercicio 2 - Crear un CRUD Básico de Productos

1. **Crear el modelo `Producto`:**
   - Crea una clase `Producto` en el paquete `model`:
     ```java
     package com.example.apirestful.model;

     import lombok.AllArgsConstructor;
     import lombok.Data;
     import lombok.NoArgsConstructor;

     @Data
     ```

```
@AllArgsConstructor
@NoArgsConstructor
public class Producto {
    private String id;
    private String nombre;
    private double precio;
}
```

2. **Crear el controlador `ProductoController`:**
   - Crea una clase `ProductoController` en el paquete `controller`:

```java
package com.example.apirestful.controller;

import com.example.apirestful.model.Producto;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

@RestController
@RequestMapping("/api/productos")
public class ProductoController {

    private List<Producto> productos = new ArrayList<>();
```

```
    @PostMapping
    public Producto agregarProducto(@RequestBody Producto producto) {
        producto.setId(UUID.randomUUID().toString());
        productos.add(producto);
        return producto;
    }

    @GetMapping
    public List<Producto> listarProductos() {
        return productos;
    }

    @GetMapping("/{id}")
    public Producto obtenerProducto(@PathVariable String id) {
        return productos.stream()
                .filter(p -> p.getId().equals(id))
                .findFirst()
                .orElseThrow(() -> new RuntimeException("Producto no encontrado"));
    }

    @PutMapping("/{id}")
    public Producto actualizarProducto(@PathVariable String id, @RequestBody
Producto producto) {
        Producto productoExistente = obtenerProducto(id);
        productoExistente.setNombre(producto.getNombre());
```

```
        productoExistente.setPrecio(producto.getPrecio());
        return productoExistente;
    }

    @DeleteMapping("/{id}")
    public void eliminarProducto(@PathVariable String id) {
        productos.removeIf(p -> p.getId().equals(id));
    }
}
```

3. **Probar el CRUD:**
   - Usa Postman para probar los endpoints:
     - **POST:** Agrega un producto.
     - **GET:** Lista todos los productos.
     - **GET:** Obtén un producto por ID.
     - **PUT:** Actualiza un producto.
     - **DELETE:** Elimina un producto.

Paso 5: Ejercicio 3 - Implementar Internacionalización (i18n)

1. **Crear archivos de mensajes:**
   - Crea los archivos `messages.properties`, `messages_es.properties` y `messages_en.properties` en `src/main/resources`:

- `messages_es.properties`:

  ```properties
  saludo=¡Hola, API RESTful en Español!
  ```

- `messages_en.properties`:

  ```properties
  saludo=Hello, RESTful API in English!
  ```

2. **Modificar el controlador `SaludoController`:**

   - Actualiza la clase `SaludoController`:

   ```java
   package com.example.apirestful.controller;

   import org.springframework.beans.factory.annotation.Autowired;
   import org.springframework.context.MessageSource;
   import org.springframework.web.bind.annotation.GetMapping;
   import org.springframework.web.bind.annotation.RequestHeader;
   import org.springframework.web.bind.annotation.RequestMapping;
   import org.springframework.web.bind.annotation.RestController;

   import java.util.Locale;

   @RestController
   @RequestMapping("/api")
   public class SaludoController {
   ```

```
    @Autowired
    private MessageSource messageSource;

    @GetMapping("/saludo")
    public String obtenerSaludo(@RequestHeader(name = "Accept-Language", required = false) Locale locale) {
        return messageSource.getMessage("saludo", null, locale);
    }
}
```

3. **Probar la internacionalización:**
   - Visita:
   ```
   GET http://localhost:8080/api/saludo?lang=es
   GET http://localhost:8080/api/saludo?lang=en
   ```
   - Deberías ver los mensajes en español e inglés respectivamente.

Paso 6: Ejercicio 4 - Crear un Endpoint Reactivo con WebFlux

1. **Modificar el controlador `ProductoController`:**
   - Actualiza la clase `ProductoController` para usar `Flux`:

```java
package com.example.apirestful.controller;

import com.example.apirestful.model.Producto;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Flux;

@RestController
@RequestMapping("/api/productos")
public class ProductoController {

    @GetMapping
    public Flux<Producto> listarProductos() {
        return Flux.just(
            new Producto("1", "Laptop", 1200.0),
            new Producto("2", "Mouse", 25.0),
            new Producto("3", "Teclado", 45.0)
        );
    }
}
```

2. **Probar el endpoint reactivo:**
   - Visita:

```

GET http://localhost:8080/api/productos

```

- Deberías ver un JSON con los productos en formato reactivo.

Paso 7: Ejercicio 5 - Implementar Pruebas con StepVerifier

1. **Crear la prueba `ProductoControllerTest`:**

- Crea una clase `ProductoControllerTest` en el paquete `controller` dentro de `src/test/java`:

```java

package com.example.apirestful.controller;

import com.example.apirestful.model.Producto;

import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.context.SpringBootTest;

import reactor.core.publisher.Flux;

import reactor.test.StepVerifier;

@SpringBootTest

public class ProductoControllerTest {

    @Autowired
```

```
    private ProductoController productoController;

    @Test
    public void testListaProductos() {
      Flux<Producto> productos = productoController.listarProductos();

      StepVerifier.create(productos)
        .expectNextMatches(p -> p.getNombre().equals("Laptop"))
        .expectNextMatches(p -> p.getNombre().equals("Mouse"))
        .expectNextMatches(p -> p.getNombre().equals("Teclado"))
        .verifyComplete();
    }
  }
```

2. **Ejecutar la prueba:**
   - Ejecuta la prueba con JUnit.
   - La prueba debería pasar correctamente.
