Documentación del Proyecto: Clínica Odontológica

Índice

Descripción del Proyecto

La aplicación **Clínica Odontológica** es un sistema de gestión para una clínica odontológica. Permite la administración de odontólogos, pacientes y turnos. La aplicación está desarrollada en Java utilizando Spring Boot y se sigue un enfoque de arquitectura de microservicios.

Estructura del Proyecto

El proyecto está organizado en los siguientes paquetes:

- `controller`: Contiene los controladores REST para manejar las solicitudes HTTP.
- `dto`: Contiene los objetos de transferencia de datos (DTO) para las solicitudes y respuestas.
  - `entrada`: DTOs para datos de entrada.
  - `salida`: DTOs para datos de salida.
- `entity`: Contiene las entidades JPA que representan las tablas de la base de datos.
- `exceptions`: Contiene las clases para el manejo de excepciones personalizadas.
- `repository`: Contiene los repositorios JPA para interactuar con la base de datos.
- `service`: Contiene las interfaces de los servicios.
- `service.impl`: Contiene las implementaciones de los servicios.
- `utils`: Contiene clases utilitarias.

Requisitos

Para ejecutar este proyecto, necesitas tener instalados los siguientes componentes:

- Java 11 o superior
- Maven 3.6.0 o superior
- MySQL 5.7 o superior (puede usarse H2 para pruebas en memoria)

Configuración del Entorno

Base de Datos

1. Configura una base de datos MySQL. Crea una base de datos llamada `clinica_odontologica`.
2. Actualiza el archivo `application.properties` con las credenciales de tu base de datos MySQL:

```
properties
Copy code
spring.datasource.url=jdbc:mysql://localhost:3306/clinica_odontologica
spring.datasource.username=tu_usuario
spring.datasource.password=tu_contraseña
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.h2.console.enabled=true
```

Maven

Ejecuta el siguiente comando para compilar y descargar las dependencias del proyecto:

```
bash
Copy code
mvn clean install
```

Ejecutar la Aplicación

Para ejecutar la aplicación, usa el siguiente comando:

```
bash
Copy code
mvn spring-boot:run
```

La aplicación estará disponible en `http://localhost:8080`.

API Endpoints

Odontólogos

- **Registrar Odontólogo**
  - **POST** `/odontologos/registrar`
  - **Request Body**: `OdontologoEntradaDto`
  - **Response**: `OdontologoSalidaDto`
- **Listar Odontólogos**
  - **GET** `/odontologos/listar`
  - **Response**: `List<OdontologoSalidaDto>`
- **Buscar Odontólogo por ID**
  - **GET** `/odontologos/{id}`
  - **Response**: `OdontologoSalidaDto`
- **Actualizar Odontólogo**
  - **PUT** `/odontologos/actualizar/{id}`
  - **Request Body**: `OdontologoEntradaDto`
  - **Response**: `OdontologoSalidaDto`
- **Eliminar Odontólogo**
  - **DELETE** `/odontologos/eliminar`

    - **Request Param**: `id`
    - **Response**: `String`

Pacientes

- **Registrar Paciente**
    - **POST** `/pacientes/registrar`
    - **Request Body**: `PacienteEntradaDto`
    - **Response**: `PacienteSalidaDto`
- **Listar Pacientes**
    - **GET** `/pacientes/listar`
    - **Response**: `List<PacienteSalidaDto>`
- **Buscar Paciente por ID**
    - **GET** `/pacientes/{id}`
    - **Response**: `PacienteSalidaDto`
- **Actualizar Paciente**
    - **PUT** `/pacientes/actualizar/{id}`
    - **Request Body**: `PacienteEntradaDto`
    - **Response**: `PacienteSalidaDto`
- **Eliminar Paciente**
    - **DELETE** `/pacientes/eliminar`
    - **Request Param**: `id`
    - **Response**: `String`

Turnos

- **Registrar Turno**
    - **POST** `/turnos/registrar`
    - **Request Body**: `TurnoEntradaDto`
    - **Response**: `TurnoSalidaDto`
- **Listar Turnos**
    - **GET** `/turnos/listar`
    - **Response**: `List<TurnoSalidaDto>`
- **Buscar Turno por ID**
    - **GET** `/turnos/{id}`
    - **Response**: `TurnoSalidaDto`
- **Actualizar Turno**
    - **PUT** `/turnos/actualizar/{id}`
    - **Request Body**: `TurnoEntradaDto`
    - **Response**: `TurnoSalidaDto`
- **Eliminar Turno**
    - **DELETE** `/turnos/eliminar`
    - **Request Param**: `id`
    - **Response**: `String`

Manejo de Excepciones

Las excepciones se manejan de forma centralizada mediante el uso de un controlador global de excepciones (`GlobalExceptionHandler`).

Ejemplos de Excepciones:

- `NotFoundException`: Se lanza cuando un recurso no se encuentra.
- `BadRequestException`: Se lanza cuando hay un error en la solicitud.

Pruebas

El proyecto incluye pruebas unitarias para los servicios, utilizando JUnit y Mockito.

Ejecución de Pruebas

Para ejecutar las pruebas, usa el siguiente comando:

```
bash
Copy code
mvn test
```

Agradecimientos

Mil gracias para la profesora que nos dio una enseñanza de enfoque práctico para un problema real.

Agradecimientos especiales a Daniel, Fabio, Yessy, Lina, Carlos por sus valiosos aportes.