Ndayambaje Virgile

1. Introduction

In this lab, we implemented a hospital management system using Spring Boot and Spring Data JPA. The primary goals were to understand the repository pattern, create repositories for data access, and utilize query methods for CRUD operations and custom queries.

2. Repository Pattern

2.1 Overview

The repository pattern is a design pattern used to manage data access and abstraction. It provides an abstraction layer between the data access layer and the business logic, helping to keep the system organized and maintainable.

2.2 Benefits

- **Abstraction**: It hides the complexity of data access and provides a simple API for data operations.
- **Separation of Concerns**: Data access logic is separated from business logic, enhancing maintainability.
- **Testability**: Repositories can be mocked for testing, ensuring isolated unit tests.
- **Flexibility**: Changes in data access implementation do not impact the business logic.

3. Implemented Repositories

3.1 Repository Interfaces

1. **DepartmentRepository**
    - **Methods**:
        - `List<Department> findByBuilding(String building);`
        - `List<Department> findByDirectorSurname(String surname);`
        - `List<Department> findByNameContaining(String keyword);`
2. **DoctorRepository**

Methods:

- `List<Doctor> findBySpecialty(String specialty);`
- `List<Doctor> findByFirstName(String firstName);`
- `List<Doctor> findBySurnameStartingWith(String prefix);`

NurseRepository

- **Methods**:
  - `List<Nurse> findByDepartmentCode(String departmentCode);`
  - `List<Nurse> findByRotation(String rotation);`
  - `List<Nurse> findBySalaryGreaterThan(float salary);`

PatientRepository

- **Methods**:
  - `List<Patient> findBySurname(String surname);`
  - `List<Patient> findByAddress(String address);`
  - `List<Patient> findByTelephoneNumber(String telephoneNumber);`

WardRepository

- **Methods**:
  - `List<Ward> findByDepartmentCode(String departmentCode);`
  - `List<Ward> findByNumberOfBedsGreaterThan(int numberOfBeds);`
  - `List<Ward> findBySupervisorSurname(String surname);`

4. Controller Implementation

The `HospitalSystemController` provides endpoints for managing the entities and performing CRUD operations.

4.1 Department Endpoints

- **Add Department**: `POST /api/departments`
- **Get Departments by Building**: `GET /api/departments/building/{building}`
- **Get Departments by Director's Surname**: `GET /api/departments/director-surname/{surname}`
- **Get Departments by Name Containing**: `GET /api/departments/name-contains/{keyword}`

4.2 Doctor Endpoints

- **Add Doctor**: `POST /api/doctors`
- **Get Doctors by Specialty**: `GET /api/doctors/specialty/{specialty}`
- **Get Doctors by First Name**: `GET /api/doctors/first-name/{firstName}`
- **Get Doctors by Surname Starting With**: `GET /api/doctors/surname-starts-with/{prefix}`

4.3 Nurse Endpoints

- **Add Nurse**: `POST /api/nurses`
- **Get Nurses by Department**: `GET /api/nurses/department/{departmentCode}`
- **Get Nurses by Rotation**: `GET /api/nurses/rotation/{rotation}`
- **Get Nurses by Salary Greater Than**: `GET /api/nurses/salary-greater-than/{salary}`

4.4 Patient Endpoints

- **Add Patient**: `POST /api/patients`
- **Get Patients by Surname**: `GET /api/patients/surname/{surname}`
- **Get Patients by Address**: `GET /api/patients/address/{address}`
- **Get Patients by Telephone Number**: `GET /api/patients/telephone/{telephoneNumber}`

4.5 Ward Endpoints

- **Add Ward**: `POST /api/wards`
- **Get Wards by Department**: `GET /api/wards/department/{departmentCode}`
- **Get Wards by Number of Beds Greater Than**: `GET /api/wards/beds-greater-than/{numberOfBeds}`
- **Get Wards by Supervisor's Surname**: `GET /api/wards/supervisor-surname/{surname}`

4.6 Hospitalization Endpoints

- **Add Hospitalization**: `POST /api/hospitalizations`
- **Get Hospitalizations by Patient Number**: `GET /api/hospitalizations/patient/{patientNumber}`
- **Get Hospitalizations by Doctor Employee Number**: `GET /api/hospitalizations/doctor/{doctorEmployeeNumber}`
- **Get Hospitalizations by Diagnosis**: `GET /api/hospitalizations/diagnosis/{diagnosis}`

5. Testing the Controller

Tests were created to ensure that the controller endpoints function correctly.

5.1 Test Class: `HospitalSystemControllerTest`

- **Dependencies**: Uses `JUnit`, `Mockito`, and `Spring Boot Test`.

- **Test Methods**:
  - `testGetDepartmentsByBuilding()`
  - `testGetDoctorsBySpecialty()`
  - `testAddDepartment()`
  - `testGetPatientsBySurname()`
  - `testAddDoctor()`

Each test method verifies the expected behavior of the controller endpoints, ensuring correct response statuses and data handling.