BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC PHENIKAA

BÁO CÁO BÀI TẬP KẾT THÚC HỌC PHẦN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Đề bài: "Phần mềm quản lý trại giam"

Giảng viên: Nguyễn Lệ Thu

Học phần: Lập trình hướng đối tượng-1-1-24

Lớp: N03

Nhóm sinh viên: 01

Thành viên:

1.Ngô Đặng Nhật Dũng_MSV:23010329

2.Hà Mạnh Long_MSV:23010390

Hà Nội, 10/2024

Lời cảm ơn

Trước hết, chúng tôi xin gửi lời cảm ơn chân thành đến cô **Nguyễn Lệ Thu**, người đã tận tình hướng dẫn và truyền đạt những kiến thức quý báu trong suốt quá trình học tập và thực hiện dự án này. Sự chỉ bảo tận tình của cô không chỉ giúp chúng tôi hiểu sâu hơn về các khái niệm, kỹ thuật lập trình mà còn tạo động lực để chúng tôi phát triển phần mềm **Quản lý trại giam** một cách hiệu quả.

Chúng tôi cũng xin gửi lời cảm ơn chân thành đến các cá nhân đã tham gia và hỗ trợ trong quá trình xây dựng và thử nghiệm phần mềm. Những phản hồi và góp ý từ phía người dùng đã giúp chúng tôi cải thiện sản phẩm, đảm bảo rằng phần mềm đáp ứng tốt các nhu cầu thực tiễn trong công tác quản lý và vận hành hệ thống trại giam.

Cuối cùng, xin cảm ơn tất cả các bạn trong nhóm dự án, những người đã không ngừng nỗ lực và cùng nhau vượt qua các thử thách để hoàn thành phần mềm này đúng tiến độ và với chất lượng tốt nhất.

Xin chân thành cảm ơn!

Tài liệu dự án

1.Vai trò của các thành viên:

STT	Họ và tên	Đóng góp	Ghi chú
1	Ngô Đặng Nhật Dũng	50%	
2	Hà Mạnh Long	50%	

2.Các liên kết

- **Github Repo:** https://github.com/nglthu/oop_group1_n03
- **Video giới thiệu:** https://youtu.be/swDENh-yDdU

MỤC LỤC

Chương 1. Giới thiệu dự án

1. Giới thiệu tổng quan

Trong thời đại công nghệ số phát triển mạnh mẽ, việc ứng dụng công nghệ thông tin vào quản lý và vận hành các hoạt động hành chính ngày càng trở nên quan trọng. Đặc biệt, trong các lĩnh vực yêu cầu quản lý dữ liệu lớn và phức tạp như hệ thống trại giam, việc sử dụng phần mềm để hỗ trợ các công việc quản lý là vô cùng cần thiết. Phần mềm Quản lý trại giam được phát triển nhằm cung cấp một giải pháp hiệu quả trong việc quản lý các thông tin liên quan đến phạm nhân và phòng giam. Phần mềm cung cấp một loạt các tính năng từ thêm, sửa, xóa thông tin phạm nhân,thông tin phòng giam; chuyển đổi phòng giam cho phạm nhân và tìm kiếm phạm nhân, phần mềm này giúp cho việc quản lý trở nên dễ dàng, nhanh chóng và chính xác hơn.

Hệ thống được thiết kế đơn giản, dễ sử dụng, giúp các cán bộ trại giam có thể nhanh chóng làm quen và áp dụng vào công việc hàng ngày, đồng thời đảm bảo bảo mật thông tin và sự chính xác trong quản lý.

2. Mục tiêu của dự án

Mục tiêu chính của dự án phần mềm quản lý trại giam bao gồm:

- Xây dựng một hệ thống quản lý thông tin phạm nhân một cách rõ ràng và hiệu quả, cho phép cập nhật và theo dõi thông tin một cách nhanh chóng.
- Tự động hóa quy trình quản lý phòng giam và phạm nhân, bao gồm việc thêm, sửa, xóa và điều chỉnh vị trí phạm nhân trong các phòng giam.

- Giảm thiểu khối lượng công việc thủ công và nguy cơ sai sót trong quá trình quản lý.
- Nâng cao khả năng quản lý chặt chẽ và dễ dàng điều phối các phạm nhân giữa các phòng giam khi cần thiết.

3. Lợi ích của dự án

Việc triển khai phần mềm quản lý trại giam mang lại nhiều lợi ích đáng kể:

- **Tăng hiệu suất làm việc**: Các quy trình như nhập dữ liệu, cập nhật thông tin phạm nhân, quản lý phòng giam đều được sử dụng tự động, giúp tiết kiệm thời gian và công sức cho cán bộ.
- **Đảm bảo tính chính xác**: Thông tin được lưu trữ và cập nhật liên tục, hạn chế tình trạng sai lệch dữ liệu do quản lý thủ công.
- **Dễ dàng quản lý và truy xuất thông tin**: Mọi dữ liệu về phạm nhân và phòng giam đều có thể truy cập nhanh chóng và dễ dàng, giúp cán bộ trại giam theo dõi sát sao và điều chỉnh kịp thời khi cần thiết.
- **Bảo mật cao**: Hệ thống đảm bảo an toàn cho thông tin nhạy cảm về phạm nhân và các hoạt động nội bộ của trại giam.

4. Công nghệ sử dụng

- Ngôn ngữ lập trình: Java_một ngôn ngữ lập trình mạnh mẽ, phổ biến và có khả năng chạy trên nhiều nền tảng khác nhau
- Framework: Spring Boot_là framework Java mà chúng tôi đã sử dụng để phát triển ứng dụng này. Nó giúp đơn giản hóa quá trình phát triển, cho phép xây dựng các ứng dụng web hoặc API một

cách nhanh chóng và dễ dàng bằng cách loại bỏ các cấu hình phức tạp..

- Postman để tương tác với API: sử dụng Postman để kiểm tra và gửi các yêu cầu HTTP (GET, POST, PUT, DELETE) đến API của ứng dụng. Postman cho phép gửi các dữ liệu dạng JSON trực tiếp tới ứng dụng Spring Boot và nhận phản hồi (response) dưới dạng JSON. Khi gửi yêu cầu, ứng dụng Spring Boot sẽ nhận dữ liệu từ Postman, xử lý và lưu vào cơ sở dữ liệu MySQL.
- Lưu trữ dữ liệu trong cơ sở dữ liệu MySQL: Ứng dụng sử dụng MySQL để lưu trữ các thông tin về phạm nhân và phòng giam. Spring Boot kết nối với cơ sở dữ liệu MySQL thông qua JPA (Java Persistence API) và Spring Data JPA. Điều này giúp bạn dễ dàng thực hiện các thao tác như thêm, sửa, xóa và truy xuất dữ liệu mà không cần viết quá nhiều mã SQL.
- Maven: tự động hóa các quy trình xây dựng phần mềm, quản lý thư viện (dependencies), và tạo cấu trúc chuẩn cho dự án.

- Cấu trúc ứng dụng Spring Boot với MySQL: Ứng dụng được chia thành các lớp nhỏ để quản lý và tương tác với cơ sở dữ liệu

Chương 2 Phân tích

1.Sơ đồ

Bao gồm sơ đồ lớp(Class Diagram) và sơ đồ tuần tự(Sequence Diagram) đã được trình bày trên wiki dự án

Link: https://github.com/nglthu/oop_group1_n03/wiki#wiki-pages-box

2.Các chức năng chính

2.1 Quản lý phạm nhân

2.1.1 Thêm phạm nhân

- Hệ thống cho phép thêm thông tin của một phạm nhân mới, bao gồm các thông tin cơ bản như họ tên, ngày tháng năm sinh, tội danh, thời gian thi hành án, v.v.
- Các thông tin them vào phải đùng kiểu dữ liệu được lập trình trước
- **Phương thức kiểm tra**: Sau khi thêm mới, hệ thống sẽ trả về thông báo xác nhận và hiển thị thông tin của phạm nhân mới được thêm vào danh sách để người quản trị có thể kiểm tra. Nếu thiếu thông tin hoặc thông tin nhập vào sai định dang, báo cho người dùng nhập lại hoặc nhập bổ sung thông tin.

2.1.2 Xóa phạm nhân

- Hệ thống hỗ trợ xóa thông tin của một phạm nhân khi người đó mãn hạn tù hoặc không còn thuộc sự quản lý của trại giam theo ID phạm nhân.
- **Phương thức kiểm tra**: Trước khi xóa kiểm tra xem có tồn tại phạm nhân đã chọn hay không. Nếu không thì báo cho người

dung biết, nếu có thì thực hiện xóa phạm nhân. Sau khi xóa, hệ thống sẽ kiểm tra xem phạm nhân đã bị loại bỏ khỏi danh sách và trả về kết quả xác nhận rằng phạm nhân đã không còn tồn tại trong cơ sở dữ liệu.

2.1.3 Cập nhật thông tin phạm nhân

- Cho phép cập nhật thông tin của một phạm nhân khi có sự thay đổi hoặc bổ sung thông tin (ví dụ: thời gian thụ án, thay đổi chi tiết cá nhân, v.v.).
- **Phương thức kiểm tra**: Sau khi sửa, hệ thống sẽ tự động cập nhật thông tin phạm nhân và cung cấp khả năng tra cứu lại thông tin qua ID để xác nhận các thay đổi đã được thực hiện chính xác.

2.1.4 Tìm kiếm phạm nhân

- Cho phép tìm kiếm và hiển thị chi tiết thông tin của một phạm nhân dựa trên mã ID duy nhất của họ.
- **Phương thức kiểm tra**: Sau khi nhập mã ID, hệ thống sẽ trả về thông tin chi tiết của phạm nhân. Phương thức kiểm tra bao gồm so khớp thông tin với cơ sở dữ liệu để đảm bảo ID chính xác

2.1.5 Hiển thị danh sách phạm nhân

- Hệ thống cần cung cấp chức năng hiển thị danh sách tất cả các phạm nhân đang được quản lý, bao gồm các thông tin cơ bản như tên, tuổi, tội danh và thời gian thi hành án.

2.1.6 Chuyển phòng giam phạm nhân

- Hệ thống cho phép thay đổi phòng giam của phạm nhân, từ phòng giam hiện tại sang một phòng giam khác.

- **Phương thức kiểm tra**: Trước khi chuyển phòng, kiểm tra xem phòng được chuyển tới có còn chỗ trống không, nếu không thì báo cho người dung, nếu có thì thực hiện chuyển phòng giam. Sau khi đổi phòng, hệ thống sẽ trả về thông báo xác nhận và cho phép kiểm tra lại danh sách phạm nhân trong phòng mới, đảm bảo phạm nhân đã được chuyển thành công từ phòng cũ sang phòng mới.

2.2 Quản lý phòng giam

2.2.1 Thêm phòng giam:

- Hệ thống cho phép thêm thông tin về một phòng giam mới, bao gồm mã phòng, sức chứa tối đa, v.v.
- **Phương thức kiểm tra**: Sau khi thêm mới, hệ thống sẽ trả về thông báo xác nhận và hiển thị thông tin chi tiết của phòng giam mới để kiểm tra.

2.2.2 Xóa phòng giam

- Hệ thống hỗ trợ xóa phòng giam khi không còn được sử dụng hoặc trong trường hợp tái cơ cấu lại khu vực giam giữ.
- **Phương thức kiểm tra**: Sau khi xóa phòng giam, hệ thống sẽ trả về xác nhận và đảm bảo rằng phòng giam không còn hiển thị trong danh sách.

2.2.3 Chỉnh sửa thông tin phòng giam

- Cung cấp chức năng để chỉnh sửa thông tin của phòng giam khi có sự thay đổi (ví dụ: sửa sức chứa, id).
- **Phương thức kiểm tra**: Sau khi sửa, thông tin phòng giam sẽ được cập nhật và có thể kiểm tra lại bằng cách xem chi tiết phòng giam qua ID hoặc tên phòng.

2.2.4 Hiển thị danh sách phạm nhân trong phòng giam

- Hệ thống cần cung cấp chức năng hiển thị danh sách các phạm nhân đang bị giam trong một phòng giam cụ thể, kèm theo thông tin chi tiết về mỗi phạm nhân.

3 Lưu trữ dữ liệu

Dữ liệu của phạm nhân sẽ được lưu trữ trong cơ sở dữ liệu MySQL

3.1 Kết nối giữa Spring Boot và MySQL

```yaml
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/finalproject
    username: root
    password: 123456

  jpa:
    hibernate:
      ddl-auto: update
    open-in-view: true
```

H3.1 Cơ sở dữ liệu MySQL

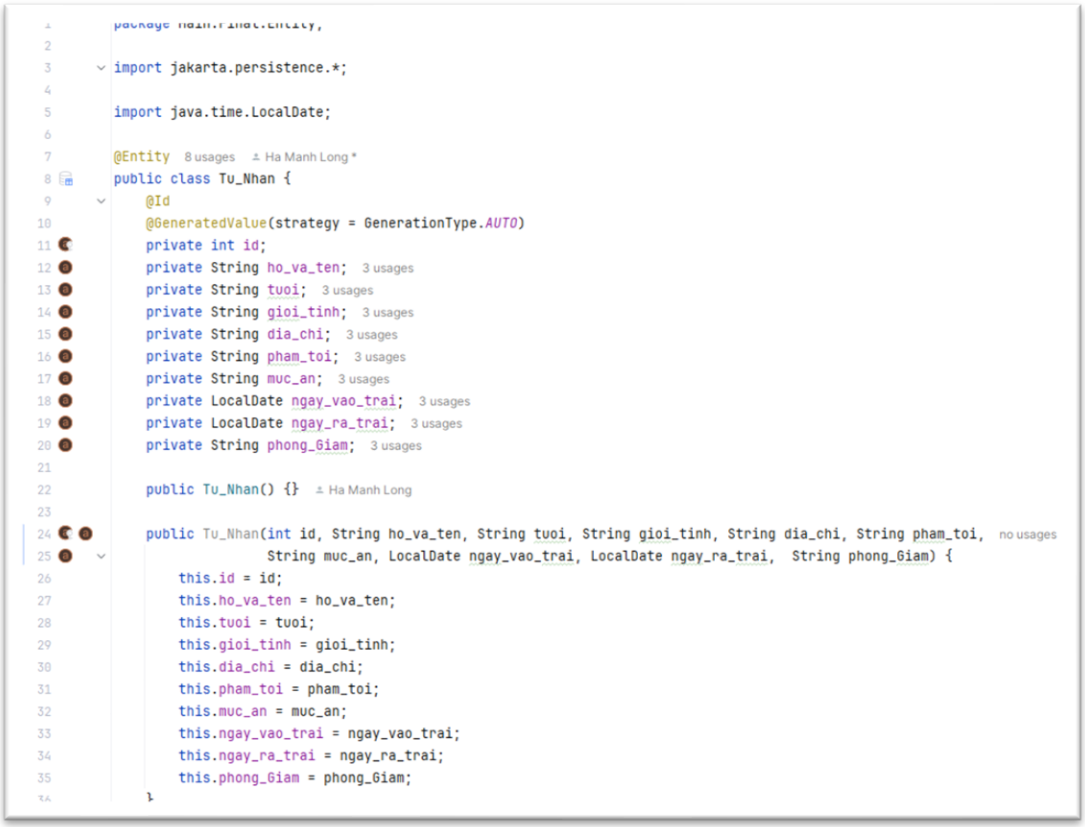
Trong đó :

- “url: jdbc:mysql://localhost:3306/finalproject” : Địa chỉ kết nối tới cơ sở dữ liệu MySQL.
- “username: root”: Tên người dùng MySQL
- “password: 123456”: Mật khẩu MySQL
- Toàn bộ phần “jpa”: Spring Boot sẽ tự động tạo hoặc cập nhật bảng trong cơ sở dữ liệu dựa trên các **Entity.**

3.2 Entity

Entity là lớp đại diện cho một bảng trong cơ sở dữ liệu. Cần tạo một lớp Java được ánh xạ tới bảng cơ sở dữ liệu bằng cách sử dụng các annotation như @Entity, @Table, @Id, và @GeneratedValue.

3.2.1 Phạm nhân



```java
import jakarta.persistence.*;

import java.time.LocalDate;

@Entity  8 usages  Ha Manh Long *
public class Tu_Nhan {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String ho_va_ten;  3 usages
    private String tuoi;  3 usages
    private String gioi_tinh;  3 usages
    private String dia_chi;  3 usages
    private String pham_toi;  3 usages
    private String muc_an;  3 usages
    private LocalDate ngay_vao_trai;  3 usages
    private LocalDate ngay_ra_trai;  3 usages
    private String phong_Giam;  3 usages

    public Tu_Nhan() {}  Ha Manh Long

    public Tu_Nhan(int id, String ho_va_ten, String tuoi, String gioi_tinh, String dia_chi, String pham_toi,  no usages
                   String muc_an, LocalDate ngay_vao_trai, LocalDate ngay_ra_trai,  String phong_Giam) {
        this.id = id;
        this.ho_va_ten = ho_va_ten;
        this.tuoi = tuoi;
        this.gioi_tinh = gioi_tinh;
        this.dia_chi = dia_chi;
        this.pham_toi = pham_toi;
        this.muc_an = muc_an;
        this.ngay_vao_trai = ngay_vao_trai;
        this.ngay_ra_trai = ngay_ra_trai;
        this.phong_Giam = phong_Giam;
    }
```

H3.2.1.1 Entity phạm nhân

- Import jakarta.persistence.* là các thư viện cần thiết để hướng đến, ánh xạ 1 đối tượng vào CSDL
- @Entity đại diện cho 1 thực thể (obj) lưu trữ dữ liệu của 1 đối tượng
- Tu_nhan là 1 entity và sẽ được ánh xạ vào trong CSDL để tạo thành 1 bảng

- @ID là trường khóa chính của đối tượng Tu_nhan, là 1 cách để xác định khóa chính cho đối tượng
- @GeneratedValue(strategy = GenerationType.AUTO) ở đây chọn auto để trường khóa chính id được tự động sinh ra bởi hệ thống.

➢ Khai báo các thuộc tính của phạm nhân (int id, string ho_va_ten,....

Constructor:

- Tu_Nhan(): Constructor mặc định, không có tham số.
- Tu_Nhan(...): Constructor có tham số, cho phép khởi tạo một đối tượng Tu_Nhan với các giá trị cụ thể.

Getter và Setter:

```
38      public int getId() { return id; }
41      public void setId(int id) { this.id = id; }
44
45      public String getHo_va_ten() { return ho_va_ten; }
48      public void setHo_va_ten(String ho_va_ten) { this.ho_va_ten = ho_va_ten; }
51
52      public String getTuoi() { return tuoi; }
55      public void setTuoi(String tuoi) { this.tuoi = tuoi; }
58
59      public String getGioi_tinh() { return gioi_tinh; }
62      public void setGioi_tinh(String gioi_tinh) { this.gioi_tinh = gioi_tinh; }
65
66      public String getDia_chi() { return dia_chi; }
69      public void setDia_chi(String dia_chi) { this.dia_chi = dia_chi; }
72
73      public String getPham_toi() { return pham_toi; }
76      public void setPham_toi(String pham_toi) { this.pham_toi = pham_toi; }
79
80      public String getMuc_an() { return muc_an; }
83      public void setMuc_an(String muc_an) { this.muc_an = muc_an; }
86
87      public LocalDate getNgay_vao_trai() { return ngay_vao_trai; }
90      public void setNgay_vao_trai(LocalDate ngay_vao_trai) { this.ngay_vao_trai = ngay_vao_trai; }
93
94      public LocalDate getNgay_ra_trai() { return ngay_ra_trai; }
97      public void setNgay_ra_trai(LocalDate ngay_ra_trai) { this.ngay_ra_trai = ngay_ra_trai; }
100
101     public String getPhong_giam() { return phong_Giam; }
104     public void setPhong_giam(String phong_Giam) { this.phong_Giam = phong_Giam; }
107 }
```

H3.2.1.2 Getter và Setter

Các phương thức getter và setter được sử dụng để truy cập và thay đổi giá trị của các thuộc tính trong lớp. Ví dụ:

- getHo_va_ten(): Lấy giá trị của thuộc tính ho_va_ten.
- setHo_va_ten(String ho_va_ten): Thiết lập giá trị cho ho_va_ten.

3.2.2 Phòng giam

Tương tự với đối tượng Phạm nhân:

```java
import ...

@Entity  8 usages  Ha Manh Long
public class Phong_Giam {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String ten_phong;  3 usages
    private String ten_tu_nhan;  3 usages
    private String so_luong_hien_tai;  3 usages
    private String so_luong_toi_da;  3 usages

    public Phong_Giam() {}  Ha Manh Long

    public Phong_Giam(int id, String ten_phong, String ten_tu_nhan, String so_luong_hien_tai, String so_luong_toi_da) {
        this.id = id;
        this.ten_phong = ten_phong;
        this.ten_tu_nhan = ten_tu_nhan;
        this.so_luong_hien_tai = so_luong_hien_tai;
        this.so_luong_toi_da = so_luong_toi_da;
    }
```

H3.2.2.1 Entity phòng giam

```java
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getTen_phong() { return ten_phong; }
    public void setTen_phong(String ten_phong) { this.ten_phong = ten_phong; }

    public String getTen_tu_nhan() { return ten_tu_nhan; }
    public void setTen_tu_nhan(String ten_tu_nhan) { this.ten_tu_nhan = ten_tu_nhan; }

    public String getSo_luong_hien_tai() { return so_luong_hien_tai; }
    public void setSo_luong_hien_tai(String so_luong_hien_tai) { this.so_luong_hien_tai = so_luong_hien_tai; }

    public String getSo_luong_toi_da() { return so_luong_toi_da; }
    public void setSo_luong_toi_da(String so_luong_toi_da) { this.so_luong_toi_da = so_luong_toi_da; }
}
```

H3.2.2.2 Getter và Setter

3.3 Repositories

Repositories được tạo để thao tác với cơ sở dữ liệu: Spring Data JPA cung cấp cơ chế để làm việc với cơ sở dữ liệu thông qua Repository. Bạn cần tạo một interface kế thừa từ JpaRepository để tương tác với MySQL.

3.3.1 Phạm nhân

```
1  package Main.Final.Repositories;
2
3  import Main.Final.Entity.Tu_Nhan;
4  import org.springframework.data.jpa.repository.JpaRepository;
5
6  public interface Tu_NhanRepositories extends JpaRepository<Tu_Nhan, Integer> {  3 usages
7  }
```

- “import Main.Final.Entity.Tu_Nhan” :
  - Nhập lớp Tu_Nhan từ package Main.Final.Entity. Interface này sẽ làm việc với entity Tu_Nhan để thực hiện các thao tác CRUD (Create, Read, Update, Delete) trên bảng cơ sở dữ liệu tương ứng với entity này.

- “import org.springframework.data.jpa.repository.JpaRepository”
  - Nhập JpaRepository từ thư viện Spring Data JPA. JpaRepository là một interface có sẵn trong Spring Data, cung cấp các phương thức mặc định để thao tác với cơ sở dữ liệu (như lưu trữ, cập nhật, xóa, tìm kiếm, v.v.).

- “public interface Tu_NhanRepositories extends JpaRepository<Tu_Nhan, Integer> {}”:
  - public interface: Đây là khai báo một interface công khai.

- Tu_NhanRepositories extends JpaRepository<Tu_Nhan, Integer>: Interface này kế thừa từ JpaRepository, nơi mà:

  Tu_Nhan: Là entity mà repository này sẽ làm việc. Nó sẽ ánh xạ với bảng tương ứng trong cơ sở dữ liệu.

- Integer: Là kiểu dữ liệu của khóa chính (id) của entity Tu_Nhan.

Ý nghĩa của interface này:

Tu_NhanRepositories sẽ tự động kế thừa tất cả các phương thức CRUD chuẩn từ JpaRepository, chẳng hạn như:

findAll(): Truy vấn tất cả phạm nhân.

findById(Integer id): Tìm một phạm nhân theo id.

save(Tu_Nhan tn): Lưu hoặc cập nhật một phạm nhân.

deleteById(Integer id): Xóa một phạm nhân theo id.

Vì kế thừa JpaRepository, không cần phải tự viết các phương thức cơ bản để tương tác với cơ sở dữ liệu. Spring Data JPA sẽ tự động cung cấp chúng khi khởi động ứng dụng Spring Boot.

3.3.2 Phòng giam

```
1  package Main.Final.Repositories;
2
3  import Main.Final.Entity.Phong_Giam;
4  import org.springframework.data.jpa.repository.JpaRepository;
5
6  public interface Phong_GiamRepositories extends JpaRepository<Phong_Giam, Integer> {  3 usages
7  }
```

H3.3.2 Repositories Phòng giam

4 Controller

Một **controller** trong Spring Boot, dùng để xử lý các yêu cầu HTTP liên quan đến việc quản lý thông tin phạm nhân (entity Tu_Nhan). Controller này cung cấp các chức năng CRUD (Create, Read, Update, Delete) thông qua các API REST.

4.1 @RestController

Đây là chú thích để đánh dấu lớp này là một controller trong Spring Boot, nó sẽ xử lý các yêu cầu HTTP và trả về dữ liệu JSON hoặc XML theo mặc định.

4.2 @RequestMapping("/prisoner")

```
15    @RequestMapping("/prisoner")
16    public class Tu_NhanController {
17        private final Tu_NhanRepositories tuNhanRepositories;  10 usages
```

Chú thích này xác định URL gốc cho tất cả các phương thức trong lớp controller. Mọi yêu cầu đến /prisoner sẽ được chuyển tới lớp này để xử lý.

4.3 Thuộc tính:

“private final Tu_NhanRepositories tuNhanRepositories;”: Đây là repository được tiêm (inject) vào controller, dùng để tương tác với cơ sở dữ liệu thông qua các phương thức CRUD.

4.4 Constructor:

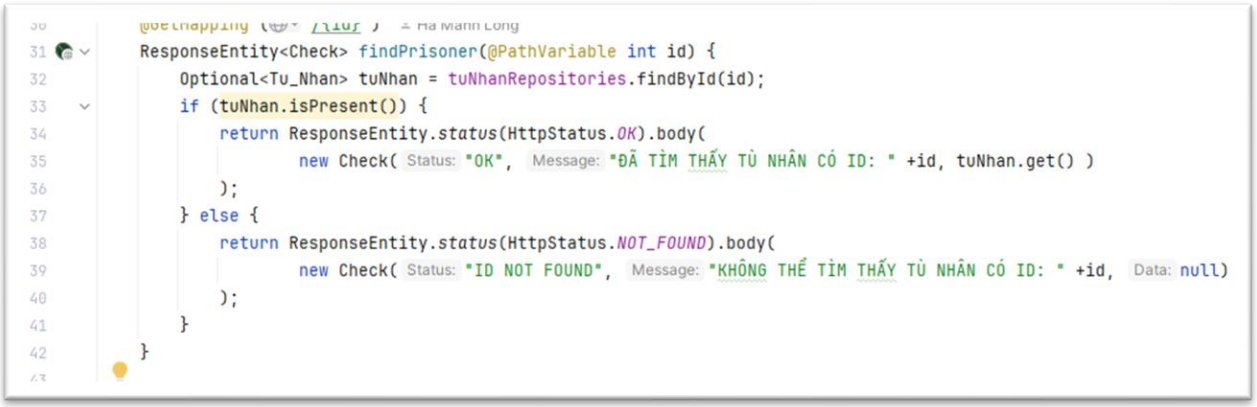
```
@Autowired  Ha Manh Long
public Tu_NhanController(Tu_NhanRepositories tuNhanRepositories) { this.tuNhanRepositories = tuNhanRepositories; }
```

@Autowired: Chú thích này để Spring tự động tiêm (inject) một đối tượng Tu_NhanRepositories vào constructor để có thể sử dụng các phương thức của repository.

4.5 @GetMapping("") - Lấy danh sách phạm nhân

List<Tu_Nhan> getAllPrisoner(): Phương thức này xử lý các yêu cầu GET /prisoner và trả về danh sách tất cả các phạm nhân. Dữ liệu được lấy từ cơ sở dữ liệu qua tuNhanRepositories.findAll().

4.6 @GetMapping("/{id}") - Tìm kiếm phạm nhân theo ID



ResponseEntity<Check> findPrisoner(@PathVariable int id): Phương thức này xử lý yêu cầu GET /prisoner/{id}, tìm kiếm phạm nhân theo id. Nó trả về một đối tượng ResponseEntity chứa thông tin về trạng thái tìm kiếm (tìm thấy hoặc không).

Nếu tìm thấy phạm nhân, nó trả về đối tượng Check với thông báo "Đã tìm thấy phạm nhân", nếu không thì trả về "ID NOT FOUND".

4.7 @PostMapping("/insert") - Thêm mới phạm nhân

```
@PostMapping("/insert")  ▲ Ha Manh Long
ResponseEntity<Check> createPrisoner(@RequestBody Tu_Nhan newPrisoner) {
    try {
        tuNhanRepositories.save(newPrisoner);
        return ResponseEntity.status(HttpStatus.CREATED).body(
                new Check( Status: "OK",  Message: "BẠN ĐÃ TẠO THÀNH CÔNG 1 TỘI PHẠM MỚI", newPrisoner )
        );
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(
                new Check( Status: "ERROR",  Message: "KHÔNG THỂ TẠO THÊM THÔNG TIN MỚI DO THÔNG TIN BẠN NHẬP VÀO KHÔNG ĐÚNG",  Data: nul
        );
    } finally {
        tuNhanRepositories.flush();
    }
}
```

ResponseEntity<Check> createPrisoner(@RequestBody Tu_Nhan newPrisoner): Phương thức này xử lý yêu cầu POST /prisoner/insert, dùng để thêm một phạm nhân mới vào cơ sở dữ liệu. Thông tin phạm nhân được gửi dưới dạng JSON trong phần thân của yêu cầu (@RequestBody).

Sau khi thêm thành công, trả về HTTP status CREATED cùng thông báo "Tạo thành công". Nếu có lỗi, trả về INTERNAL_SERVER_ERROR với thông báo lỗi.

4.8 @PutMapping("/update") - Cập nhật thông tin phạm nhân

```
@PutMapping("/update")  ▲ Ha Manh Long
ResponseEntity<Check> updatePrisoner(@RequestBody Tu_Nhan updatePrisoner) {
    try {
        tuNhanRepositories.save(updatePrisoner);
        return ResponseEntity.status(HttpStatus.OK).body(
                new Check( Status: "OK",  Message: "BẠN ĐÃ CẬP NHẬP THÔNG TIN THÀNH CÔNG", updatePrisoner )
        );
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(
                new Check( Status: "ERROR",  Message: "CHÚNG TÔI KHÔNG THỂ CẬP NHẬP CHO BẠN VUI LÒNG KIỂM TRA LẠI THÔNG TIN",  Data: null)
        );
    } finally {
        tuNhanRepositories.flush();
    }
}
```

ResponseEntity<Check> updatePrisoner(@RequestBody Tu_Nhan updatePrisoner): Phương thức này xử lý yêu cầu PUT /prisoner/update, dùng để cập nhật thông tin của một phạm nhân có sẵn.

Nếu cập nhật thành công, trả về HTTP status OK cùng thông báo "Cập nhật thành công". Nếu xảy ra lỗi, trả về thông báo lỗi.

4.9 @DeleteMapping("/{id}") - Xóa phạm nhân theo ID

```
76      @DeleteMapping ("/{id}")  Ha Manh Long
77      public ResponseEntity<Check> deletePrisoner(@PathVariable int id) {
78          Optional<Tu_Nhan> deleteTuNhan = tuNhanRepositories.findById(id);
79          try {
80              tuNhanRepositories.delete(deleteTuNhan.get());
81              return ResponseEntity.status(HttpStatus.OK).body(
82                      new Check( Status: "OK",  Message: "BẠN ĐÃ XÓA THÔNG TIN THÀNH CÔNG", deleteTuNhan.get() )
83              );
84          } catch (Exception e) {
85              return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(
86                      new Check( Status: "ERROR",  Message: "KHÔNG TÌM THẤY THÔNG TIN CỦA PHẠM NHÂN CẦN XÓA",  Data: null)
87              );
88          } finally {
89              tuNhanRepositories.flush();
90          }
91      }
```

ResponseEntity<Check> deletePrisoner(@PathVariable int id): Phương thức này xử lý yêu cầu DELETE /prisoner/{id}, dùng để xóa phạm nhân theo id.

Nếu tìm thấy và xóa thành công, trả về HTTP status OK cùng thông báo "Xóa thành công". Nếu có lỗi (ví dụ như không tìm thấy phạm nhân), trả về INTERNAL_SERVER_E

RROR.

4.10 Các lớp liên quan:

```java
package Main.Final.Check;

public class Check {  26 usages  Ha Manh Long
    private String Status;  3 usages
    private String Message;  3 usages
    private Object Data;  3 usages

    public Check() {}  no usages  Ha Manh Long

    public Check(String Status, String Message, Object Data) {  16 usages
        this.Status = Status;
        this.Message = Message;
        this.Data = Data;
    }

    public String getStatus() { return Status; }
    public void setStatus(String Status) { this.Status = Status; }

    public String getMessage() { return Message; }
    public void setMessage(String Message) { this.Message = Message; }

    public Object getData() { return Data; }
    public void setData(Object Data) { this.Data = Data; }
}
```

Check: Là một lớp tùy chỉnh (có thể chưa được hiển thị ở đây), được dùng để gửi thông tin phản hồi lại cho người dùng, chứa các thông tin về kết quả xử lý yêu cầu, như thông báo trạng thái và dữ liệu liên quan.

Tổng kết:

- Lớp **Tu_NhanController** này được dùng để tạo một API REST cho việc quản lý phạm nhân (Tu_Nhan). Nó cung cấp các điểm cuối để lấy tất cả phạm nhân, tìm kiếm phạm nhân theo ID, thêm mới, cập nhật và xóa phạm nhân.
- Controller này hoạt động nhờ Spring Boot và Spring Data JPA, giúp tự động hóa rất nhiều trong việc xử lý truy vấn và quản lý cơ sở dữ liệu.

Chương 3 Tổng kết dự án

1. Tổng kết quá trình phát triển

Dự án Quản lý trại giam đã được xây dựng theo mô hình lập trình hướng đối tượng (OOP), tận dụng tối đa các đặc tính như tính kế thừa, tính đa hình, và tính đóng gói. Các đối tượng chính như Phạm Nhân và Phòng Giam được định nghĩa dưới dạng các lớp (class) với các thuộc tính và phương thức quản lý tương ứng, giúp hệ thống dễ dàng mở rộng và duy trì.

Quá trình phát triển trải qua nhiều giai đoạn từ phân tích yêu cầu, thiết kế hệ thống, đến lập trình, và kiểm thử. Nhóm đã áp dụng các nguyên tắc OOP một cách chặt chẽ nhằm đảm bảo tính nhất quán và tái sử dụng mã nguồn. Việc sử dụng công nghệ Java Spring Boot giúp tăng cường khả năng phát triển dự án theo từng module, dễ dàng tích hợp và mở rộng.

2.Đánh giá lợi ích và hiệu quả

Phần mềm đã mang lại nhiều lợi ích thiết thực cho việc quản lý trại giam. Một số điểm nổi bật bao gồm:

- Tính chính xác và minh bạch: Nhờ lập trình hướng đối tượng, thông tin về phạm nhân và phòng giam được quản lý theo các thực thể rõ ràng, giúp hạn chế tối đa sai sót trong quá trình nhập liệu và quản lý.

- Tiết kiệm thời gian: Các chức năng tự động như thêm, sửa, xóa phạm nhân hay phòng giam, cùng với việc chuyển phạm nhân giữa các phòng, giúp cho quá trình quản lý nhanh chóng và hiệu quả hơn.
- Dễ dàng mở rộng và bảo trì: Với OOP, hệ thống dễ dàng thêm các tính năng mới mà không làm ảnh hưởng đến các phần khác của ứng dụng, giúp cho việc bảo trì và nâng cấp phần mềm trở nên thuận tiện.

3. Hướng phát triển trong tương lai

Trong tương lai, dự án có thể tiếp tục được phát triển và mở rộng theo nhiều hướng khác nhau:

- Tích hợp tính năng bảo mật cao cấp: Sử dụng các kỹ thuật mã hóa và quản lý người dùng nâng cao để bảo vệ dữ liệu nhạy cảm của trại giam.
- Xây dựng hệ thống báo cáo tự động: Giúp quản lý trại giam có thể nắm bắt nhanh chóng các thông tin liên quan đến phạm nhân và phòng giam qua các báo cáo định kỳ.
- Thêm các chức năng mới giúp quản lí chuyên sâu và rộng khắp hơn hoạt động của trại giam.
- Phát triển giao diện người dùng thân thiện: Mặc dù hiện tại phần mềm chủ yếu tương tác qua Postman, nhưng trong tương lai có thể phát triển một giao diện web hoặc ứng dụng di động để tăng tính tiện dụng cho người dùng.

Với các tiềm năng trên, phần mềm Quản lý trại giam hứa hẹn sẽ trở thành một công cụ hữu ích, góp phần nâng cao hiệu quả và tính chính xác trong công tác quản lý trại giam.