**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**BÁO CÁO BÀI TẬP LỚN**

**LÊ ĐỨC MINH**

Minh.ld200395@sis.hust.edu.vn

|  |  |
| --- | --- |
| **Giảng viên:** | TS. Trịnh Tuấn Đạt |
| **Bộ môn:** | Thiết kế và xây dựng phần mềm |
| **Lớp:** | 143801 |
| **Nhóm:** | 6 |
| **HÀ NỘI, 12/2023** | |

**MỤC LỤC**

[CHƯƠNG 1. PHÂN TÍCH YÊU CẦU (BÀI 2) 1](#_Toc154531856)

[1.1 Biểu đồ use case 1](#_Toc154531857)

[1.1.1 Biểu đồ use case tổng quan 1](#_Toc154531858)

[1.1.2 Biểu đồ use case phân rã “Xem thông tin chấm công” 3](#_Toc154531859)

[1.1.3 Chỉnh sửa thông tin chấm công 4](#_Toc154531860)

[1.2 Đặc tả Use case 5](#_Toc154531861)

[1.2.1 Use case “Sửa thông tin chấm công theo ngày” 5](#_Toc154531862)

[1.3 Từ điển thuật ngữ 6](#_Toc154531863)

[1.3.1 Comment 6](#_Toc154531864)

[1.3.2 Click 6](#_Toc154531865)

[1.3.3 Import 7](#_Toc154531866)

[1.3.4 Export 7](#_Toc154531867)

[1.4 Đặc tả phụ trợ 7](#_Toc154531868)

[1.4.1 Hiệu năng 7](#_Toc154531869)

[1.4.2 Độ tin cậy 7](#_Toc154531870)

[1.4.3 An toàn, bảo mật 7](#_Toc154531871)

[1.4.4 Bảo trì 7](#_Toc154531872)

[1.4.5 Khả năng sử dụng 7](#_Toc154531873)

[1.4.6 Khả năng mở rộng 8](#_Toc154531874)

[CHƯƠNG 2. PHÂN TÍCH USECASE (BÀI 3) 9](#_Toc154531875)

[2.1 Thiết kế biểu đồ tương tác 9](#_Toc154531876)

[2.1.1 Use case “Xem báo cáo chấm công của đơn vị nhân viên văn phòng” 9](#_Toc154531877)

[2.2 Thiết kế biểu đồ lớp 13](#_Toc154531878)

[2.2.1 Use case “Sửa đổi thông tin chấm công theo ngày” 13](#_Toc154531879)

[CHƯƠNG 3. THIẾT KẾ GIAO DIỆN (BÀI 4) 14](#_Toc154531880)

[3.1 Sơ đồ chuyển đổi màn hình 14](#_Toc154531881)

[3.1.1 Sơ đồ chuyển đổi màn hình của UC Sửa đổi thông tin chấm công theo ngày 14](#_Toc154531882)

[3.2 Đặc tả màn hình 15](#_Toc154531883)

[3.2.1 Đặc tả màn hình RequestHomeView 15](#_Toc154531884)

[3.2.2 Đặc tả màn hình RequestDetailView 16](#_Toc154531885)

[3.3 Biểu đồ trình tự cho Subsystem 17](#_Toc154531886)

[3.4 Biểu đồ lớp cho Subsystem 19](#_Toc154531887)

[CHƯƠNG 4. THIẾT KẾ CHI TIẾT LỚP (BÀI 5) 20](#_Toc154531888)

[4.1 Thiết kế biểu đồ trình tự mức thiết kế 20](#_Toc154531889)

[4.1.1 Use case “Sửa đổi thông tin chấm công theo ngày” 20](#_Toc154531890)

[4.2 Biểu đồ lớp mức thiết kế 24](#_Toc154531891)

[4.2.1 Use case “Sửa đổi thông tin chấm công theo ngày” 24](#_Toc154531892)

[CHƯƠNG 5. CẢI THIỆN BIỂU ĐỒ TRÌNH TỰ VÀ LỚP MỨC THIẾT KẾ (BÀI 6) 28](#_Toc154531893)

[5.1 Cách chỉnh sửa và ý nghĩa mang lại 28](#_Toc154531894)

[5.1.1 Áp dụng mẫu thiết kế Singleton 28](#_Toc154531895)

[5.1.2 Áp dụng mẫu thiết kế Factory 28](#_Toc154531896)

[5.1.3 Áp dụng nguyên tắc Cohesion and Coupling 28](#_Toc154531897)

[5.1.4 Áp dụng nguyên lý SOLID 28](#_Toc154531898)

[5.2 Thiết kế biểu đồ trình tự mức thiết kế 30](#_Toc154531899)

[5.2.1 Use case “Sửa đổi thông tin chấm công theo ngày” 30](#_Toc154531900)

[5.3 Biểu đồ lớp mức thiết kế 34](#_Toc154531901)

[5.3.1 Use case “Sửa đổi thông tin chấm công theo ngày” 34](#_Toc154531902)

[CHƯƠNG 6. LẬP TRÌNH VÀ KIỂM THỬ ĐƠN VỊ 38](#_Toc154531903)

[6.1 Thiết kế test case cho module: request 38](#_Toc154531904)

[6.1.1 Mô tả module: 38](#_Toc154531905)

[6.1.2 Thiết kế test case cho phương thức getRequestOfEmployee: 38](#_Toc154531906)

# PHÂN TÍCH YÊU CẦU (BÀI 2)

## Biểu đồ use case

### Biểu đồ use case tổng quan

A diagram of people with text

Description automatically generated

Giải thích về các tác nhân:

* **Nhân viên**: Người dùng cấp thấp nhất, chỉ có thể xem được các thông tin chấm công liên quan đến mình
* **Nhân viên văn phòng**: Kế thừa nhân viên, chỉ xem được thông tin chấm công dạng nhân viên văn phòng.
* **Công nhân**: Kế thừa nhân viên, chỉ xem được thông tin chấm công dạng công nhân.
* **Trưởng đơn vị**: Người dùng cấp cao hơn nhân viên, có thể xem được thông tin chấm công của các thành viên trong đơn vị của mình quản lý.
* **Quản lý nhân sự**: Người dùng cấp cao nhất, có thể xem được thông tin chấm công của toàn bộ nhân viên công ty, có quyền chỉnh sửa thông tin chấm công, xem báo cáo, import/export dữ liệu chấm công.
* **Máy quét vân tay**: Hệ thống bên ngoài có nhiệm vụ cung cấp API cho hệ thống lấy dữ liệu chấm công
* **Hệ thống quản lý nhân sự**: Hệ thống bên ngoài có nhiệm vụ cung cấp API cho hệ thống chấm công lấy thông tin của nhân viên.
* **Hệ thống kế toán**: Hệ thống bên ngoài có khả năng lấy thông tin chấm công từ hệ thống chấm công để tính lương của nhân viên cho từng tháng.

Giải thích về các use case:

Composite use case:

* **Xem thông tin chấm công**: Use case tổng quát của các use case thực hiện các hành vi nghiệp vụ lên máy chấm công vân tay.
* **Chỉnh sửa thông tin chấm công**: Use case tổng quát của các use case thực hiện các hành vi nghiệp vụ lên

Actual use case:

* **Cung cấp dữ liệu chấm công nhân viên qua API**
* **Đăng kí tài khoản**
* **Đăng nhập**
* **Lấy dữ liệu chấm công nhân viên qua API**

### A diagram of a diagram Description automatically generatedBiểu đồ use case phân rã “Xem thông tin chấm công”

### Chỉnh sửa thông tin chấm công

A diagram of a person's diagram

Description automatically generated

## Đặc tả Use case

### Use case “Sửa thông tin chấm công theo ngày”

|  |  |  |  |
| --- | --- | --- | --- |
| Mã Use case | UC004 | Tên Use case | Sửa thông tin chấm công theo ngày |
| Tác nhân | Người quản lý nhân sự, Nhân viên văn phòng | | |
| Tiền điều kiện | Nhân viên văn phòng đã xem thông tin chấm công do hệ thống hiển thị | | |
| Luồng sự kiện chính  (Thành công) | |  |  |  | | --- | --- | --- | | STT | Thực hiện bởi | Hành động | | 1 | Nhân viên | Nhân viên chọn chức năng xem chi tiết thông tin chấm công | | 2 | Nhân viên | Tạo yêu cầu sửa đổi chấm công | | 3 | Hệ thống | Ghi nhận yêu cầu thay đổi thông tin chấm công | | 4 | Quản lý nhân sự | Chọn chức năng yêu cầu chỉnh sửa chấm công | | 5 | Hệ thống | Hiển thị danh sách yêu cầu chỉnh sửa chấm công | | 6 | Quản lý nhân sự | Chọn xem chi tiết yêu cầu | | 7 | Hệ thống | Hiển thị thông tin chấm công hiện tại và yêu cầu chỉnh sửa | | 8 | Quản lý nhân sự | Chọn chấp nhận yêu cầu sửa đổi | | 9 | Hệ thống | Sửa đổi thông tin chấm công trong cơ sở dữ liệu | | 10 | Hệ thống | Cập nhật trạng thái yêu cầu | | 11 | Hệ thống | Gửi thông báo về yêu cầu tới nhân viên | | | |
| Luồng sự kiện thay thế | |  |  |  | | --- | --- | --- | | STT | Thực hiện bởi | Hành động | | 8a. | Quản lý nhân sự | Từ chối yêu cầu chấm công | | 9a | Hệ thống | Cập nhật trạng thái yêu cầu | | 10a | Hệ thống | Gửi thông báo về yêu cầu tới nhân viên | | 9a1 | Hệ thống | Thông báo lỗi: yêu cầu đã được duyệt | | 9b | Hệ thống | Thông báo lỗi: yêu cầu đã được duyệt | | | |
| Hậu điều kiện | * Hệ thống cập nhật thông tin chấm công nếu yêu cầu được chấp nhận * Người dùng nhận được thông báo về yêu cầu | | |

\*Biểu đồ hoạt động:

A diagram of a work flow

Description automatically generated

## Từ điển thuật ngữ

Sau đây là một số thuật ngữ được sử dụng trong tài liệu:

### Comment

* Thêm miêu tả cho một cái gì đó, cụ thể trong tài liệu này là thêm miêu tả cho mã nguồn

### Click

* Hành động bấm vào nút bên trái chuột 2 lần liên tiếp trong một khoảng thời gian rất ngắn.

### Import

* Hành động nhập dữ liệu bằng file.

### Export

* Hành động xuất dữ liệu ra một file.

## Đặc tả phụ trợ

### Hiệu năng

* Trải nghiệm sử dụng mượt mà, ổn định.
* Thời gian phản hồi của ứng dụng dưới 1s.
* Cho phép 100 người dùng truy cập đồng thời.
* Cập nhật thông tin chấm công từ máy chấm công vân tay đến hệ thống chấm công với độ trễ ít hơn 1s

### Độ tin cậy

* Đảm bảo nhận diện thành công ít nhất 98% lần thử vân tay.
* Dữ liệu được backup thường xuyên đảm bảo không bị mất mát (1 lần/tuần).
* Có công cụ phục hồi dữ liệu từ cơ sở dữ liệu dự phòng nếu cơ sở dữ liệu chính gặp sự cố.
* Cho phép hệ thống hoạt động liên tục 24 giờ/ngày, 7 ngày/tuần, tăng khả năng chịu lỗi bằng việc duy trì hệ thống chấm công trên 2 node.
* Không quá 1 lỗi trên 800 dòng mã.

### An toàn, bảo mật

* Sử dụng chuẩn băm SHA-256 để mã hoá dữ liệu nhạy cảm như đặc trưng vân tay, mật khẩu của người dùng.
* Phân quyền người dùng để những người không liên quan có thể truy cập đến dữ liệu nhạy cảm.
* Yêu cầu xác thực người dùng (đăng nhập) trước khi thực hiện các hành động truy cập vào tài nguyên của hệ thống để tránh dữ liệu bị rò rỉ ra bên ngoài.
* Hệ thống được triển khai trên server cục bộ và được ngắt kết nối với internet để đảm bảo an toàn cho dữ liệu nhân viên (Hệ thống chỉ truy cập được bằng wifi của công ty).

### Bảo trì

* Phần mềm được tách ra thành các module cho từng tính năng để thuận tiện cho quá trình bảo trì.
* Phần mềm sẽ được duy trì trong khoảng 3 năm.
* Tất cả các phương thức trong mã nguồn đều phải được comment cũng như có tài liệu về cấu trúc hệ thống, cấu trúc mã nguồn.

### Khả năng sử dụng

* Giao diện Material thân thiện kèm tài liệu sử dụng, hầu như không cần đào tạo để sử dụng thành thạo.
* Giao diện người dùng có thể chạy được trên các trình duyệt phổ biến, truy cập chỉ bằng 1 click.

### Khả năng mở rộng

* Ứng dụng phải được thiết kế sao cho thuận tiện cho việc mở rộng quy mô về tính năng cũng như số lượng người dùng, số lượng dữ liệu liên tục mở rộng.

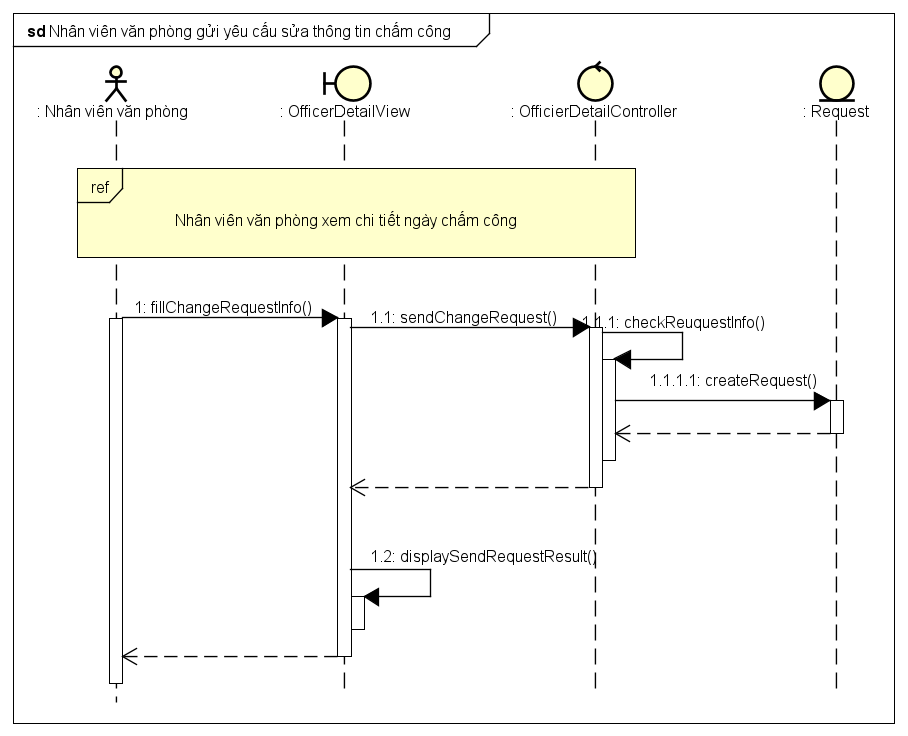
# PHÂN TÍCH USECASE (BÀI 3)

## Thiết kế biểu đồ tương tác

### Use case “Sửa đổi thông tin chấm công theo ngày”

#### Biểu đồ trình tự (mức phân tích)

\* Scenario 1: Nhân viên văn phòng gửi yêu cầu sửa thông tin chấm công



\*Scenario 2: Nhân viên văn phòng gửi yêu cầu sửa thông tin chấm công

A diagram of a project

Description automatically generated

\*Scenario 3: Quản lý từ chối yêu cầu thay đổi thông tin chấm công

A diagram of a project

Description automatically generated

#### Biểu đồ giao tiếp biểu diễn tương đương biểu đồ trình tự

\* Main Scenario: Quản lý nhân sự phê duyệt yêu cầu thay đổi thông tin chấm công

A diagram of a diagram

Description automatically generated

## Thiết kế biểu đồ lớp

### Use case “Sửa đổi thông tin chấm công theo ngày”

A screenshot of a computer

Description automatically generated

# THIẾT KẾ GIAO DIỆN (BÀI 4)

## Sơ đồ chuyển đổi màn hình

### Sơ đồ chuyển đổi màn hình của UC Sửa đổi thông tin chấm công theo ngày

A diagram of a login page

Description automatically generated

## Đặc tả màn hình

### Đặc tả màn hình RequestHomeView

A screenshot of a computer

Description automatically generated

\* Đặc tả điều khiển

|  |  |  |
| --- | --- | --- |
| **Điều khiển** | **Hành vi** | **Chức năng** |
| Button Trang chủ | Click | Điều hướng về Trang chủ của HR |
| Button Nhập dữ liệu chấm công | Click | Mở trang nhập dữ liệu chấm công |
| Button Yêu cầu chỉnh sửa chấm công | Click | Mở trang yêu cầu chấm công |
| Text chờ phê duyệt | Initial | Hiển thị số yêu cầu chờ phê duyệt |
| Text chấp nhận | Initial | Hiển thị số yêu cầu chấp nhận |
| Text từ chối | Initial | Hiển thị số yêu cầu đã từ chối |
| Textfield Mã nhân viên | Type | Chứa thông tin mã nhân viên cần tìm kiếm |
| Button Tìm kiếm | Click | Tìm kiếm hàng có mã nhân viên trong Textfield Mã nhân viên và hiển thị cho người dùng |
| Table yêu cầu sửa đổi chấm công | Initial | Hiển thị thông tin danh sách yêu cầu sửa đổi thông tin chấm công |
| Button Xem chi tiết | Click | Điều hướng đến trang Hiển thị thông tin chi tiết yêu cầu chỉnh sửa |
| Text Tổng số bản ghi | Initial | Hiển thị tổng số dòng của bảng |
| Textfield Ngày | Type | Chứa thông tin ngày cần tìm kiếm |
| Combobox Tháng | Choose | Chứa thông tin tháng cần tìm kiếm |
| Combobox Năm | Choose | Chứa thông tin năm cần tìm kiếm |
| Combobox Loại | Choose | Chứa thông tin loại yêu cầu cần tìm kiếm |

\* Đặc tả dữ liệu trên từng dòng của bảng chấm công của từng nhân viên

|  |  |  |
| --- | --- | --- |
| **Tên trường dữ liệu** | **Kiểu dữ liệu** | **Kiểu cách** |
| RequestID | int | Căn giữa |
| Mã nhân viên | String | Căn giữa |
| Thời gian | String | Căn giữa |
| Trạng thái | String | Căn giữa |

### Đặc tả màn hình RequestDetailView

A screenshot of a computer

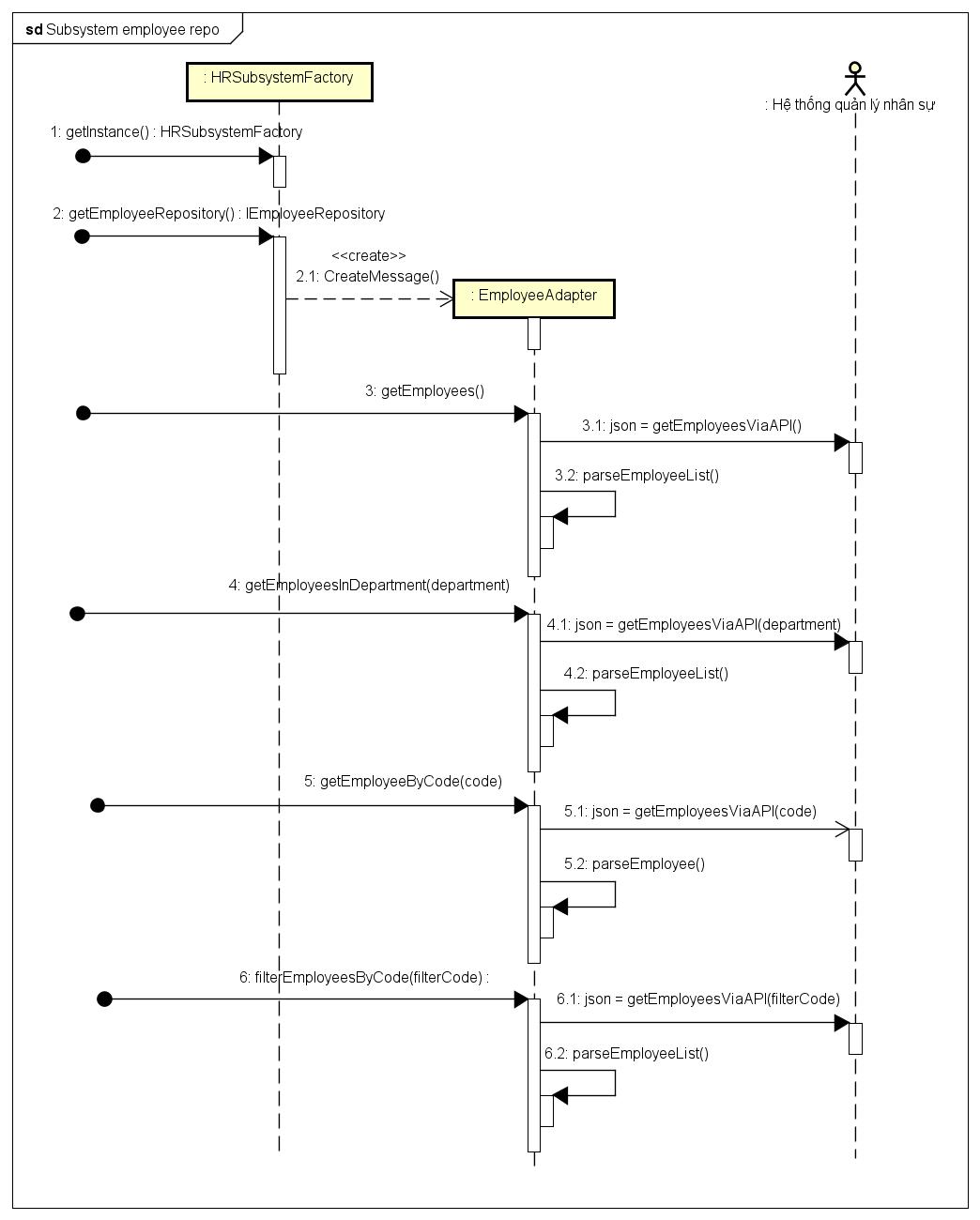
Description automatically generated

\* Đặc tả điều khiển

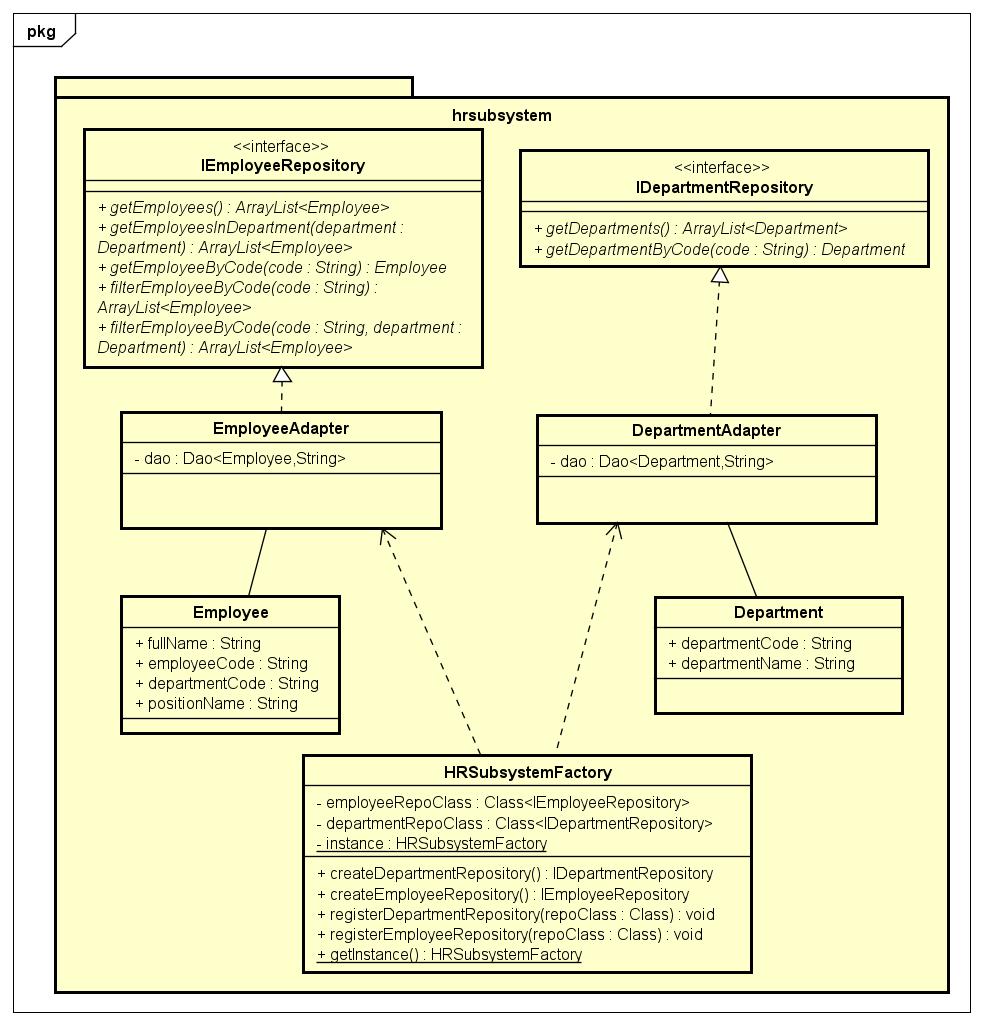
|  |  |  |
| --- | --- | --- |
| **Điều khiển** | **Hành vi** | **Chức năng** |
| Button Trang chủ | Click | Điều hướng về Trang chủ của HR |
| Button Nhập dữ liệu chấm công | Click | Mở trang nhập dữ liệu chấm công |
| Button Yêu cầu chỉnh sửa chấm công | Click | Mở trang yêu cầu chấm công |
| Button Quay lại | Click | Quay lại trang xem danh sách yêu cầu chấm công |
| Button Chấp nhận | Click | Phê duyệt yêu cầu chấm công |
| Button Từ chối | Click | Từ chối yêu cầu chấm công |
| Text Request ID | Initial | Hiển thị ID của yêu cầu |
| Text Log ID | Initial | Hiển thị ID của log chấm công cần sửa |
| Text Mã nhân viên | Initial | Hiển thị mã nhân viên |
| Text Ngày | Initial | Hiển thị ngày gửi yêu cầu |
| Text Trạng thái | Initial | Hiển thị trạng thái của yêu cầu |
| Text Sáng hiện tại | Initial | Hiển thị nhân viên có đi làm buổi sáng không trong log chấm công hiện tại |
| Text Chiều hiện tại | Initial | Hiển thị nhân viên có đi làm buổi chiều không trong log chấm công hiện tại |
| Text Đi muộn hiện tại | Initial | Hiển thị nhân viên đi làm muộn bao nhiêu giờ trong log chấm công hiện tại |
| Text Về sớm hiện tại | Initial | Hiển thị nhân viên về sớm bao nhiêu giờ trong log chấm công hiện tại |
| Text Lý do | Initial | Hiển thị lý do nhân viên muốn sửa đổi |
| Combobox Sáng sửa đổi | Choose | Hiển thị thông tin nhân viên muốn sửa |
| Combobox Chiều sửa đổi | Choose | Hiển thị thông tin nhân viên muốn sửa |
| Textfield Đi muộn sửa đổi | Type | Hiển thị thông tin nhân viên muốn sửa |
| Textfield Về sớm sửa đổi | Type | Hiển thị thông tin nhân viên muốn sửa |
| Textfield Phản hồi | Type | Hiển thị phản hồi của HR về yêu cầu |

## Biểu đồ trình tự cho Subsystem

\*Biểu đồ trình tự liên quan đến EmployeeAdapter



## Biểu đồ lớp cho Subsystem



# THIẾT KẾ CHI TIẾT LỚP (BÀI 5)

## Thiết kế biểu đồ trình tự mức thiết kế

### Use case “Sửa đổi thông tin chấm công theo ngày”

\* Tạo ra các đối tượng IUserRepository, IofficerAttendanceRepository, IRequestRepository

A diagram of a company

Description automatically generatedA diagram of a program

Description automatically generated

A diagram of a process

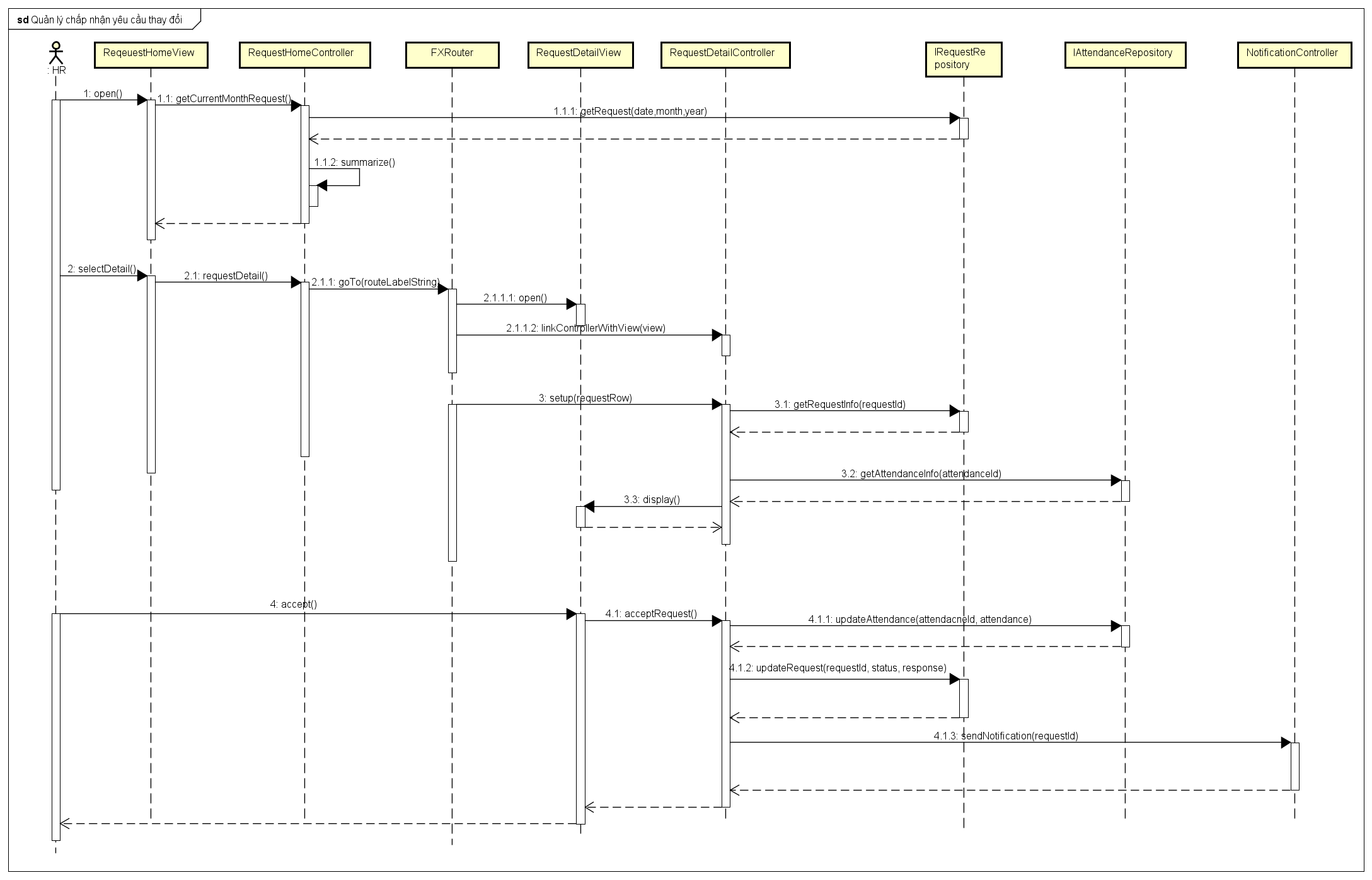
Description automatically generated

\* Scenario 1: Nhân viên văn phòng gửi yêu cấu thay đổi thông tin chấm công

A diagram of a diagram

Description automatically generated

\* Scenario 2: Quản lý nhân sự chấp nhận yêu cầu chấm công



\* Scenario 3: Quản lý nhân sự từ chối yêu cầu chấm công

A diagram of a project

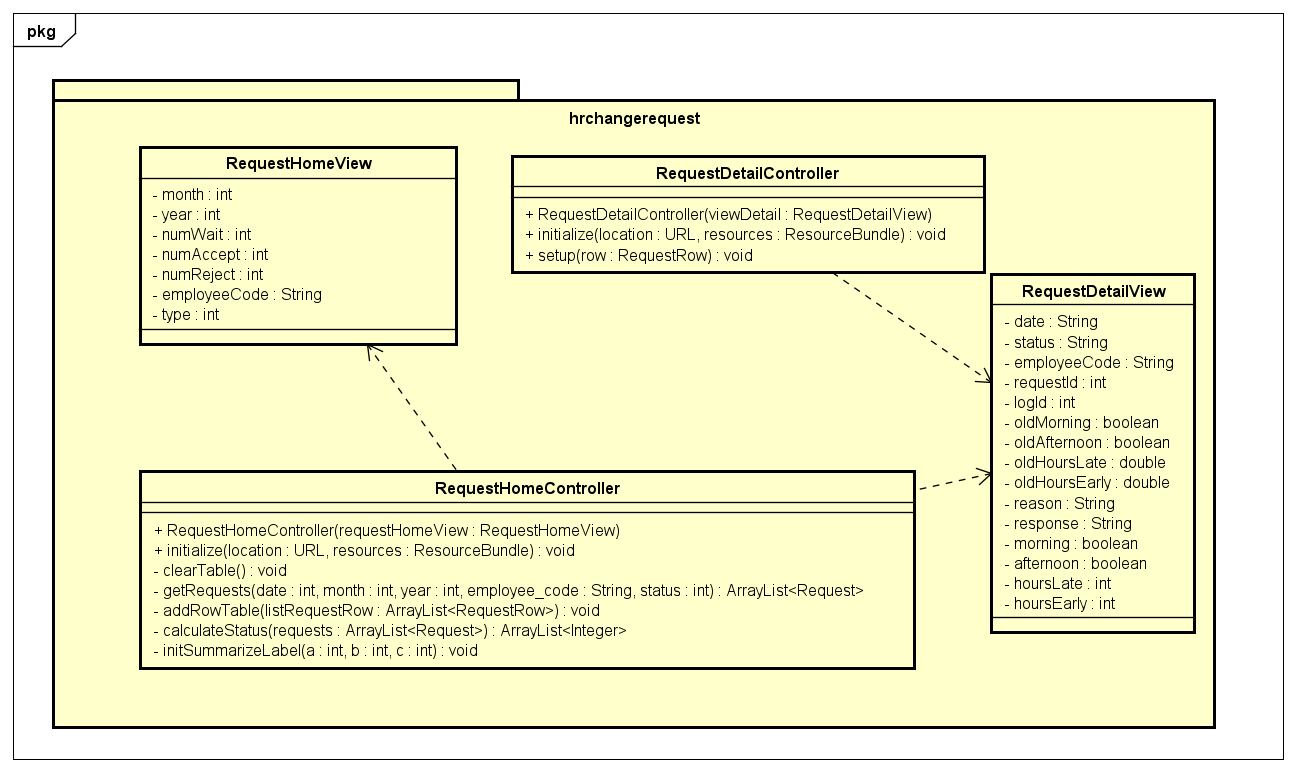
Description automatically generated

## Biểu đồ lớp mức thiết kế

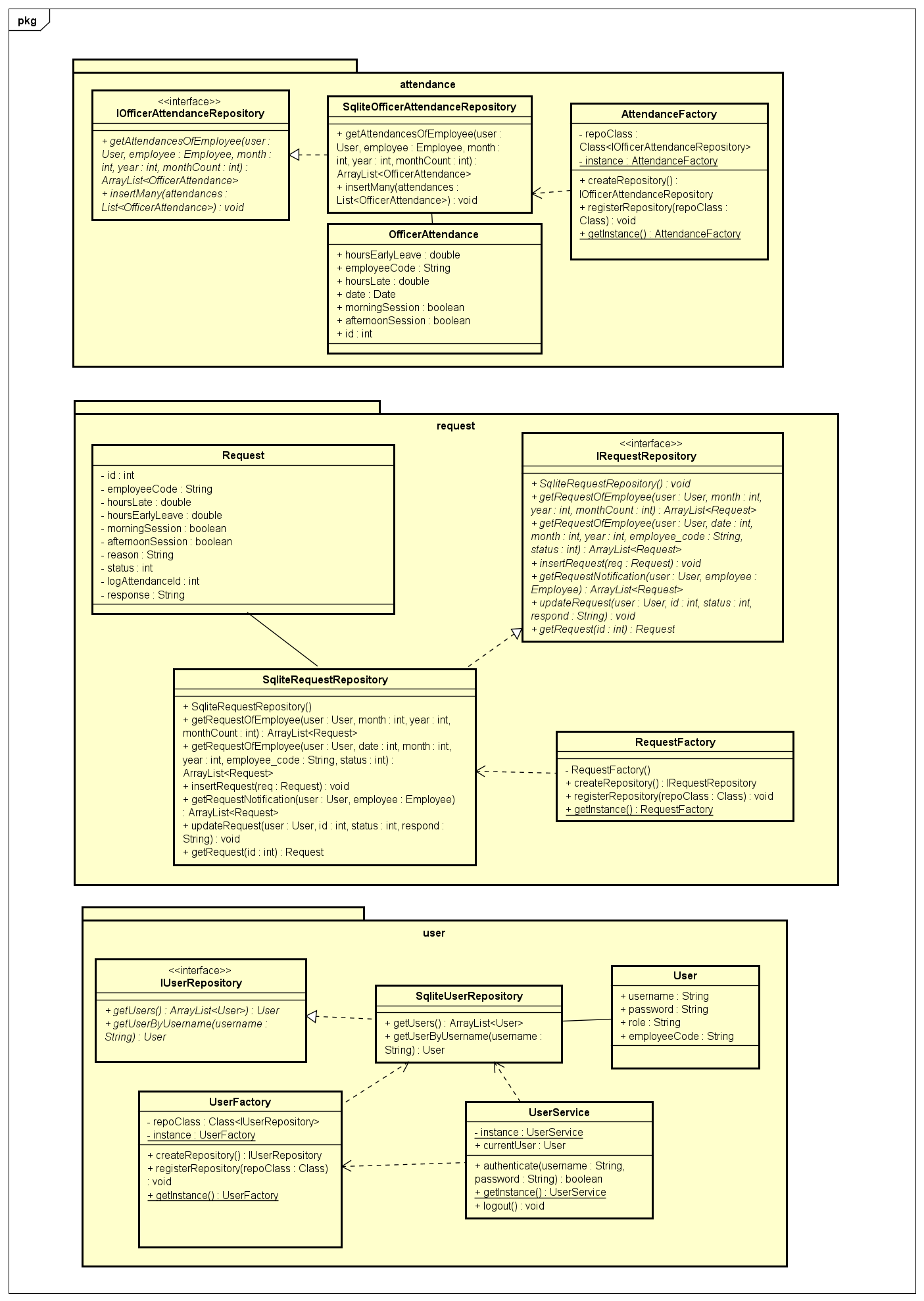
### Use case “Sửa đổi thông tin chấm công theo ngày”

A diagram of a diagram

Description automatically generated



A screenshot of a computer program

Description automatically generated

# CẢI THIỆN BIỂU ĐỒ TRÌNH TỰ VÀ LỚP MỨC THIẾT KẾ (BÀI 6)

## Cách chỉnh sửa và ý nghĩa mang lại

### Áp dụng mẫu thiết kế Singleton

Mẫu thiết kế Singleton là một trong những mẫu thiết kế phổ biến trong lập trình, được sử dụng trong những lớp **UserService, UserFactory, AttendanceFactory, HRSubsystemFactory**

để đảm bảo rằng:

* Một lớp chỉ có duy nhất một thể hiện.
* Có thể dễ dàng truy cập toàn cục đối tượng.
* Không cần phải quan tâm cách khởi tạo đối tượng.

### Áp dụng mẫu thiết kế Factory

Mẫu thiết kế Singleton là một trong những mẫu thiết kế phổ biến trong lập trình, được sử dụng trong những lớp **IEmployeeRepository, IDepartmentRepository, IOfficerAttendanceReposito-ry, IUserRepository** với các lớp Factory gồm **UserFactory, AttendanceFactory, HRSubsystem-Factory** được nhằm mục đích:

* **Tạo đối tượng mà không cần biết chi tiết cụ thể**: Mẫu Factory giúp ẩn đi logic cụ thể của việc tạo đối tượng. Khi sử dụng một phương thức tạo đối tượng từ Factory, bạn không cần biết chi tiết cách đối tượng đó được khởi tạo hoặc được xử lý bên trong.
* **Đảm bảo tuân thủ nguyên tắc "Open/Closed"**: Mẫu Factory giúp đảm bảo rằng khi thêm một triển khai mới của interface mà không cần phải sửa đổi mã nguồn hiện tại. Thay vào đó, chỉ cần thêm một lớp mới vào Factory.
* **Tăng sự linh hoạt của hệ thống**: Factory có thể được cấu hình để tạo ra các đối tượng thuộc các lớp con khác nhau dựa trên điều kiện nào đó. Điều này tăng sự linh hoạt và tái sử dụng mã nguồn.
* **Giảm sự phụ thuộc với đối tượng cụ thể**: Khi sử dụng một Factory để tạo đối tượng thì không cần phải biết chi tiết cụ thể cách khởi tạo của đối tượng đó. Điều này giảm sự phụ thuộc và giúp giữ cho mã nguồn linh hoạt hơn.

### Áp dụng nguyên tắc Cohesion and Coupling

Các lớp được phân lại thành các gói **pages, user, report, attendance, hrsubsystem** với mục đính tăng tính kết dính cho các lớp trong một gói, và giảm sự phụ thuộc giữa các gói với nhau. Những lớp có cùng một nhiệm vụ, cùng một nghiệp vụ thì được gom nhóm lại một gói, và mỗi gói lại có một nhiệm vụ khác nhau.

Cohesion giúp tăng tính đóng gói (encapsulation) và giảm sự phụ thuộc giữa các module. Nó làm cho mã nguồn dễ đọc, dễ bảo trì, và dễ kiểm thử. Cohesion cao cũng giúp tái sử dụng mã nguồn dễ dàng hơn.

Coupling giúp tăng tính linh hoạt và tái sử dụng của hệ thống. Khi các thành phần độc lập và không phụ thuộc quá mức lớn vào nhau, sự thay đổi trong một thành phần không ảnh hưởng quá lớn đến các thành phần khác. Điều này làm cho hệ thống linh hoạt và dễ mở rộng, bảo trì.

### Áp dụng nguyên lý SOLID

* *S - Một class chỉ nên giữ một trách nhiệm duy nhất*

Tất cả các lớp đều chỉ thực hiện một trách nhiệm duy nhất, như các lớp Model **OfficerAtten-dance, OfficerAndAttendance, User, Employee, Department** chỉ chứa dữ liệu về đối tượng, trong khi các lớp **EmployeeAdapter, DepartmentAdapter, SqliteOfficerAtte-ndanceRepository, SqliteUserRepository** phụ trách việc thao tác lưu trữ đối tượng trong database hoặc tương tác với hệ thống bên ngoài để lấy đối tượng, …

* *O - Có thể thoải mái mở rộng 1 module, nhưng hạn chế sửa đổi bên trong module đó*

Phần lớn các hàm đều được thiết kế theo dạng subcribe, unsubscribe để tránh việc thay đổi logic hàm, như các hàm **createRepository** và **registerRepository** trong lớp **UserFactory, AttendanceFactory**

* *L - Trong một chương trình, các object của class con có thể thay thế class cha mà không làm thay đổi tính đúng đắn của chương trình*

Không có lớp nào sử dụng kế thừa nên xem như hợp nguyên tắc.

* *I - Thay vì dùng 1 interface lớn, ta nên tách thành nhiều interface nhỏ, với nhiều mục đích cụ thể*

Subsystem được tách thành **IEmployeeRepository, IDepartmentRepository** nhằm để thực hiện truy vấn với từng kiểu dữ liệu lần lượt là **Employee, IDepartment** thay vì dồn vào 1 lớp duy nhất.

* *D - Các module cấp cao không nên phụ thuộc vào các module cấp thấp. Cả 2 nên phụ thuộc vào abstraction. Interface (abstraction) không nên phụ thuộc vào chi tiết, mà ngược lại. (Các class giao tiếp với nhau thông qua interface, không phải thông qua implementation.)*

Thay vì phải để mã phụ thuộc vào **EmployeeAdapter, DepartmentAdapter, SqliteOfficerAtte-ndanceRepository, SqliteUserRepository** thì cho những đoạn mã phụ thuộc thay thế lần lượt vào **IEmployeeRe-pository, IDepartmentRepository, IOfficerAttendanceRepository, IUserRepository** sẽ mang lại những điều sau:

* **Định hình Cấu trúc Hệ thống**: Nguyên tắc Dependency Convention giúp định hình cấu trúc của hệ thống bằng cách quy ước về cách các thành phần phụ thuộc lẫn nhau. Điều này tạo ra một cấu trúc tự nhiên và dễ hiểu, giúp người đọc mã nguồn dễ dàng theo dõi và hiểu cấu trúc tổng thể của hệ thống.
* **Giảm Độ phức tạp**: Nguyên tắc Dependency Convention giúp giảm độ phức tạp của hệ thống bằng cách hạn chế sự phụ thuộc giữa các thành phần (thay vào đó là phụ thuộc vào interface). Điều này giúp giảm nguy cơ xung đột và khó khăn trong quá trình phát triển và bảo trì.
* **Dễ Bảo trì và Nâng cấp**: Nguyên tắc Dependency Convention giúp việc bảo trì và nâng cấp trở nên dễ dàng hơn. Các thay đổi có thể được thực hiện mà không làm ảnh hưởng đến toàn bộ hệ thống, chỉ bằng cách thay đổi implement này bằng implement khác của interface, và người phát triển có thể tập trung vào các thành phần cụ thể mà họ đang làm việc.
* **Tăng Tính tái sử dụng**: Dependency Convention thúc đẩy việc sử dụng lại mã nguồn bằng cách tạo ra các thành phần độc lập, có thể được tái sử dụng trong các phần khác nhau của hệ thống hoặc trong các dự án khác.
* **Tăng Tính Mô đun hóa**: Nguyên tắc này hỗ trợ tính mô đun hóa của hệ thống, trong đó mỗi mô đun (hoặc thành phần) có trách nhiệm và chức năng cụ thể. Điều này giúp tạo ra các mô đun độc lập, dễ kiểm thử và duy trì.

## Thiết kế biểu đồ trình tự mức thiết kế

### Use case “Sửa đổi thông tin chấm công theo ngày”

\* Tạo ra các đối tượng IUserRepository, IofficerAttendanceRepository, IRequestRepository

A diagram of a company

Description automatically generatedA diagram of a program

Description automatically generated

A diagram of a process

Description automatically generated

\* Scenario 1: Nhân viên văn phòng gửi yêu cấu thay đổi thông tin chấm công

A diagram of a diagram

Description automatically generated

\* Scenario 2: Quản lý nhân sự chấp nhận yêu cầu chấm công

A diagram of a project

Description automatically generated

\* Scenario 3: Quản lý nhân sự từ chối yêu cầu chấm công

A diagram of a project

Description automatically generated

## Biểu đồ lớp mức thiết kế

### Use case “Sửa đổi thông tin chấm công theo ngày”

A diagram of a diagram

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generatedA screenshot of a computer screen

Description automatically generated

# LẬP TRÌNH VÀ KIỂM THỬ ĐƠN VỊ

## Thiết kế test case cho module: request

### Mô tả module:

Module chứa các hàm liên quan tới việc thống kê và quản lý yêu cầu chỉnh sửa thay đổi thông tin chấm công. Bên trong class này có sử dụng các dependency là các instance của các Class có chức năng làm việc với cơ sở dữ liệu, cung cấp các phương thức để làm các nghiệp vụ của Class Request.

### Thiết kế test case cho phương thức getRequestOfEmployee:

**Tên phương thức**: getRequestOfEmployee

**Kiểu trả về**: ArrayList<Request>

**Danh sách tham số**: User user, int date, int month, int year, String employee\_code, int status

**Chi tiết**: Hàm nhận đầu vào là thông tin của user, thông tin về ngày, tháng, năm, mã nhân viên và trạng thái và trả lại những yêu cầu thay đổi thông tin chấm công trong cơ sở dữ liệu tương ứng với những thông tin trên. Khi ngày = 0 thì ta tìm theo tất cả ngày trong tháng. Nếu mã nhân viên là rỗng hoặc null thì trả về yêu cầu của tất cả nhân viên

#### Kỹ thuật sử dụng kiểm thử hộp đen và phân vùng tương đương.

Kiểm thử hộp đen: Không cần biết logic bên trong code, chỉ cần biết định dạng thông số ngày hợp lệ và không hợp lệ, kết hợp với giao diện người dùng để kiểm thử.

Phân vùng tương đương cho phép phân chia tập hợp các điều kiện kiểm tra thành một phân vùng nên được coi là giống nhau. Để thiết kế được các test case cho phương thức này, đầu tiên xác định các vùng tương đương của đầu vào, ở đây là ngày, tháng và năm.

**Vùng tương đương hợp lệ:**

+ Ngày nhận giá trị nguyên dương và hợp lệ với các giá trị month, year.

**Vùng tương đương không hợp lệ**:

+ Ngày nhận giá trị âm

+ Ngày nhận giá trị hợp lệ với month và year

#### Thiết kế test case.

**Test case Query\_Success**

|  |  |
| --- | --- |
| **Test case #:** 1.1 | **Tên test case:** Query |
| **Hệ thống:** Hệ thống quản lý chấm công | **Hệ thống con:** Hệ thống quản lý nhân sự |
| **Thiết kế bởi:** Lê Đức Minh 20200395 | **Ngày thiết kế:** 20/12/2023 |
| **Thực thi bởi:** SqliteRequestRespository | **Ngày thực thi:** 20/12/2023 |
| **Mô tả ngắn gọn:** Kiểm tra phương thức truy vấn cơ sở dữ liệu với kết quả trả về thành công |  |

|  |
| --- |
| **Tiền điều kiện:**  Hệ thống đang hiển thị giao diện Xem thông tin yêu cầu sửa thông tin chấm công  Dữ liệu đầu vào hợp lệ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bước | Hành động | Phản hồi mong muốn từ hệ thống | Pass/Fail | Comment |
| 1 | Người dùng nhập mã nhân viên rỗng và ngày hợp lệ | Hệ thống hiển thị yêu cầu của tất cả nhân viên | Pass |  |
| 2 | Người dùng nhập mã nhân viên đúng và ngày hợp lệ | Hệ thống hiển thị danh sách tương ứng ngày và nhân viên đó | Pass |  |
| 3 | Người dùng chọn ngày bằng 0 và mã nhân viên rỗng | Hệ thống hiển thị yêu cầu của tất cả nhân viên trong tháng | Pass |  |
| 4 | Người dùng chọn ngày hợp lệ và mã nhân viên đúng | Hệ thống hiển thị yêu cầu của nhân viên trong ngày hôm đó | Pass |  |

|  |
| --- |
| **Trạng thái của hệ thống sau khi chạy hàm:**  Hệ thống hiển thị danh sách các yêu cầu thay đổi thông tin chấm công tương ứng với thông tin tìm kiếm |

**Test case Query\_InvalidData**

|  |  |
| --- | --- |
| **Test case #:** 1.2 | **Tên test case:** Query |
| **Hệ thống:** Hệ thống quản lý chấm công | **Hệ thống con:** Hệ thống quản lý nhân sự |
| **Thiết kế bởi:** Lê Đức Minh 20200395 | **Ngày thiết kế:** 20/12/2023 |
| **Thực thi bởi:** SqliteRequestRespository | **Ngày thực thi:** 20/12/2023 |
| **Mô tả ngắn gọn:** Kiểm tra phương thức truy vấn cơ sở dữ liệu với kết quả trả về không thành công |  |

|  |
| --- |
| **Tiền điều kiện:**  Hệ thống đang hiển thị giao diện Xem thông tin chấm công nhân viên văn phòng  Dữ liệu đầu không hợp lệ (month = 1, year = 2024) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bước | Hành động | Phản hồi mong muốn từ hệ thống | Pass/Fail | Comment |
| 1 | Người dùng nhập tháng là số âm | Hệ thống trả về danh sách yêu cầu là rỗng | Pass |  |
| 2 | Người dùng nhập tháng là số nguyên dương lớn hơn 31 | Hệ thống trả về yêu cầu là rỗng | Fail |  |
| 3 | Người dùng nhập tháng là số nguyên trong khoảng 0-31 và không tương thích với tháng | Hệ thống trả về yêu cầu là rỗng | Fail |  |

|  |
| --- |
| **Trạng thái của hệ thống sau khi chạy hàm:**  Hệ thống trở lại trạng thái ban đầu. |