ĐẠI HỌC BÁCH KHOA HÀ NỘI

**TRƯỜNG CNTT & TT**

----- 🙡 🕮 🙣 -----



**LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

***Đề tài:* Website quản lý dự án**

Giảng viên: **Lê Đức Hậu**

Nhóm sinh viên thực hiện:

|  |  |  |
| --- | --- | --- |
| STT | Họ và tên | MSSV |
| 1 | Nguyễn Bùi Tuấn Linh | 20225732 |
| 2 | Nguyễn Bùi Việt Linh | 20225733 |
| 3 | Nguyễn Khánh Toàn | 20225936 |
| 4 | Hồ Tuấn Huy | 20225856 |
| 5 | Nguyễn Quỳnh Anh | 20225785 |
| 6 | Phạm Quốc Cường | 20225604 |

**Hà Nội, năm 2024**

**Đánh giá đóng góp của mỗi thành viên**

|  |  |  |  |
| --- | --- | --- | --- |
| **Họ và tên** | **MSSV** | **Công việc** | **Đánh giá** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[**Lời nói đầu** 4](#_Toc185386560)

[**Chương 1: Giới thiệu chung** 5](#_Toc185386561)

[**1.1** **Yêu cầu đặt ra** 5](#_Toc185386562)

[**1.2 Hướng giải quyết** 6](#_Toc185386563)

[**Chương 2: Thiết kế và xây dựng phần mềm** 8](#_Toc185386564)

[**2.1 Biểu đồ phụ thuộc gói** 8](#_Toc185386565)

[**2.2 Biểu đồ lớp** 8](#_Toc185386566)

[2.2.1. Các lớp trong gói Data Layer 8](#_Toc185386567)

[2.2.2. Các lớp trong gói Business Layer 10](#_Toc185386568)

[2.2.3. Các lớp trong gói Presentation Layer 12](#_Toc185386569)

[**Chương 3: Các kỹ thuật lập trình hướng đối tượng đã áp dụng** 13](#_Toc185386570)

[3.1 Đóng gói 13](#_Toc185386571)

[3.2 Trừu tượng 13](#_Toc185386572)

[3.3 Kế thừa 14](#_Toc185386573)

[3.4 Đa hình 14](#_Toc185386574)

# **Lời nói đầu**

Hiện nay, trong bối cảnh thị trường ngày càng cạnh tranh và phát triển nhanh chóng, các tổ chức và doanh nghiệp đang ngày càng phụ thuộc vào sự hiệu quả trong việc quản lý dự án để đạt được mục tiêu chiến lược và duy trì lợi thế cạnh tranh. Quản lý dự án không chỉ đơn thuần là việc phân bổ công việc hay giám sát tiến độ, mà còn là một quá trình phức tạp đòi hỏi sự phối hợp chặt chẽ giữa các thành viên trong đội ngũ, sự minh bạch trong thông tin, và việc sử dụng các nguồn lực một cách hợp lý. Để đạt được điều này, các công cụ hỗ trợ mạnh mẽ và chuyên nghiệp là yếu tố không thể thiếu nhằm tối ưu hóa quy trình, cải thiện hiệu suất làm việc và đảm bảo dự án được hoàn thành đúng thời hạn, trong phạm vi ngân sách và đạt được chất lượng mong đợi.

Trong bối cảnh đó, phần mềm quản lý dự án đã trở thành một giải pháp quan trọng, mang tính cách mạng, giúp các tổ chức và doanh nghiệp tổ chức công việc một cách khoa học, theo dõi tiến độ chặt chẽ, và đảm bảo việc phân bổ nguồn lực được thực hiện một cách hiệu quả nhất. Các phần mềm này không chỉ giúp quản lý thời gian và chi phí, mà còn hỗ trợ trong việc giao tiếp giữa các bên liên quan, phân tích dữ liệu để đưa ra các dự đoán chính xác và hỗ trợ ra quyết định trong suốt vòng đời của dự án. Với nhu cầu ngày càng cao từ thị trường, việc phát triển một phần mềm quản lý dự án hiện đại, hiệu quả và thân thiện với người dùng đã trở thành xu hướng tất yếu.

Xuất phát từ những nhu cầu thực tiễn và tầm quan trọng của quản lý dự án trong thời đại số, chúng em đã quyết định lựa chọn đề tài "**Phần mềm quản lý dự án**" với mục tiêu xây dựng một hệ thống hoàn chỉnh, có giao diện thân thiện, dễ sử dụng và đáp ứng được các yêu cầu cơ bản của một công cụ quản lý dự án hiện đại. Phần mềm này không chỉ hỗ trợ các chức năng như lập kế hoạch, phân công công việc, theo dõi tiến độ, mà còn tích hợp các tính năng phân tích dữ liệu, báo cáo thông minh và khả năng tùy chỉnh để phù hợp với từng loại hình dự án khác nhau. Báo cáo này sẽ trình bày chi tiết về các kết quả mà chúng em đã đạt được trong quá trình nghiên cứu, thiết kế và triển khai hệ thống.

Bên cạnh đó, chúng em cũng muốn bày tỏ lòng biết ơn sâu sắc đến thầy Lê Đức Hậu. Những bài giảng cuốn hút, dễ hiểu, những kinh nghiệm thực tế mà thầy chia sẻ không chỉ giúp chúng em giải quyết những khó khăn trong quá trình thực hiện mà còn đem lại cho chúng em những góc nhìn mới, đóng góp quan trọng vào sự hoàn thiện của sản phẩm. Chính nhờ những điều đó, chúng em đã có thể vượt qua các thách thức và đạt được những kết quả như mong đợi.

# **Chương 1: Giới thiệu chung**

## **Yêu cầu đặt ra**

**Mục tiêu chung:**

Xây dựng hệ thống quản lý dự án nhằm tối ưu hóa quy trình làm việc, theo dõi tiến độ, và phân chia nhiệm vụ trong các dự án. Hệ thống cần đảm bảo chức năng quản lý người dùng, công việc, dự án và các tương tác khác như thanh toán và tin nhắn.

**Yêu cầu cụ thể:**

1. **Quản lý người dùng**:
   1. Cho phép người dùng đăng ký, đăng nhập và phân quyền dựa trên **Role** (ADMIN/USER).
   2. Quản lý thông tin cá nhân người dùng, bao gồm tên đầy đủ, tên đăng nhập, địa chỉ, trạng thái tài khoản, và hình ảnh đại diện.
   3. Đảm bảo an toàn tài khoản thông qua các trạng thái như khóa/không khóa, hết hạn/không hết hạn.
2. **Quản lý dự án**:
   1. Cho phép tạo, sửa, xóa các dự án với thông tin như **tên dự án, mô tả, trạng thái, và hình ảnh minh họa**.
   2. Mỗi dự án bao gồm nhiều **Task** và liên kết với người thực hiện.
   3. Cho phép quản lý danh sách dự án theo trạng thái.
3. **Quản lý công việc và nhiệm vụ con**:
   1. Mỗi dự án chứa danh sách các **Task** và mỗi Task có thể chia thành **Sub Task**.
   2. Quản lý thông tin công việc như tiêu đề, mô tả, mức độ ưu tiên, trạng thái, và hình ảnh minh họa.
   3. Gán công việc cho người thực hiện và theo dõi tiến độ.
4. **Bình luận và tương tác**:
   1. Người dùng có thể thêm bình luận vào các **Task** để trao đổi thông tin.
   2. Mỗi bình luận cần ghi nhận **nội dung** và người viết.
5. **Quản lý giao dịch thanh toán**:
   1. Hệ thống ghi nhận người dùng đăng ký thành viên để sử dụng toàn bộ chức năng của hệ thống.
6. **Tin nhắn nội bộ**:
   1. Cho phép người dùng gửi tin nhắn nội bộ dự án, bao gồm **nội dung tin nhắn** và **người gửi**.
7. **Yêu cầu chung**:
   1. Đảm bảo tính mở rộng và bảo trì dễ dàng.
   2. Áp dụng các chức năng cơ bản chung như ghi nhận thời gian tạo và cập nhật dữ liệu.

## **1.2 Hướng giải quyết**

Để giải quyết bài toán này, hệ thống sẽ được thiết kế và phát triển theo mô hình **3 Layer (3 lớp)** và áp dụng các kỹ thuật phát triển phần mềm hiện đại.

#### **Phân chia chức năng chính:**

1. **Tầng giao diện (Presentation Layer)**:
   1. Xây dựng giao diện người dùng trực quan bằng **Vue.js.**
   2. Các màn hình chức năng chính:
      1. Đăng ký, đăng nhập.
      2. Quản lý người dùng (thông tin cá nhân).
      3. Bảng điều khiển dự án.
      4. Trang thông tin quản lý task được giao.
      5. Tạo, chỉnh sửa, và quản lý Task, Sub Task.
      6. Màn hình bình luận và tin nhắn nội bộ.
      7. Quản lý thanh toán và thông tin giao dịch.
2. **Tầng dịch vụ (Service Layer)**:
   1. Phát triển các API bằng **Spring Boot** để xử lý các yêu cầu từ giao diện:
      1. Quản lý người dùng và phân quyền.
      2. CRUD cho các dự án, Task và Sub Task.
      3. Chức năng bình luận và tin nhắn.
      4. Quản lý thanh toán.
   2. Áp dụng các **RESTful APIs** để đảm bảo hệ thống mở rộng dễ dàng.
3. **Tầng dữ liệu (Data Access Layer)**:
   1. Sử dụng **MySQL** để lưu trữ dữ liệu.
   2. Thiết kế các bảng tương ứng với các entity trong UML:
      1. User, Role, Project, Task, Sub Task, Comment, Message, PaymentTransaction.
   3. Áp dụng các quan hệ 1-n và n-n

# **Chương 2: Thiết kế và xây dựng phần mềm**

## **2.1 Biểu đồ phụ thuộc gói**

**A diagram of a diagram

Description automatically generated**

Sự liên kết của các gói trên thể hiện cho mô hình 3-Layer mà ứng dụng đã được xây dựng dựa theo. Với Controllers là gói gốc chứa các class giao tiếp với giao diện của ứng dụng, nơi sẽ gọi đến service để thực thi các hàm, các hàm được gọi tùy theo sự kiện được kích hoạt khi người dùng thao tác với giao diện. Services là gói kiểm soát các hoạt động hệ thống của chương trình, liên kết tới các gói khác để thực hiện chức năng như đọc dữ liệu, tìm kiếm, sắp xếp. Những hàm này đều phải lấy dữ liệu từ Database, nên đó là lí do Services sẽ liên kết với Repository (chứa hàm lấy dữ liệu từ Database) và models.

## **2.2 Biểu đồ lớp**

### 2.2.1. Các lớp trong gói Data Layer

- Gói con repository:

A screenshot of a computer screen

Description automatically generated

Hình 1: Biểu đồ lớp cho gói repository

Interface **JpaRepository** thuộc **Spring Data JPA**, một phần của **Spring Framework** cung cấp các chức năng và phương thức linh hoạt để tương tác với cơ sở dữ liệu. Các repository con định nghĩa thêm các phương thức truy vấn tùy chỉnh để phục vụ cho từng chức năng cụ thể trong ứng dụng đối với từng đối tượng.

- Gói con models:

A diagram of a computer

Description automatically generated

* Gói model bao gồm các lớp đại diện cho các thực thể được thu thập. Có 10 lớp như sau:

|  |  |
| --- | --- |
| **Tên gói** | **Vai trò** |
| BaseEntity | Lớp cơ sở cho các đối tượng được tạo |
| User | Lớp đại diện cho người dùng |
| Role | Lớp đại diện cho quyền của người dùng |
| PaymentTransaction | Lớp đại diện cho giao dịch thanh toán |
| Project | Lớp đại diện cho Project được quản lý |
| Task | Lớp đại diện cho Task thuộc Project |
| Comment | Lớp đại diện cho Comment trong Task |
| Message | Lớp đại diện cho tin nhắn của User trong Project |
| SubTask | Lớp đại diện cho các Task phụ |

### 2.2.2. Các lớp trong gói Business Layer

**-** Gói con service:

A screenshot of a computer screen

Description automatically generated

Gói con service sẽ chứa các hàm cung cấp các chức năng chính, xử lý dữ liệu đối với từng đối tượng cụ thể.

* Gói con converter:

A screenshot of a computer program

Description automatically generated

Gói này cung cấp các lớp hỗ trợ cho việc chuyển đổi giữa các đối tượng

* Gói con utils:

A close-up of a card

Description automatically generated

Cung cấp các chức năng phụ trợ cho service

### 2.2.3. Các lớp trong gói Presentation Layer

* Gói controller:

A screenshot of a computer program

Description automatically generated

Gói có chức năng điều hướng giao diện và xử lý sự kiện.

* Gói DTOs:

A diagram of a computer

Description automatically generated

DTO là các lớp không chứa **business logic**và chỉ dùng để lưu trữ dữ liệu. Dựa vào dữ liệu cần nhận và dữ liệu cần gửi, DTOs đóng gói dữ liệu và chuyển đổi dữ liệu giữa Frontend và Backend.

# **Chương 3: Các kỹ thuật lập trình hướng đối tượng đã áp dụng**

## 3.1 Đóng gói

Tính đóng gói là một trong những nguyên tắc quan trọng của lập trình hướng đối tượng. Nó đề cập đến việc ẩn thông tin và chi tiết triển khai của một đối tượng khỏi bên ngoài, chỉ cho phép truy cập thông qua các phương thức public. Tính chất này được chúng em áp dụng cho tất cả các đối tượng ở trong chương trình.

## 3.2 Trừu tượng

Tính trừu tượng nói đến việc ẩn đi chi tiết triển khai cụ thể của một đối tượng và chỉ hiển thị các khía cạnh quan trọng và cần thiết.

## 3.3 Kế thừa

Kế thừa là một khái niệm quan trọng trong lập trình hướng đối tượng. Kế thừa giúp tái sử dụng mã nguồn, giảm độ phức tạp của mã, và tăng tính linh hoạt. Lớp dẫn xuất có thể thừa hưởng tất cả hoặc một phần các đặc tính và hành vi của lớp cơ sở, và sau đó mở rộng hoặc ghi đè lên chúng nếu cần. Tính chất này được chúng em sử dụng nhiều trong sản phẩm phần mềm của nhóm.

Điển hình, trong bài này tính kế thừa được sử dụng trong lớp repository.

Qua đó, ta có thể thấy rằng: việc kế thừa sẽ khiến sản phẩm vừa linh hoạt lại dễ dàng hơn trong khâu bảo trì, sửa đổi.

## 3.4 Đa hình

Đa hình là một khái niệm quan trọng trong lập trình hướng đối tượng, nó thể hiện khả năng của một đối tượng có thể nhận biết và thực hiện các hành vi khác nhau tùy thuộc vào ngữ cảnh.

Có hai loại đa hình: đa hình tại thời điểm biên dịch (compile-time polymorphism) và đa hình tại thời điểm chạy (runtime polymorphism).

Đa hình tại thời điểm biên dịch (Compile-time Polymorphism): Được thực hiện thông qua quá tải toán tử (operator overloading) và quá tải hàm (function overloading). Trong trường hợp này, trình biên dịch quyết định được loại hàm hoặc toán tử nào sẽ được gọi dựa trên các tham số đầu vào và kiểu dữ liệu tại thời điểm biên dịch.

Đa hình tại thời điểm chạy (Runtime Polymorphism): Thường được thực hiện thông qua kỹ thuật gọi đa hình và thừa kế. Trong trường hợp này, đối tượng có thể thay đổi hành vi của nó tại thời điểm chạy. Các ví dụ phổ biến là sử dụng interfaces và abstract classes trong Java hoặc C#.

Ví dụ, bạn có thể có một lớp cơ sở như Shape và các lớp con như Circle và Rectangle. Tất cả chúng có thể có phương thức draw(), nhưng mỗi lớp con có thể triển khai nó theo cách riêng biệt. Khi bạn gọi phương thức draw() trên một đối tượng Shape, hành vi cụ thể sẽ được quyết định tại thời điểm chạy dựa trên loại của đối tượng đó.

Đa hình giúp tạo ra mã nguồn linh hoạt và giảm sự phụ thuộc vào các chi tiết triển khai cụ thể.