ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Nền tảng quản lý API Document

Hoàng Thế Anh
Anh.ht203948@sis.hust.edu.vn

Ngành Công nghệ thông tin

Giảng viên hướng dẫn: ThS. Nguyễn Mạnh Tuấn _____

Chữ kí GVHD

Khoa: Khoa học máy tính

Trường: Công nghệ Thông tin và Truyền thông

HÀ NỘI, 06/2024

LỜI CAM KẾT

Họ và tên sinh viên: Hoàng Thế Anh
Điện thoại liên lạc: 0981934729
Email: Anh.ht204938@sis.hust.edu.vn
Lớp: Việt Nhật 04 - K65
Hệ đào tạo: Công nghệ thông tin Việt - Nhật

Tôi – *Hoàng Thế Anh* – cam kết Đồ án Tốt nghiệp (ĐATN) là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *ThS. Nguyễn Mạnh Tuấn*. Các kết quả nêu trong ĐATN là trung thực, là thành quả của riêng tôi, không sao chép theo bất kỳ công trình nào khác. Tất cả những tham khảo trong ĐATN – bao gồm hình ảnh, bảng biểu, số liệu, và các câu từ trích dẫn – đều được ghi rõ ràng và đầy đủ nguồn gốc trong danh mục tài liệu tham khảo. Tôi xin hoàn toàn chịu trách nhiệm với dù chỉ một sao chép vi phạm quy chế của nhà trường.

Hà Nội, ngày 30 tháng 6 năm 2024

Tác giả ĐATN

Họ và tên sinh viên

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành tới những người thân trong gia đình, những người bạn đã luôn bên cạnh, động viên và hỗ trợ em trong suốt quá trình thực hiện đồ án tốt nghiệp. Cảm ơn vì mọi người đã ở bên em những lúc khó khăn nhất, những lời quan tâm và khích lệ của mọi người là nguồn động lực lớn giúp em vượt qua những khó khăn, thử thách và hoàn thành kỳ đồ án một cách suôn sẻ.

Em xin cảm ơn các thầy cô Đại học Bách Khoa Hà Nội, đặc biệt là thầy Nguyễn Mạnh Tuấn. Cảm ơn thầy đã quan tâm, hướng dẫn tận tình em suốt những kỳ học vừa qua. Cảm ơn thầy đã dẫn lối, truyền đạt kiến thức, chia sẻ những kinh nghiệm quý báu về chuyên môn và nghiệp vụ vô cùng thực tế giúp em hoàn thành đồ án tốt nghiệp này.

Em xin chân thành cảm ơn!.

TÓM TẮT NỘI DUNG ĐỒ ÁN

Quản lý API là các hoạt động liên quan đến việc tổ chức và quản lý các tài liệu API, hỗ trợ quá trình phát triển, bảo trì và mở rộng API một cách hiệu quả. Việc quản lý API không chỉ giúp đảm bảo tính nhất quán và chất lượng của API mà còn giúp tăng cường sự hợp tác và giao tiếp giữa các thành viên trong dự án.

Trong các dự án phần mềm, quản lý API là rất quan trọng. Tuy nhiên, việc theo dõi và duy trì tài liệu API có thể tốn nhiều thời gian và công sức. Các hệ thống quản lý API hiện tại mặc dù có nhiều ưu điểm nhưng vẫn tồn tại nhiều hạn chế. Các hệ thống có sẵn thường chỉ tập trung vào việc quản lý tài liệu API mà bỏ qua các quy trình phát triển, dẫn đến việc không khai thác hết được tầm quan trọng và tính tiện lợi của tài liệu API . Từ thực tiễn đó, việc phát triển một hệ thống quản lý API hiệu quả đã trở thành nhu cầu cấp thiết.

Đồ án này tập trung vào việc xây dựng một website quản lý tài liệu API cho dự án, hỗ trợ phát triển, bảo trì và mở rộng API theo bốn giai đoạn chính: Define, Design, Develop, và Test. Mỗi giai đoạn đều có các chức năng tương ứng, từ đó giúp quy trình phát triển và bảo trì API trở nên hiệu quả hơn.

Ngoài ra, website còn cung cấp các chức năng phụ như dashboard và scheduler để theo dõi due date của các task, quản lý issue, quản lý database, quản lý file chung của dự án và quản lý thành viên dự án. Những chức năng này giúp dự án được quản lý một cách toàn diện, từ việc phát triển ban đầu cho đến bảo trì và mở rộng sau này.

Đồ án này nhằm khắc phục các vấn đề còn tồn tại trong việc quản lý tài liệu API, giúp các doanh nghiệp phát triển phần mềm dễ dàng hơn trong việc quản lý và duy trì API, đồng thời tăng cường hiệu quả và chất lượng của quá trình phát triển, bảo trì và mở rộng phần mềm. Nhờ vào hệ thống này, các doanh nghiệp có thể tối ưu hóa quy trình làm việc, giảm thiểu rủi ro và tăng cường khả năng phát triển bền vững trong tương lai.

Sinh viên thực hiện

(Ký và ghi rõ họ tên)

ABSTRACT

API management encompasses activities related to organizing and managing API documentation, supporting the effective development, maintenance, and expansion of APIs. Effective API management ensures consistency and quality of APIs while enhancing collaboration and communication among project members.

In software projects, API management is crucial. However, tracking and maintaining API documentation can be time-consuming and labor-intensive. Existing API management systems, despite their many advantages, still have significant limitations. They often focus solely on managing API documentation while neglecting the development processes, failing to fully exploit the importance and convenience of API documentation. This practical need has led to the urgent development of an effective API management system.

This project focuses on building a website for managing API documentation for projects, supporting the development, maintenance, and expansion of APIs through four main stages: Define, Design, Develop, and Test. Each stage is equipped with corresponding functions to make the API development and maintenance process more efficient.

Additionally, the website offers auxiliary functions such as a dashboard and scheduler to track the due dates of tasks, manage issues, manage databases, manage common project files, and manage project members. These functions ensure comprehensive project management, from initial development to subsequent maintenance and expansion.

This project aims to address existing issues in API documentation management, facilitating easier management and maintenance of APIs for software development companies while enhancing the efficiency and quality of the development, maintenance, and expansion processes. Through this system, companies can optimize workflows, minimize risks, and enhance sustainable development capabilities in the future.

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Thuật ngữ	Ý nghĩa
API	Giao diện lập trình ứng dụng (Application Programming Interface)
API lifecycle	Vòng đời của API (API lifecycle)
BA	Chuyên viên phân tích nghiệp vụ (Business Analyst)
Backend	Phần phía sau của ứng dụng, chịu trách nhiệm xử lý logic và quản lý cơ sở dữ liệu (Backend)
CI/CD	Continuous Integration/Continuous Deployment
Client	Thiết bị hoặc ứng dụng gửi yêu cầu đến máy chủ (Client)
Database	Cơ sở dữ liệu
Define	Giai đoạn định nghĩa
Design	Giai đoạn thiết kế
DEV	Nhà phát triển (Developer)
Develop	Giai đoạn phát triển
Frontend	Phần phía trước của ứng dụng, chịu trách nhiệm hiển thị giao diện và tương tác với người dùng (Frontend)
Issue	Vấn đề hoặc lỗi phát sinh cần giải quyết (Issue)
JWT	JSON Web Token
Server	Máy chủ, thiết bị hoặc chương trình máy tính cung cấp dịch vụ cho các máy tính khác (Server)
Task	Nhiệm vụ hoặc công việc cần hoàn thành (Task)
Test	Giai đoạn kiểm thử

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề

Trong thời đại số hóa hiện nay, các doanh nghiệp ngày càng phụ thuộc vào công nghệ thông tin để quản lý và vận hành các quy trình kinh doanh. Một trong những yếu tố then chốt của các hệ thống phần mềm hiện đại là các API (Application Programming Interface). API không chỉ giúp kết nối và trao đổi dữ liệu giữa các ứng dụng mà còn đóng vai trò quan trọng trong việc phát triển và mở rộng hệ thống. Tuy nhiên, việc quản lý các API, đặc biệt là các tài liệu liên quan, gặp nhiều thách thức như không đồng bộ, thiếu thông tin chi tiết và khó theo dõi tiến độ.

Quy trình phát triển API (API lifecycle) [1] thường bao gồm nhiều giai đoạn: trong đó có định nghĩa (Define), thiết kế (Design), phát triển (Develop), kiểm thử (Test). Mỗi giai đoạn này yêu cầu sự quản lý cẩn thận để đảm bảo API hoạt động đúng như mong đợi, dễ bảo trì và mở rộng. Thách thức lớn nhất trong việc quản lý tài liệu API là đảm bảo tính nhất quán và cập nhật liên tục thông tin qua từng giai đoạn, đồng thời tạo điều kiện thuận lợi cho sự hợp tác giữa các thành viên trong dự án. Việc thiếu công cụ quản lý chi tiết và theo dõi tiến độ có thể dẫn đến sự chồng chéo trong công việc, giảm hiệu quả làm việc và tăng rủi ro lỗi hệ thống.

Xuất phát từ nhu cầu thực tế này, việc phát triển một hệ thống quản lý tài liệu API hiệu quả trở nên cấp thiết. Hệ thống này không chỉ giúp đảm bảo tính nhất quán và chất lượng của API mà còn tăng cường sự hợp tác giữa các thành viên trong dự án, từ đó nâng cao hiệu quả làm việc và giảm thiểu rủi ro. Việc quản lý tài liệu API một cách khoa học và hệ thống sẽ mang lại lợi ích lớn cho các doanh nghiệp phát triển phần mềm, giúp họ tiết kiệm thời gian, công sức và tối ưu hóa quy trình phát triển, bảo trì cũng như mở rộng các API.

Đề tài nghiên cứu và phát triển hệ thống quản lý tài liệu API Document nhằm giải quyết những thách thức này. Hệ thống sẽ hỗ trợ quản lý toàn bộ quy trình phát triển API, từ giai đoạn định nghĩa yêu cầu, thiết kế cấu trúc, phát triển mã nguồn, kiểm thử cho đến triển khai và giám sát. Với hệ thống này, các doanh nghiệp có thể đảm bảo tính nhất quán của tài liệu, tăng cường sự phối hợp giữa các thành viên trong dự án và nâng cao chất lượng của các API được phát triển.

1.2 Mục tiêu và phạm vi đề tài

Hiện nay, có nhiều hệ thống và công cụ hỗ trợ quản lý API trên thị trường, như Postman, Swagger, và API Gateway của AWS. Tuy nhiên, mỗi hệ thống đều có những hạn chế riêng, như khả năng tích hợp kém, khó khăn trong việc theo dõi và

cập nhật tài liệu, hoặc không hỗ trợ toàn diện quy trình phát triển và bảo trì API.

Trên cơ sở phân tích và đánh giá các hệ thống hiện tại, đề tài này hướng tới phát triển một website quản lý tài liệu API toàn diện cho các dự án phần mềm. Mục tiêu chính của hệ thống là khắc phục các hạn chế hiện có và cung cấp các chức năng chính sau: (1) Quản lý API theo bốn giai đoạn: Define, Design, Develop, và Test. Hỗ trợ thử nghiệm API và theo dõi tiến độ phát triển. (2) Tạo và quản lý task liên quan đến từng giai đoạn của API lifecycle. (3) Tích hợp quản lý thông tin các database server và các bảng. (4) Quản lý issue nhằm theo dõi, đánh giá và đưa ra giải pháp cho các lỗi, bug của dự án. (5) Cung cấp dashboard và scheduler để theo dõi due date của các task. (6) Quản lý file chung và thành viên của dự án. Hệ thống này không chỉ hỗ trợ việc phát triển API ban đầu mà còn giúp duy trì và mở rộng API một cách hiệu quả, đảm bảo tính liên tục và chất lượng của các dịch vụ phần mềm.

1.3 Định hướng giải pháp

Giải pháp được đưa ra là phát triển một website quản lý tài liệu API với giao diện người dùng thân thiện, tích hợp đầy đủ các công cụ và tiện ích hỗ trợ quản lý API qua các giai đoạn Define, Design, Develop và Test. Hệ thống này nhằm giải quyết các thách thức trong quản lý tài liệu API, đảm bảo tính nhất quán và chất lượng, tăng cường sự hợp tác và giao tiếp giữa các thành viên trong dự án, giảm thiểu rủi ro và tiết kiệm thời gian cho doanh nghiệp.

Các chức năng chính bao gồm:

1. **Define:** Thu thập và quản lý yêu cầu của API.
2. **Design:** Thiết kế chi tiết API, quản lý tài liệu và liên kết cơ sở dữ liệu.
3. **Develop:** Phát triển và thử nghiệm API, quản lý task phát triển.
4. **Test:** Kiểm thử API, gửi minh chứng và kiểm thử UI.
5. **Quản lý task:** Quản lý, giao nhiệm vụ cho các thành viên theo từng giai đoạn phát triển của API

Ngoài ra, hệ thống còn có chức năng khác như quản lý issue, quản lý daily report, quản lý thông tin database của dự án, quản lý các file chung của dự án, dashboard, scheduler của dự án.

Hệ thống này sẽ được xây dựng bằng React.js[2] cho frontend, Java Spring Boot[3] cho backend, cơ sở dữ liệu với MySQL [4], đóng gói bằng Docker[5], CI/CD bằng GitHub Actions[6] và authentication bằng JWT[7]. Kết quả là một hệ thống quản lý tài liệu API hoàn chỉnh, giúp doanh nghiệp tối ưu hóa quy trình phát

triển và bảo trì API.

1.4 Bố cục đồ án

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau:

Chương 2: Khảo sát và phân tích yêu cầu

Trong chương này, sẽ trình bày về khảo sát và phân tích nhu cầu quản lý API trong các dự án phần mềm hiện nay. Đánh giá mức độ đáp ứng của các công cụ quản lý API hiện có trên thị trường như Postman, Swagger, và API Gateway của AWS. Từ đó, rút ra những lợi thế và những bất cập của các công cụ này, đồng thời xác định các yêu cầu cần thiết để xây dựng một hệ thống quản lý API hiệu quả.

Chương 3: Công nghệ sử dụng

Chương này giới thiệu các công nghệ được sử dụng để xây dựng hệ thống quản lý API, bao gồm Java Spring Boot cho backend, React.js cho frontend, MySQL cho cơ sở dữ liệu và Docker để đóng gói ứng dụng.

Chương 4: Thiết kê, triển khai và đánh giá hệ thống

Chương này trình bày chi tiết quá trình triển khai và phát triển hệ thống quản lý API theo từng giai đoạn. Bố cục bao gồm: (i) thiết kế kiến trúc hệ thống, (ii) thiết kế chi tiết, (iii) xây dựng ứng dụng, (iv) kiểm thử và (v) triển khai ứng dụng.

Chương 5: Các giải pháp và đóng góp nổi bật

Chương này sẽ tập trung trình bày các giải pháp và đóng góp của hệ thống quản lý API đã xây dựng, cùng với những lợi ích thực tế khi áp dụng.

Chương 6: Kết luận và hướng Phát triển

Chương cuối cùng sẽ trình bày các kết quả đã đạt được trong quá trình thực hiện đồ án tốt nghiệp, cùng với định hướng phát triển hệ thống trong tương lai. Ngoài ra, chương này cũng sẽ chia sẻ những kinh nghiệm và bài học quý báu đã học được trong quá trình nghiên cứu và triển khai đồ án.

CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

2.1 Khảo sát hiện trạng

2.1.1 Nhu cầu của người dùng/khách hàng

Trong bối cảnh phát triển phần mềm hiện đại, việc quản lý API đóng vai trò quan trọng trong việc đảm bảo tính nhất quán và hiệu quả của các dịch vụ. Tuy nhiên, việc quản lý các API, đặc biệt là các tài liệu liên quan, gặp nhiều thách thức như không đồng bộ, thiếu thông tin chi tiết và khó theo dõi tiến độ, điều này tạo nên những nhu cầu từ người dùng:

- **Tính nhất quán và dễ dàng theo dõi:** Người dùng mong muốn có một hệ thống quản lý API giúp đảm bảo tính nhất quán trong việc phát triển, thử nghiệm và triển khai API. Họ cần một công cụ cho phép theo dõi và quản lý các thay đổi một cách dễ dàng.
- **Hỗ trợ toàn diện quy trình phát triển:** Người dùng mong muốn một hệ thống có thể hỗ trợ đầy đủ các giai đoạn của lifecycle API từ định nghĩa, thiết kế, phát triển đến kiểm thử.
- **Khả năng hợp tác và phân công công việc:** Một yếu tố quan trọng khác là khả năng tạo và quản lý các task liên quan đến từng giai đoạn của API lifecycle, giúp phân công công việc rõ ràng và tăng cường sự hợp tác giữa các thành viên trong nhóm.

2.1.2 Phân tích các hệ thống hiện có

Postman [1]

Ưu điểm: Postman là một công cụ mạnh mẽ cho việc thiết kế, thử nghiệm và tài liệu hóa API. Nó hỗ trợ việc tạo và gửi các request HTTP, đồng thời cung cấp giao diện người dùng thân thiện và dễ sử dụng.

Nhược điểm: Postman chủ yếu tập trung vào việc thử nghiệm API và không hỗ trợ đầy đủ các giai đoạn khác của lifecycle API. Khả năng tích hợp với các hệ thống quản lý dự án và công cụ khác cũng còn hạn chế.

Swagger [8]

Ưu điểm: Swagger cung cấp một bộ công cụ toàn diện cho việc thiết kế, xây dựng và tài liệu hóa API. Nó hỗ trợ tự động tạo tài liệu API từ mã nguồn và cung cấp giao diện tương tác cho việc thử nghiệm API.

Nhược điểm: Mặc dù Swagger rất mạnh mẽ trong việc tài liệu hóa và thử nghiệm API, nhưng nó thiếu khả năng quản lý task và theo dõi tiến độ phát triển

API.

API Gateway của AWS [9]

Ưu điểm: API Gateway của AWS cung cấp giải pháp mạnh mẽ cho việc quản lý và triển khai API trên nền tảng đám mây. Nó hỗ trợ việc quản lý traffic, kiểm soát truy cập, và giám sát hiệu năng API.

Nhược điểm: API Gateway của AWS yêu cầu kiến thức chuyên sâu về nền tảng AWS và có chi phí cao. Nó cũng thiếu các tính năng hỗ trợ cho việc quản lý task và theo dõi tiến độ phát triển API.

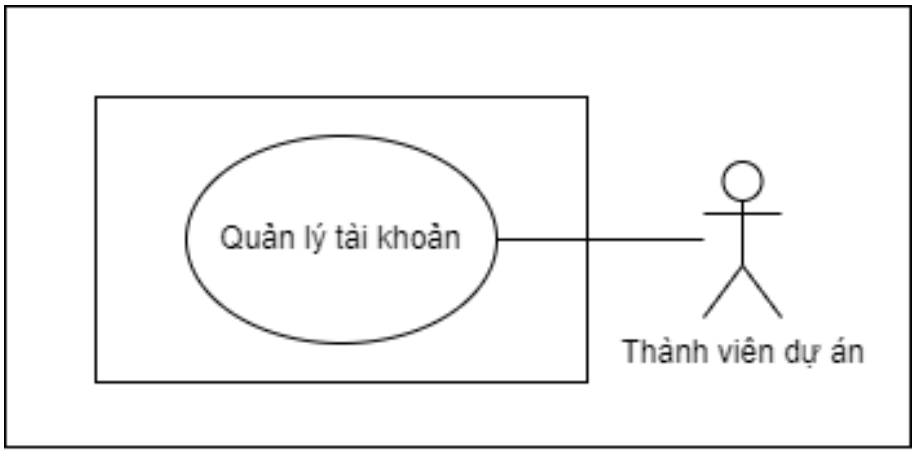
2.1.3 Tổng kết khảo sát

Dựa trên kết quả khảo sát người dùng và phân tích các công cụ hiện có, có thể thấy rằng nhu cầu về một hệ thống quản lý API toàn diện là rất lớn. Các công cụ hiện tại tuy có nhiều ưu điểm nhưng vẫn chưa đáp ứng đầy đủ các yêu cầu về quản lý toàn diện lifecycle API, khả năng tích hợp và quản lý task.

Những điểm hạn chế này cần được khắc phục để xây dựng một hệ thống quản lý API hiệu quả, hỗ trợ tốt cho việc phát triển, bảo trì và mở rộng API. Hệ thống này cần tích hợp các tính năng từ việc định nghĩa, thiết kế, phát triển đến kiểm thử API, đồng thời cung cấp khả năng quản lý task và theo dõi tiến độ một cách hiệu quả.

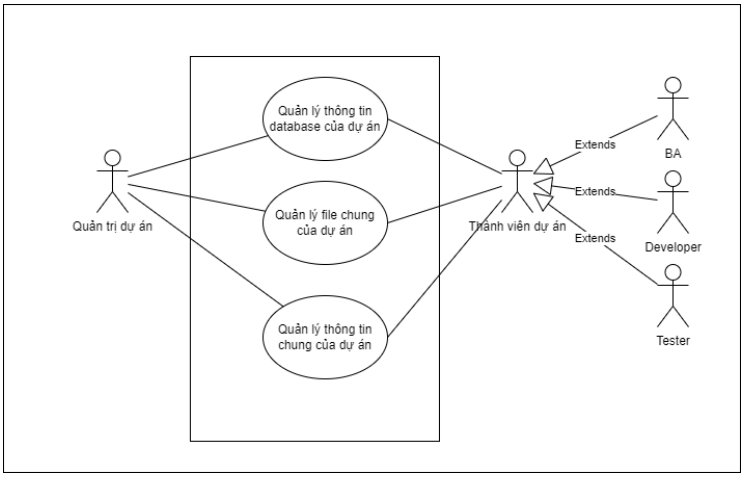
2.2 Tổng quan chức năng

2.2.1 Biểu đồ usecase tổng quát



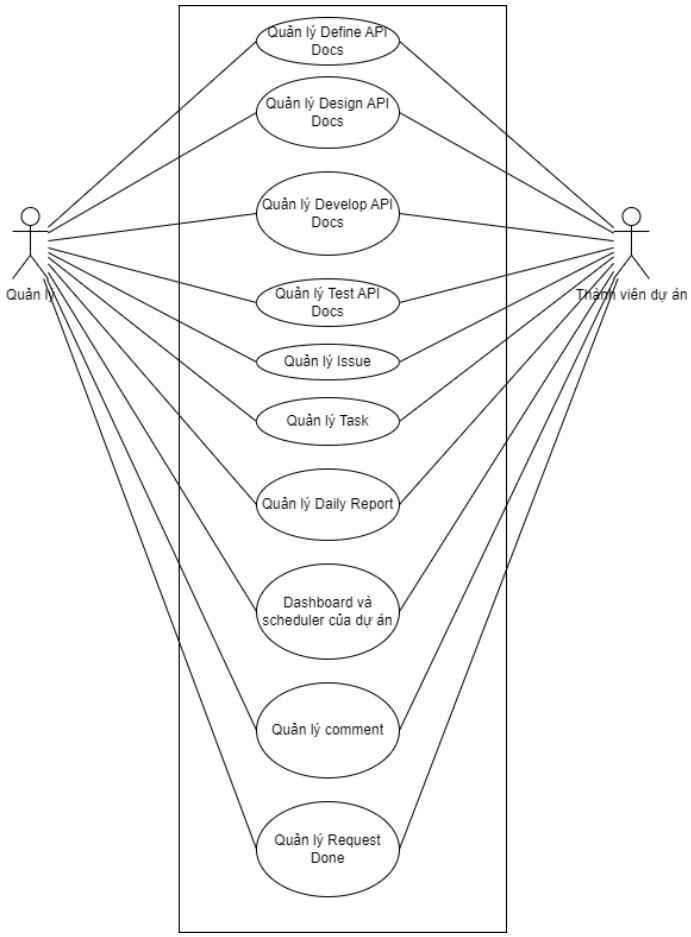
Hình 2.1: Module quản lý tài khoản

Người dùng có đầy đủ quyền với tài khoản cá nhân của mình



Hình 2.2: Module quản lý thông tin chung của dự án

Với vai trò quản lý dự án, người dùng có đầy đủ các quyền thêm, sửa, xóa các thông tin chung của dự án như thông tin database, file chung, thành viên và các thông tin khác của dự án. Với các thành viên, mọi người có quyền xem các thông tin chung của dự án. Các thành viên dự án ngoài quản lý dự án gồm BA, Dev, Tester

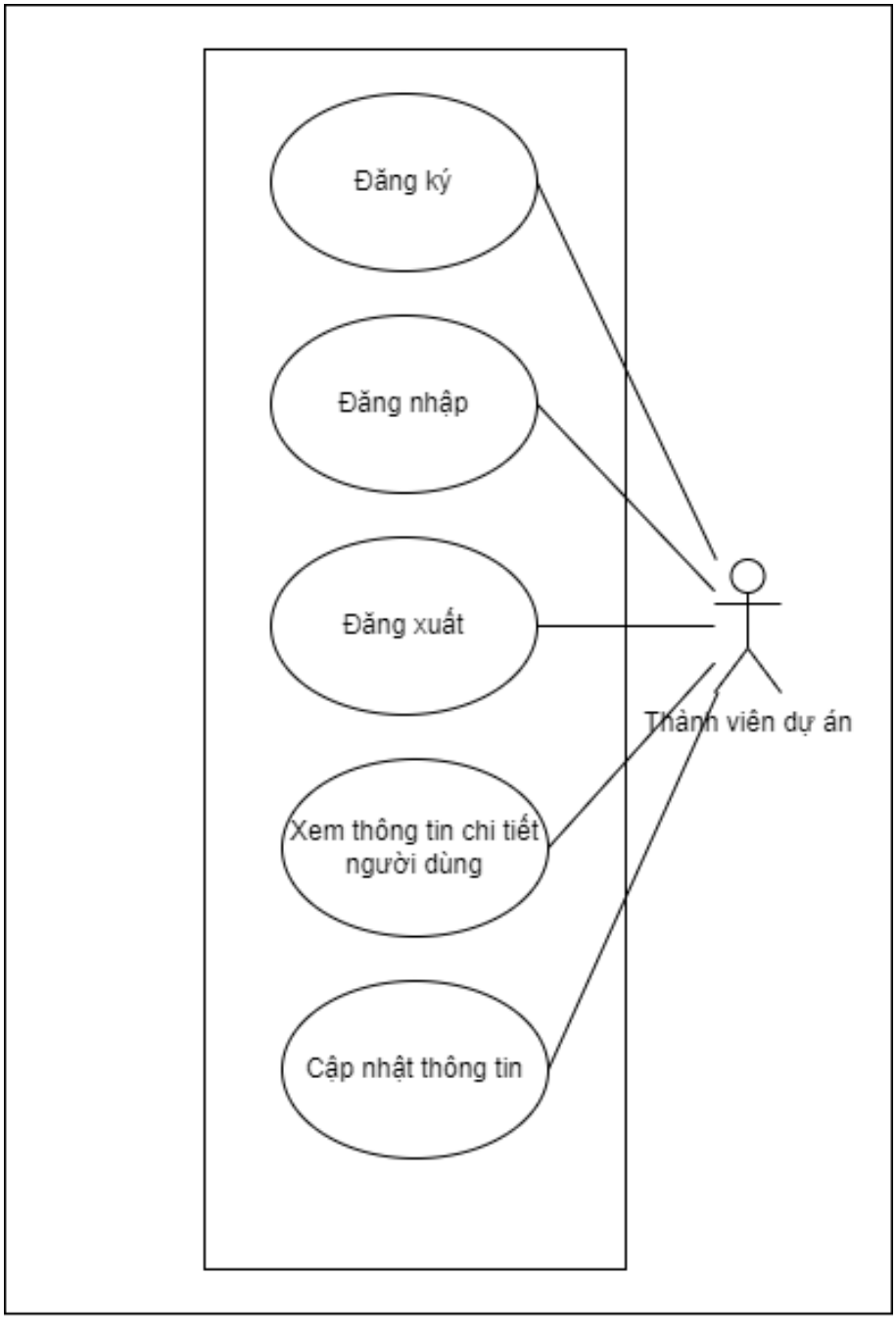


Hình 2.3: Module quản lý tài liệu và tiến độ dự án

Với vai trò quản lý dự án và thành viên dự án , người dùng có các quyền gồm

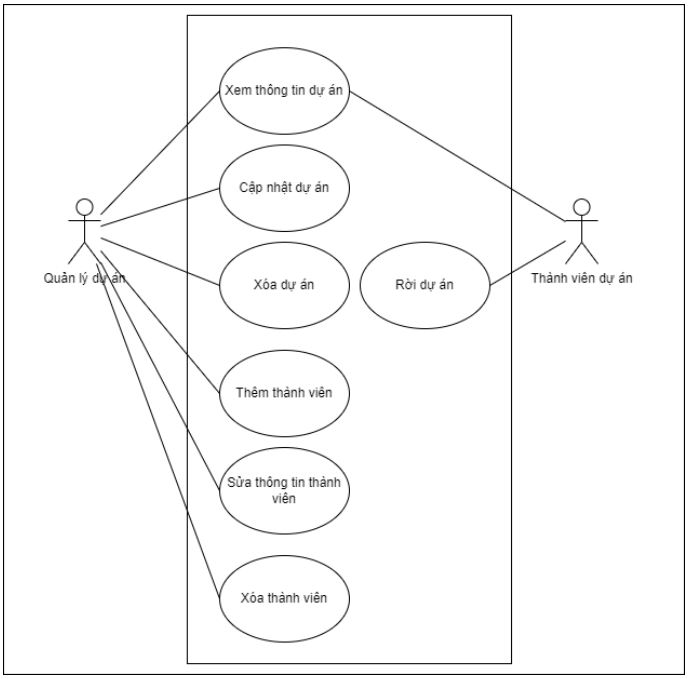
quản lý các thông tin gồm tài liệu khởi tạo API, tài liệu thiết kế API, tài liệu phát triển API, tài liệu test API, issue, task, daily report và xem dashboard và scheduler của dự án. Tuy nhiên, với vai trò thành viên, người dùng sẽ chỉ có các quyền hạn chế (xem, quyền sửa nếu quản lý dự án cho phép) Module quản lý tài liệu và tiến độ dự án gồm 8 usecase chính và 42 usecase khi phân rã

2.2.2 Biểu đồ usecase phân rã



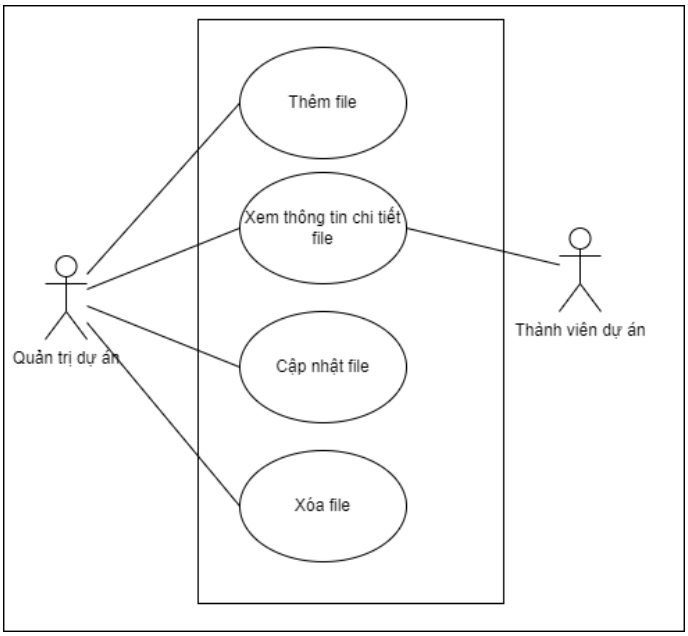
Hình 2.4: Biểu đồ phân rã usecase quản lý tài khoản

Mọi người dùng đều có quyền quản lý thông tin cá nhân của bản thân



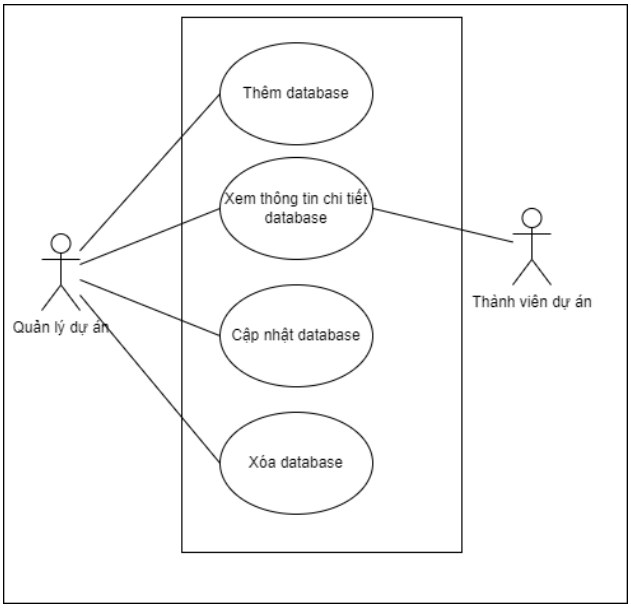
Hình 2.5: Biểu đồ phân rã usecase quản lý thông tin chung dự án

Quản lý dự án có quyền cập nhật lại các thông tin chung của dự án, thêm hoặc xóa thành viên cho dự án. Các thành viên còn lại có quyền xem các thông tin này và rời dự án.



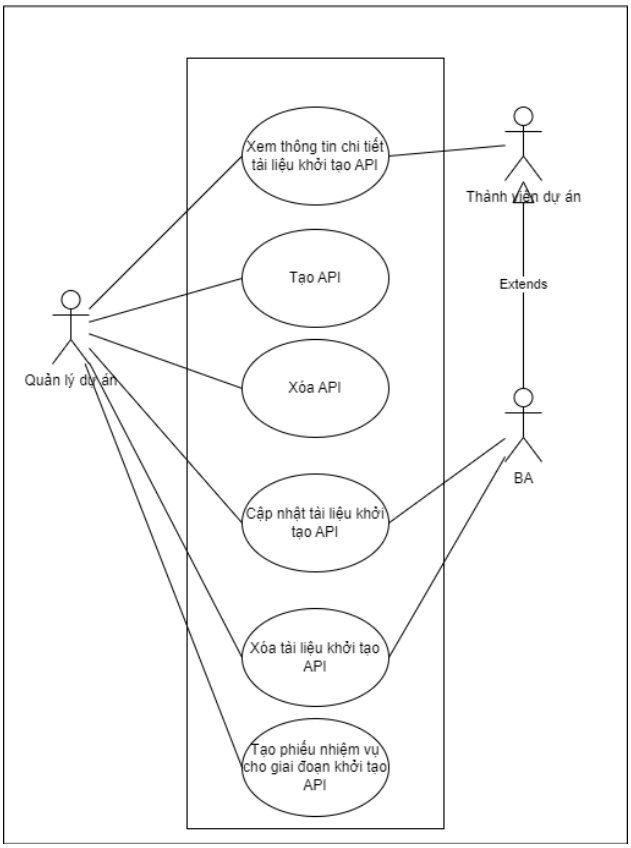
Hình 2.6: Biểu đồ phân rã usecase quản lý file chung của dự án

Quản lý dự án có quyền cập nhật lại các file chung của dự án. Các thành viên còn lại có quyền xem các thông tin này.



Hình 2.7: Biểu đồ phân rã usecase quản lý thông tin database của dự án

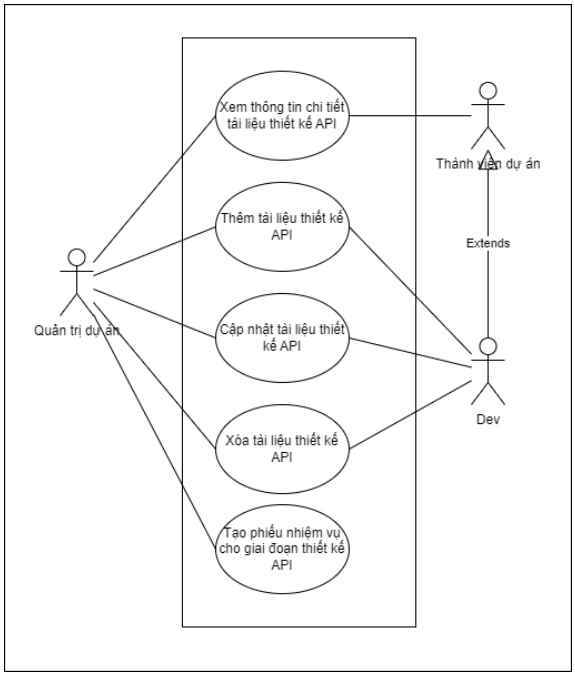
Quản lý dự án có quyền cập nhật lại thông tin database của dự án. Các thành viên còn lại có quyền xem các thông tin này.



Hình 2.8: Biểu đồ phân rã usecase quản lý tài liệu khởi tạo API

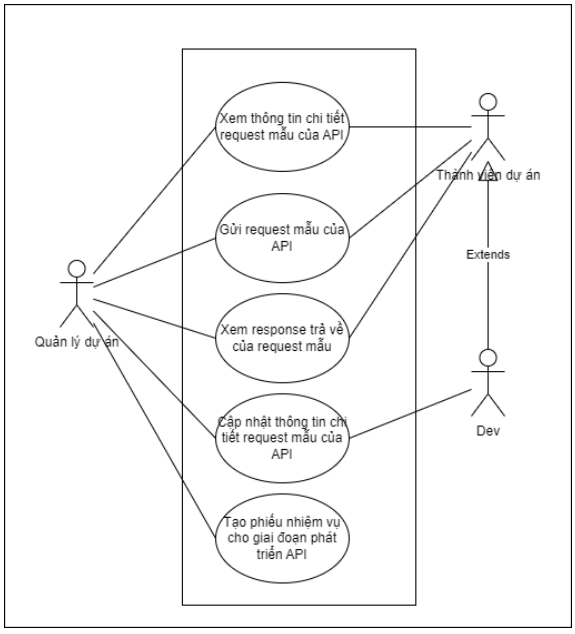
Quản lý dự án có quyền tạo, xóa api, cập nhật các tài liệu, tạo task cập nhật tài liệu khởi tạo API. BA được giao task có quyển cập nhật các tài liệu khởi tạo API.

Các thành viên còn lại có quyền xem các thông tin này.



Hình 2.9: Biểu đồ phân rã usecase quản lý tài liệu thiết kế API

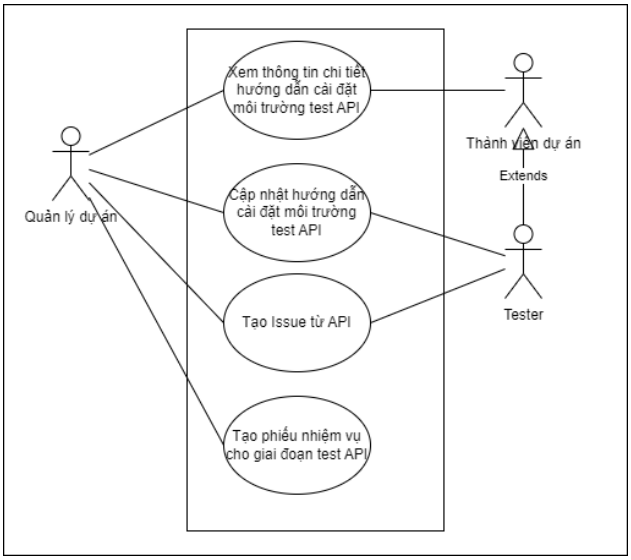
Quản lý dự án có quyền cập nhật các tài liệu thiết kế, tạo task. Dev được giao task có quyền cập nhật các tài liệu thiết kế API. Các thành viên còn lại có quyền xem các thông tin này.



Hình 2.10: Biểu đồ phân rã usecase quản lý tài liệu phát triển API

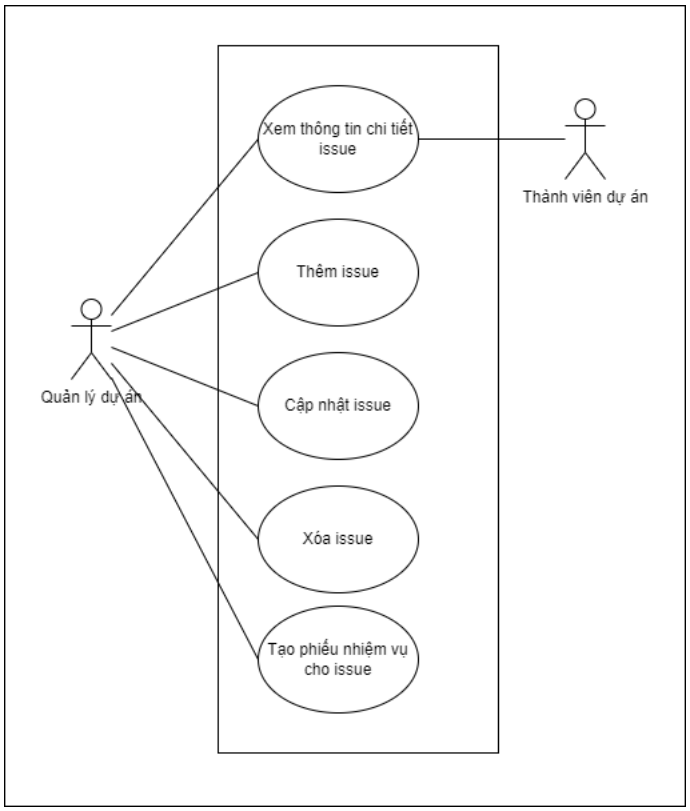
Quản lý dự án có đầy đủ các quyền gửi request, cập nhật thông tin request, xem response, tạo task. Dev được giao task có quyền cập nhật request mẫu. Các thành

viên còn lại có quyền gửi request và xem các thông tin này.



Hình 2.11: Biểu đồ phân rã usecase quản lý tài liệu test API

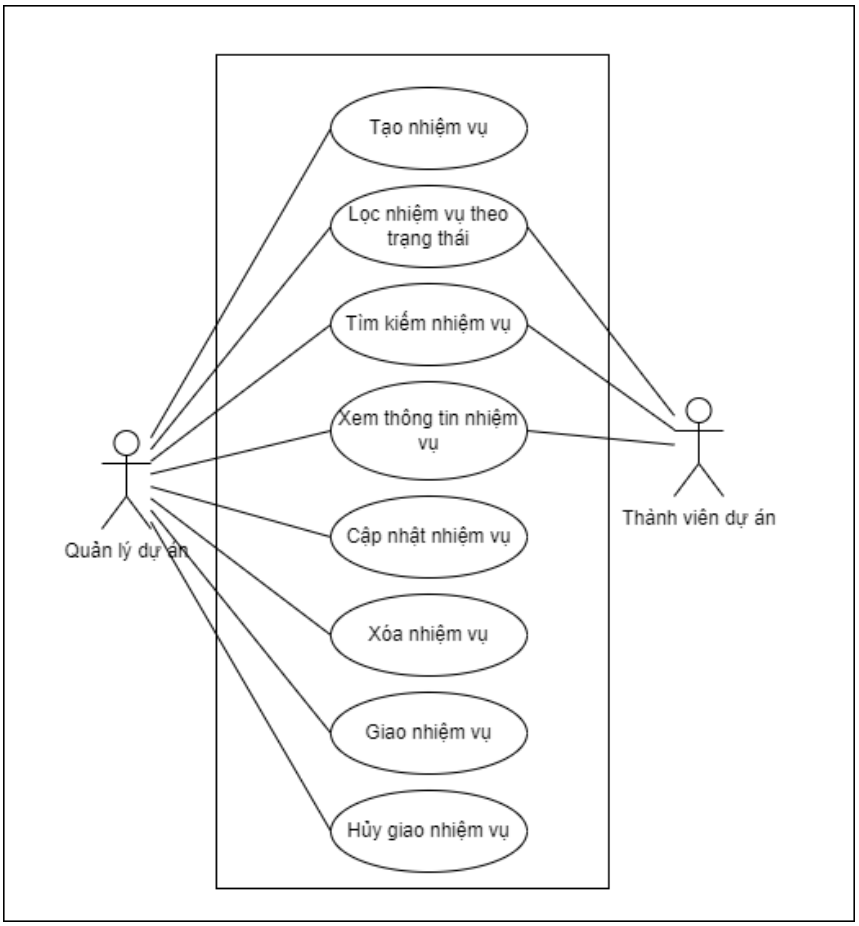
Quản lý dự án có quyền cập nhật các tài liệu test, tạo task. Tester được giao task có quyền cập nhật các tài liệu test API. Các thành viên còn lại có quyền xem các thông tin này.



Hình 2.12: Biểu đồ phân rã usecase quản lý issue của dự án

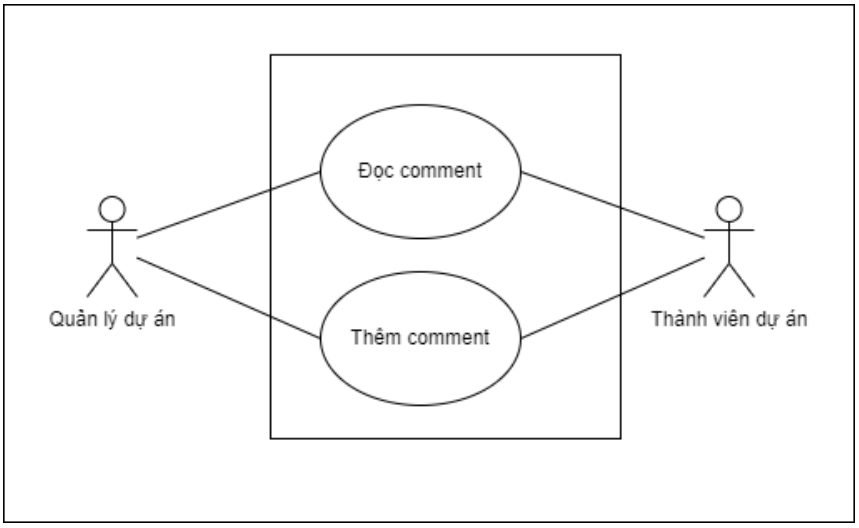
Quản lý dự án có quyền thêm, sửa, xóa, xem chi tiết issue và tạo task. Các thành

viên còn lại có quyền xem các thông tin này.



Hình 2.13: Biểu đồ phân rã usecase quản lý task của dự án

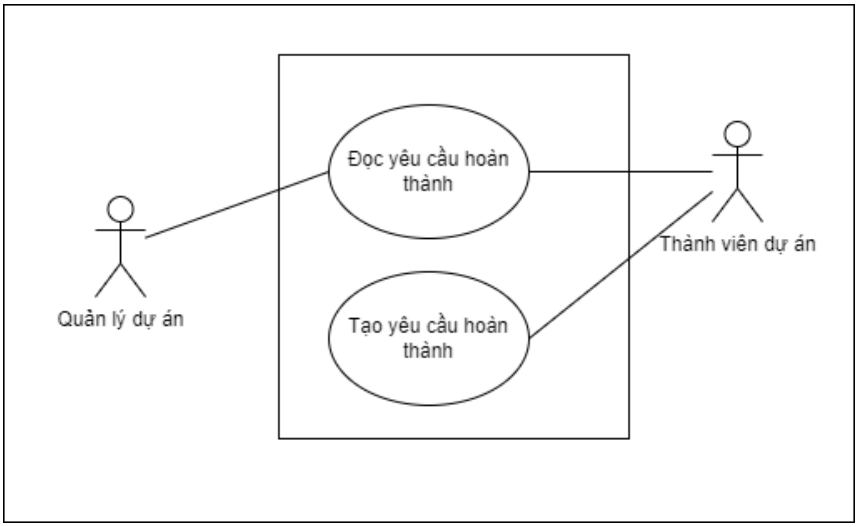
Quản lý dự án có quyền thêm, sửa, xóa, xem chi tiết task và giao task. Các thành viên còn lại có quyền xem các thông tin này.



Hình 2.14: Biểu đồ phân rã usecase quản lý comment của task

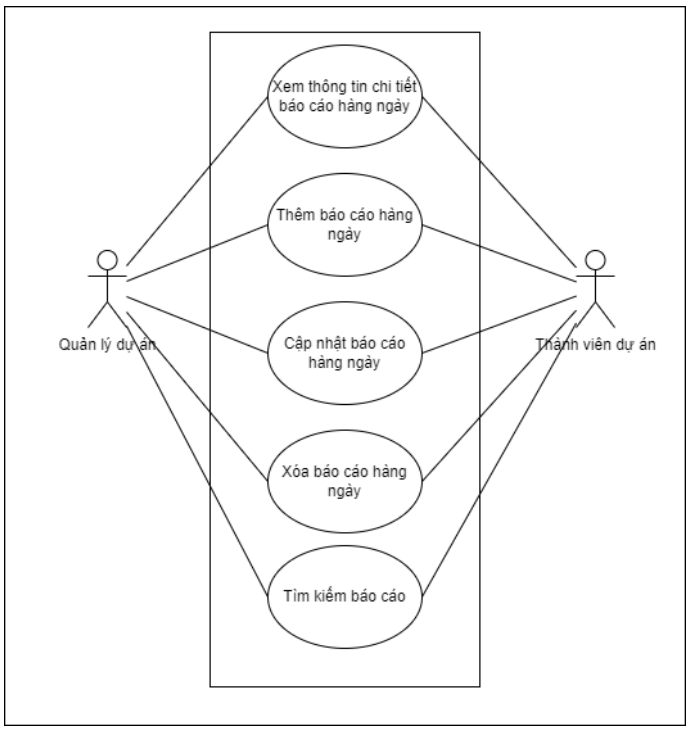
Mọi thành viên trong dự án đều có quyền comment và đọc comment của các

task



Hình 2.15: Biểu đồ phân rã usecase quản lý yêu cầu hoàn thành của task

Các thành viên được giao task đều có quyền tạo yêu cầu duyệt hoàn thành task và xem các yêu cầu duyệt hoàn thành task trước đó. Các thành viên còn lại có quyền xem các lần yêu cầu này.

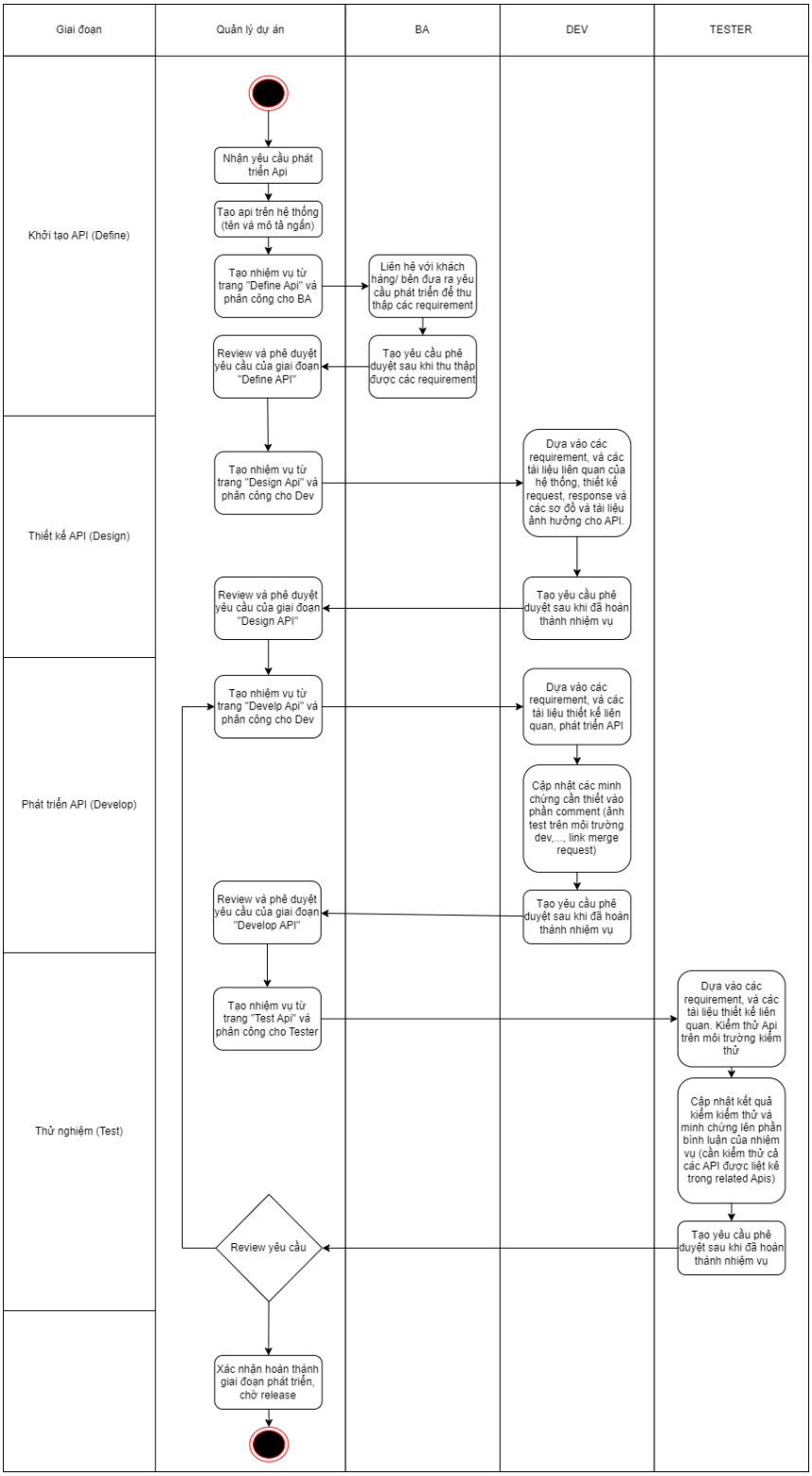


Hình 2.16: Biểu đồ phân rã usecase quản lý daily report của dự án

Mọi thành viên trong dự án đều có đầy đủ chức năng quản lý báo cáo hàng ngày do chính mình tạo.

2.2.3 Quy trình nghiệp vụ

Sau khi hoàn thiện, hệ thống có các quy trình nghiệp vụ sau : Quản lý thông tin chung của dự án, quản lý issue của dự án, quy trình bảo trì API của dự án, quy trình phát triển API của dự án. Bên dưới là quy trình nghiệp vụ phát triển API :



Hình 2.17: Quy trình nghiệp vụ phát triển API

Quy trình nghiệp vụ bao gồm các giai đoạn từ Khởi tạo API (Define), Thiết kế API (Design), Phát triển API (Develop), đến Thử nghiệm API (Test). Sơ đồ này mô tả chi tiết các bước thực hiện trong mỗi giai đoạn, với sự tham gia của các vai trò như Quản lý dự án, BA (Business Analyst), DEV (Developer), và Tester.

a, Giai đoạn Khởi tạo API (Define):

1. **Quản lý dự án:**
   - Nhận yêu cầu phát triển API.
   - Tạo API trên hệ thống (tên và mô tả ngắn).
   - Tạo task từ trang "Define API" và phân công cho BA.
   - Review và phê duyệt yêu cầu của giai đoạn "Define API".
2. **BA:**
   - Liên hệ với khách hàng/bên đưa ra yêu cầu phát triển để thu thập các yêu cầu (requirement).
   - Tạo yêu cầu phê duyệt sau khi thu thập được các requirement.

b, Giai đoạn Thiết kế API (Design):

1. **Quản lý dự án:**
   - Tạo task từ trang "Design API" và phân công cho DEV.
   - Review và phê duyệt yêu cầu của giai đoạn "Design API".
2. **DEV:**
   - Dựa vào các requirement, và các tài liệu liên quan của hệ thống, thiết kế request, response và các sơ đồ và tài liệu ảnh hưởng cho API.
   - Tạo yêu cầu phê duyệt sau khi đã hoàn thành task.

c, Giai đoạn Phát triển API (Develop):

1. **Quản lý dự án:**
   - Tạo task từ trang "Develop API" và phân công cho DEV.
   - Review và phê duyệt yêu cầu của giai đoạn "Develop API".
2. **DEV:**
   - Dựa vào các requirement, và các tài liệu thiết kế liên quan, phát triển API.
   - Cập nhật các minh chứng cần thiết vào phần comment (ảnh test trên môi trường dev, link merge request).
   - Tạo yêu cầu phê duyệt sau khi đã hoàn thành task.

d, Giai đoạn Thử nghiệm API (Test):

1. **Quản lý dự án:** Tạo task từ trang "Test API" và phân công cho Tester.
2. **Tester:**
   - Dựa vào các requirement, và các tài liệu thiết kế liên quan. Kiểm thử API trên môi trường kiểm thử.
   - Cập nhật kết quả kiểm kiểm thử và minh chứng lên phần bình luận của task (cần kiểm thử cả các API được liệt kê trong related APIs).
   - Tạo yêu cầu phê duyệt sau khi đã hoàn thành task.

e, Cuối cùng:

Quản lý dự án:

- Review yêu cầu, nếu xảy ra lỗi khi kiểm thử API, xem xét quay lại các bước trước
- Nếu không có lỗi, xác nhận hoàn thành giai đoạn phát triển, chờ release.

Sơ đồ này giúp minh họa một quy trình rõ ràng và chi tiết từ việc nhận yêu cầu phát triển API đến khi hoàn thành và sẵn sàng để release, đảm bảo tất cả các bước và task được thực hiện và phê duyệt đúng quy trình.

2.3 Đặc tả chức năng

2.3.1 Đặc tả usecase giao task

Bảng 2.1: Đặc tả Usecase giao task.

<table>
<tr><td>Mã usecase</td><td colspan="3">UC002</td></tr>
<tr><td>Tên usecase</td><td colspan="3">Giao task (Giao nhiệm vụ)</td></tr>
<tr><td>Tác nhân</td><td colspan="3">Quản lý dự án</td></tr>
<tr><td>Tiền điều kiện</td><td colspan="3">Đã đăng nhập với vai trò quản lý dự án và đang ở trong giao diện của dự án</td></tr>
<tr><td>Luồng sự kiện chính</td><td colspan="3"><table>
<tr><th>STT</th><th>Tác nhân</th><th>Hành động</th></tr>
<tr><td>1</td><td>Quản lý dự án</td><td>Truy cập vào danh sách task.</td></tr>
<tr><td>2</td><td>Hệ thống</td><td>Hiển thị danh sách task của dự án</td></tr>
<tr><td>3</td><td>Quản lý dự án</td><td>Bấm vào task muốn giao.</td></tr>
<tr><td>4</td><td>Hệ thống</td><td>Hiển thị thông tin chi tiết task.</td></tr>
<tr><td>5</td><td>Quản lý dự án</td><td>Cập nhật trạng thái nhiệm.</td></tr>
<tr><td>6</td><td>Hệ thống</td><td>Cập nhật trạng thái task mà quản lý vừa chọn.</td></tr>
<tr><td>7</td><td>Quản lý dự án</td><td>Bấm vào Assign User và nhập username của thành viên muốn giao.</td></tr>
<tr><td>8</td><td>Quản lý dự án</td><td>Bấm Assign Task.</td></tr>
<tr><td>9</td><td>Hệ thống</td><td>Kiểm tra xem username có trong dự án không.</td></tr>
<tr><td>10</td><td>Hệ thống</td><td>Nếu có, cập nhật thông tin, thực hiện gửi gmail tới cho người được giao task dựa trên email đăng ký tài khoản của user.</td></tr>
</table></td></tr>
<tr><td>Luồng sự kiện phát sinh</td><td colspan="3"><table>
<tr><th>STT</th><th>Tác nhân</th><th>Hành động</th></tr>
<tr><td>9a</td><td>Hệ thống</td><td>Thông báo cho quản lý dự án rằng không tìm thấy username</td></tr>
</table></td></tr>
<tr><td>Hậu điều kiện</td><td colspan="3">Trong phần Assign User có thành viên vừa được giao task</td></tr>
</table>

2.3.2 Đặc tả usecase gửi yêu cầu duyệt hoàn thành task

Bảng 2.2: Đặc tả Usecase gửi yêu cầu duyệt hoàn thành task.

<table>
<tr><td>Mã usecase</td><td colspan="3">UC003</td></tr>
<tr><td>Tên usecase</td><td colspan="3">Yêu cầu duyệt hoàn thành task</td></tr>
<tr><td>Tác nhân</td><td colspan="3">Thành viên dự án</td></tr>
<tr><td>Tiền điều kiện</td><td colspan="3">Đã đăng nhập với vai trò thành viên dự án đang ở trong giao diện của dự án</td></tr>
<tr><td>Luồng sự kiện chính</td><td colspan="3"><table>
<tr><th>STT</th><th>Tác nhân</th><th>Hành động</th></tr>
<tr><td>1</td><td>Thành viên dự án</td><td>Truy cập vào danh sách task.</td></tr>
<tr><td>2</td><td>Hệ thống</td><td>Hiển thị danh sách task của dự án</td></tr>
<tr><td>3</td><td>Thành viên dự án</td><td>Bấm vào task muốn gửi yêu cầu.</td></tr>
<tr><td>4</td><td>Hệ thống</td><td>Hiển thị thông tin chi tiết task.</td></tr>
<tr><td>5</td><td>Thành viên dự án</td><td>Bấm vào nút "request done".</td></tr>
<tr><td>6</td><td>Hệ thống</td><td>Hiển thị lịch sử các lần "request done" của task.</td></tr>
<tr><td>7</td><td>Thành viên dự án</td><td>Nhập các tài liệu và minh chứng cần thiết để chứng minh hoàn thành task vào ô "content".</td></tr>
<tr><td>8</td><td>Thành viên dự án</td><td>Bấm Submit.</td></tr>
<tr><td>9</td><td>Hệ thống</td><td>Gửi mail thông báo có request cho quản lý dự án thông qua email đã đăng ký tài khoản.</td></tr>
</table></td></tr>
<tr><td>Luồng sự kiện phát sinh</td><td colspan="3"><table>
<tr><th>STT</th><th>Tác nhân</th><th>Hành động</th></tr>
<tr><td>8a</td><td>Hệ thống</td><td>Thông báo cho người dùng lỗi nếu chưa nhập thông tin vào ô content</td></tr>
</table></td></tr>
<tr><td>Hậu điều kiện</td><td colspan="3">Hiển thị request done vừa được gửi trong phần lịch sử</td></tr>
</table>

2.3.3 Đặc tả usecase cập nhật tài liệu thiết kế API

Bảng 2.3: Đặc tả Usecase cập nhật tài liệu thiết kế API.

<table>
<tr><td>Mã usecase</td><td colspan="3">UC004</td></tr>
<tr><td>Tên usecase</td><td colspan="3">Cập nhật tài liệu thiết kế API</td></tr>
<tr><td>Tác nhân</td><td colspan="3">Thành viên dự án, quản lý dự án</td></tr>
<tr><td>Tiền điều kiện</td><td colspan="3">Đăng nhập với vai trò thành viên dự án, quản lý dự án, ở trang danh sách folder</td></tr>
<tr><td>Luồng sự kiện chính</td><td colspan="3"><table>
<tr><th>STT</th><th>Tác nhân</th><th>Hành động</th></tr>
<tr><td>1</td><td>Thành viên dự án, quản lý dự án</td><td>Truy cập vào danh sách API.</td></tr>
<tr><td>2</td><td>Hệ thống</td><td>Hiển thị danh sách API của folder</td></tr>
<tr><td>3</td><td>Thành viên dự án, quản lý dự án</td><td>Bấm vào API cần cập nhật.</td></tr>
<tr><td>4</td><td>Hệ thống</td><td>Hiển thị thông tin chi tiết giai đoạn khởi tạo API.</td></tr>
<tr><td>5</td><td>Thành viên dự án, quản lý dự án</td><td>Bấm vào nút "Design".</td></tr>
<tr><td>6</td><td>Hệ thống</td><td>Hiển thị thông tin chi tiết giai đoạn thiết kế API.</td></tr>
<tr><td>7</td><td>Thành viên dự án, quản lý dự án</td><td>Nhập các thông tin cần thiết của API (header, param, ...).</td></tr>
<tr><td>8</td><td>Thành viên dự án</td><td>Bấm vào "Design docs"</td></tr>
<tr><td>9</td><td>Hệ thống</td><td>Hiển thị danh sách các sơ đồ tổng quan, các bảng và API liên quan.</td></tr>
<tr><td>10</td><td>Thành viên dự án</td><td>Cập nhật các sơ đồ tổng quan, các bảng và API liên quan.</td></tr>
<tr><td>11</td><td>Thành viên dự án</td><td>Bấm cập nhật.</td></tr>
<tr><td>12</td><td>Hệ thống</td><td>Thông báo cập nhật thành công.</td></tr>
</table></td></tr>
<tr><td>Luồng sự kiện phát sinh</td><td colspan="3"><table>
<tr><th>STT</th><th>Tác nhân</th><th>Hành động</th></tr>
<tr><td>12a</td><td>Hệ thống</td><td>Thông báo cho người dùng cập nhật thất bại do UID của bảng hoặc API bị sai</td></tr>
</table></td></tr>
<tr><td>Hậu điều kiện</td><td colspan="3">Hiển thị thành công các cập nhật</td></tr>
</table>

2.3.4 Đặc tả usecase cập nhật tài liệu phát triển API

Bảng 2.4: Đặc tả Usecase cập nhật tài liệu phát triển API.

<table>
<tr><td>Mã usecase</td><td colspan="3">UC005</td></tr>
<tr><td>Tên usecase</td><td colspan="3">Cập nhật tài liệu phát triển API</td></tr>
<tr><td>Tác nhân</td><td colspan="3">Thành viên dự án, quản lý dự án</td></tr>
<tr><td>Tiền điều kiện</td><td colspan="3">Đăng nhập với vai trò thành viên dự án, quản lý dự án, ở trang danh sách folder</td></tr>
<tr><td>Luồng sự kiện chính</td><td colspan="3"><table>
<tr><th>STT</th><th>Tác nhân</th><th>Hành động</th></tr>
<tr><td>1</td><td>Thành viên dự án, quản lý dự án</td><td>Truy cập vào danh sách API.</td></tr>
<tr><td>2</td><td>Hệ thống</td><td>Hiển thị danh sách API của folder</td></tr>
<tr><td>3</td><td>Thành viên dự án, quản lý dự án</td><td>Bấm vào API cần cập nhật.</td></tr>
<tr><td>4</td><td>Hệ thống</td><td>Hiển thị thông tin chi tiết giai đoạn khởi tạo API.</td></tr>
<tr><td>5</td><td>Thành viên dự án, quản lý dự án</td><td>Bấm vào nút "Design".</td></tr>
<tr><td>6</td><td>Thành viên dự án, quản lý dự án</td><td>Bấm vào nút "Develop".</td></tr>
<tr><td>7</td><td>Hệ thống</td><td>Hiển thị thông tin chi tiết giai đoạn phát triển API.</td></tr>
<tr><td>8</td><td>Thành viên dự án, quản lý dự án</td><td>Nhập các thông tin cần thiết của API (header, param, ...).</td></tr>
<tr><td>9</td><td>Thành viên dự án, quản lý dự án</td><td>Bấm vào "Save"</td></tr>
<tr><td>10</td><td>Hệ thống</td><td>Thông báo cập nhật thành công.</td></tr>
<tr><td>11</td><td>Thành viên dự án, quản lý dự án</td><td>Bấm vào "Send"</td></tr>
<tr><td>12</td><td>Hệ thống</td><td>Hiển thị Response trả về.</td></tr>
</table></td></tr>
<tr><td>Luồng sự kiện phát sinh</td><td colspan="3"><table>
<tr><th>STT</th><th>Tác nhân</th><th>Hành động</th></tr>
<tr><td>10a</td><td>Hệ thống</td><td>Thông báo cho người dùng cập nhật thất bại</td></tr>
</table></td></tr>
<tr><td>Hậu điều kiện</td><td colspan="3">Hiển thị thành công các cập nhật</td></tr>
</table>

2.4 Yêu cầu phi chức năng

Hiệu năng:

- **Thời gian phản hồi:** Hệ thống phải đảm bảo thời gian phản hồi ngắn cho các thao tác cơ bản như tạo, cập nhật, hoặc xóa tài liệu API.
- **Tối ưu hóa truy vấn:** Các truy vấn cơ sở dữ liệu được tối ưu hóa để đảm bảo hiệu suất tốt nhất

Độ tin cậy:

- **Tính sẵn sàng:** Hệ thống đạt được mức độ sẵn sàng cao, đảm bảo dịch vụ luôn hoạt động liên tục và không bị gián đoạn.

Tính dễ dùng:

- **Giao diện người dùng:** Giao diện thân thiện, dễ sử dụng và trực quan, giúp người dùng dễ dàng thao tác mà không cần quá nhiều hướng dẫn.

Tính bảo mật:

- **Bảo mật:** Hệ thống sử dụng JWT để xác thực, mật khẩu người dùng được mã hóa. Những người dùng bên ngoài dự án hoặc không có quyền hạn sẽ không được xem các thông tin quan trọng của dự án

Các yêu cầu phi chức năng này đảm bảo rằng hệ thống quản lý API không chỉ hoạt động chính xác và hiệu quả mà còn mang lại trải nghiệm tốt nhất cho người dùng, dễ dàng bảo trì và mở rộng trong tương lai.

CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG

3.1 React.js [2]

3.1.1 Giới thiệu

React.js là một thư viện JavaScript mạnh mẽ và phổ biến do Facebook phát triển và duy trì. Nó được sử dụng chủ yếu để xây dựng các giao diện người dùng (UI) tương tác cho các ứng dụng web. Được giới thiệu lần đầu vào năm 2013, React.js đã nhanh chóng trở thành một trong những công nghệ frontend hàng đầu nhờ vào sự linh hoạt, hiệu quả và khả năng mở rộng cao.

3.1.2 Kiến trúc và nguyên tắc hoạt động

Component-Based Architecture (Kiến trúc dựa trên thành phần): React.js được xây dựng theo mô hình kiến trúc dựa trên thành phần. Mỗi thành phần (component) là một đơn vị độc lập có thể tái sử dụng, giúp dễ dàng quản lý và duy trì mã nguồn. Thành phần có thể là các phần nhỏ của giao diện người dùng như button, form, hoặc thậm chí là các thành phần lớn hơn như toàn bộ trang hoặc ứng dụng.

Virtual DOM: React.js sử dụng Virtual DOM để tối ưu hóa hiệu suất. Virtual DOM là một bản sao của DOM thực tế, giúp React.js xác định những thay đổi cần thực hiện mà không phải làm mới toàn bộ trang web. Khi có sự thay đổi trong giao diện, React.js sẽ cập nhật Virtual DOM trước, sau đó so sánh với DOM thực và chỉ áp dụng những thay đổi cần thiết, giảm thiểu thao tác DOM và cải thiện hiệu suất.

One-Way Data Binding: React.js sử dụng cơ chế binding dữ liệu một chiều, tức là dữ liệu chỉ đi theo một chiều từ cha đến con. Điều này giúp dễ dàng theo dõi luồng dữ liệu và quản lý trạng thái của ứng dụng, giảm thiểu lỗi và tăng tính dự đoán.

3.1.3 Các tính năng chính

JSX (JavaScript XML): JSX là một cú pháp mở rộng cho JavaScript, cho phép viết HTML bên trong JavaScript. JSX giúp mã nguồn trở nên rõ ràng và dễ hiểu hơn, đồng thời hỗ trợ lập trình viên xây dựng giao diện người dùng một cách trực quan.

Component Lifecycle: React.js cung cấp các phương thức lifecycle (vòng đời) cho các thành phần, bao gồm các giai đoạn như mounting, updating và unmounting. Các phương thức này cho phép lập trình viên can thiệp vào các thời điểm cụ thể trong vòng đời của một thành phần để thực hiện các hành động cần thiết như cập nhật dữ liệu, làm sạch tài nguyên,...

State và Props: State và Props là hai khái niệm cơ bản trong React.js. State đại diện cho trạng thái của một thành phần, có thể thay đổi trong suốt vòng đời của thành phần đó. Props (properties) là các tham số được truyền từ thành phần cha xuống thành phần con, giúp tạo ra mối liên kết giữa các thành phần.

Hooks: Hooks là một tính năng mới được giới thiệu trong React 16.8, cho phép sử dụng state và các tính năng khác của React trong các functional component. Các hook phổ biến bao gồm useState, useEffect, useContext,...

Context API: Context API cho phép truyền dữ liệu qua các cấp độ khác nhau của component tree mà không cần phải truyền props qua từng cấp. Điều này rất hữu ích cho việc quản lý state toàn cục hoặc chia sẻ dữ liệu giữa các thành phần.

3.1.4 Lợi ích và lý do áp dụng vào dự án

Lợi ích: Hiệu Suất Cao: Với Virtual DOM và cơ chế tối ưu hóa, React.js giúp cải thiện hiệu suất của ứng dụng, giảm thiểu thời gian render và tăng trải nghiệm người dùng.

Tái Sử Dụng Thành Phần: Kiến trúc dựa trên thành phần giúp tái sử dụng mã nguồn, giảm thiểu việc viết lại mã và tăng tính modular của ứng dụng.

Cộng Đồng Hỗ Trợ Mạnh Mẽ: React.js có một cộng đồng phát triển lớn và năng động, cung cấp nhiều tài liệu, thư viện và công cụ hỗ trợ lập trình viên.

Tương Thích và Linh Hoạt: React.js có thể tích hợp với nhiều công nghệ khác nhau và dễ dàng mở rộng để đáp ứng các yêu cầu phức tạp của ứng dụng.

Lý do áp dụng vào dự án: Tính Linh Hoạt và Dễ Sử Dụng: React.js cho phép phát triển các ứng dụng web với giao diện người dùng phong phú và tương tác cao. Khả năng tái sử dụng các thành phần giúp giảm thời gian phát triển và bảo trì.

Hỗ Trợ Tốt Cho Phát Triển Modular: React.js cho phép xây dựng ứng dụng theo từng thành phần nhỏ, giúp dễ dàng quản lý, bảo trì và mở rộng.

Cộng Đồng và Hệ Sinh Thái Mạnh Mẽ: Với một cộng đồng lớn và nhiều thư viện hỗ trợ, việc phát triển và giải quyết các vấn đề kỹ thuật trở nên dễ dàng hơn.

Với những lý do trên, React.js là một lựa chọn lý tưởng cho việc phát triển giao diện người dùng trong dự án quản lý API, đảm bảo tính hiệu quả, dễ bảo trì và mở rộng trong tương lai.

3.2 Spring Boot [3]

3.2.1 Giới Thiệu

Spring Boot là một framework mạnh mẽ và phổ biến dựa trên nền tảng Spring Framework, được thiết kế để đơn giản hóa quá trình phát triển và triển khai các ứng

dụng Java. Được phát triển bởi Pivotal, Spring Boot cung cấp các tính năng tiên tiến giúp lập trình viên xây dựng các ứng dụng microservices, web và enterprise nhanh chóng và hiệu quả. Spring Boot được ưa chuộng nhờ vào khả năng tự động cấu hình, giảm thiểu cấu hình thủ công và cung cấp một tập hợp các công cụ mạnh mẽ để phát triển ứng dụng.

3.2.2 Kiến trúc và nguyên tắc hoạt động

Autoconfiguration (Tự động cấu hình): Spring Boot sử dụng tính năng tự động cấu hình để tự động cấu hình các thành phần và thư viện cần thiết dựa trên các dependencies được khai báo trong dự án. Điều này giúp giảm thiểu công việc cấu hình thủ công và tăng tốc độ phát triển ứng dụng.

Spring Boot Starters: Spring Boot cung cấp các starter packages, là các dependencies được đóng gói sẵn cho các tính năng phổ biến như Spring MVC, Spring Data JPA, Spring Security,... Sử dụng các starter này giúp đơn giản hóa việc thêm các dependencies cần thiết vào dự án.

Embedded Servers (Máy chủ nhúng): Spring Boot đi kèm với các máy chủ nhúng như Tomcat, Jetty, và Undertow, giúp dễ dàng triển khai và chạy ứng dụng mà không cần cài đặt máy chủ ứng dụng riêng biệt. Điều này giúp đơn giản hóa quá trình phát triển và triển khai ứng dụng.

Spring Boot CLI: Spring Boot cung cấp một công cụ dòng lệnh (CLI) giúp lập trình viên nhanh chóng tạo, xây dựng và chạy các ứng dụng Spring Boot từ dòng lệnh mà không cần cấu hình phức tạp.

3.2.3 Các tính năng chính

Spring Initializr: Spring Initializr là một công cụ trực tuyến cho phép lập trình viên dễ dàng tạo một dự án Spring Boot mới với các dependencies cần thiết. Người dùng có thể cấu hình dự án, chọn các dependencies và tải xuống một bộ khung dự án đã được cấu hình sẵn.

Spring Boot Actuator: Spring Boot Actuator cung cấp các endpoint cho phép giám sát và quản lý ứng dụng. Các endpoint này cung cấp thông tin về tình trạng sức khỏe của ứng dụng, metrics, thông tin cấu hình, và nhiều hơn nữa.

Spring Boot DevTools: DevTools là một bộ công cụ giúp cải thiện trải nghiệm phát triển ứng dụng, bao gồm tính năng tự động reload khi có thay đổi trong mã nguồn, giúp lập trình viên thấy ngay các thay đổi mà không cần phải khởi động lại ứng dụng.

Security Integration: Spring Boot tích hợp sẵn với Spring Security, giúp dễ dàng thêm các tính năng bảo mật vào ứng dụng, bao gồm xác thực, phân quyền, và bảo

vệ ứng dụng khỏi các mối đe dọa bảo mật phổ biến.

3.2.4 Lợi Ích và lý do áp dụng vào dự án

Lợi ích: Tăng Tốc Độ Phát Triển: Spring Boot giúp giảm thiểu công việc cấu hình thủ công, cho phép lập trình viên tập trung vào việc phát triển tính năng và logic của ứng dụng. Điều này giúp tăng tốc độ phát triển và ra mắt sản phẩm.

Dễ Dàng Bảo Trì và Mở Rộng: Cấu trúc module của Spring Boot giúp dễ dàng bảo trì và mở rộng ứng dụng. Các thành phần của ứng dụng có thể được cập nhật hoặc thay thế một cách độc lập mà không ảnh hưởng đến toàn bộ hệ thống.

Tính Linh Hoạt và Tích Hợp: Spring Boot cung cấp khả năng tích hợp mạnh mẽ với nhiều công nghệ và thư viện khác, giúp xây dựng các ứng dụng phức tạp và linh hoạt. Khả năng tích hợp dễ dàng với các dịch vụ và công cụ khác cũng giúp tối ưu hóa quy trình phát triển và triển khai.

Cộng Đồng Hỗ Trợ Mạnh Mẽ: Spring Boot có một cộng đồng phát triển lớn và năng động, cung cấp nhiều tài liệu, hướng dẫn và công cụ hỗ trợ lập trình viên giải quyết các vấn đề kỹ thuật.

Lý do áp dụng vào dự án: Tự Động Cấu Hình và Dễ Dàng Sử Dụng: Spring Boot tự động cấu hình các thành phần và thư viện cần thiết, giúp lập trình viên tiết kiệm thời gian và công sức trong việc thiết lập môi trường phát triển.

Tích Hợp Hoàn Hảo với Các Công Nghệ Khác: Spring Boot dễ dàng tích hợp với các công nghệ và công cụ khác như React.js, Docker, và MySQL, tạo nên một hệ sinh thái phát triển hoàn chỉnh và hiệu quả.

Khả Năng Mở Rộng và Bảo Trì Cao: Kiến trúc module của Spring Boot giúp dễ dàng mở rộng và bảo trì ứng dụng, đảm bảo ứng dụng luôn hoạt động ổn định và hiệu quả.

Hiệu Suất Cao và Bảo Mật: Spring Boot cung cấp các công cụ và tính năng giúp cải thiện hiệu suất và bảo mật của ứng dụng, đảm bảo ứng dụng luôn đáp ứng tốt các yêu cầu của người dùng và doanh nghiệp.

Với những lý do trên, Spring Boot là một lựa chọn lý tưởng cho việc phát triển backend trong dự án quản lý API, đảm bảo tính hiệu quả, dễ bảo trì và mở rộng trong tương lai.

3.3 MySQL [4]

3.3.1 Giới Thiệu

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở phổ biến nhất hiện nay, được phát triển bởi Oracle Corporation. Với tính năng mạnh mẽ

và hiệu suất cao, MySQL là một lựa chọn hàng đầu cho nhiều ứng dụng web và doanh nghiệp. Nó hỗ trợ lưu trữ và quản lý dữ liệu một cách hiệu quả, đảm bảo tính toàn vẹn và nhất quán của dữ liệu.

3.3.2 Kiến trúc và nguyên tắc hoạt động

Kiến Trúc Client-Server: MySQL hoạt động theo mô hình client-server, trong đó server chịu trách nhiệm quản lý cơ sở dữ liệu và xử lý các yêu cầu từ client. Các client có thể là các ứng dụng hoặc công cụ quản trị kết nối tới server thông qua giao thức TCP/IP.

Hệ Thống Lưu Trữ: MySQL hỗ trợ nhiều công cụ lưu trữ (storage engines) khác nhau, như InnoDB, MyISAM, Memory, v.v. Trong đó, InnoDB là storage engine mặc định, cung cấp các tính năng như ACID (Atomicity, Consistency, Isolation, Durability) và hỗ trợ khóa ngoại (foreign key) để đảm bảo tính toàn vẹn dữ liệu.

Query Optimization (Tối Ưu Hóa Truy Vấn): MySQL sử dụng một loạt các kỹ thuật tối ưu hóa truy vấn để cải thiện hiệu suất, bao gồm tối ưu hóa SQL, sử dụng index, cache kết quả truy vấn, và phân tích thống kê dữ liệu. Điều này giúp giảm thiểu thời gian xử lý và tối ưu hóa tài nguyên hệ thống.

Replication (Sao Chép Dữ Liệu): MySQL hỗ trợ nhiều phương pháp sao chép dữ liệu (replication), giúp cải thiện khả năng chịu lỗi và khả năng mở rộng của hệ thống. Có hai loại replication chính: Master-Slave và Master-Master, giúp phân phối tải và đảm bảo tính sẵn sàng cao.

3.3.3 Các tính năng chính

ACID Compliance: MySQL hỗ trợ các giao dịch ACID, đảm bảo rằng tất cả các thao tác trên cơ sở dữ liệu đều nhất quán và đáng tin cậy. Điều này rất quan trọng trong các ứng dụng yêu cầu tính toàn vẹn dữ liệu cao.

Multi-User Access: MySQL cho phép nhiều người dùng truy cập và thao tác với cơ sở dữ liệu cùng một lúc mà không làm gián đoạn hoặc làm giảm hiệu suất của hệ thống. Nó sử dụng cơ chế khóa để quản lý truy cập đồng thời một cách hiệu quả.

Scalability (Khả Năng Mở Rộng): MySQL có thể mở rộng dễ dàng bằng cách thêm nhiều server hoặc sử dụng các kỹ thuật phân tán dữ liệu. Nó hỗ trợ cả scaling ngang (horizontal scaling) và scaling dọc (vertical scaling).

Backup and Recovery (Sao Lưu và Phục Hồi): MySQL cung cấp các công cụ mạnh mẽ để sao lưu và phục hồi dữ liệu, bao gồm mysqldump, MySQL Enterprise Backup, và tính năng snapshot. Điều này giúp bảo vệ dữ liệu và đảm bảo tính liên tục của hệ thống.

Security (Bảo Mật): MySQL cung cấp nhiều tính năng bảo mật như xác thực người dùng, mã hóa dữ liệu, và kiểm soát truy cập chi tiết. Nó hỗ trợ các phương thức xác thực mạnh mẽ và khả năng mã hóa dữ liệu tại rest và in-transit.

3.3.4 Lợi ích và lý do áp dụng vào dự án

Lợi ích: Hiệu Suất Cao: MySQL được tối ưu hóa cho các ứng dụng web với hiệu suất cao và khả năng xử lý lượng lớn dữ liệu một cách nhanh chóng. Nó sử dụng các kỹ thuật tối ưu hóa truy vấn và quản lý tài nguyên hiệu quả.

Dễ Sử Dụng: MySQL cung cấp giao diện người dùng thân thiện và dễ sử dụng, bao gồm các công cụ như MySQL Workbench. Điều này giúp người dùng dễ dàng quản lý và thao tác với cơ sở dữ liệu.

Cộng Đồng và Hỗ Trợ: MySQL có một cộng đồng lớn và năng động, cung cấp nhiều tài liệu, hướng dẫn và công cụ hỗ trợ. Ngoài ra, còn có các dịch vụ hỗ trợ chuyên nghiệp từ Oracle và các nhà cung cấp khác.

Tính Linh Hoạt: MySQL hỗ trợ nhiều loại ứng dụng khác nhau, từ các trang web nhỏ đến các hệ thống doanh nghiệp lớn. Nó có thể chạy trên nhiều nền tảng và hệ điều hành, bao gồm Windows, Linux và macOS.

Lý do áp dụng vào dự án: Khả Năng Tích Hợp Tốt: MySQL dễ dàng tích hợp với nhiều công nghệ và công cụ khác như Java Spring Boot, React.js, Docker. Điều này tạo điều kiện thuận lợi cho việc phát triển một hệ thống hoàn chỉnh và hiệu quả.

Hiệu Suất và Độ Tin Cậy: MySQL cung cấp hiệu suất cao và độ tin cậy trong việc quản lý và truy xuất dữ liệu, đảm bảo hệ thống hoạt động mượt mà và ổn định.

Bảo Mật và Quản Lý Dữ Liệu: Với các tính năng bảo mật mạnh mẽ và khả năng quản lý dữ liệu hiệu quả, MySQL đảm bảo an toàn và bảo mật cho dữ liệu quan trọng của ứng dụng.

Hỗ Trợ Tính Mở Rộng: MySQL cung cấp các giải pháp mở rộng linh hoạt, giúp hệ thống dễ dàng thích ứng với nhu cầu phát triển và mở rộng của dự án.

Với những lý do trên, MySQL là một lựa chọn lý tưởng cho việc quản lý cơ sở dữ liệu trong dự án quản lý API, đảm bảo tính hiệu quả, độ tin cậy và khả năng mở rộng trong tương lai.

3.4 Docker [5]

3.4.1 Giới Thiệu

Docker là một nền tảng mã nguồn mở giúp tự động hóa việc triển khai các ứng dụng dưới dạng các container nhẹ, di động và tự cung cấp. Được phát triển bởi

Docker, Inc., Docker cho phép các nhà phát triển đóng gói ứng dụng cùng với tất cả các phần mềm phụ thuộc của nó vào một container. Điều này đảm bảo rằng ứng dụng sẽ chạy mượt mà trên bất kỳ môi trường nào, từ máy tính cá nhân của nhà phát triển đến các máy chủ sản xuất.

3.4.2 Kiến trúc và nguyên tắc hoạt động

Containerization (Đóng Gói Container): Docker sử dụng công nghệ container để đóng gói ứng dụng và các phụ thuộc của nó vào một đơn vị nhỏ gọn. Container là một môi trường cách ly, nơi ứng dụng có thể chạy mà không ảnh hưởng đến hệ thống chủ hoặc các ứng dụng khác.

Docker Engine: Docker Engine là thành phần chính của Docker, cung cấp khả năng tạo, chạy và quản lý các container. Nó bao gồm hai phần chính: Docker Daemon (chạy trên máy chủ và quản lý các đối tượng Docker) và Docker CLI (giao diện dòng lệnh để tương tác với Docker Daemon).

Docker Images: Docker Image là một mẫu chỉ đọc để tạo container. Mỗi image bao gồm hệ điều hành, phần mềm, thư viện, và mã nguồn ứng dụng. Docker sử dụng Dockerfile để định nghĩa cách tạo ra một image. Dockerfile là một tập tin chứa các lệnh và hướng dẫn để xây dựng image từ nền tảng cơ bản đến các phần mềm phụ thuộc và mã nguồn ứng dụng.

Docker Hub: Docker Hub là một kho lưu trữ trực tuyến, nơi người dùng có thể tải lên và chia sẻ các Docker images. Người dùng có thể tìm kiếm và sử dụng các image đã có sẵn hoặc tải lên các image của riêng họ.

3.4.3 Các tính năng chính

Isolation (Cách Ly): Docker container cung cấp môi trường cách ly, đảm bảo rằng các ứng dụng chạy độc lập và không xung đột với nhau. Điều này giúp giảm thiểu rủi ro và cải thiện độ tin cậy của hệ thống.

Portability (Di Động): Với Docker, các container có thể chạy trên bất kỳ môi trường nào có Docker Engine, từ máy tính cá nhân đến các máy chủ sản xuất và các nền tảng đám mây. Điều này giúp đảm bảo tính nhất quán và dễ dàng di chuyển ứng dụng giữa các môi trường.

Scalability (Khả Năng Mở Rộng): Docker hỗ trợ việc triển khai và quản lý các ứng dụng ở quy mô lớn. Các container có thể được dễ dàng sao chép và phân phối để đáp ứng nhu cầu tải cao, giúp cải thiện khả năng mở rộng của hệ thống.

Automation (Tự Động Hóa): Docker cho phép tự động hóa quá trình xây dựng, kiểm thử và triển khai ứng dụng thông qua các công cụ như Docker Compose và Docker Swarm. Điều này giúp giảm thiểu công việc thủ công và cải thiện hiệu suất

làm việc.

3.4.4 Lợi Ích và lý do áp dụng vào dự án

Lợi ích: Tính Nhất Quán: Docker đảm bảo rằng ứng dụng sẽ chạy nhất quán trên mọi môi trường, giảm thiểu các vấn đề liên quan đến sự khác biệt môi trường phát triển và sản xuất.

Hiệu Suất Cao: Docker container nhẹ và khởi động nhanh chóng, giúp cải thiện hiệu suất và thời gian phản hồi của ứng dụng.

Dễ Dàng Quản Lý và Triển Khai: Docker cung cấp các công cụ mạnh mẽ để quản lý và triển khai các container, giúp đơn giản hóa quy trình phát triển và triển khai ứng dụng.

Giảm Chi Phí: Docker giúp tối ưu hóa việc sử dụng tài nguyên hệ thống, giảm thiểu chi phí phần cứng và hạ tầng.

Lý do áp dụng vào dự án: Tích Hợp MySQL: Docker cho phép dễ dàng triển khai và quản lý MySQL database dưới dạng container.. Điều này giúp đơn giản hóa quá trình thiết lập và cấu hình cơ sở dữ liệu, đảm bảo tính nhất quán và khả năng di động cao.

Đóng Gói Ứng Dụng: Docker giúp đóng gói toàn bộ ứng dụng quản lý API, bao gồm backend (Java Spring Boot), frontend (React.js), và các phụ thuộc khác vào các container. Điều này đảm bảo rằng ứng dụng có thể chạy mượt mà trên mọi môi trường mà không cần cấu hình lại.

Quy Trình CI/CD: Docker tích hợp tốt với các công cụ CI/CD như GitHub Actions, giúp tự động hóa quá trình kiểm thử và triển khai ứng dụng.

Tính Mở Rộng và Khả Năng Chịu Lỗi: Docker hỗ trợ việc triển khai và quản lý các ứng dụng ở quy mô lớn, giúp cải thiện khả năng mở rộng và khả năng chịu lỗi của hệ thống. Các container có thể được dễ dàng sao chép và phân phối để đáp ứng nhu cầu tải cao.

Với những lý do trên, Docker là một công cụ lý tưởng cho việc quản lý và triển khai MySQL database và đóng gói dự án quản lý API, đảm bảo tính hiệu quả, nhất quán và khả năng mở rộng trong tương lai.

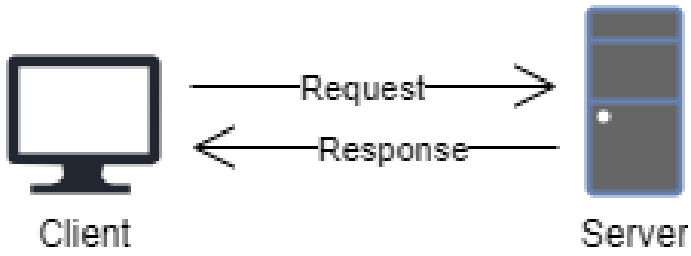
CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

4.1 Thiết kế kiến trúc

4.1.1 Lựa chọn kiến trúc phần mềm

Dự án quản lý API Document được thiết kế dựa trên kiến trúc Client-Server. Kiến trúc này giúp tách biệt rõ ràng giữa các thành phần của ứng dụng, tăng cường tính mô-đun và dễ dàng bảo trì, mở rộng. Kiến trúc Client-Server bao gồm hai thành phần chính: client (máy khách) và server (máy chủ).

Client-Server:



Hình 4.1: Kiến trúc Client-Server

Kiến trúc Client-Server là một mô hình phổ biến trong việc phát triển các ứng dụng web. Trong mô hình này, client gửi yêu cầu (request) đến server, server xử lý yêu cầu đó và gửi phản hồi (response) trở lại client. Điều này giúp tách biệt logic xử lý và giao diện người dùng, cho phép chúng hoạt động độc lập và tăng tính linh hoạt cho ứng dụng.

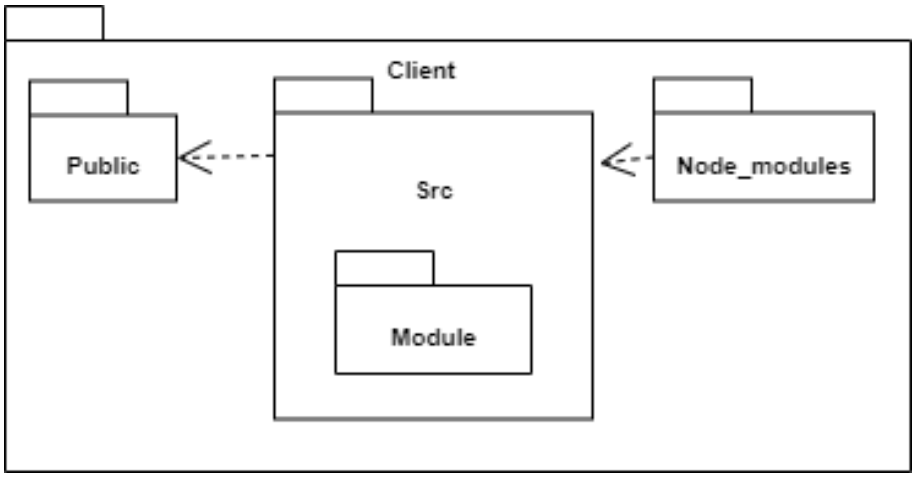
Giao Tiếp Giữa Client và Server: Giao tiếp giữa client và server được thực hiện thông qua các yêu cầu HTTP. Client gửi các yêu cầu HTTP (GET, POST, PUT, DELETE) đến server qua các endpoint được định nghĩa trong các lớp controller. Server nhận các yêu cầu này, xử lý logic nghiệp vụ thông qua các lớp service, tương tác với cơ sở dữ liệu thông qua các lớp repository, và cuối cùng trả về phản hồi cho client.

Lý Do Lựa Chọn Kiến Trúc Client-Server:

Tách biệt các thành phần: Kiến trúc Client-Server giúp tách biệt rõ ràng giữa logic xử lý và giao diện người dùng, giúp mã nguồn dễ bảo trì và mở rộng. Khả năng mở rộng: Cho phép mở rộng từng thành phần một cách độc lập, tăng cường khả năng xử lý và hiệu suất của hệ thống. Tính linh hoạt: Client và server có thể được phát triển và triển khai trên các nền tảng khác nhau, miễn là chúng tuân thủ giao thức HTTP để giao tiếp. Với kiến trúc Client-Server, dự án quản lý API có thể dễ dàng mở rộng, bảo trì và cung cấp một nền tảng vững chắc cho việc phát triển

các tính năng mới trong tương lai.

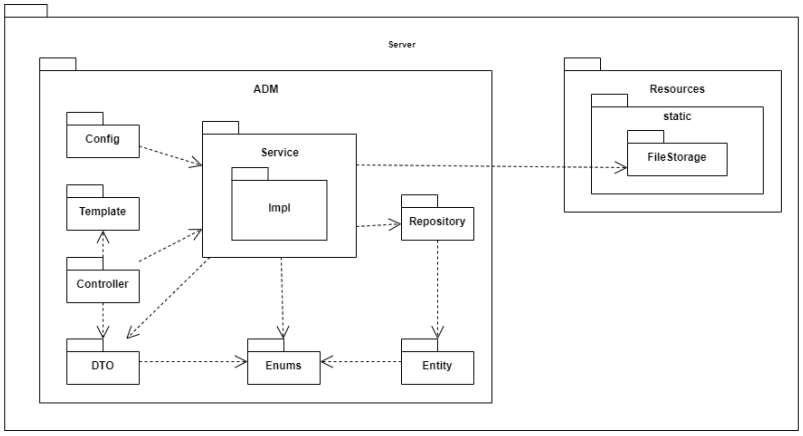
4.1.2 Thiết kế tổng quan



Hình 4.2: Biểu Đồ Gói Tổng Quan Cho Máy Khách

a. Biểu Đồ Gói Tổng Quan Cho Máy Khách

1. **Gói Public:** Chứa các tài nguyên tĩnh như hình ảnh, CSS, và các file tĩnh khác. Các tài nguyên trong gói này được phục vụ trực tiếp đến trình duyệt người dùng mà không cần qua quá trình xử lý của ứng dụng, đảm bảo rằng các tài nguyên tĩnh được tải nhanh chóng và hiệu quả, cải thiện trải nghiệm người dùng.
2. **Gói Src:**
   - **Module**: bao gồm các module của hệ thống, trong mỗi folder trong module sẽ có các file mã của từng trang giao diện tương ứng với module. Ngoài ra, trong module chứa các file component để tái sử dụng ở các trang, chứa các file HOC(Higher-Order Components), được sử dụng để bảo vệ các route trong ứng dụng, kiểm tra xem người dùng có quyền truy cập vào một route cụ thể hay không, dựa trên xác thực và quyền hạn của người dùng. Module cũng chứa cấu hình AxiosInstanse, Axios được sử dụng để gửi các request HTTP tới server. Axios được cấu hình để tự động thêm token xác thực vào header của mỗi request, giúp tăng tính bảo mật và linh hoạt cho các request.
3. **Gói Node_modules:** Chứa tất cả các module và thư viện phụ thuộc của ứng dụng. Các module này được quản lý bởi npm (Node Package Manager) và được sử dụng để cung cấp các chức năng cần thiết cho ứng dụng, đảm bảo rằng tất cả các phụ thuộc được cài đặt và quản lý một cách nhất quán, giúp phát triển và bảo trì ứng dụng dễ dàng hơn.



Hình 4.3: Biểu Đồ Gói Tổng Quan Cho Máy Chủ

b. Biểu Đồ Gói Tổng Quan Cho Máy Chủ

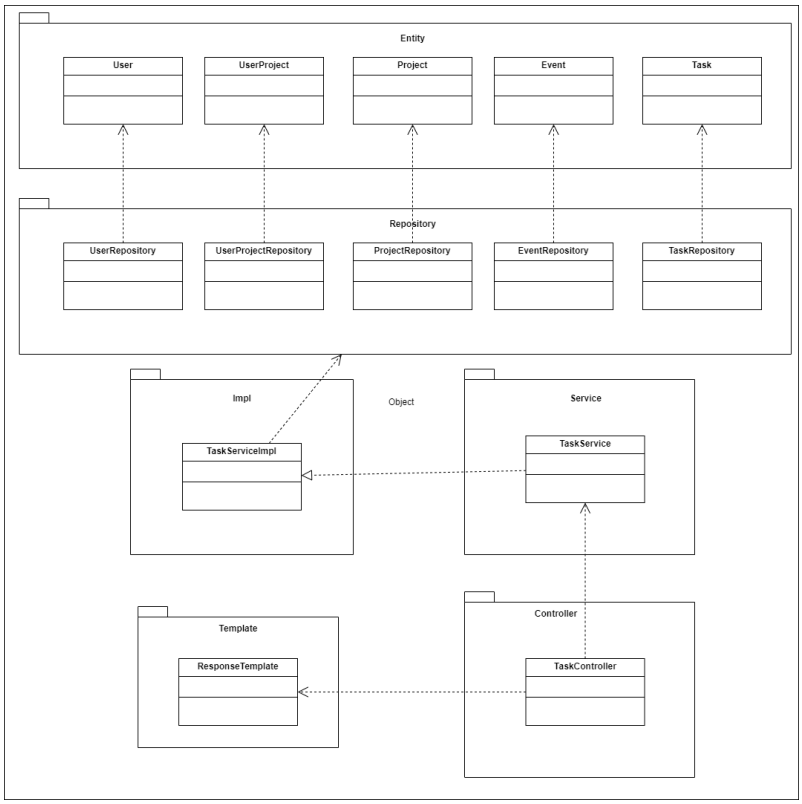
1. **Gói Config:** Chứa các cấu hình của ứng dụng như CORS, JWT, và bất đồng bộ.
2. **Gói Controller:** Chứa các lớp controller xử lý các yêu cầu HTTP từ client. Các controller nhận các yêu cầu từ phía client, gọi các service tương ứng để xử lý nghiệp vụ và trả về kết quả cho client. Các response template được sử dụng để chuẩn hóa các phản hồi từ controller.
3. **Gói DTO (Data Transfer Object):** Chứa các đối tượng truyền dữ liệu giữa các lớp khác nhau. DTO giúp tách biệt dữ liệu giữa các tầng của ứng dụng, giúp tăng tính bảo mật và dễ dàng bảo trì mã nguồn.
4. **Gói Entity:** Chứa các lớp mô hình dữ liệu, tương ứng với các bảng trong cơ sở dữ liệu. Các lớp entity đại diện cho cấu trúc dữ liệu và các ràng buộc của chúng trong cơ sở dữ liệu.
5. **Gói Repository:** Chứa các lớp để tương tác với cơ sở dữ liệu, thực hiện các thao tác CRUD (Create, Read, Update, Delete). Các repository sử dụng Spring Data JPA để truy vấn và quản lý dữ liệu trong cơ sở dữ liệu.
6. **Gói Service:** Chứa các lớp logic nghiệp vụ của ứng dụng, được chia thành interface và các lớp triển khai trong gói `impl`. Các service xử lý các nghiệp vụ phức tạp, tương tác với repository và các thành phần khác để thực hiện các chức năng của ứng dụng.
7. **Gói Impl:** Chứa các lớp triển khai cho các interface trong gói service. Các lớp này thực hiện các logic nghiệp vụ cụ thể và tương tác với các repository để truy xuất và xử lý dữ liệu.
8. **Gói Enums:** Chứa các enum được sử dụng trong toàn bộ ứng dụng. Các enum

đại diện cho các hằng số hoặc các giá trị cố định, giúp mã nguồn dễ đọc và bảo trì hơn.

9. **Gói Template:** Chứa các mẫu response template của các controller.Các template này giúp chuẩn hóa và tái sử dụng các phản hồi từ controller, đảm bảo tính nhất quán trong toàn bộ ứng dụng.
10. **Gói Resources:** Chứa các tài nguyên tĩnh và các file cấu hình của ứng dụng. Gói này bao gồm thư mục static, nơi lưu trữ các file được tải lên bởi service "quản lý file chung của dự án". FileStorage: Chứa các file được tải lên, giúp quản lý và lưu trữ các tài nguyên tĩnh và file của dự án.

4.1.3 Thiết kế chi tiết gói

Chi tiết gói cho module quản lý task



Hình 4.4: Thiết kế gói cho module quản lý task

Mô Tả Chi Tiết Biểu Đồ:

1. **Gói Entity:**
   - **User:** Đại diện cho người dùng trong hệ thống.
   - **UserProject:** Thể hiện mối quan hệ giữa người dùng và dự án, bao gồm vai trò của người dùng trong từng dự án cụ thể.
   - **Project:** Đại diện cho dự án trong hệ thống.

- **Event:** Đại diện cho các sự kiện trong hệ thống, bao gồm các sự kiện liên quan đến task và các hoạt động khác.
- **Task:** Đại diện cho các task trong hệ thống.

2. **Gói Repository:**
   - **UserRepository:** Quản lý các thao tác CRUD cho đối tượng User.
   - **UserProjectRepository:** Quản lý các thao tác CRUD cho đối tượng User-Project.
   - **ProjectRepository:** Quản lý các thao tác CRUD cho đối tượng Project.
   - **EventRepository:** Quản lý các thao tác CRUD cho đối tượng Event.
   - **TaskRepository:** Quản lý các thao tác CRUD cho đối tượng Task.
3. **Gói Service:**
   - **TaskService:** Interface định nghĩa các phương thức nghiệp vụ liên quan đến quản lý task.
   - **TaskServiceImpl:** Lớp triển khai của TaskService, chứa các logic nghiệp vụ chi tiết để quản lý task, bao gồm việc tương tác với repository và xử lý các yêu cầu từ controller.
4. **Gói Controller: TaskController:** Xử lý các yêu cầu HTTP liên quan đến task từ client, gọi các phương thức của TaskService để thực hiện các thao tác và trả về kết quả cho client.
5. **Gói Template: ResponseTemplate:** Định nghĩa các mẫu phản hồi chuẩn hóa từ controller, đảm bảo tính nhất quán trong việc phản hồi các yêu cầu từ client.

4.2 Thiết kế chi tiết

4.2.1 Thiết kế giao diện

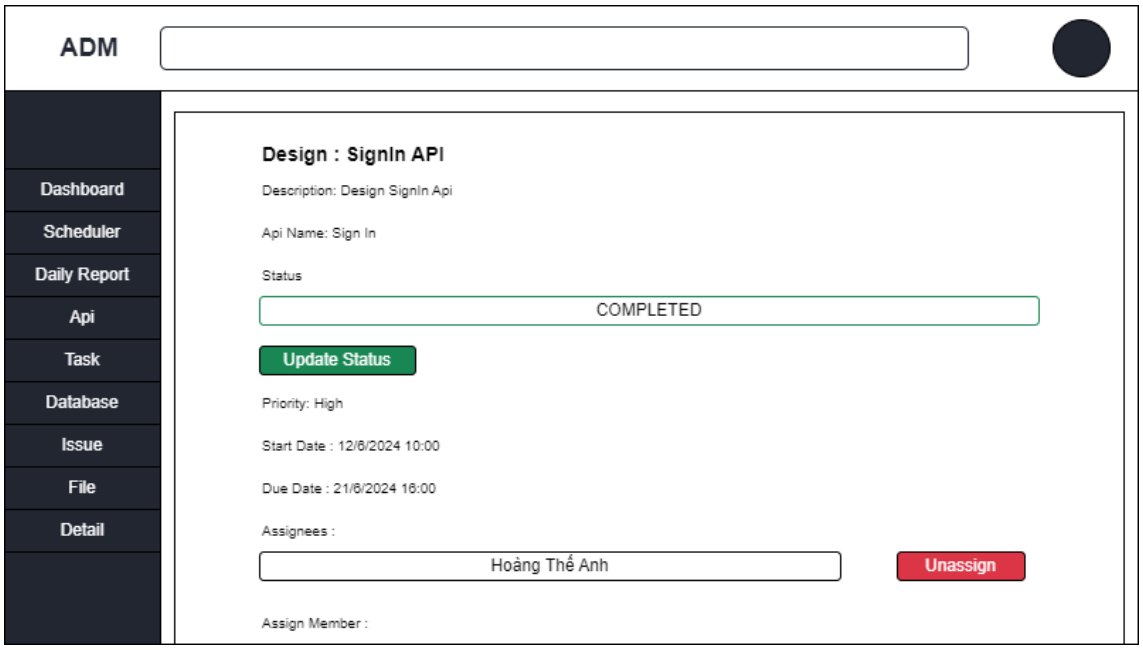
a. Yêu Cầu Về Mặt Thiết Kế Khi tiến hành xây dựng giao diện, cần phải đảm bảo các yếu tố sau để tạo ra một ứng dụng thân thiện với người dùng và hiệu quả:

1. **Tương Tác Người Dùng:**
   - Giao diện phải đảm bảo khả năng tương tác tốt với người dùng. Mỗi hành động của người dùng nên có phản hồi để người dùng biết được kết quả của hành động của mình. Ví dụ, khi người dùng nhấn vào một nút, nút đó nên thay đổi màu sắc hoặc hình dạng để chỉ ra rằng thao tác đã được ghi nhận.
   - Các tương tác này cũng cần được gợi ý bởi các hình thức khác như high-

light, thông báo, giúp người dùng dễ dàng nhận biết được các tính năng và hành động có sẵn.

2. **Phản Hồi Người Dùng:**
   - Hệ thống cần có cơ chế phản hồi lại các tương tác của người dùng để người dùng biết được kết quả thực hiện hành động của mình.
   - Thông điệp phản hồi nên rõ ràng và dễ hiểu, sử dụng màu sắc và biểu tượng để nhấn mạnh loại thông báo (thành công, cảnh báo, lỗi).
3. **Linh Hoạt và Thân Thiện Với Người Dùng:**
   - Giao diện người dùng của ứng dụng phải hướng đến sự linh hoạt và thân thiện với người dùng. Nó cần tương thích trên nhiều thiết bị như máy tính, máy tính bảng, điện thoại, đảm bảo trải nghiệm nhất quán trên tất cả các nền tảng.
   - Đối với các loại dữ liệu có độ rộng và độ dài vượt quá kích thước hiển thị của màn hình, cần phải có cách hiển thị khác, chẳng hạn như cuộn ngang, thu nhỏ hoặc hiển thị tóm tắt với khả năng mở rộng chi tiết khi cần.
4. **Điều Hướng và Trải Nghiệm Người Dùng:**
   - Việc di chuyển giữa các trang và điều hướng trong ứng dụng là vấn đề cốt lõi khi thiết kế trải nghiệm người dùng (UX). Giao diện cần đảm bảo rằng việc điều hướng là dễ dàng và trực quan, không gây nhầm lẫn cho người dùng.
   - Giao diện cần đảm bảo trải nghiệm người dùng liền mạch từ đầu đến cuối tác vụ, nhất quán từ đầu đến cuối. Các thành phần giao diện nên được bố trí hợp lý để người dùng có thể dễ dàng tìm thấy và sử dụng các tính năng của ứng dụng.
5. **Màu Sắc và Thẩm Mỹ:**
   - Màu sắc sử dụng trên giao diện phải phù hợp với từng loại chức năng. Màu sắc cần tuân theo một tông màu chủ đạo, không dùng các màu sắc quá sặc sỡ hoặc khó chịu cho mắt.
   - Các màu sắc cần phải được lựa chọn sao cho dễ quan sát, không bị nhập nhằng, gây khó chịu cho người dùng và phù hợp với chức năng muốn truyền tải. Ví dụ, màu xanh lá cây có thể được sử dụng cho các hành động thành công, màu đỏ cho các cảnh báo hoặc lỗi.
6. **Độ Phân Giải và Kích Thước Màn Hình:**

- Ứng dụng cần tương thích với các thiết bị có độ phân giải màn hình từ 1280x720 (HD) trở lên, với độ phân giải tối ưu là 1920x1080 (Full HD) để đảm bảo các thành phần giao diện được hiển thị rõ ràng và sắc nét.
- Thiết kế giao diện phải đảm bảo hiển thị đầy đủ thông tin trên các màn hình từ 13 inch trở lên, phổ biến cho các laptop và màn hình máy tính để bàn. Trên các thiết bị có màn hình nhỏ hơn, giao diện sẽ tự động điều chỉnh để sử dụng tối ưu không gian hiển thị mà không gây rối mắt người dùng.



Hình 4.5: Thiết kế giao diện

b. Thiết Kế Giao Diện Giao diện chính của website bao gồm các phần chính: App Bar, Side Bar và Content.

- **App Bar:**
  - App Bar được đặt phía trên cùng của website, có chức năng cung cấp chức năng điều hướng tới trang chủ (home), trang cá nhân (profile) và chức năng đăng xuất (sign out).
  - Các biểu tượng và nút điều hướng trên App Bar được sắp xếp gọn gàng và rõ ràng, giúp người dùng dễ dàng thao tác.
- **Side Bar:**
  - Side Bar nằm bên trái màn hình, có chức năng điều hướng tới các chức năng chính của website.

  - Các mục điều hướng trong Side Bar bao gồm Dashboard, Scheduler, Daily Report, API, Task, Database, Issue, File, và Detail.
  - Mỗi mục điều hướng được biểu diễn bằng biểu tượng và tên chức năng, giúp người dùng nhanh chóng nhận biết và truy cập vào các chức năng cần thiết.

- **Content:**
  - Phần Content chứa các nội dung chính của trang, được hiển thị ở giữa màn hình, chiếm phần lớn diện tích hiển thị.
  - Nội dung trong phần Content thay đổi linh hoạt dựa trên các lựa chọn điều hướng từ Side Bar, cung cấp thông tin và công cụ cần thiết cho người dùng.
  - Ví dụ, khi người dùng chọn chức năng "Task" từ Side Bar, phần Content sẽ hiển thị thông tin chi tiết về các task, bao gồm trạng thái, mô tả, ngày bắt đầu, ngày kết thúc, và người được giao task.

Dưới đây là mô tả cho giao diện chi tiết của chức năng "Chi tiết Task":

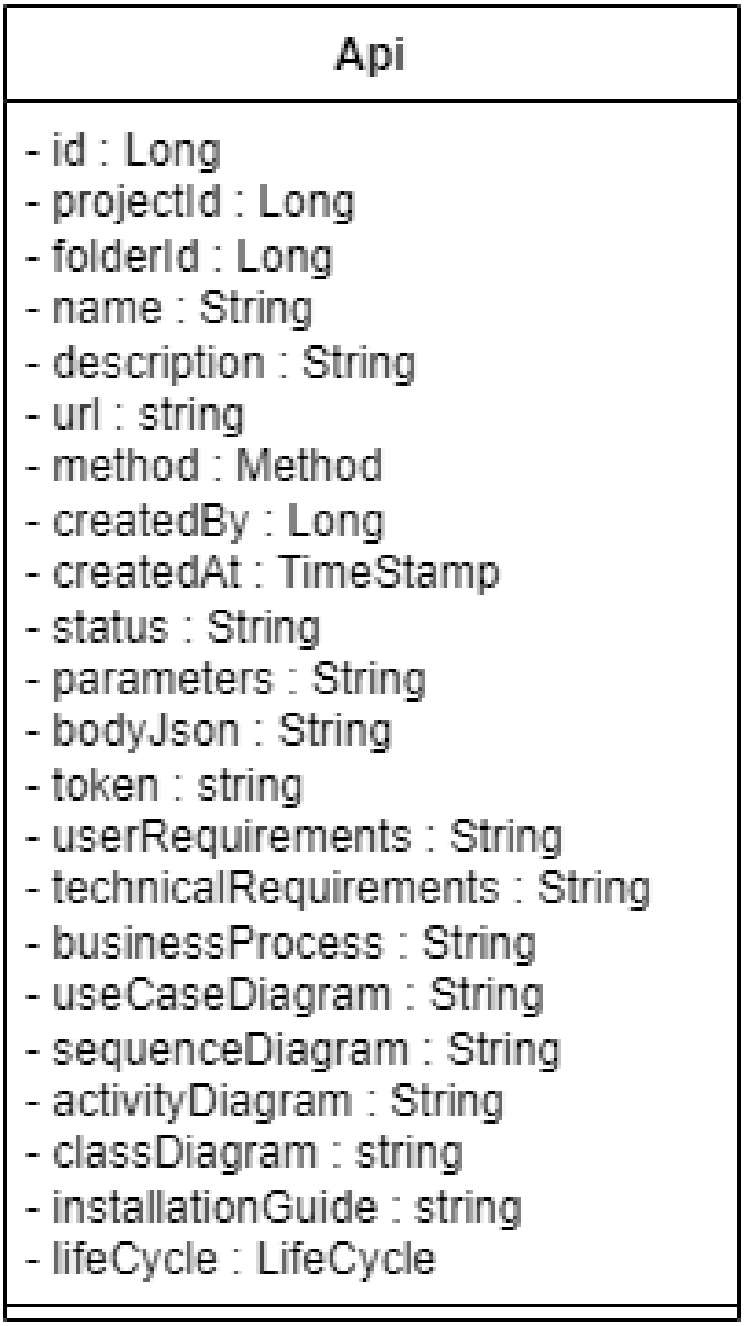
Chi Tiết Giao Diện Chi Tiết Task:

- **Tiêu Đề:** Hiển thị tên của API và mô tả ngắn gọn về chức năng của API đó.
- **Trạng Thái:** Hiển thị trạng thái hiện tại của API, ví dụ như "COMPLETED". Người dùng có thể cập nhật trạng thái bằng cách nhấn vào nút "Update Status".
- **Độ Ưu Tiên:** Hiển thị độ ưu tiên của task, ví dụ như "High".
- **Ngày Bắt Đầu và Ngày Kết Thúc:** Hiển thị thông tin về ngày bắt đầu và ngày kết thúc của task.
- **Người Được Giao task:** Hiển thị danh sách những người được giao task cho task này. Người dùng có thể thêm hoặc xóa người được giao task bằng cách sử dụng các nút tương ứng.

Thiết kế giao diện này hướng đến sự linh hoạt, thân thiện với người dùng, và đảm bảo rằng mọi thông tin và chức năng cần thiết đều được hiển thị một cách rõ ràng và dễ dàng truy cập. Việc phân chia rõ ràng giữa App Bar, Side Bar và Content giúp người dùng dễ dàng điều hướng và sử dụng các chức năng của website một cách hiệu quả.

4.2.2 Thiết kế lớp

Thiết kế chi tiếp lớp Api:



Hình 4.6: Lớp Api

id: Kiểu dữ liệu: long. Ý nghĩa: Định danh duy nhất cho API.

projectId: Kiểu dữ liệu: long. Ý nghĩa: Định danh của dự án mà API thuộc về.

folderId: Kiểu dữ liệu: long. Ý nghĩa: Định danh của thư mục chứa API.

name: Kiểu dữ liệu: string. Ý nghĩa: Tên của API.

description: Kiểu dữ liệu: string. Ý nghĩa: Mô tả ngắn gọn về API.

url: Kiểu dữ liệu: string. Ý nghĩa: URL của API.

method: Kiểu dữ liệu: Method. Ý nghĩa: Phương thức HTTP của API (GET, POST, PUT, DELETE, v.v.).

createdBy: Kiểu dữ liệu: long. Ý nghĩa: Định danh của người tạo API.

status: Kiểu dữ liệu: string. Ý nghĩa: Trạng thái hiện tại của API.

parameters: Kiểu dữ liệu: string. Ý nghĩa: Các tham số của API, dùng khi tạo request API.

bodyJson: Kiểu dữ liệu: string. Ý nghĩa: Nội dung body của API dưới dạng chuỗi JSON, dùng khi tạo request API.

token: Kiểu dữ liệu: string. Ý nghĩa: Token xác thực cho API, dùng khi tạo request API.

userRequirements: Kiểu dữ liệu: string. Ý nghĩa: Các yêu cầu của người dùng đối với API.

technicalRequirements: Kiểu dữ liệu: string. Ý nghĩa: Các yêu cầu kỹ thuật của API.

businessProcess: Kiểu dữ liệu: string. Ý nghĩa: Mô tả quy trình nghiệp vụ liên quan đến API.

useCaseDiagram: Kiểu dữ liệu: string. Ý nghĩa: URL hoặc đường dẫn đến biểu đồ use case của API.

sequenceDiagram: Kiểu dữ liệu: string. Ý nghĩa: URL hoặc đường dẫn đến biểu đồ trình tự của API.

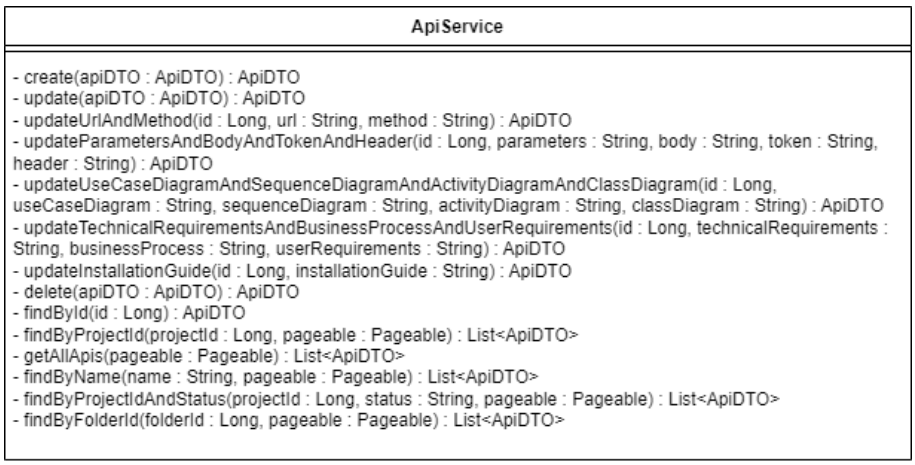
activityDiagram: Kiểu dữ liệu: string. Ý nghĩa: URL hoặc đường dẫn đến biểu đồ hoạt động của API.

classDiagram: Kiểu dữ liệu: string. Ý nghĩa: URL hoặc đường dẫn đến biểu đồ lớp của API.

installationGuide: Kiểu dữ liệu: string. Ý nghĩa: Hướng dẫn cài đặt API.

lifeCycle: Kiểu dữ liệu: LifeCycle. Ý nghĩa: Chu kỳ sống của API, gồm các giai đoạn như define, design, develop, test.

Biểu đồ chi tiết lớp Api service:



Hình 4.7: Lớp ApiService

create(apiDTO : ApiDTO) : ApiDTO: Kiểu dữ liệu: ApiDTO. Ý nghĩa: Tạo mới một API từ đối tượng ApiDTO và trả về đối tượng ApiDTO đã được tạo.

update(apiDTO : ApiDTO) : ApiDTO: Kiểu dữ liệu: ApiDTO. Ý nghĩa: Cập nhật thông tin của một API từ đối tượng ApiDTO và trả về đối tượng ApiDTO đã được cập nhật.

updateUrlAndMethod(id : Long, url : String, method : String) : ApiDTO: Kiểu dữ liệu: ApiDTO. Ý nghĩa: Cập nhật URL và phương thức HTTP của một API theo ID và trả về đối tượng ApiDTO đã được cập nhật.

updateInstallationGuide(id : Long, installationGuide : String) : ApiDTO: Kiểu dữ liệu: ApiDTO. Ý nghĩa: Cập nhật hướng dẫn cài đặt của một API theo ID và trả về đối tượng ApiDTO đã được cập nhật.

delete(apiDTO : ApiDTO) : ApiDTO: Kiểu dữ liệu: ApiDTO. Ý nghĩa: Xóa một API từ đối tượng ApiDTO và trả về đối tượng ApiDTO đã được xóa.

findById(id : Long) : ApiDTO: Kiểu dữ liệu: ApiDTO. Ý nghĩa: Tìm một API theo ID và trả về đối tượng ApiDTO tương ứng.

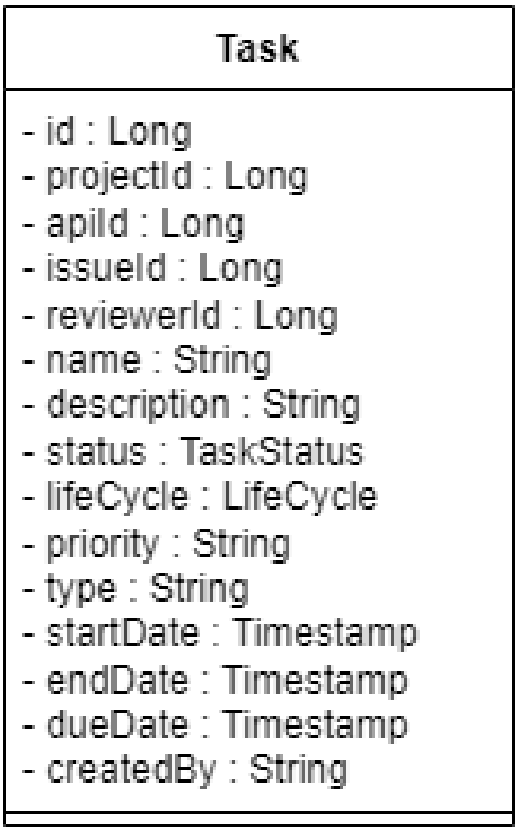
findByProjectId(projectId : Long, pageable : Pageable) : List<ApiDTO>: Kiểu dữ liệu: List<ApiDTO>. Ý nghĩa: Tìm các API theo ID của dự án và trả về danh sách các đối tượng ApiDTO.

findByProjectIdAndStatus(projectId : Long, status : String, pageable : Pageable) : List<ApiDTO>: Kiểu dữ liệu: List<ApiDTO>. Ý nghĩa: Tìm các API theo ID của dự án và trạng thái, trả về danh sách các đối tượng ApiDTO.

findByFolderId(folderId : Long, pageable : Pageable) : List<ApiDTO>: Kiểu dữ liệu: List<ApiDTO>. Ý nghĩa: Tìm các API theo ID của thư mục và trả về

danh sách các đối tượng ApiDTO.

Biểu đồ chi tiết lớp Task



Hình 4.8: Lớp Task

id: Kiểu dữ liệu: Long. Ý nghĩa: Định danh duy nhất cho Task.

projectId: Kiểu dữ liệu: Long. Ý nghĩa: Định danh của dự án mà Task thuộc về.

apiId: Kiểu dữ liệu: Long. Ý nghĩa: Định danh của API liên quan đến Task, nếu có.

issueId: Kiểu dữ liệu: Long. Ý nghĩa: Định danh của Issue liên quan đến Task, nếu có.

reviewerId: Kiểu dữ liệu: Long. Ý nghĩa: Định danh của người kiểm tra Task.

name: Kiểu dữ liệu: String. Ý nghĩa: Tên của Task.

description: Kiểu dữ liệu: String. Ý nghĩa: Mô tả ngắn gọn về Task.

status: Kiểu dữ liệu: TaskStatus. Ý nghĩa: Trạng thái hiện tại của Task (Pending, In Progress, Completed, v.v.).

lifeCycle: Kiểu dữ liệu: LifeCycle. Ý nghĩa: Chu kỳ sống của Task, gồm các giai đoạn như define, design, develop, test.

priority: Kiểu dữ liệu: String. Ý nghĩa: Độ ưu tiên của Task (High, Medium, Low).

type: Kiểu dữ liệu: String. Ý nghĩa: Loại Task (Bug, Feature, Improvement, v.v.).

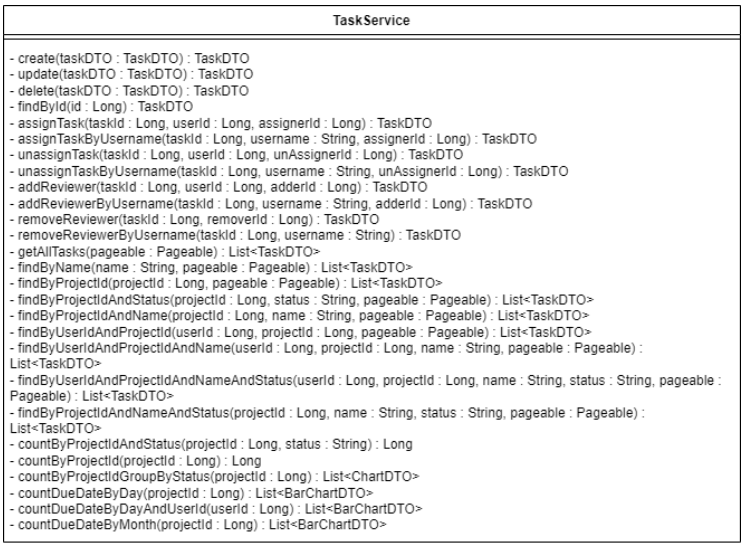
startDate: Kiểu dữ liệu: Timestamp. Ý nghĩa: Ngày bắt đầu của Task.

endDate: Kiểu dữ liệu: Timestamp. Ý nghĩa: Ngày kết thúc của Task.

dueDate: Kiểu dữ liệu: Timestamp. Ý nghĩa: Hạn chót của Task.

createdBy: Kiểu dữ liệu: String. Ý nghĩa: Định danh của người tạo Task.

Biểu đồ chi tiết lớp Task Service:



Hình 4.9: Lớp TaskService

create(TaskDTO taskDTO): Kiểu dữ liệu: TaskDTO. Ý nghĩa: Tạo mới một Task từ đối tượng TaskDTO và trả về đối tượng TaskDTO đã được tạo.

update(TaskDTO taskDTO): Kiểu dữ liệu: TaskDTO. Ý nghĩa: Cập nhật thông tin của một Task từ đối tượng TaskDTO và trả về đối tượng TaskDTO đã được cập nhật.

delete(TaskDTO taskDTO): Kiểu dữ liệu: TaskDTO. Ý nghĩa: Xóa một Task từ đối tượng TaskDTO và trả về đối tượng TaskDTO đã được xóa.

findById(Long id): Kiểu dữ liệu: TaskDTO. Ý nghĩa: Tìm một Task theo ID và trả về đối tượng TaskDTO tương ứng.

assignTask(Long taskId, Long userId, Long assignerId): Kiểu dữ liệu: TaskDTO. Ý nghĩa: Giao một Task cho một người dùng theo ID và trả về đối tượng TaskDTO đã được giao.

assignTaskByUsername(Long taskId, String username, Long assignerId): Kiểu dữ liệu: TaskDTO. Ý nghĩa: Giao một Task cho một người dùng theo tên đăng nhập và trả về đối tượng TaskDTO đã được giao.

unassignTask(Long taskId, Long userId, Long unAssignerId): Kiểu dữ liệu: TaskDTO. Ý nghĩa: Hủy giao một Task từ một người dùng theo ID và trả về đối tượng TaskDTO đã được hủy giao.

unassignTaskByUsername(Long taskId, String username, Long unAssignerId): Kiểu dữ liệu: TaskDTO. Ý nghĩa: Hủy giao một Task từ một người dùng theo tên đăng nhập và trả về đối tượng TaskDTO đã được hủy giao.

addReviewer(Long taskId, Long userId, Long adderId): Kiểu dữ liệu: TaskDTO. Ý nghĩa: Thêm một người kiểm tra Task theo ID và trả về đối tượng TaskDTO đã được thêm.

addReviewerByUsername(Long taskId, String username, Long adderId): Kiểu dữ liệu: TaskDTO. Ý nghĩa: Thêm một người kiểm tra Task theo tên đăng nhập và trả về đối tượng TaskDTO đã được thêm.

removeReviewer(Long taskId, Long removerId): Kiểu dữ liệu: TaskDTO. Ý nghĩa: Xóa một người kiểm tra Task theo ID và trả về đối tượng TaskDTO đã được xóa.

removeReviewerByUsername(Long taskId, String username): Kiểu dữ liệu: TaskDTO. Ý nghĩa: Xóa một người kiểm tra Task theo tên đăng nhập và trả về đối tượng TaskDTO đã được xóa.

getAllTasks(Pageable pageable): Kiểu dữ liệu: List<TaskDTO>. Ý nghĩa: Lấy tất cả các Task và trả về danh sách các đối tượng TaskDTO.

findByName(String name, Pageable pageable): Kiểu dữ liệu: List<TaskDTO>. Ý nghĩa: Tìm các Task theo tên và trả về danh sách các đối tượng TaskDTO.

findByProjectId(Long projectId, Pageable pageable): Kiểu dữ liệu: List<TaskDTO>. Ý nghĩa: Tìm các Task theo ID của dự án và trả về danh sách các đối tượng TaskDTO.

findByProjectIdAndStatus(Long projectId, String status, Pageable pageable): Kiểu dữ liệu: List<TaskDTO>. Ý nghĩa: Tìm các Task theo ID của dự án và trạng thái, trả về danh sách các đối tượng TaskDTO.

findByProjectIdAndName(Long projectId, String name, Pageable pageable): Kiểu dữ liệu: List<TaskDTO>. Ý nghĩa: Tìm các Task theo ID của dự án và tên, trả về danh sách các đối tượng TaskDTO.

findByUserIdAndProjectId(Long userId, Long projectId, Pageable pageable): Kiểu dữ liệu: List<TaskDTO>. Ý nghĩa: Tìm các Task theo ID của người dùng và ID của dự án, trả về danh sách các đối tượng TaskDTO.

findByUserIdAndProjectIdAndName(Long userId, Long projectId, String name, Pageable pageable): Kiểu dữ liệu: List<TaskDTO>. Ý nghĩa: Tìm các Task theo ID của người dùng, ID của dự án và tên, trả về danh sách các đối tượng TaskDTO.

findByUserIdAndProjectIdAndNameAndStatus(Long userId, Long projectId, String name, String status, Pageable pageable): Kiểu dữ liệu: List<TaskDTO>. Ý nghĩa: Tìm các Task theo ID của người dùng, ID của dự án, tên và trạng thái, trả về danh sách các đối tượng TaskDTO.

findByProjectIdAndNameAndStatus(Long projectId, String name, String status, Pageable pageable): Kiểu dữ liệu: List<TaskDTO>. Ý nghĩa: Tìm các Task theo ID của dự án, tên và trạng thái, trả về danh sách các đối tượng TaskDTO.

countByProjectIdAndStatus(Long projectId, String status): Kiểu dữ liệu: Long. Ý nghĩa: Đếm số lượng Task theo ID của dự án và trạng thái.

countByProjectId(Long projectId): Kiểu dữ liệu: Long. Ý nghĩa: Đếm số lượng Task theo ID của dự án.

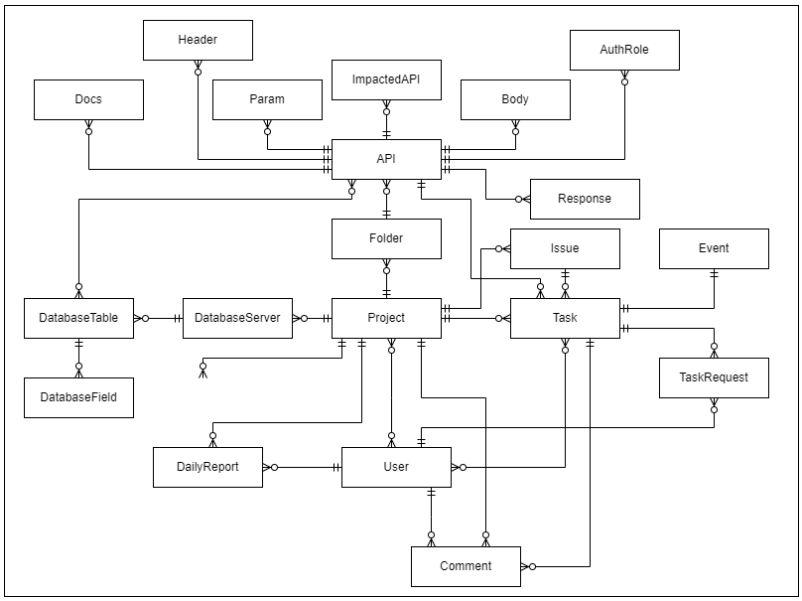
countByProjectIdGroupByStatus(Long projectId): Kiểu dữ liệu: List<ChartDTO>. Ý nghĩa: Đếm số lượng Task theo ID của dự án và nhóm theo trạng thái, trả về danh sách đối tượng ChartDTO.

countDueDateByDay(Long projectId): Kiểu dữ liệu: List<BarChartDTO>. Ý nghĩa: Đếm số lượng Task theo hạn chót trong ngày, trả về danh sách đối tượng BarChartDTO.

countDueDateByDayAndUserId(Long userId): Kiểu dữ liệu: List<BarChartDTO>. Ý nghĩa: Đếm số lượng Task theo hạn chót trong ngày và ID của người dùng, trả về danh sách đối tượng BarChartDTO.

countDueDateByMonth(Long projectId): Kiểu dữ liệu: List<BarChartDTO>. Ý nghĩa: Đếm số lượng Task theo hạn chót trong tháng, trả về danh sách đối tượng BarChartDTO.

4.2.3 Thiết kế cơ sở dữ liệu



Hình 4.10: Sơ đồ quan hệ thực thể

Thuộc tính	**Kiểu dữ liệu**	**Giải thích**
id	Long	Khóa chính, tự động tăng
projectId	Long	ID của dự án liên quan đến API
folderId	Long	ID của thư mục chứa API
name	String	Tên của API
description	String	Mô tả chi tiết về API
url	String	URL endpoint của API
method	String	Phương thức HTTP (GET, POST, PUT, DELETE, ...)
createdBy	Long	ID của người tạo API
status	String	Trạng thái của API
parameters	String	Tham số truyền vào của API
bodyJson	String	Nội dung body của request (dạng JSON)
token	String	Token xác thực cho API
userRequirements	String	Yêu cầu người dùng
technicalRequirements	String	Yêu cầu kỹ thuật
businessProcess	String	Quy trình nghiệp vụ
useCaseDiagram	String	Biểu đồ use case của API
sequenceDiagram	String	Biểu đồ trình tự của API
activityDiagram	String	Biểu đồ hoạt động của API
classDiagram	String	Biểu đồ lớp của API
installationGuide	String	Hướng dẫn cài đặt API
lifeCycle	String	Vòng đời của API

Bảng 4.1: Chi tiết dữ liệu cho Api

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính, tự động tăng
apiId	Long	ID của API chính
impactApiId	Long	ID của API bị ảnh hưởng
apiImpactName	String	Tên của API bị ảnh hưởng
status	String	Trạng thái của API bị ảnh hưởng
impactDescription	String	Mô tả chi tiết về ảnh hưởng
impactPriority	String	Mức độ ưu tiên của ảnh hưởng
solution	String	Giải pháp để khắc phục ảnh hưởng

Bảng 4.2: Chi tiết dữ liệu cho ApiImpact

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính, tự động tăng
bodyKey	String	Khóa của body
type	String	Kiểu dữ liệu của body
description	String	Mô tả chi tiết về body
apiId	Long	ID của API
sample	String	Mẫu dữ liệu của body

Bảng 4.3: Chi tiết dữ liệu cho Body

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính, tự động tăng
content	String	Nội dung của bình luận
taskId	Long	ID của task liên quan
userId	Long	ID của người dùng đã bình luận
createdAt	Timestamp	Thời điểm bình luận được tạo

Bảng 4.4: Chi tiết dữ liệu cho Comment

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
createdBy	Long	ID của người tạo báo cáo
projectId	Long	ID của dự án liên quan
name	String	Tên báo cáo
description	String	Mô tả chi tiết về báo cáo
date	Timestamp	Ngày tạo báo cáo

Bảng 4.5: Chi tiết dữ liệu cho DailyReport

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
fieldName	String	Tên của trường cơ sở dữ liệu
type	String	Kiểu dữ liệu của trường
description	String	Mô tả chi tiết về trường
databaseTableId	Long	ID của bảng cơ sở dữ liệu liên quan
sample	String	Mẫu dữ liệu của trường

Bảng 4.6: Chi tiết dữ liệu cho DatabaseField

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
projectId	Long	ID của dự án
description	String	Mô tả về server cơ sở dữ liệu
name	String	Tên của server cơ sở dữ liệu
type	String	Loại cơ sở dữ liệu (ví dụ: MySQL, PostgreSQL)
url	String	Địa chỉ URL của server cơ sở dữ liệu
username	String	Tên đăng nhập để truy cập server
password	String	Mật khẩu để truy cập server

Bảng 4.7: Chi tiết dữ liệu cho DatabaseServer

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
name	String	Tên của bảng cơ sở dữ liệu
description	String	Mô tả về bảng cơ sở dữ liệu
databaseServerId	Long	ID của server cơ sở dữ liệu chứa bảng
uuid	String	Mã định danh duy nhất cho bảng

Bảng 4.8: Chi tiết dữ liệu cho DatabaseTable

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
description	String	Mô tả về response
value	String	Giá trị của response
apiId	Long	ID của API liên quan

Bảng 4.9: Chi tiết dữ liệu cho Response

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
paramKey	String	Tên của tham số
type	String	Kiểu dữ liệu của tham số
description	String	Mô tả tham số
apiId	Long	ID của API liên quan
sample	String	Ví dụ mẫu của tham số

Bảng 4.10: Chi tiết dữ liệu cho Param

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
projectId	Long	ID của dự án
apiId	Long	ID của API liên quan
issueId	Long	ID của Issue liên quan
reviewerId	Long	ID của người kiểm duyệt
name	String	Tên của task
description	String	Mô tả task
status	String	Trạng thái của task
lifeCycle	String	Vòng đời của task
priority	String	Mức độ ưu tiên của task
type	String	Loại task
startDate	Timestamp	Ngày bắt đầu
endDate	Timestamp	Ngày kết thúc
dueDate	Timestamp	Ngày hết hạn
createdBy	String	Người tạo

Bảng 4.11: Chi tiết dữ liệu cho Task

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
name	String	Tên của sự kiện
description	String	Mô tả về sự kiện
taskId	Long	ID của task liên quan
projectId	Long	ID của dự án liên quan
status	String	Trạng thái của sự kiện
priority	String	Mức độ ưu tiên của sự kiện
startDate	Timestamp	Ngày bắt đầu sự kiện
endDate	Timestamp	Ngày kết thúc sự kiện

Bảng 4.12: Chi tiết dữ liệu cho Event

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
name	String	Tên của thư mục
projectId	Long	ID của dự án liên quan

Bảng 4.13: Chi tiết dữ liệu cho Folder

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
headerKey	String	Tên của header
type	String	Kiểu dữ liệu của header
description	String	Mô tả về header
apiId	Long	ID của API liên quan
sample	String	Ví dụ mẫu của header

Bảng 4.14: Chi tiết dữ liệu cho Header

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
projectId	Long	ID của dự án
name	String	Tên của file
description	String	Mô tả file
type	String	Loại file
uuid	String	Mã định danh duy nhất của file
localPath	String	Đường dẫn lưu trữ file nội bộ
url	String	URL truy cập file

Bảng 4.15: Chi tiết dữ liệu cho ProjectFile

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
projectId	Long	ID của dự án
apiId	String	ID của API liên quan
description	String	Mô tả về issue
content	String	Nội dung chi tiết của issue
url	String	Đường dẫn liên quan đến issue
status	String	Trạng thái của issue
priority	String	Mức độ ưu tiên của issue
createdBy	String	Người tạo issue
createdAt	Timestamp	Thời gian tạo issue
solvedAt	Timestamp	Thời gian giải quyết issue

Bảng 4.16: Chi tiết dữ liệu cho Issue

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
name	String	Tên của dự án
description	String	Mô tả dự án
leaderId	Long	ID của người lãnh đạo dự án
status	String	Trạng thái của dự án
startDate	Date	Ngày bắt đầu dự án
expectedEndDate	Date	Ngày kết thúc dự kiến
version	String	Phiên bản của dự án
coverImage	String	Đường dẫn ảnh bìa của dự án
numberOfMembers	Long	Số lượng thành viên

Bảng 4.17: Chi tiết dữ liệu cho Project

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
description	String	Mô tả về response
value	String	Giá trị của response
apiId	Long	ID của API liên quan

Bảng 4.18: Chi tiết dữ liệu cho Docs

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính, tự động tăng
apiId	Long	ID của API
role	String	Vai trò của người dùng trong dự án

Bảng 4.19: Chi tiết dữ liệu cho AuthRole

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
projectId	Long	ID của dự án
description	String	Mô tả yêu cầu
url	String	URL liên quan đến yêu cầu
content	String	Nội dung của yêu cầu
userId	Long	ID của người dùng tạo yêu cầu
taskId	Long	ID của task liên quan
createdAt	Timestamp	Thời gian tạo yêu cầu

Bảng 4.20: Chi tiết dữ liệu cho TaskRequest

Thuộc tính	Kiểu dữ liệu	Giải thích
id	Long	Khóa chính
name	String	Tên người dùng
phoneNumber	String	Số điện thoại
email	String	Địa chỉ email
username	String	Tên đăng nhập
password	String	Mật khẩu
age	Long	Tuổi của người dùng
avatar	String	URL của ảnh đại diện
createdAt	LocalDateTime	Thời gian tạo tài khoản
lastLogin	LocalDateTime	Thời gian đăng nhập lần cuối

Bảng 4.21: Chi tiết dữ liệu cho User

4.3 Xây dựng ứng dụng

4.3.1 Thư viện và công cụ sử dụng

Mục đích	Công cụ/Thư viện	Địa chỉ URL
Kết nối MySQL	MySQL Connector	`https://mvnrepository.com/artifact/mysql/mysql-connector-java`
IDE lập trình Backend	IntelliJ IDEA	`https://www.jetbrains.com/idea/`
Framework backend	Spring Boot	`https://spring.io/projects/spring-boot`
IDE lập trình Frontend	Visual Studio Code	`https://code.visualstudio.com/`
Xây dựng giao diện người dùng	React	`https://reactjs.org/`
Thiết kế giao diện	Bootstrap	`https://getbootstrap.com/`
Điều hướng trong React	React Router DOM	`https://reactrouter.com/`
Đóng gói ứng dụng	Docker	`https://www.docker.com/`

Bảng 4.22: Danh sách công cụ và thư viện sử dụng

4.3.2 Kết quả đạt được

Dự án quản lý API Document đã phát triển và triển khai thành công, giải quyết hiệu quả các vấn đề liên quan đến quản lý tài liệu API, bao gồm việc quản lý người dùng, dự án, API Document, task, issue, file, lịch trình, báo cáo hàng ngày và cơ

sở dữ liệu. Các module này được tích hợp thành một ứng dụng hoàn chỉnh, mỗi module đảm nhận vai trò cụ thể và hỗ trợ các chức năng quản lý khác nhau trong dự án. Dưới đây là mô tả chi tiết về các module chính và vai trò của chúng:

- **User Management:**
  - **Thành phần:** Quản lý thông tin người dùng, bao gồm việc đăng ký, đăng nhập, cập nhật thông tin cá nhân, và phân quyền.
  - **Ý nghĩa:** Đảm bảo rằng chỉ những người dùng có quyền hợp lệ mới có thể truy cập và thực hiện các chức năng trong hệ thống.
  - **Vai trò:** Bảo mật hệ thống, quản lý quyền truy cập và phân công vai trò cho từng người dùng.
- **Project Management:**
  - **Thành phần:** Quản lý thông tin dự án, cho phép tạo mới, sửa đổi, xóa và xem chi tiết các dự án. Bao gồm cả việc quản lý trạng thái dự án và thành viên tham gia.
  - **Ý nghĩa:** Tổ chức và quản lý hiệu quả các dự án, đảm bảo rằng tất cả các thông tin dự án được lưu trữ và truy cập dễ dàng.
  - **Vai trò:** Quản lý toàn bộ vòng đời của dự án từ khởi tạo đến hoàn thành.
- **API Management:**
  - **Thành phần:** Quản lý thông tin API, bao gồm việc định nghĩa, thiết kế, phát triển và kiểm thử API. Hỗ trợ các giai đoạn phát triển từ định nghĩa yêu cầu, thiết kế cấu trúc, phát triển mã nguồn, đến kiểm thử chức năng.
  - **Ý nghĩa:** Cung cấp một hệ thống quản lý API toàn diện, giúp theo dõi và quản lý quá trình phát triển API một cách chặt chẽ và khoa học.
  - **Vai trò:** Đảm bảo rằng các API được phát triển theo quy trình chuẩn, giảm thiểu lỗi và tăng hiệu quả phát triển.
- **Task Management:**
  - **Thành phần:** Quản lý các task trong dự án, cho phép tạo, sửa, xóa, giao và theo dõi tiến độ các task. Hỗ trợ phân loại, ưu tiên và trạng thái của từng task.
  - **Ý nghĩa:** Tăng cường quản lý công việc, đảm bảo rằng tất cả các task được phân công và theo dõi một cách rõ ràng.
  - **Vai trò:** Giúp các thành viên dự án biết được task của mình và tiến độ thực hiện, từ đó nâng cao hiệu suất làm việc.

- **Issue Management:**
  - **Thành phần:** Quản lý các vấn đề phát sinh trong dự án, bao gồm việc tạo mới, sửa đổi, xóa và theo dõi tiến độ giải quyết các vấn đề.
  - **Ý nghĩa:** Đảm bảo rằng tất cả các vấn đề trong dự án được ghi nhận, theo dõi và giải quyết kịp thời.
  - **Vai trò:** Giúp giảm thiểu rủi ro và giải quyết các vấn đề một cách nhanh chóng và hiệu quả.
- **File Management:**
  - **Thành phần:** Quản lý các file chung của dự án, bao gồm việc tải lên, xóa và xem chi tiết các file.
  - **Ý nghĩa:** Cung cấp một nơi lưu trữ tập trung cho tất cả các tài liệu liên quan đến dự án, giúp dễ dàng truy cập và chia sẻ.
  - **Vai trò:** Hỗ trợ quản lý tài liệu dự án, đảm bảo rằng các file quan trọng luôn được bảo mật và dễ dàng truy cập.
- **Scheduler:**
  - **Thành phần:** Hiển thị lịch trình và quản lý các sự kiện liên quan đến các task và milestone trong dự án.
  - **Ý nghĩa:** Giúp lập kế hoạch và theo dõi tiến độ dự án một cách trực quan và hiệu quả.
  - **Vai trò:** Tạo điều kiện thuận lợi cho việc quản lý thời gian và tiến độ của các hoạt động dự án.
- **Daily Report:**
  - **Thành phần:** Quản lý và theo dõi báo cáo hàng ngày của các thành viên trong dự án.
  - **Ý nghĩa:** Cung cấp thông tin cập nhật hàng ngày về tiến độ và tình trạng của các công việc trong dự án.
  - **Vai trò:** Giúp các quản lý dự án nắm bắt được tiến độ và hiệu suất làm việc của các thành viên.
- **Database Management:**
  - **Thành phần:** Quản lý thông tin về các server cơ sở dữ liệu, các bảng và trường trong cơ sở dữ liệu liên quan đến dự án.
  - **Ý nghĩa:** Đảm bảo rằng tất cả các thông tin liên quan đến cơ sở dữ liệu

được quản lý và truy cập một cách hiệu quả.

– **Vai trò:** Hỗ trợ quản lý và bảo trì cơ sở dữ liệu, đảm bảo tính toàn vẹn và an toàn của dữ liệu.

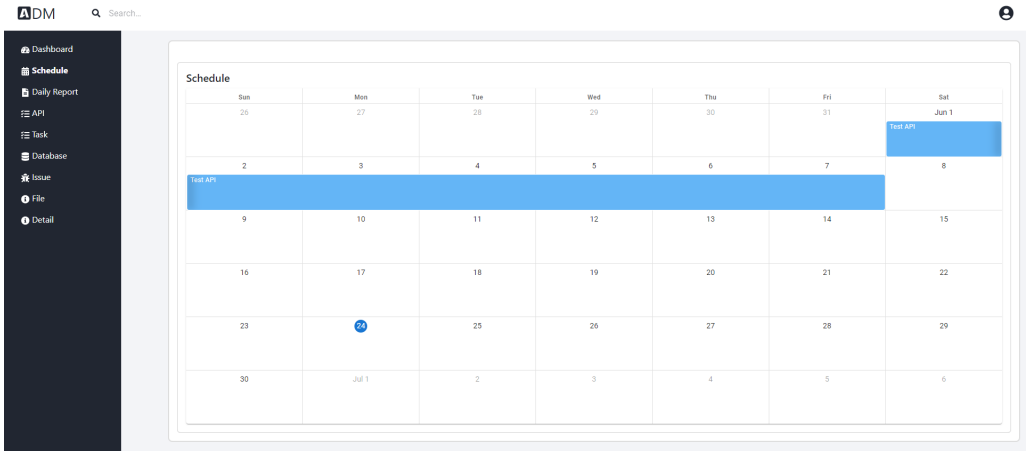
Thống kê :

Số gói: 12 gói

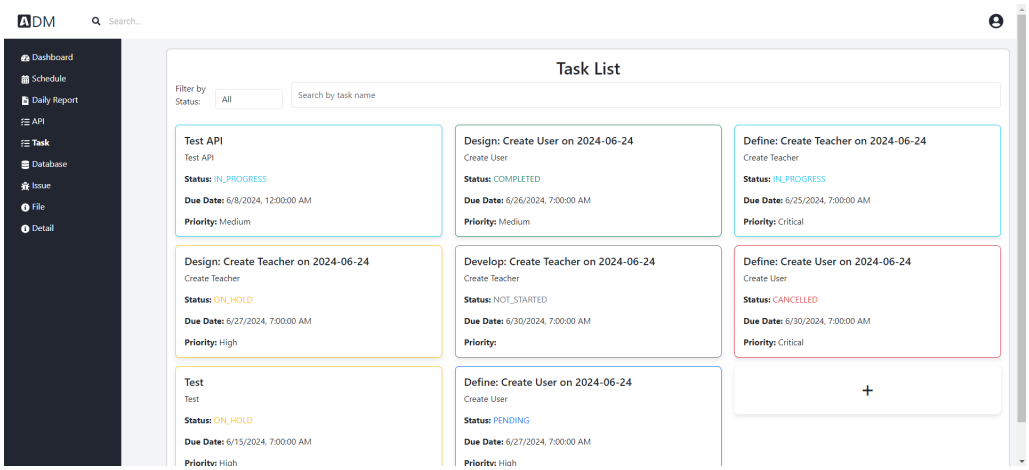
Số file: 201 file

Số dòng code: 29517 dòng code

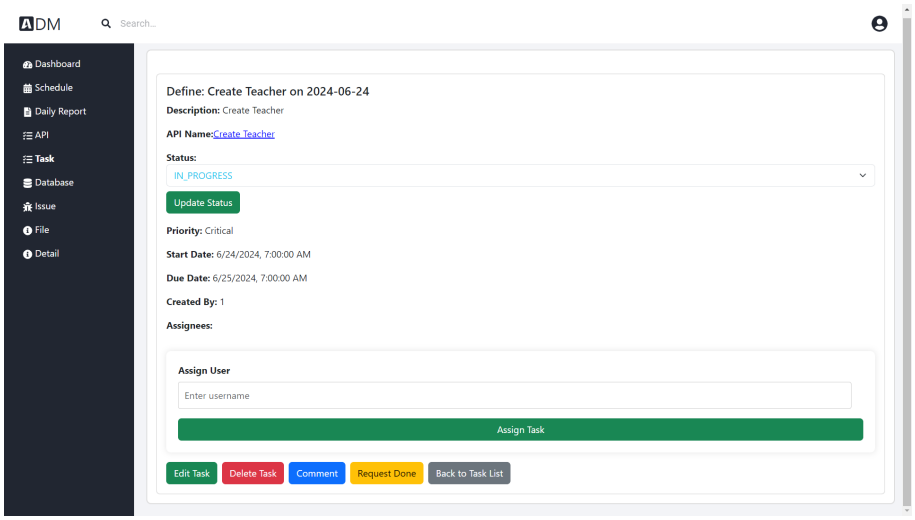
4.3.3 Minh họa các chức năng chính



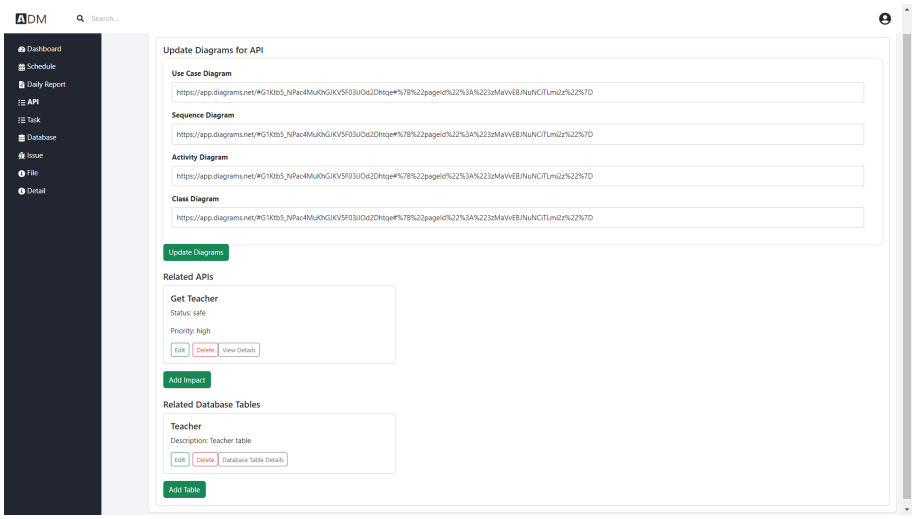
Hình 4.11: Scheduler



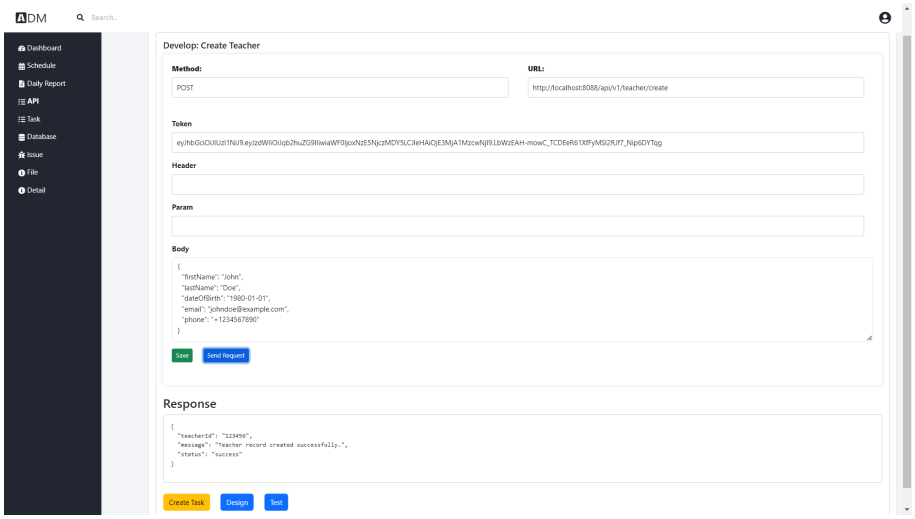
Hình 4.12: Task List



Hình 4.13: Tài liệu thiết kế API



Hình 4.14: Tài liệu thiết kế API



Hình 4.15: Tài liệu thiết kế API

Các chức năng chính sẽ được trình bày chi tiết hơn tại chương 5 trong phần 5.1 và 5.2

4.4 Kiểm thử

Bảng kiểm thử hộp đen cho các chức năng của website:

Bảng 4.23: Kiểm thử hộp đen (Phần 1)

Chức năng	**Thao tác**	**Đầu ra**	**Kết quả**
Đăng nhập	Nhập đúng username, password, bấm đăng nhập	Hiển thị danh sách dự án tham gia	Đạt
Đăng nhập	Nhập sai username, password, bấm đăng nhập	Thông báo sai username hoặc password	Đạt
Xem danh sách database của dự án	Bấm vào "Database"	Hiển thị danh sách database, các database có màu viền tương ứng với loại database	Đạt
Xem danh sách task của dự án	Bấm vào "Task"	Hiển thị danh sách task, các task có màu viền tương ứng với trạng thái	Đạt
Xem danh sách issue của dự án	Bấm vào "Issue"	Hiển thị danh sách issue, các issue có màu viền tương ứng với trạng thái	Đạt
Tạo database mới	Bấm vào nút thêm database, nhập đủ các trường, xác nhận	Database mới xuất hiện trong danh sách database, đúng các thông tin đã nhập và có màu viền tương ứng	Đạt
Tạo task mới	Bấm vào nút thêm task, nhập đủ các trường, xác nhận	Task mới xuất hiện trong danh sách task, đúng các thông tin đã nhập và có màu viền tương ứng, thông tin task cũng được cập nhật lên scheduler	Đạt
Thêm thành viên mới vào dự án	Chọn vào mục thêm thành viên, nhập username, bấm thêm	Thành viên mới xuất hiện trong phần thông tin dự án, trường số lượng thành viên tăng 1, dự án sẽ xuất hiện trong danh sách dự án tham gia của thành viên mới	Đạt

Bảng 4.24: Kiểm thử hộp đen (Phần 2)

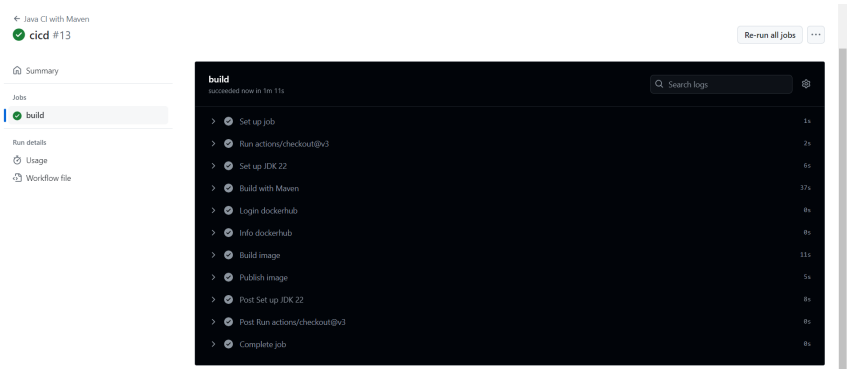
Xóa thành viên của dự án	Chọn thành viên cần xóa, bấm nút xóa, xác nhận	Thành viên bị xóa khỏi phần thông tin dự án, trường số lượng thành viên giảm 1, dự án sẽ không còn xuất hiện trong danh sách dự án tham gia của thành viên bị xóa	Đạt
Giao việc cho thành viên	Nhập username của thành viên, bấm giao task	Thành viên xuất hiện trong phần thông tin task, thành viên sẽ nhận được gmail thông báo từ gmail dự án	Đạt
Gửi yêu cầu duyệt hoàn thành task	Chọn "request done", nhập thông tin vào phần content, sau đó bấm gửi yêu cầu	Thông tin của yêu cầu hiển thị trên phần lịch sử yêu cầu, quản lý dự án nhận được gmail thông báo từ người gửi yêu cầu từ gmail dự án	Đạt

4.5 Triển khai

Website được triển khai phần backend lên máy chủ Lightsail của Amazon Web Services (AWS). Đầu tiên, cần cài đặt môi trường cho Lightsail server theo các bước sau:

1. **Cài đặt Docker trên Lightsail server:**
    - Trước tiên, cần cài đặt Docker lên máy chủ Lightsail. Docker sẽ giúp quản lý các container, tạo môi trường nhất quán cho ứng dụng.
2. **Chạy MySQL image:**
    - Sau khi Docker được cài đặt, tiến hành chạy MySQL image để tạo một container MySQL, container này có vai trò là database server của dự án.

Trong source code của dự án gồm có 1 Dockerfile và 1 file maven.yml, Dockerfile có vai trò build ra docker image của dự án. File maven.yml là một file workflows của github action, khi có lệnh push, máy ảo của github action sẽ tự động cài đặt source code, build code, đăng nhập dockerhub, chạy Dockerfile để build image cho dự án, sau đó push image vừa tạo được lên dockerhub. Sau khi image đã được đẩy lên dockerhub thông qua github action, chạy lệnh pull image từ dockerhub về server và chạy image này.



Hình 4.16: Kết quả chạy thành công file maven.yml

Link sản phẩm đã deploy: http://175.41.136.195/

Link backup deploy backend trên https://railway.app và frontend trên https://vercel.com : https://project-management-one-zeta.vercel.app/

Tài khoản đã có database : Quản lý dự án - tài khoản : theanh - mật khẩu : 123
Thành viên dự án - tài khoản : vanduc - mật khẩu : 123

CHƯƠNG 5. CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT

5.1 Phát triển hệ thống quản lý tài liệu API toàn diện

5.1.1 Giới thiệu vấn đề

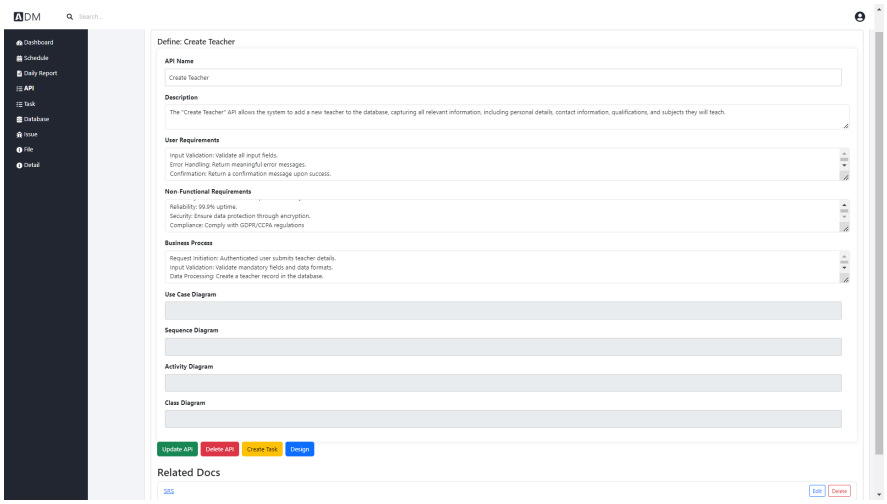
Trong thời đại công nghệ số hiện nay, API (Application Programming Interface) đóng vai trò quan trọng trong việc kết nối các hệ thống phần mềm, dịch vụ và ứng dụng với nhau. API không chỉ giúp các hệ thống giao tiếp một cách hiệu quả mà còn là nền tảng cho việc phát triển và mở rộng các chức năng phần mềm. Tuy nhiên, việc quản lý tài liệu API gặp nhiều thách thức, đặc biệt là khi số lượng API tăng lên và yêu cầu về tính nhất quán, độ tin cậy và bảo mật ngày càng cao.

Các công cụ hiện có như Postman và Swagger tuy hỗ trợ tốt việc thử nghiệm và tài liệu hóa API nhưng chưa đủ để quản lý toàn bộ vòng đời phát triển API. Cụ thể, chúng không hỗ trợ đầy đủ việc quản lý các giai đoạn phát triển API như định nghĩa, thiết kế, phát triển và kiểm thử. Điều này dẫn đến nhiều khó khăn trong việc theo dõi tiến độ, đảm bảo chất lượng và quản lý tài liệu liên quan đến API. Hơn nữa, việc thiếu tính năng tích hợp để quản lý các yêu cầu người dùng, tài liệu thiết kế chi tiết và tài liệu kiểm thử càng làm tăng thêm độ phức tạp và rủi ro trong quá trình phát triển API.

5.1.2 Giải pháp

Hệ thống quản lý tài liệu API toàn diện được phát triển nhằm giải quyết các vấn đề nêu trên, bao gồm các giai đoạn quản lý tài liệu theo từng giai đoạn: Define, Design, Develop và Test. Mỗi giai đoạn được thiết kế với các chức năng chi tiết, giúp quản lý tài liệu API một cách thông minh, dễ dàng và hiệu quả.

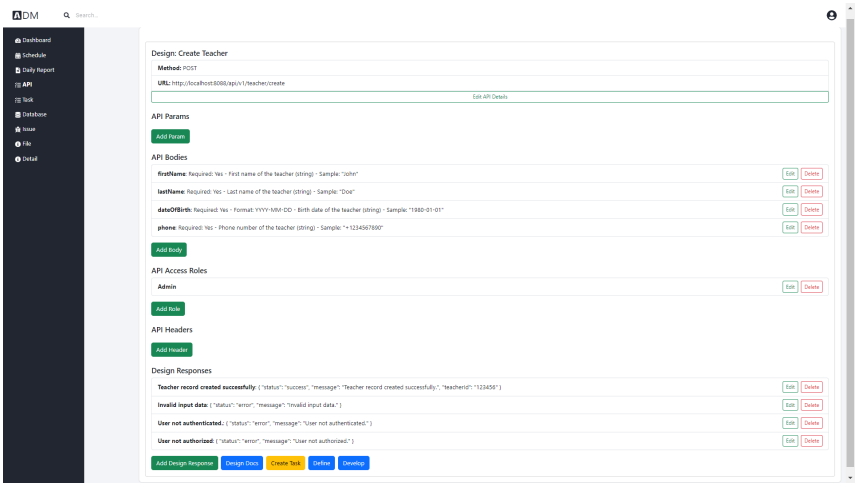
1. **Define Phase:**
    - **Yêu cầu người dùng:** Ghi nhận và quản lý các yêu cầu của người dùng về API, bao gồm các yêu cầu chức năng và phi chức năng.
    - **Yêu cầu phi chức năng:** Định nghĩa các yêu cầu về bảo mật, hiệu suất, và các tiêu chuẩn kỹ thuật cần tuân thủ.
    - **Quy trình nghiệp vụ:** Mô tả chi tiết các quy trình nghiệp vụ mà API sẽ hỗ trợ, giúp đảm bảo rằng API đáp ứng đầy đủ các yêu cầu kinh doanh.
    - **Tài liệu liên quan:** Quản lý các tài liệu liên quan như biểu đồ luồng dữ liệu, các quy tắc kinh doanh và các tài liệu mô tả khác.



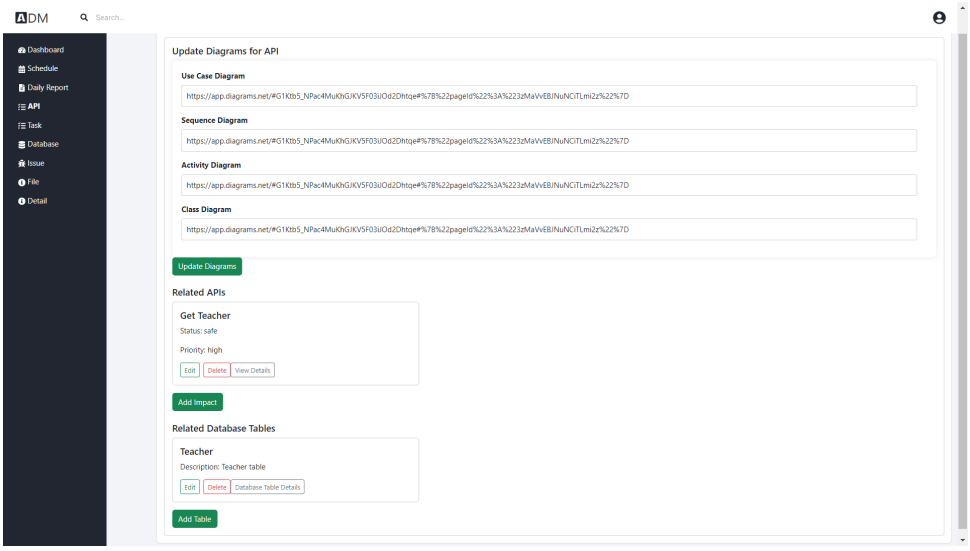
Hình 5.1: Giao diện giai đoạn khởi tạo API theo góc nhìn của quản lý dự án

2. **Design Phase:**

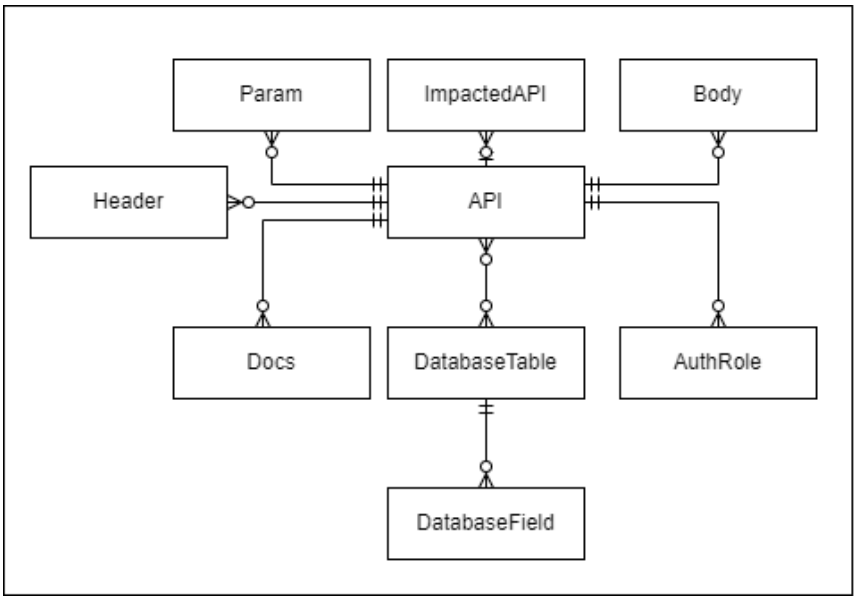
- **Thiết kế API chi tiết:** Cho phép người dùng thiết kế URL, phương thức HTTP (GET, POST, PUT, DELETE), các tham số, body của request và response, các header và vai trò truy cập cho API.
- **Tài liệu thiết kế:** Quản lý các tài liệu thiết kế API như Use Case Diagram, Sequence Diagram, Activity Diagram, Class Diagram, và các tài liệu liên quan khác.
- **Cơ sở dữ liệu liên quan:** Liên kết đến các bảng cơ sở dữ liệu liên quan, giúp người dùng dễ dàng xác định các nguồn dữ liệu mà API sẽ tương tác.
- **API bị ảnh hưởng:** Cung cấp danh sách các API có thể bị ảnh hưởng khi phát triển API hiện tại, bao gồm mô tả ảnh hưởng, mức độ ảnh hưởng, và giải pháp khắc phục.



Hình 5.2: Giao diện giai đoạn thiết kế API theo góc nhìn của quản lý dự án

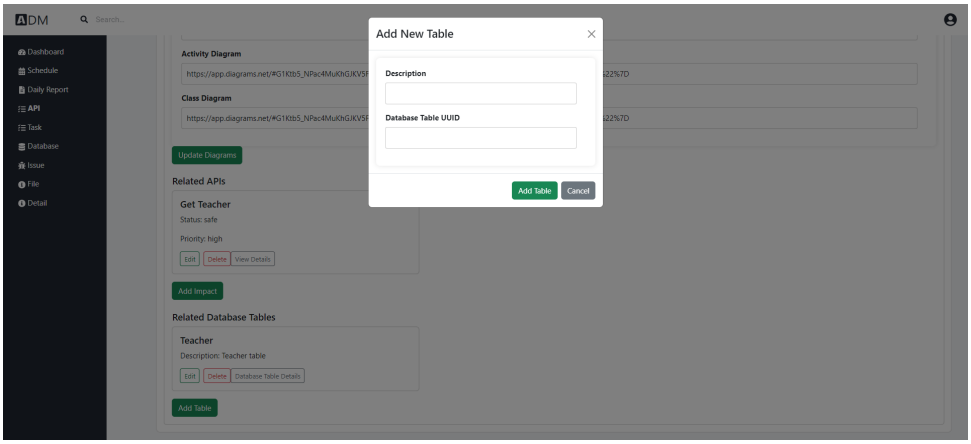


Hình 5.3: Giao diện tải liệu thiết kế API theo góc nhìn của quản lý dự án

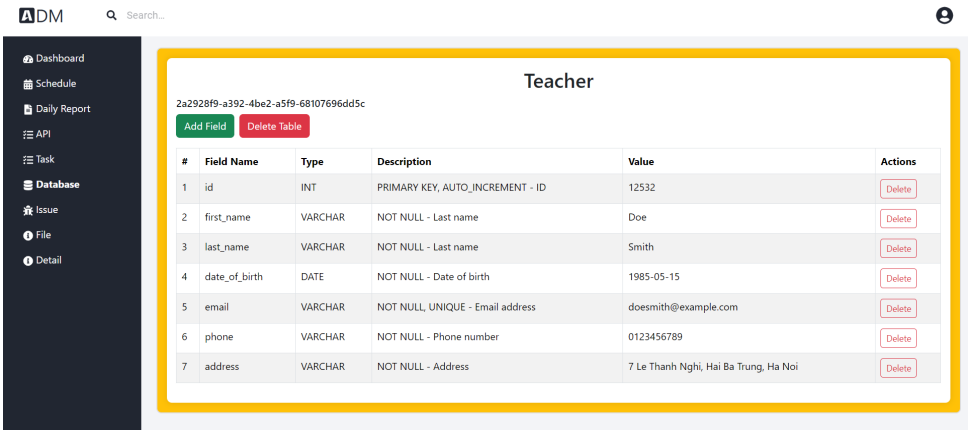


Hình 5.4: Thiết kế database lữu trữ các tài liệu của Api

Các tài liệu thiết kế của API được thiết kế chi tiết, cho phép người quản lý đầy đủ các thông tin. Mỗi khi quản lý dự án tạo 1 bảng cơ sở dữ liệu mới, hệ thống sẽ tự động tạo một UUID (Universally Unique Identifier - một chuỗi định danh duy nhất được tạo ra để nhận diện các đối tượng một cách duy nhất trên toàn cầu) cho cơ sở dữ liệu đó, người dùng sẽ dễ dàng dựa vào UUID đó để tạo liên kết giữa API và các bảng liên quan, khi nhấn vào "Database Table Details" sẽ được chuyển hướng tới trang quản lý bảng cơ sở dữ liệu đó.



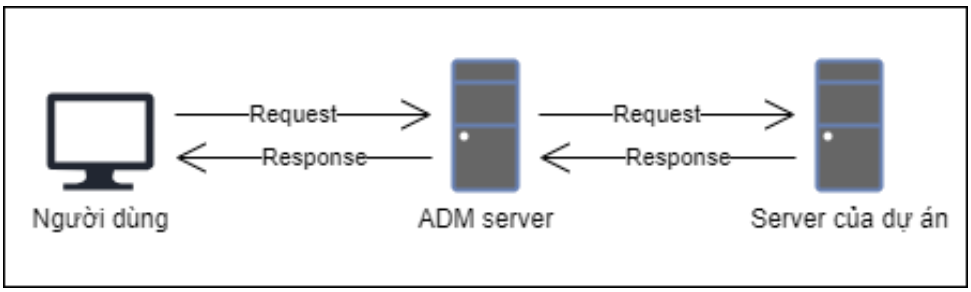
Hình 5.5: Form tạo thông tin bảng liên quan



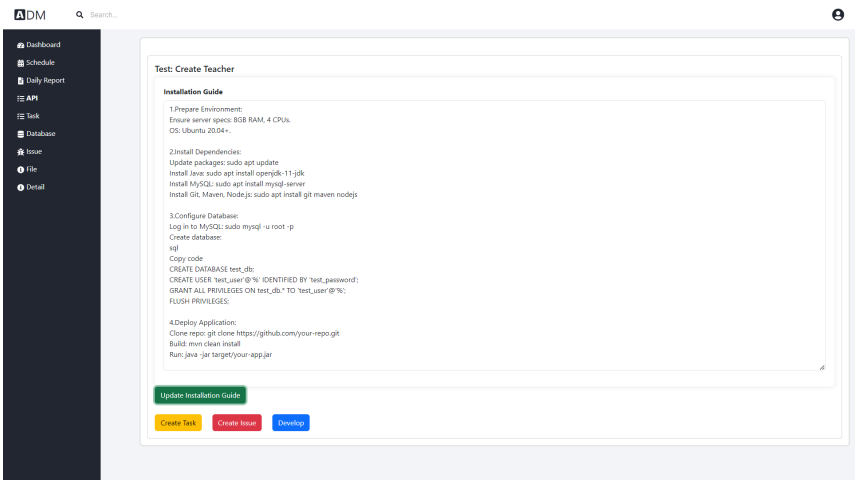
Hình 5.6: Thông tin bảng cơ sở dữ liệu

3. **Develop Phase:**

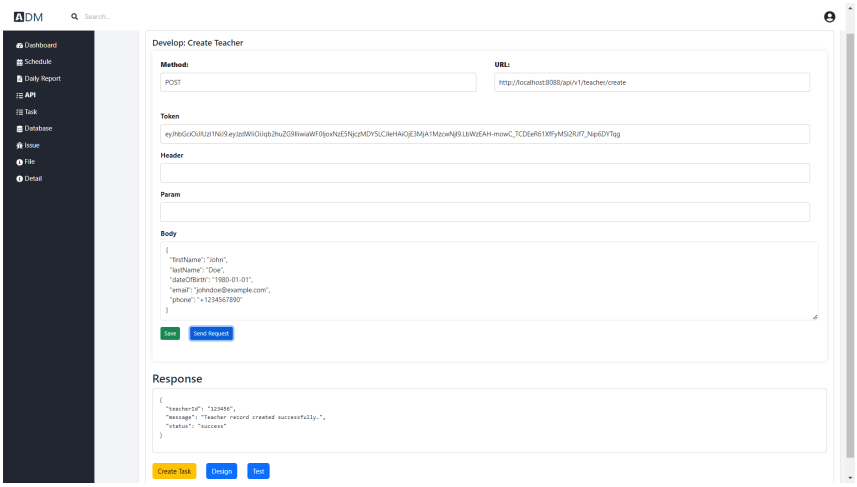
- **Test API:** Cho phép người dùng test API, khi người dùng nhấn "send", hệ thống sẽ dựa vào các thông tin được nhập vào như url, method, param, body, header để tạo request tới dự án người dùng quản lý và nhận response tương ứng.



Hình 5.7: Sơ đồ luồng gửi và nhận reponse thông qua ADM



Hình 5.9: Giao diện tài liệu test API theo góc nhìn của quản lý dự án

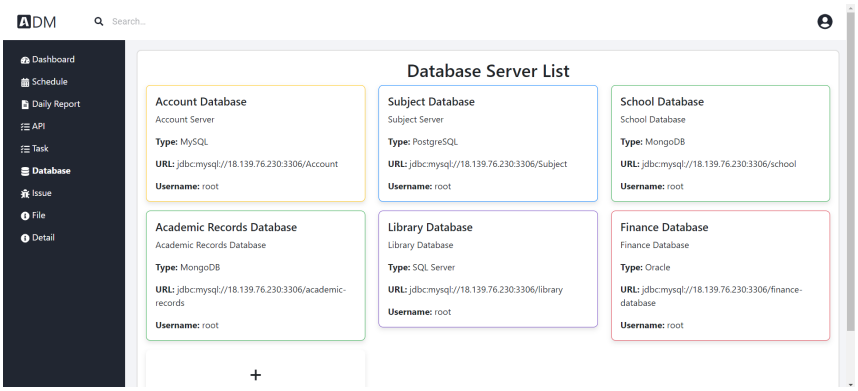


Hình 5.8: Giao diện tài liệu phát triển API theo góc nhìn của quản lý dự án

4. **Test Phase:**

   - **Hướng dẫn cài đặt môi trường:** Cung cấp hướng dẫn chi tiết về cách cài đặt môi trường test, giúp đảm bảo rằng môi trường kiểm thử giống với môi trường sản xuất.
   - **Tạo issue:** Cho phép người dùng tạo issue liên quan tới API khi xảy ra vấn đề trong quá trình kiểm thử
   - **Gửi minh chứng kiểm thử:** Cho phép người dùng gửi minh chứng kiểm thử API đã đúng thiết kế hay chưa trong phần task. Điều này giúp các thành viên trong dự án dễ dàng theo dõi các thay đổi, trạng thái của API theo từng thời điểm, nâng cao chất lượng phát triển và bảo trì.

Hệ thống còn quản lý các tài liệu khác như tài liệu liên quan đến dự án và các thông tin về database server.



Hình 5.10: Giao diện danh sách database server của dự án

- **Quản lý file:** Cho phép thành viên dự án tải lên và quản lý các file tài liệu liên quan đến dự án. Các file này có thể là tài liệu thiết kế, tài liệu hướng dẫn, báo cáo tiến độ, hoặc bất kỳ tài liệu nào khác cần thiết cho quá trình phát triển và quản lý dự án.
- **Quản lý thông tin database server:** Quản lý thông tin về các server database liên quan đến API. Bao gồm các thông tin về loại server, danh sách các bảng trong database, và các chi tiết về kết nối và quyền truy cập. Điều này giúp đảm bảo rằng API có thể truy cập và tương tác với các nguồn dữ liệu một cách hiệu quả và an toàn. Dựa vào loại hệ thống quản lý cơ sở dữ liệu (Database Management System - DBMS) mà hệ thống sẽ hiển thị màu chủ đạo cho mỗi database, nhằm tăng tính trực quan và độ thân thiện với người dùng

5.1.3 Kết quả đạt được

Giải pháp quản lý tài liệu API toàn diện đã mang lại nhiều lợi ích cho quá trình phát triển và quản lý API.

1. **Dễ dàng quản lý và tìm kiếm tài liệu:**
   - Các tài liệu được tổ chức một cách thông minh và chi tiết theo từng giai đoạn phát triển, giúp người dùng dễ dàng tìm kiếm và truy cập.
   - Việc quản lý tài liệu theo từng giai đoạn phát triển giúp đảm bảo rằng tất cả các thông tin cần thiết đều được ghi nhận và cập nhật kịp thời.
2. **Tăng cường tính nhất quán và chất lượng của API:**
   - Hệ thống giúp đảm bảo rằng tất cả các yêu cầu nghiệp vụ và phi chức năng đều được ghi nhận và hiểu rõ ràng bởi tất cả các thành viên trong dự án.
   - Các tài liệu thiết kế chi tiết giúp đảm bảo rằng API được thiết kế một cách nhất quán và đáp ứng đầy đủ các yêu cầu kỹ thuật.

3. **Tiết kiệm thời gian và công sức:**
    - Việc quản lý tài liệu một cách khoa học và hệ thống giúp tiết kiệm thời gian và công sức cho các thành viên trong dự án.
    - Hệ thống giúp quy trình hóa nhiều quy trình quản lý tài liệu, giảm thiểu rủi ro và tăng cường độ tin cậy của hệ thống.

Với giải pháp này, hệ thống quản lý tài liệu API toàn diện không chỉ đáp ứng các yêu cầu hiện tại mà còn tạo ra một nền tảng vững chắc cho việc phát triển và mở rộng trong tương lai. Hệ thống có thể dễ dàng được tùy chỉnh và mở rộng để đáp ứng các yêu cầu cụ thể của từng dự án, giúp đảm bảo sự thành công và hiệu quả của quy trình phát triển API.

5.2 Giải pháp về quản lý tiến độ dự án

5.2.1 Giới thiệu vấn đề

Trong bất kỳ dự án phát triển phần mềm nào, việc quản lý tiến độ là một thách thức lớn. Việc đảm bảo rằng mọi task được hoàn thành đúng hạn, đúng yêu cầu và phối hợp hiệu quả giữa các thành viên trong nhóm là điều không dễ dàng. Các API, với vai trò quan trọng trong việc kết nối và tương tác giữa các hệ thống, yêu cầu một quy trình quản lý chặt chẽ từ giai đoạn định nghĩa, thiết kế, phát triển đến kiểm thử. Tuy nhiên, nhiều công cụ hiện nay như Postman và Swagger chủ yếu tập trung vào việc tài liệu hóa và thử nghiệm API, chưa hỗ trợ đầy đủ cho việc quản lý tiến độ và phân công task. Nhiều công cụ quản lý dự án như JIRA[10] và Trello[11] cung cấp các chức năng quản lý task và tiến độ, nhưng chúng lại không tích hợp chặt chẽ với các công cụ quản lý tài liệu API. Điều này dẫn đến việc phải sử dụng nhiều công cụ khác nhau và gây khó khăn trong việc theo dõi và quản lý toàn diện.

Khó khăn trong việc quản lý tiến độ bao gồm việc theo dõi tiến độ của các task, quản lý trạng thái của các task, giao task cho các thành viên và nhận phản hồi từ họ. Hơn nữa, việc đảm bảo rằng các task được hoàn thành đúng thời hạn và đáp ứng đầy đủ các yêu cầu kỹ thuật và nghiệp vụ là một thách thức lớn. Điều này đòi hỏi một hệ thống quản lý tiến độ hiệu quả, có khả năng tích hợp chặt chẽ với các quy trình phát triển API, từ giai đoạn định nghĩa, thiết kế, phát triển đến kiểm thử.

Khó khăn trong việc quản lý tiến độ dự án bao gồm:

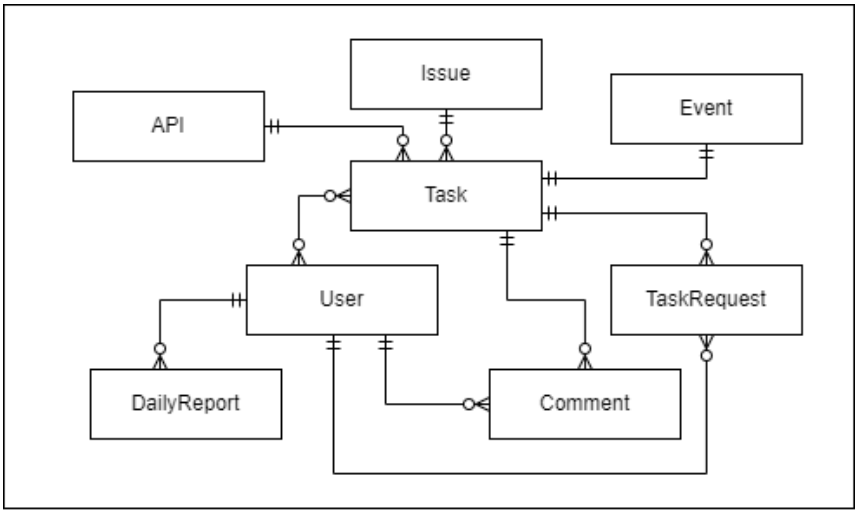
1. **Phối hợp giữa các thành viên:**
    - Đảm bảo tất cả thành viên dự án nắm bắt được tiến độ và task cần hoàn thành.
    - Khó khăn trong việc giao task và theo dõi tiến độ của từng thành viên.

- Cần có sự phối hợp liên tục và hiệu quả giữa các thành viên để đảm bảo task được hoàn thành đúng hạn và đạt chất lượng yêu cầu.

2. **Quản lý thông tin và tài liệu:**
   - Khó khăn trong việc lưu trữ và quản lý tài liệu liên quan đến từng task.
   - Việc phải chuyển đổi giữa nhiều công cụ để quản lý tài liệu API và tiến độ dự án.
   - Tài liệu liên quan đến API và tiến độ dự án thường nằm rải rác trong nhiều hệ thống, gây khó khăn trong việc tìm kiếm và theo dõi thông tin.
3. **Đảm bảo tiến độ và chất lượng:**
   - Khó khăn trong việc kiểm soát tiến độ của từng giai đoạn phát triển API.
   - Đảm bảo chất lượng API và task được thực hiện đúng theo yêu cầu.
   - Cần theo dõi sát sao tiến độ và chất lượng công việc của từng thành viên để đảm bảo dự án không bị chậm tiến độ và đạt chất lượng mong muốn.
4. **Theo dõi tiến độ và quản lý trạng thái:**
   - Việc theo dõi tiến độ của các task, quản lý trạng thái của các task, giao task cho các thành viên và nhận phản hồi từ họ là một thách thức lớn.
   - Đảm bảo rằng các task được hoàn thành đúng thời hạn và đáp ứng đầy đủ các yêu cầu kỹ thuật và nghiệp vụ.

5.2.2 Giải pháp

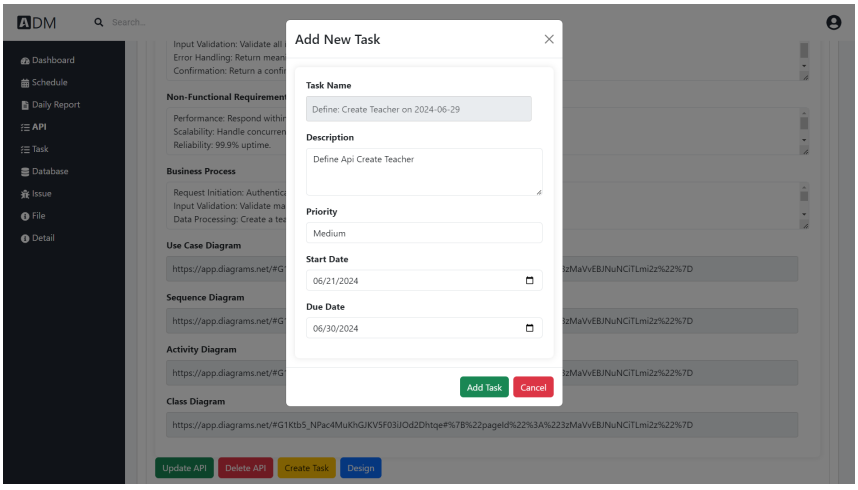
Hệ thống quản lý API Document không chỉ tập trung vào việc quản lý tài liệu mà còn tích hợp chặt chẽ với cơ chế quản lý tiến độ dự án. Giải pháp này khéo léo kết hợp quản lý tài liệu API và quản lý tiến độ phát triển vào cùng một hệ thống, giúp tối ưu hóa quy trình làm việc và tăng cường hiệu quả phối hợp giữa các thành viên dự án.



Hình 5.11: Thiết kế cơ sở dữ liệu quản lý task của 1 dự án

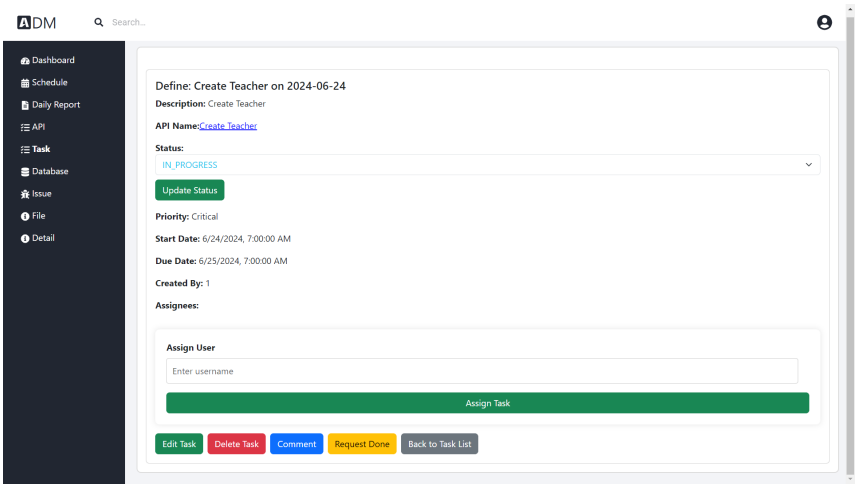
Các chức năng chính bao gồm:

1. **Tạo và giao task:** Quản lý dự án có thể dễ dàng tạo và giao task cho từng thành viên. Mỗi task đều được gắn liền với các tài liệu API liên quan, giúp đảm bảo rằng thành viên hiểu rõ yêu cầu và tiêu chí hoàn thành. Khi tạo, hệ thống sẽ tự động tạo tên task dựa trên giai đoạn, tên api và ngày tạo task, điều này giúp người dùng dễ dàng hình dung được nội dung task chỉ cần dựa vào tên.



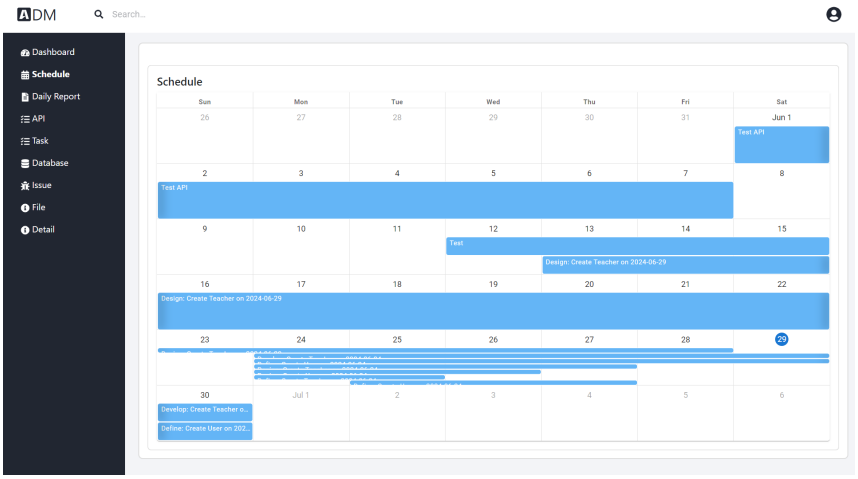
Hình 5.12: Giao diện tạo task từ giai đoạn define

[H] Trong phần chi tiết nhiệm vu sẽ có mô tả đầy đủ về task, ngoài ra sẽ có đường dẫn tới API và giai đoạn tương ứng với task ở phần "API name"



Hình 5.13: Giao diện chi tiết task

Ngoài ra, khi có task được tạo, hệ thống cũng sẽ tự động tạo hoặc cập nhật 1 event tương ứng để cập nhật vào scheduler và dashboard của dự án



Hình 5.14: Giao diện scheduler của dự án

2. **Theo dõi tiến độ và trạng thái:**
   - Hệ thống cung cấp các công cụ theo dõi tiến độ chi tiết, cho phép quản lý dự án giám sát tiến độ thực hiện từng task, từ giai đoạn định nghĩa, thiết kế, phát triển đến kiểm thử.
   - Trạng thái của task được cập nhật liên tục, giúp quản lý dự án và các thành viên dễ dàng nắm bắt được tiến độ tổng thể.
3. **Bình luận, trao đổi và phản hồi:**
   - Hệ thống hỗ trợ tính năng bình luận vào từng task, giúp các thành viên dự án trao đổi, cập nhật thông tin và giải quyết các vấn đề phát sinh một cách nhanh chóng.

- Các bình luận được lưu trữ và hiển thị theo thứ tự thời gian, giúp theo dõi các quyết định và thay đổi liên quan đến task.
- Hệ thống có chức năng gửi yêu cầu hoàn thành task nhằm gửi yêu cầu để quản lý dự án duyệt hoàn thành task và thay đổi trạng thái.

4. **Thông báo và nhắc nhở:**
   - Hệ thống sử dụng thư viện SimpleMailMessage[12] trong Springboot để tự động gửi thông báo và nhắc nhở qua email cho các thành viên liên quan khi có sự thay đổi của task.
   - Các thông báo giúp đảm bảo rằng tất cả các thành viên dự án luôn cập nhật và không bỏ lỡ các task quan trọng.
5. **Biểu đồ, lịch trình và báo cáo hàng ngày:**
   - Hệ thống cung cấp chức năng biểu đồ và lịch trình giúp các thành viên dự án theo dõi được các mốc quan trọng của dự án
   - Ngoài ra hệ thống có chức năng báo cáo hàng ngày giúp quản lý dự án nắm được tiến độ của từng người, đồng bộ hóa các công việc trong dự án và cũng đánh giá được năng lực và chất lượng làm việc của từng người.
6. **Tích hợp với quy trình phát triển API:**
   - Hệ thống quản lý tiến độ được tích hợp chặt chẽ với các giai đoạn phát triển API, từ định nghĩa, thiết kế, phát triển đến kiểm thử.
   - Mỗi task đều được liên kết với các tài liệu và tài nguyên liên quan, giúp đảm bảo tính nhất quán và chất lượng của API trong suốt quá trình phát triển.

5.2.3 Kết quả đạt được

Giải pháp quản lý tiến độ dự án đã mang lại nhiều kết quả đáng kể:

1. **Cải thiện quy trình phát triển:**
   - Hệ thống giúp quản lý và theo dõi quy trình phát triển API một cách chi tiết và hiệu quả. Các thành viên trong dự án có thể dễ dàng theo dõi tiến độ, cập nhật và quản lý các task liên quan đến API.
   - Việc tích hợp quản lý tiến độ và tài liệu giúp đảm bảo rằng mọi thông tin liên quan đều được cập nhật và theo dõi chặt chẽ.
2. **Đảm bảo chất lượng:**
   - Hệ thống đảm bảo tính nhất quán và chất lượng của API từ giai đoạn định

nghĩa đến kiểm thử. Mọi thay đổi và cập nhật đều được kiểm soát chặt chẽ, giúp giảm thiểu lỗi và đảm bảo rằng API hoạt động đúng như mong đợi.

- Việc theo dõi tiến độ giúp phát hiện và xử lý các vấn đề kịp thời, đảm bảo rằng dự án không bị chậm tiến độ.

3. **Tăng cường giao tiếp:**
    - Hệ thống cung cấp một nền tảng giao tiếp mạnh mẽ giúp các thành viên trong dự án trao đổi thông tin và cập nhật tiến độ một cách hiệu quả. Các chức năng bình luận và theo dõi giúp cải thiện sự phối hợp giữa các thành viên và tăng cường sự hiểu biết về yêu cầu và thiết kế của API.
    - Các thông báo và nhắc nhở giúp đảm bảo rằng tất cả các thành viên dự án luôn cập nhật và không bỏ lỡ các task quan trọng.
4. **Quản lý task hiệu quả:**
    - Hệ thống giúp phân loại và quản lý task một cách hiệu quả, đảm bảo rằng tất cả các yêu cầu đều được xử lý kịp thời và đúng quy trình. Điều này giúp tăng cường tính minh bạch và rõ ràng trong quản lý dự án.
    - Báo cáo và phân tích giúp quản lý dự án đánh giá tiến độ tổng thể và hiệu suất làm việc của các thành viên, từ đó đưa ra các biện pháp cải thiện kịp thời.
5. **Nâng cao hiệu suất làm việc:** Bằng cách cung cấp các công cụ và chức năng hỗ trợ quản lý tiến độ và tài liệu API, hệ thống giúp tăng cường hiệu suất làm việc của các thành viên trong dự án. Các công việc được hoàn thành nhanh chóng và hiệu quả hơn, giúp tiết kiệm thời gian và tài nguyên.

Với giải pháp quản lý tiến độ dự án, hệ thống không chỉ đạt được các mục tiêu về chất lượng và tiến độ mà còn tạo ra một nền tảng vững chắc cho việc phát triển và mở rộng trong tương lai. Hệ thống này có thể dễ dàng được tùy chỉnh và mở rộng để đáp ứng các yêu cầu cụ thể của từng dự án, giúp đảm bảo sự thành công và hiệu quả của quy trình phát triển API.

CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Trong quá trình thực hiện đồ án tốt nghiệp này, hệ thống quản lý tài liệu API đã được phát triển và triển khai thành công, mang lại nhiều tiện ích và cải thiện quy trình quản lý API cho các dự án phần mềm. So với các công cụ hiện có như Postman và Swagger, hệ thống này không chỉ dừng lại ở việc thử nghiệm và tài liệu hóa API mà còn tích hợp các giai đoạn phát triển khác nhau từ Define, Design, Develop đến Test, giúp quản lý tiến độ và tài liệu API một cách toàn diện và chi tiết hơn.

a, Kết quả đạt được

1. **Phát triển hệ thống quản lý tài liệu API toàn diện:** Hệ thống giúp theo dõi và quản lý các tài liệu API từ giai đoạn phát triển đến bảo trì, đảm bảo tính nhất quán và chất lượng của các API.
2. **Tích hợp cơ chế quản lý tiến độ dự án:** Hệ thống hỗ trợ việc giao nhiệm vụ, theo dõi tiến độ và cập nhật trạng thái công việc, giúp tối ưu hóa quy trình làm việc và tăng cường sự phối hợp giữa các thành viên trong dự án.
3. **Cải thiện quy trình phát triển API:** Hệ thống giúp các thành viên dự án dễ dàng theo dõi tiến độ, cập nhật và quản lý các task liên quan đến API, đảm bảo rằng tất cả các yêu cầu đều được xử lý kịp thời và đúng quy trình.
4. **Tăng cường giao tiếp và hợp tác:** Hệ thống cung cấp một nền tảng giao tiếp mạnh mẽ, giúp các thành viên trong dự án trao đổi thông tin và cập nhật tiến độ một cách hiệu quả.

b, Bài học kinh nghiệm

Trong suốt quá trình thực hiện đồ án, tôi đã học được nhiều bài học quan trọng:

- **Kỹ năng quản lý dự án:** Việc phân chia công việc thành các giai đoạn cụ thể và theo dõi tiến độ giúp quản lý dự án một cách hiệu quả.
- **Kỹ năng làm việc nhóm:** Sự phối hợp giữa các thành viên trong nhóm là yếu tố quan trọng để đảm bảo dự án được thực hiện đúng tiến độ và đạt chất lượng.
- **Kỹ năng giải quyết vấn đề:** Trong quá trình phát triển, nhiều vấn đề phát sinh đòi hỏi phải tìm ra giải pháp phù hợp và nhanh chóng.

Đồ án này đã giải quyết được các vấn đề cơ bản trong việc quản lý tài liệu và tiến độ phát triển API, mang lại nhiều lợi ích cho các dự án phần mềm. Hệ thống không chỉ giúp quản lý tài liệu API một cách hiệu quả mà còn tăng cường sự phối

hợp giữa các thành viên trong dự án, từ đó nâng cao chất lượng và hiệu quả làm việc. Với những định hướng phát triển trong tương lai, hệ thống hứa hẹn sẽ tiếp tục mang lại nhiều giá trị cho các doanh nghiệp phần mềm.

6.2 Hướng phát triển

a, Hoàn thiện các chức năng hiện tại

1. **Cải thiện giao diện người dùng:** Nâng cao trải nghiệm người dùng bằng cách tối ưu hóa giao diện và thêm các tính năng hữu ích.
2. **Tăng cường bảo mật:** Đảm bảo hệ thống an toàn hơn bằng cách tích hợp các biện pháp bảo mật hiện đại.
3. **Tối ưu hóa hiệu năng:** Cải thiện hiệu năng của hệ thống để đảm bảo xử lý nhanh chóng và hiệu quả các yêu cầu của người dùng.

b, Hướng đi mới

1. **Mở rộng chức năng kiểm thử:** Tích hợp các công cụ kiểm thử tự động để nâng cao chất lượng API.
2. **Tích hợp với các hệ thống khác:** Mở rộng khả năng tích hợp với các công cụ quản lý dự án và hệ thống CI/CD để tăng cường hiệu quả làm việc.
3. **Phát triển các tính năng mới:** Nghiên cứu và phát triển thêm các tính năng mới như giám sát hiệu năng API, phân tích và báo cáo chi tiết, hỗ trợ thêm các bước Deploy và Monitoring trong quy trình phát triển

TÀI LIỆU THAM KHẢO

[1] Postman, *Api lifecycle*. [Online]. Available: https://www.postman.com/api-platform/api-lifecycle/ (visited on 06/24/2024).

[2] React Documentation, *React - a javascript library for building user interfaces*. [Online]. Available: https://reactjs.org/docs/getting-started.html (visited on 06/24/2024).

[3] Spring Framework Documentation, *Spring framework reference documentation*. [Online]. Available: https://docs.spring.io/spring-framework/docs/current/reference/html/ (visited on 06/24/2024).

[4] O. Corporation, *Mysql :: The world's most popular open source database*, https://www.mysql.com/, Truy cập ngày 25/06/2024.

[5] Docker Documentation, *Docker documentation*. [Online]. Available: https://docs.docker.com/ (visited on 06/24/2024).

[6] GitHub, *Github actions documentation*. [Online]. Available: https://docs.github.com/en/actions (visited on 06/24/2024).

[7] M. B. Jones, J. Bradley, and N. Sakimura, *Json web token (jwt)*, https://tools.ietf.org/html/rfc7519, Accessed: 2024-06-24, 2015.

[8] Swagger, *Swagger documentation*. [Online]. Available: https://swagger.io/docs/ (visited on 06/24/2024).

[9] A. W. Services, *Amazon api gateway documentation*. [Online]. Available: https://docs.aws.amazon.com/apigateway/ (visited on 06/24/2024).

[10] Atlassian, *Jira software documentation*. [Online]. Available: https://www.atlassian.com/software/jira/guides (visited on 06/24/2024).

[11] Atlassian, *Trello documentation*. [Online]. Available: https://help.trello.com/ (visited on 06/24/2024).

[12] Spring, *Simplemailmessage api documentation*. [Online]. Available: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/mail/SimpleMailMessage.html (visited on 06/24/2024).