BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THÀNH PHỐ HỒ CHÍ MINH

**KHOA khoa CÔNG NGHỆ THÔNG TIN**



**MÔN: KIẾN TRÚC PHẦN MỀM**

**ĐỀ CƯƠNG NGHIÊN CỨU**

***Đề tài:***

**ĐĂNG KÝ HỌC PHẦN**

**Môn học:** Kiến trúc và Thiết kế phần mềm

**Lớp:** DHKTPM16A

**GVHD:** Võ Văn Hải

Nhóm 3 – Sinh viên thực hiện

1. La Minh Tâm – 21023911
2. Võ Đình Thông – 21082081

Thành phố Hồ Chí Minh, 19 tháng 05 năm 2024

Nội dung

[CHƯƠNG 1 MÔ TẢ YÊU CẦU 3](#_Toc167104001)

[CHƯƠNG 2 THIẾT KẾ KIẾN TRÚC 4](#_Toc167104002)

[**2.1.** **Phân tích yêu cầu** 4](#_Toc167104005)

[**2.2.** **Kiến trúc phần mềm** 5](#_Toc167104006)

[**2.3.** **Mô hình kiến trúc Microservice** 7](#_Toc167104007)

[CHƯƠNG 3 PHÂN TÍCH VÀ THIẾT KẾ 8](#_Toc167104008)

[**3.1.** **Phân tích yêu cầu bằng UML** 8](#_Toc167104012)

[**3.1.1.** **Usecase tổng quát** 8](#_Toc167104013)

[**3.1.2.** **Danh sách tác nhân và mô tả** 8](#_Toc167104014)

[**3.1.3.** **Danh sách tình huống hoạt động** 9](#_Toc167104015)

[**3.2.** **Class diagram** 22](#_Toc167104016)

[**3.3.** **Deployment diagram** 27](#_Toc167104017)

[CHƯƠNG 4 KẾT LUẬN 28](#_Toc167104018)

[**4.1.** **Kết quả đạt được** 28](#_Toc167104019)

[**4.2.** **Hạn chế của đồ án** 28](#_Toc167104020)

[**4.3.** **Hướng phát triển** 28](#_Toc167104021)

[TÀI LIỆU THAM KHẢO 29](#_Toc167104022)

# CHƯƠNG 1 MÔ TẢ YÊU CẦU

Một trường đại học cần quản lý hệ thống đăng ký học phần của sinh viên theo theo cơ chế tín chỉ. Các sinh viên sẽ được biên chế ở các khoa theo mỗi ngành học của mình đăng ký.

Trong mỗi học kỳ, nhà trường sẽ chọn một số môn học của từng ngành học cho sinh viên đăng ký học. Sinh viên đăng ký theo học vào một lớp nhất định (lớp tín chỉ) với số sinh viên được đăng ký tối đa được qui định cho từng lớp. Môn học được mở có thể là môn học tự chọn. Mỗi môn học có một hoặc nhiều môn học tiên quyết. Sinh viên sẽ không được đăng ký môn học mà có môn tiên quyết mình chưa học. Mỗi học kỳ sinh viên được đăng ký tối đa 30 tín chỉ. Nếu quá số này, hệ thống sẽ không cho đăng ký. Để phòng ngừa trường hợp sinh viên đăng ký xong rồi hủy bỏ, nhà trường yêu cầu sinh viên phải xác nhận trước khi đăng ký.

Vào ngày mở đăng ký, sinh viên sẽ đăng nhập vào hệ thống và sẽ nhìn thấy danh sách các môn học mà mình có khả năng đăng ký. Sinh viên chọn các môn học và tiến hành đăng ký. Trường hợp các lớp đã đầy, sinh viên sẽ được đưa vào một danh sách dự bị để nhà trường cân nhắc có mở thêm lớp hay không. Nếu không mở thêm lớp, sinh viên sẽ bị hủy đăng ký môn đó.

Sau khi đăng ký thành công, một email thông báo sẽ được gửi cho sinh viên xác nhận việc đăng ký và nhận quyết định đóng học phí.

Ngoài ra, hệ thống còn có khả năng cho phép sinh viên xem thông tin học tập của mình (số tín chỉ đã đạt, số môn đã học, điểm môn học, điểm trung bình tích lũy…), thời khóa biểu theo tuần và các tiện ích khác.

Sau khi số tín chỉ đã đạt theo từng ngành, sinh viên có quyền đăng ký xét tốt nghiệp. Nếu mọi tiêu chuẩn đều thỏa mãn, sinh viên sẽ được cấp bằng tốt nghiệp và sẽ được đưa vào danh sách các cựu sinh viên. Thông tin về bằng cấp sẽ được công khai trên trang web của nhà trường. Thông tin của cựu sinh viên sẽ được lưu giữ để theo dõi quá trình làm việc (nếu sinh viên đồng ý), làm các cuộc survey, cũng như nhiều hoạt động khác.

# CHƯƠNG 2 THIẾT KẾ KIẾN TRÚC



## **Phân tích yêu cầu**

**Tổng Quan Yêu Cầu**

Đại học cần quản lý hệ thống đăng ký học phần theo tín chỉ với các yêu cầu sau:

* **Quản lý Đăng ký Học phần:** Sinh viên đăng ký học phần theo tín chỉ và được phân theo ngành học, khoa.
* **Giới hạn Đăng ký Lớp học:** Mỗi lớp học có số lượng sinh viên đăng ký tối đa.
* **Quản lý Môn học Tiên quyết:** Sinh viên phải hoàn thành môn tiên quyết trước khi đăng ký môn học kế tiếp.
* **Xác nhận Đăng ký:** Yêu cầu sinh viên xác nhận trước khi hoàn tất đăng ký.
* **Thông báo qua Email:** Gửi email xác nhận sau khi sinh viên đăng ký thành công.
* **Cung cấp Thông tin Học tập:** Sinh viên có thể tra cứu số tín chỉ đã đạt, các môn đã học, điểm số và điểm trung bình tích lũy.
* **Quản lý Thời khóa biểu:** Cung cấp thời khóa biểu theo tuần và các tiện ích liên quan.
* **Quản lý Cựu Sinh viên:** Theo dõi thông tin cựu sinh viên và lưu trữ thông tin về quá trình làm việc, khảo sát và các hoạt động khác.

**Yêu Cầu và Mục Tiêu Mở Rộng**

* **Quy Mô và Lưu lượng Truy cập**

Hệ thống cần chịu tải cao trong các giai đoạn đăng ký học phần và giảm tải vào các thời gian khác. Điều quan trọng là hệ thống phải duy trì sự đồng bộ và ổn định ngay cả khi có lượng yêu cầu lớn trong các giai đoạn này**.**

* **Cải thiện Hiệu quả và Trải nghiệm Sinh viên**

Hệ thống được kỳ vọng giúp sinh viên đăng ký môn học phù hợp mà không mất quá nhiều thời gian, từ đó nâng cao chất lượng và uy tín của trường.

* **Quản lý và Bảo trì Hệ thống**

Hệ thống có thể cần tạm ngưng để cập nhật và kiểm soát tính nhất quán của dữ liệu, nhưng cần cố gắng giảm thiểu thời gian gián đoạn (downtime).

* **Thanh toán Trực tuyến**

Nhà trường dự kiến chuyển đổi số trong việc quản lý học phí và thanh toán trực tuyến để thuận tiện cho sinh viên.

**Kết luận**

Việc xây dựng hệ thống quản lý đăng ký học phần không chỉ giải quyết các nhu cầu cơ bản của việc quản lý học phần mà còn hướng tới tương lai với nhiều tính năng mở rộng và hiện đại. Hệ thống này sẽ góp phần quan trọng trong việc nâng cao chất lượng dịch vụ giáo dục, uy tín của nhà trường và đáp ứng nhu cầu của sinh viên một cách tốt nhất.

## **Kiến trúc phần mềm**

Để giải quyết vấn đề liên quan đế biến động về lượng truy cập theo kỳ đăng ký học phần, nhà trường cần phải mở rộng, nâng cấp sức mạnh của server. Nhưng việc mở rộng chỉ cần thiết khi đến kỳ đăng ký học phần, nên không thể nâng cấp cho một hệ thống, và kỳ đăng ký học phần chỉ diễn ra vài lần trong năm. bên cạnh đó, khi server chịu tải lớn trong kỳ đăng ký học phần, thường là vào gần kỳ thi, ảnh hướng đến việc các sinh viên khác xem lịch thi, vì dùng chung một server. Cách khác là chia ra các server nhỏ đảm nhận các trách nhiệm khác nhau, và lựa chọn nâng cấp server cụ thể khi cần. Vì vậy nhà trường có thể lựa chon kiến trúc Microservice.

**Điểm mạnh của kiến trúc** **Microservice**

* **Đáp ứng nhu cầu biến động lớn của lưu lượng truy cập**

Microservice là kiến trúc tối ưu khi phải đối mặt với sự biến động lớn nhưng có thể dự đoán trước được về lưu lượng truy cập, đặc biệt trong các kỳ đăng ký học phần. Với khả năng mở rộng linh hoạt, hệ thống có thể tăng số lượng server để xử lý lượng truy cập tăng cao và giảm số lượng server khi không cần thiết, tiết kiệm tài nguyên.

* **Tối ưu hóa hiệu suất và tài nguyên**

Khác với kiến trúc monolithic, khi cần mở rộng hệ thống phải triển khai lại toàn bộ, kiến trúc microservice chỉ yêu cầu triển khai từng dịch vụ cụ thể cần mở rộng. Điều này giúp tránh lãng phí tài nguyên và tăng hiệu suất hệ thống.

* **Giảm thiểu thời gian gián đoạn (downtime)**

Downtime là yếu tố quan trọng khi đánh giá độ tin cậy của hệ thống. Với microservice, các dịch vụ hoạt động độc lập, khi cần cập nhật chỉ cần dừng các dịch vụ liên quan thay vì toàn bộ hệ thống. Điều này giúp các dịch vụ còn lại vẫn khả dụng, đảm bảo hệ thống hoạt động liên tục.

* **Tính năng bảo mật và xử lý lỗi tốt hơn**

Microservice giúp tách biệt các dịch vụ thành các thành phần nhỏ, dễ quản lý và bảo trì. Khi một dịch vụ gặp sự cố, chỉ cần kiểm tra và khắc phục dịch vụ đó mà không ảnh hưởng đến toàn bộ hệ thống. Điều này cũng giúp tối ưu hóa bảo mật cho các dịch vụ chứa dữ liệu quan trọng như thông tin sinh viên và tài chính, trong khi các dịch vụ khác có thể giảm mức độ bảo mật để tăng hiệu suất.

* **Linh hoạt trong việc phát triển và triển khai tính năng mới**

Kiến trúc microservice cho phép phát triển các tính năng mới một cách dễ dàng và nhanh chóng. Các tính năng mới có thể được phát triển thành các dịch vụ độc lập và triển khai mà không ảnh hưởng đến hệ thống hiện tại.

* **Hỗ trợ sử dụng nhiều ngôn ngữ lập trình và công nghệ**

Với microservice, mỗi dịch vụ có thể được phát triển bằng ngôn ngữ lập trình và công nghệ khác nhau, tạo điều kiện thuận lợi cho việc áp dụng các xu hướng công nghệ mới.

**Điểm yếu của kiến trúc Microservice**

Quản lý phức tạp: Cần quản lý, theo dõi và bảo trì từng dịch vụ độc lập, tăng khối lượng công việc quản lý.

Yêu cầu về chi phí: Đòi hỏi cơ sở hạ tầng và tài nguyên vận hành phức tạp hơn, dẫn đến chi phí vận hành và quản lý cao hơn.

Khó khăn trong kiểm thử và gỡ lỗi: Khó khăn trong việc xử lý lỗi là một đánh đổi của mô hình này.

**Kết luận**

Kiến trúc microservice mang lại nhiều lợi ích vượt trội trong việc đáp ứng các yêu cầu về tính năng, hiệu suất và bảo mật cho hệ thống đăng ký học phần của nhà trường. Mặc dù có một số điểm yếu cần lưu ý, lợi ích mà kiến trúc này mang lại xứng đáng để đầu tư và triển khai.

## **Mô hình kiến trúc Microservice**

A diagram of a computer network

Description automatically generated with medium confidence

# CHƯƠNG 3 PHÂN TÍCH VÀ THIẾT KẾ



## **Phân tích yêu cầu bằng UML**

### **Usecase tổng quát**

A diagram of a diagram

Description automatically generated

### **Danh sách tác nhân và mô tả**

|  |  |
| --- | --- |
| **Tác nhân** | **Mô tả tác nhân** |
| Sinh Viên | Người sử dụng hệ thống để đăng nhập, đăng xuất và đăng ký môn học và quản lý thông tin học tập của mình, cũng như thực hiện các tác vụ thanh toán học phí, xem chương trình khung, xem công nợ, xem lịch theo tuần và xem phiếu thu. |
| Hệ thống thanh toán VNPAY | Hệ thống thanh toán bên thứ 3 do VNPAY cung cấp để thanh toán học phí trực tuyến |

### **Danh sách tình huống hoạt động**

|  |  |
| --- | --- |
| **ID** | **Tên Use case** |
| UC01 | Đăng nhập sinh viên |
| UC02 | Đăng ký lớp học phần |
| UC03 | Chuyển lớp học phần |
| UC04 | Thanh toán học phí |
| UC05 | Xem lịch theo tuần |

**UC01: Đăng nhập**

1. Đặc tả use case.

|  |  |
| --- | --- |
| **Use case: Đăng nhập** | |
| Mục đích: | Giúp sinh viên đăng nhập vào hệ thống để sử dụng dịch vụ. |
| Mô tả: | Sinh viên đăng nhập vào tài khoản của mình để sử dụng các dịch vụ được cung cấp từ bên trường. |
| Tác nhân: | Sinh viên |
| Điều kiện trước: | Sinh viên bắt buộc phải có tài khoản và mật khẩu đăng nhập. |
| Điều kiện sau: | Tài khoản và mật khẩu phải đúng với dữ liệu lưu trên hệ thống. |
| ***Luồng sự kiện chính (Basic flows):*** | |
| ***Sinh viên*** | ***Hệ thống*** |
| 1. Sinh viên chọn chức năng đăng nhập |  |
| 2. Sinh viên nhập thông tin đăng nhập bao gồm mã số sinh viên và mật khẩu và nhấn nút đăng nhập |  |
|  | 3. Hệ thống kiểm tra thông tin đăng nhập |
|  | 4. Hệ thống thông báo đăng nhập thành công và điều hướng đến trang sinh viên |
| ***Luồng sự kiện phụ (Alternative Flows):*** | |
|  | 3.1. Hệ thống thông báo mã số sinh viên hoặc mật khẩu không đúng. |
|  | 3.2. Hệ thống thông báo lỗi và yêu cầu sinh viên nhập lại thông tin (Quay lại bước 2) |

1. Activity Diagram

A diagram of a diagram

Description automatically generated

UC02: Đăng ký lớp học phần:

1. Đặc tả usecase

|  |  |
| --- | --- |
| **Use case: Đăng ký lớp học phần** | |
| Mục đích: | Giúp sinh viên đăng ký lớp học phần cho học kỳ sắp tới |
| Mô tả: | Sinh viên chọn môn học sau đó chọn lớp học phù hợp và nhấn nút đăng ký. |
| Tác nhân: | Sinh viên |
| Điều kiện trước: | Sinh viên đã đăng nhập vào hệ thống. |
| Điều kiện sau: | Hệ thống lưu thông tin lớp học phần mà sinh viên đã đăng ký vào cơ sở dữ liệu |
| ***Luồng sự kiện chính (Basic flows):*** | |
| ***Sinh viên*** | ***Hệ thống*** |
| 1. Sinh viên chọn chức năng đăng ký học phần |  |
| 2. Sinh viên chọn môn học |  |
| 3. Sinh viên chọn lớp học phần có trạng thái là đang chờ sinh viên đăng ký thuộc môn học đã chọn |  |
| 4. Sinh viên nhấn nút đăng ký |  |
|  | 5. Hệ thống kiểm tra thông tin lớp học phần mà sinh viên đăng ký |
|  | 6. Hệ thống thông báo đăng ký thành công và lưu thông tin lớp học phần xuống cơ sở dữ liệu |
| ***Luồng sự kiện phụ (Alternative Flows):*** | |
|  | 3.1. Hệ thống thông báo lớp học đã đầy. |
| 3.2 Sinh viên quay lại bước |  |

1. Activity Diagram

A diagram of a flowchart

Description automatically generated

UC03: Chuyển lớp học phần

1. Đặc tả usecase

|  |  |
| --- | --- |
| **Use case: Chuyển lớp học phần** | |
| Mục đích: | Giúp sinh viên chuyển lớp học phần của một môn học mà không cần phải hủy rồi đăng ký lại |
| Mô tả: | Sinh viên chọn môn học và chọn lớp học cần chuyển sau đó nhấn nút chuyển |
| Tác nhân: | Sinh viên |
| Điều kiện trước: | Sinh viên đã đăng nhập vào hệ thống. Và đã đăng ký lớp học của môn học cần chuyển |
| Điều kiện sau: | Hệ thống lưu thông tin vào cơ sở dữ liệu |
| ***Luồng sự kiện chính (Basic flows):*** | |
| ***Sinh viên*** | ***Hệ thống*** |
| 1. Sinh viên chọn môn học |  |
| 2. Sinh viên chọn lớp học phần có trạng thái là đang chờ sinh viên đăng ký thuộc môn học đã chọn |  |
| 3. Sinh viên nhấn nút đăng ký |  |
|  | 4. Hệ thống kiểm tra thông tin lớp học phần mà sinh viên đăng ký |
|  | 5. Hệ thống thông báo chuyển lớp thành công và cập nhật lại thông tin lớp học phần xuống cơ sở dữ liệu |
| ***Luồng sự kiện phụ (Alternative Flows):*** | |
|  | 3.1. Hệ thống thông báo lớp học đã đầy. |
| 3.2 Sinh viên quay lại bước |  |

1. Activity Diagram

A diagram of a flowchart

Description automatically generated

UC04: Thanh toán học phí:

1. Đặc tả usecase

|  |  |
| --- | --- |
| **Use case: Chuyển lớp học phần** | |
| Mục đích: | Giúp sinh viên thanh toán học phí các môn học đã đăng ký |
| Mô tả: | Sinh viên chọn môn học và nhấn thanh toán sau đó hệ thống sẽ kiểm tra và chuyển sang hệ thống thanh toán trực tuyến ở VNPAY sau khi thanh toán thành công hệ thống thanh toán sẽ chuyển hướng về lại trang xác nhận để trừ công nợ. |
| Tác nhân: | Sinh viên |
| Điều kiện trước: | Sinh viên đã đăng ký ít nhất 1 môn học và chọn ít nhất 1 môn để thanh toán, thời gian thanh toán nằm trong khoản thời gian mà trường đưa ra. |
| Điều kiện sau: | Hệ thống lưu thông tin phiếu thu vào cơ sở dữ liệu |
| ***Luồng sự kiện chính (Basic flows):*** | |
| ***Sinh viên*** | ***Hệ thống*** |
| 1. Sinh viên chọn môn học |  |
| 2. Sinh viên nhấn nút thanh toán |  |
|  | 3. Hệ thống kiểm tra thông tin thanh toán |
|  | 4. Hệ thống điều hướng sang trang thanh toán của VNPAY |
| 5. Sinh viên thực hiện thao tác thanh toán trên VNPAY |  |
|  | 6. Hệ thống thanh toán VNPAY kiểm tra thông tin và điều hướng về trang xác nhận thanh toán của hệ thống gốc |
| 7. Sinh viên nhấn xác nhận |  |
|  | 8. Hệ thống kiểm tra thông tin thanh toán |
|  | 9. Hệ thống thông báo thanh toán thành công và lưu hóa đơn xuống cơ sở dữ liệu |
| ***Luồng sự kiện phụ (Alternative Flows):*** | |
|  | 8.1. Hệ thống thông báo giao dịch thất bại hoặc giao dịch không hợp lệ |
| 8.2 Sinh viên quay xác nhận và kết thúc |  |

1. Activity Diagram

A diagram of a flowchart

Description automatically generated

UC05: Xem lịch theo tuần

1. Đặc tả usecase

|  |  |
| --- | --- |
| **Use case: Chuyển lớp học phần** | |
| Mục đích: | Giúp sinh viên xem lịch học hay lịch thi hoặc cả hai |
| Mô tả: | Sinh viên chọn lịch học hay lịch thi hoặc chọn lịch theo tuần trên thanh sidebar |
| Tác nhân: | Sinh viên |
| Điều kiện trước: | Sinh viên đã đăng ký ít nhất 1 môn học. |
| Điều kiện sau: | Hệ thống hiển thị lịch học |
| ***Luồng sự kiện chính (Basic flows):*** | |
| ***Sinh viên*** | ***Hệ thống*** |
| 1. Sinh viên chọn chức năng xem lịch học |  |
| 2. Sinh viên chọn loại lịch (tất cả, lịch học, lịch thi |  |
|  | 3. Hệ thống hiển thị thông tin lịch theo tuần dựa trên loại lịch |
| 4. Sinh viên xem lịch theo tuần. |  |
| ***Luồng sự kiện phụ (Alternative Flows):*** | |

1. Activiy Diagram

A diagram of a diagram

Description automatically generated with medium confidence

### **Class diagram**

* + 1. Auth Service

A screenshot of a computer

Description automatically generated

* + 1. Course service

A screenshot of a computer

Description automatically generated

* + 1. Enroll service

A screenshot of a computer

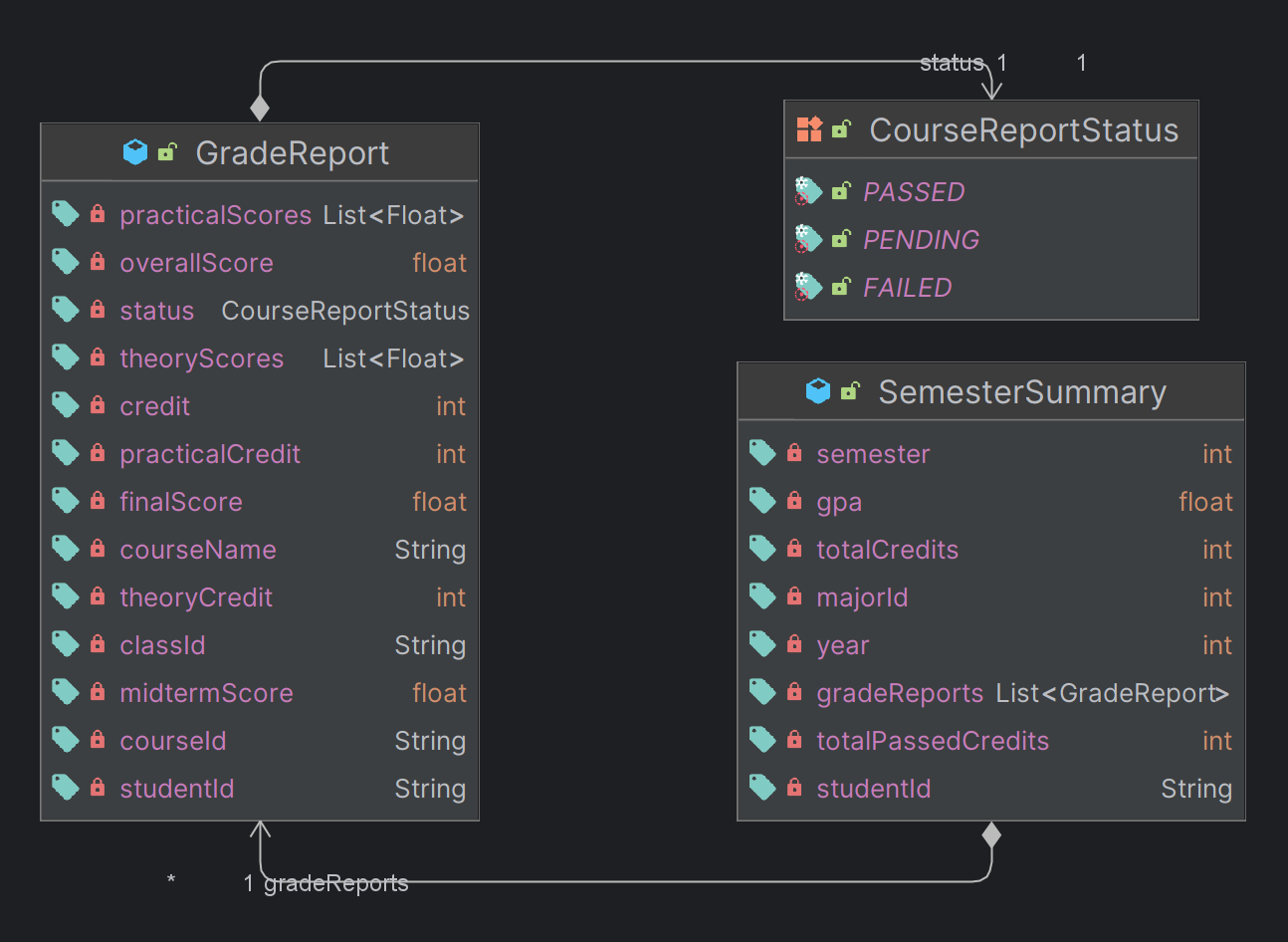
Description automatically generated

* + 1. Faculty service

A screenshot of a computer

Description automatically generated

* + 1. Grade Management service

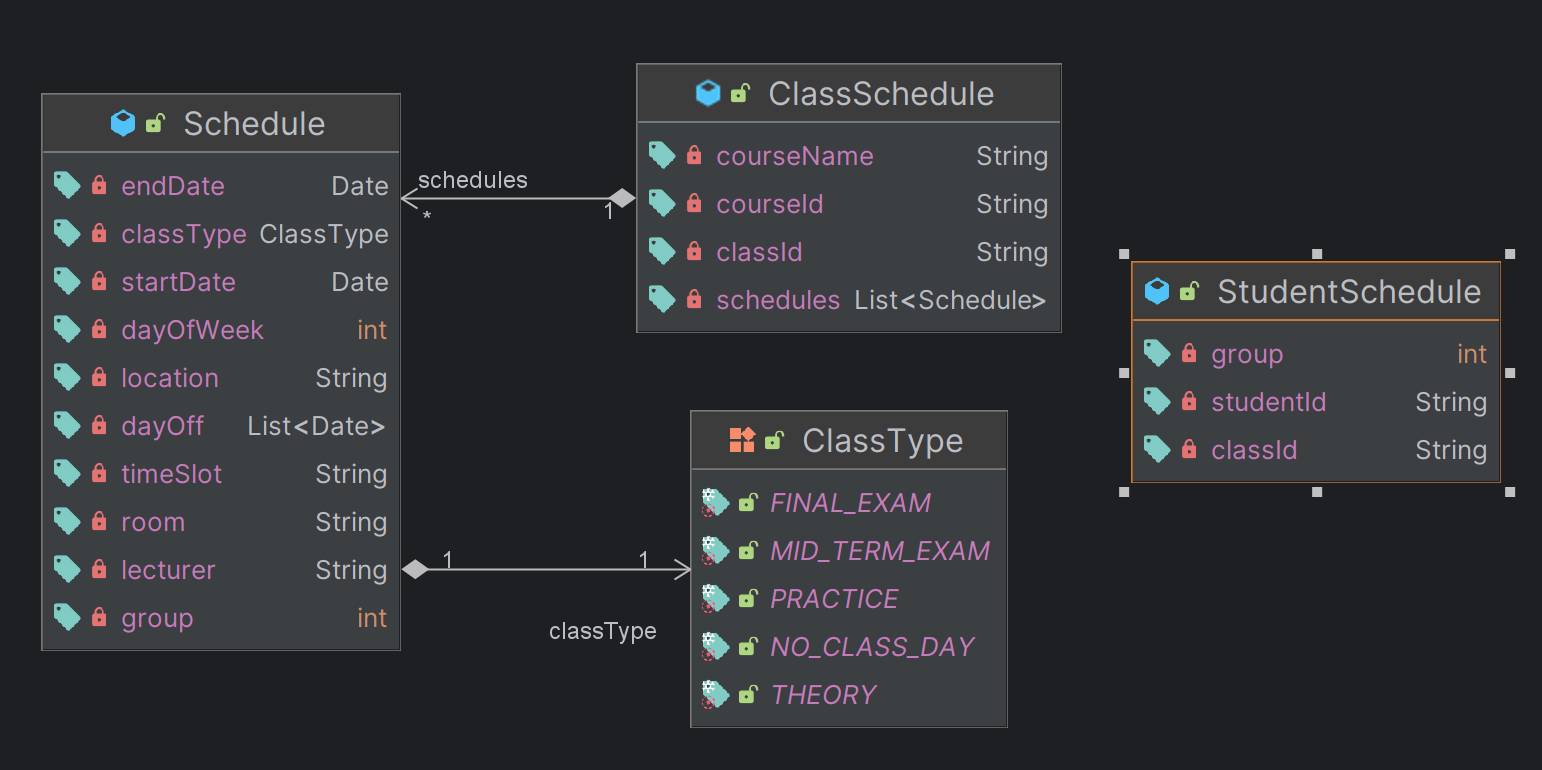


* + 1. Payment service

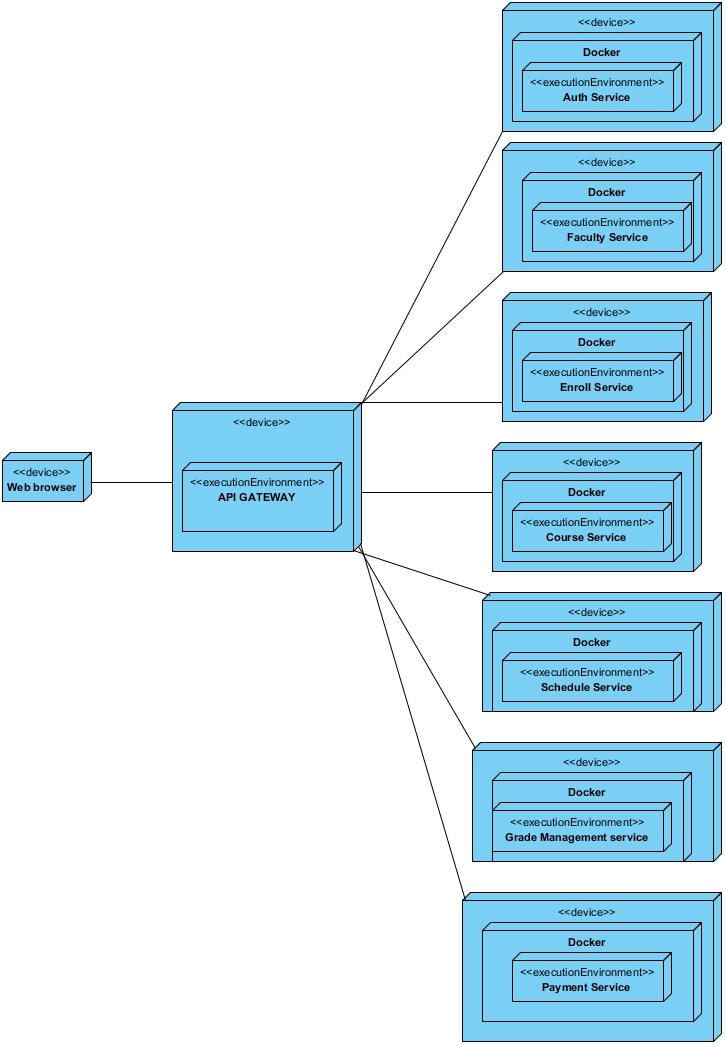
A screenshot of a computer

Description automatically generated

* + 1. Schedule service



### **Deployment diagram**



# CHƯƠNG 4 KẾT LUẬN

## **Kết quả đạt được**

Áp dúng các công nghệ: reactive programming, RabbitMQ, Redis, tích hợp vnpay, jwt token, docker.

RabbitMQ: Tăng khả năng chịu lỗi và bất động bộ khi phải thực hiện nhiều request từ client.

JWT: bảo mật bằng access token và refresh token, thời gian hết hạn access token 5 phút, refresh token 2 tiếng.

Redis: lưu cache các tiến trình tính toán phức tạp, và lưu các refresh token đã logout để tránh người dùng bị hack tấn công đánh mật refresh token.

Docker: dễ dàng deploy và tích hợp, giúp chạy trên các hệ điều hành khác nhau

## **Hạn chế của đồ án**

Chưa áp dùng đủ các công nghệ cần thiết cho microservice, ví dụ: circuit breaker, gRPC

Hệ thống chưa được deploy lên server thực, do thiếu thốn về kinh phí

Vẫn còn nhiều điểm chưa được tối ưu, chưa test performance của server với lượng lớn request

Các chức năng cho sinh viên đã tốt nghiệp và hệ thống cho giáo vụ khoa, giảng viên chưa thực hiện

## **Hướng phát triển**

Áp dụng các công nghệ mới cần thiết cho hệ thống: NGINX, gRPC, Resilience4j…

Hoàn thiện service cho sinh viên đã tốt nghiệp

Hoàn thiện hệ thống cho giáo vụ khoa và giảng viên

Thực hiện kiểm thử hệ thống và triển khai để đảm bảo server có thể chịu đựng lượng lớn request, đảm báo tính toàn vẹn của dữ liệu

# TÀI LIỆU THAM KHẢO

**Các tài liệu từ Internet**

1. Spring. Spring Document Guide [Online]. Từ <https://spring.io/quickstart>
2. Ant Design. Ant Design UI components Document [Online]. Từ <https://ant.design/components/overview/>
3. Ant Design. Ant Design UI components Document [Online]. Từ <https://ant.design/components/overview/>
4. W3Schools. W3Schools Online Web Tutorials. Từ <https://www.w3schools.com/>