**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM**



**KIẾN TRÚC VÀ THIẾT KẾ PHẦN MỀM**

*Hướng dẫn:* TS.Võ Văn Hải

*Sinh viên thực hiện:*

|  |  |  |  |
| --- | --- | --- | --- |
| Mssv | Họ và Tên | Nhiệm vụ | Mô tả công nghệ |
| 20049891 | Nguyễn Văn Sơn  [sonnees](https://github.com/sonnees) | Backend,  Ý tưởng word. | Spring Boot, JWT, Reactive Programming.  RESTful API, gRPC.  MongoDB Reactive, PostgreSQL.  Gateway, Eureka.  Docker, Mongo Atlas, Render. |
| 20047071 | Trần Thiện Đạt | Front\_End for Administrator UI | HTML, Bootstrap 5, Js.  Node.js, gRPC Client. |
| 20046761 | Tô Hiếu Đông | Front\_End for Student UI |

Nội dung

[CHƯƠNG 1 MÔ TẢ YÊU CẦU 2](#_Toc167000245)

[CHƯƠNG 2 XÂY DỰNG KIẾN TRÚC 3](#_Toc167000246)

[2.1 Phân Tích Yêu Cầu 3](#_Toc167000247)

[2.2 Phác thảo kiến trúc: Microservice 5](#_Toc167000248)

[CHƯƠNG 3 PHÂN TÍCH VÀ THIẾT KẾ 9](#_Toc167000249)

[3.1 Phân tích yêu cầu bằng UML 9](#_Toc167000250)

[3.1.1 Usecase tổng quát: 9](#_Toc167000251)

[3.1.2 Danh sách tác nhân và mô tả 10](#_Toc167000252)

[3.1.3 Danh sách các tình huống hoạt động 10](#_Toc167000253)

[3.2 Class diagram: 37](#_Toc167000254)

[3.3 Package diagram: 42](#_Toc167000255)

[3.4 Deployment diagram: 43](#_Toc167000256)

[CHƯƠNG 4 KẾT LUẬN 44](#_Toc167000257)

[4.1 Kết quả đạt được: 44](#_Toc167000258)

[4.2 Hạn chế của đồ án: 45](#_Toc167000259)

[4.3 Hướng phát triển: 46](#_Toc167000260)

[TÀI LIỆU THAM KHẢO 47](#_Toc167000261)

# 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 quy 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.

# XÂY DỰNG KIẾN TRÚC

## Phân Tích Yêu Cầu

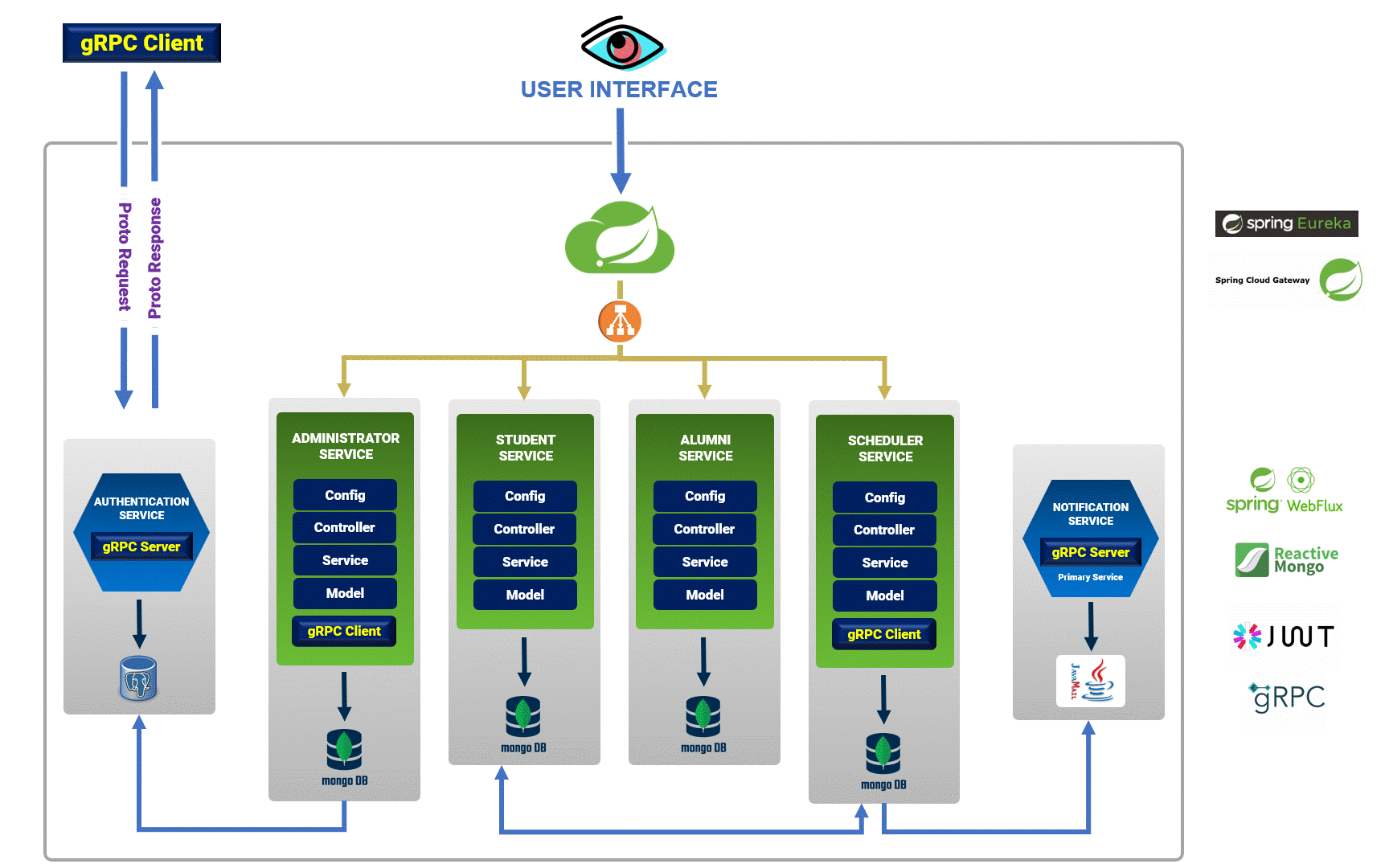
*Một số điểm đáng chú ý dựa trên mô tả yêu cầu:*

* Quản lý đăng ký học phần theo tín chỉ.
* Giới hạn số lượng sinh viên đăng ký vào mỗi lớp.
* Kiểm soát đăng ký dựa trên môn học tiên quyết.
* Yêu cầu xác nhận trước khi đăng ký.
* Gửi email thông báo đăng ký thành công.
* Cung cấp thông tin học tập và thời khóa biểu.
* Quản lý thông tin cựu sinh viên.

*Các mô tả, yêu cầu mở rộng thêm từ bên yêu cầu dịch vụ (Hiệu trưởng,…)*

* Hiểu được vấn đề giữa chất lượng với giá cả và thời gian. Sau khi họp với ban điều hành, nhà trường đã quyết định chọn ‘chất lượng’.
* Bình quân trường sẽ có 40.000 sinh viên. Mỗi năm thường sẽ có 3 đợt đăng ký học phần, một lần sẽ kéo dài tầm 20 ngày. Lượng truy cập vào hệ thống vì thế sẽ đạt đỉnh vào các móc thời gian này. Còn những khoảng thời gian khác, lượng traffic rất ít.
* Trong tương lai nhà trường sẽ mở rộng thêm nhiều chi nhánh, và các sinh viên ở mỗi chi nhánh sẽ đăng ký học phần sao cho thuận tiện với bản thân nhất, hệ thống cần đảm bảo rằng thông tin đăng ký và danh sách lớp học được đồng bộ giữa các cơ sở.
* Hệ thống được kỳ vọng sẽ giúp trường nâng cao thứ hạng, uy tín và chất lượng khi giúp sinh viên có thể đăng ký đúng môn, đúng giảng viên và chọn được giờ học ưng ý nhất mà lại không mất quá nhiều thời gian. Điều mà ít trường có thể làm được ở thời điểm hiện tại.
* Đôi khi hệ thống cần tạm ngưng để phục vụ cho việc cập nhật và kiểm soát tính tính nhất quán của dữ liệu (ví dụ như thêm lớp học online thay cho lớp học vật lý trong thời điểm bùng dịch). Cần cố gắng giảm ‘downtime’ hết mức có thể.
* Trong tương lại gần nhà trường cũng muốn chuyển đổi số cho việc quản lý học phí, thành toán online. Áp dụng trí tuệ nhân tạo để phân tích thông tin của sinh viên để có thể đề xuất sinh viên sẽ phù hợp học môn nào, sẽ phù hợp với cách dạy của giảng viên nào và đi đến đề xuất lớp học phần nào là phù hợp nhất cho sinh viên.
* Đây là cuộc đua về chuyển đổi số cho lĩnh vực giáo dục, dự án marketing được đầu tư lớn nhất từ trước đến nay. Vậy nên hãy áp dụng nhiều trend công nghệ để đạt được hiệu quả truyền thông tốn nhất.

## Phác thảo kiến trúc: Microservice



Chú thích:

* + ‘Administrator service’: là dịch vụ liên quan đến quản trị hệ thống, quản lý khoa, chuyên ngành, lớp học,…
  + ‘Student service’: là dịch vụ liên quan đến hoạt động của sinh viên.
  + ‘Alumni service’: là dịch vụ liên quan đến cựu sinh viên.
  + ‘Scheduler service’: là dịch vụ liên quan đến đăng ký học phần.
  + Module ‘Service’: Đại diện cho ‘utility’ và ‘repository’.
  + Module ‘Model’: Đại diện cho ‘entity’, ‘dto’, ‘enum’.
  1. Phân tích kiến trúc:

*Vì sao chọn kiến trúc:*

Việc nhà trường yêu cầu một hệ thống mạnh mẽ, chất lượng cao, vận hành ổn định ngay từ ban đầu mà không cần quan tâm đến chi phí và thời gian phát triển không quá gấp rút thì kiến trúc microservice là một sự lựa chọn hoàn hảo.

Lượng traffic biến động lớn nhưng có thể đoán trước được lại là đặc điểm cần cân nhắc sử dụng kiến trúc này. Vì microservice đặc biệt tối ưu khi áp dụng vào hệ thống cần tăng nhanh server nhưng lại có thể giảm chúng đi một cách đơn giản mà không mất nhiều chi phí. Nếu các kiến trúc monolithic khác khi cần mở rộng phải deploy toàn bộ hệ thống, rất lãng phí tài nguyên. Nhưng đối với kiến trúc microservice thì chỉ cần deploy nhỏ lẻ cho từng service nào cần tăng thôi.

Ngoài ra, nếu nhà trường sẽ mở thêm các chi nhánh trong tương lại thì micservice cũng xử lý rất tốt điều đó. Có thể triển khai hệ thống tại từng khu vực tăng hiệu suất truy vấn, trong đó dữ liệu sẽ được đồng bộ về máy chủ trung tâm để đảm bảo sự nhất quán dữ liệu. Việc phân thành khu vực cũng có thể hạn chế downtime cho toàn bộ hệ thống.

Downtime luôn là yếu tố then chốt nhất khi đánh giá độ tin cậy của một hệ thống. Đặc điểm của microservice là độc lập vậy nên khi hệ thống cần cập nhật thì chỉ cần ‘hạ’ các service có liên quan xuống thay vì cả hệ thống, các service còn lại vẫn khả dụng để người dùng sử dụng. Cũng với tính độc lập đó giúp hệ thống chụi lỗi tốt hơn, bảo mật hơn. Khi có sự cố ở một service nào đó thì chỉ cần kiểm tra service đó, dữ liệu có bị đánh cắp thì cũng nhỏ lẻ từng service. Vì vậy có thể tối ưu bảo mật tại các service, nghĩa là service nào chứa dữ liệu quan trọng liên quan đến sống còn của nhà trường thì cần siết chặt bảo mật như thông tin sinh viên, tài chính nhà trường,… những service còn lại thì có thể giảm bớt bảo mật để tăng hiệu suất và độ trải nghiệm người dùng.

Việc tương lai có thể có thêm các chứng năng khác điều đó là rất dể dàng nếu ta chọn kiến trúc microservice, vì đây là tính chất được đánh giá 5 sao của kiến trúc này. Với cơ chế độ lập, ta có thể phát triển các chức năng mới thành các service và deploy chúng lên mà không ảnh hưởng đến hệ thống đang chạy.

Vì các chức năng được phân chia thành các service độc lập để phát triển. Vậy nên mỗi service sẽ có thể viết bằng các ngôn ngữ khác nhau, có thể áp dụng nhiều công nghệ khác nhau. Việc ‘đú trend công nghệ’ trở nên dể dàng hơn bảo giờ hết.

*Điểm mạnh:*

* Kiến trúc microservice cho phép hệ thống dễ dàng mở rộng và scale để đáp ứng lượng truy cập đồng thời vào những thời điểm cao điểm trong quá trình đăng ký học phần mỗi năm mà không gặp phải các vấn đề liên quan đến hiệu suất.
* Phân tán các dịch vụ nhỏ lẻ trong kiến trúc microservice giúp giảm thiểu thời gian downtime để cập nhật và kiểm soát tính nhất quán của dữ liệu. Điều này đảm bảo rằng hệ thống hoạt động liên tục mà không gây ra sự gián đoạn đáng kể.
* Microservice giúp tách biệt các dịch vụ thành các thành phần nhỏ, dễ dàng tích hợp và triển khai trên các môi trường khác nhau. Điều này cũng tạo điều kiện thuận lợi cho việc sử dụng công nghệ và ngôn ngữ lập trình khác nhau cho từng dịch vụ.
* Xử lý lỗi, hạn chế tin tặc tấn công hệ thống.

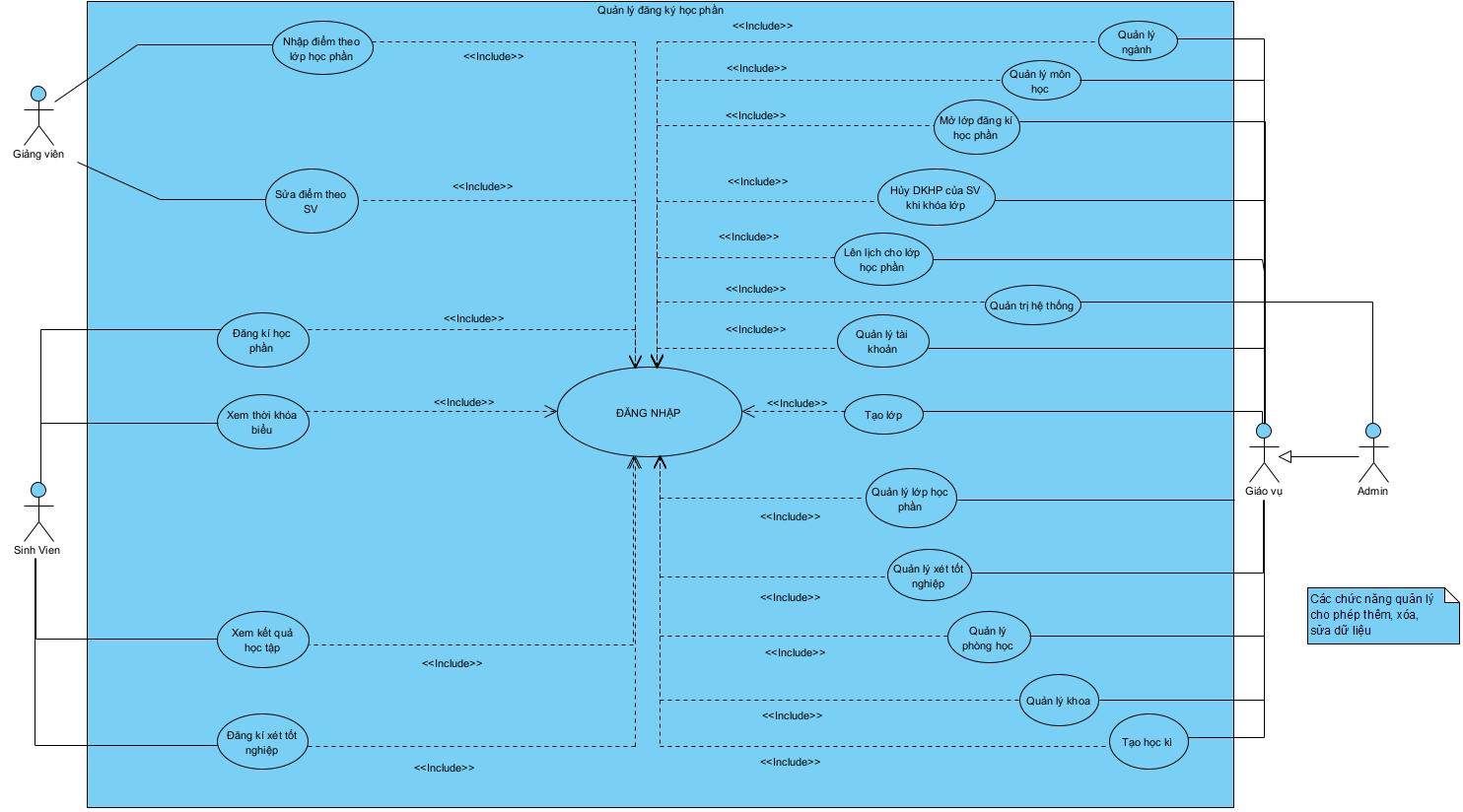
*Điểm yếu:*

* Có thể dẫn đến sự phức tạp trong việc quản lý toàn bộ hệ thống. Mỗi microservice cần được quản lý, theo dõi và bảo trì một cách độc lập, tạo ra một lượng công việc quản lý tăng lên đáng kể.
* Đòi hỏi cơ sở hạ tầng và tài nguyên vận hành phức tạp hơn, do cần triển khai và vận hành nhiều dịch vụ khác nhau. Điều này có thể dẫn đến tăng chi phí vận hành và quản lý.
* Vì microservice tách biệt nhau và chạy độc lập, việc kiểm thử và gỡ lỗi có thể trở nên phức tạp hơn so với kiến trúc monolithic. Điều này đặc biệt đúng khi cần phải kiểm thử tích hợp giữa các dịch vụ.

# PHÂN TÍCH VÀ THIẾT KẾ

## Phân tích yêu cầu bằng UML

### Usecase tổng quát:



### Danh sách tác nhân và mô tả

Danh sách tác nhân và mô tả xem ở Bảng 3-1.

|  |  |
| --- | --- |
| **Tác nhân** | **Mô tả tác nhân** |
| Sinh Viên | Người sử dụng hệ thống để đăng ký môn học và quản lý thông tin học tập của mình. |
| Admin | Người quản lý hệ thống và cung cấp hỗ trợ kỹ thuật. |
| Giáo Vụ | Người sử dụng hệ thống với các khả năng quản lý bên khoa |
| Giảng Viên | Người sử dụng hệ thống để đăng ký dạy học và nhập điểm cho lớp mình dạy |

Bảng 3-1 Danh sách tác nhân và mô tả

### Danh sách các tình huống hoạt động

|  |  |
| --- | --- |
| **ID** | **Tên Use case** |
| UC01 Dong | Đăng nhập sinh viên |
| UC02 Dong | Đăng ký lớp học phần |
| UC03 Dat | Đăng ký xét tốt nghiệp |
| UC04 Dat | Xét duyệt tốt nghiệp |
| UC05 Dat | Xem thông tin học tập |
| UC06 Dat | Mở danh sách lớp học phần |

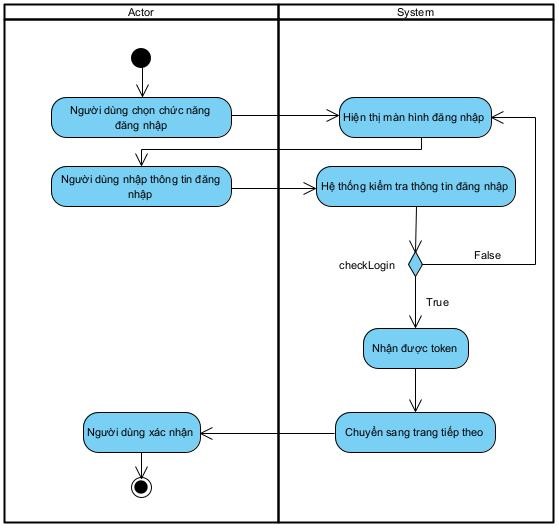
Bảng 3-2 Danh sách các tình huống trong hệ thống

**UC01: Đăng nhập (Sử dụng gRPC)**

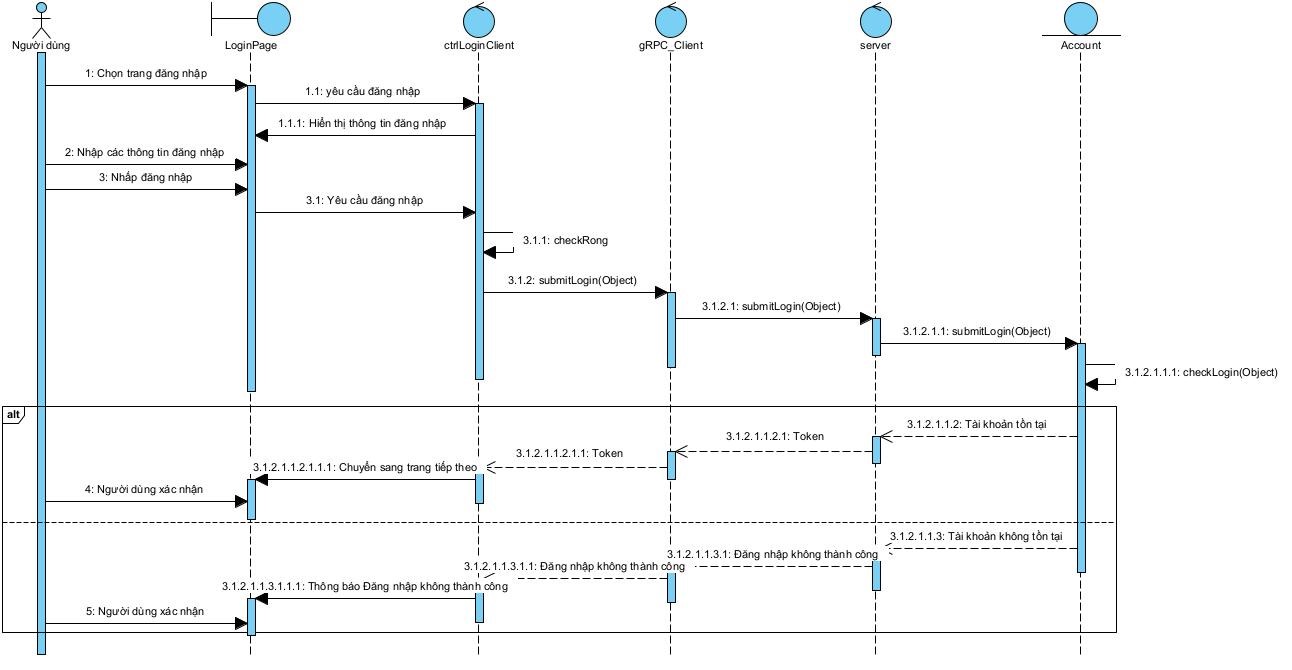
a. Đặc tả use case.

|  |  |
| --- | --- |
| **Use case: Đăng ký** | |
| Mục đích: | Giúp người dùng đăng nhập vào hệ thống để sử dụng dịch vụ. |
| Mô tả: | Đăng nhập vào tài khoản của mình để sử dụng các dịch vụ được cung cấp từ bên trường. |
| Tác nhân: | Sinh viên, Giảng viên, Giáo vụ, … |
| Điều kiện trước: | Người dùng 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):*** | |
| ***Actor*** | ***System*** |
| 1. Người dùng chọn chức năng đăng nhập |  |
| 2. Người dùng nhập thông tin đăng nhập (Mã số sinh viên và mật khẩu) |  |
|  | 3. Hệ thống kiểm tra thông tin đăng nhập |
|  | 4. Nếu thông tin hợp lệ, hệ thống cho phép người dùng đăng nhập thành công |
| 5. Chuyển vào trang cho người dùng |  |
| ***Luồng sự kiện phụ (Alternative Flows):*** | |
|  | 3.1. Hệ thống phát hiện mã số sinh viên hoặc mật khẩu không đúng. |
| 3.2. Hệ thống thông báo lỗi và yêu cầu người dùng nhập lại thông tin (Quay lại bước 2) |  |

b. Activity diagram.



c. Sequence diagram

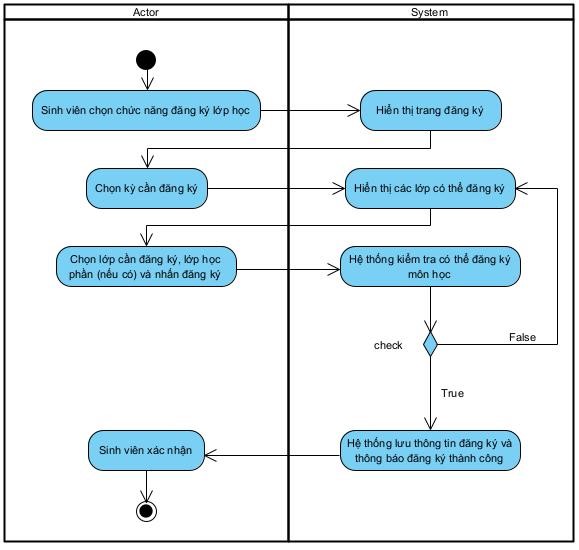


**UC02: Đăng ký môn học**

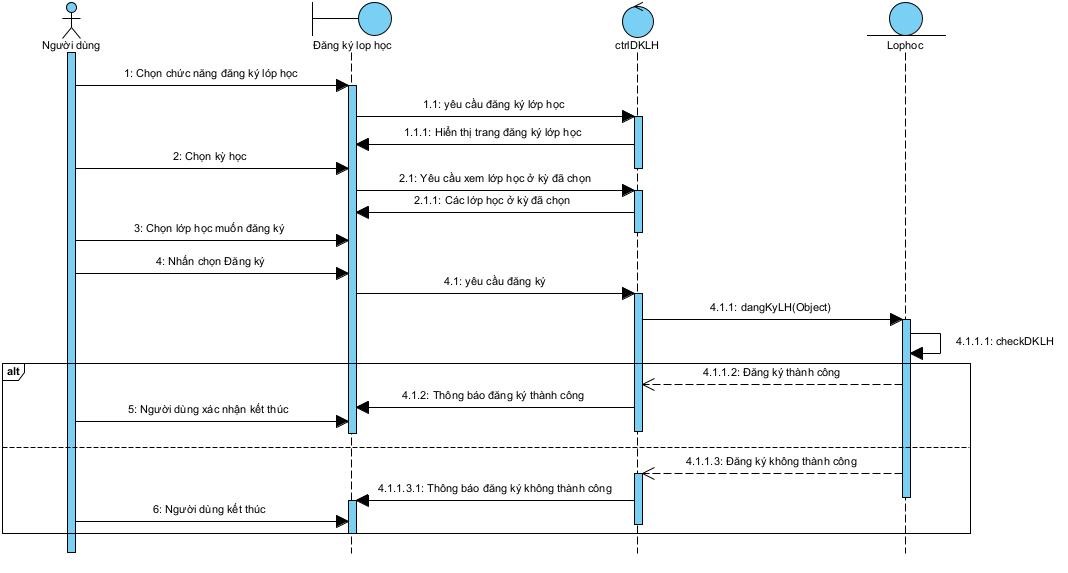
a. Đặc tả use case.

|  |  |
| --- | --- |
| **Use case: Đăng ký** | |
| Mục đích: | Giúp sinh viên đăng ký các môn học cho kỳ học mới. |
| Mô tả: | Giúp sinh viên có thể tìm và đăng ký các môn học phù hợp với bản thân ở học kỳ mới. |
| Tác nhân: | Sinh viên. |
| Điều kiện trước: | Sinh viên đã có tài khoản đăng nhập vào hệ thống và được phép đăng ký môn học trong kỳ học hiện tại. |
| Điều kiện sau: | Các môn học đã đăng ký hợp lệ và được lưu vào hệ thống thành công. |
| ***Luồng sự kiện chính (Basic flows):*** | |
| ***Tác nhân*** | ***Hệ thống*** |
| 1. Sinh viên chọn chức năng đăng ký môn học |  |
|  | 2. Hệ thống kiểm tra quyền đăng ký môn học của sinh viên. Nếu hợp lệ, hệ thống cung cấp danh sách các môn học có sẵn trong kỳ hiện tại. |
| 3. Sinh viên xem danh sách các môn học có sẵn |  |
| 4. Sinh viên chọn các môn học muốn đăng ký và nhóm thực hành (nếu là môn học thực hành) |  |
|  | 5. Hệ thống kiểm tra tính khả dụng của các môn học (số lượng sinh viên tối đa, điều kiện tiên quyết, thời khóa biểu không trùng) |
|  | 6. Nếu các môn học hợp lệ, hệ thống lưu thông tin đăng ký môn học của sinh viên |
| 7. Hệ thống hiển thị thông báo đăng ký thành công |  |
| ***Luồng sự kiện phụ (Alternative Flows): Môn học không khả dụng*** | |
|  | 5.1. Hệ thống phát hiện môn học đã đầy, không thỏa mãn điều kiện tiên quyết hoặc thời khóa biểu bị trùng |
|  | 5.2. Hệ thống thông báo lỗi và hiển thị lý do cụ thể cho sinh viên |
| 5.3. Sinh viên chọn lại các môn học khác hoặc điều chỉnh thời khóa biểu |  |
| ***Luồng sự kiện phụ (Alternative Flows): Sinh viên không đủ điều kiện đăng ký*** | |
|  | 2.1 Hệ thống phát hiện sinh viên không đủ điều kiện để đăng ký (chưa đóng học phí, chưa hoàn thành các môn học tiên quyết, đã đăng ký đủ 30 tín chỉ, quá hạn đăng ký môn học, …) |
| 2.2. Hệ thống thông báo lỗi và yêu cầu sinh viên hoàn tất các điều kiện cần thiết trước khi đăng ký lại |  |

b. Activity diagram.



c. Sequence diagram:

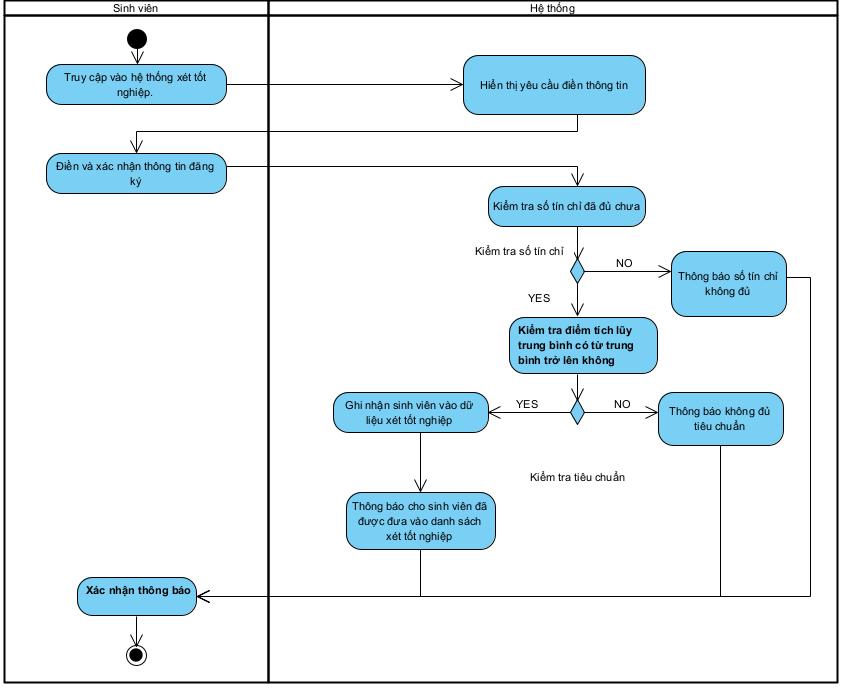


**UC03:** Đăng ký xét tốt nghiệp

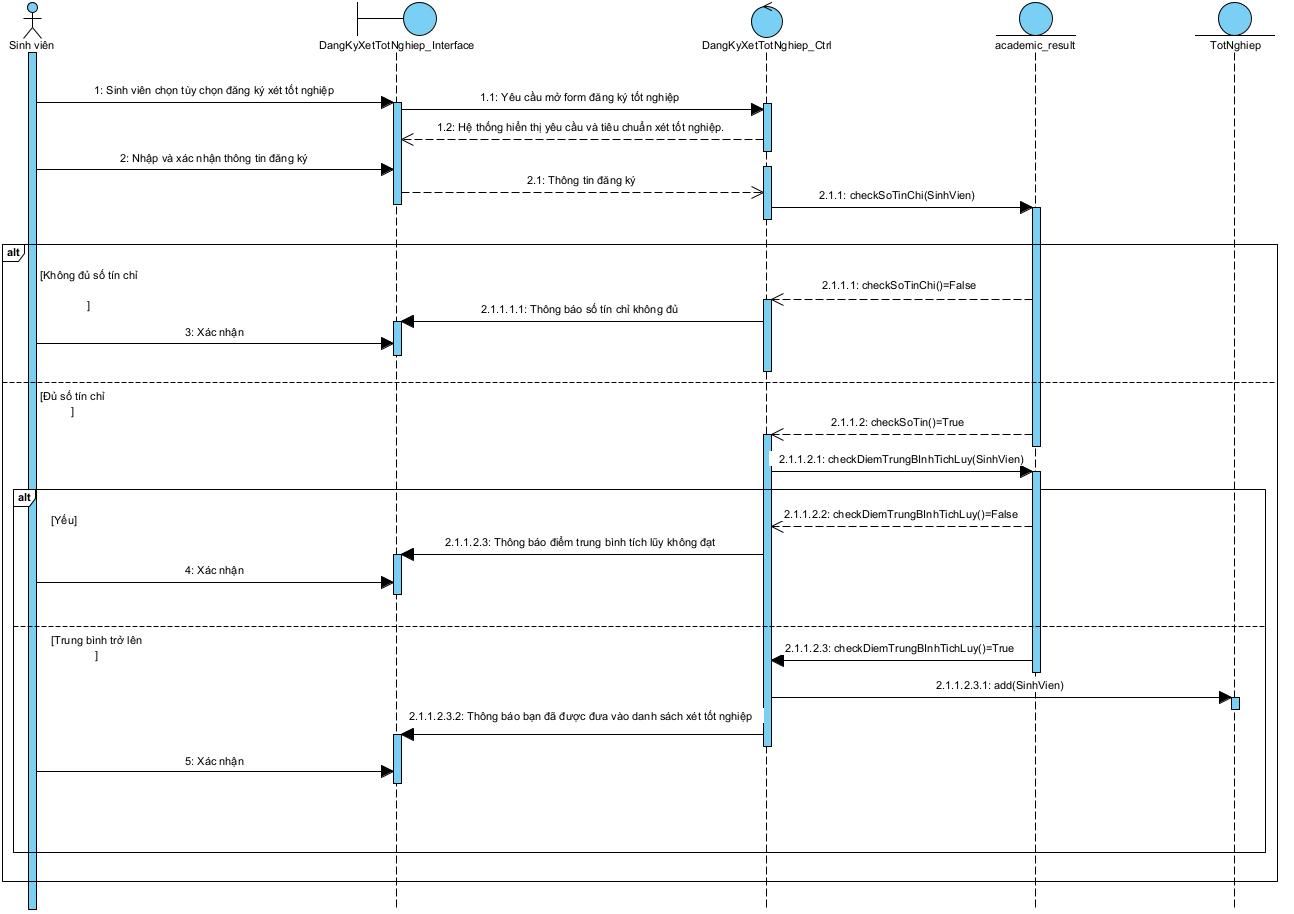
1. Đặc tả use case.

|  |  |
| --- | --- |
| **Use case:** Đăng ký xét tốt nghiệp | |
| Mục đích: | Giúp sinh viên có thể ký xét tốt nghiệp |
| Mô tả: | Sau khi số tín chỉ đã đạt theo từng ngành, sinh viên có quyền đăng ký xét tốt nghiệp. Nếu điểm trung bình tích lũy từ trung bình trở lên , sinh viên sẽ được đưa vào danh sách xét duyệt tốt nghiệp |
| Tác nhân: | Sinh viên |
| Điều kiện trước: | Có tài khoản đăng nhập là sinh viên.  Đạt đủ số tín chỉ trong ngành.  Điểm trung bình tích lũy từ trung bình trở lên |
| Điều kiện sau: | Đạt được đủ tiêu chuẩn xét tốt nghiệp |
| ***Luồng sự kiện chính (Basic flows):*** | |
| ***Actor*** | ***System*** |
| 1. Sinh viên truy cập vào hệ thống xét tốt nghiệp. |  |
|  | 2. Hệ thống hiển thị form yêu cầu thông tin |
| 3. Sinh viên điền và kiểm tra thông tin đăng ký |  |
|  | Hệ thống kiểm tra xem sinh viên đã đủ số tín chỉ chưa |
|  | 5. Hệ thống kiểm tra điểm tích lũy trung bình có từ trung bình trở lên không |
|  | 6. Hệ thống ghi nhận sinh viên vào dữ liệu xét tốt nghiệp |
|  | 7. Hệ thống thông báo cho sinh viên đã được đưa vào danh sách xét tốt nghiệp |
| 8. Sinh viên xác nhận |  |
| ***Luồng sự kiện phụ (Alternative Flows):*** | |
|  | 5.1 Hệ thống thông báo cho sinh viên về việc không đủ tiêu chuẩn. Sinh viên cần bổ sung các tiêu chuẩn còn thiếu trước khi đăng ký lại. |
| 5.2: Sinh viên xác nhận |  |

b. Activity diagram.



c. Sequence diagram:

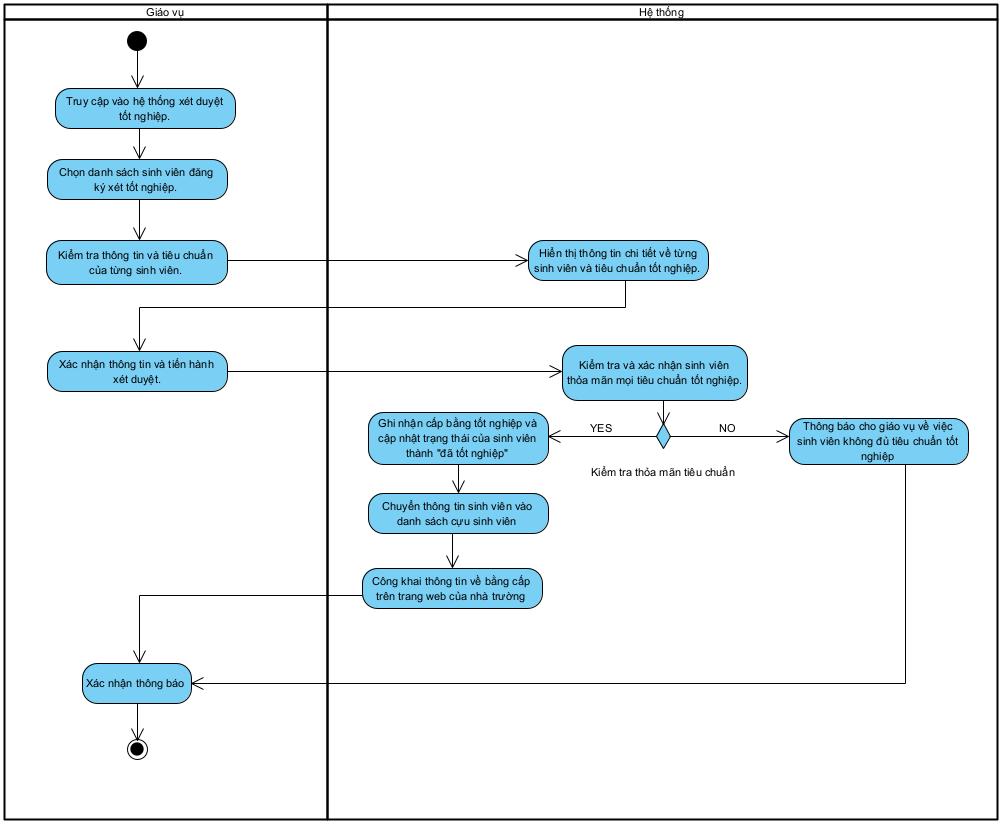


**UC04:** Xét duyệt tốt nghiệp

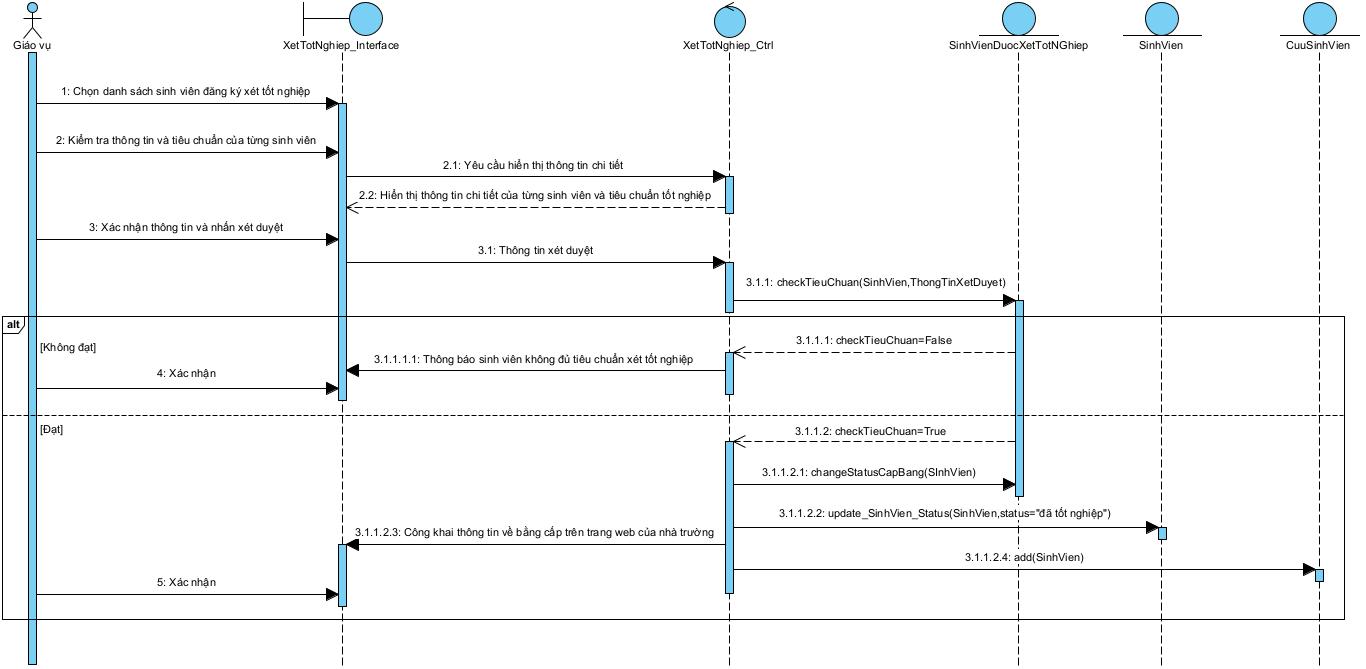
1. Đặc tả use case.

|  |  |
| --- | --- |
| **Use case:** Xét duyệt tốt nghiệp | |
| Mục đích: | Xét duyệt và hoàn tất quy trình tốt nghiệp cho sinh viên. |
| Mô tả: | Xét duyệt các sinh viên đã đăng ký xét duyệt tốt nghiệp, Cấp bằng tốt nghiệp và đưa vào danh sách các cựu sinh viên. Thông tin về bằng cấp sẽ được công khai trên trang web của nhà trường. |
| Tác nhân: | Giáo vụ |
| Điều kiện trước: | -Sinh viên đã đăng ký xét tốt nghiệp và hệ thống đã nhận được yêu cầu.  -Giáo vụ có quyền truy cập vào hệ thống xét duyệt tốt nghiệp. |
| Điều kiện sau: | -Sinh viên được xét duyệt và cấp bằng tốt nghiệp nếu thỏa mãn tất cả các tiêu chuẩn.  -Thông tin sinh viên tốt nghiệp được công khai trên trang web của nhà trường.  -Sinh viên được đưa vào danh sách các cựu sinh viên. |
| ***Luồng sự kiện chính (Basic flows):*** | |
| ***Actor*** | ***System*** |
| 1. Giáo vụ truy cập vào hệ thống xét duyệt tốt nghiệp. |  |
| 2. Giáo vụ chọn danh sách sinh viên đăng ký xét tốt nghiệp. |  |
| 3. Giáo vụ kiểm tra thông tin và tiêu chuẩn của từng sinh viên. |  |
|  | 4. Hệ thống hiển thị thông tin chi tiết về từng sinh viên và tiêu chuẩn tốt nghiệp. |
| 5. Giáo vụ xác nhận thông tin và tiến hành xét duyệt. |  |
|  | 6. Hệ thống kiểm tra và xác nhận sinh viên thỏa mãn mọi tiêu chuẩn tốt nghiệp. |
|  | 7. Nếu thỏa mãn, hệ thống ghi nhận cấp bằng tốt nghiệp và cập nhật trạng thái của sinh viên thành "đã tốt nghiệp". |
|  | 8. Hệ thống công khai thông tin về bằng cấp trên trang web của nhà trường. |
|  | 9. Hệ thống chuyển thông tin sinh viên vào danh sách cựu sinh viên. |
| ***Luồng sự kiện phụ (Alternative Flows):*** | |
|  | 6.1. Hệ thống thông báo cho giáo vụ về việc sinh viên không đủ tiêu chuẩn tốt nghiệp. |
| 6.2. Thông báo cho sinh viên về các tiêu chuẩn còn thiếu và yêu cầu bổ sung. |  |

1. Activity diagram.



c. Sequence diagram:

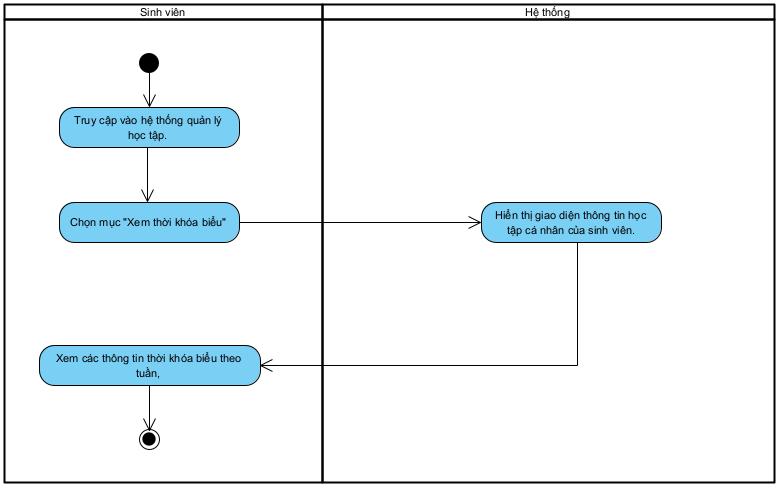


**UC05:** Xem thời khóa biểu

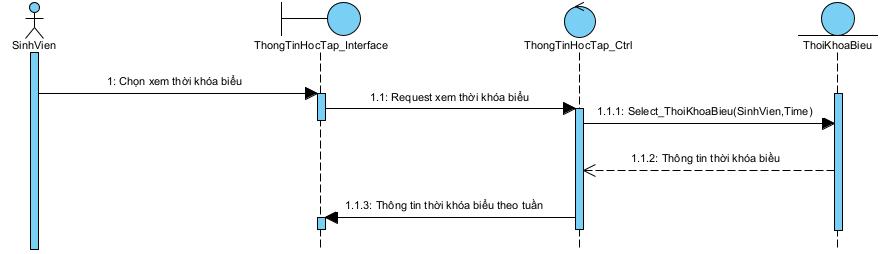
1. Đặc tả use case.

|  |  |
| --- | --- |
| **Use case:** Xem thông tin học tập cá nhân | |
| Mục đích: | Giúp sinh viên có thể xem thời khóa biểu theo tuần |
| Mô tả: | Sinh viên xem thời khóa biểu theo tuần, v.v. |
| Tác nhân: | Sinh viên |
| Điều kiện trước: | Sinh viên đã đăng nhập vào hệ thống quản lý học tập. Hệ thống có dữ liệu cập nhật về thời khóa biểu của sinh viên. |
| Điều kiện sau: | Sinh viên xem được thời khóa biểu. |
| ***Luồng sự kiện chính (Basic flows):*** | |
| ***Actor*** | ***System*** |
| 1. Sinh viên truy cập vào hệ thống quản lý học tập. |  |
| 2. Sinh viên chọn mục "Xem thời khóa biểu" |  |
|  | 4. Hệ thống hiển thị giao diện thông tin thời khóa biểu cá nhân của sinh viên. |
| 5. Sinh viên xem các thông tin thời khóa biểu theo tuần, v.v. |  |
| ***Luồng sự kiện phụ (Alternative Flows):*** | |
|  | 4.1.Nếu không có dữ liệu cập nhật về thời khóa biểu của sinh viên, hệ thống sẽ thông báo cho sinh viên biết và đề nghị liên hệ với phòng giáo vụ |

b. Activity diagram.



c. Sequence diagram:

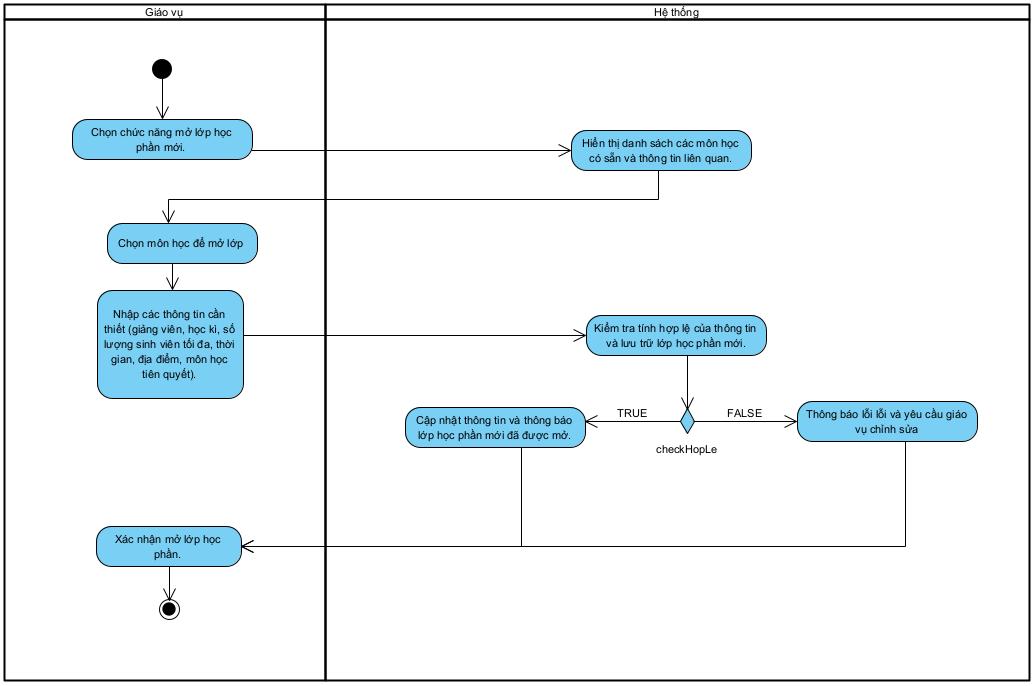


**UC06:** Mở lớp đăng ký học phần

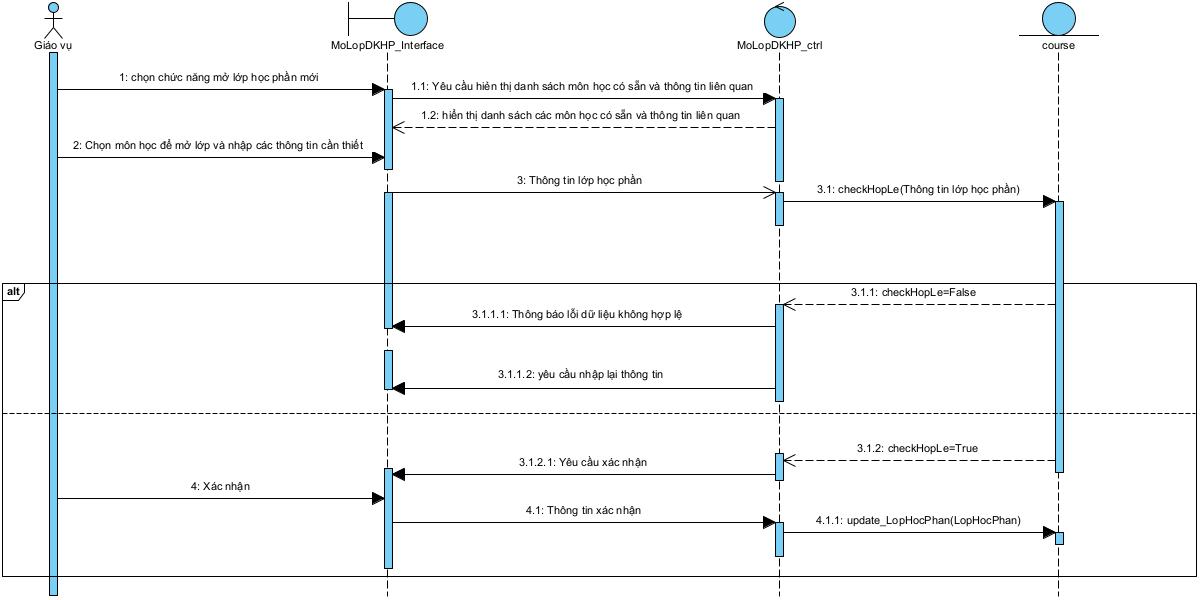
1. Đặc tả use case.

|  |  |
| --- | --- |
| **Use case:** Mở lớp đăng ký học phần | |
| Mục đích: | Giáo vụ mở các lớp học phần để sinh viên có thể đăng ký trong kỳ học mới. |
| Mô tả: | Giáo vụ chọn các môn học và tạo các lớp học phần để sinh viên có thể đăng ký trong kỳ học mới. Mỗi lớp học phần sẽ có các thông tin như môn học, giảng viên, số lượng sinh viên tối đa, thời gian, địa điểm, học kì và các môn học tiên quyết (nếu có). |
| Tác nhân: | Giáo vụ |
| Điều kiện trước: | Hệ thống đã có danh sách các môn học và thông tin giảng viên.  Giáo vụ đã đăng nhập vào hệ thống với quyền quản trị. |
| Điều kiện sau: | Các lớp học phần được mở và sinh viên có thể bắt đầu đăng ký. |
| ***Luồng sự kiện chính (Basic flows):*** | |
| ***Actor*** | ***System*** |
| 1. Giáo vụ chọn chức năng mở lớp học phần mới.. |  |
|  | 2. Hệ thống hiển thị danh sách các môn học có sẵn và thông tin liên quan. |
| 3. Giáo vụ chọn môn học để mở lớp và nhập các thông tin cần thiết (giảng viên, số lượng sinh viên tối đa, thời gian,học kì địa điểm, môn học tiên quyết). |  |
|  | 4. Hệ thống kiểm tra tính hợp lệ của thông tin và lưu trữ lớp học phần mới. |
| 5. Giáo vụ xác nhận mở lớp học phần. |  |
|  | 6. Hệ thống cập nhật thông tin và thông báo lớp học phần mới đã được mở. |
| ***Luồng sự kiện phụ (Alternative Flows):*** | |
|  | 4.1. Nếu thông tin nhập không hợp lệ (ví dụ: trùng thời gian, số lượng sinh viên vượt quá quy định), hệ thống sẽ hiển thị thông báo lỗi và yêu cầu giáo vụ chỉnh sửa. |

b. Activity diagram.



c. Sequence diagram:



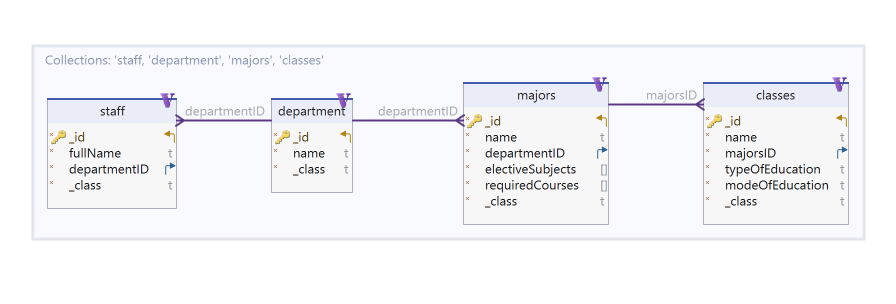
## Class diagram:

Tại sao chúng tôi lại chọn lưu trữ dữ liệu bằng mongodb, noSQL mà không phải SQL bởi vì một vài lý do sau đây:

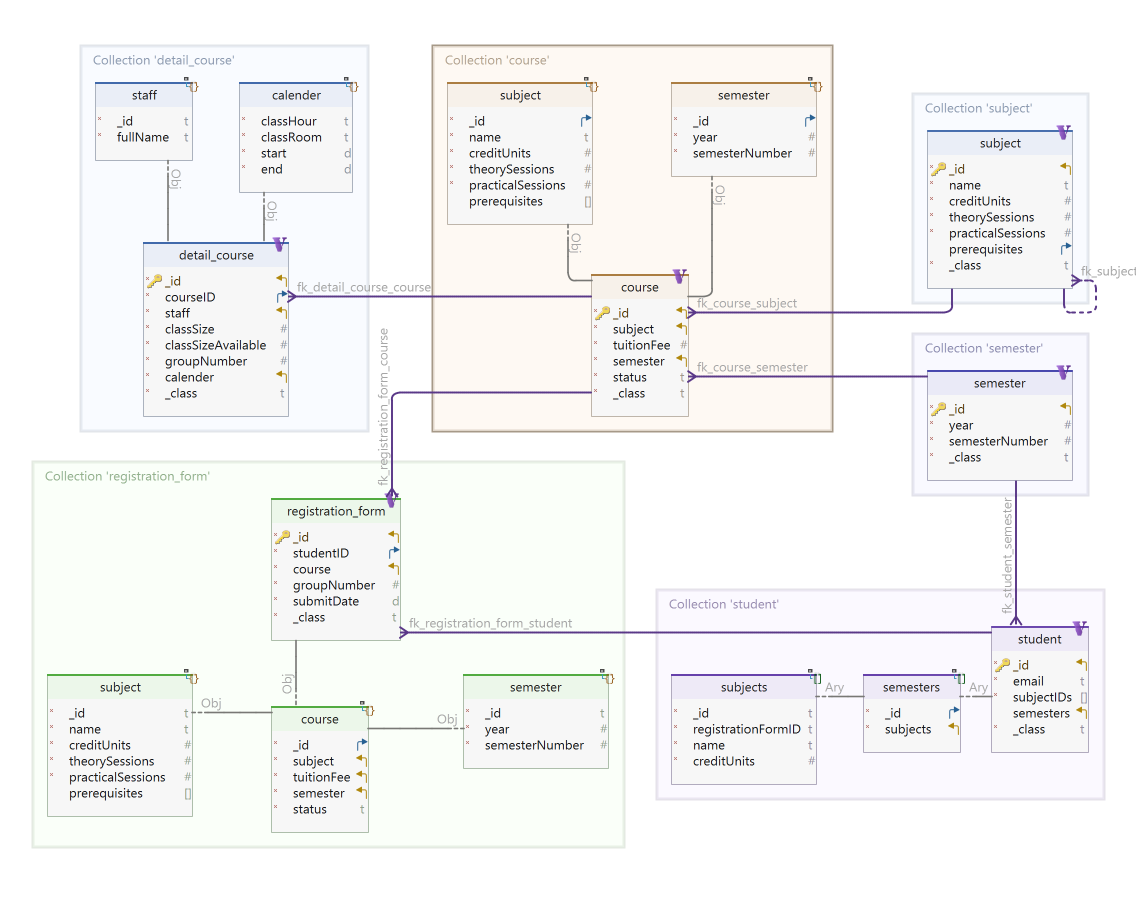
* Chúng tôi tự tin có thể xử lý được sự ràng buộc dữ liệu giữa các bảng có liên quan.
* Đưa hiệu suất lên hành đầu, ngoài việc hỗ trợ reactive thì mongodb có nhiều cơ chế vượt trội như index, change streams, các mẫu pattern … giúp đáp ứng nhanh các yêu cầu. Ngoài ra còn có thể sử dụng các công nghệ hiện đại như hadoop, spark để phân tích, thống kê dữ liệu lớn.
* Việc có thể mở rộng ngang cũng là 1 lợi thế vì nó hỗ trợ cơ chế sharding, cho phép phân tán dữ liệu trên nhiều server khác nhau. Điều này giúp hệ thống có thể mở rộng dễ dàng và hiệu quả khi khối lượng dữ liệu tăng lên.
* Lý do cuối và cũng là lý do mà nhóm tâm đắc nhất đó chính là mongodb hỗ trợ dữ liệu phi cấu trúc. Điều này rất quan trọng vì field trong các bảng nó là vô định, chúng có thể thay đổi bất cứ lúc nào, nếu chúng ta không có sự chuyển bị tốt thì rất có thể sẽ tốn cực kì nhiều chi phí để có thể chạy theo sự thay đổi.

Ví dụ như trước đại dịch, phòng học được hiểu theo là phòng học vật lý tại trường là A1.1, E13.7,... và nó là đã trở thành quy chuẩn nhưng dịch Covid-19 làm cho sinh viên không thể đến trường buộc phải chuyển qua học online. vậy khái niệm phòng học đã bị thay đổi, ngoài phòng vật lý ra ta còn có phòng online nữa. Như vậy nếu như ta xây dựng bằng SQL thì sẽ phải xử lý rất nhiều hoặc chữ cháy bằng các cách khác nhau nhưng nó sẽ tiềm ẩn nhiều lỗi không biết trước được. Vậy sử dụng mongodb được xem là 1 cách tối ưu nhất vì có thể áp dụng *Schema Versioning Pattern*để quản lý sự thay đổi này. (Ngoài phi cấu trúc ở vấn đề phòng học thì còn thêm thông tin như tên chi nhánh nếu trường mở rộng thêm cơ sở)

1. Database “AdministratorService”:



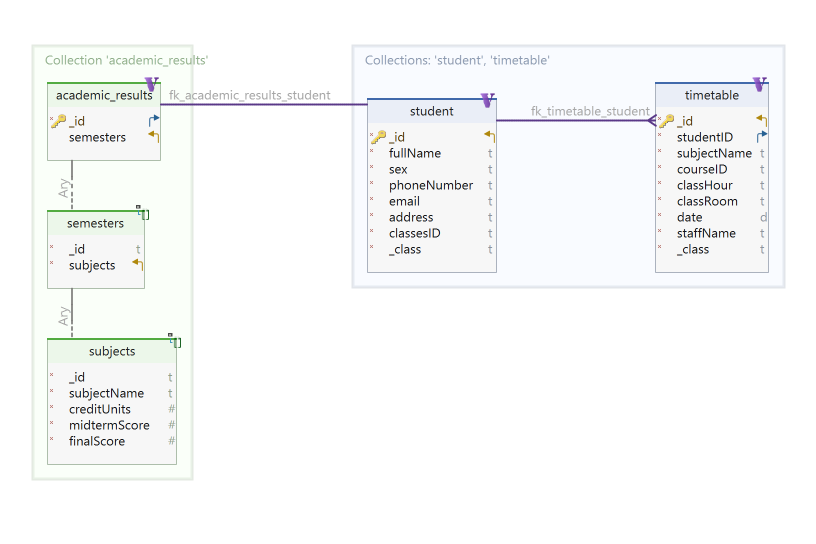
1. Database “SchedulerService”:



Áp dụng schema versioning pattern cho bảng detail\_course:

|  |  |  |
| --- | --- | --- |
| Version: 0 hoặc không có version nào | Version 1: Phòng học là object có thể là {id: string, pass: string} | Version 2: Có thêm thông tin chi nhánh |
|  |  |  |

1. Database “StudentService”:

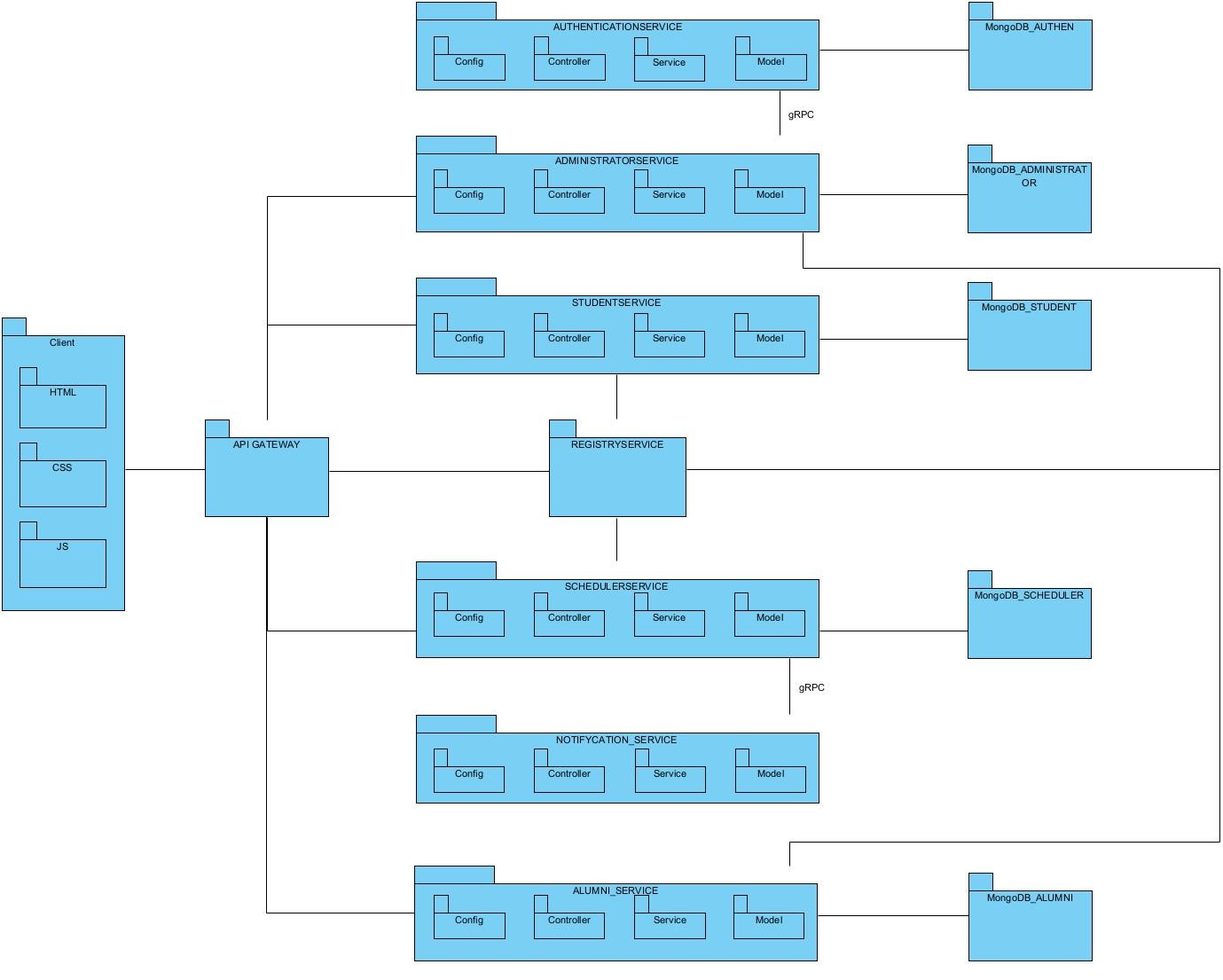


1. Database “Authentication Service”:

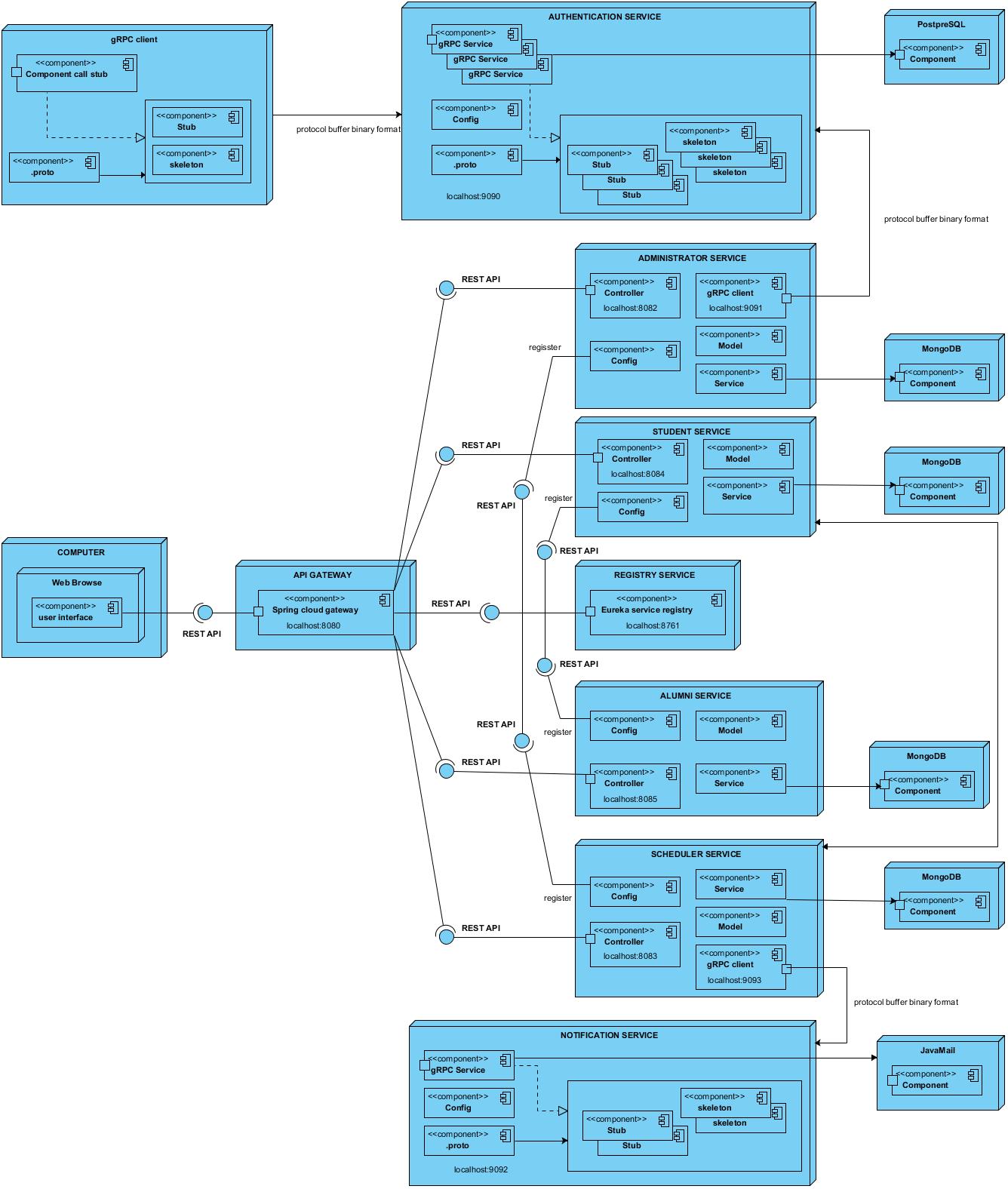
Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

## Package diagram:



## Deployment diagram:



# KẾT LUẬN

## Kết quả đạt được:

*Về phía backend:*

Có nhiều công nghệ trend: kiến trúc microservice, reactive programming, reactive mongo, gRPC.

Đã xây dựng được các service sau:

* + Administrator Service: giúp admin và giáo vụ quản lý giảng viên, khoa, chuyên ngành, lớp học.
  + Authentication Service: là dịch vụ liên qua đến quản lý thông tin đăng nhập, tạo token, tạo tài khoản.
  + Notification Service: sẽ hỗ trợ việc thông báo, hiện tại là thông báo qua mail.
  + Scheduler Service: giúp giảng viên có thể quản lý các lớp học phần và quá trình đăng ký học phần của sinh viên.
  + Student Service: giúp sinh viên xem thông tin cá nhân, lịch học, kết quả học tập.
  + Eureka làm registry và Spring Discovery để hỗ trợ cơ chế Load Balancing.

Dùng JWT để bảo mật hệ thống, trừ các api public ra thì còn lại để có thể sử dụng dịch vụ cần phải cung cấp token có hiệu lực. Ngoài ra chỉ một vài domain được ghi nhận mới có thể gọi dịch vụ (có chặn cors).

Có swagger ở mỗi service giúp cho việc quản lý các api trở nên đơn giản hơn cho backend và để theo dõi hơn cho frontend.

Đã deploy database lên [Render](https://render.com/)  và [Mongo Atlas](https://www.mongodb.com/atlas). Tạo Dockerfile và Docker compose để có thể chạy được chương trình ở mọi hệ điều hành và các phiên bản máy tính khác nhau.

*Về phía front end:*

Đã xây dựng được 1 website có sự phân bổ riêng giữa website đăng nhập của sinh viên và giáo vụ.

Việc thiết kế front end cho student về cơ bản đã có những chức năng như xem thông tin cá nhân, xem lịch học trong tuần, xem kết quả học tập và đăng kí học phần.

Phía frontend cho administrator đã làm được một số chức năng như tạo và xem học kì, tạo và xem khoa, tạo và xem môn học.

## Hạn chế của đồ án:

Dự án của chúng tôi hiện vẫn còn nhiều khía cạnh cần được cải thiện để đạt được hiệu quả cao nhất.

*Trước tiên, phải kể đến phần back-end:*

Kiến trúc microservice chưa được triển khai một cách tối ưu. Điều này chủ yếu bắt nguồn từ đội ngũ nhân sự vẫn đang là sinh viên, chưa từng có kinh nghiệm nào liên quan đến kiến trúc này trước đây. Dẫn đến việc chúng tôi chưa thể khai thác hết tiềm năng và lợi ích mà mô hình microservice mang lại.

Hiện tại, chúng tôi vẫn chưa hoàn thành được service dành cho cựu sinh viên, vẫn còn thiếu một vài các api nhỏ lẻ có thể sẽ cần khi frontend triển khai.

Chưa áp dụng công nghệ một cách hợp lý giữa gRPC và REST, dẫn đến việc giao tiếp giữa các dịch vụ chưa thực sự hiệu quả.

CI/CD đang là thách thức lớn vì có rất ít bên thứ ba hỗ trợ gRPC và việc nên hệ thỗng vẫn chỉ chạy ở local.

*Tiếp theo là phần front-end:*

Các tính năng quan trọng như: tạo lớp học phần, nhập điểm, tạo account cho sinh viên vẫn chưa hoàn thiện và vẫn còn thiếu các logic cần thiết để hỗ trợ người dùng trong quá trình đăng ký lớp học phần.

Hiện tại, giao diện người dùng chưa thực sự thân thiện và chưa đáp ứng đầy đủ các yêu cầu chức năng của dự án. Các tính năng hiện có vẫn đang trong quá trình phát triển và chưa đạt được mức độ hoàn chỉnh như mong muốn, điều này có thể gây ra khó khăn cho người dùng khi thao tác.

## Hướng phát triển:

* Tiếp tục hoàn thiện thêm các chức năng chưa đặt được như service quản lý cựu sinh viên ‘Alumni Service’, đăng ký xét tốt nghiệp,... Tinh chỉnh để các service có thể độc lập hơn.
* Có thể sử dụng Kafka vào quy trình đăng ký học phần để tránh trường hợp ‘thắt cổ chai’ và là trung gian trao đổi giữa nội bộ các microservice giúp giảm hết mức sự phụ thuộc.
* Ứng dụng trí tuệ nhân tạo để có thể xếp lịch học, xếp lịch giáo viên, xếp phòng,... trong service quản lý lớp học phần. Hiện tại phần này chỉ đang import file excel bên ngoài để xử lý.
* Tiếp tục tối ưu các chức năng của hệ thống, đảm bảo tính toàn vẹn và hiệu suất khi có 1 lượng traffic lớn sử dụng hệ thống. Có thể mở rộng và thu hẹp các service một cách hiệu quả để giảm chi phí vận hành.
* Thêm service ‘hỗ trợ và phản hồi’ để kịp thời giải đáp các thắc mắc, lỗi, tính năng bị chậm phản hồi cũng như ghi nhận sự đóng góp để có thể cải thiện độ hài lòng của sinh viên, giáo vụ,...

# 

# TÀI LIỆU THAM KHẢO

**Các tài liệu từ Internet**

1. Spring. Spring Document Guide [Online]. Từ <https://spring.io/quickstart>
2. Mongodb. Mongodb Document Guide [Online]. Từ [https://www.mongodb.com/docs/manual/core/document](https://www.mongodb.com/docs/manual/core/document/)
3. GRPC. GRPC Document [Online]. Từ [https://grpc.io/docs](https://grpc.io/docs/)
4. W3Schools. W3Schools Online Web Tutorials. Từ <https://www.w3schools.com/>
5. Font Awesome Docs. Font Awesome Docs | Font Awesome Docs. Từ <https://docs.fontawesome.com/>