



**ĐĂNG KÝ HỌC PHẦN THEO TÍN CHỈ**

**SỬ DỤNG KIẾN TRÚC MICROSERVICE**

***Nhóm 08 - Sinh viên thực hiện***

1. Lê Duy Khánh – 20086641
2. Nguyễn Quang Toán – 20081201
3. Thái Quang Bảo – 20093191

GVHD: TS.Võ Văn Hải

# MỤC LỤC

[MỤC LỤC 3](#_1fob9te)

[DANH MỤC CÁC BẢNG BIỂU 5](#_3znysh7)

[CHƯƠNG 1](#_2et92p0) : GIỚI THIỆU 6

[1.1](#_tyjcwt) Tổng quan 6

[1.2](#_1t3h5sf) Mục tiêu đề tài 6

[1.2.1 Mục tiêu chính: 6](#_4d34og8)

[1.2.2 Mục tiêu cụ thể: 7](#_2s8eyo1)

[1.3 Phạm vi đề tài 7](#_17dp8vu)

[1.3.1 Mục tiêu: 7](#_3rdcrjn)

[1.3.2 Phạm vi: 8](#_26in1rg)

[1.3.3 Kết quả mong đợi: 8](#_lnxbz9)

[1.4 Yêu cầu chức năng của hệ thống quản lý lớp học tín chỉ 8](#_35nkun2)

[1.4.1 Quản lý đăng ký học phần: 8](#_1ksv4uv)

[1.4.2 Quản lý thời khóa biểu: 9](#_44sinio)

[1.4.4 Xem thông tin học tập: 9](#_2jxsxqh)

[CHƯƠNG 2](#_z337ya) : CƠ SỞ LÝ THUYẾT 9

[2.1 Ngôn ngữ phát triển ứng dụng 9](#_3j2qqm3)

[2.1.1 Ngôn ngữ lập trình Java 9](#_1y810tw)

[2.1.2 Ngôn ngữ lập trình Javascript 9](#_4i7ojhp)

[2.2 Những framework hỗ trợ phát triển ứng dụng 10](#_2xcytpi)

[2.2.1 Framework Spring Boot 10](#_1ci93xb)

[2.2.2 React.js 11](#_3whwml4)

[2.2.3 Spring Security 12](#_2bn6wsx)

[2.2.4 Redis 12](#_qsh70q)

[2.2.4 Spring Cloud Gateway 13](#_3as4poj)

[2.1](#_1pxezwc) Kiến trúc phần mềm 13

[2.1.1  Giới thiệu chung về microservice 13](#_49x2ik5)

[2.1.2 So sánh monolithic và microservices 15](#_3o7alnk)

[2.1.3 Áp dụng vào dự án đăng ký học phần: 18](#_2grqrue)

[CHƯƠNG 3](#_vx1227) : PHÂN TÍCH VÀ THIẾT KẾ 24

[3.1](#_3fwokq0) Phân tích yêu cầu bằng UML 24

[3.1.1](#_1v1yuxt) Usecase tổng quát 24

[3.1.2](#_2u6wntf) Danh sách tác nhân và mô tả 24

[3.1.3](#_3tbugp1) Danh sách các tình huống hoạt động (Use cases) 24

[3.1.4](#_nmf14n) Tình huống hoạt động 26

[3.2](#_46r0co2) Class diagram 31

[CHƯƠNG 4](#_2lwamvv) : HIỆN THỰC 31

[4.1](#_111kx3o) Giao diện của hệ thống 32

[4.1.1](#_3l18frh) Giao diện web 32

[Hình 4.4 giao diện khi hủy đăng ký học phần thành công 34](#_3ygebqi)

[Hình 4.4 giao diện khi đăng ký học phần thành công 34](#_sqyw64)

[CHƯƠNG 5](#_3cqmetx) : KẾT LUẬN 34

[5.1](#_1rvwp1q) Kết quả đạt được 34

[5.2](#_4bvk7pj) Hạn chế của đồ án 35

[5.3](#_2r0uhxc) Hướng phát triển 35

[TÀI LIỆU THAM KHẢO 36](#_3q5sasy)

# DANH MỤC CÁC BẢNG BIỂU VÀ HÌNH

[Bảng 1. So sánh giữa Monolithic vs Microservices 18](#_41mghml)

[Bảng 3-1 Danh sách tác nhân và mô tả 24](#_19c6y18)

[Bảng 3-2 Danh sách các tình huống trong hệ thống 25](#_28h4qwu)

[Hình 4.1 giao diện đăng nhập 32](#_206ipza)

[Hình 4.2 giao diện xem và đăng ký học phần 32](#_4k668n3)

[Hình 4.3 giao diện xem thời khóa biểu 33](#_1egqt2p)

[Hình 4.5 giao diện khi hủy đăng ký học phần thành công 34](#_3ygebqi)

[Hình 4.6 giao diện khi đăng ký học phần thành công 34](#_sqyw64)

# : GIỚI THIỆU

## Tổng quan

Kiến trúc microservice đã trở thành một trong những xu hướng kiến trúc phần mềm hot nhất hiện nay. Nó đã thay đổi cách chúng ta xây dựng các ứng dụng lớn một cách vĩnh viễn. Thay vì áp dụng cách tiếp cận monolithic cũ, chậm chạp và phức tạp ở quá khứ, các developer và công ty trên toàn thế giới đang chuyển sang kiến trúc microservice để đơn giản hóa và mở rộng cấu trúc.

Các công ty lớn như Amazon, Netflix, Spotify và Uber cũng đã thực hiện quá trình chuyển đổi sang kiến trúc microservice này. Kiến trúc này giúp họ phát triển và triển khai các ứng dụng nhanh chóng hơn, linh hoạt hơn và dễ dàng mở rộng khi cần thiết.

Nếu bạn đang quan tâm đến lĩnh vực phát triển phần mềm, hãy nắm vững kiến thức về kiến trúc microservice vì đây chính là xu hướng không thể bỏ qua trong thế giới công nghệ ngày nay.

Trường ĐH Công nghiệp TP.HCM (IUH) là một trong những trường đại học hàng đầu của Bộ Công thương, cũng là trường có quy mô đào tạo lớn với hơn 30000 sinh viên học tập. Trường cũng đào tạo đa ngành, đa bậc học và nhiều chương trình liên kết quốc tế Trường được đầu tư cơ sở vật chất tốt, đáp ứng nhu cầu của người học và các tiêu chuẩn quốc tế. Bên cạnh, đội ngũ giảng viên có trình độ cao cùng với nhiều chương trình đào tạo đạt chuẩn quốc tế như AUN. Chính vì lẽ đó, nhà trường cần có một hệ thống website hỗ trợ đào tạo một cách hiệu quả và kịp thời cho người học. Tuy nhiên, hệ thống website hỗ trợ cho người học hiện nay còn nhiều hạn chế, điển hình là khả năng đáp ứng truy cập đồng thời của sinh viên trong một số thời điểm nhất định không được ổn định. Ví dụ như vào thời điểm nhà trường mở đăng ký lớp học phần cho học kỳ mới, mở đợt xét tuyển đầu cho các thí sinh thì nhu cầu truy cập website tăng cao một cách đột ngột. Lúc này hiện tượng trang không phản hồi khi truy cập website của nhà trường xảy ra thường xuyên trong nhiều ngày hoặc không thực hiện các chức năng trên hệ thống. Sinh viên không thể làm gì được trong vòng nhiều giờ dù đã tốn công sức và thời gian để đăng ký học phần chẳng hạn. Để xảy ra tình trạng này cũng có nhiều nguyên nhân. Nguyên nhân đầu tiên là số lượng sinh viên truy cập website của trường tăng đột biến, từ đó dẫn đến việc số lượng yêu cầu (request) gửi lên máy chủ tăng cao. Thứ hai, xử lý quá nhiều yêu cầu vượt quá giới hạn của máy chủ gây hiện tượng trang không phản hồi trong vào nhiều giờ, nhiều ngày liền. Bên cạnh đó, trong lúc hệ thống đăng ký học phần bị quá tải, sinh viên cũng khó vào xem được lịch học, lịch thi dẫn tới khi có cập nhật mới của giảng viên về phòng học hoặc thời gian học thì sinh viên không thể tiếp cận nhanh chóng.

## Mục tiêu đề tài

### 1.2.1 Mục tiêu chính:

Thiết kế và phát triển hệ thống quản lý lớp học tín chỉ cho trường đại học đáp ứng các yêu cầu:

* + Quản lý đăng ký học phần theo cơ chế tín chỉ hiệu quả và chính xác.
  + Hỗ trợ sinh viên đăng ký môn học phù hợp với chương trình đào tạo.
  + Cung cấp thông tin học tập đầy đủ và cập nhật cho sinh viên.
  + Lưu trữ thông tin cựu sinh viên phục vụ mục đích theo dõi và hỗ trợ sau tốt nghiệp.

### 1.2.2 Mục tiêu cụ thể:

* Quản lý sinh viên:
  + Biên chế sinh viên vào các khoa theo ngành học.
  + Cập nhật thông tin sinh viên đầy đủ và chính xác.
  + Phân quyền quản lý sinh viên theo khoa, ngành học.
* Quản lý môn học:
  + Danh sách các môn học theo ngành học.
  + Mô tả chi tiết từng môn học (tên môn học, mã môn học, số tín chỉ, giảng viên).
  + Mở đăng ký cho các môn học theo học kỳ.
  + Quản lý số lượng sinh viên tối đa cho mỗi lớp tín chỉ.
  + Quản lý điều kiện tiên quyết cho các môn học.
* Quản lý đăng ký học phần:
  + Cho phép sinh viên đăng ký môn học trực tuyến.
  + Kiểm tra điều kiện đăng ký (tín chỉ đã tích lũy, môn học tiên quyết, số lượng tín chỉ tối đa).
  + Cảnh báo các trường hợp đăng ký sai quy định.
  + Yêu cầu sinh viên xác nhận đăng ký trước khi hoàn tất.
  + Ghi nhận lịch sử đăng ký học phần của sinh viên.
* Quản lý thời khóa biểu:
* Tự động tạo thời khóa biểu cho sinh viên dựa trên các môn học đã đăng ký.
* Cho phép sinh viên xem thời khóa biểu trực tuyến.
* Cập nhật thời khóa biểu theo các thay đổi về lịch học, giảng viên. .(cái này cần xem xét lại về cái giảng viên)
* Quản lý cựu sinh viên:
  + Lưu trữ thông tin cựu sinh viên (tên, ngành học, năm tốt nghiệp...).
  + Cung cấp các dịch vụ hỗ trợ cho cựu sinh viên (cập nhật thông tin việc làm, tổ chức hội thảo...).

Ngoài ra, hệ thống còn có thể đáp ứng các yêu cầu khác:

* Đảm bảo tính bảo mật và an toàn dữ liệu.

## 1.3 Phạm vi đề tài

### 1.3.1 Mục tiêu:

* Thiết kế và phát triển hệ thống quản lý lớp học tín chỉ cho trường đại học đáp ứng các yêu cầu:
  + Quản lý sinh viên theo khoa, ngành học.
  + Quản lý môn học theo ngành học, bao gồm thông tin môn học, điều kiện tiên quyết, số lượng sinh viên tối đa.
  + Quản lý đăng ký học phần cho sinh viên, đảm bảo tuân thủ các quy định về tín chỉ, môn học tiên quyết, số lượng môn đăng ký tối đa.
  + Quản lý thời khóa biểu cho sinh viên.
  + Quản lý xét tốt nghiệp cho sinh viên.
  + Quản lý thông tin cựu sinh viên.
  + Cung cấp các chức năng hỗ trợ khác như: xem thông tin học tập, thời khóa biểu, tra cứu thông tin cựu sinh viên...

### 1.3.2 Phạm vi:

* Hệ thống quản lý lớp học tín chỉ sẽ bao gồm các chức năng chính sau:
  + Quản lý môn học:
    - Danh sách các môn học theo ngành học.
    - Mô tả chi tiết từng môn học (tên môn học, mã môn học, số tín chỉ, giảng viên).
    - Mở đăng ký cho các môn học theo học kỳ.
    - Quản lý số lượng sinh viên tối đa cho mỗi lớp tín chỉ.
    - Quản lý điều kiện tiên quyết cho các môn học.
  + Quản lý đăng ký học phần:
    - Cho phép sinh viên đăng ký môn học trực tuyến.
    - Kiểm tra điều kiện đăng ký (tín chỉ đã tích lũy, môn học tiên quyết, số lượng tín chỉ tối đa...).
    - Cảnh báo các trường hợp đăng ký sai quy định.
    - Yêu cầu sinh viên xác nhận đăng ký trước khi hoàn tất.
    - Ghi nhận lịch sử đăng ký học phần của sinh viên.
  + Quản lý thời khóa biểu:
    - Tự động tạo thời khóa biểu cho sinh viên dựa trên các môn học đã đăng ký.
    - Cho phép sinh viên xem thời khóa biểu trực tuyến.
* Hệ thống sẽ được xây dựng trên nền tảng công nghệ web hiện đại, đảm bảo tính bảo mật và an toàn dữ liệu.
* Hệ thống có giao diện người dùng thân thiện, dễ sử dụng cho sinh viên

### 1.3.3 Kết quả mong đợi:

* Hoàn thành thiết kế và phát triển hệ thống quản lý lớp học tín chỉ đáp ứng đầy đủ các yêu cầu đề ra.
* Hệ thống được triển khai và sử dụng hiệu quả trong nhà trường.
* Góp phần nâng cao hiệu quả quản lý đào tạo theo hệ thống tín chỉ.
* Giảm thiểu thủ tục hành chính cho sinh viên và cán bộ quản lý.
* Cung cấp thông tin học tập đầy đủ và cập nhật cho sinh viên.
* Tạo điều kiện cho sinh viên đăng ký học phần một cách chủ động và hiệu quả.
* Góp phần nâng cao chất lượng giáo dục đại học.

## 1.4 Yêu cầu chức năng của hệ thống quản lý lớp học tín chỉ

### 1.4.1 Quản lý đăng ký học phần:

* **Đăng ký học phần:**
  + Cho phép sinh viên đăng ký học phần trực tuyến.
  + Kiểm tra điều kiện đăng ký (tín chỉ đã tích lũy, môn học tiên quyết, số lượng tín chỉ tối đa).
  + Yêu cầu sinh viên xác nhận đăng ký trước khi hoàn tất.
  + Ghi nhận lịch sử đăng ký học phần của sinh viên.

### 1.4.2 Quản lý thời khóa biểu:

* **Tự động tạo thời khóa biểu:**
  + Tự động tạo thời khóa biểu cho sinh viên dựa trên các môn học đã đăng ký.
* **Xem thời khóa biểu:**
  + Cho phép sinh viên xem thời khóa biểu trực tuyến.
* **Cập nhật thời khóa biểu:**
* Cập nhật thời khóa biểu theo các thay đổi về lịch học, giảng viên.

### 1.4.4 Xem thông tin học tập:

* **Số tín chỉ đã đạt:**
  + Hiển thị số tín chỉ đã đạt của sinh viên theo từng học kỳ

# : CƠ SỞ LÝ THUYẾT

## 2.1 Ngôn ngữ phát triển ứng dụng

### 2.1.1 Ngôn ngữ lập trình Java

Java là ngôn ngữ lập trình hướng đối tượng đang được duy trì và phát triển bởi Oracle. Java được sử dụng trong phát triển phần mềm ứng dụng desktop, website, game hay ứng dụng trên các thiết bị di động. Java cũng là một ngôn ngữ cho phép phát triển các ứng dụng chạy trên mọi nền tảng. Hiện nay, đội ngũ phát triển ứng ụng dùng ngôn ngữ Java ngày càng hùng hậu. Đội ngũ này cũng cung cấp một tập hợp các chương trình giúp lập trình viên phát triển và chạy các ứng dụng một cách hiệu quả. Java được tổ chức bao gồm một công cụ thực thi, một trình biên dịch và một bộ thư viện. James Gosling đã phát triển nền tảng Java tại Sun Microsystems và tập đoàn Oracle sau đó đã mua lại nó và phát triển đến ngày hôm nay.

### 2.1.2 Ngôn ngữ lập trình Javascript

Javascript ra đời năm 1995, ban đầu Javascript không được xem là một ngôn ngữ lập trình mà chỉ được xem là một ngôn ngữ kịch bản (scripting language). Nguyên nhân là thời bấy giờ Javascript chỉ được sử dụng để xây dựng các tương tác với trang Web. Bản thân nó phải thực thi trong một chương trình không phải hệ điều hành (cụ thể là trình duyệt Web). Nhưng sau này, khi vai trò và sự phổ biến của websites ngày càng lớn, cộng với đó là sự ra đời và lớn mạnh thần tốc của NodeJS (2009) giúp Javascript không còn là một ngôn ngữ phải thực thi trong trình duyệt. Hiện Javascript chính thức được công nhận là một ngôn ngữ lập trình chính thống. Javascript hoạt động được trên nhiều nền tảng (Web, Server) và là ngôn ngữ đơn giản, dễ học cho người mới sử dụng. Hiện nay, nhiều công nghệ xây dựng dựa trên Javascript như Nodejs, React, VueJS...Đồng thời Javasript cũng được dùng cho việc xây dựng nhiều ứng dụng với vai trò khác nhau từ Web, ứng dụng Mobile và một số ứng dụng khác.

A white circle with colorful lines around it

Description automatically generated

## 2.2 Những framework hỗ trợ phát triển ứng dụng

### 2.2.1 Framework Spring Boot

Spring Boot là khuôn mẫu (framework) phát triển dựa trên ngôn ngữ Java trong hệ sinh thái Spring framework. Nó giúp cho các lập trình viên đơn giản hóa quá trình lập trình một ứng dụng với Spring. Trong Spring Boot, người lập trình chỉ tập trung vào việc phát triển các nghiệp vụ cho ứng dụng mà không quan tâm nhiều về kiến trúc và mã nguồn của ứng dụng. Một kiến trúc webserver xây dựng dựa trên spring boot gồm các thành phần cơ bản như sau: Repository Class Extending, Service Layer, Model, Controller.

Repository Class Extending là các thư viện hoặc framework được phát triển và duy trì bởi cộng đồng các lập trình viên Java để hỗ trợ quá trình xây dựng hệ thống một cách nhanh chóng hoặc cũng có thể là các lớp chứa các phương thức xử lý, truy xuất dữ liệu (CRUD) do lập trình viên viết ra; Service Layer là lớp dịch vụ chứa các phương thức (hàm) xử lý các yêu cầu như chỉnh sửa, lớp Service thường sử dụng các phương thức của các lớp trong Repository Class Extending (bằng cách gọi tới một hay kết hợp nhiều phương thức với nhau) để truy xuất, thêm, xóa dữ liệu trong cơ sở dữ liệu. Nó là nơi tập trung các xử lý nghiệp vụ của hệ thống Back-end; Model là thành phần được đính kèm theo các truy vấn của Services Layer xuống cơ sở dữ liệu. Model đóng nhiều vai trò quan trọng như phát sinh hệ cơ sở dữ liệu theo liên kết các lớp của lập trình viên, chuyển đổi kết quả truy xuất từ cơ sở dữ liệu thành các thực thể lớp; Controller là thành phần giao tiếp trực tiếp với phía client, tồn tại dưới dạng các API. Model có trách nhiệm xử lý yêu cầu của client, xác minh các yêu cầu đó. Model truy cập đến phương thức phù hợp trong Service Layer để đạt được kết quả mong muốn, trả kết quả về cho phía client.

A diagram of a spring boot flow

Description automatically generated

### 2.2.2 React.js

ReactJS là một thư viện mã nguồn mở được sử dụng để phát triển giao diện người dùng (UI) đặc biệt cho các ứng dụng trang web một trang. ReactJS cho phép các lập trình viên tạo ra các ứng dụng web lớn có thể sử dụng dữ liệu và thay đổi theo thời gian mà không cần tải lại trang. Do đó, ReactJS có một thuật toán so sánh thông minh mà nó sử dụng để chỉ tải lại DOM cần được tải lại trong khi giữ lại tất cả những gì khác nguyên vẹn. Việc sử dụng các thành phần có thể tái sử dụng cung cấp một cách dễ dàng để xây dựng ứng dụng của chúng ta. Những tiện ích của ReactJS cũng khiến việc sắp xếp giao diện người dùng trở nên nhất quán và giảm bớt áp lực đối với các lập trình viên để họ có thể tập trung vào các chức năng và logic nghiệp vụ quan trọng hơn. ReactJS cũng không ép buộc một cách cụ thể để thực hiện một nghiệp vụ cụ thể nào đó. Nó cung cấp một bộ thư viện phong phú mà người dùng có thể lựa chọn để thực hiện một nghiệp vụ cụ thể. Phương thức Lifecycle và React Hooks là các tính năng quan trọng khác quản lý chuỗi sự kiện được gọi trong suốt vòng đời của một thành phần. ReactJs là công nghệ sử dụng việc xây dựng ứng dụng web dựa trên JavaScript được duy trì và phát triển bởi Facebook. ReactJs được nhận xét là dễ học, dễ hiểu, dễ làm quen nhưng hiệu quả lại cao, phù hợp với nhiều ứng dụng web, kể cả những ứng dụng có quy mô cực lớn (như Facebook). Tương tự React Native, ReactJs cũng có cộng đồng lớn mạnh, nhiều thư viện và framework hỗ trợ. Đội ngũ phát triển React cũng đang không ngừng làm việc hằng ngày để phát triển công nghệ này ngày một tốt hơn. Theo khảo sát của Stack Overflow, ReactJS là công nghệ Front-end được nhiều người quan tâm nhất trong vòng những năm từ 2019 trở lại đây (Hình 2.2.3). Cụ thể trong năm 2020, React.js dẫn đầu bảng khảo sát ở mức độ phổ biến và yêu thích trong danh sách các framework Front-end.

A graph of a graph showing the number of years

Description automatically generated with medium confidence

### 2.2.3 Spring Security

Spring Security được phát triển bởi SpringSource (hiện thuộc Pivotal) và được xem là một trong những framework bảo mật phổ biến nhất cho ứng dụng Java. Spring Security cung cấp các tính năng xác thực (authentication) và phân quyền (authorization) cho các ứng dụng, cũng như hỗ trợ các tiêu chuẩn và giao thức bảo mật như HTTPS, OAuth2, JWT, LDAP, SAML, OpenID Connect. Spring Security được thiết kế theo kiến trúc plugin, cho phép tùy biến linh hoạt và dễ dàng theo nhu cầu của ứng dụng và được tích hợp sẵn với các thành phần khác của Spring Framework, như Spring Boot, Spring MVC, Spring Data, Spring Cloud, và Spring WebFlux.

A screenshot of a computer

Description automatically generated

### 2.2.4 Redis

Redis (REmote DIctionary Server) là một mã nguồn mở được dùng để lưu trữ dữ liệu có cấu trúc, có thể sử dụng như một database, bộ nhớ cache hay một message broker. Redis là cơ sở dữ liệu trong bộ nhớ nhanh nhất thế giới. Nó cung cấp các giải pháp đám mây và tại chỗ cho bộ nhớ đệm, tìm kiếm vectơ và cơ sở dữ liệu NoSQL phù hợp liền mạch với mọi nhóm công nghệ giúp người sử dụng dễ dàng xây dựng, mở rộng quy mô và triển khai các ứng dụng nhanh. Redis cung cấp  các cấu trúc dữ liệu  như  strings, hashes, lists, sets, sorted sets với các truy vấn phạm vi,  bitmaps, hyperloglog, geospatial indexes, and streams. Redis có tính năng  sao chép tích hợp, Lua scripting, LRU eviction, transactions và các mức độ duy trì trên đĩa khác nhau, đồng thời cung cấp tính sẵn sàng cao thông qua Redis Sentinel và phân vùng tự động với Redis Cluster.

A red cube with white symbols on it

Description automatically generated

### 2.2.4 Spring Cloud Gateway

Spring Cloud Gateway là một thành phần trong Spring Cloud được sử dụng để xây dựng cổng API cho các hệ thống microservices. Dưới đây là một số lý do tại sao bạn nên sử dụng Spring Cloud Gateway: Định tuyến và Load Balancing: Spring Cloud Gateway kiểm soát việc định tuyến các yêu cầu đến các dịch vụ khác nhau trong hệ thống. Nó cũng kiêm luôn vai trò load balancer, giúp giảm thiểu rủi ro hệ thống bị quá tải. Bảo mật: Gateway là một lớp bảo mật bổ sung cho hệ thống. Requests không được gửi trực tiếp đến backend, giúp tăng tính bảo mật. Quản lý và kiểm soát quyền truy cập: Spring Cloud Gateway cho phép bạn quản lý và kiểm soát quyền truy cập đến các API. Bạn có thể áp dụng các chính sách bảo mật, xác thực và ủy quyền tại cổng này. Tăng hiệu suất và khả năng mở rộng: Gateway giúp tối ưu hóa việc gửi yêu cầu đến các dịch vụ, cải thiện hiệu suất và dễ dàng mở rộng hệ thống.

## Kiến trúc phần mềm

### 2.1.1  Giới thiệu chung về microservice

#### 2.1.1.1 Khái niệm

Microservices là một phong cách kiến trúc cực kỳ phổ biến và đã đạt được đà phát triển đáng kể trong những năm gần đây.

Microservices là một kiến trúc mà trong đó ứng dụng được phát triển và triển khai dưới dạng một tập hợp các dịch vụ độc lập và nhỏ gọn. Mỗi dịch vụ đảm nhận một phần nhỏ của toàn bộ ứng dụng và chúng tương tác với nhau thông qua các giao diện được định rõ. Mỗi dịch vụ có thể được triển khai, mở rộng và quản lý độc lập, giúp tăng tính linh hoạt và mở rộng của hệ thống.

Do tính chất đơn mục đích, quy mô dịch vụ trong microservice nhỏ hơn nhiều so với các kiến trúc phân tán khác, chẳng hạn như kiến trúc hướng dịch vụ điều phối. Các kiến trúc sư mong đợi mỗi dịch vụ bao gồm tất cả các bộ phận cần thiết để hoạt động độc lập, bao gồm cơ sở dữ liệu và các thành phần phụ thuộc khác.

Microservices tạo thành một kiến trúc phân tán: mỗi dịch vụ chạy trong quá trình của riêng nó, ban đầu ám chỉ một máy tính vật lý nhưng nhanh chóng phát triển thành máy ảo và các container. Việc phân tách các dịch vụ đến mức độ này cho phép một giải pháp đơn giản cho một vấn đề phổ biến trong các kiến trúc có tính đến cơ sở hạ tầng multitenant (hệ thống để phục vụ nhiều khách hàng hoặc "tenant" (người thuê) khác nhau từ một môi trường chung) để lưu trữ ứng dụng. Microservices được ảnh hưởng mạnh mẽ bởi các ý tưởng trong thiết kế theo miền (DDD viết tắt của domain-driven design), một quy trình thiết kế logic cho các dự án phần mềm. Một khái niệm cụ thể từ DDD, bounded context, đã rõ ràng ảnh hưởng đến microservices. Khái niệm bounded context đại diện cho một kiểu phân tách. Khi một nhà phát triển xác định một miền, miền đó bao gồm nhiều thực thể và hành vi, được xác định trong các tài liệu như mã và cơ sở dữ liệu.

Microservices đưa khái niệm của một kiến trúc được phân chia theo miền tới mức độ cực đại. Mỗi dịch vụ được ý định đại diện cho một miền hoặc miền phụ; ở nhiều cách, microservices là sự hiện thân vật lý của các khái niệm logic trong thiết kế theo miền.

#### 2.1.1.2 Ưu nhược điểm của microservices

**2.1.1.2.1 Ưu điểm của kiến trúc microservice**

Dựa trên việc chia nhỏ hệ thống thành nhiều service riêng biệt, kiến trúc microservice có những ưu điểm sau:

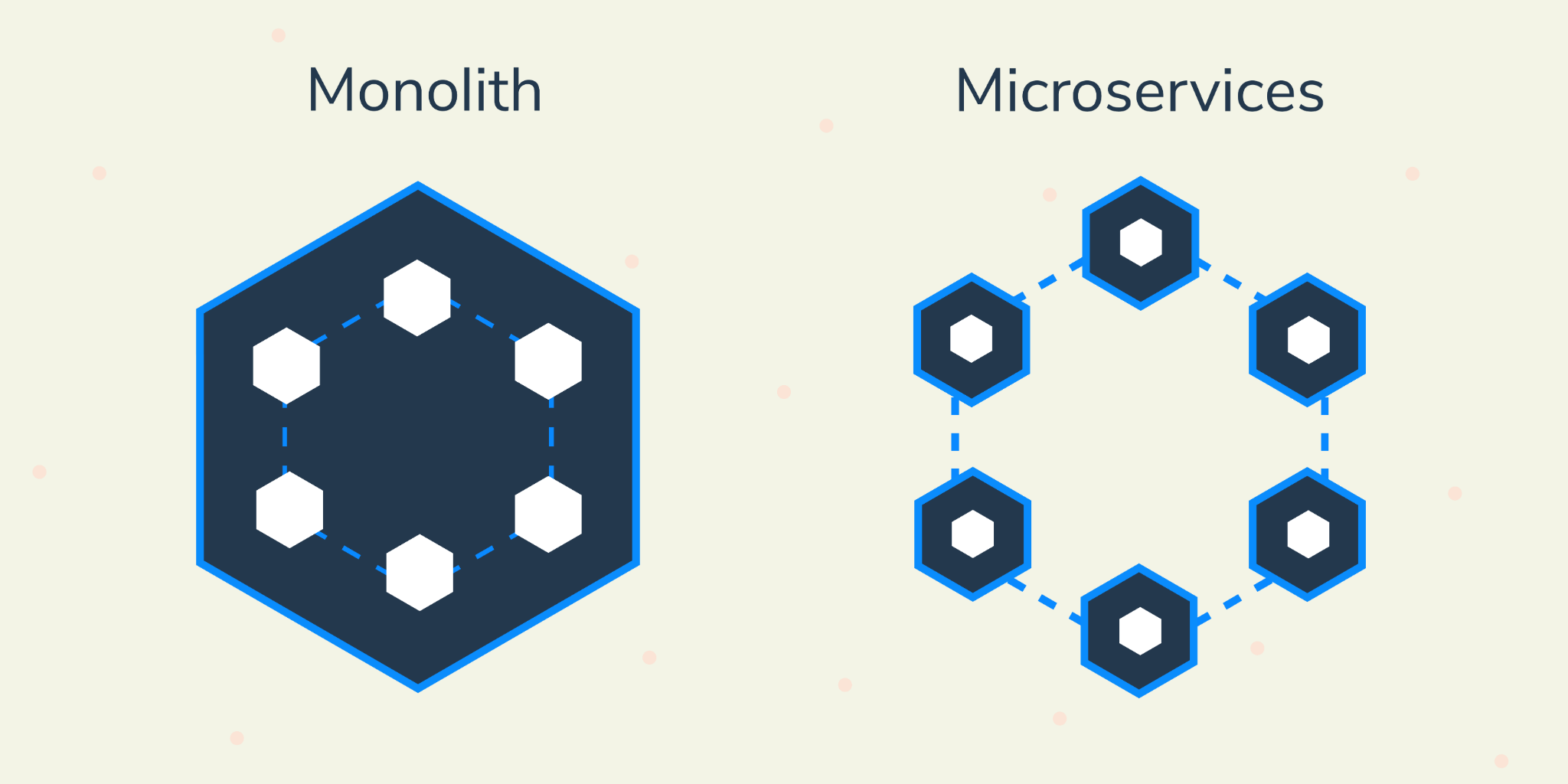
* Các service có thể được bảo trì độc lập: Các service là riêng biệt, vì vậy khi có một service được bảo trì nó không làm ảnh hưởng tới các service khác. Lưu ý, không ảnh hưởng ở đây là không gây “chết” các service khác nếu chẳng may server đang bảo trì gặp sự cố, chứ không phải là thiếu service mà hệ thống vẫn hoạt động đầy đủ tính năng.
* Dễ dàng nắm bắt các business logic: Mỗi service xử lý một nghiệp vụ riêng, vì vậy mà ta có thể dễ dàng nắm bắt được các business logic. Tên của mỗi service sẽ đại diện cho chức năng mà service đó thực hiện(USER AUTH SERVICE, COURSES SERVICE,...) giúp cho người quản lý dễ dàng nắm bắt được tiến độ cũng như nghiệp vụ mà hệ thống đang thực thi.
* Áp dụng được nhiều công nghệ: Mỗi service được coi như một dự án riêng, vì vậy mà chúng có thể sử dụng các công nghệ (như ngôn ngữ lập trình, database) khác nhau. Chứng chỉ cần đảm bảo tuân theo chuẩn giao tiếp được đặt ra từ trước, còn bên trong chúng thực hiện ra sao và sử dụng công nghệ gì không ảnh hưởng tới các service còn lại. Ví dụ một trang web với phần lớn các service được viết bằng Java, nhưng riêng service thực hiện gửi – nhận thông báo realtime lại được viết bằng NodeJS. Trên thực tế, một hệ thống phần mềm triển khai theo kiến trúc microservice mà có các service phát triển trên các nền công nghệ khác nhau là chuyện hết sức bình thường.
* Dễ dàng mở rộng khi hệ thống trở nên phức tạp: Microservice có kiến trúc khá tương đồng với kiến trúc của một hệ phân tán, nên nó cũng có khả năng mở rộng theo chiều ngang. Hiểu nhanh, nếu bạn cần mở rộng hệ thống bạn chỉ cần bổ sung thêm service.

**2.1.1.2.2 Nhược điểm của kiến trúc microservice**

Bên cạnh những ưu điểm, kiến trúc microservice có các nhược điểm:

* Có độ trễ cao: Các service giao tiếp với nhau thông qua môi trường mạng, nên sẽ có độ trễ cao hơn so với kiến trúc monolithic. Để giảm độ trễ xuống mức tối thiểu, buộc các developer phải thiết kế sao cho chúng chỉ thật sự gọi tới nhau khi cần.
* Khó đảm bảo tính toàn vẹn dữ liệu: Vì dữ liệu có thể được lưu trên nhiều server khác nhau, nên khó có thể đảm bảo được tính toàn vẹn. Giả sử khi đăng ký học phần, thì thông tin sinh viên phải tạo trước rồi mới tạo thông tin học phần được đăng ký. Với kiến trúc monolithic, database được lưu tập trung và chúng thường được ràng buộc khóa ngoại để đảm bảo tính toàn vẹn. Nhưng với microservice khi “user” và “học phần” là hai service khác nhau thì không có gì (về lý thuyết) để đảm bảo rằng dữ liệu phải được tạo ra theo đúng thứ tự cả.
* Khả năng chịu lỗi và độ tin cậy bị ảnh hưởng khi sử dụng quá nhiều giao tiếp giữa các dịch vụ.

### 2.1.2 So sánh monolithic và microservices

****

|  |  |  |
| --- | --- | --- |
|  | **Monolithic** | **Microservices** |
| **Khái niệm** | Kiểu kiến trúc truyền thống khi muốn xây dựng và triển khai ứng dụng. Kiến trúc này dựa trên khái niệm về một thể duy nhất, không tách rời bao gồm server, client và database. | Kiểu kiến trúc mà cấu trúc một ứng dụng bằng các service được ghép nối lỏng (*loosely coupled*). Các collection hoặc mô-đun này có thể được phát triển, triển khai và bảo trì độc lập (develop, deploy, maintain). |
| **Ưu điểm** | * Đơn giản hóa quá trình vận hành và phát triển * Hiệu suất tối ưu | * Mỗi team có thể hoạt động độc lập về logic, domain, giúp giảm thiểu effort. * Miễn là API đầu ra chuẩn thì bất kỳ ngôn ngữ nào cũng có thể sử dụng để viết microservice. * Có thể vừa làm việc với hệ thống cũ, vừa dần cải thiện mã nguồn và thay thế các phần cũ của hệ thống. * Tích hợp nhanh và đơn giản |
| **Nhược điểm** | * Hệ thống monolithic khó để maintain. * Nhiều hệ thống cũ được dựng sơ sài, khó test và sử dụng công nghệ lỗi thời. * Khó mở rộng * Kém linh hoạt | * Thường phải làm việc ở những dự án phức tạp. * Vận hành một hệ thống microservice yêu cầu nhiều effort hơn bình thường. * Dù tất cả microservice phải được test tích hợp, thì mỗi microservice cũng cần phải test độc lập. * Những thay đổi mà ảnh hưởng đến nhiều microservice sẽ khó được thực hiện. |
| **Hiệu suất và khả năng mở rộng** | Xây dựng ứng dụng dựa trên một cấu trúc duy nhất, trong đó tất cả các thành phần và chức năng được tích hợp vào một codebase lớn. Điều này giúp tăng hiệu suất cho các tác vụ đơn giản, ít phức tạp. | Mô hình phân tách ứng dụng thành các dịch vụ nhỏ độc lập, cho phép mở rộng linh hoạt theo nhu cầu. Mỗi dịch vụ có thể được mở rộng riêng biệt và quản lý độc lập. Điều này giúp cải thiện hiệu suất và khả năng mở rộng của ứng dụng khi tải tăng. |
| **Quản lý dự án và phát triển** | Quản lý dự án và phát triển trở nên đơn giản hơn. Tất cả các thành phần, chức năng nằm trong một module duy nhất, giúp việc quản lý và triển khai trở nên dễ dàng, tiết kiệm thời gian. Tuy nhiên, khi dự án lớn hơn và độ phức tạp cao, việc quản lý trở nên khó khăn hơn. | Yêu cầu phải quản lý nhiều services riêng biệt. Điều này đòi hỏi một quy trình quản lý dự án tốt hơn và sự phối hợp chặt chẽ giữa các đội phát triển. Tuy nhiên, mô hình này cung cấp khả năng linh hoạt và phát triển độc lập cho từng services giúp developer dễ dàng thay đổi và quản lý từng thành phần nhỏ. |
| **Độ linh hoạt và sửa lỗi** | Độ linh hoạt giới hạn do tất cả các thành phần của ứng dụng được gắn kết chặt chẽ nên khi lỗi một phần của ứng dụng có thể ảnh hưởng đến các phần khác, làm tăng độ phức tạp của việc debug và tìm ra nguyên nhân gốc rễ của lỗi. | Có khả năng linh hoạt cao hơn trong việc sửa lỗi và mở rộng tính năng do mỗi microservice hoạt động độc lập, giúp giảm thiểu rủi ro và tối ưu quy trình sửa lỗi. |
| **Quản lý tài nguyên và chi phí** | Yêu cầu ít tài nguyên hơn trong việc triển khai và quản lý. Một máy chủ đơn lẻ có thể đủ để chạy toàn bộ ứng dụng. Điều này giúp giảm chi phí phát triển và vận hành. | Yêu cầu phải có quy trình quản lý tài nguyên tốt hơn. Mỗi dịch vụ cần có môi trường chạy độc lập và tài nguyên riêng. Điều này có thể tạo ra một số chi phí bổ sung, nhưng cung cấp khả năng mở rộng linh hoạt và tối ưu hóa sử dụng tài nguyên. |

## Bảng 1. So sánh giữa Monolithic vs Microservices

Với tính chất cần scale cao và lượng request mỗi lần đăng ký là rất lớn, Monolithic phù hợp các hệ thống và ứng dụng nhỏ, khó có thể đáp ứng được yêu cầu của hệ thống, dẫn đến hệ thống có thể bị sập trong lúc người dùng sử dụng như web trường ở thời điểm hiện tại. Việc phân tách ứng dụng và mỗi dịch vụ chạy độc lập, cho phép các tính năng khác(đăng nhập, xem lịch học,...) không bị ảnh hưởng khi số lượng người dùng lớn thực hiện chức một chức năng (đăng ký học phần). Bên cạnh đó nếu một chức năng lỗi cũng không ảnh hưởng đến toàn hệ thống như kiến trúc Monolithic là một ưu điểm lớn của Microservices đối với một hệ thống lớn có nhiều request cùng lúc như hệ thống đăng ký học phần. Khả năng phát hiện và sửa lỗi ở mỗi chức năng của Microservices cũng là một điểm mạnh so với Monolithic do tính độc lập của các service.

Trên đây là những điểm khác nhau giữa monolithic vs microservices. từ những điểm khác biệt của hai kiểu kiến trúc, nhóm chọn kiến trúc Microservices để hiện thực đề tài đăng ký học phần theo tín chỉ.

### 2.1.3 Áp dụng vào dự án đăng ký học phần:

Lý do áp dụng kiến trúc microservice vào dự án:

Do tính chất phân tán của microservice và có thể giải quyết được vấn đề của cấu trúc nguyên khối.

* Dễ dàng update hệ thống khi có thay đổi nhỏ,mỗi khi có một chức năng có sự thay đổi sẽ không cần phải chạy lại cả server mà chỉ cần chạy lại service vừa cập nhật. Điều này giúp người dùng có thể thực hiện các chức năng khác mà không bị out khỏi hệ thống. Điều này cũng giúp lập trình viên có thể thêm, bớt hoặc chỉnh sửa một dịch vụ mà không ảnh hưởng đến dịch vụ hoặc các nghiệp vụ khác.
* Với khả năng mở rộng độc lập đối với mỗi microservice. Mỗi khi nhóm muốn thêm một service mới hoặc một chức năng mới chỉ cần làm riêng ra một trình biên dịch mới gắn URL của service đó vào api gateway sau đó thêm đường dẫn và tham số truy vấn của service đó vào đăng ký dựa vào đó gateway sẽ truy vấn đến dịch vụ backend phù hợp, vì vậy nên với gateway có thể dễ dàng mở rộng hơn. Từ đó, hệ thống được scale một cách dễ dàng giúp phục vụ người dùng với nhiều chức năng cũng như hỗ trợ cho số lượng người dùng lớn hơn cùng một lúc.
* Dữ liệu có thể phi tập trung. Nhóm có thể chọn dùng các databases hay storage khác nhau. Có thể chọn NoSQL database hoặc relational database nếu chúng hiệu quả cho dự án. Một microservice cũng có thể chỉ cần một database được lưu trữ theo key đơn giản như Redis. Nhóm có thể chọn sự kết hợp giữa Cloud, MySQL và MongoDB.Nhóm có thể tận dụng các cơ sở dữ liệu khác nhau để lưu trữ các loại dữ liệu khác nhau. Điều này giảm thiểu việc truy xuất cơ sở dữ liệu trong khi đăng ký học phần, người dùng vẫn có thể đăng nhập và xem lịch học bình thường mặc dù số lượng người dùng đang đăng ký học phần là rất lớn.

Dựa vào những lý do trên nhóm sẽ thực hiện những công việc:

- Nghiệp vụ xử lý được tự động tối đa và có thể cài đặt linh động cho các yêu cầu phát sinh trong quá trình vận hành như thêm tự động thêm lịch học vào thời khóa biểu sau khi đăng ký hoặc tự động ném lỗi khi người dùng đăng ký trùng lịch,...

- Hệ thống Web có giao diện người dùng trực quan, chạy được trên nhiều kích thước màn hình, các thao tác xử lý được thực hiện

Từ những định nghĩa trên, nhóm áp dụng vào dự án có những yêu cầu sau về hệ thống quản lý lớp học tín chỉ của trường đại học:

* Quản lý đăng ký học phần của sinh viên theo cơ chế tín chỉ, bao gồm biên chế sinh viên vào các khoa theo ngành học của họ.
* Chọn một số môn học cho từng ngành và mở đăng ký cho sinh viên. Mỗi lớp tín chỉ có số lượng sinh viên tối đa được quy định.
* Sinh viên chỉ được đăng ký môn học nếu đã hoàn thành các môn học tiên quyết tương ứng.
* Sinh viên có thể đăng ký tối đa 30 tín chỉ trong mỗi học kỳ.
* Đảm bảo rằng sinh viên không đăng ký các môn học trùng lịch và chỉ cho phép đăng ký vào các môn học không trùng lịch.
* Yêu cầu sinh viên xác nhận trước khi đăng ký môn học và không cho phép hủy bỏ đăng ký sau khi đã xác nhận.
* Cung cấp khả năng hủy bỏ đăng ký cho các trường hợp đặc biệt sau khi được phê duyệt từ phía trường.
* Cho phép sinh viên xem thông tin học tập của mình, thời khóa biểu và các thông tin khác.
* Lưu trữ thông tin của các cựu sinh viên để theo dõi và hỗ trợ các hoạt động sau khi tốt nghiệp.

Sơ đồ kiến trúc cho dự án:

A diagram of a company

Description automatically generated

Cấu trúc Microservice Architecture trong hệ thống quản lý lớp học tín chỉ được mô tả trong ảnh như sau:

* USER INTERFACE (Giao diện người dùng):
  + Đây là phần giao diện mà người dùng tương tác trực tiếp.
  + Kết nối với cả hai dịch vụ dưới đây.
  + Một tầng UI duy nhất giúp giảm sự phức tạp trong việc triển khai và duy trì hệ thống.
  + Không cần phải quản lý nhiều mã nguồn giao diện khác nhau.
  + Kiểm tra và gỡ lỗi giao diện người dùng dễ dàng hơn khi chỉ có một tầng UI.
* API GATEWAY
  + Hoạt động như một điểm truy cập duy nhất.
  + Vai trò là dịch vụ biên giúp hiển thị các dịch vụ ra UI dưới dạng API được quản lý.
  + Có nhiệm bổ sung như cân bằng tải đơn giản, xác thực và ủy quyền, xử lý lỗi, kiểm tra, dịch giao thức và định tuyến.
  + Tách người dùng khỏi các service. Các service có thể được cập nhật hoặc tái cấu trúc mà không cần cập nhật tất cả các máy khách.
  + API Gateway có thể thực hiện các chức năng xuyên suốt như xác thực, ghi nhật ký và cân bằng tải.
* USER AUTH SERVICE (Dịch vụ xác thực người dùng):
  + Quản lý thông tin người dùng, bao gồm xác thực và quản lý quyền truy cập.
  + Kết nối với cơ sở dữ liệu USER DATA.
* CLASS-COURSE-SERVICE (Dịch vụ quản lý lớp học và khóa học):
  + Quản lý thông tin về lớp học và khóa học.
  + Kết nối với cơ sở dữ liệu COURSE DATA và USER COURSE DATA.
* COURSE DETAIL SERVICE (Dịch vụ chi tiết khóa học):
  + Quản lý thông tin chi tiết về khóa học.
  + Kết nối với cơ sở dữ liệu COURSE\_DETAIL\_ENTITY.
* COURSES SERVICE (Dịch vụ quản lý khóa học):
  + Quản lý danh sách khóa học.
  + Kết nối với cơ sở dữ liệu COURSES\_ENTITY.
* CLASS SERVICE (Dịch vụ quản lý lớp học):
  + Quản lý danh sách lớp học.
  + Kết nối với cơ sở dữ liệu CLASS\_ENTITY.
* USER COURSE REGISTER SERVICE (Dịch vụ quản lý khóa học của người dùng):
  + Quản lý thông tin đăng ký khóa học của người dùng.
  + Kết nối với cơ sở dữ liệu USER COURSE DATA.
* DATABASE
  + Chia làm 3 phần tương ứng với 3 hoạt động chính của hệ thống (đăng nhập, tương tác với học phần, )
  + Mỗi dịch vụ quản lý dữ liệu của chính nó, bao gồm toàn bộ schema hoặc một số bảng.
  + Thay đổi trong cơ sở dữ liệu của một dịch vụ không ảnh hưởng đến các dịch vụ khác.
  + Dễ dàng mở rộng cơ sở dữ liệu cho từng dịch vụ.
  + Cho phép sử dụng nhiều loại cơ sở dữ liệu khác nhau cho các dịch vụ khác nhau.
  + Không có điểm thất bại duy nhất trong ứng dụng.
  + DATABASE tương ứng với dịch vụ của user

A screenshot of a computer

Description automatically generated

* + DATABASE tương ứng với dịch vụ của học phần

A diagram of a course

Description automatically generated

* + DATABASE tương ứng với dịch vụ của quản lý user và học phần mà user đăng ký

A screenshot of a computer register

Description automatically generated

# : PHÂN TÍCH VÀ THIẾT KẾ

## Phân tích yêu cầu bằng UML

### Usecase tổng quát

A diagram of a system

Description automatically generated

### 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** |
| User | Đăng nhập  Đăng ký học phần  Xem lịch học |

## Bảng 3-1 Danh sách tác nhân và mô tả

### Danh sách các tình huống hoạt động (Use cases)

|  |  |
| --- | --- |
| **ID use case** | **Tên Use case** |
| UC01 | Đăng nhập |
| UC02 | Đăng ký học phần |

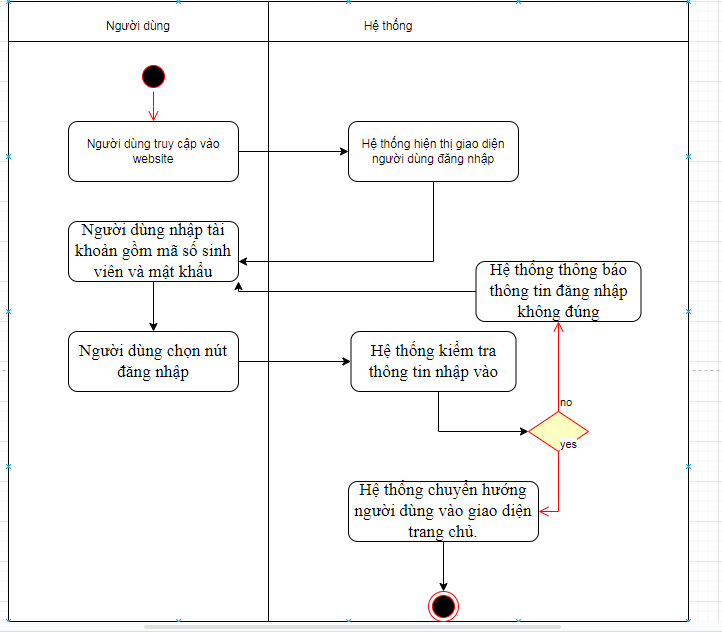
## Bảng 3-2 Danh sách các tình huống trong hệ thống

### Tình huống hoạt động

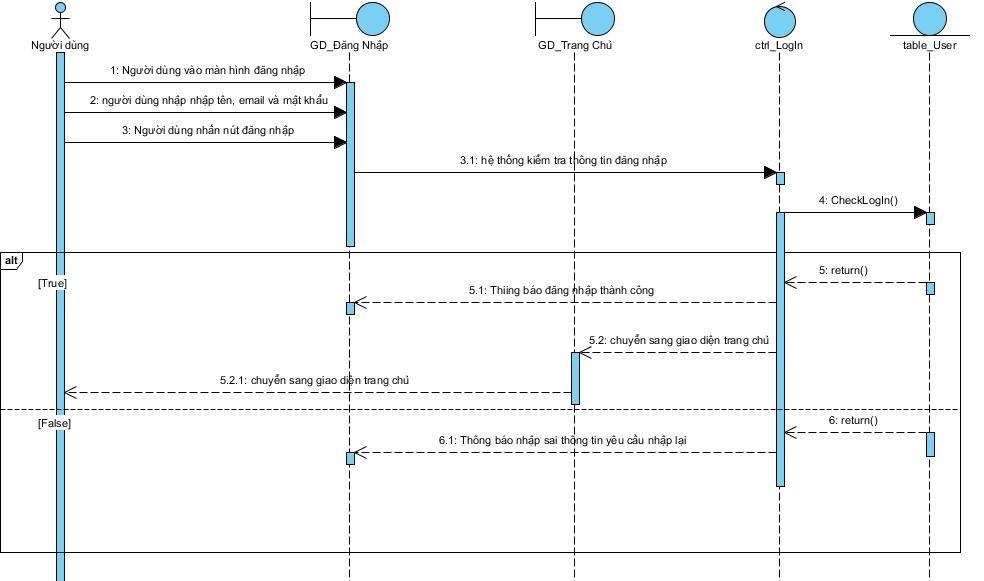
#### UC001\_Đăng nhập

|  |  |
| --- | --- |
| * **Tên use case**: UC001\_Đăng nhập | |
| * **Mô tả sơ lược**: Sinh viên đăng nhập vào hệ thống để đăng ký học phần | |
| * **Actor chính**: Sinh viên | |
| * **Actor phụ**: không có | |
| * **Tiền điều kiện (Pre-condition):** Người dùng đã có tài khoản do trường cấp | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Người vào trang đăng nhập | 1. Hệ thống hiển thị giao diện nhập thông tin tài khoản |
| 1. Người dùng nhập tài khoản gồm mã số sinh viên và mật khẩu |  |
| 1. Chọn nút đăng nhập | 1. Hệ thống kiểm tra thông tin nhập vào |
|  | 1. Hệ thống chuyển hướng người dùng vào giao diện trang chủ. |
| * **Luồng sự kiện thay thế (alternative flow):** | |
|  | 5.1. Hệ thống thông báo thông tin tài khoản đăng nhập không đúng |
|  | 5.2. Hệ thống quay lại bước 3 |

Activity diagram:



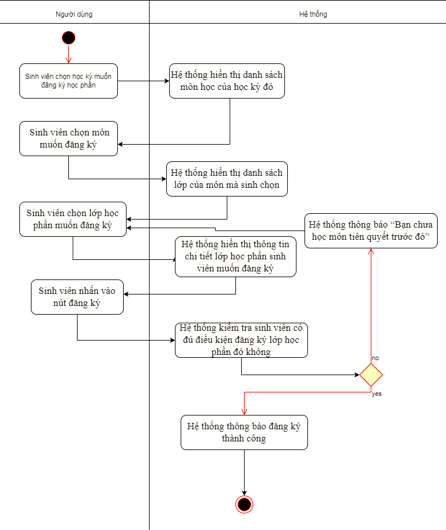
Sequence diagram:



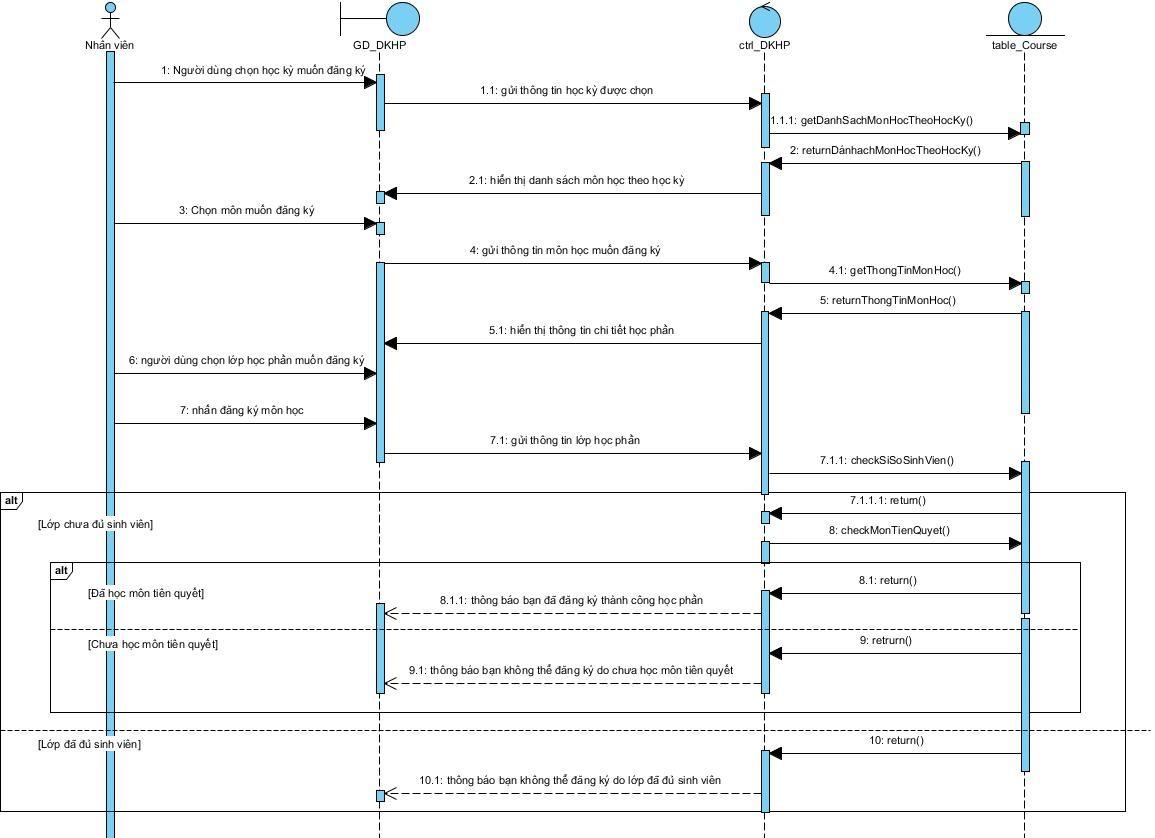
#### UC002\_Đăng ký học phần

|  |  |
| --- | --- |
| * **Tên use case**: UC002\_Đăng ký học phần | |
| * **Mô tả sơ lược**: Sinh viên đăng ký học phần mà mình chọn | |
| * **Actor chính**: Sinh viên | |
| * **Actor phụ**: không có | |
| * **Tiền điều kiện (Pre-condition):** Sinh viên phải đăng nhập vào hệ thống | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Sinh viên chọn học kỳ muốn đăng ký | 1. Hệ thống hiển thị danh sách môn học của học kỳ đó |
| 1. Sinh viên chọn môn muốn đăng ký | 1. Hệ thống hiển thị danh sách lớp của môn mà sinh chọn |
| 1. Sinh viên chọn lớp học phần muốn đăng ký | 1. Hệ thống hiển thị thông tin chi tiết lớp học phần sinh viên muốn đăng ký |
| 1. Sinh viên nhấn vào nút đăng ký | 1. Hệ thống kiểm tra sinh viên có đủ điều kiện đăng ký lớp học phần đó không |
|  | 1. Hệ thống thông báo đăng ký thành công |
| * **Luồng sự kiện thay thế (alternative flow):** | |
|  | 8.1. Hệ thống thông báo “Bạn chưa học môn tiên quyết trước đó” |
|  | 8.2. Hệ thống quay lại bước 5 |

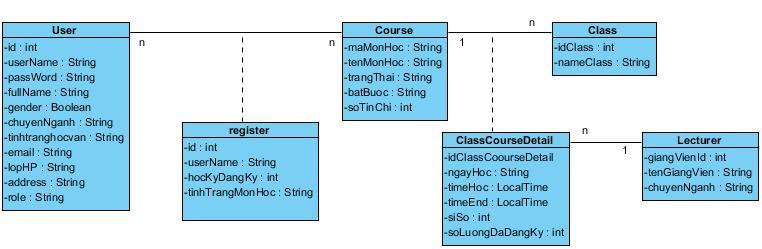
Activity diagram:



Sequence diagram:



## Class diagram



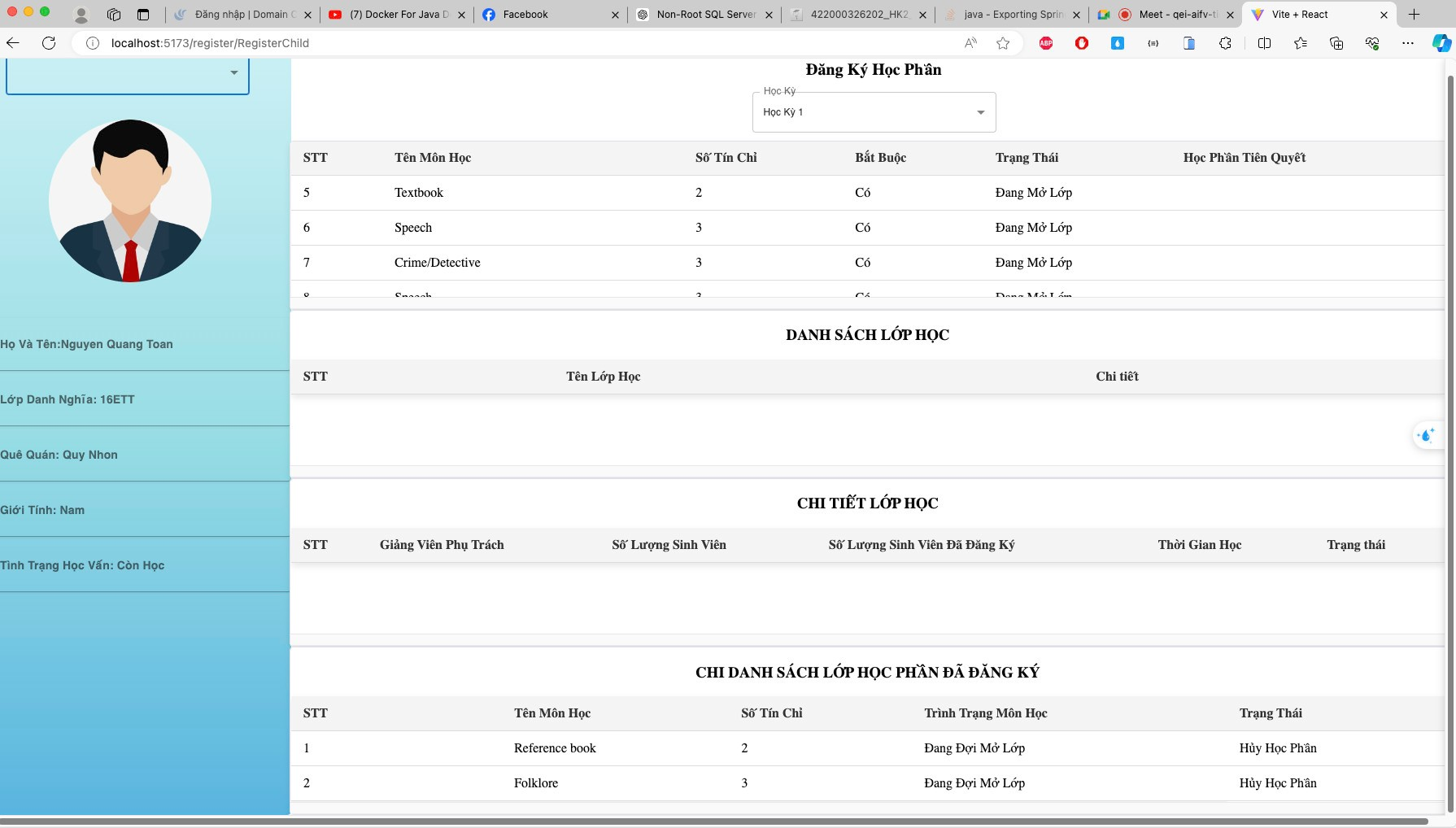
# : HIỆN THỰC

## Giao diện của hệ thống

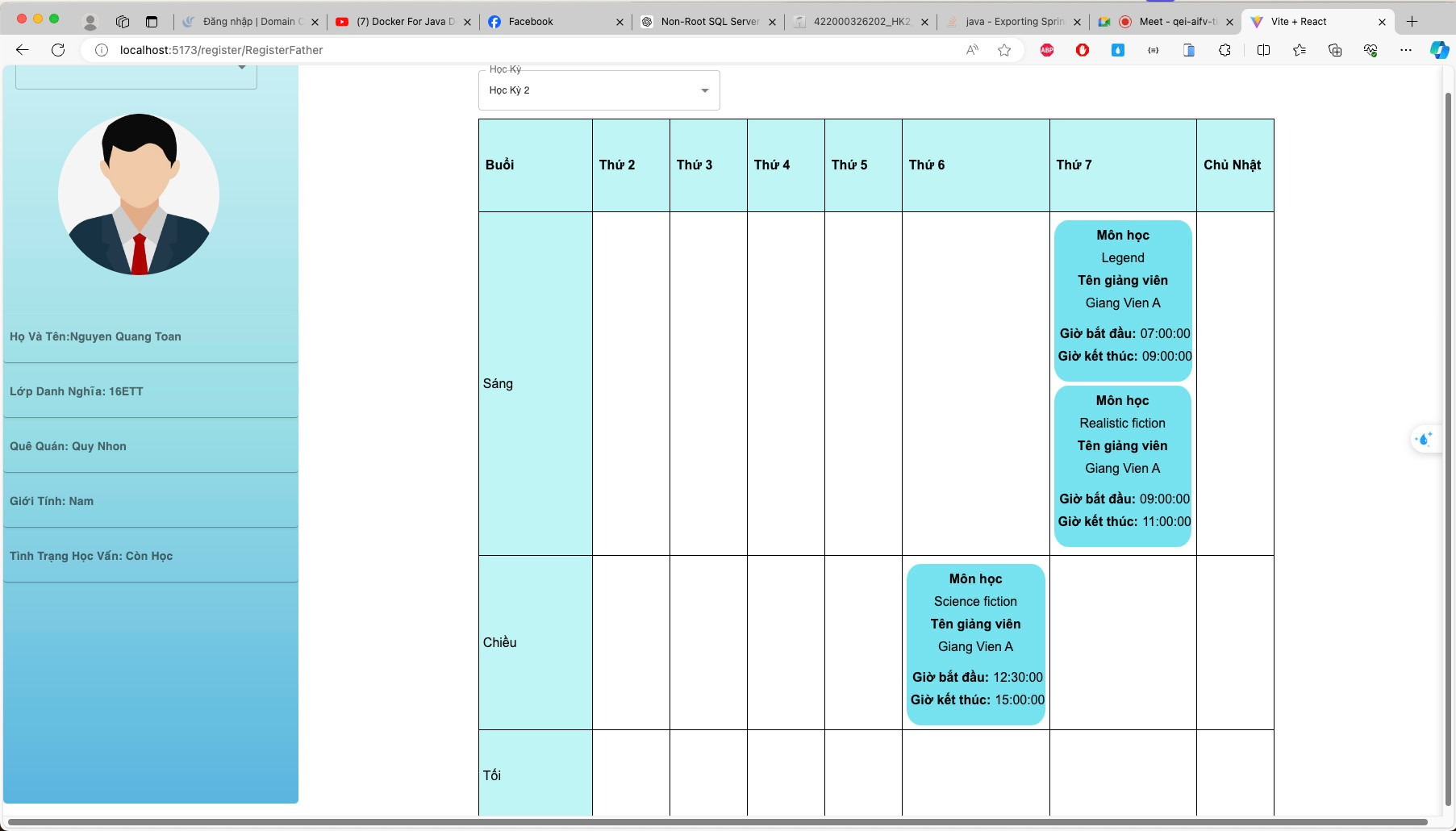
### Giao diện web



## Hình 4.1 giao diện đăng nhập



## Hình 4.2 giao diện xem và đăng ký học phần



## Hình 4.3 giao diện xem thời khóa biểu

## Hình 4.5 giao diện khi hủy đăng ký học phần thành công

## 

## Hình 4.6 giao diện khi đăng ký học phần thành công

# : KẾT LUẬN

## Kết quả đạt được

## Hạn chế của đồ án

* Hệ thống không quản lý các hoạt động học tập khác của sinh viên như: thi cử, đánh giá, cấp bằng...
* Hệ thống không quản lý các hoạt động quản lý khác của nhà trường như: quản lý tài chính, quản lý cán bộ...

## Hướng phát triển

* **Tích hợp với các hệ thống khác**: Hệ thống có thể được tích hợp với các hệ thống khác trong nhà trường như hệ thống quản lý tài chính, hệ thống thư viện,...
* **Hỗ trợ học tập trực tuyến**: Hệ thống có thể hỗ trợ học tập trực tuyến bằng cách cung cấp các bài giảng trực tuyến, bài tập trực tuyến, ...
* **Phát triển trên thiết bị di động**: Hệ thống có thể được phát triển trên các thiết bị di động để sinh viên có thể dễ dàng truy cập thông tin và thực hiện các thao tác trên hệ thống.
* **Sử dụng trí tuệ nhân tạo**: Hệ thống có thể sử dụng trí tuệ nhân tạo để đề xuất các môn học phù hợp với sinh viên, dự báo kết quả học tập của sinh viên,...

# TÀI LIỆU THAM KHẢO

**Các tài liệu Tiếng Anh**

1. Richards, Mark, and Neal Ford. Fundamentals of software architecture: an engineering approach. O'Reilly Media, 2020.

**Các tài liệu từ Internet**

1. <https://viblo.asia/p/microservice-architecture-la-gi-tai-sao-bay-gio-chung-ta-can-den-no-gGJ599gJ5X2>, [Hồ Việt Tuấn](https://viblo.asia/u/tuanhv123), 2019.
2. <https://viblo.asia/p/gioi-thieu-ve-microservices-Ljy5Vz895ra>, Dang Van, 2019
3. <https://developer.ibm.com/articles/why-should-we-use-microservices-and-containers/?fbclid=IwAR2EeHXvzYkVU2KUCNpZcBk57CelJQwEbXQxhNu0Jyh_MnCq4fVyVkK6T8M>, [Anthony Amanse](https://developer.ibm.com/profiles/anthony.amanse), 2020
4. https://www.process.st/how-to/authenticate-microsoft-services/