ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN
CHUYÊN NGÀNH: KỸ THUẬT MÁY TÍNH

ĐỀ TÀI:

Xây dựng kiến trúc Microservice đăng nhập 1 cửa cho các hệ thống phần mềm trường ĐHBK. Tích hợp ứng dụng đăng ký tín chỉ.

Người hướng dẫn: **TS. Đặng Thiên Bình**

Sinh viên thực hiện: **Trần Hữu Tuân**

Số thẻ sinh viên: **102210239**

Lớp**: 21TCLC_DT3**

Đà Nẵng, 06/2025

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN
CHUYÊN NGÀNH: KỸ THUẬT MÁY TÍNH

ĐỀ TÀI:

Xây dựng kiến trúc Microservice đăng nhập 1 cửa cho các hệ thống phần mềm trường ĐHBK. Tích hợp ứng dụng đăng ký tín chỉ.

Người hướng dẫn: **TS. Đặng Thiên Bình**

Sinh viên thực hiện: **Trần Hữu Tuân**

Số thẻ sinh viên: **102210239**

Lớp**: 21TCLC_DT3**

Đà Nẵng, 06/2025

ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Trần Hữu Tuân Số thẻ sinh viên: 102210239

Lớp: 21TCLC_DT3 Khoa: Công nghệ Thông tin Ngành: Kỹ thuật máy tính

1. ***Tên đề tài đồ án:*** **Xây dựng kiến trúc Microservice đăng nhập 1 cửa cho các hệ thống phần mềm trường ĐHBK: Tích hợp ứng dụng đăng ký tín chỉ.**
2. ***Đề tài thuộc diện:*** ☐ ***Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện***
3. ***Các số liệu và dữ liệu ban đầu:*** Không có
4. ***Nội dung các phần thuyết minh và tính toán:***

Mở đầu: Giới thiệu đề tài: Lý do chọn đề tài, bối cảnh thực tế. Mục tiêu và phạm vi nghiên cứu.

Chương 1: Cơ sở lý thuyết: Trình bày những cơ sở lý thuyết được áp dụng trong đề tài.

Chương 2: Phân tích và thiết kế hệ thống:

Khảo sát hệ thống hiện có và các vấn đề gặp phải.

Đề xuất mô hình kiến trúc Microservices cho hệ thống đăng nhập một cửa (SSO).

Thiết kế hệ thống đăng ký tín chỉ.

Chương 3: Triển khai và vận hành hệ thống.

Chương 4: Kiểm thử và đánh giá hệ thống.

Kết luận: Tổng kết những kết quả đạt được. Đề xuất hướng phát triển trong tương lai, bao gồm khả năng mở rộng hệ thống và tích hợp thêm các dịch vụ khác.

5. ***Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ):***
6. ***Họ và tên người hướng dẫn: TS. Đặng Thiên Bình***
7. ***Ngày giao nhiệm vụ đồ án: ……../……./2025***
8. ***Ngày hoàn thành đồ án: ……../……./2025.***

Đà Nẵng, ngày tháng 06 năm 2025

Trưởng Bộ môn …….. **Người hướng dẫn**

TS. Đặng Thiên Bình

LỜI NÓI ĐẦU

Trong quá trình học tập tại trường cũng như thời gian làm đồ án tốt, em luôn nhận được sự hỗ trợ nhiệt tình từ phía nhà trường, sự giảng dạy, hướng dẫn, giúp đỡ tận tình của các thầy cô và các bạn trong Khoa Công nghệ Thông tin. Em xin gửi lời cảm ơn chân thành và sâu sắc đến các thầy, các cô đã truyền đạt cho em kiến thức cần thiết và bổ ích để em hoàn thiện được đồ án lần này.

Đặc biệt, em xin gửi lời cảm ơn chân thành đến TS. Đặng Thiên Bình, người đã hướng dẫn, góp ý, hỗ trợ tận tình cho đồ án tốt nghiệp này. Bên cạnh đó, em gửi lời cảm ơn đến Phòng Đào tạo đã cung cấp dữ liệu, đưa ra những lời góp ý, bổ sung cho đồ án tốt nghiệp. Không thể thiếu Tổ Công nghệ thông tin đã giúp xây dựng hệ thống các phần mềm trường Đại học Bách Khoa Đà Nẵng.

Trong quá trình học tập suốt quá trình học tập, làm đồ án tốt nghiệp, không thể tránh xảy ra các thiếu xót, em rất mong nhận được sự góp ý quý báu của các thầy cô, bạn bè để đồ án tốt nghiệp của em lần này hoàn thiện và đạt kết quả tốt hơn.

Một lần nữa em xin chân thành cảm ơn!

Đà Nẵng, ngày … tháng 06 năm 2025

Sinh viên thực hiện

Trần Hữu Tuân

CAM ĐOAN

Tôi xin cam đoan:

1. **Những nội dung trong đồ án này là do tôi thực hiện dưới sự hướng dẫn trực tiếp của TS. Đặng Thiên Bình.**
2. **Mọi tham khảo dùng trong đồ án đều được trích dẫn rõ ràng tên tác giả, tên công trình, thời gian, địa điểm công bố.**
3. **Nếu có những sao chép không hợp lệ, vi phạm, tôi xin chịu hoàn toàn trách nhiệm.**

Đà Nẵng, ngày … tháng 06 năm 2025

Sinh viên thực hiện

Trần Hữu Tuân

MỤC LỤC

DANH SÁCH HÌNH ẢNH

DANH SÁCH BẢNG

DANH SÁCH TỪ VIẾT TẮT

Từ viết tắt	Diễn giải
API	Application Programming Interface.
RAM	Random Access Memory
DB	Database
CPU	Central Processing Unit
SQL	Structured Query Language

MỞ ĐẦU

1. Giới thiệu đề tài

Tên đề tài: Xây dựng kiến trúc Microservice đăng nhập 1 cửa cho các hệ thống phần mềm trường ĐHBK: Tích hợp ứng dụng đăng ký tín chỉ.

Hiện nay, các hệ thống phần mềm tại Trường Đại học Bách khoa Đà Nẵng chưa có sự thống nhất về quy trình xác thực người dùng. Mỗi hệ thống triển khai một phương thức đăng nhập riêng biệt, dẫn đến tình trạng sinh viên, giảng viên và cán bộ phải ghi nhớ nhiều tài khoản và mật khẩu khác nhau. Điều này không chỉ gây bất tiện mà còn tiềm ẩn nhiều rủi ro về bảo mật thông tin.

Bên cạnh đó, Trường Đại học Bách khoa Đà Nẵng đang trong quá trình triển khai các ứng dụng di động nhằm hỗ trợ quản lý đào tạo và hành chính. Việc có một hệ thống đăng nhập thống nhất sẽ giúp cải thiện trải nghiệm người dùng, tăng cường bảo mật và tối ưu hóa quy trình xác thực.

Hiện nay, hệ thống đăng ký tín chỉ tại Trường Đại học Bách khoa Đà Nẵng thường xuyên gặp phải tình trạng quá tải và gián đoạn dịch vụ khi lượng sinh viên truy cập tăng cao, đặc biệt trong các đợt đăng ký học phần mỗi học kỳ. Mặc dù nhà trường đã áp dụng phương pháp phân chia thời gian đăng ký theo từng khoa nhằm giảm tải cho hệ thống, nhưng tình trạng sập trang, tốc độ xử lý chậm và lỗi hệ thống vẫn diễn ra, gây ảnh hưởng lớn đến trải nghiệm của sinh viên.

2. Mục tiêu, nhiệm vụ

Mục tiêu

Thiết kế và xây dựng một hệ thống đăng nhập một cửa (Single Sign-On - SSO) dựa trên kiến trúc Microservices để thống nhất quá trình xác thực người dùng trên các hệ thống phần mềm của nhà trường.

Cung cấp một giải pháp bảo mật, linh hoạt và dễ dàng mở rộng trong tương lai.

Hỗ trợ tích hợp với các hệ thống hiện có cũng như các ứng dụng di động mới.

Ứng dụng di động cho phép sinh viên đăng ký tín chỉ. Tối ưu hệ thống đăng ký tín chỉ và giảm trình trạng quá tải và gián đoạn dịch vụ khi lượng sinh viên truy cập tăng cao.

Nhiệm vụ

a) Nghiên cứu về hệ thống xác thực tập trung:

+ Tìm hiểu các phương thức xác thực phổ biến như OAuth 2.0, OpenID Connect, SAML.

+ Phân tích ưu và nhược điểm của từng phương thức để lựa chọn giải pháp phù hợp.

b) Thiết kế kiến trúc hệ thống:

+ Đề xuất mô hình kiến trúc Microservices cho hệ thống xác thực tập trung.

+ Xây dựng luồng xử lý đăng nhập, cấp phát và xác thực phiên làm việc.

+ Thiết kế cơ sở dữ liệu để lưu trữ thông tin người dùng và phiên đăng nhập.

c) Xây dựng hệ thống đăng ký tín chỉ:

+ Quản lý danh sách học phần

+ Quản lý lịch học và lớp học

+ Xây dựng quy trình đăng ký tín chỉ

+ Xử lý xung đột lịch học

+ Quản lý giới hạn tín chỉ

+ Hỗ trợ ưu tiên và xét duyệt đăng ký

+ Hệ thống theo dõi và thông báo

+ Quản lý danh sách sinh viên theo lớp học phần

d) Tìm hiểu nguyên nhân dẫn đến tình trạng quá tải và gián đoạn dịch vụ khi lượng sinh viên truy cập tăng cao.

e) Tối ưu hóa hiệu suất hệ thống đăng ký tín chỉ:

+ Sử dụng caching (Redis, Memcached) để lưu trữ tạm thời các dữ liệu truy cập nhiều như danh sách học phần, lịch học nhằm giảm truy vấn trực tiếp vào cơ sở dữ liệu.

+ Sử dụng cơ chế hàng đợi (Message Queue - RabbitMQ, Kafka) để xử lý yêu cầu đăng ký tín chỉ theo thứ tự, tránh tình trạng xung đột khi nhiều sinh viên đăng ký cùng lúc.

+ Sử dụng sharding hoặc replication để phân tán tải khi có lượng lớn sinh viên truy vấn đồng thời.

+ Áp dụng cơ chế batch processing để xử lý nhóm yêu cầu cùng lúc thay vì từng yêu cầu riêng lẻ.

+ Sử dụng Rate Limiting để giới hạn số lượng yêu cầu từ một người dùng trong một khoảng thời gian nhằm ngăn chặn spam hoặc tấn công DdoS

3. Công nghệ sử dụng

Ngôn ngữ lập trình sử dụng:

+ C#

+ Dart

+ Bash (script)

+ Dockerfile Phát triển giao diện người dùng:

+ Flutter

Công nghệ triển khai và máy chủ:

+ Nginx (Reverse Proxy & Web Server)

+ YARP (Reverse Proxy trong .NET)

+ Docker (Containerization) Cơ sở dữ liệu & Lưu trữ dữ liệu:

+ MySQL Sharding (Phân mảnh dữ liệu)

+ Redis (Cache & Message Broker)

Hệ thống xử lý dữ liệu & Giao tiếp giữa các dịch vụ:

+ Kafka (Message Queue & Event Streaming)

+ Batch Processing (Xử lý hàng loạt)

Công nghệ tối ưu hóa hiệu suất & An toàn hệ thống

+ Rate limiting

4. Kết quả dự kiến đạt được

Cải thiện hiệu suất và độ ổn định của hệ thống

+ Giảm tình trạng quá tải và sập hệ thống khi có lượng lớn sinh viên truy cập đồng thời.

+ Tối ưu cơ sở dữ liệu, caching, xử lý hàng đợi giúp tăng tốc độ xử lý yêu cầu.

+ Cải thiện thời gian phản hồi khi sinh viên thực hiện đăng ký tín chỉ.

Hỗ trợ quy trình đăng ký tín chỉ hiệu quả

+ Cho phép sinh viên tra cứu, đăng ký, hủy hoặc thay đổi lớp học phần một cách thuận tiện.

+ Xử lý tự động xung đột lịch học, giới hạn tín chỉ, ưu tiên đăng ký theo điều kiện của sinh viên.

+ Tích hợp thông báo real-time giúp sinh viên nhận được cập nhật về trạng thái đăng ký.

Tăng cường bảo mật hệ thống

+ Áp dụng **các cơ chế bảo mật** như xác thực an toàn, mã hóa dữ liệu, ngăn chặn tấn công DDoS.

+ Giới hạn số lượng yêu cầu từ một người dùng trong một khoảng thời gian để ngăn spam.

+ Đảm bảo **an toàn dữ liệu** và quyền riêng tư cho sinh viên, giảng viên và nhà trường.

Đánh giá và đề xuất triển khai chính thức

+ Kiểm thử hệ thống với các tình huống thực tế để đánh giá hiệu quả.

+ So sánh hiệu suất với hệ thống cũ, đưa ra các đề xuất cải thiện.

+ Xây dựng tài liệu hướng dẫn sử dụng cho sinh viên, giảng viên và quản trị viên.

5. Nội dung đồ án

Mở đầu

Chương 1 - Cơ sở lý thuyết

Chương 2 - Phân tích và thiết kế hệ thống

Chương 3 - Triển khai và vận hành hệ thống

Chương 4 - Kết luận và hướng phát triển

Tài liệu tham khảo

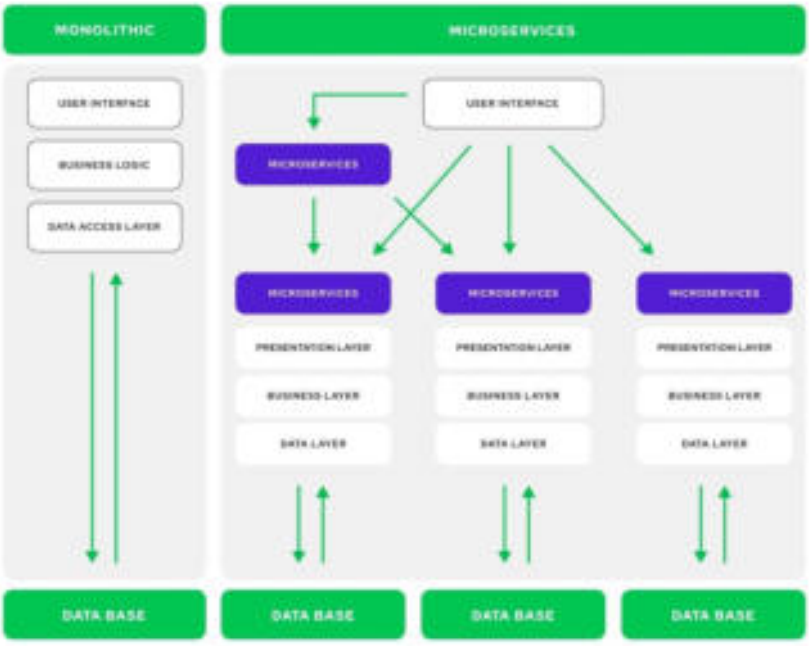
CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

1.1. Tổng quan về kiến trúc Microservice

1.1.1. Kiến trúc Microservice

a. Khái niệm

Kiến trúc Microservices[1] là một phương pháp phát triển phần mềm, trong đó một ứng dụng lớn được cấu thành từ nhiều dịch vụ nhỏ, độc lập và có khả năng triển khai riêng lẻ. Mỗi dịch vụ chạy trong một quy trình riêng và giao tiếp với các dịch vụ khác thông qua các cơ chế gọn nhẹ, thường là qua giao diện lập trình ứng dụng (API) dựa trên HTTP/REST.



Hình 1.1. Kiến trúc Monolithic và Microservices

Các đặc điểm lý thuyết cốt lõi của kiến trúc này bao gồm:

- **Tập trung vào nghiệp vụ (Business-Oriented):** Mỗi microservice được thiết kế để giải quyết một chức năng nghiệp vụ cụ thể của hệ thống (ví dụ: quản lý người dùng, xử lý thanh toán, gửi thông báo).
- **Tính tự trị (Autonomous):** Các đội phát triển có thể tự do xây dựng, triển khai và mở rộng dịch vụ của mình mà không cần phải phối hợp chặt chẽ với các đội khác.
- **Giao tiếp qua "ống dẫn thông minh" (Smart endpoints and dumb pipes):** Các dịch vụ tự xử lý logic nghiệp vụ và giao tiếp với nhau qua các

kênh đơn giản như API REST hoặc hàng đợi tin nhắn (Message Queue), thay vì dựa vào một lớp trung gian phức tạp để điều phối.

- **Phi tập trung hóa (Decentralized):** Kiến trúc này khuyến khích việc phi tập trung hóa cả về dữ liệu và quản trị. Mỗi dịch vụ có thể có cơ sở dữ liệu riêng và được phát triển bằng những công nghệ, ngôn ngữ phù hợp nhất với nó.

b. Phân tích ưu điểm và nhược điểm

Việc áp dụng kiến trúc Microservices[1] mang lại nhiều lợi ích đáng kể:

- **Khả năng mở rộng linh hoạt (Enhanced Scalability):** Thay vì phải mở rộng toàn bộ ứng dụng như kiến trúc nguyên khối, bạn có thể chỉ mở rộng những dịch vụ đang chịu tải cao. Điều này giúp tối ưu hóa việc sử dụng tài nguyên và chi phí.
- **Tăng tốc độ phát triển (Increased Agility):** Các đội phát triển nhỏ, tập trung có thể làm việc song song trên các dịch vụ khác nhau. Quy trình phát triển, kiểm thử và triển khai diễn ra nhanh hơn vì phạm vi tác động nhỏ hơn.
- **Tự do lựa chọn công nghệ (Technology Freedom):** Mỗi dịch vụ có thể được xây dựng bằng một ngăn xếp công nghệ (technology stack) khác nhau. Ví dụ, một dịch vụ đòi hỏi hiệu năng cao có thể được viết bằng Go hoặc Rust, trong khi một dịch vụ khác có thể sử dụng .NET hoặc Python.
- **Tăng cường khả năng phục hồi (Improved Resilience):** Nếu một dịch vụ gặp lỗi, nó sẽ không làm sập toàn bộ hệ thống. Các dịch vụ khác vẫn có thể hoạt động bình thường, giúp hệ thống có khả năng chịu lỗi cao hơn.
- **Dễ dàng bảo trì và hiểu (Easier Maintenance):** Mỗi dịch vụ có một codebase nhỏ hơn, dễ hiểu và dễ quản lý hơn so với một ứng dụng nguyên khối khổng lồ.
- **Triển khai độc lập (Independent Deployment):** Có thể cập nhật và triển khai từng dịch vụ riêng lẻ mà không ảnh hưởng đến các dịch vụ khác, tạo điều kiện cho quy trình Tích hợp liên tục và Triển khai liên tục (CI/CD) hiệu quả hơn.

Bên cạnh những ưu điểm, kiến trúc Microservices cũng đi kèm với nhiều thách thức:

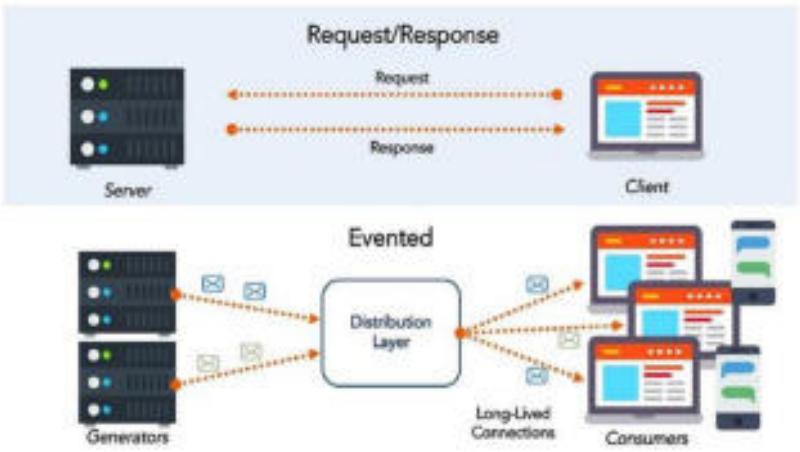
- **Sự phức tạp trong vận hành (Operational Complexity):** Việc quản lý, triển khai và giám sát một hệ thống bao gồm hàng chục hoặc hàng trăm dịch vụ phức tạp hơn rất nhiều so với một ứng dụng nguyên khối. Điều này đòi hỏi các kỹ năng về DevOps, containerization (như Docker), và các công cụ điều phối (như Kubernetes).

- **Tính nhất quán dữ liệu (Data Consistency):** Vì mỗi dịch vụ có thể có cơ sở dữ liệu riêng, việc đảm bảo tính nhất quán dữ liệu trên toàn hệ thống trở nên khó khăn. Các giao dịch (transactions) trải dài trên nhiều dịch vụ cần được xử lý cẩn thận thông qua các mẫu thiết kế như "Saga" để đạt được "nhất quán cuối cùng" (eventual consistency).
- **Độ trễ mạng và giao tiếp (Network Latency and Communication):** Giao tiếp giữa các dịch vụ thông qua mạng luôn chậm hơn so với giao tiếp trong cùng một quy trình (in-process calls). Điều này có thể ảnh hưởng đến hiệu năng toàn hệ thống nếu không được thiết kế tối ưu.
- **Kiểm thử phức tạp (Complex Testing):** Việc kiểm thử tích hợp (integration testing) trở nên phức tạp hơn vì cần phải đảm bảo sự tương tác chính xác giữa nhiều dịch vụ đang chạy.
- **Yêu cầu kỹ năng cao:** Đội ngũ phát triển cần có kiến thức sâu về lập trình phân tán, các mẫu thiết kế hệ thống, cũng như các công nghệ hỗ trợ để triển khai và vận hành hiệu quả.

c. Các thành phần và phương án hỗ trợ

Để triển khai thành công một hệ thống Microservices[1], cần áp dụng các mẫu thiết kế và công cụ chuyên dụng:

- **Containerization và Điều phối (Containerization and Orchestration):**
  - **Docker:** Là công nghệ phổ biến nhất để đóng gói một dịch vụ và tất cả các phụ thuộc của nó vào một "container", đảm bảo môi trường chạy nhất quán ở mọi nơi.
  - **Kubernetes (K8s):** Là nền tảng điều phối container hàng đầu, giúp tự động hóa việc triển khai, mở rộng và quản lý các ứng dụng container hóa.
- **Cổng API (API Gateway):**
  - Là một máy chủ đóng vai trò là điểm vào duy nhất cho tất cả các yêu cầu từ client. Nó chịu trách nhiệm định tuyến yêu cầu đến các dịch vụ tương ứng, xác thực người dùng, giới hạn tốc độ truy cập (rate limiting), và ghi log.
  - *Ví dụ:* YARP[2], Kong, Ocelot, Spring Cloud Gateway.
- **Giao tiếp giữa các dịch vụ (Inter-service Communication):**
  - **Đồng bộ (Synchronous):** Sử dụng API REST hoặc gRPC khi client cần nhận phản hồi ngay lập tức.



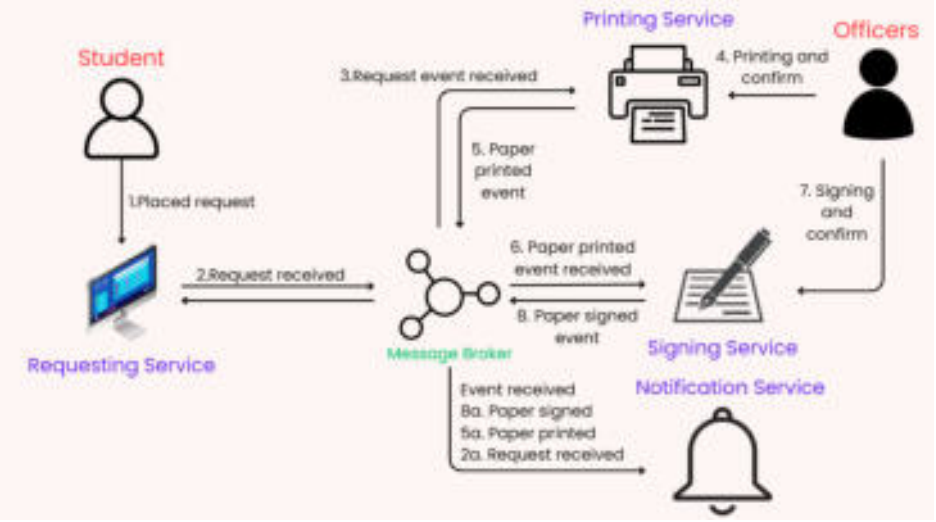
Hình 1.2. Sử dụng hàng đợi tin nhắn

- **Bất đồng bộ (Asynchronous):** Sử dụng hàng đợi tin nhắn (Message Broker) như **RabbitMQ**, **Apache Kafka**, hoặc **Google Pub/Sub** để tách rời các dịch vụ. Dịch vụ này chỉ cần gửi một sự kiện mà không cần quan tâm dịch vụ nào sẽ xử lý nó.

- **Phát hiện Dịch vụ (Service Discovery):**
  - Trong một môi trường động, địa chỉ IP của các dịch vụ có thể thay đổi liên tục. Service Discovery giúp các dịch vụ tìm thấy và giao tiếp với nhau.
  - *Ví dụ:* Eureka, Consul.
- **Giám sát và Ghi log tập trung (Centralized Monitoring and Logging):**
  - Vì hệ thống bị phân tán, việc theo dõi và gỡ lỗi trở nên khó khăn. Cần có một hệ thống ghi log tập trung để thu thập log từ tất cả các dịch vụ vào một nơi.
  - **Logging:** Ngăn xếp ELK (Elasticsearch, Logstash, Kibana).
  - **Monitoring & Alerting:** Prometheus, Grafana.
  - **Distributed Tracing:** Jaeger, Zipkin, để theo dõi một yêu cầu khi nó đi qua nhiều dịch vụ.

1.1.2. Ví dụ minh họa: Phân tích quy trình nghiệp vụ

Hãy xem xét một ví dụ thực tế: quy trình yêu cầu cấp giấy chứng nhận sinh viên. Quy trình này thường bao gồm các bước: sinh viên truy cập website để đăng ký, chờ cán bộ xác nhận, kiểm tra thông báo và đến văn phòng để nhận giấy.



Hình 1.3. Áp dụng Microservices và quy trình yêu cầu cấp giấy xác nhận sinh viên

Trong kiến trúc Microservices, quy trình này có thể được chia thành bốn dịch vụ riêng biệt: **Dịch vụ Yêu cầu (Requesting)**, **Dịch vụ In ấn (Printing)**, **Dịch vụ Xác nhận ký (SignConfirmation)**, và **Dịch vụ Thông báo (Notification)**. Luồng hoạt động sẽ diễn ra như sau:

1. Khi một sinh viên gửi yêu cầu xin giấy chứng nhận trên website, **Dịch vụ Yêu cầu** (Requesting service) sẽ tiếp nhận yêu cầu, xử lý và gửi một sự kiện (event) đến **Message Broker** (hệ thống trung gian gửi/nhận tin nhắn).
2. **Dịch vụ In ấn** (Printing service) và **Dịch vụ Thông báo** (Notification service), vốn đã đăng ký (subscribed) để lắng nghe các sự kiện từ Dịch vụ Yêu cầu, sẽ lần lượt nhận được sự kiện này. Dịch vụ In ấn bắt đầu quá trình xử lý in ấn, còn Dịch vụ Thông báo sẽ gửi một email xác nhận đã nhận yêu cầu cho sinh viên.
3. Sau khi in xong, **Dịch vụ In ấn** gửi một sự kiện "đã in xong" đến Message Broker.
4. **Dịch vụ Xác nhận ký** (SignConfirmation), vốn đăng ký lắng nghe sự kiện này, sẽ nhận thông tin và xử lý việc ký duyệt. Sau khi hoàn tất, nó tiếp tục gửi một sự kiện mới đến Message Broker.
5. Cuối cùng, **Dịch vụ Thông báo** nhận được sự kiện hoàn tất ký duyệt và gửi một thông báo nữa cho sinh viên với thông tin xác nhận cuối cùng (ví dụ: "Giấy chứng nhận của bạn đã sẵn sàng để nhận tại văn phòng").

Mỗi dịch vụ có thể sử dụng công nghệ khác nhau:

- **Dịch vụ Yêu cầu, Dịch vụ Xác nhận ký:** Có thể sử dụng .NET, Spring... với cơ sở dữ liệu MySQL.

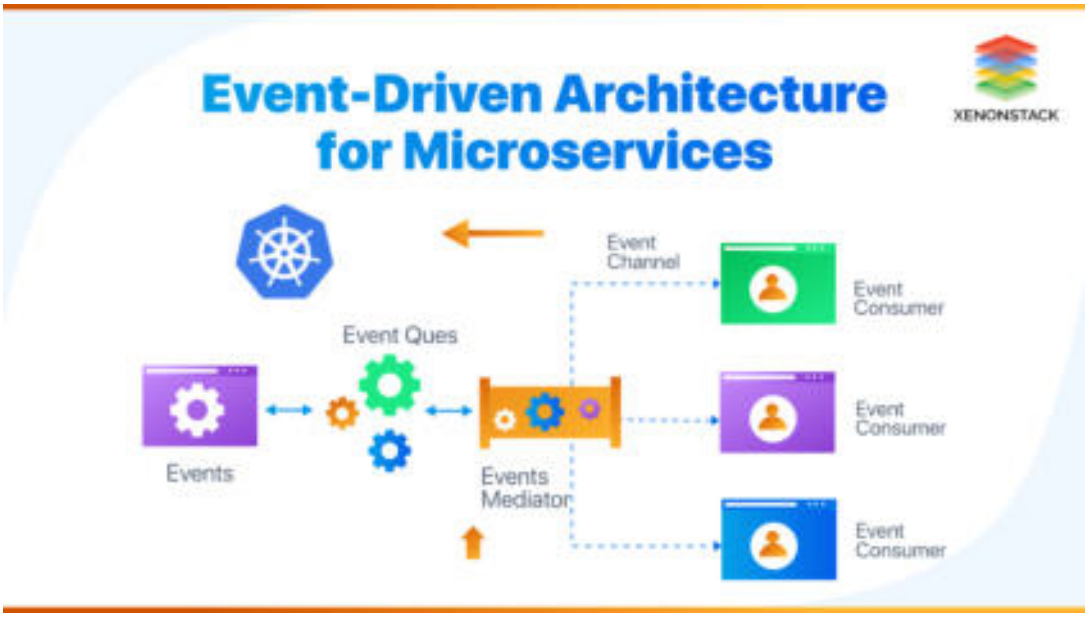
- **Dịch vụ In ấn:** Có thể sử dụng .NET, Spring... với cơ sở dữ liệu NoSQL (để lưu trạng thái in ấn) và Redis (dùng cho caching).
- **Dịch vụ Thông báo:** Có thể sử dụng NodeJS + Express... và kết nối với dịch vụ của bên thứ ba như Firebase.

Lợi ích của Message Broker trong hệ thống:

- **Giao tiếp bất đồng bộ (Asynchronous Communication):** Các dịch vụ như In ấn, Xác nhận ký và Thông báo không cần phải chờ nhau hoàn thành công việc. Chúng hoạt động độc lập và song song, giúp tăng hiệu suất toàn hệ thống.
- **Tách rời các thành phần (Decoupling):** Các dịch vụ hoạt động độc lập và chỉ giao tiếp với nhau một cách gián tiếp thông qua Message Broker. Điều này làm cho hệ thống dễ dàng bảo trì và mở rộng hơn vì thay đổi ở một dịch vụ không ảnh hưởng trực tiếp đến các dịch vụ khác.

1.2. Kiến trúc hướng sự kiện (EDA) và vai trò của Apache Kafka

1.2.1. Kiến trúc hướng sự kiện (EDA)



Hình 1.4. Kiến trúc hướng sự kiện

Trong khi kiến trúc Microservice định nghĩa *cấu trúc* của hệ thống (chia thành các dịch vụ nhỏ), thì Kiến trúc hướng sự kiện (Event-Driven Architecture - EDA) lại định nghĩa *cách thức giao tiếp* giữa các thành phần đó. Đây là một mẫu hình kiến trúc ngày càng phổ biến, đặc biệt khi kết hợp với Microservices.

a. Định nghĩa và Nguyên lý hoạt động

Kiến trúc hướng sự kiện (EDA)[3] là một mẫu thiết kế phần mềm, trong đó các thành phần của hệ thống giao tiếp với nhau thông qua việc tạo ra và phản ứng lại các

"sự kiện" (events). Một sự kiện là một bản ghi về một thay đổi trạng thái hoặc một hành động quan trọng đã xảy ra trong hệ thống (ví dụ: "Người dùng đã đặt hàng", "Sản phẩm đã được giao", "Thanh toán thành công").

Thay vì một dịch vụ trực tiếp gọi một dịch vụ khác để yêu cầu thực hiện hành động (mô hình Request/Response), trong EDA, một dịch vụ chỉ cần "phát" (publish) một sự kiện ra một kênh trung gian. Các dịch vụ khác quan tâm đến sự kiện đó sẽ "đăng ký" (subscribe) lắng nghe và xử lý khi sự kiện xảy ra.

b. Các thành phần chính

Một hệ thống EDA thường bao gồm ba thành phần cốt lõi:

- **Nhà sản xuất sự kiện (Event Producer):** Là thành phần phát hiện và tạo ra sự kiện. Ví dụ, dịch vụ "Thanh toán" là một Producer khi nó tạo ra sự kiện "Thanh toán thành công".
- **Người tiêu dùng sự kiện (Event Consumer):** Là thành phần lắng nghe và phản ứng lại các sự kiện. Ví dụ, dịch vụ "Giao hàng" và dịch vụ "Thông báo" đều là các Consumer, cùng phản ứng lại sự kiện "Thanh toán thành công" để bắt đầu quy trình giao hàng và gửi email cho khách.
- **Kênh sự kiện (Event Channel / Message Broker):** Là hạ tầng trung gian chịu trách nhiệm nhận sự kiện từ các Producer và chuyển chúng đến các Consumer tương ứng. Đây là xương sống của kiến trúc EDA.

c. Lợi ích của Kiến trúc hướng sự kiện

- **Tách rời cao (High Decoupling):** Producer và Consumer không cần biết về sự tồn tại của nhau. Chúng chỉ cần tương tác với Event Channel. Điều này giúp các dịch vụ hoàn toàn độc lập, dễ dàng thay đổi, nâng cấp hoặc thay thế mà không ảnh hưởng đến phần còn lại của hệ thống.
- **Khả năng mở rộng và hiệu suất cao:** Vì các tác vụ được xử lý bất đồng bộ, hệ thống có thể xử lý đồng thời một lượng lớn sự kiện mà không bị tắc nghẽn. Có thể dễ dàng tăng số lượng Consumer để xử lý công việc song song.
- **Khả năng phục hồi và chống lỗi (Resilience):** Nếu một Consumer bị lỗi (ví dụ: dịch vụ "Thông báo" bị sập), các sự kiện vẫn được lưu trong Event Channel. Khi dịch vụ hoạt động trở lại, nó có thể tiếp tục xử lý các sự kiện đó, đảm bảo không mất mát dữ liệu.
- **Phản ứng theo thời gian thực (Real-time Responsiveness):** EDA rất phù hợp cho các hệ thống cần phản ứng ngay lập tức với các thay đổi, như hệ thống giám sát, phân tích dữ liệu luồng (stream analytics) hoặc IoT.

1.2.2. Apache Kafka

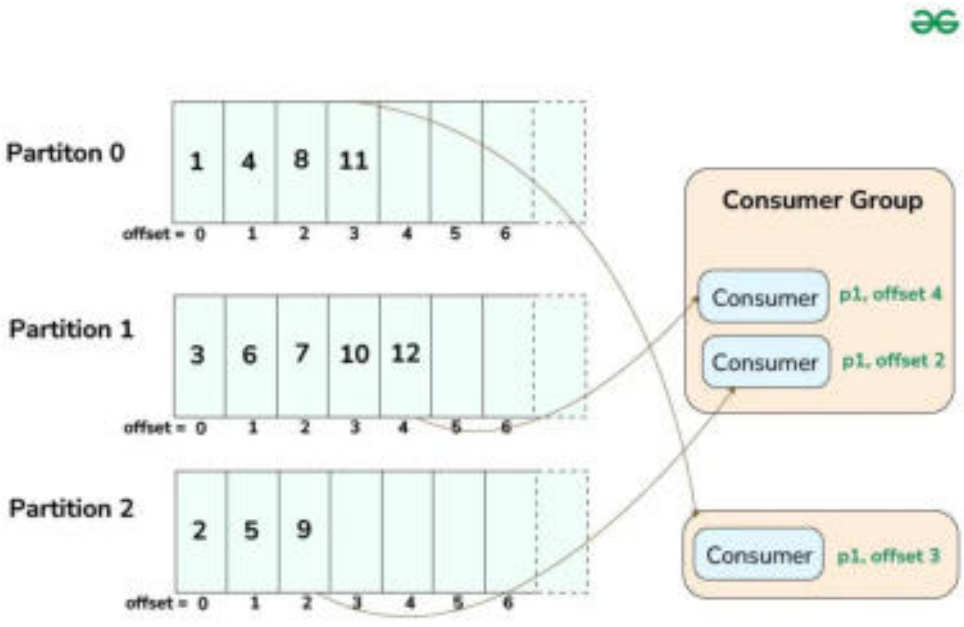
Apache Kafka không chỉ là một Message Broker thông thường, nó đã phát triển thành một nền tảng truyền tải sự kiện (event streaming platform) mạnh mẽ, đóng vai trò trung tâm trong nhiều hệ thống EDA hiện đại.

a. Giới thiệu về Apache Kafka

Apache Kafka [3] là một hệ thống mã nguồn mở, phân tán, được thiết kế để xử lý một lượng cực lớn dữ liệu theo thời gian thực. Ban đầu được phát triển tại LinkedIn, Kafka có khả năng xử lý hàng nghìn tỷ sự kiện mỗi ngày với các đặc tính nổi bật như:

- **Thông lượng cực cao (High Throughput):** Có khả năng xử lý hàng triệu tin nhắn mỗi giây.
- **Độ trễ thấp (Low Latency):** Đảm bảo tin nhắn được truyền đi và nhận lại trong vài mili giây.
- **Khả năng chịu lỗi và độ bền cao (Fault-Tolerant and Durable):** Dữ liệu được sao chép (replicated) trên nhiều máy chủ (broker), đảm bảo không bị mất ngay cả khi một máy chủ gặp sự cố.
- **Khả năng mở rộng vượt trội (Highly Scalable):** Có thể dễ dàng mở rộng cụm (cluster) Kafka bằng cách thêm các broker mới.

b. Các khái niệm cốt lõi của Kafka



Hình 1.5. Sự tương tác trong kiến trúc Kafka Architecture

- **Event (Sự kiện):** Đơn vị dữ liệu cơ bản trong Kafka, chứa thông tin về một sự kiện đã xảy ra.

- **Topic:** Tương tự như một kênh hoặc một danh mục, là nơi các Producer gửi sự kiện đến và các Consumer đọc sự kiện từ đó. Ví dụ: order-created, payment-processed.
- **Partition (Phân vùng):** Mỗi Topic được chia thành nhiều Partition. Dữ liệu trong một partition được sắp xếp và bất biến. Việc chia partition cho phép xử lý dữ liệu song song, là chìa khóa cho khả năng mở rộng của Kafka.
- **Producer:** Ứng dụng gửi (publish) các sự kiện vào một hoặc nhiều Topic.
- **Consumer:** Ứng dụng đọc (subscribe) các sự kiện từ một hoặc nhiều Topic.
- **Consumer Group:** Một nhóm các Consumer cùng làm việc để tiêu thụ dữ liệu từ một Topic. Mỗi partition chỉ được tiêu thụ bởi một Consumer duy nhất trong một group, cho phép xử lý song song một cách hiệu quả.
- **Broker:** Một máy chủ Kafka. Một cụm Kafka (Kafka Cluster) bao gồm nhiều broker làm việc cùng nhau.

Apache Kafka [3] không chỉ là một lựa chọn, mà thường được coi là "hệ thần kinh trung ương" (central nervous system) cho các hệ thống Microservice hiện đại áp dụng EDA. Vai trò của nó thể hiện qua các điểm sau:

1. **Xương sống Giao tiếp (Communication Backbone):** Kafka cung cấp một kênh giao tiếp tin cậy, hiệu suất cao và bất đồng bộ, giúp tách rời hoàn toàn các microservice. Một dịch vụ chỉ cần "nói" với Kafka, và Kafka sẽ đảm bảo thông điệp được chuyển đến tất cả các bên quan tâm.
2. **Lưu trữ sự kiện bền bỉ (Durable Event Storage):** Đây là một trong những điểm khác biệt lớn nhất của Kafka so với các message broker truyền thống. Kafka không chỉ giao tin nhắn mà còn **lưu trữ chúng một cách bền bỉ** trong một khoảng thời gian có thể cấu hình. Điều này mang lại khả năng:
    - **"Phát lại" (Replay):** Nếu một dịch vụ bị lỗi, sau khi sửa xong nó có thể đọc lại các sự kiện từ quá khứ để khôi phục trạng thái.
    - **Dễ dàng thêm Consumer mới:** Một dịch vụ phân tích mới có thể được thêm vào hệ thống và xử lý toàn bộ lịch sử dữ liệu của một Topic.
    - **Kiểm tra và gỡ lỗi (Auditing and Debugging):** Toàn bộ luồng sự kiện được lưu lại, giúp việc truy vết và gỡ lỗi dễ dàng hơn.
3. **Nền tảng cho Xử lý Dữ liệu Luồng (Stream Processing):** Với các thư viện như Kafka Streams và ksqlDB, Kafka cho phép các ứng dụng không chỉ nhận sự kiện mà còn có thể **xử lý, tổng hợp và biến đổi dữ liệu ngay trên luồng** (in-flight), tạo ra các luồng dữ liệu mới. Điều này mở ra các ứng dụng như phát hiện gian lận, phân tích hành vi người dùng theo thời gian thực...

Tóm lại, trong khi kiến trúc Microservice giúp chia nhỏ ứng dụng, thì EDA với Apache Kafka cung cấp một phương tiện mạnh mẽ để các mảnh ghép đó giao tiếp với nhau một cách linh hoạt, đáng tin cậy và có khả năng mở rộng gần như vô hạn.

1.3. Kỹ thuật Caching và tối ưu hiệu năng truy xuất dữ liệu với Redis

1.3.1. Kỹ thuật Caching

Khi một hệ thống Microservices phát triển, lượng truy cập đến các dịch vụ và cơ sở dữ liệu sẽ tăng lên, dẫn đến nguy cơ quá tải và giảm hiệu năng. Caching[4] (tạo bộ nhớ đệm) là một trong những kỹ thuật hiệu quả nhất để giải quyết vấn đề này bằng cách giảm thiểu số lần truy xuất đến các tài nguyên chậm như cơ sở dữ liệu. Trong số các công cụ caching, Redis nổi lên như một lựa chọn hàng đầu.

a. Định nghĩa và Mục đích

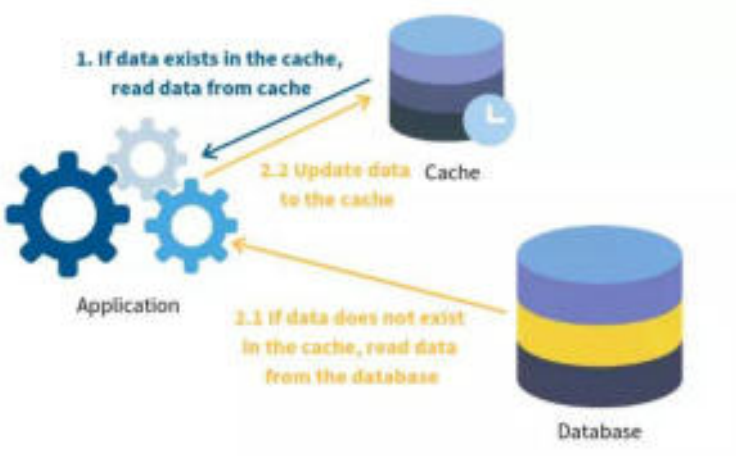
Caching [4] là kỹ thuật lưu trữ tạm thời các dữ liệu thường xuyên được truy cập vào một lớp lưu trữ có tốc độ cao (bộ nhớ đệm), gần với ứng dụng hơn. Thay vì phải thực hiện các thao tác tốn kém (như truy vấn cơ sở dữ liệu, gọi API của một dịch vụ khác) mỗi khi có yêu cầu, ứng dụng sẽ kiểm tra trong bộ nhớ đệm trước. Nếu dữ liệu tồn tại (cache hit), nó sẽ được trả về ngay lập tức. Nếu không (cache miss), ứng dụng mới truy xuất từ nguồn gốc, sau đó lưu một bản sao vào bộ nhớ đệm để sử dụng cho các lần sau.

Mục đích chính của caching:

- **Giảm độ trễ (Reduce Latency):** Truy cập dữ liệu từ bộ nhớ đệm (thường là trong RAM) nhanh hơn hàng nghìn lần so với từ đĩa cứng của cơ sở dữ liệu.
- **Giảm tải cho hệ thống nguồn (Reduce Backend Load):** Giảm số lượng truy vấn đến cơ sở dữ liệu hoặc các dịch vụ khác, giúp chúng hoạt động ổn định hơn và tránh bị quá tải.
- **Tăng thông lượng (Increase Throughput):** Hệ thống có thể phục vụ nhiều yêu cầu hơn trong cùng một khoảng thời gian vì thời gian xử lý mỗi yêu cầu được rút ngắn.

b. Các chiến lược Caching phổ biến

- **Cache-Aside (Lazy Loading):** Đây là chiến lược phổ biến nhất.



Hình 1.6. Chiến lược cache-aside (Lazy loading)

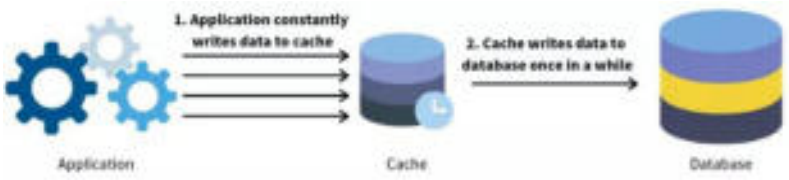
- Ứng dụng kiểm tra xem dữ liệu có trong cache không.
- Nếu có (cache hit), trả về dữ liệu.
- Nếu không (cache miss), ứng dụng truy vấn từ cơ sở dữ liệu, sau đó lưu dữ liệu vào cache rồi mới trả về cho người dùng.
- *Ưu điểm:* Chỉ cache những dữ liệu thực sự được yêu cầu. Hệ thống vẫn hoạt động (dù chậm hơn) nếu cache gặp sự cố.
- *Nhược điểm:* Độ trễ của lần truy cập đầu tiên (cache miss) cao hơn. Dữ liệu trong cache có thể bị cũ (stale) nếu dữ liệu trong DB thay đổi.

- **Write-Through:**

Hình 1.7. Chiến lược Write-Through

- Khi ghi dữ liệu, ứng dụng sẽ ghi đồng thời vào cả cache và cơ sở dữ liệu.
- *Ưu điểm:* Đảm bảo dữ liệu trong cache và DB luôn nhất quán.
- *Nhược điểm:* Thao tác ghi sẽ có độ trễ cao hơn vì phải đợi ghi thành công ở cả hai nơi.

- **Write-Back (Write-Behind):**

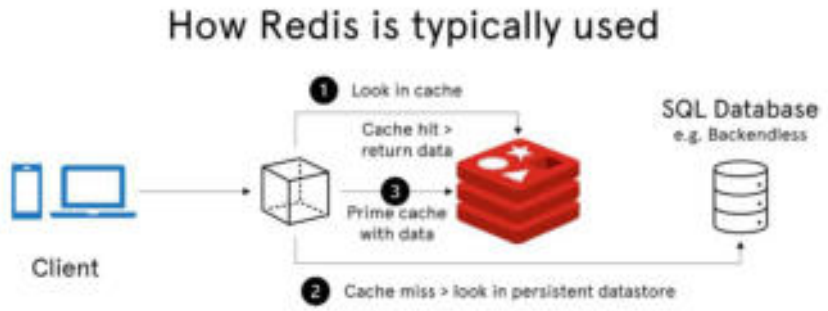


Hình 1.8. Chiến lược Write-back

- Khi ghi dữ liệu, ứng dụng chỉ ghi vào cache và trả về ngay lập tức.
- Thao tác ghi xuống cơ sở dữ liệu sẽ được thực hiện bất đồng bộ sau một khoảng thời gian hoặc theo một lô (batch).
- *Ưu điểm:* Tối ưu hóa hiệu năng ghi, độ trễ rất thấp.
- *Nhược điểm:* Có nguy cơ mất dữ liệu nếu cache bị lỗi trước khi dữ liệu kịp được ghi xuống cơ sở dữ liệu.

Redis (Remote Dictionary Server) là một kho lưu trữ dữ liệu key-value trong bộ nhớ, mã nguồn mở, nổi tiếng về hiệu suất cực cao. Nó không chỉ là một cache đơn giản mà còn hỗ trợ nhiều cấu trúc dữ liệu phức tạp.

1.3.2. Tối ưu hiệu năng truy xuất dữ liệu với Redis



Hình 1.9. Minh họa hệ thống áp dụng Redis

- **Tốc độ cực nhanh:** Vì hoạt động chủ yếu trên RAM, Redis có thể thực hiện hàng triệu thao tác mỗi giây với độ trễ dưới một mili giây.
- **Hỗ trợ đa dạng cấu trúc dữ liệu:** Khác với các hệ thống cache key-value đơn thuần chỉ lưu chuỗi (string), Redis hỗ trợ các cấu trúc dữ liệu phong phú như:
  - **Strings:** Lưu trữ các đối tượng JSON, HTML,...
  - **Hashes:** Lưu trữ các đối tượng có cấu trúc (như thông tin người dùng), cho phép cập nhật từng trường riêng lẻ mà không cần đọc/ghi lại toàn bộ đối tượng.

  - **Lists:** Lưu trữ các danh sách, phù hợp cho việc caching dòng thời gian (timeline).
  - **Sets:** Lưu trữ các tập hợp giá trị duy nhất.
  - **Sorted Sets:** Lưu trữ các tập hợp có thứ hạng, cực kỳ hiệu quả cho việc xây dựng bảng xếp hạng (leaderboard) thời gian thực.
- **Tích hợp cơ chế hết hạn (Expiration):** Redis cho phép đặt thời gian tồn tại (Time-To-Live - TTL) cho mỗi key. Sau khoảng thời gian này, key sẽ tự động bị xóa, giúp giải quyết vấn đề dữ liệu cũ (stale data) một cách tự động.
- **Tính bền bỉ tùy chọn (Optional Persistence):** Mặc dù hoạt động trên RAM, Redis cung cấp cơ chế để lưu dữ liệu xuống đĩa (RDB, AOF), đảm bảo dữ liệu trong cache có thể được khôi phục sau khi khởi động lại.
- **Khả năng mở rộng và Sẵn sàng cao (Scalability & High Availability):** Hỗ trợ các mẫu hình như Redis Sentinel (tự động chuyển đổi dự phòng) và Redis Cluster (phân mảnh dữ liệu theo chiều ngang) để xây dựng các hệ thống cache lớn, có khả năng chịu lỗi cao.

1.3.3. Kết chương

Chương này trình bày về cơ sở lý thuyết, các phương pháp, kỹ thuật được áp dụng vào giải quyết các yêu cầu của đề tài.

CHƯƠNG 2. Phân tích và thiết kế hệ thống

2.1. Phân tích hiện trạng và các vấn đề tồn tại của hệ thống

2.1.1. Kiến trúc chung

Qua quá trình khảo sát thực tế hệ thống cổng thông tin điện tử của Trường Đại học Bách khoa, Đại học Đà Nẵng (DUT), có thể nhận thấy hệ thống được cấu thành từ nhiều trang web độc lập, phục vụ các nhu cầu khác nhau. Một số trang web chính bao gồm:

- **Trang chủ: dut.udn.vn**
- **Cổng thông tin sinh viên: sv.dut.udn.vn**
- **Cổng thông tin cán bộ: cb.dut.udn.vn**
- **Trang đăng ký học phần: dk.dut.udn.vn**
- **Trang hỏi đáp: fr.dut.udn.vn**
- **Thư viện: lib.dut.udn.vn**
- **Khoa học công nghệ: khcn.dut.udn.vn**
- **Và một số trang web khác đang được xây dựng: LMS, Chuẩn đầu ra (chuandaura.dut.udn.vn), Tuyển sinh, Lịch tuần (lichtuan.dut.udn.vn).**

Tuy nhiên, mô hình hoạt động hiện tại bộc lộ nhiều hạn chế cần được giải quyết, có thể tổng kết thành các nhóm vấn đề chính như sau:

Vấn đề cốt lõi của hệ thống hiện tại là phần lớn các trang web được triển khai theo **kiến trúc nguyên khối (Monolithic)**. Kiến trúc này gây ra các trở ngại lớn:

- **Khó khăn trong việc nâng cấp và bảo trì:** Sự liên kết chặt chẽ (tight-coupling) giữa các module chức năng trong một khối thống nhất khiến việc thay đổi một thành phần nhỏ cũng có thể ảnh hưởng đến toàn bộ hệ thống, làm tăng rủi ro và kéo dài thời gian phát triển, triển khai.
- **Phức tạp trong quản lý cơ sở dữ liệu:** Việc sử dụng một cơ sở dữ liệu dùng chung hoặc nhiều cơ sở dữ liệu thiếu sự đồng bộ chặt chẽ dẫn đến tình trạng dữ liệu bị rối, phân mảnh và khó quản lý. Điều này đặc biệt rõ ràng khi có nhiều hệ thống con cùng thao tác trên một tập dữ liệu.

Đây là hệ quả trực tiếp của vấn đề kiến trúc, thể hiện rõ qua các quy trình nghiệp vụ:

- **Đồng bộ hóa thủ công:** Dữ liệu đăng ký tín chỉ từ trang dk.dut.udn.vn không được tự động cập nhật sang trang sinh viên (sv.dut.udn.vn), mà phải thực hiện qua các quy trình thủ công. Điều này gây ra độ trễ lớn, sinh viên không thể thấy kết quả ngay lập tức trên trang thông tin cá nhân.

- **Cập nhật thông tin chậm trễ:** Thông tin về các học phần thay thế, học phần tương đương thường được cập nhật chậm. Điều này gây hoang mang và phàn nàn từ phía sinh viên, đặc biệt là các sinh viên năm cuối cần hoàn thành chương trình đào tạo để xét tốt nghiệp.

Sự thiếu vắng một cơ chế đồng bộ dữ liệu tự động, theo thời gian thực giữa các hệ thống là một điểm yếu nghiêm trọng, ảnh hưởng trực tiếp đến tính chính xác của thông tin và trải nghiệm của người dùng.

Hệ thống hiện tại thiếu một kênh giao tiếp chủ động để tự động gửi thông báo đến người dùng qua các kênh cá nhân như email. Điều này dẫn đến việc sinh viên bị bỏ lỡ các thông tin quan trọng:

- **Phụ thuộc vào các kênh không chính thức:** Các thông báo quan trọng như đợt đăng ký thi Anh văn định kỳ, Anh văn chuẩn đầu ra thường được đăng tải qua các kênh mạng xã hội (Facebook), khiến những sinh viên không theo dõi có thể bị bỏ lỡ.
- **Không có cơ chế nhắc nhở tự động:** Thiếu hệ thống tự động nhắc nhở sinh viên thực hiện các hoạt động cần đúng hạn như đánh giá điểm rèn luyện, dẫn đến tình trạng sinh viên quên thực hiện.
- **Thông báo bị động:** Các thông báo về thay đổi lịch học (nghỉ, học bù) phụ thuộc hoàn toàn vào việc sinh viên phải tự giác truy cập trang sinh viên để kiểm tra, thay vì được hệ thống chủ động gửi đến.

Sự tồn tại của quá nhiều tên miền riêng biệt cho các chức năng khác nhau tạo ra một trải nghiệm người dùng rời rạc và khó khăn:

- **Khó ghi nhớ và truy cập:** Sinh viên và giảng viên phải ghi nhớ nhiều địa chỉ web khác nhau, gây khó khăn và bất tiện trong quá trình sử dụng.
- **Hệ thống xác thực bị phân mảnh (Thiếu Single Sign-On - SSO):** Đây là một trong những bất cập lớn nhất. Việc các trang web hoạt động tách rời dẫn đến mỗi hệ thống có một cơ chế xác thực và quản lý người dùng riêng. Hậu quả là người dùng (sinh viên, giảng viên) phải duy trì nhiều tài khoản và đăng nhập lặp lại trên các trang khác nhau, gây ra sự phiền toái và làm gia tăng rủi ro về bảo mật (người dùng có xu hướng đặt mật khẩu yếu và giống nhau ở nhiều nơi).
- **Cấu trúc thiếu tập trung:** Trang thông tin sinh viên (sv.dut.udn.vn), vốn được xem là cổng thông tin chính, lại có cấu trúc lộn xộn. Một số chức năng được tích hợp trực tiếp, trong khi nhiều chức năng quan trọng khác lại điều hướng người dùng sang một trang web hoàn toàn khác. Điều này phá vỡ tính liền mạch và gây khó khăn cho người dùng khi tìm kiếm thông tin.

- **Nguy cơ gia tăng sự phân mảnh:** Việc tiếp tục xây dựng các trang web mới với tên miền riêng mà không có một chiến lược tích hợp tổng thể sẽ càng làm trầm trọng thêm vấn đề này trong tương lai.

2.1.2. Hệ thống đăng ký tín chỉ

Hệ thống Đăng ký tín chỉ là một trong những thành phần quan trọng và nhạy cảm nhất trong hệ sinh thái ứng dụng của nhà trường. Tuy nhiên, qua quá trình khảo sát, hệ thống hiện tại bộc lộ nhiều điểm yếu nghiêm trọng về mặt kiến trúc và hiệu năng, dẫn đến tình trạng quá tải và sập hệ thống thường xuyên vào các thời điểm cao điểm.

a. Kiến trúc nguyên khối (Monolithic)

Hệ thống được xây dựng trên kiến trúc nguyên khối, trong đó toàn bộ chức năng được đóng gói chung vào một khối ứng dụng duy nhất. Kiến trúc này gây ra các hậu quả trực tiếp:

- Điểm nghẽn cổ chai (Bottleneck): Một tác vụ chậm, ví dụ như truy vấn cơ sở dữ liệu, sẽ làm tắc nghẽn toàn bộ ứng dụng.
- Một lỗi phát sinh ở bất kỳ chức năng nào cũng có nguy cơ làm sập toàn bộ hệ thống.
- Không thể mở rộng quy mô một cách linh hoạt mà buộc phải nhân bản toàn bộ khối ứng dụng, gây lãng phí tài nguyên.

b. Thao tác Cơ sở dữ liệu không được Tối ưu hóa

- Thiếu chỉ mục (Indexing): Các bảng dữ liệu quan trọng không được đánh chỉ mục hợp lý, khiến cơ sở dữ liệu phải thực hiện quét toàn bộ bảng (full table scan) — một thao tác cực kỳ chậm và tốn tài nguyên.
- Câu truy vấn phức tạp: Các câu lệnh SQL để kiểm tra điều kiện đăng ký (JOIN nhiều bảng) trở thành gánh nặng cực lớn cho cơ sở dữ liệu khi không có chỉ mục hỗ trợ.

c. Không tận dụng cơ chế Caching (Bộ nhớ đệm)

Hệ thống thiếu hoàn toàn cơ chế caching, một trong những kỹ thuật cơ bản và hiệu quả nhất để tối ưu hiệu năng. Các dữ liệu có tính tĩnh hoặc ít thay đổi (như danh sách lớp học phần, điểm đã học) đều được truy vấn trực tiếp từ cơ sở dữ liệu mỗi khi có yêu cầu, gây ra hàng vạn lượt truy vấn thừa thãi và lãng phí tài nguyên.

d. Vòng lặp tiêu cực từ hành vi người dùng: Nạn "Auto-click"

Một vấn đề nghiêm trọng khác, vừa là nguyên nhân vừa là hệ quả của sự chậm chạp của hệ thống, là hành vi sử dụng các công cụ auto-click.

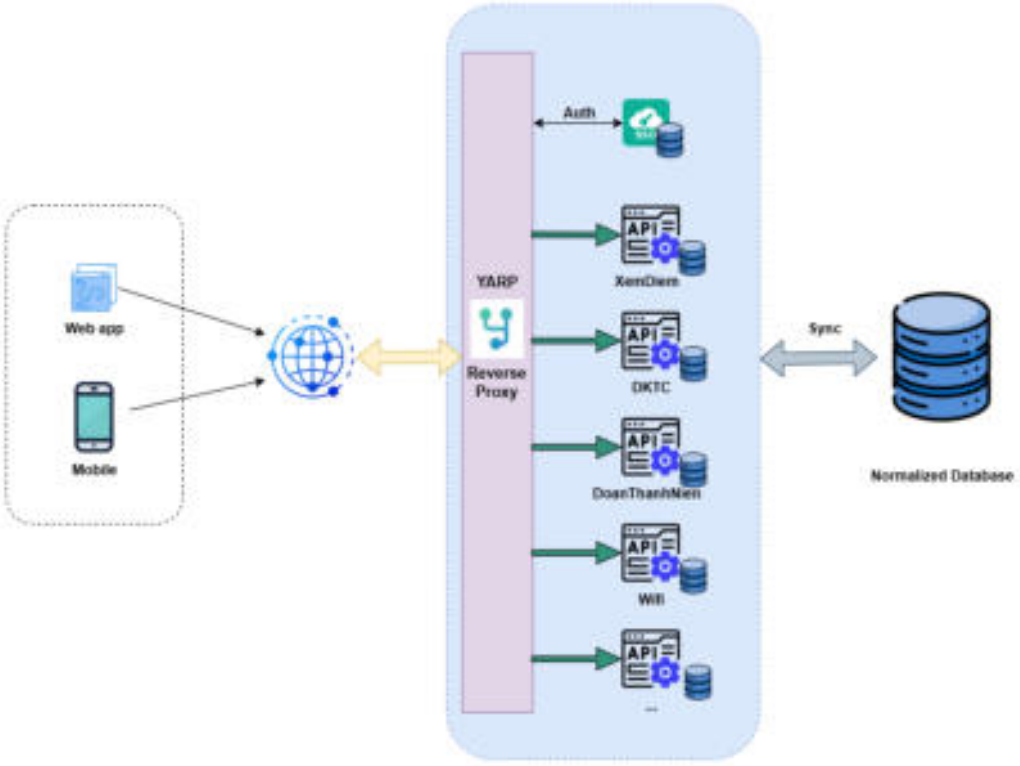
Nguyên nhân: Do lo sợ không đăng ký được vào các lớp học phần mong muốn vì hệ thống thường xuyên bị treo, sinh viên đã tìm đến các giải pháp tự động. Họ sử dụng

các đoạn mã kịch bản (script) hoặc tiện ích mở rộng (extension) trên trình duyệt để gửi yêu cầu đăng ký một cách liên tục với tần suất cực cao.

2.2. Thiết kế hệ thống

2.2.1. Xây dựng kiến trúc Microservice cho trường Đại học Bách Khoa

a. Mô hình kiến trúc Microservice đề xuất



Hình 2.1. Mô hình kiến trúc Microservice đề xuất

- **Lớp client:** Bao gồm các ứng dụng phía người dùng cuối như Web App (dành cho trình duyệt máy tính) và Mobile App
- **Lớp cổng vào API (Reverse Proxy): Sử dụng YARP (Yet Another Reverse Proxy), một thư viện Reverse Proxy hiệu năng cao, linh hoạt do Microsoft phát triển. Giúp điều hướng và xử lý các tác vụ chung.**
- **Lớp dịch vụ:**
    - **SSO Service**: Người dùng chỉ cần đăng nhập một lần duy nhất tại Auth Service, sau đó có thể truy cập tất cả các dịch vụ khác (XemDiem, DKTC...) mà không cần phải đăng nhập lại. Điều này giải quyết triệt để vấn đề đăng nhập lặp lại đã tồn tại ở hiện trạng.
    - **Các Microservices nghiệp vụ:** Mỗi chức năng lớn của trường sẽ được xây dựng thành một microservice độc lập, mỗi microservice có DB riêng

- **Lớp dữ liệu:** Dữ liệu từ các cơ sở dữ liệu riêng lẻ của các microservice sẽ được đồng bộ hóa (sync) một cách định kỳ hoặc theo sự kiện vào cơ sở dữ liệu tập trung này.

Kết luận: Mô hình đề xuất này không chỉ là một sự nâng cấp công nghệ mà còn là một giải pháp kiến trúc toàn diện, giải quyết các vấn đề cốt lõi của hệ thống hiện tại, mang lại một hệ thống thống nhất, linh hoạt, dễ bảo trì và có khả năng mở rộng mạnh mẽ trong tương lai.

b. Lợi ích của triển khai hệ thống trên

Áp dụng kiến trúc này cho hệ thống thông tin của Đại học Bách Khoa mang lại nhiều lợi ích vượt trội so với kiến trúc nguyên khối (Monolithic) truyền thống:

- **Linh hoạt và Dễ mở rộng (Scalability):** Mỗi dịch vụ (ví dụ: Quản lý sinh viên, Quản lý môn học) có thể được mở rộng quy mô một cách độc lập. Nếu hệ thống đăng ký học phần quá tải vào đầu kỳ, chỉ cần tăng tài nguyên cho dịch vụ đó mà không ảnh hưởng đến các dịch vụ khác.
- **Tăng cường sự ổn định (Resilience):** Nếu một dịch vụ gặp lỗi (ví dụ: dịch vụ Thư viện), các dịch vụ khác như Cổng thông tin sinh viên hay Hệ thống học trực tuyến vẫn hoạt động bình thường.
- **Phát triển và triển khai độc lập:** Các nhóm phát triển khác nhau có thể làm việc song song trên các dịch vụ khác nhau. Việc cập nhật, sửa lỗi cho một dịch vụ có thể được triển khai nhanh chóng mà không cần phải triển khai lại toàn bộ hệ thống.
- **Tự do lựa chọn công nghệ:** Mỗi dịch vụ có thể được xây dựng bằng ngôn ngữ lập trình và công nghệ phù hợp nhất với chức năng của nó (ví dụ: dùng Python cho dịch vụ xử lý dữ liệu, dùng Java/Node.js cho các dịch vụ nghiệp vụ).

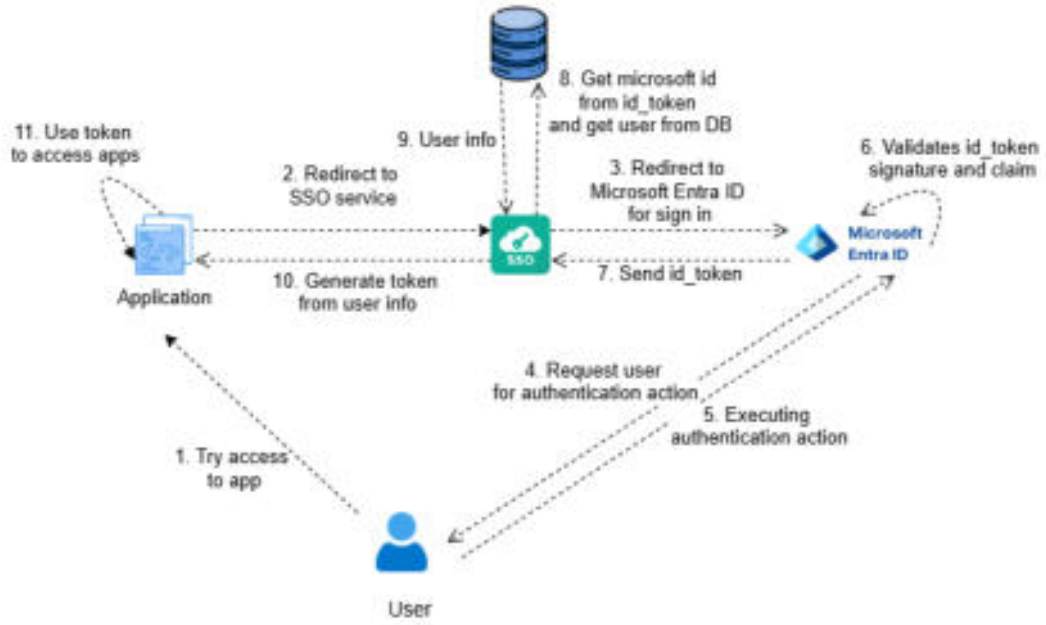
2.2.2. Hệ thống đăng nhập 1 cửa với Microsoft entra id

a. Bài toán và giải pháp

- **Bài toán:** Hiện tại, sinh viên và giảng viên có thể phải sử dụng nhiều tài khoản và mật khẩu khác nhau để truy cập các hệ thống riêng lẻ của trường (Cổng thông tin, hệ thống E-learning, Email, Wifi, Thư viện...). Điều này gây ra:
  - Trải nghiệm người dùng kém, khó ghi nhớ.
  - Rủi ro bảo mật cao (mật khẩu yếu, sử dụng lại mật khẩu).
  - Khó khăn cho bộ phận IT trong việc quản lý, cấp và thu hồi quyền truy cập.

- **Giải pháp:** Triển khai hệ thống Đăng nhập một cửa (Single Sign-On - SSO) sử dụng Microsoft Entra ID làm nhà cung cấp danh tính trung tâm (Identity Provider - IdP). Với SSO, người dùng chỉ cần đăng nhập một lần duy nhất bằng tài khoản của trường (ví dụ: 102210239@sv1.dut.udn.vn) để truy cập vào tất cả các ứng dụng đã được cấp phép.

b. Luồng hoạt động của hệ thống SSO với Microsoft Entra ID



Hình 2.2. Luồng hoạt động của hệ thống SSO với Microsoft Entra ID

- **Các thành phần chính:**
  - **User (Người dùng):** Sinh viên, giảng viên.
  - **Application (Ứng dụng):** Các ứng dụng của trường cần được bảo vệ (ví dụ: Cổng thông tin, E-learning).
  - **SSO Service:** Một dịch vụ trung tâm do trường xây dựng, đóng vai trò "cổng xác thực". Nó giao tiếp với cả ứng dụng, người dùng và Microsoft Entra ID.
  - **Database (Cơ sở dữ liệu):** CSDL của trường, chứa thông tin chi tiết về người dùng (mã số sinh viên, khoa, lớp, v.v.).
  - **Microsoft Entra ID[6]:** Dịch vụ xác thực trung tâm của Microsoft, nơi lưu trữ tài khoản và mật khẩu của người dùng.
- **Luồng hoạt động diễn ra theo các bước sau:**

1. **Try access to app (Cố gắng truy cập ứng dụng):** Người dùng mở trình duyệt và truy cập vào địa chỉ của một ứng dụng (ví dụ: https://portal.dut.udn.vn).
2. **Redirect to SSO service (Chuyển hướng đến Dịch vụ SSO):** Ứng dụng nhận thấy người dùng chưa đăng nhập. Thay vì hiển thị form đăng nhập của riêng mình, nó chuyển hướng người dùng đến **Dịch vụ SSO** trung tâm của hệ thống.
3. **Redirect to Microsoft Entra ID (Chuyển hướng đến Microsoft Entra ID): Dịch vụ SSO** cũng thấy người dùng chưa được xác thực, và nó tiếp tục chuyển hướng người dùng đến trang đăng nhập của **Microsoft Entra ID**.
4. **Request user for authentication action (Yêu cầu người dùng xác thực): Microsoft Entra ID** hiển thị giao diện đăng nhập quen thuộc của Microsoft, yêu cầu người dùng nhập tên tài khoản (...sv1.dut.udn.vn) và mật khẩu. (Tại bước này, các chính sách bảo mật như Xác thực đa yếu tố - MFA có thể được áp dụng).
5. **Executing authentication action (Thực hiện xác thực):** Người dùng nhập thông tin đăng nhập.
6. **Validates id_token signature and claim (Tạo và ký id_token):** Sau khi người dùng đăng nhập thành công, **Microsoft Entra ID** tạo ra một id_token. Đây là một "chứng minh thư số" chứa thông tin xác thực đã được mã hóa và ký điện tử để đảm bảo tính toàn vẹn.
7. **Send id_token (Gửi id_token): Microsoft Entra ID** chuyển hướng người dùng trở lại **Dịch vụ SSO**, đính kèm theo id_token.
8. **Get user from DB (Lấy thông tin người dùng từ CSDL): Dịch vụ SSO** nhận id_token. Nó giải mã và xác thực token này. Sau đó, nó trích xuất thông tin định danh của người dùng (ví dụ: email) và dùng nó để truy vấn vào **Cơ sở dữ liệu** nội bộ.
9. **User info (Nhận thông tin người dùng): Cơ sở dữ liệu** trả về thông tin chi tiết của người dùng cho **Dịch vụ SSO** (ví dụ: mã số sinh viên, họ tên, vai trò là 'Sinh viên' hay 'Giảng viên', các quyền hạn đặc biệt).
10. **Generate token from user info (Tạo token nội bộ):** Dựa trên thông tin đầy đủ vừa nhận được, **Dịch vụ SSO** tạo ra một **token nội bộ** (thường là JWT - JSON Web Token). Token này chứa tất cả thông tin mà ứng dụng cần để nhận diện và cấp quyền cho người dùng.
11. **Use token to access apps (Sử dụng token để truy cập ứng dụng): Dịch vụ SSO** chuyển hướng người dùng quay trở lại **Ứng dụng** ban đầu, kèm theo **token nội bộ** vừa tạo. Ứng dụng nhận và xác thực token này. Khi token hợp lệ, ứng dụng sẽ xác nhận người dùng đã đăng nhập thành công và cấp quyền truy cập vào các tài nguyên tương ứng.

c. Lợi ích của kiến trúc này

- **Vai trò của Dịch vụ SSO trung gian:** Đây là một thiết kế rất linh hoạt. Thay vì để mỗi ứng dụng phải tự tích hợp với **Microsoft Entra ID**, chúng chỉ cần "tin tưởng" và giao tiếp với **Dịch vụ SSO** trung tâm. Lợi ích:

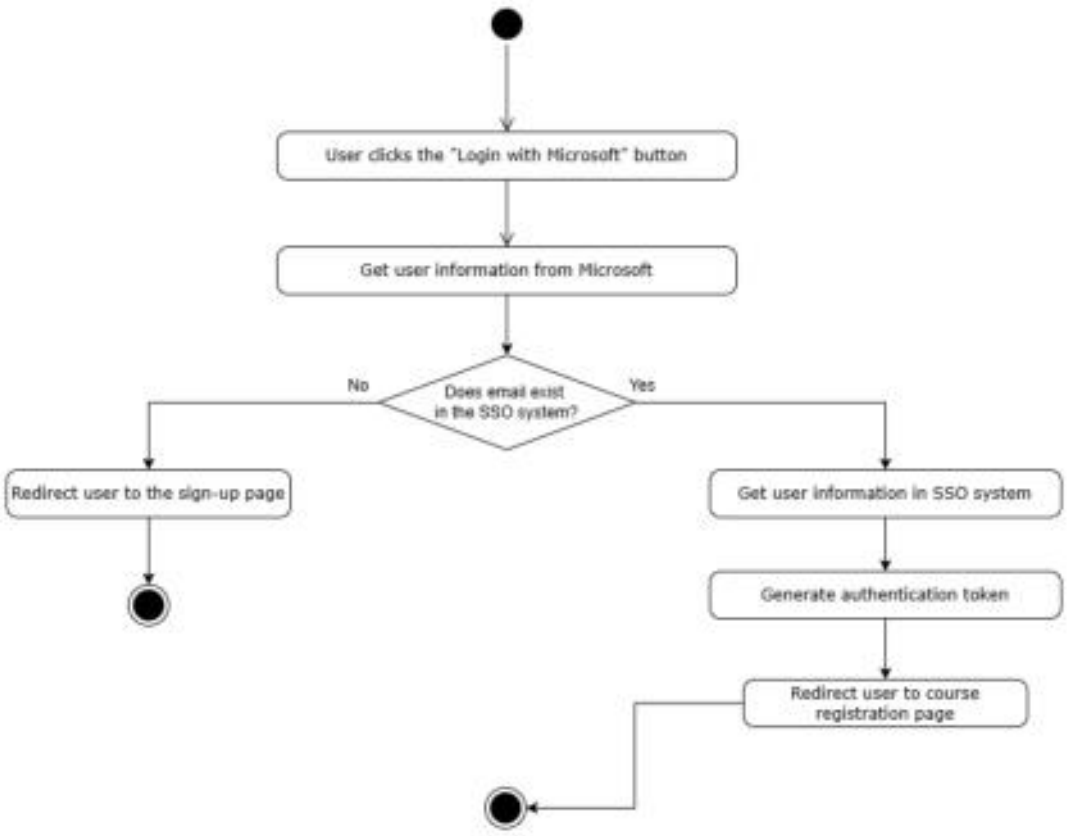
- **Giảm sự phức tạp:** Các lập trình viên ứng dụng không cần biết về logic xác thực phức tạp với Entra ID.
- **Quản lý tập trung:** Mọi logic liên quan đến việc liên kết tài khoản Microsoft với hồ sơ sinh viên/giảng viên trong CSDL đều được xử lý tại một nơi duy nhất.
- **Linh hoạt:** Nếu sau này trường muốn đổi nhà cung cấp danh tính (ví dụ: từ Microsoft sang Google), chỉ cần thay đổi cấu hình ở **Dịch vụ SSO** mà không cần sửa code của tất cả các ứng dụng.

- **Hai loại Token: Luồng này sử dụng hai loại token:**
  - **id_token từ Microsoft Entra ID:** Chỉ dùng để xác thực người dùng một lần duy nhất giữa Dịch vụ SSO và Microsoft.
  - **Token nội bộ từ Dịch vụ SSO:** Dùng để quản lý phiên đăng nhập và cấp quyền cho người dùng trên tất cả các ứng dụng của trường.

2.3. Tích hợp ứng dụng đăng ký tín chỉ

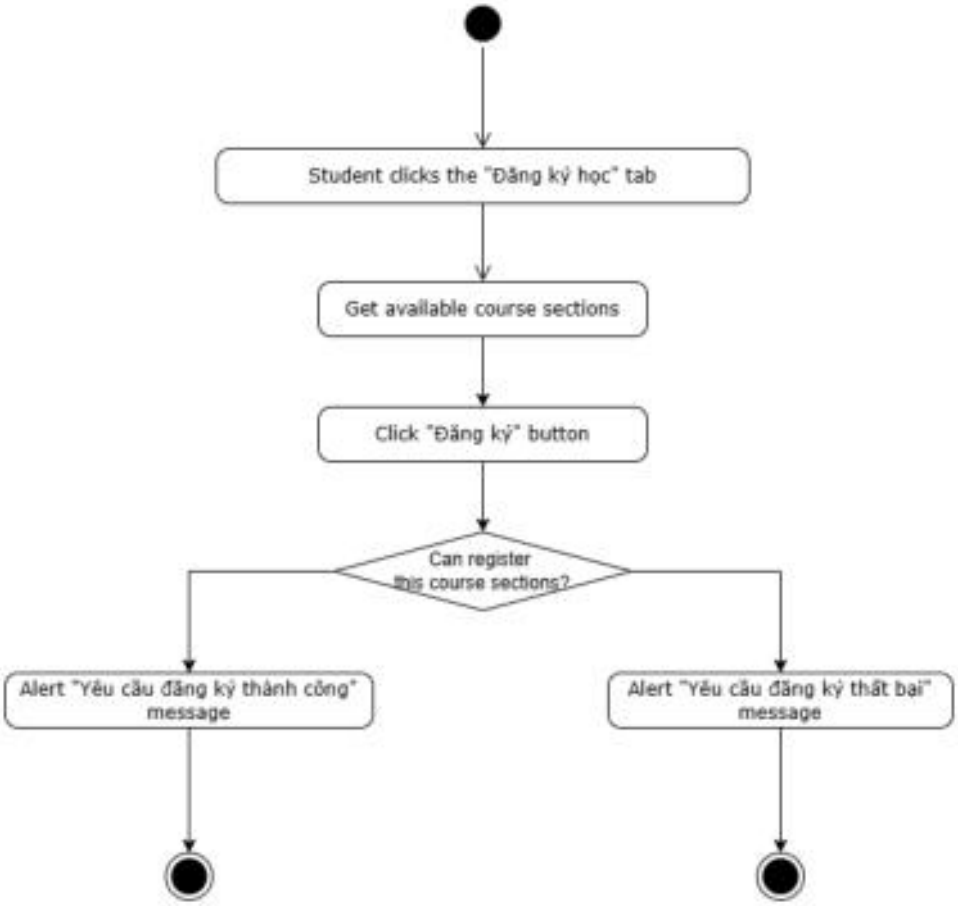
2.3.1. Biểu đồ hoạt động

a. Đăng nhập bằng Microsoft



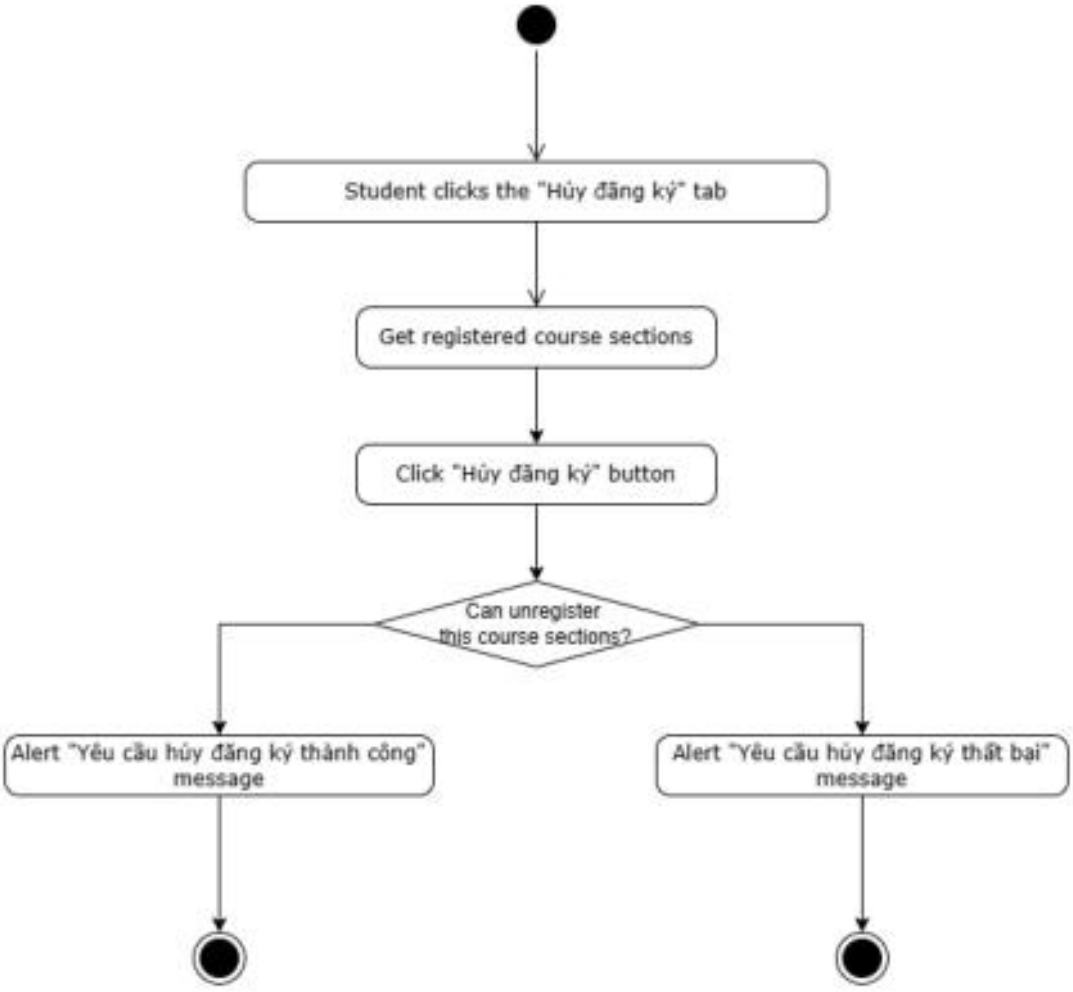
Hình 2.3. Biểu đồ hoạt động đăng nhập bằng Microsoft

b. Đăng ký lớp học phần



Hình 2.4. Biểu đồ hoạt động đăng ký lớp học phần

c. Hủy đăng ký lớp học phần



Hình 2.5. Biểu đồ hoạt động hủy đăng ký lớp học phần

2.3.2. Các tác nhân

Hiện tại hệ thống có các tác nhân sau: Admin, Sinh viên

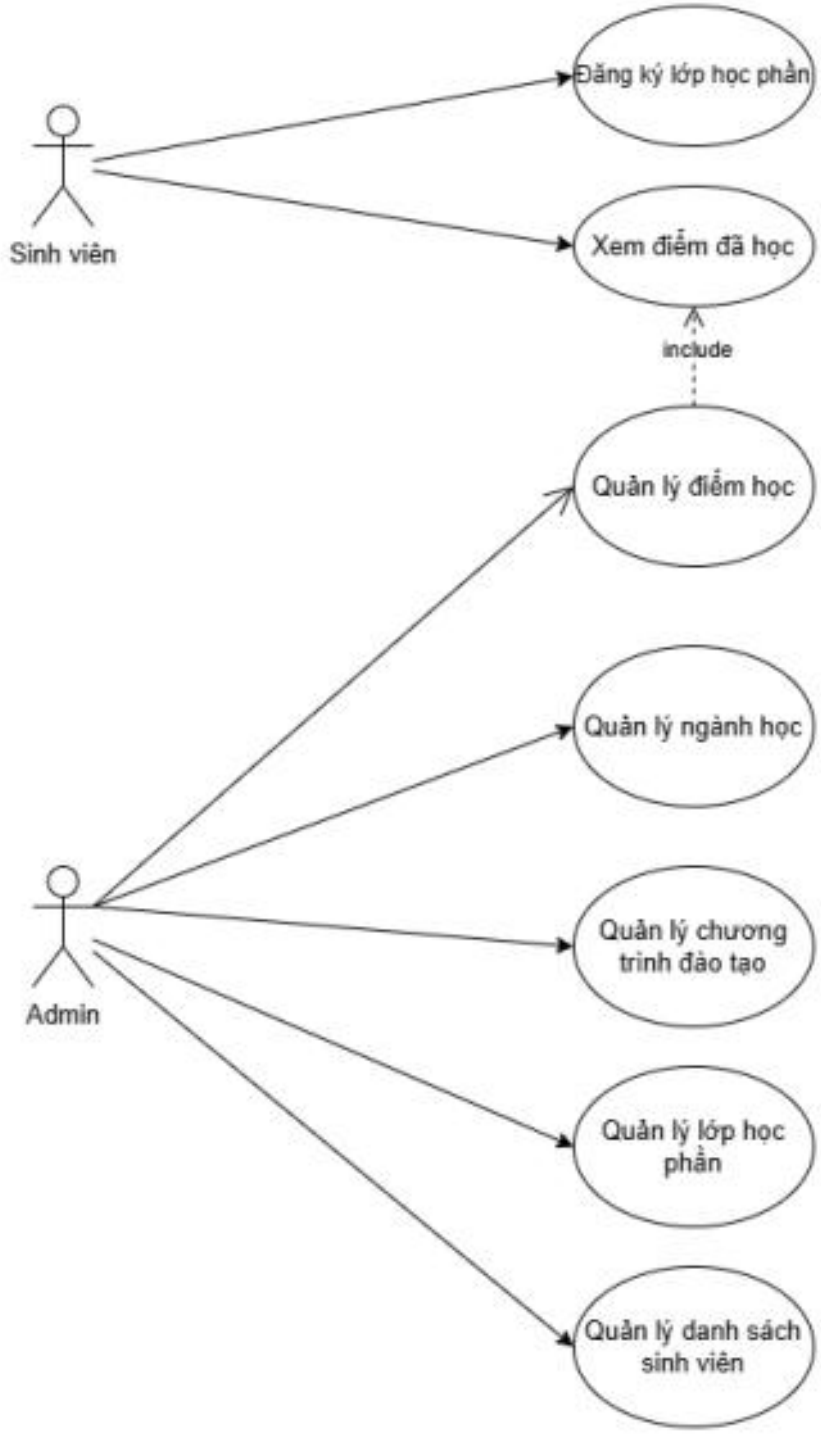
Bảng 2.1. Bảng mô tả cá tác nhân

Tác nhân	Mô tả
Admin	Quản lý các chức năng hệ thống như dữ liệu chương trình đào tạo, lớp sinh hoạt, sinh viên, môn học, lớp học phần, điểm đã học
Sinh viên	Người tham gia đăng ký các lớp học phần

2.3.3. Biểu đồ Use case

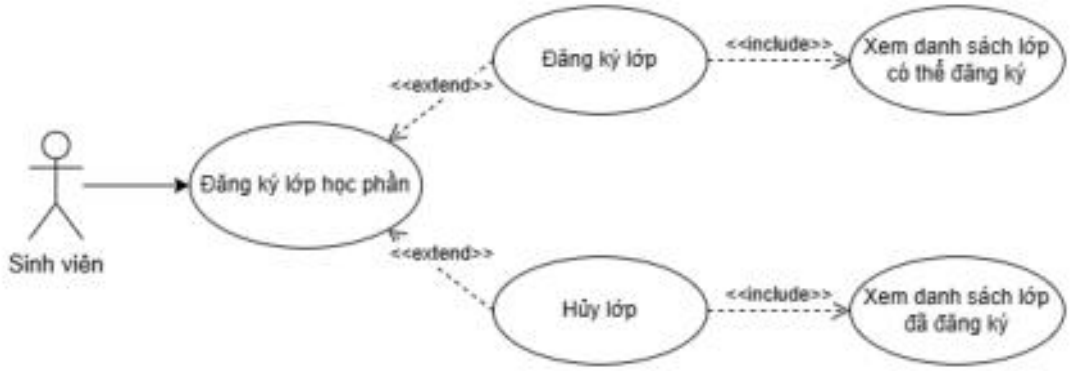
Sơ đồ sau đây cung cấp một cái nhìn tổng quan về các nhóm chức năng, tác nhân tham gia vào *Xây dựng ứng dụng đăng ký tín chỉ*

a. Use case tổng quát



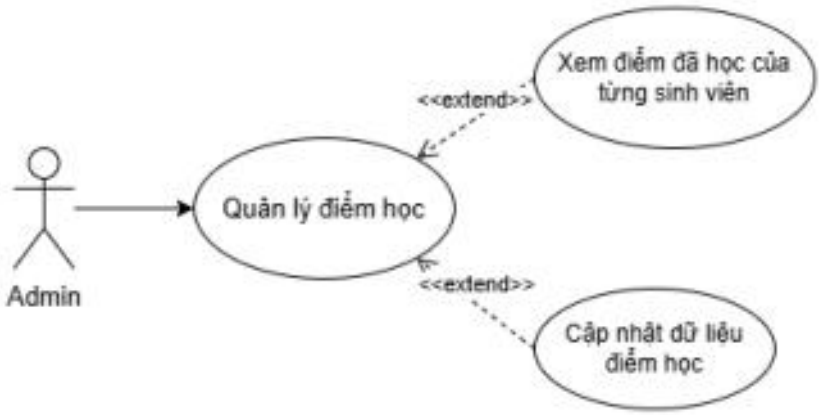
Hình 2.6. Biểu đồ usacase tổng quát

b. Đăng ký lớp học phần



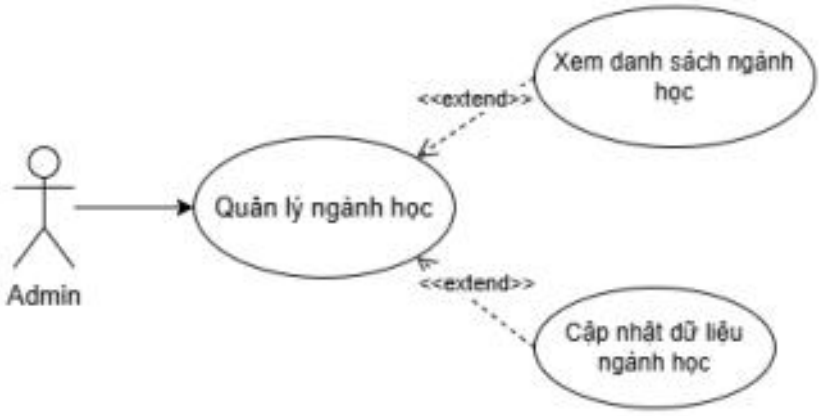
Hình 2.7. Biểu đồ usacase đăng ký lớp học phần

c. Quản lý điểm học



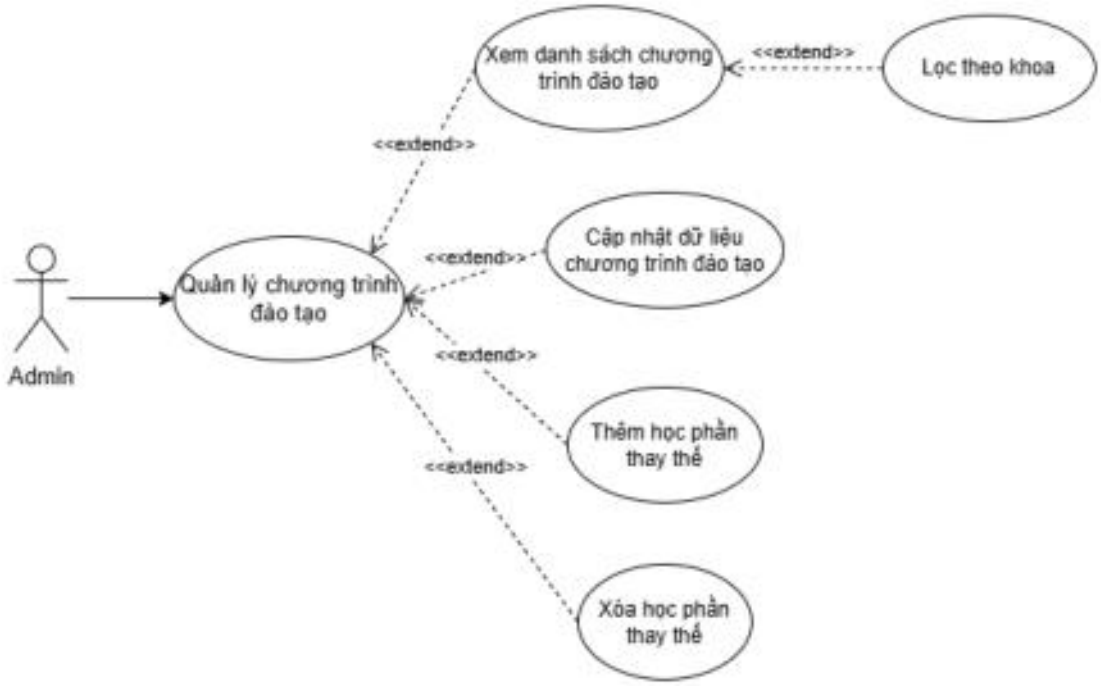
Hình 2.8. Biểu đồ usacase quản lý điểm học

d. Quản lý ngành học



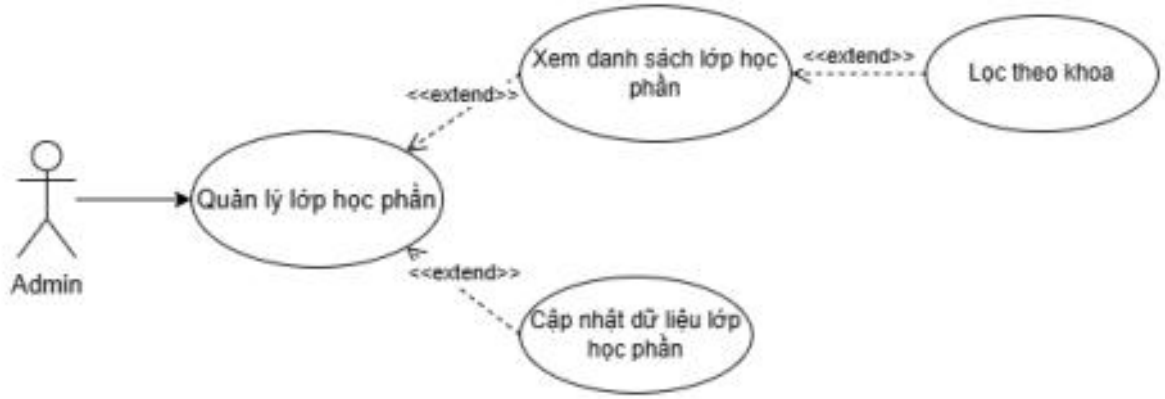
Hình 2.9. Biểu đồ usecase quản lý ngành học

e. Quản lý chương trình đào tạo



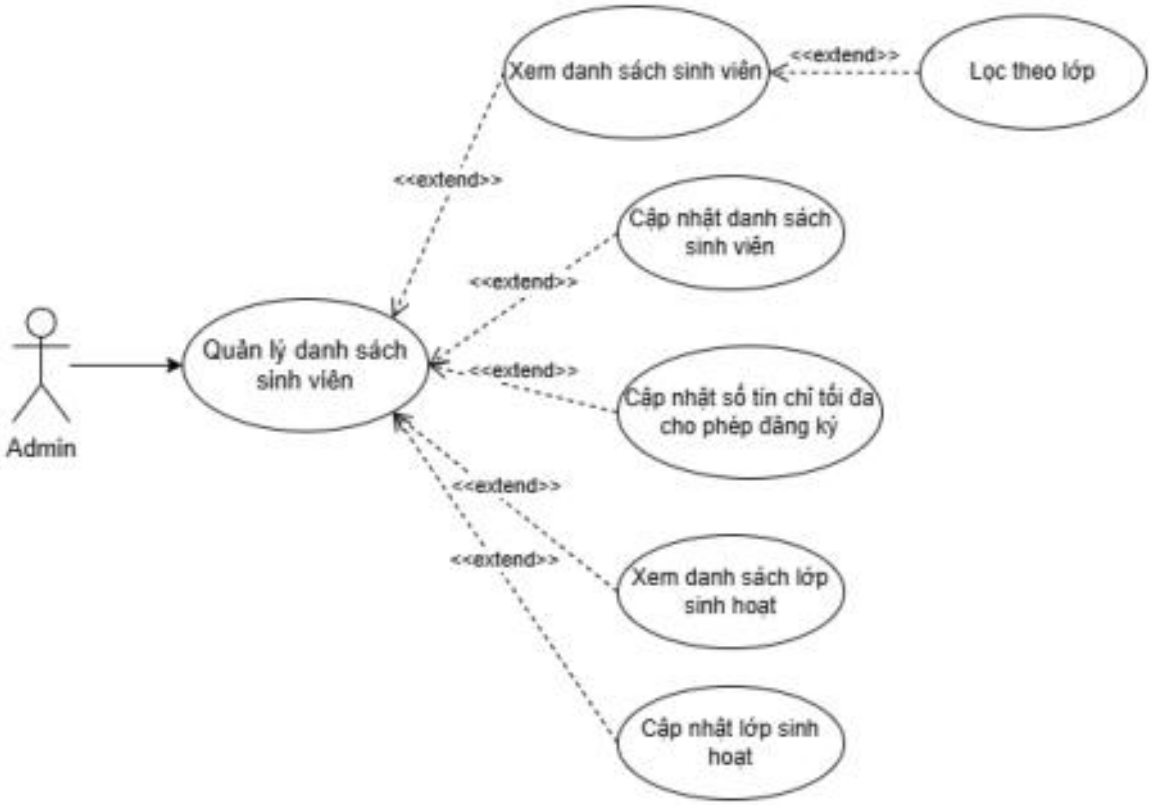
Hình 2.10. Biểu đồ usacase quản lý chương trình đào tạo

f. Quản lý lớp học phần



Hình 2.11. Biêu đồ usacase quản lý lớp học phần

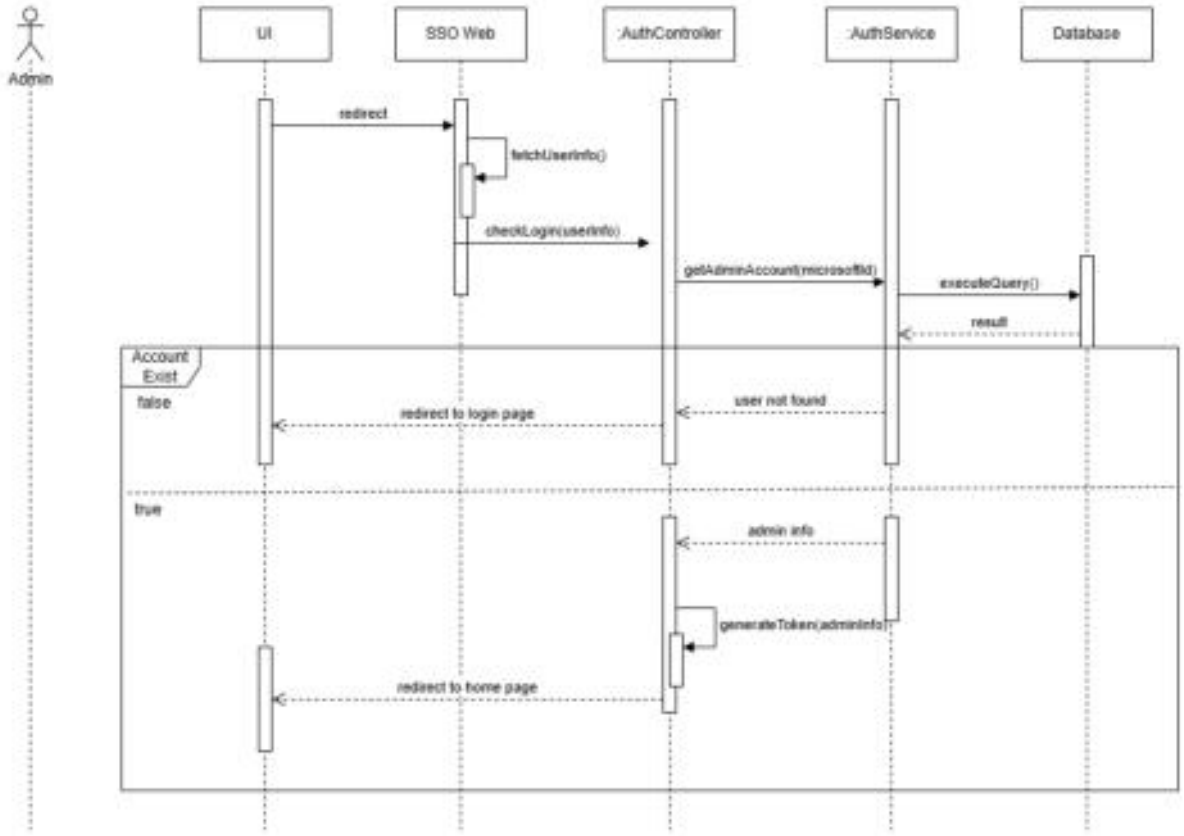
g. Quản lý danh sách sinh viên



Hình 2.12. Biểu đồ quản lý danh sách sinh viên

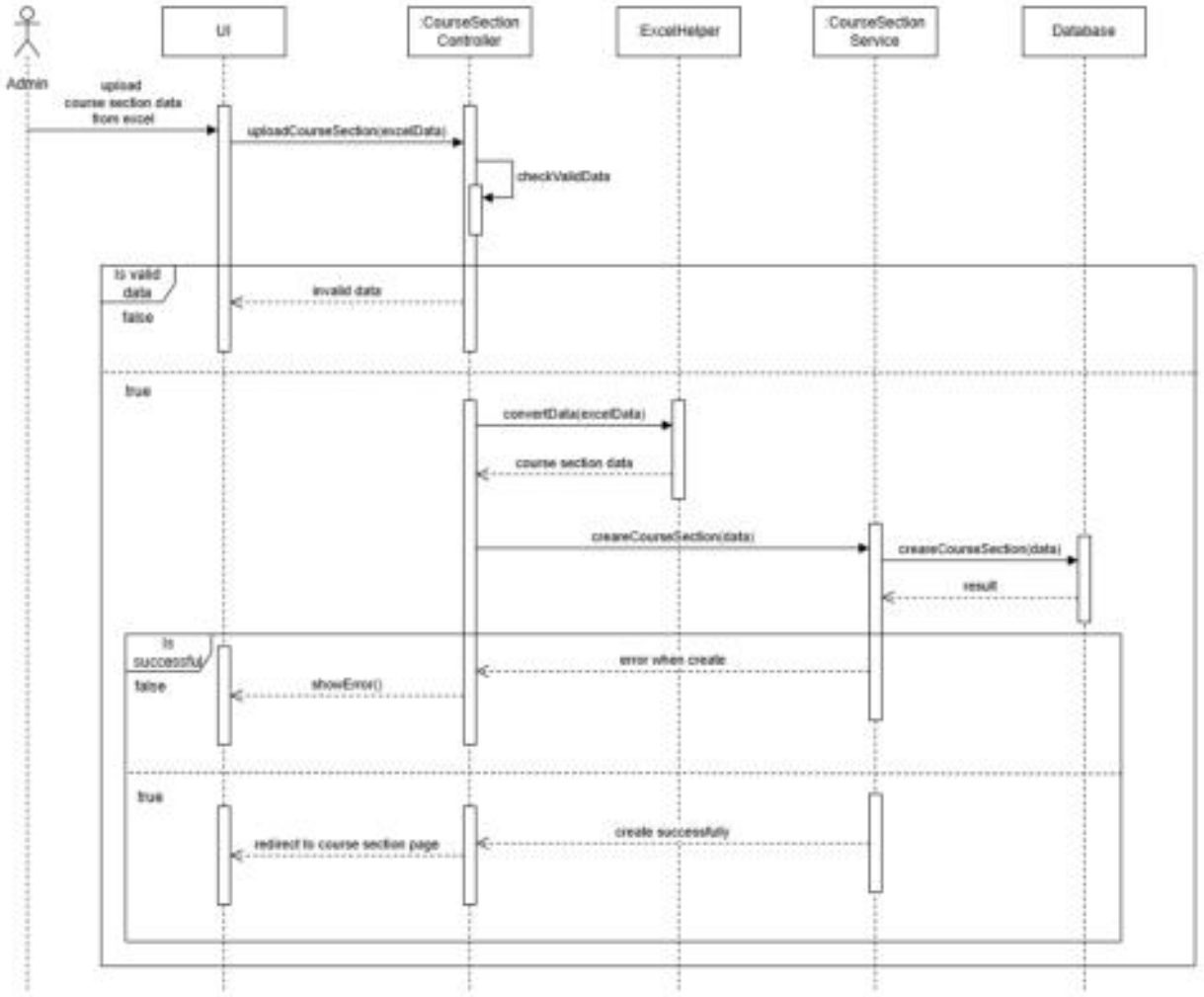
2.3.4. Biểu đồ tuần tự

a. Đăng nhập bằng tài khoản Microsoft



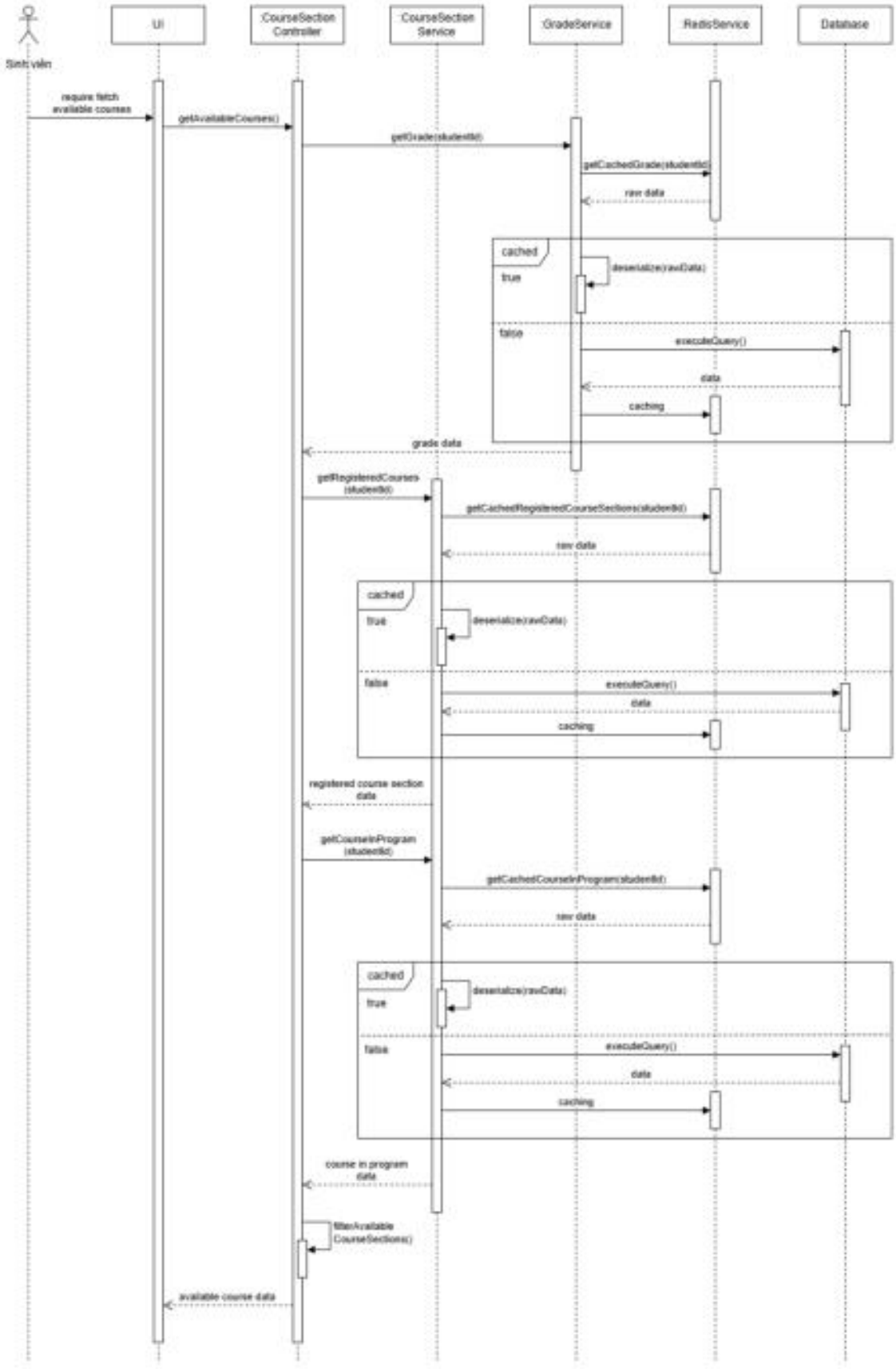
Hình 2.13. Biểu đồ tuần tự đăng nhập bằng tài khoản Microsoft

b. Tạo lớp học phần



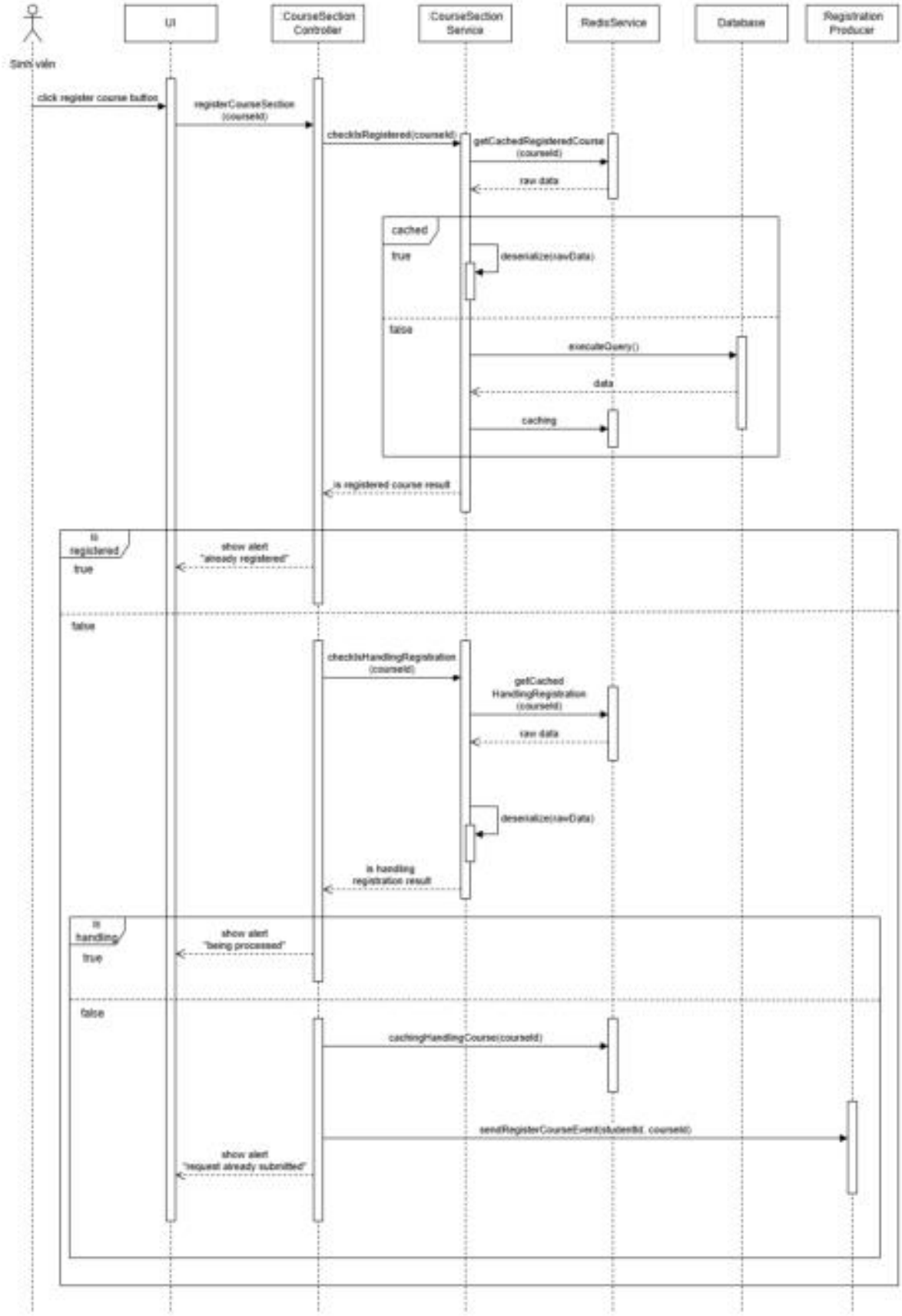
Hình 2.14. Biểu đồ tuần tự tạo lớp học phần

c. Lấy dữ liệu lớp học phần có thể đăng ký



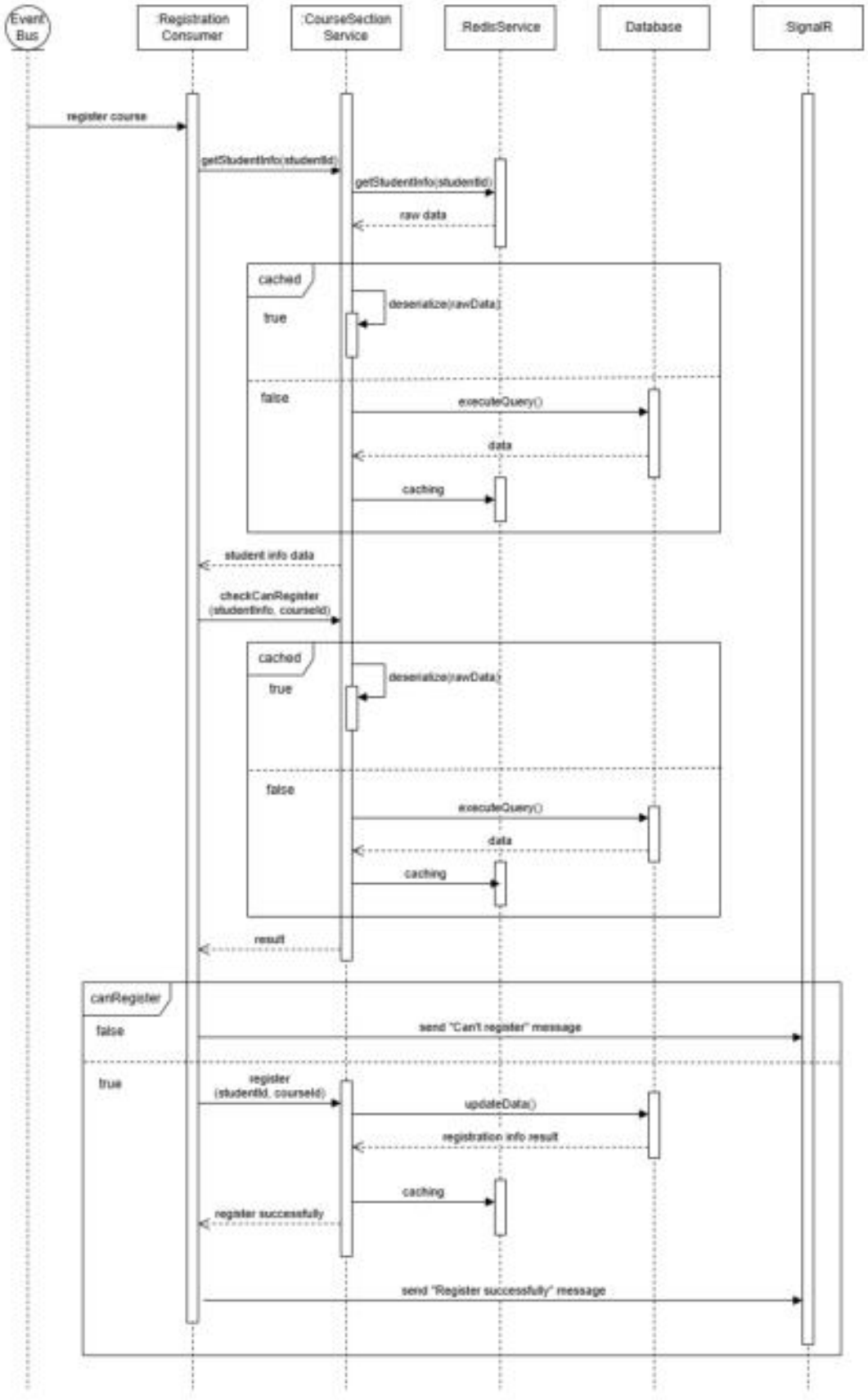
Hình 2.15. Biểu đồ tuần tự lấy dữ liệu lớp học phần có thể đăng ký

d. Đăng ký lớp học phần



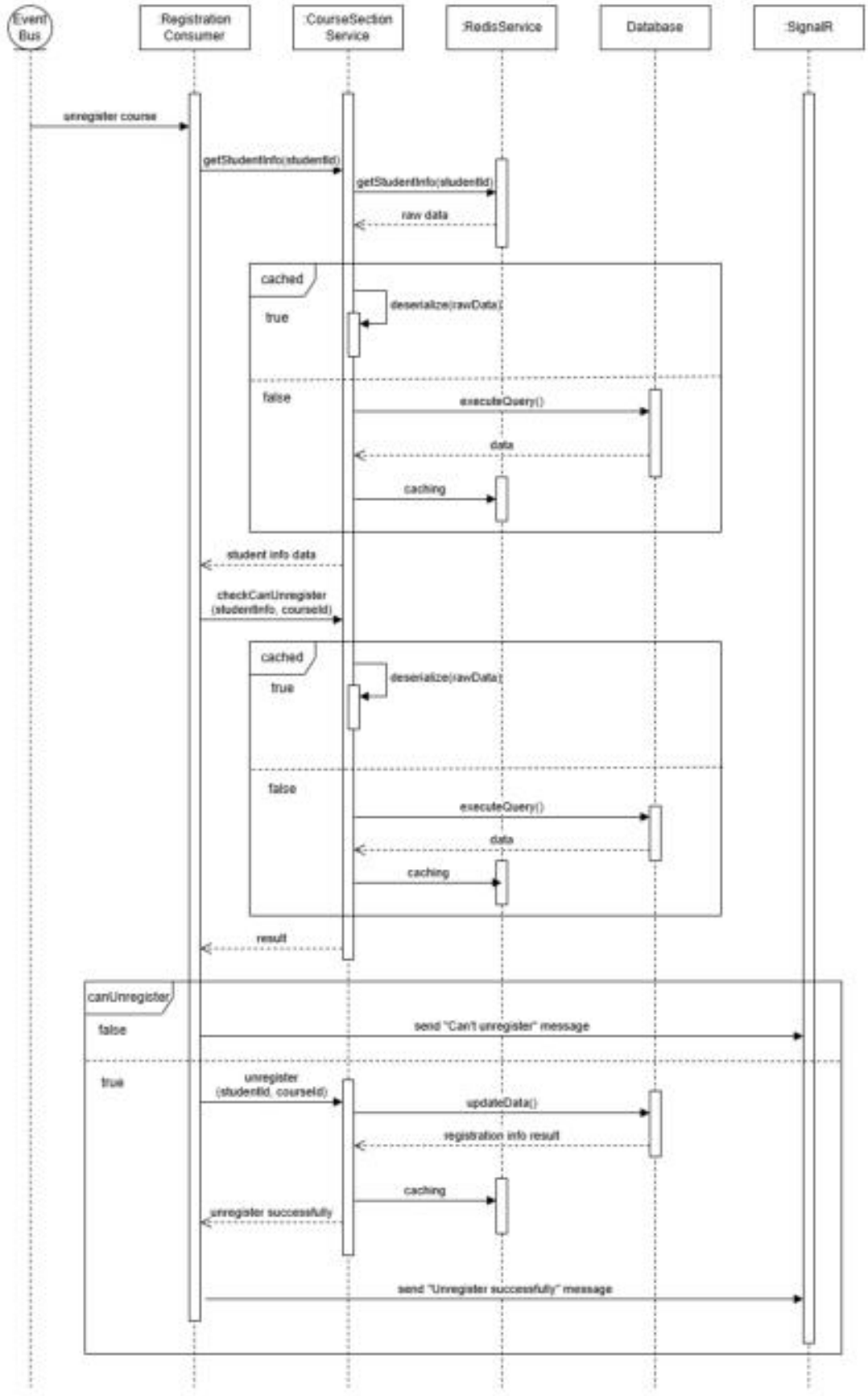
Hình 2.16. Biểu đồ tuần tự đăng ký lớp học phần

e. Xử lý sự kiện đăng ký lớp học phần



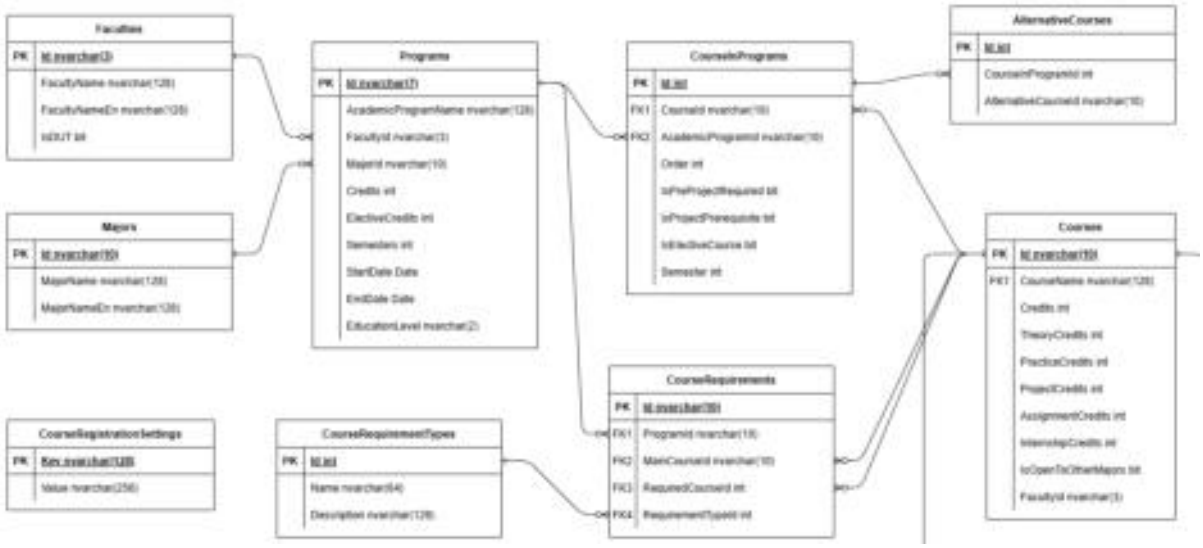
Hình 2.17. Biểu đồ tuần tự xử lý sự kiện đăng ký lớp học phần

f. Xử lý sự kiện hủy đăng ký lớp học phần

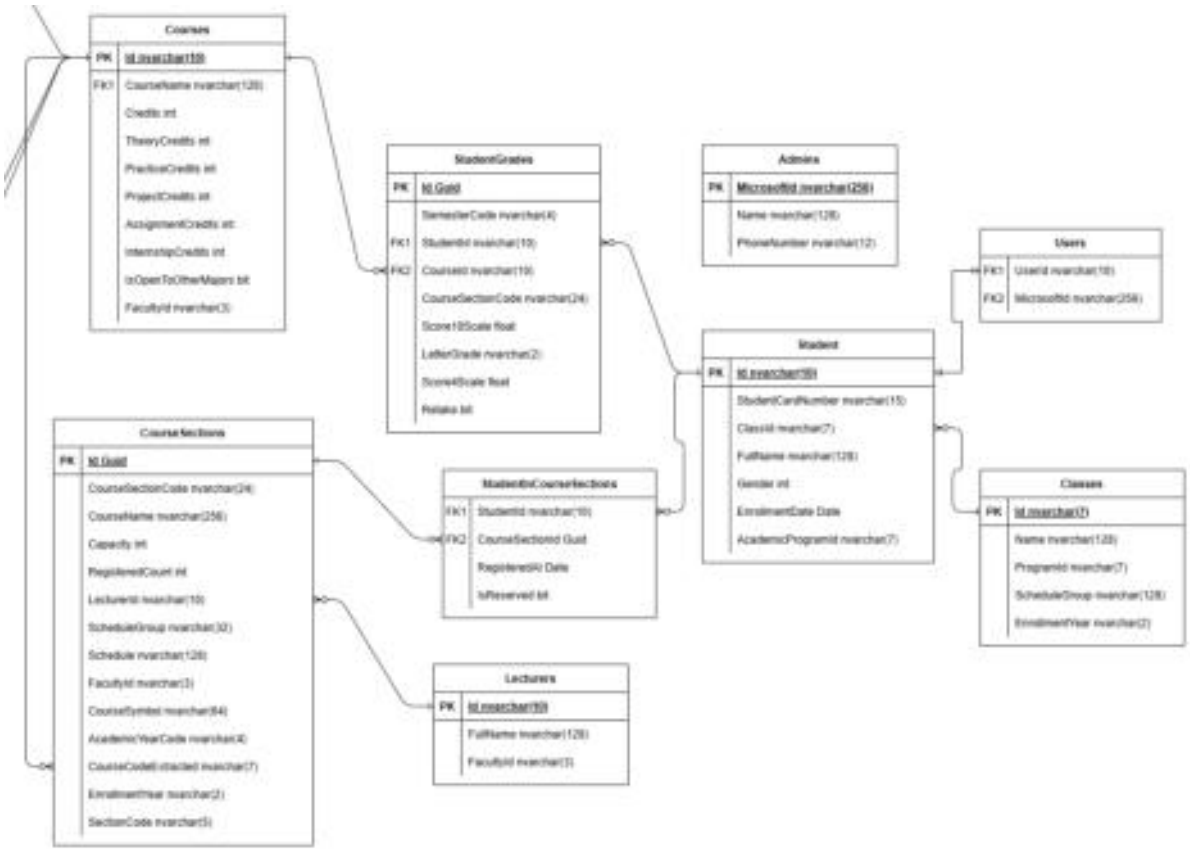


Hình 2.18. Biểu đồ tuần tự xử lý sự kiện hủy đăng ký lớp học phần

2.3.5. Sơ đồ mối quan hệ thực thể (ERD)



Hình 2.19. Sơ đồ mối quan hệ thực thể của hệ thống đăng ký tín chỉ (1)



Hình 2.20. Sơ đồ mối quan hệ thực thể của hệ thống đăng ký tín chỉ (2)

2.3.6. Kết chương

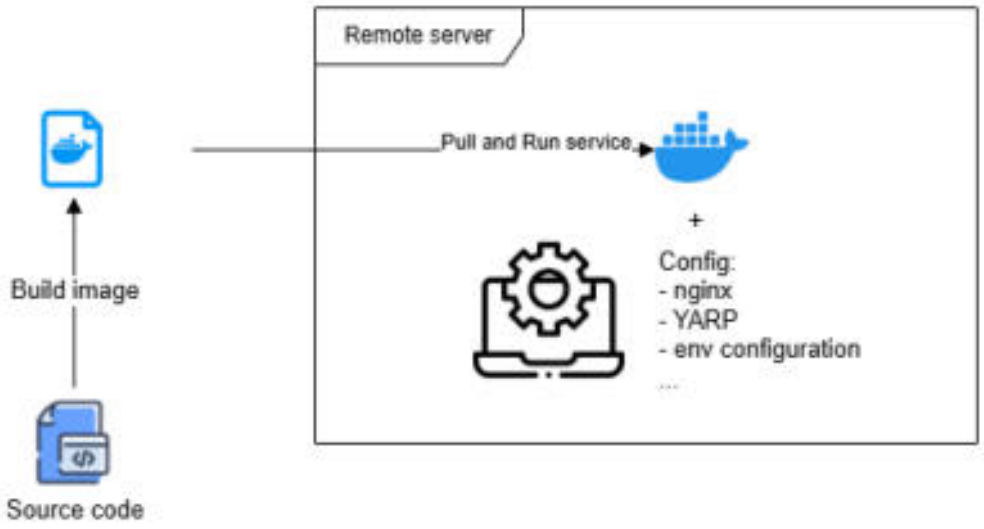
Trong chương này cung cấp cho người đọc cái nhìn tổng quan về thiết kế hệ thống theo kiến trúc Microservices đăng nhập 1 cửa cho các hệ thống phần mềm trường Đại học Bách khoa Đà Nẵng. Bên cạnh đó là cái nhìn tổng quan về nghiệp vụ của hệ thống đăng ký tín chỉ.

CHƯƠNG 3. TRIỂN KHAI VÀ VẬN HÀNH HỆ THỐNG

3.1. Triển khai hệ thống

3.1.1. Đối với các hệ thống phần mềm trường ĐHBK

Để tự động hóa và đảm bảo tính nhất quán khi triển khai các microservice, hệ thống áp dụng một quy trình Tích hợp và Triển khai Liên tục (CI/CD). Quy trình này giúp chuyển đổi mã nguồn từ máy của lập trình viên thành một dịch vụ chạy ổn định trên máy chủ sản phẩm một cách nhanh chóng và an toàn. Mô hình này được mô tả chi tiết như sau:



Hình 3.1. Quy trình triển khai các service của các phềm mềm ĐHBK

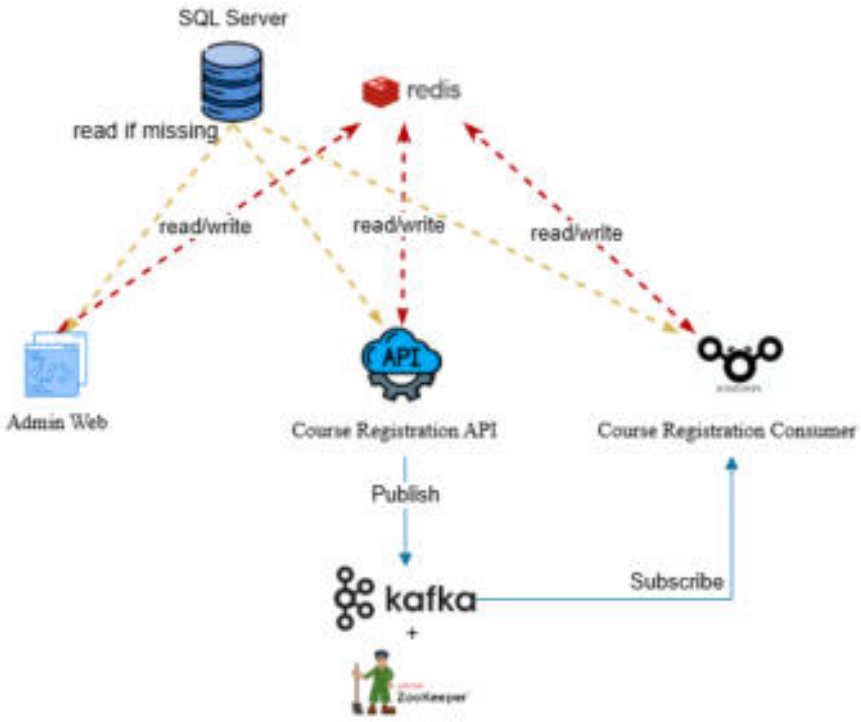
Quy trình được chia thành hai giai đoạn chính: Xây dựng (Build) và Triển khai (Deploy).

Việc chạy container chỉ là một phần của quá trình. Để dịch vụ có thể hoạt động đúng và được truy cập từ bên ngoài, môi trường trên **Remote Server** cần có các cấu hình hỗ trợ:

- Cấu hình nginx, YARP: Trong kiến trúc microservice, không thể để người dùng truy cập trực tiếp vào từng container. Thay vào đó, một Reverse Proxy hoặc API Gateway sẽ đóng vai trò là điểm vào duy nhất cho toàn bộ hệ thống.
- Cấu hình môi trường cho từng service…

3.1.2. Đối với hệ thống đăng ký tín chỉ

a. Sơ đồ hệ thống



Hình 3.2. Sơ đồ hệ thống đăng ký tín chỉ

- **Hệ thống được tổ chức theo mô hình Microservices với 4 service đảm nhiệm các chức năng sau:**
  - **CourseRegistrationConsumer**: nhận nhiệm vụ xử lý các tác vụ về đăng ký/hủy đăng ký lớp học phần.
  - **CourseRegistrationAPIService**: cung cấp API cho ứng dụng di động sử dụng**.**
  - **AdminWebService**: đảm nhận các nhiệm vụ quản lý dữ liệu lớp sinh hoạt, sinh viên, chương trình đào tạo, học phần, lớp học phần, điểm đã học…
  - **StudentWebService**: trang web dành cho sinh viên đăng ký lớp học phần.
- **Các thành phần hạ tầng hỗ trợ:**
  - **Kafka + zookeeper:** nhận event đăng ký/hủy đăng ký lớp học phần từ sinh viên
  - **Redis:** caching lại dữ liệu lớp học phần, điểm đã học của sinh viên…
  - **SQL Server:** đảm bảo sự nhất quán về dữ liệu

b. Môi trường triển khai

Để đảm bảo tính nhất quán, khả năng cô lập và linh hoạt trong vận hành, hệ thống được đề xuất triển khai theo mô hình container hóa sử dụng nền tảng Docker.

Hệ điều hành: Linux

Nền tảng Container: Docker Engine

Model name: Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz

CPU(s): 12

RAM: 24 GB

Lưu trữ: SSD (Để tối ưu tốc độ đọc/ghi cho cơ sở dữ liệu và Kafka)

3.2. Mô tả chức năng

3.2.1. Website dành cho admin

a. Đăng nhập

Hình 3.3. Màn hình đăng nhập website quản trị viên

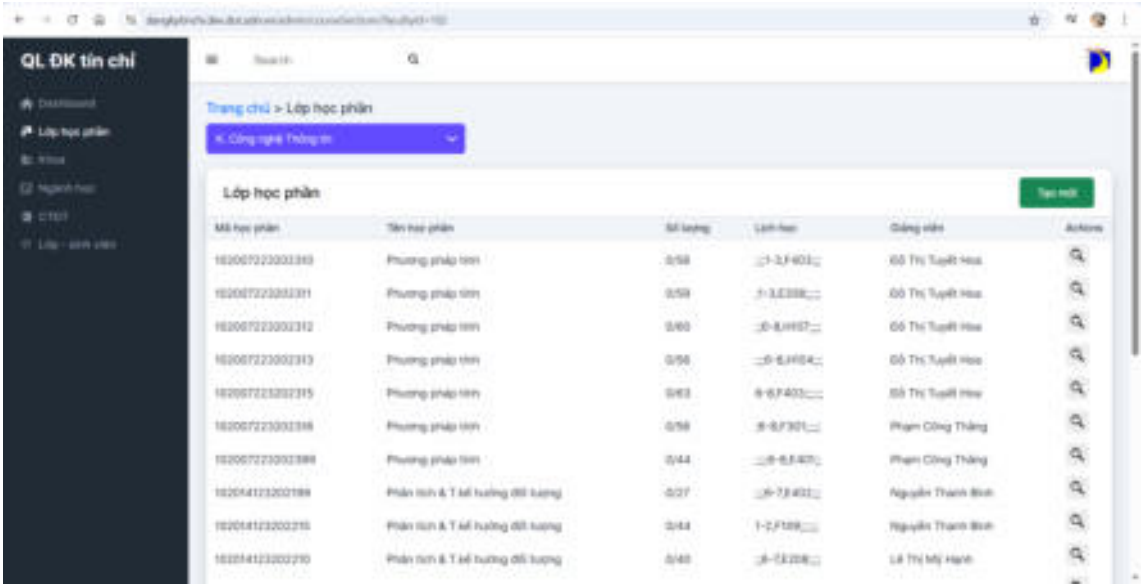
Bảng 3.1. Bảng mô tả màn hình đăng nhập website quản trị viên

Screen	Đăng nhập		
Description	Cho phép quản trị viên đăng nhập thông qua tài khoản Microsoft		
Screen Access	Được điều hướng tới màn hình này nếu quản trị viên chưa xác thực tài khoản		
Screen Content			
Item	Type	Data	Description
Header	Image, Text		Logo, tên trường
Hero image	Image		
Đăng nhập bằng Microsoft	Button		Chuyển hướng trang sang màn hình đăng nhập bằng Microsoft

Screen Actions

Action Name	Description	Success	Failure
Đăng nhập với tài khoản Microsoft	Quản trị viên có thể chọn nút Đăng nhập bằng Microsoft để đăng nhập qua tài khoản Microsoft	Chuyển hướng đến trang đang nhập bằng tài khoản Microsoft Quản trị viên được điều hướng đến trang chủ nếu đăng nhập thành công	Chuyển hướng về trang đăng nhập nếu tài khoản không có trong hệ thống

b. Quản lý lớp học phần



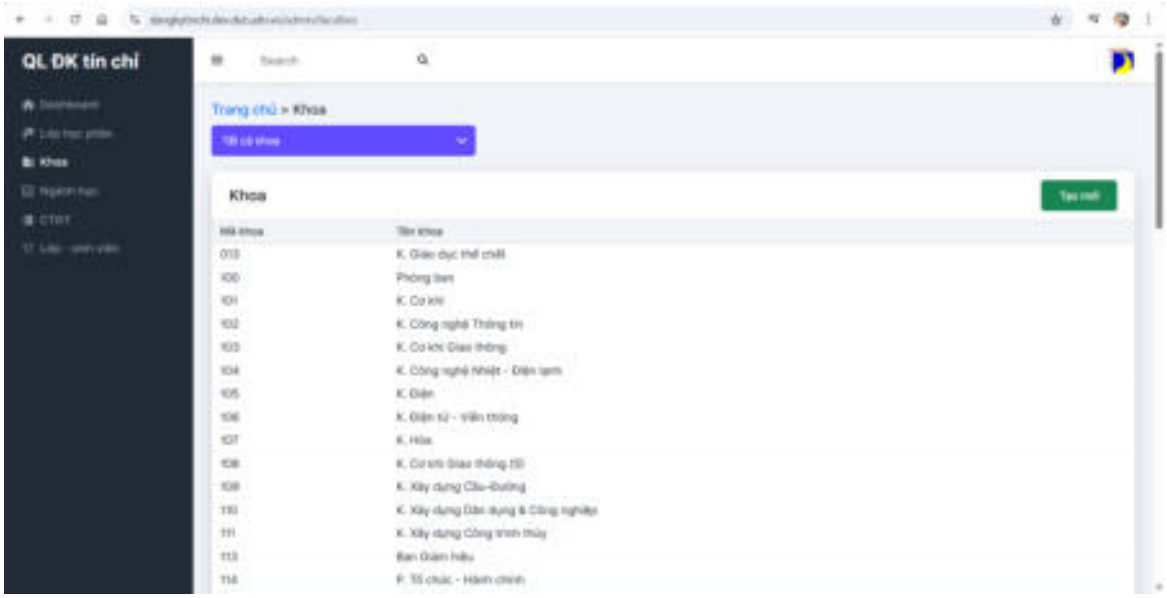
Hình 3.4. Màn hình quản lý lớp học phần

Bảng 3.2. Bảng mô tả màn hình quản lý lớp học phần

Screen	Quản lý lớp học phần		
Description	Cho phép quản trị viên quản lý danh sách lớp học phần trên hệ thống. Có thể xem danh sách sinh viên đã đăng ký vào lớp học phần.		
Screen Access	Người dùng nhấn vào nút “Lớp học phần” ở thanh sidebar bên trái		
Screen Content			
Item	Type	Data	Description
Chọn khoa	Dropdown		Lọc danh sách lớp học phần theo khoa
Tạo mới	Button		Nút bấm để chuyển hướng sang trang upload dữ liệu lớp học phần từ file Excel
Xem chi tiết	Button		Xem danh sách sinh viên đã đăng ký vào lớp học phần đã chọn
Bảng lớp học phần	Table		Bảng hiển thị danh sách lớp học phần trên hệ thống

Screen Actions			
Action Name	**Description**	**Success**	**Failure**
Chọn "Khoa"	Khi người dùng chọn "Khoa" từ dropdown thì danh sách lớp học phần sẽ hiển thị theo điều kiện tìm kiếm.	Bảng danh sách lớp học phần sẽ hiển thị theo điều kiện tìm kiếm.	Hiện thông báo "Lỗi lấy dữ liệu."
Nhấn nút "Tạo mới"	Khi người dùng nhấn vào nút "Tạo mới" thì sẽ chuyển hướng sang trang upload dữ liệu lớp học phần từ file Excel.	Chuyển hướng sang trang upload dữ liệu lớp học phần từ file Excel.	
Nhấn vào biểu tượng "Xem chi tiết"	Khi người dùng nhấn vào biểu tượng "Xem chi tiết" sẽ chuyển hướng sang trang xem danh sách sinh viên đã đăng ký vào lớp học phần đã chọn.	Chuyển hướng sang trang xem danh sách sinh viên đã đăng ký vào lớp học phần đã chọn.	

c. Quản lý khoa

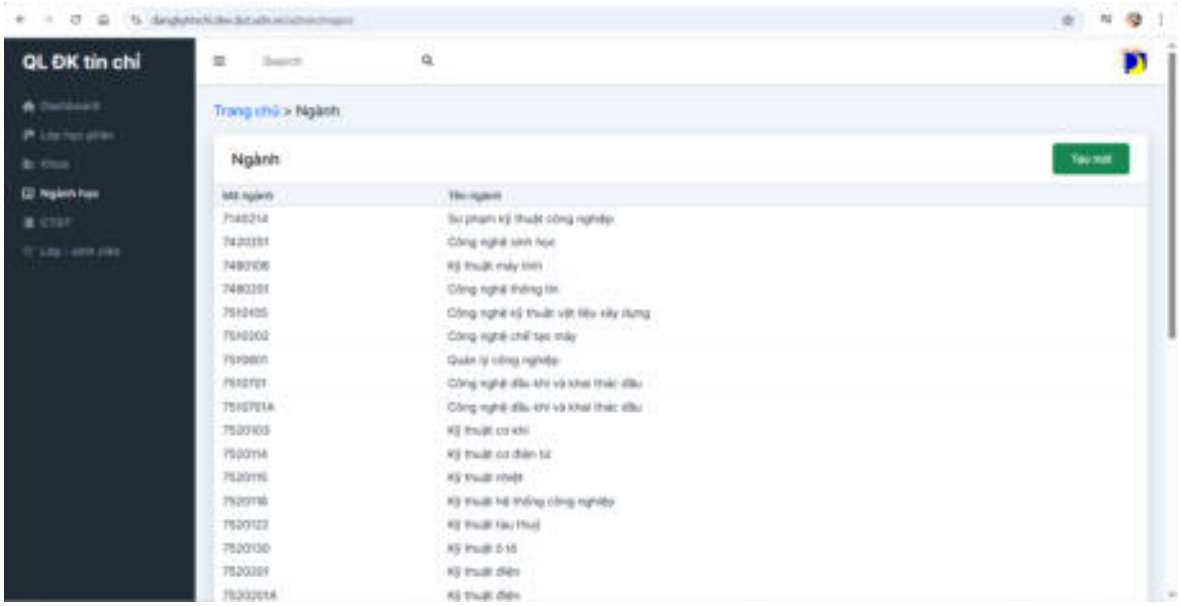


Hình 3.5. Màn hình quản lý khoa

Bảng 3.3. Bảng mô tả màn hình quản lý khoa

Screen	Quản lý khoa		
Description	Cho phép quản trị viên quản lý danh sách khoa trên hệ thống.		
Screen Access	Người dùng nhấn vào nút "Khoa" ở thanh sidebar bên trái		
Screen Content			
Item	Type	Data	Description
Chọn loại khoa	Dropdown		Lọc danh sách khoa theo "Trong trường", "Ngoài trường"
Tạo mới	Button		Nút bấm để chuyển hướng sang trang upload dữ liệu khoa từ file Excel
Bảng khoa	Table		Bảng hiển thị danh sách khoa trên hệ thống
Screen Actions			
Action Name	Description	Success	Failure
Chọn loại "Loại khoa"	Khi người dùng chọn "Loại khoa" từ dropdown thì danh sách khoa sẽ hiển thị theo điều kiện tìm kiếm.	Bảng danh sách khoa sẽ hiển thị theo điều kiện tìm kiếm.	Hiện thông báo "Lỗi lấy dữ liệu."
Nhấn nút "Tạo mới"	Khi người dùng nhấn vào nút "Tạo mới" thì sẽ chuyển hướng sang trang upload dữ liệu khoa từ file Excel.	Chuyển hướng sang trang upload dữ liệu khoa từ file Excel.	

d. Quản lý ngành học

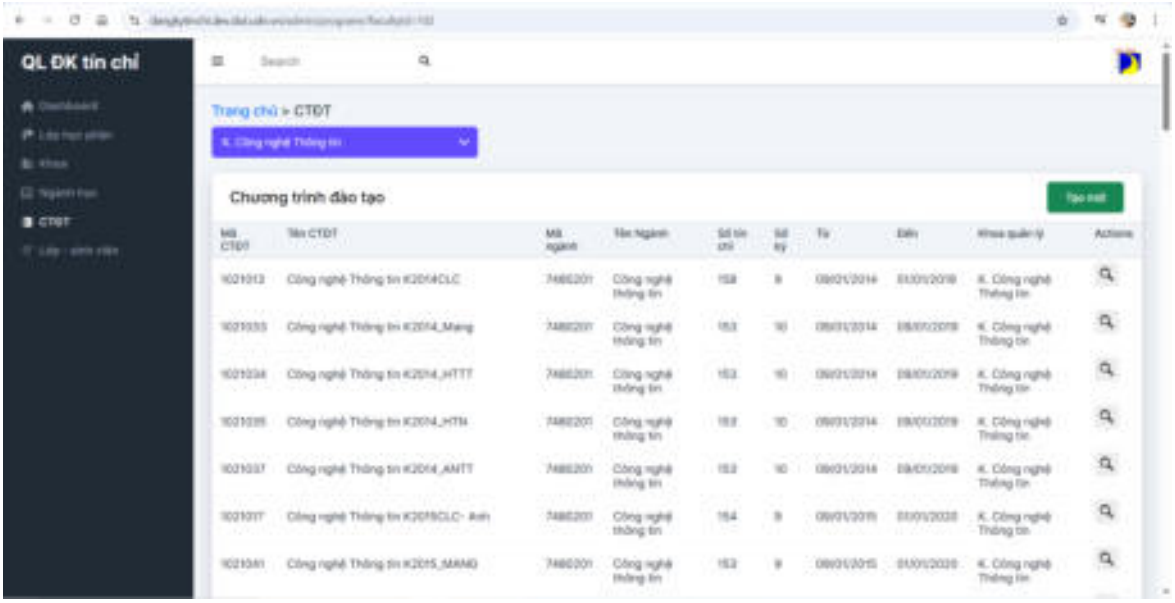


Hình 3.6. Màn hình quản lý ngành học

Bảng 3.4. Bảng mô tả màn hình quản lý ngành học

Screen	Quản lý ngành học		
Description	Cho phép quản trị viên quản lý danh sách ngành học trên hệ thống.		
Screen Access	Người dùng nhấn vào nút "Ngành học" ở thanh sidebar bên trái		
Screen Content			
Item	Type	Data	Description
Tạo mới	Button		Nút bấm để chuyển hướng sang trang upload dữ liệu ngành học từ file Excel
Bảng ngành học	Table		Bảng hiển thị danh sách ngành học trên hệ thống
Screen Actions			
Action Name	Description	Success	Failure
Nhấn nút "Tạo mới"	Khi người dùng nhấn vào nút "Tạo mới" thì sẽ chuyển hướng sang trang upload dữ liệu ngành học từ file Excel.	Chuyển hướng sang trang upload dữ liệu ngành học từ file Excel.	

e. Quản lý chương trình đào tạo



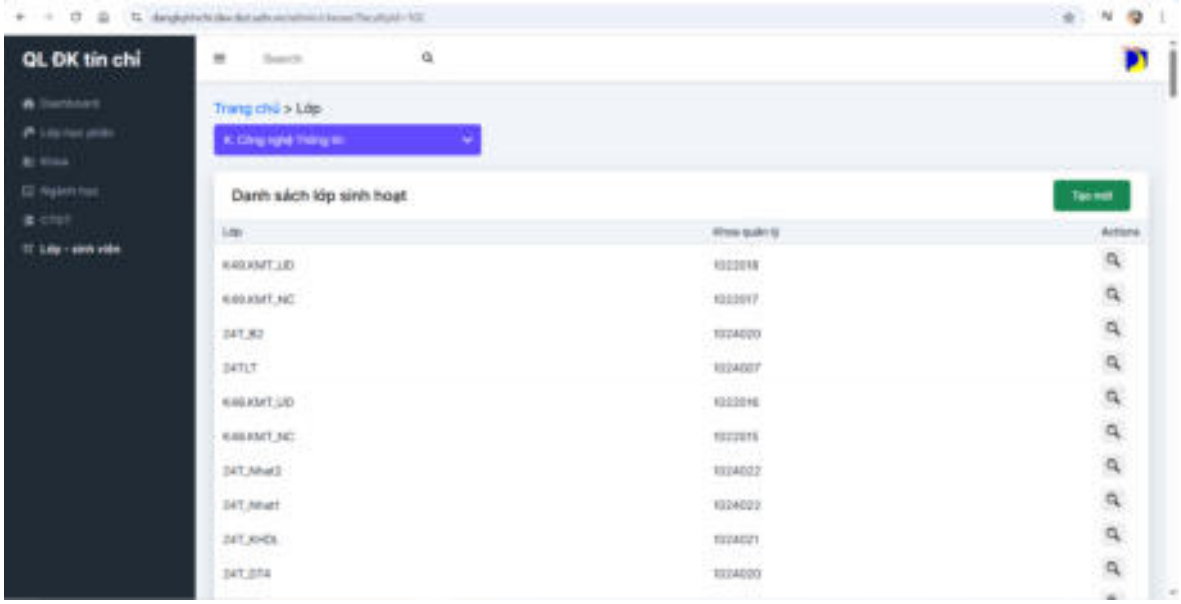
Hình 3.7. Màn hình quản lý chương trình đào tạo

Bảng 3.5. Bảng mô tả màn hình quản lý chương trình đào tạo

Screen	Quản lý chương trình đào tạo		
Description	Cho phép quản trị viên quản lý danh sách chương trình đào tạo trên hệ thống. Có thể xem danh sách học phần của chương trình đào tạo.		
Screen Access	Người dùng nhấn vào nút "CTĐT" ở thanh sidebar bên trái		
Screen Content			
Item	Type	Data	Description
Chọn khoa	Dropdown		Lọc danh sách chương trình đào tạo theo khoa
Tạo mới	Button		Nút bấm để chuyển hướng sang trang upload dữ liệu chương trình đào tạo từ file Excel
Xem chi tiết	Button		Xem danh sách các học phần thuộc chương trình đào tạo đó
Bảng chương trình đào tạo	Table		Bảng hiển thị danh sách chương trình đào tạo trên hệ thống

Screen Actions			
Action Name	**Description**	**Success**	**Failure**
Chọn "Khoa"	Khi người dùng chọn "Khoa" từ dropdown thì danh sách chương trình đào tạo sẽ hiển thị theo điều kiện tìm kiếm.	Bảng danh sách chương trình đào tạo sẽ hiển thị theo điều kiện tìm kiếm.	Hiện thông báo "Lỗi lấy dữ liệu."
Nhấn nút "Tạo mới"	Khi người dùng nhấn vào nút "Tạo mới" thì sẽ chuyển hướng sang trang upload dữ liệu chương trình đào tạo từ file Excel.	Chuyển hướng sang trang upload dữ liệu chương trình đào tạo từ file Excel.	
Nhấn vào biểu tượng "Xem chi tiết"	Khi người dùng nhấn vào biểu tượng "Xem chi tiết" sẽ chuyển hướng sang trang xem danh sách học phần của chương trình đào tạo đó.	Chuyển hướng sang trang xem danh sách học phần của chương trình đào tạo đó.	

f. Quản lý lớp sinh hoạt



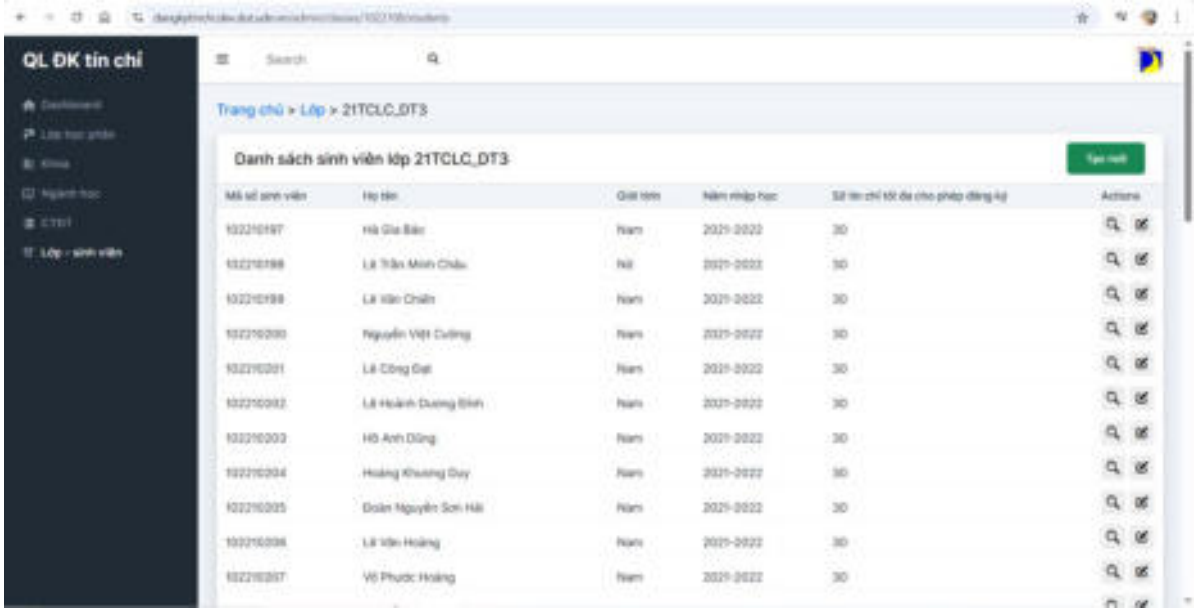
Hình 3.8. Màn hình quản lý lớp sinh hoạt

Bảng 3.6. Bảng mô tả màn hình quản lý lớp sinh hoạt

Screen	Quản lý lớp sinh hoạt		
Description	Cho phép quản trị viên quản lý danh sách lớp sinh hoạt trên hệ thống.		
Screen Access	Người dùng nhấn vào nút "Lớp – Sinh viên" ở thanh sidebar bên trái		
Screen Content			
Item	Type	Data	Description
Chọn khoa	Dropdown		Lọc danh sách lớp sinh hoạt theo khoa
Tạo mới	Button		Nút bấm để chuyển hướng sang trang upload dữ liệu lớp sinh hoạt từ file Excel
Xem chi tiết	Button		Xem danh sách sinh viên trong lớp sinh hoạt đó
Bảng lớp sinh hoạt	Table		Bảng hiển thị danh sách lớp sinh hoạt trên hệ thống
Screen Actions			
Action Name	Description	Success	Failure
Chọn "Khoa"	Khi người dùng chọn "Khoa" từ dropdown thì danh sách lớp sinh hoạt sẽ hiển thị theo điều kiện tìm kiếm.	Bảng danh sách lớp sinh hoạt sẽ hiển thị theo điều kiện tìm kiếm.	Hiện thông báo "Lỗi lấy dữ liệu."
Nhấn nút "Tạo mới"	Khi người dùng nhấn vào nút "Tạo mới" thì sẽ chuyển hướng sang trang upload dữ liệu lớp sinh hoạt từ file Excel.	Chuyển hướng sang trang upload dữ liệu lớp sinh hoạt từ file Excel.	
Nhấn vào biểu tượng "Xem chi tiết"	Khi người dùng nhấn vào biểu tượng "Xem chi tiết" sẽ chuyển hướng sang	Chuyển hướng sang trang xem danh sách sinh	

	trang xem danh sách sinh viên trong lớp sinh hoạt đó.	viên trong lớp sinh hoạt đó.	

g. Quản lý sinh viên



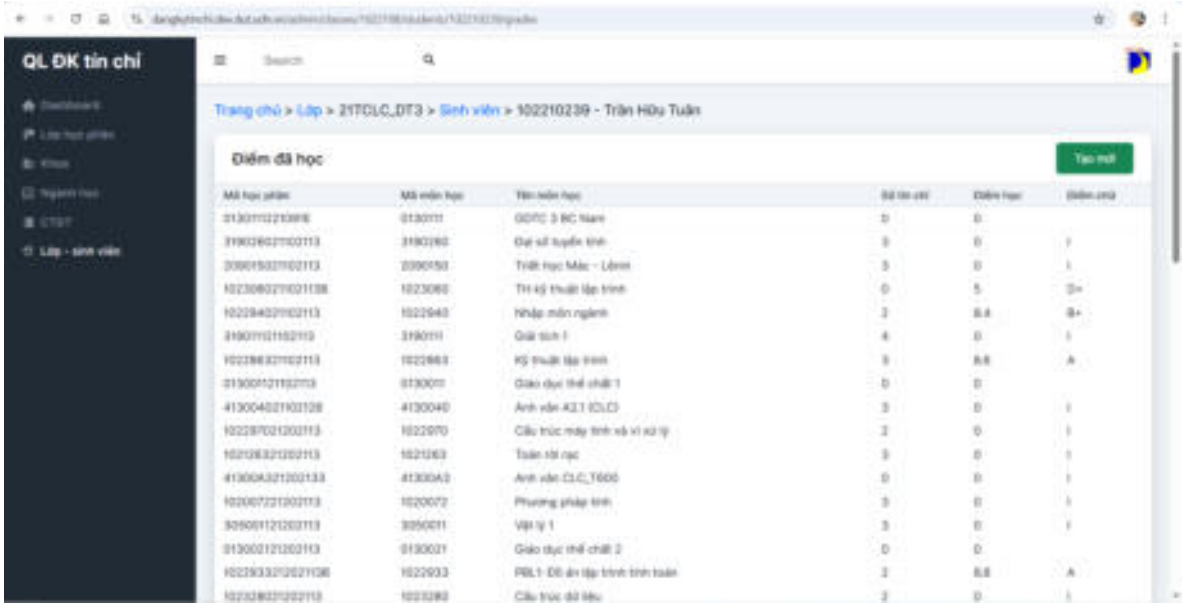
Hình 3.9. Màn hình quản lý sinh viên

Bảng 3.7. Bảng mô tả màn hình quản lý sinh viên

Screen	Quản lý sinh viên		
Description	Cho phép quản trị viên quản lý danh sách sinh viên trên hệ thống.		
Screen Access	Người dùng nhấn vào nút "Xem chi tiết" lớp ở màn hình "Lớp sinh hoạt"		
Screen Content			
Item	Type	Data	Description
Tạo mới	Button		Nút bấm để chuyển hướng sang trang upload dữ liệu sinh viên từ file Excel
Xem điểm học	Button		Xem điểm đã học của 1 sinh viên
Cập nhật số tín chỉ tối đa	Button		Cập nhật số tín chỉ tối đa của 1 sinh viên

Bảng sinh viên	Table		Bảng hiển thị danh sách sinh viên của 1 lớp sinh hoạt trên hệ thống
Screen Actions			
Action Name	**Description**	**Success**	**Failure**
Nhấn nút "Tạo mới"	Khi người dùng nhấn vào nút "Tạo mới" thì sẽ chuyển hướng sang trang upload dữ liệu sinh viên từ file Excel.	Chuyển hướng sang trang upload dữ liệu sinh viên từ file Excel.	
Nhấn vào biểu tượng "Xem điểm học"	Khi người dùng nhấn vào biểu tượng "Xem điểm học" sẽ chuyển hướng sang trang xem điểm đã học của sinh viên đó.	Chuyển hướng sang trang xem điểm đã học của sinh viên đó.	
Nhấn vào biểu tượng "Cập nhật số tín chỉ tối đa"	Khi người dùng nhấn vào biểu tượng "Cập nhật số tín chỉ tối đa" sẽ chuyển hướng sang cập nhập số tín chỉ tối đa của sinh viên đó.	Chuyển hướng sang cập nhập số tín chỉ tối đa của sinh viên đó	

h. Quản lý điểm đã học

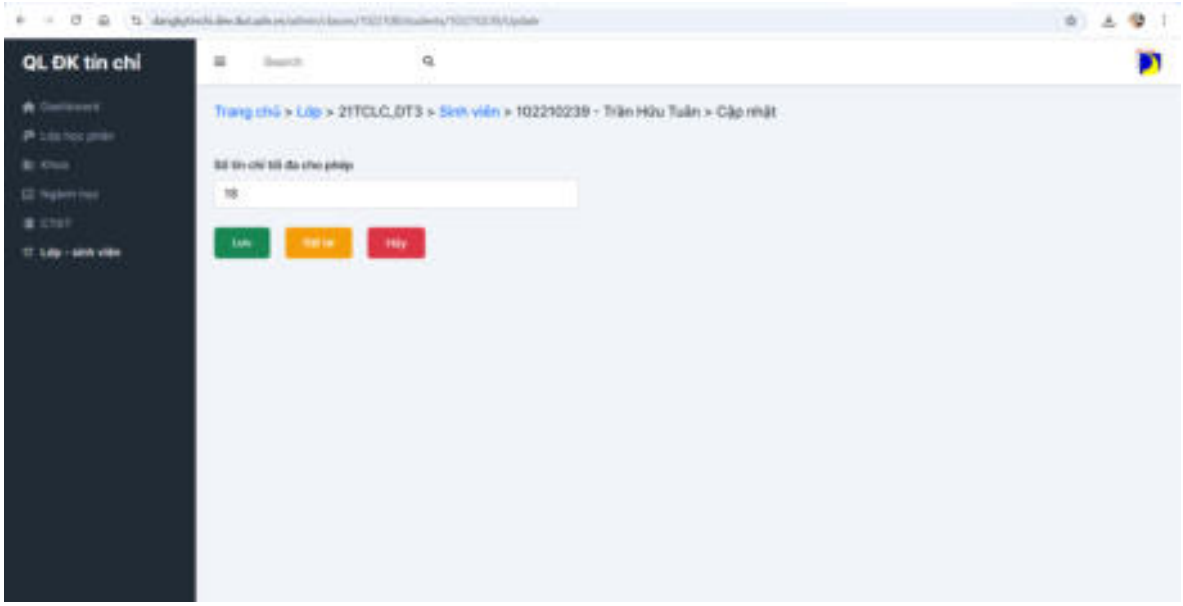


Hình 3.10. Màn hình quản lý điểm đã học

Bảng 3.8. Bảng mô tả màn hình quản lý điểm đã học

Screen	Quản lý điểm đã học		
Description	Cho phép quản trị viên quản lý danh sách điểm đã học của từng sinh viên trên hệ thống.		
Screen Access	Người dùng nhấn vào nút "Xem điểm học" ở màn hình "Sinh viên"		
Screen Content			
Item	Type	Data	Description
Tạo mới	Button		Nút bấm để chuyển hướng sang trang upload dữ liệu điểm đã học từ file Excel
Bảng điểm	Table		Bảng hiển thị điểm đã học của 1 sinh viên trên hệ thống
Screen Actions			
Action Name	Description	Success	Failure
Nhấn nút "Tạo mới"	Khi người dùng nhấn vào nút "Tạo mới" thì sẽ chuyển hướng sang trang upload dữ liệu sinh viên từ file Excel.	Chuyển hướng sang trang upload dữ liệu sinh viên từ file Excel.	

i. Cập nhập số tín chỉ tối đa cho phép của sinh viên



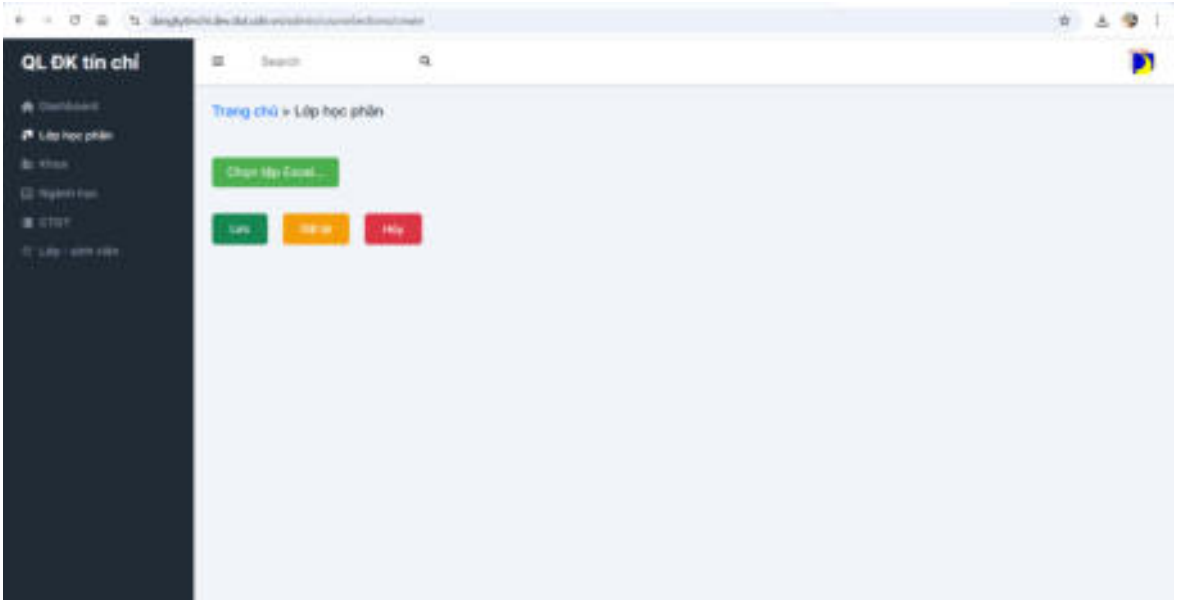
Hình 3.11. Màn hình cập nhập số tín chỉ tối đa cho phép của sinh viên

Bảng 3.9. Bảng mô tả màn hình cập nhập số tín chỉ tối đa cho phép của sinh viên

Screen	Cập nhật số tín chỉ tối đa cho phép của sinh viên		
Description	Cho phép quản trị viên cập nhật số tín chỉ tối đa cho phép của sinh viên		
Screen Access	Người dùng nhấn vào nút "Cập nhật số tín chỉ tối đa" ở màn hình "Sinh viên"		
Screen Content			
Item	Type	Data	Description
Số tín chỉ tối đa cho phép	Text Field		Nhập vào số tín chỉ tối đa cho phép
Screen Actions			

Action Name	Description	Success	Failure
Nhấn nút "Lưu"	Khi người dùng nhấn vào nút "Lưu" thì sẽ cập nhật số tín chỉ tối đa cho phép của 1 sinh viên.	Thông báo cập nhật dữ liệu thành công	Thông báo cập nhật dữ liệu thất bại
Nhấn nút "Đặt lại"	Đặt lại dữ liệu về giá trị ban đầu	Đặt lại dữ liệu về giá trị ban đầu	
Nhấn nút "Hủy"	Trở lại màn hình trước	Trở lại màn hình trước	

j. Upload dữ liệu từ file Excel



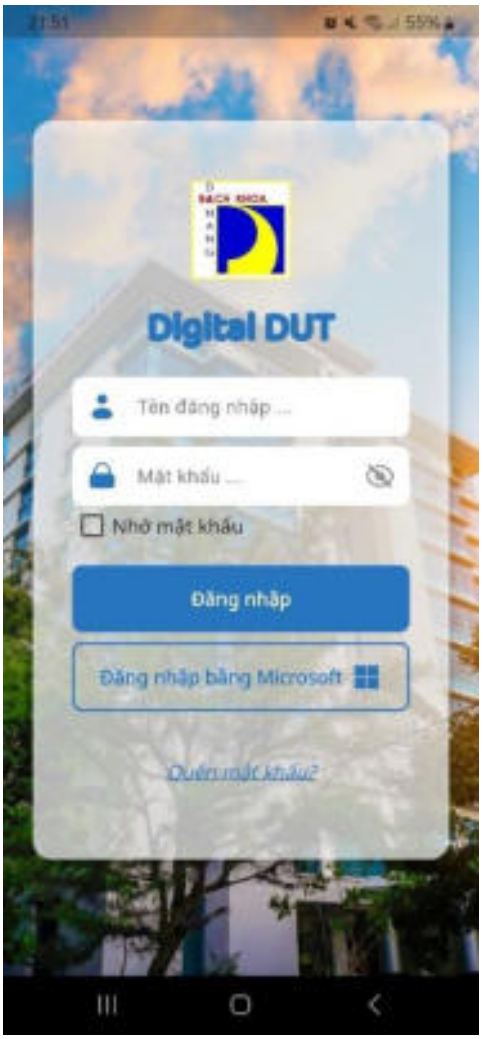
Hình 3.12. Màn hình upload dữ liệu từ file Excel

Bảng 3.10. Bảng mô tả màn hình upload dữ liệu từ file Excel

Screen	Upload dữ liệu từ file Excel		
Description	Cho phép quản trị viên upload dữ liệu từ file Excel		
Screen Access	Người dùng nhấn vào nút "Tạo mới"		
Screen Content			
Item	Type	Data	Description
File	File		Chọn file Excel từ thiết bị
Screen Actions			
Action Name	Description	Success	Failure
Nhấn nút "Lưu"	Khi người dùng nhấn vào nút "Lưu" thì sẽ tạo dữ liệu.	Thông báo "Tạo dữ liệu thành công"	Thông báo "Tạo dữ liệu thất bại"
Nhấn nút "Đặt lại"	Đặt lại dữ liệu về giá trị ban đầu	Đặt lại dữ liệu về giá trị ban đầu	
Nhấn nút "Hủy"	Trở lại màn hình trước	Trở lại màn hình trước	

3.2.2. Ứng dụng di động

a. Đăng nhập



Hình 3.13. Màn hình đăng nhập

Bảng 3.11. Bảng mô tả màn hình đăng nhập

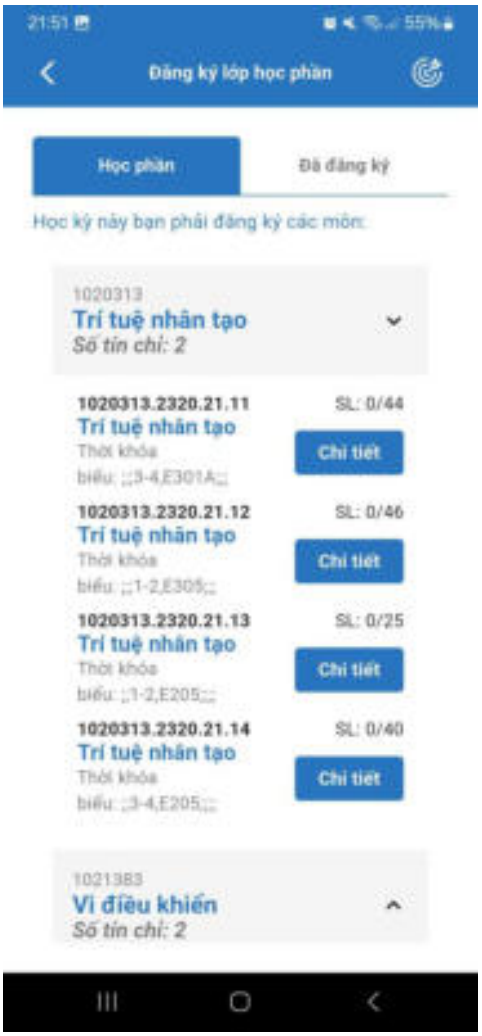
Screen	Đăng nhập		
Description	Cho phép người dùng đăng nhập thông qua Tên đăng nhập/Mật khẩu hoặc tài khoản Microsoft của sinh viên		
Screen Access	Ứng dụng được điều hướng tới màn hình này nếu người dùng chưa xác thực tài khoản		
Screen Content			
Item	Type	Data	Description
Tên đăng nhập	Text field - String (128)		Trường dành cho người dùng nhập Mail của tài khoản Microsoft sinh

			viên
Mật khẩu	Password – MaxString		Trường dành cho người dùng nhập mật khẩu
Đăng nhập	Button		Nút bấm để thực hiện đăng nhập với thông tin đã nhập
Quên mật khẩu	Text Button		Điều hướng người dùng tới màn hình lấy lại mật khẩu
Đăng nhập bằng Microsoft	Button		Tùy chọn đăng nhập thông qua tài khoản Microsoft sinh viên
Nhớ mật khẩu	Check Box		Tùy chọn nhớ mật khẩu
Screen Actions			

ActionName	Description	Success	Failure
Đăng nhập	Khi người dùng nhập thông tin vào các trường Tên đăng nhập hoặc Mật khẩu, hệ thống sẽ kiểm tra tính hợp lệ ngay khi dữ liệu được nhập: - Nếu bất kỳ trường nào để trống hoặc không hợp lệ, hệ thống sẽ hiển thị thông báo lỗi màu đỏ bên dưới trường đó, yêu cầu người dùng sửa đổi dữ liệu cho hợp lệ.	Người dùng được điều hướng vào màn hình chính của ứng dụng	Hiển thị thông báo lỗi nếu thông tin đăng nhập không chính xác hoặc không tồn tại
Đăng nhập với tài khoản Microsoft	Người dùng có thể chọn nút Đăng nhập bằng Microsoft để đăng nhập qua tài khoản Microsoft sinh viên	Người dùng được điều hướng đến màn hình đăng nhập bằng tài khoản Microsoft	Hiển thị thông báo lỗi nếu tài khoản Microsoft không hợp lệ hoặc không kết nối được với hệ thống
Quên mật khẩu	Khi người dùng quên mật khẩu tài khoản cá nhân có	Người dùng được điều	

	thể nhấn vào liên kết Forgot password?	hướng tới màn hình khôi phục mật khẩu	
Nhớ mật khẩu	Khi người dùng muốn nhớ Tên đăng nhập/Mật khẩu	Ghi nhớ Tên đăng nhập/Mật khẩu	

b. Danh sách các lớp học phần có thể đăng ký



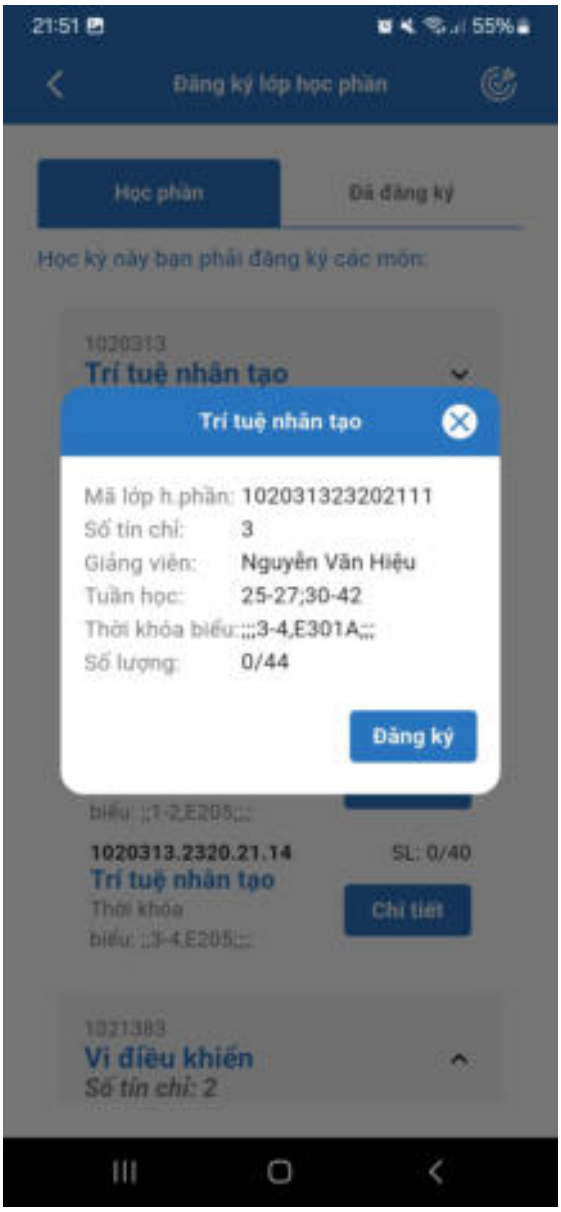
Hình 3.14. Màn hình hiển thị danh sách lớp học phần có thể đăng ký

Bảng 3.12. Bảng mô tả màn hình hiển thị danh sách lớp học phần có thể đăng ký

Screen	Danh sách các lớp học phần có thể đăng ký		
Description	Hiển thị danh sách lớp học phần có thể đăng ký của sinh viên đã đăng nhập vào ứng dụng. Các học phần mô tả tên môn học, mã lớp học phần, thời khóa biểu, số lượng/số lượng tối đa của 1 lớp học phần		
Screen Access	Người dùng nhấn vào mục Đăng ký tín chỉ trong màn hình Trang chủ và tab Học phần.		
Screen Content			
Item	Type	Data	Description
Học phần	Tab		Điều hướng đến tab Danh sách lớp học phần có thể đăng ký
Course Card	Card		Hiển thị thông tin học phần, bao gồm mã học phần, tên học phần và số tín chỉ.
Course Section Card	Card		Hiển thị thông tin lớp học phần, bao gồm mã lớp học phần, thời khóa biểu, số lượng/số lượng tối đa.
Chi tiết	Button		Hiển thị Dialog mô tả chi tiết lớp học phần có thể đăng ký.
Screen Actions			
Action Name	Description	Success	Failure
Xem điểm đã học	Người dùng nhấn vào biểu tượng Điểm đã học để xem điểm đã học các học phần trước đó.	Điều hướng đến trang Xem điểm đã học.	
Hiển thị/ẩn danh sách lớp học phần của 1 học phần	Người dùng nhấn vào thẻ học phần để hiển thị/ẩn danh sách lớp học phần của học phần đó.	Hiển thị/ẩn danh sách lớp học phần của 1 học phần.	

Xem chi tiết lớp học phần có thể đăng ký.	Người dùng nhấn vào Chi tiết trong 1 lớp học phần.	Hiển thị Dialog mô tả chi tiết lớp học phần có thể đăng ký.	

c. Đăng ký lớp học phần

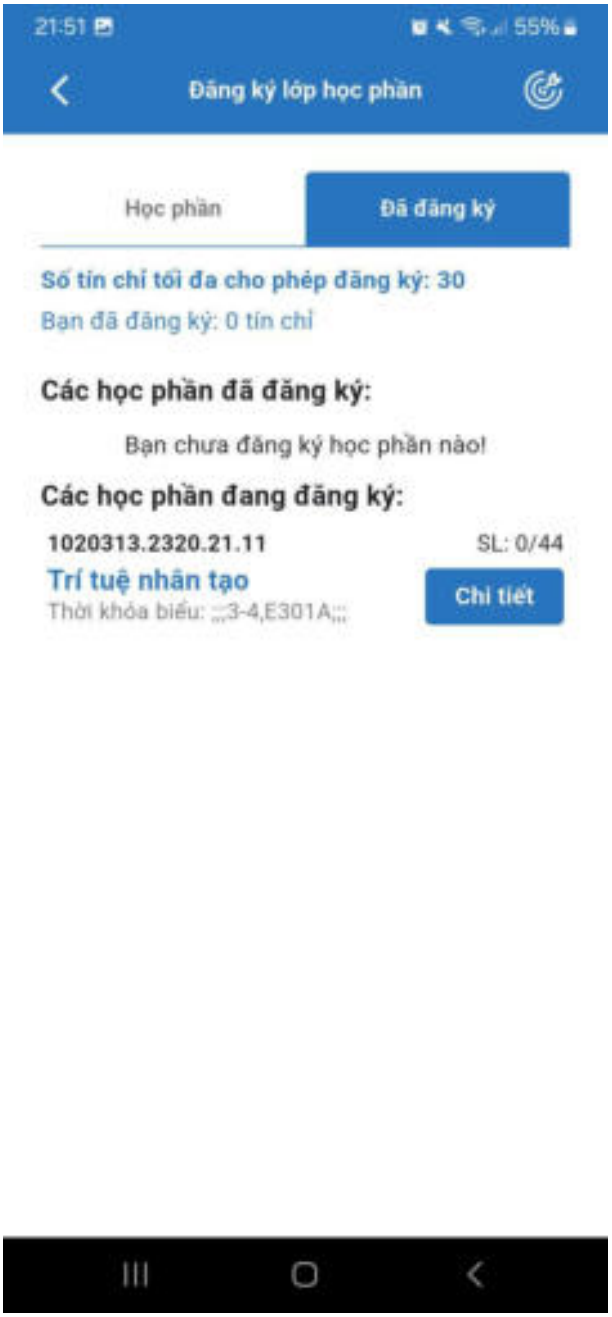


Hình 3.15. Màn hình đăng ký lớp học phần

Bảng 3.13. Bảng mô tả màn hình đăng ký lớp học phần

Screen	Dialog mô tả thông tin chi tiết lớp học phần		
Description	Hiển thị thông tin chi tiết lớp học phần bao gồm: tên học phần, mã lớp học phần, số tín chỉ, giảng viên, tuần học, thời khóa biểu, số lượng sinh viên đã đăng ký/số lượng tối đa cho phép.		
Screen Access	Người dùng nhấn vào Chi tiết lớp học phần.		
Screen Content			
Item	Type	Data	Description
Title	Text		Tên học phần
Content	List View		Hiển thị thông tin học phần, bao gồm: mã lớp học phần, số tín chỉ, giảng viên, tuần học, thời khóa biểu, số lượng sinh viên đã đăng ký/số lượng tối đa cho phép.
Đăng ký	Button		Đăng ký lớp học phần hiện tại.
Screen Actions			
Action Name	Description	Success	Failure
Đóng dialog mô tả thông tin chi tiết lớp học phần	Người dùng nhấn vào biểu tượng Đóng để đóng dialog mô tả thông tin chi tiết lớp học phần.	Đóng dialog.	
Đăng ký lớp học phần	Người dùng nhấn vào Đăng ký.	Đợi xử lý đăng ký và tải lại trang.	Thông báo lớp học phần đã được đăng ký hoặc đang xử lý đăng ký.

d. Danh sách lớp học phần đã đăng ký



Hình 3.16. Màn hình hiển thị danh sách lớp học phần đã đăng ký

Bảng 3.14. Bảng mô tả màn hình hiển thị danh sách lớp học phần đã đăng ký

Screen	Danh sách các lớp học phần đã đăng ký
Description	Hiển thị danh sách lớp học phần đã và đang đăng ký của sinh viên. Các học phần mô tả tên môn học, mã lớp học phần, thời khóa biểu, số lượng/số lượng tối đa của 1 lớp học phần.
Screen Access	Người dùng nhấn vào mục Đăng ký tín chỉ trong màn hình Trang chủ và tab Đã đăng ký.

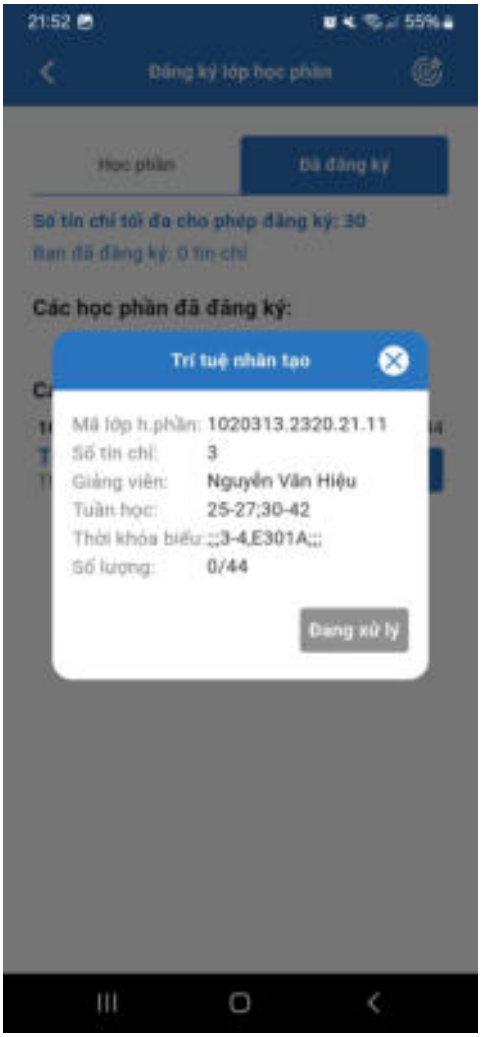
Screen Content

Item	Type	Data	Description
Đã đăng ký	Tab		Điều hướng đến tab Danh sách lớp học phần đã đăng ký
Số tín chỉ cho phép tối đa	Text		Hiển thị số lượng số tín chỉ tối đa có thể đăng ký.
Số tín chỉ đã đăng ký	Text		Hiển thị số lượng số tín chỉ đã đăng ký.
Course Section Card	Card		Hiển thị thông tin lớp học phần, bao gồm mã lớp học phần, thời khóa biểu, số lượng/số lượng tối đa.
Chi tiết	Button		Hiển thị Dialog mô tả chi tiết lớp học phần đã đăng ký.

Screen Actions

Action Name	Description	Success	Failure
Xem điểm đã học	Người dùng nhấn vào biểu tượng Điểm đã học để xem điểm đã học các học phần trước đó.	Điều hướng đến trang Xem điểm đã học.	
Xem chi tiết lớp học phần.	Người dùng nhấn vào Chi tiết trong 1 lớp học phần.	Hiển thị Dialog mô tả chi tiết lớp học phần.	

e. Đang xử lý đăng ký lớp học phần



Hình 3.17. Màn hình đang dăng ký lớp học phần

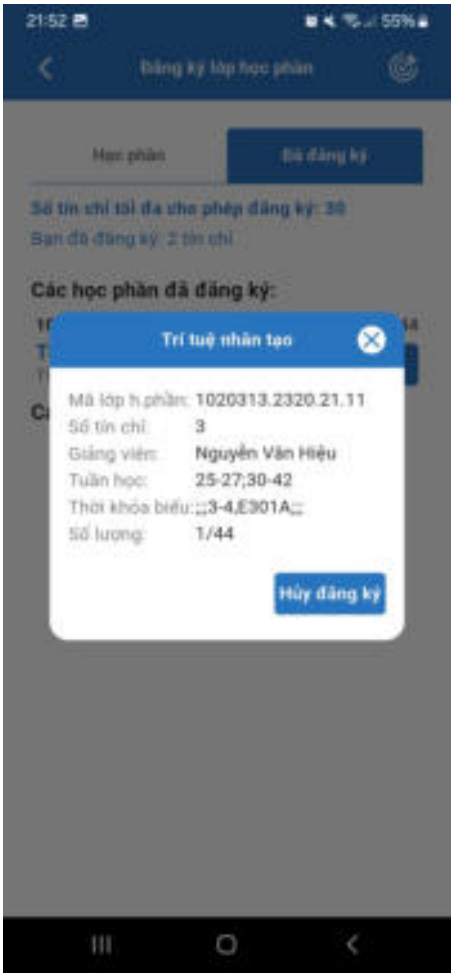
Bảng 3.15. Bảng mô tả màn hình đang dăng ký lớp học phần

Screen	Dialog mô tả thông tin chi tiết lớp học phần đang xử lý đăng ký		
Description	Hiển thị thông tin chi tiết lớp học phần bao gồm: tên học phần, mã lớp học phần, số tín chỉ, giảng viên, tuần học, thời khóa biểu, số lượng sinh viên đã đăng ký/số lượng tối đa cho phép.		
Screen Access	Người dùng nhấn vào Chi tiết lớp học phần.		
Screen Content			
Item	Type	Data	Description
Title	Text		Tên học phần
Content	List View		Hiển thị thông tin học phần, bao gồm: mã lớp

			học phần, số tín chỉ, giảng viên, tuần học, thời khóa biểu, số lượng sinh viên đã đăng ký/số lượng tối đa cho phép.
Đang xử lý	Disable Button		Học phần đang được xử lý đăng ký.

Screen Actions			
Action Name	Description	Success	Failure
Đóng dialog mô tả thông tin chi tiết lớp học phần	Người dùng nhấn vào biểu tượng Đóng để đóng dialog mô tả thông tin chi tiết lớp học phần.	Đóng dialog.	

f. Hủy đăng ký lớp học phần

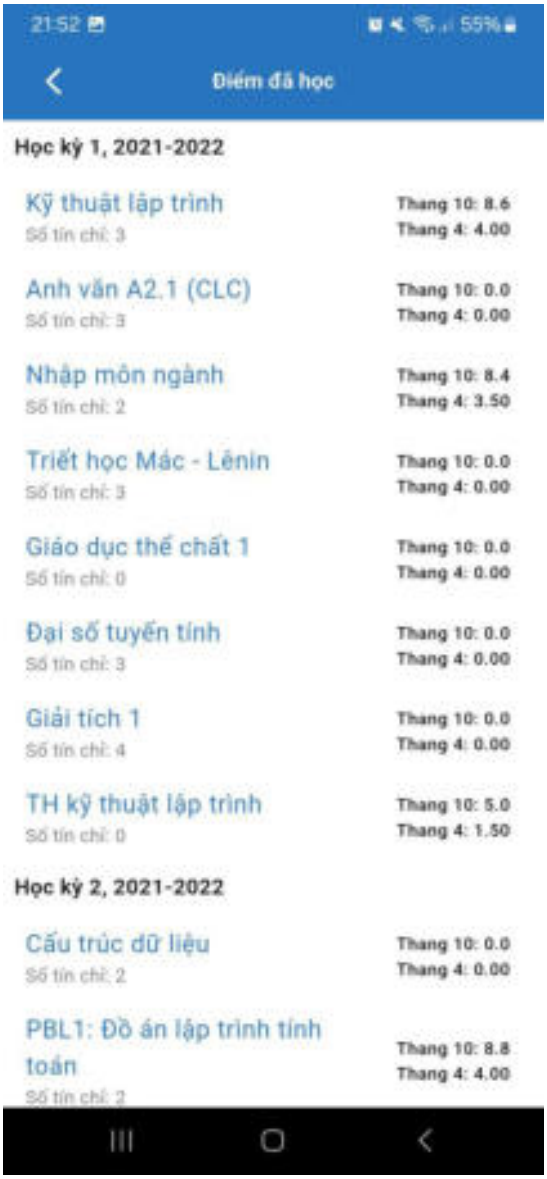


Hình 3.18. Màn hình hủy đăng ký lớp học phần

Bảng 3.16. Bảng mô tả màn hình hủy đăng ký lớp học phần

Screen	Dialog mô tả thông tin chi tiết lớp học phần đã được đăng ký		
Description	Hiển thị thông tin chi tiết lớp học phần bao gồm: tên học phần, mã lớp học phần, số tín chỉ, giảng viên, tuần học, thời khóa biểu, số lượng sinh viên đã đăng ký/số lượng tối đa cho phép.		
Screen Access	Người dùng nhấn vào biểu tượng Xem điểm đã học.		
Screen Content			
Item	Type	Data	Description
Kỳ học	Text		Thông tin kỳ học (kỳ 1, kỳ 2, kỳ hè) và năm học
Tên học phần	Text		Tên lớp học phần.
Số tín chỉ	Text		Số tín chỉ.
Điểm học	Text		Điểm học theo thang 10, thang 4.

g. Xem điểm đã học



Hình 3.19. Màn hình xem điểm đã học

Bảng 3.17. Bảng mô tả màn hình xem điểm đã học

Screen	Hiển thị điểm của cá lớp học phần đã học		
Description	Hiển thị điểm của cá lớp học phần đã học bao gồm: thông tin môn học, điểm học theo thang điểm.		
Screen Access	Người dùng nhấn vào Chi tiết lớp học phần.		
Screen Content			
Item	Type	Data	Description
Title	Text		Tên học phần
Content	List View		Hiển thị thông tin học phần, bao gồm: mã lớp học phần, số tín chỉ, giảng viên, tuần học, thời khóa biểu, số lượng sinh viên đã đăng ký/số lượng tối đa cho phép.
Hủy đăng ký	Button		Hủy đăng ký lớp học phần hiện tại.
Screen Actions			
Action Name	Description	Success	Failure
Đóng dialog mô tả thông tin chi tiết lớp học phần	Người dùng nhấn vào biểu tượng Đóng để đóng dialog mô tả thông tin chi tiết lớp học phần.	Đóng dialog.	
Hủy đăng ký lớp học phần	Người dùng nhấn vào Hủy đăng ký.	Đợi xử lý hủy đăng ký và tải lại trang.	Thông báo lớp học phần đang xử lý hủy đăng ký.

3.3. Kiểm thử hiệu năng hệ thống đăng ký tín chỉ

Để xác thực hiệu quả của kiến trúc đã đề xuất, một trong những bước quan trọng nhất là thực hiện kiểm thử hiệu năng (Performance Testing). Mục tiêu là để đánh giá khả năng chịu tải của hệ thống trong các kịch bản sử dụng cao điểm, đặc biệt là trong giai đoạn đăng ký học phần, khi hàng nghìn sinh viên truy cập đồng thời.

3.3.1. Xây dựng kịch bản kiểm thử

Locust là một công cụ kiểm thử hiệu năng mã nguồn mở, được lựa chọn cho dự án này vì những lý do sau:

- **Viết kịch bản bằng Python:** Cho phép viết các kịch bản kiểm thử (mô phỏng hành vi người dùng) một cách linh hoạt và dễ dàng bằng ngôn ngữ Python, thay vì phải sử dụng các giao diện đồ họa phức tạp.
- **Hỗ trợ kiểm thử phân tán:** Locust cho phép chạy kiểm thử trên nhiều máy (worker), giúp giả lập một lượng lớn người dùng đồng thời mà không bị giới hạn bởi tài nguyên của một máy duy nhất. Đây là tính năng cốt lõi để tạo ra tải trọng lớn và thực tế.
- **Giao diện Web trực quan:** Cung cấp một giao diện web để theo dõi kết quả kiểm thử theo thời gian thực, với các biểu đồ rõ ràng về thông lượng, thời gian phản hồi, và tỷ lệ lỗi.

a. Mục tiêu Kiểm thử:

Đánh giá khả năng chịu tải của hệ thống đăng ký học phần, cụ thể là API lấy danh sách các lớp học phần có sẵn theo mã sinh viên, vì đây là một trong những thao tác được thực hiện nhiều nhất.

b. Các bước thực hiện:

1. **Thiết lập Môi trường Locust:** Khởi tạo một cụm Locust bao gồm 1 master node để điều phối và 9 worker node để tạo tải (tương ứng với thông số "Workers: 9" trong kết quả).
2. **Xây dựng Kịch bản:** Viết mã Python để mô phỏng một sinh viên thực hiện yêu cầu GET đến API lấy danh sách lớp học phần.
3. **Thực thi và Giám sát:** Chạy kịch bản kiểm thử và theo dõi các chỉ số hiệu năng trên giao diện web của Locust theo thời gian thực.

Hình 3.20. Kết quả của quá trình kiểm thử mô phỏng lấy dữ liệu lớp học phần

3.3.2. Kết quả thu được

Bảng 3.18. Bảng tóm tắt kết quả thu được

Chỉ số	**Kết quả**	**Đánh giá**
Tổng thông lượng (RPS)	~ 5,850 req/s	Hệ thống có khả năng chịu tải cao
Tỷ lệ lỗi (Failures)	~ 1%	Tỷ lệ lỗi thấp trong điều kiện tải nặng

Thông lượng hệ thống (Total Requests per Second - RPS):

- **Phân tích:** Đường màu xanh lá cây trên biểu đồ trên cùng cho thấy số lượng yêu cầu mà hệ thống xử lý thành công mỗi giây. Ngay khi bắt đầu, thông lượng tăng vọt và nhanh chóng đạt đến trạng thái ổn định.
- **Kết luận:** Hệ thống đã duy trì một thông lượng cực kỳ ấn tượng và ổn định ở mức xấp xỉ **5,850 RPS**. Điều này chứng tỏ kiến trúc được thiết kế có khả năng xử lý một khối lượng yêu cầu đồng thời rất lớn, đáp ứng tốt cho kịch bản nhiều sinh viên truy cập cùng lúc. Tỷ lệ lỗi (đường màu đỏ) duy trì ở mức rất thấp, gần như bằng không trong suốt quá trình.

Thời gian Phản hồi (Response Times):

- **Thời gian phản hồi trung vị (50th percentile - đường màu vàng):** Đại diện cho thời gian phản hồi của 50% số yêu cầu nhanh nhất. Chỉ số này duy trì ở mức rất thấp và ổn định, cho thấy phần lớn người dùng sẽ có trải nghiệm cực kỳ nhanh và mượt mà.
- **Thời gian phản hồi ngưỡng 95% (95th percentile - đường màu tím):** Đại diện cho thời gian phản hồi của 95% số yêu cầu (tức là chỉ có 5% số yêu cầu là chậm hơn mức này).

- **Phân tích:** Chỉ số này phần lớn cũng duy trì ở mức thấp và ổn định. Tuy nhiên, có một **thời điểm đột biến (spike)** rõ rệt vào khoảng 2:16:45 AM, khi thời gian phản hồi tăng vọt lên trên 30,000 ms (30 giây) rồi nhanh chóng quay trở lại mức bình thường.
- **Lý giải:** Hiện tượng này có thể xảy ra do nhiều nguyên nhân trong một hệ thống chịu tải cao như: quá trình "warm-up" ban đầu của một dịch vụ, một tiến trình dọn dẹp bộ nhớ (garbage collection) trong JVM của Kafka hoặc runtime của các service, hoặc một thao tác khóa tài nguyên tạm thời trong cơ sở dữ liệu. Điều quan trọng là **hệ thống đã tự phục hồi rất nhanh chóng**, cho thấy khả năng phục hồi tốt.

3.3.3. Kết chương

Chương này trình bày một số yêu cầu đối với môi trường triển khai hệ thống đăng ký tín chỉ, quá trình vận hành hệ thống trong thực tế giúp người đọc có cái nhìn tổng quan và cụ thể hơn Hệ thống đăng ký tín chỉ. Xây dựng quy trình kiểm thử hiệu năng của hệ thống và đánh giá kết quả.

CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1. Đánh giá chung về hệ thống

Sau quá trình thiết kế, triển khai và kiểm thử, có thể đưa ra những đánh giá tổng quan về hệ thống đã xây dựng dựa trên kiến trúc Microservices.

4.1.1. Kết quả đạt được

- Việc áp dụng mô hình bất đồng bộ với Kafka và Redis đã giúp hệ thống có khả năng xử lý hàng nghìn yêu cầu mỗi giây. Giao diện người dùng có thời gian phản hồi cực nhanh vì không phải chờ đợi các logic nghiệp vụ phức tạp được xử lý xong.
- Kiến trúc tách rời giữa các service đảm bảo rằng sự cố ở một thành phần sẽ không làm sập toàn bộ hệ thống.
- Nhờ có Kafka lưu trữ lại các sự kiện, nếu CourseRegistrationConsumer bị lỗi và khởi động lại, nó vẫn có thể tiếp tục xử lý các yêu cầu đăng ký đã được gửi đến mà không gây mất mát dữ liệu.
- Khả năng mở rộng linh hoạt đối với các service có lượng traffic cao.
- Việc triển khai các service độc lập giúp cho việc dễ bảo trì, sửa lỗi và thêm tính năng.

4.1.2. Hạn chế

Hệ thống có độ phức tạp trong vận hành cao, yêu cầu có kiến thức về DevOps và các công cụ container hóa.

Tính nhất quán cuối cùng: Vì hệ thống xử lý bất đồng bộ, sinh viên sẽ không nhận được kết quả thành công hay thất bại ngay lập tức. Sẽ có độ trễ nhỏ cho đến khi kết quả được cập nhật.

Xử lý logic của hệ thống đăng ký tín chỉ chưa đầy đủ.

Dịch vụ đăng nhập 1 cửa (SSO) vẫn chưa tối ưu hiệu xuất, khả năng chịu tải.

4.2. Hướng phát triển

Tìm hiểu, thử nghiệm và bổ sung những phương pháp về container hóa, tự động deployment.

Bổ sung thêm thống báo kết quả đăng ký qua email, thông báo đẩy…

Cấu hình để Consumer lấy ra một lô (batch) các sự kiện từ Kafka trong một lần (ví dụ: 100 yêu cầu đăng ký cùng lúc). Sau đó, xử lý lô này và tương tác với cơ sở dữ liệu chỉ một lần.

Tài liệu tham khảo

[1] Google, “Microservice Architecture pattern” [Online]. Available: https://microservices.io/patterns/microservices.html [Accessed 15 03 2025].

[2] Google, “Securing APIs with YARP: Authentication and Authorization in .NET 8 Minimal APIs” [Online]. Available: https://dev.to/leandroveiga/securing-apis-with-yarp-authentication-and-authorization-in-net-8-minimal-apis-2960

[Accessed 01 04 2025].

[3] Google, “Kafka Architecture” [Online]. Available: https://www.geeksforgeeks.org/kafka-architecture

[Accessed 10 04 2025].

[4] Google, “Chiến lược caching (Caching strategies)” [Online]. Available: https://viblo.asia/p/chien-luoc-caching-caching-strategies-zXRJ8jPOVGq [Accessed 10 04 2025].

[5] Google, “Arcturus — Inventory Processing System” [Online]. Available: https://engineering.tiki.vn/arcturus-inventory-processing-system/ [Accessed 20 04 2025].

[6] Google, “Manage an external authentication method in Microsoft Entra ID” [Online]. Available: https://learn.microsoft.com/en-us/entra/identity/authentication/how-to-authentication-external-method-manage

[Accessed 22 03 2025].