

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**



NGUYỄN VĂN MAU

**PHÁT TRIỂN WEBSITE HỌC TẬP TRỰC TUYẾN
DỰA TRÊN KIẾN TRÚC MICROSERVICES**

**ĐỒ ÁN NGÀNH
NGÀNH KHOA HỌC MÁY TÍNH**

TP. HỒ CHÍ MINH, 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH



NGUYỄN VĂN MAU

PHÁT TRIỂN WEBSITE HỌC TẬP TRỰC TUYẾN
DỰA TRÊN KIẾN TRÚC MICROSERVICES

Mã số sinh viên: 2151010218

ĐỒ ÁN NGÀNH
KHOA HỌC MÁY TÍNH

Giảng viên hướng dẫn: DƯƠNG THÁI BẢO

TP. HỒ CHÍ MINH, 2024

LỜI CẢM ƠN

Trong nhiều năm học tập và gắn bó với trường Đại Học Mở. Đối với em đây là khoảng thời gian đáng nhớ nhất trong tuổi học trò, đây cũng là giai đoạn không chỉ giúp em có nhiều kiến thức mà còn rèn luyện cho em sự kiên trì, nhẫn nại vượt qua khó khăn. Bài báo cáo này không chỉ là kết quả của quá trình học tập và rèn luyện mà còn là dịp để em có thể hệ thống hóa và áp dụng những kiến thức mà các thầy cô đã truyền đạt trong quá trình học tập.

Cá nhân em xin được gửi lời tri ân sâu sắc đến Trường Đại Học Mở toàn thể quý thầy cô thuộc Khoa Công nghệ Thông tin. Sự tận tụy trong giảng dạy, cũng như là những kiến thức quý báu mà thầy cô đã truyền đạt, sẽ giúp em không chỉ là hoàn thiện đồ án mà nó còn là hành trang vững chắc cho con đường sự nghiệp phía trước của em.

Em cũng muốn bày tỏ sự biết ơn đặc biệt đến thầy Dương Thái Bảo, người đã không quản ngại thời gian và công sức để tận tình hướng dẫn em trong suốt quá trình thực hiện đồ án. Nhờ sự chỉ dẫn tận tình của thầy, em đã học hỏi được nhiều kiến thức để áp dụng vào thực tế và hoàn thiện đồ án một cách tốt nhất có thể.

Dù đã cố gắng hết mình, nhưng cá nhân em nhận thấy đồ án của mình vẫn có 1 số hạn chế nhất định do nhiều vấn đề cũng như hạn chế về thời gian. Em mong nhận được những đóng góp quý báu từ thầy cô để có thể cải thiện hơn trong tương lai.

Cuối cùng em xin kính chúc các thầy cô trong nhà trường sức khỏe, hạnh phúc và luôn luôn thành công trong sự nghiệp giáo dục. Xin chân thành cảm ơn sự hỗ trợ và những động viên của thầy cô trong suốt hành trình học tập của em.

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

TÓM TẮT ĐỒ ÁN NGÀNH

Đồ án tập trung vào việc phát triển một hệ thống website học tập trực tuyến, áp dụng mô hình kiến trúc microservices và các công nghệ hiện đại. Hệ thống này được xây dựng nhằm đáp ứng nhu cầu học tập trực tuyến hiện nay, với các tính năng như quản lý khóa học, giảng dạy, tương tác giữa giảng viên và học viên, và tích hợp trí tuệ nhân tạo (AI) để hỗ trợ quá trình học tập.

Kiến trúc microservices cho phép các dịch vụ trong hệ thống được phát triển, triển khai và hoạt động độc lập, giúp tăng tính linh hoạt, khả năng mở rộng, và dễ bảo trì. Hệ thống được tích hợp với các công nghệ hiện đại và phù hợp với hệ thống như Docker, Kafka và Keycloak nhằm tối ưu hóa việc triển khai, giao tiếp bất đồng bộ, và đảm bảo bảo mật.

Trong quá trình phát triển, đồ án đã chú trọng đến việc tối ưu hóa hiệu suất hệ thống, đảm bảo khả năng mở rộng khi có lượng lớn người dùng truy cập. Ngoài ra, tính năng giám sát hiệu suất và tình trạng của hệ thống cũng được tích hợp nhằm đảm bảo hệ thống luôn hoạt động ổn định.

Kết quả của đồ án đã chứng minh tính hiệu quả và khả thi của kiến trúc microservices và khả năng ứng dụng trong việc phát triển các hệ thống học tập trực tuyến hiện đại, linh hoạt, và dễ dàng mở rộng trong tương lai.

ABSTRACT

The project focuses on developing an online learning website system using the microservices architecture and modern technologies. This system is designed to meet the current demand for online learning, featuring course management, teaching, interaction between instructors and students, and the integration of artificial intelligence (AI) to support the learning process.

The microservices architecture allows the system's services to be developed, deployed, and operated independently, enhancing flexibility, scalability, and maintainability. The system is integrated with modern and suitable technologies such as Docker, Kafka, and Keycloak to optimize deployment, enable asynchronous communication, and ensure security.

During development, the project focused on optimizing the system's performance and ensuring scalability to handle a large number of users. Additionally, real-time monitoring features were implemented to track system performance and status, ensuring stable operation.

The results of the project have demonstrated the efficiency and feasibility of the microservices architecture and its applicability in developing modern, flexible, and scalable online learning systems for the future.

MỤC LỤC

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN.....	2
TÓM TẮT ĐỒ ÁN NGÀNH.....	3
ABSTRACT	4
DANH MỤC TỪ VIẾT TẮT	8
DANH MỤC HÌNH VẼ	9
DANH MỤC BẢNG	11
Chương 1. GIỚI THIỆU ĐỀ TÀI	12
1.1. Giới thiệu	12
1.2. Lý do chọn đề tài	12
1.3. Mục tiêu của đề tài.....	13
1.4. Phạm vi của đề tài.....	13
1.5. Phương pháp nghiên cứu	13
Chương 2. CƠ SỞ LÝ THUYẾT	15
2.1. Sơ lược về các công nghệ và kiến trúc phía Back-end.....	15
2.1.1. Ngôn ngữ Java.....	15
2.1.2. Spring Boot.....	16
2.1.3. WebSocket.....	17
2.1.4. Amazon Web Services S3	19
2.1.5. Apache Kafka	19
2.1.6. Các loại cơ sở dữ liệu được sử dụng trong hệ thống.....	22
2.1.7. Một số kiến trúc dùng cho hệ thống	24
2.1.8. Các dịch vụ khác của hệ thống.....	28
2.2. Sơ lược về các công nghệ và kiến trúc phía Front-end	32
2.2.1. ReactJS	32

2.2.2.	React Router	33
2.2.3.	Material-UI (MUI)	34
2.2.4.	Axios	34
2.2.5.	Fetch API.....	35
Chương 3.	TRIỂN KHAI HỆ THỐNG	36
3.1.	Giới thiệu hệ thống	36
3.2.	Kiến trúc hệ thống	37
3.2.1.	Các thành phần chính của hệ thống.....	37
3.2.2.	Cơ chế xác thực và phân quyền của hệ thống	39
3.2.3.	Cơ chế tạo tài khoản cho người dùng của hệ thống	41
3.2.4.	Cơ chế giao tiếp đồng bộ của các service	42
3.2.5.	Cơ chế giao tiếp bất đồng bộ của các service.....	43
3.3.	Cơ sở dữ liệu của các service	44
3.3.1.	Cơ sở dữ liệu của Profile Service.....	44
3.3.2.	Cơ sở dữ liệu của Course Service	44
3.3.3.	Cơ sở dữ liệu của Lecture Service	45
3.3.4.	Cơ sở dữ liệu của Assignment Service.....	46
3.3.5.	Cơ sở dữ liệu của Exam Result Service	46
3.3.6.	Cơ sở dữ liệu của Enrollment Service.....	47
3.3.7.	Cơ sở dữ liệu của Chat Service	47
3.3.8.	Cơ sở dữ liệu của AI Service	48
3.3.9.	Cơ sở dữ liệu của Blog Service	48
3.4.	Giao diện của hệ thống	50
3.4.1.	Giao diện trang đăng nhập	50
3.4.2.	Giao diện trang đăng kí	51
3.4.3.	Giao diện trang chủ	51

3.4.4.	Giao diện tạo khóa học cho giáo viên	52
3.4.5.	Giao diện chỉnh sửa thông tin cá nhân	53
3.4.6.	Giao diện chỉnh sửa khóa học của giáo viên	53
3.4.7.	Giao diện thêm bài giảng cho giáo viên	54
3.4.8.	Giao diện trò chuyện trực tuyến với AI	54
3.4.9.	Giao diện thêm bài tập trong bài giảng	55
3.4.10.	Giao diện các bài giảng trong khóa học	55
3.4.11.	Giao diện diễn đàn	56
3.4.12.	Giao diện làm bài tập của học viên	56
3.4.13.	Giao diện nạp tiền qua VNPAY	57
3.4.14.	Giao diện thống kê của người quản trị	57
3.4.15.	Giao diện trò chuyện trực tuyến	58
3.4.16.	Giao diện quản lý người dùng của người quản trị	58
3.4.17.	Giao diện thống kê giám sát hiệu suất của hệ thống	59
Chương 4.	KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	60
4.1.	Kết luận	60
4.2.	Hướng phát triển	61
	TÀI LIỆU THAM KHẢO	62
	PHỤ LỤC	63

DANH MỤC TỪ VIẾT TẮT

STT	Từ viết tắt	Giải thích
1	AI	Artificial Intelligence
2	MVC	Model –Control -View
3	JVM	Java Virtual Machine
4	API	Application Programming Interface
5	OOP	Object-Oriented Programming
6	AWS	Amazon Web Services
7	HTTP	HyperText Transfer Protocol
8	JSON	JavaScript Object Notation
9	SQL	Structured Query Language
10	SSO	Single Sign-On
11	DOM	Document Object Model
12	SPA	Single Page Application

DANH MỤC HÌNH VẼ

Hình 2.1: Cách thức hoạt động của SteamAPI.....	15
Hình 2.2: Kiến trúc của Spring Boot.....	16
Hình 2.3: Cấu trúc WebSocket.....	18
Hình 2.4: : Mô hình hoạt động của Apache Kafka.....	20
Hình 2.5: Mô hình hoạt động của Kafka Broker.....	20
Hình 2.6 Kiến trúc sử dụng Apache Avro với Kafka và Schema Registry.....	22
Hình 2.7: Kiến trúc MongoDB và các thành phần liên quan	23
Hình 2.8: Kiến trúc MySQL và các thành phần liên quan	24
Hình 2.9: Kiến trúc của mô hình 3 lớp.....	25
Hình 2.10: Mô hình kiến trúc của Docker.....	28
Hình 2.11: Cơ chế hoạt động của Websocket	29
Hình 2.12: Cơ chế bảo mật của Keycloak trong hệ thống Microservices.....	30
Hình 2.13: Cơ chế hoạt động của Grafana	31
Hình 2.14: Cơ chế hoạt động của Virtual DOM trong React.....	32
Hình 3.1: Kiến trúc Microservices hệ thống website học tập trực tuyến.....	37
Hình 3.2: Cơ chế xác thực và phân quyền của hệ thống	40
Hình 3.3: Cơ chế đăng kí tài khoản của hệ thống	42
Hình 3.4: Cơ chế giao tiếp bất đồng thông qua message broker của hệ thống	43
Hình 3.5: Giao diện trang đăng nhập.....	50
Hình 3.6: Giao diện trang đăng kí	51
Hình 3.7: Giao diện trang chủ	51
Hình 3.8: Giao diện trang tạo khóa học cho giáo viên	52
Hình 3.9: Giao diện trang tạo khóa học cho người quản trị	52
Hình 3.10: Giao diện trang chỉ sửa thông tin người dùng	53
Hình 3.11: Giao diện trang chỉnh sửa khóa học cho giáo viên	53
Hình 3.12: Giao diện trang thêm bài giảng cho giáo viên.....	54
Hình 3.13: Giao diện trang trò chuyện với trí tuệ nhân tạo.....	54
Hình 3.14: Giao diện trang thêm bài tập cho khóa học	55
Hình 3.15: Giao diện trang bài giảng của khóa học	55
Hình 3.16: Giao diện trang diễn đàn	56
Hình 3.17: Giao diện trang làm bài tập của học viên	56

Hình 3.18: Giao diện trang nạp tiền qua Vnpay	57
Hình 3.19: Giao diện trang thống kê cho người quản trị.....	57
Hình 3.20: Giao diện trang trò chuyện trực tuyến cho người dùng.....	58
Hình 3.21: Giao diện trang quản lý người dùng.....	58
Hình 3.22: Giao diện trang giám sát hiệu suất hệ thống	59

DANH MỤC BẢNG

Bảng 3.1: Bảng dữ liệu collection profile	44
Bảng 3.2: Bảng dữ liệu collection course.....	45
Bảng 3.3: Bảng dữ liệu collection lecture	45
Bảng 3.4: Bảng dữ liệu collection assignment	46
Bảng 3.5: Bảng dữ liệu collection question.....	46
Bảng 3.6: Bảng dữ liệu collection assignmentResponse.....	46
Bảng 3.7: Bảng dữ liệu collection questionResult	47
Bảng 3.8: Bảng dữ liệu collection enrollment.....	47
Bảng 3.9: Bảng dữ liệu collection chat	48
Bảng 3.10: Bảng dữ liệu collection chatAi	48
Bảng 3.11: Bảng dữ liệu collection blog	49
Bảng 3.12: Bảng dữ liệu collection comment	49
Bảng 3.13: Bảng dữ liệu collection interaction	50

Chương 1. GIỚI THIỆU ĐỀ TÀI

1.1. Giới thiệu

Với sự phát triển không ngừng của thời đại công nghệ số, nhu cầu về học tập trực tuyến ngày càng trở nên cấp thiết hơn bao giờ hết. Đặc biệt, trong bối cảnh sự phát triển của ngành giáo dục ngày càng mạnh mẽ, việc xây dựng và triển khai các hệ thống học tập trực tuyến không còn chỉ là yêu cầu tạm thời mà đã trở thành xu hướng tất yếu trong tương lai. Điều này đòi hỏi các tổ chức giáo dục phải xây dựng những hệ thống đủ mạnh mẽ để đáp ứng nhu cầu của người dùng.

Nhận thấy nhiều hệ thống hiện nay hay gặp không ít vấn đề liên quan đến khả năng mở rộng, tính linh hoạt và hiệu suất khi phải phục vụ cho lượng người dùng lớn trong thời gian ngắn và một trong những lý do lớn nhất dẫn đến việc này là do các hệ thống đó chưa được xây dựng trên kiến trúc đủ mạnh và linh hoạt, Do đó em đã tiến hành nghiên cứu và phát triển một website học tập trực tuyến dựa trên kiến trúc microservices. Đây là một mô hình hiện đại đã được chứng minh là mang lại tính hiệu quả cao và ổn định bởi nhiều công ty công nghệ lớn của thế giới. Bằng cách áp dụng kiến trúc microservices hệ thống sẽ dễ dàng mở rộng, bảo trì và nâng cao hiệu suất một cách hiệu quả hơn.

Với dự án này, em mong muốn không chỉ phát triển một nền tảng học tập trực tuyến đáp ứng nhu cầu thực tế mà còn muốn mang lại một giải pháp kiến trúc công nghệ vững chắc, có khả năng ứng dụng rộng rãi trong các hệ thống học tập tương lai.

1.2. Lý do chọn đề tài

Em đã phát triển đề tài này với mục đích nghiên cứu, xây dựng và triển khai mô hình microservices vào thực tế. Mục tiêu là tối ưu hóa hiệu suất của hệ thống, kế thừa và phát huy những ưu điểm của microservices, đồng thời cải thiện khả năng mở rộng, bảo trì và hiệu suất cao cho hệ thống. Ngoài ra, hệ thống còn được tích hợp các tính năng mới như trí tuệ nhân tạo (AI) nhằm mang lại trải nghiệm học tập thông minh và sáng tạo cho người dùng. Việc áp dụng microservices giúp hệ thống trở nên bảo mật, linh hoạt, và dễ bảo trì trong tương lai.

1.3. Mục tiêu của đề tài

Nghiên cứu, ứng dụng và triển khai kiến trúc microservices: Khảo sát, phân tích và hiểu rõ kiến trúc microservices, từ đó đưa vào việc xây dựng một hệ thống tiên tiến. Mục tiêu là nâng cao khả năng quản lý, mở rộng, và bảo trì hệ thống một cách hiệu quả hơn so với các kiến trúc truyền thống.

Xây dựng hệ thống học tập trực tuyến hiện đại: Phát triển một hệ thống có tính năng học tập trực tuyến với khả năng đáp ứng lượng người dùng lớn, hoạt động ổn định, dễ dàng nâng cấp và bảo trì.

Tích hợp trí tuệ nhân tạo (AI): Xây dựng tính năng chat thông minh bằng AI để hỗ trợ người dùng học tập hiệu quả hơn. Tính năng này nhằm cải thiện sự tương tác và mang lại trải nghiệm học tập cá nhân hóa, giúp người dùng giải quyết vấn đề nhanh chóng và hiệu quả.

Tối ưu hóa hiệu suất và bảo mật: Đảm bảo hệ thống hoạt động ổn định, với hiệu suất cao và các tính năng bảo mật được cải thiện, giảm thiểu rủi ro về dữ liệu cũng như các vấn đề bảo mật khác.

Khả năng mở rộng và bảo trì: Thiết kế hệ thống linh hoạt để dễ dàng mở rộng khi số lượng người dùng tăng hoặc khi có các yêu cầu mới, đồng thời đảm bảo chi phí bảo trì thấp.

1.4. Phạm vi của đề tài

Phạm vi của nghiên cứu tập trung vào việc tìm hiểu, nghiên cứu, và triển khai một hệ thống dựa trên kiến trúc microservices, tích hợp các công nghệ tiên tiến và hiện đại để đảm bảo hệ thống hoạt động ổn định và có tính năng mở rộng cao.

1.5. Phương pháp nghiên cứu

Tiến hành khảo sát và phân tích nhu cầu thực tế của người dùng.

Tìm kiếm và nghiên cứu tài liệu liên quan về các kiến trúc microservices và các công nghệ liên quan.

Tổng hợp thông kê và so sánh, những công nghệ nên áp dụng cho hệ thống dựa trên nhu cầu thực tế

Phân tích và thiết kế hệ thống dựa trên các nghiên cứu và và khảo sát thực tiễn để đảm bảo tính khả thi.

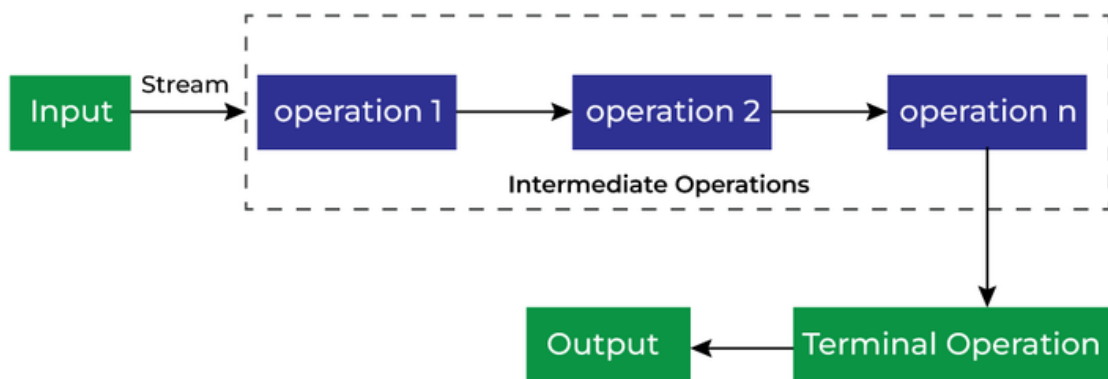
Chương 2. CƠ SỞ LÝ THUYẾT

2.1. Sơ lược về các công nghệ và kiến trúc phía Back-end

2.1.1. Ngôn ngữ Java

Java là một ngôn ngữ lập trình bậc cao phổ biến, hướng đối tượng và được phát triển bởi Sun Microsystems vào năm 1995. Java nổi bật với khả năng đa nền tảng (cross-platform), cho phép các chương trình Java chạy trên rất nhiều hệ điều hành khác nhau nhờ vào Java Virtual Machine (JVM). Một trong những đặc điểm chính của Java là tính chất "Viết một lần, chạy mọi nơi" (Write Once, Run Anywhere), có thể hiểu là mã nguồn Java chỉ cần viết một lần và có thể chạy trên tất cả các hệ thống nào có hỗ trợ JVM mà không cần phải chỉnh sửa mã nguồn.

Java hỗ trợ nhiều tính năng mạnh mẽ và hiện đại, nổi bật như Stream API và Functional Programming (lập trình hàm), được giới thiệu từ phiên bản Java 8.



Hình 2.1: Cách thức hoạt động của StreamAPI

(Nguồn ảnh: <https://www.geeksforgeeks.org/stream-in-java/>)

Stream API là một công cụ mạnh mẽ cho phép lập trình viên xử lý các tập dữ liệu theo cách tuần tự hoặc song song mà không cần phải qua các vòng lặp phức tạp. Stream API cung cấp các phương thức như `filter()`, `map()`, và `reduce()` để thao tác dữ liệu một cách dễ dàng, đồng thời tăng hiệu suất xử lý khi làm việc với các tập dữ liệu lớn. Stream API hỗ trợ xử lý song song, tận dụng tối đa khả năng của các CPU đa nhân, giúp cải thiện hiệu suất xử lý dữ liệu.

Bên cạnh đó, Functional Programming trong Java giúp mã nguồn trở nên ngắn gọn và dễ hiểu hơn. Với sự hỗ trợ của lambda expressions và các phương thức như `forEach()`, `map()`, và `filter()`, lập trình viên có thể viết mã một cách ngắn gọn, rõ ràng, và tránh các lỗi không đáng có trong quá trình phát triển phần mềm. Lập trình hàm giúp tối ưu hóa các tác vụ xử lý dữ liệu, đồng thời tạo ra mã nguồn dễ bảo trì và phát triển lâu dài.

Nhờ sự kết hợp giữa các tính năng truyền thống như hướng đối tượng (OOP) và các yếu tố hiện đại như Stream API và Functional Programming, Java trở thành ngôn ngữ lý tưởng cho việc phát triển các hệ thống lớn, linh hoạt và hiệu quả và được sử dụng rộng rãi đến ngày nay.

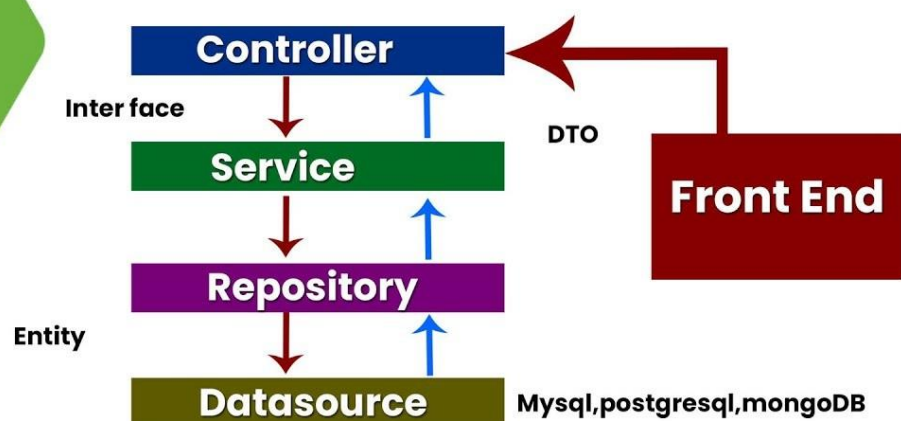
2.1.2. Spring Boot

Spring Boot là một phần mở rộng của framework Spring, ra đời nhằm đơn giản hóa quá trình phát triển ứng dụng Java. Mục tiêu chính của Spring Boot là giúp các lập trình viên dễ dàng khởi tạo và phát triển ứng dụng Spring mà không cần phải thiết lập quá nhiều cấu hình phức tạp so với các ứng dụng Spring truyền thống. Với Spring Boot, các cấu hình mặc định được cung cấp sẵn, giúp tiết kiệm thời gian và giảm thiểu lỗi phát sinh từ các cấu hình phức tạp.

Spring Boot



Layered Architecture



Hình 2.2: Kiến trúc của Spring Boot

(Nguồn ảnh: <https://www.linkedin.com/pulse/spring-boot-onkar-kulkarni-eutzf/>)

Một trong những ưu điểm nổi bật của Spring Boot là cơ chế "Convention over Configuration" (Quy ước hơn Cấu hình), nghĩa là thay vì yêu cầu lập trình viên phải cấu hình tất cả các thành phần chi tiết của ứng dụng, Spring Boot cung cấp các cấu hình mặc định hợp lý dựa trên các quy ước phổ biến. Điều này giúp lập trình viên tập trung hơn vào việc phát triển tính năng của ứng dụng mà không cần lo lắng về việc cấu hình chi tiết từng thành phần.

Spring Boot cũng tích hợp sẵn embedded server (máy chủ nhúng), chẳng hạn như Tomcat hoặc Jetty, cho phép bạn chạy ứng dụng ngay lập tức mà không cần phải triển khai trên máy chủ ngoài. Điều này giúp tăng tốc quá trình phát triển và thử nghiệm ứng dụng, vì bạn chỉ cần chạy lệnh để khởi động ứng dụng mà không cần qua nhiều bước thiết lập như trước.

Ngoài ra, Spring Boot cũng cung cấp Spring Initializr, một công cụ trực tuyến cho phép lập trình viên nhanh chóng khởi tạo dự án Spring với các thành phần cần thiết. Người dùng chỉ cần chọn các module, công nghệ cần sử dụng và Spring Initializr sẽ tự động tạo ra một dự án mẫu với cấu trúc và các thiết lập cơ bản, sẵn sàng để phát triển.

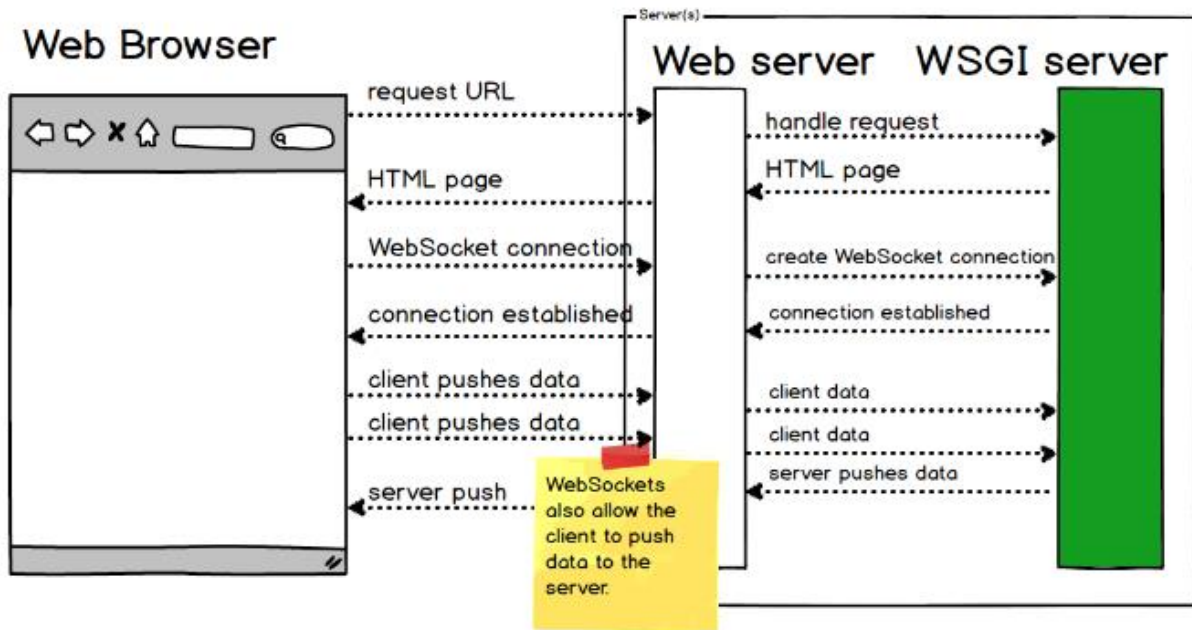
Spring Boot còn hỗ trợ rất tốt việc tích hợp với các công nghệ hiện đại như RESTful APIs, microservices, và cloud services, giúp ứng dụng dễ dàng mở rộng và triển khai trên các nền tảng đám mây như AWS, Azure, hoặc Google Cloud. Ngoài ra, Spring Boot còn đi kèm với các công cụ giám sát và quản lý, giúp dễ dàng theo dõi hiệu suất và tình trạng của ứng dụng trong quá trình hoạt động.

Nhờ vào sự đơn giản, linh hoạt và khả năng mở rộng, Spring Boot đã trở thành một trong những framework được sử dụng phổ biến nhất trong việc phát triển ứng dụng doanh nghiệp và ứng dụng web hiện đại.

2.1.3. WebSocket

WebSocket là một giao thức phổ biến để truyền thông mạng cho phép thiết lập kênh liên lạc hai chiều, toàn thời gian giữa máy chủ và client thông qua một kết nối duy nhất. Không giống như giao thức HTTP truyền thống, vốn hoạt động theo kiểu yêu cầu-đáp ứng (request-response), WebSocket cho phép dữ liệu được truyền liên tục giữa máy chủ và client mà không cần phải gửi yêu cầu mới mỗi khi có sự thay đổi.

WebSockets



Hình 2.3: Cấu trúc WebSocket

(Nguồn ảnh: <https://hocspringboot.net/2020/12/10/websocket-la-gi/>)

WebSocket được thiết kế để cải thiện và tối ưu hiệu suất, tính tương tác trong các ứng dụng web thời gian thực, nơi cần truyền tải dữ liệu liên tục và nhanh chóng như các ứng dụng trò chuyện (chat), theo dõi thị trường tài chính, hoặc các game trực tuyến. Sau khi kết nối được thiết lập thành công, WebSocket sử dụng cùng một cổng và duy trì kết nối mở, cho phép dữ liệu được gửi và nhận ngay lập tức khi có sự kiện phát sinh từ bất kỳ phía nào (client hoặc server).

Một ưu điểm lớn và nổi bật của WebSocket so với HTTP truyền thống là giảm thiểu độ trễ trong việc trao đổi dữ liệu. Thay vì phải gửi nhiều yêu cầu HTTP đơn lẻ từ client đến server, WebSocket chỉ cần thiết lập một kết nối duy nhất và sử dụng nó trong suốt thời gian chạy của ứng dụng. Điều này giúp tiết kiệm tài nguyên, tối ưu và giảm tải cho cả hai phía.

Kiến trúc WebSocket hoạt động dựa trên việc khởi tạo một "handshake" thông qua giao thức HTTP trong giai đoạn đầu của kết nối. Sau khi quá trình bắt tay thành công, kết nối được chuyển sang sử dụng giao thức WebSocket, cho phép truyền tải dữ liệu

theo cả hai hướng mà không cần gửi lại bất kỳ yêu cầu HTTP nào. Kết nối WebSocket sẽ tiếp tục duy trì cho đến khi một trong hai phía dùng kết nối.

Nhờ vào tính năng giao tiếp hai chiều và hiệu quả cao và tối ưu hiệu suất tốt mà WebSocket đã trở thành lựa chọn lý tưởng cho các ứng dụng yêu cầu tương tác liên tục và theo thời gian thực.

2.1.4. Amazon Web Services S3

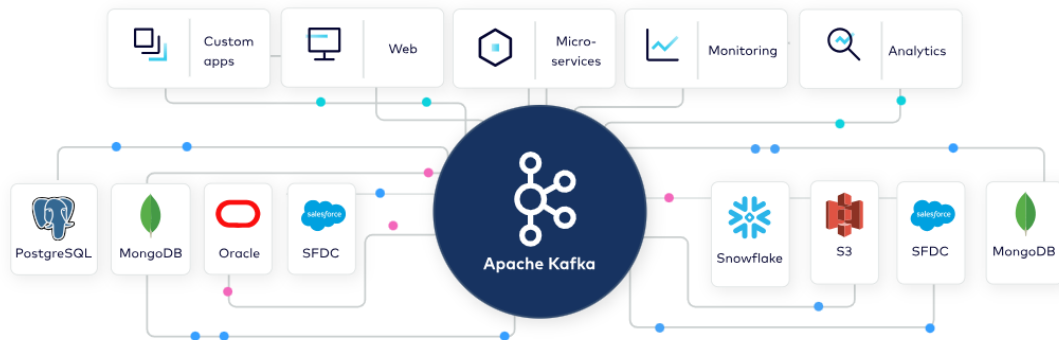
Amazon Web Services S3 (Simple Storage Service) là dịch vụ kho lưu trữ mạnh mẽ được cung cấp bởi tập đoàn Amazon, cho phép người dùng lưu trữ và truy xuất lượng dữ liệu lớn một cách linh hoạt, nhanh chóng và an toàn. AWS S3 hỗ trợ nhiều loại lưu trữ các loại tệp tin như hình ảnh, video, tài liệu và dữ liệu ứng dụng, với khả năng mở rộng vô hạn. Điểm mạnh của S3 là tính sẵn sàng cao, đảm bảo dữ liệu được lưu trữ an toàn với khả năng sao lưu và khả năng phục hồi mạnh mẽ. Ngoài ra, S3 còn cung cấp các nhiều tính năng mạnh mẽ như quản lý quyền truy cập, phân quyền bảo mật, và tích hợp dễ dàng với nhiều dịch vụ khác trong hệ sinh thái của AWS.

2.1.5. Apache Kafka

Apache Kafka là một nền tảng phân tán dùng để truyền tải các luồng dữ liệu theo thời gian thực. Kafka hoạt động theo mô hình publish-subscribe, nơi mà các producer gửi thông điệp (messages) vào các chủ đề (topics) và các consumer sẽ lắng nghe và xử lý các thông điệp này. Kafka được thiết kế để xử lý dữ liệu nhanh chóng và đáng tin cậy, đảm bảo khả năng chịu lỗi (fault-tolerance) trong hệ thống phân tán.

Kafka thường được sử dụng trong các hệ thống có yêu cầu cao về khả năng mở rộng và hiệu suất, như thu thập log, xử lý các sự kiện bất đồng bộ, và các kiến trúc microservices, nơi mà các thành phần của hệ thống cần trao đổi dữ liệu với nhau một cách linh hoạt và hiệu quả.

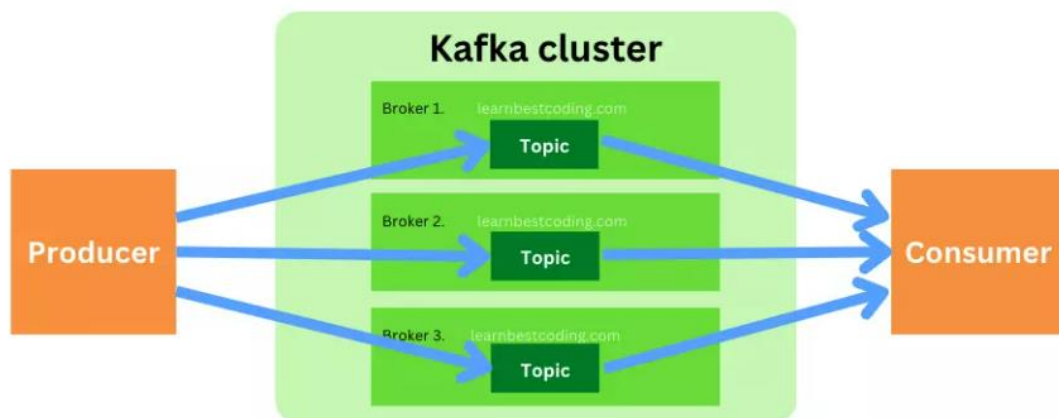
Apache Kafka in Action



Hình 2.4: : Mô hình hoạt động của Apache Kafka
(Nguồn ảnh: <https://www.confluent.io/what-is-apache-kafka/>)

2.1.5.1. Kafka Broker

Broker trong Kafka là thành phần chịu trách nhiệm quản lý, lưu trữ và xử lý các luồng dữ liệu. Mỗi Kafka cluster có thể bao gồm nhiều broker để phân tán tải và đảm bảo tính sẵn sàng. Broker nhận các thông điệp từ producer và lưu trữ chúng trong các phân vùng (partitions), sau đó gửi thông điệp cho consumer khi có yêu cầu. Một cụm Kafka có thể chứa nhiều broker để đảm bảo dữ liệu được phân phối đồng đều và tăng tính chịu lỗi của hệ thống.



Hình 2.5: Mô hình hoạt động của Kafka Broker
(Nguồn ảnh: <https://www.learnbestcoding.com/post/123/how-to-create-multiple-brokers-in-apache-kafka>)

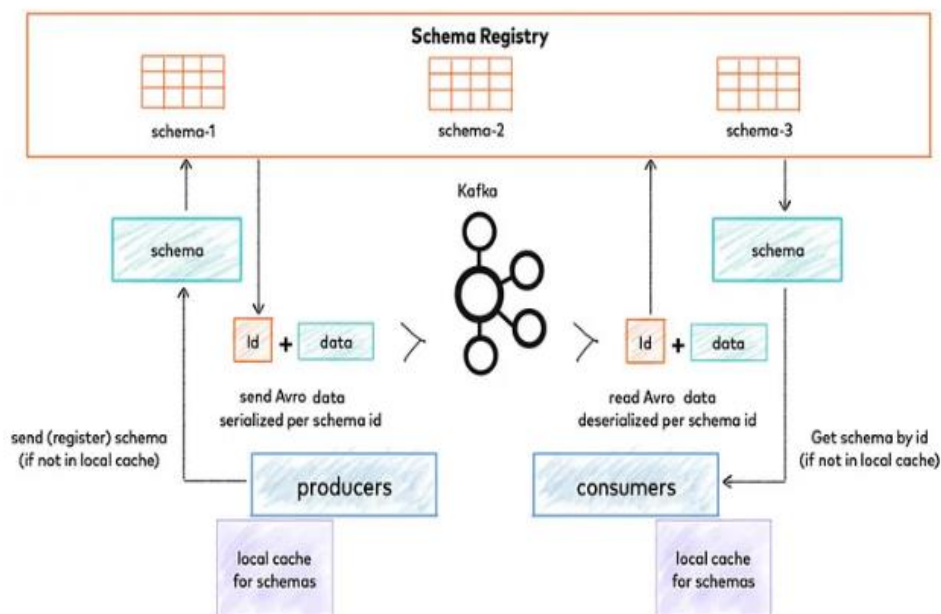
2.1.5.2. Apache Zookeeper

Zookeeper là một dịch vụ quản lý cấu hình và điều phối trong hệ thống phân tán, và trong Kafka, Zookeeper đóng vai trò quan trọng trong việc quản lý metadata, theo dõi trạng thái của broker, phân vùng và chủ đề. Zookeeper giúp các broker trong cụm Kafka giữ kết nối với nhau, đồng thời đảm bảo việc phân phối tải và phục hồi khi một hoặc nhiều broker gặp sự cố.

Zookeeper cũng hỗ trợ trong việc quản lý việc phân bổ các partitions và đảm bảo các consumer có thể truy cập đúng dữ liệu. Dù Zookeeper là một thành phần quan trọng trong kiến trúc Kafka, nhưng các phiên bản Kafka mới hơn đã dần chuyển sang sử dụng KRaft, một cơ chế mới thay thế Zookeeper để quản lý cụm Kafka.

2.1.5.3. Apache Avro

Apache Avro là một framework nguồn mở, được sử dụng chủ yếu cho việc serial hóa và deserial hóa dữ liệu trong hệ thống phân tán, đặc biệt được sử dụng phổ biến trong các kiến trúc dựa trên Hadoop và Kafka. Avro được phát triển như một phần của dự án Apache Hadoop và được thiết kế để xử lý dữ liệu có cấu trúc một cách nhanh chóng, hiệu quả và linh hoạt. Một số điểm nổi bật đáng kể đến của Avro định nghĩa cấu trúc của dữ liệu, khả năng mở rộng cao, dễ phân biệt và khó bị trùng lặp, tích hợp dễ dàng với Kafka, hiệu suất cao và không bị phụ thuộc vào ngôn ngữ. Từ những điểm nổi bật trên Avro đã được sử dụng phổ biến cho việc định dạng dữ liệu khi truyền qua các topic trong các hệ thống phân tán.

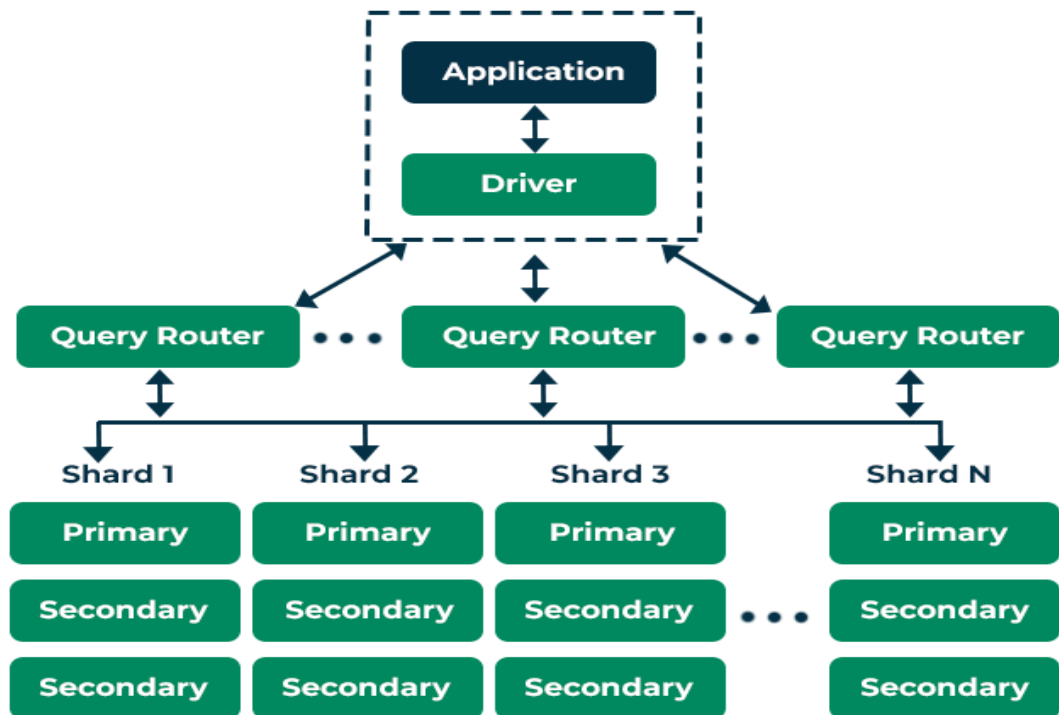


Hình 2.6 Kiến trúc sử dụng Apache Avro với Kafka và Schema Registry
(Nguồn ảnh: <https://levelup.gitconnected.com/apache-avro-bringing-structured-data-to-kafka-b225ae9eaa47>)

2.1.6. Các loại cơ sở dữ liệu được sử dụng trong hệ thống

2.1.6.1. MongoDB

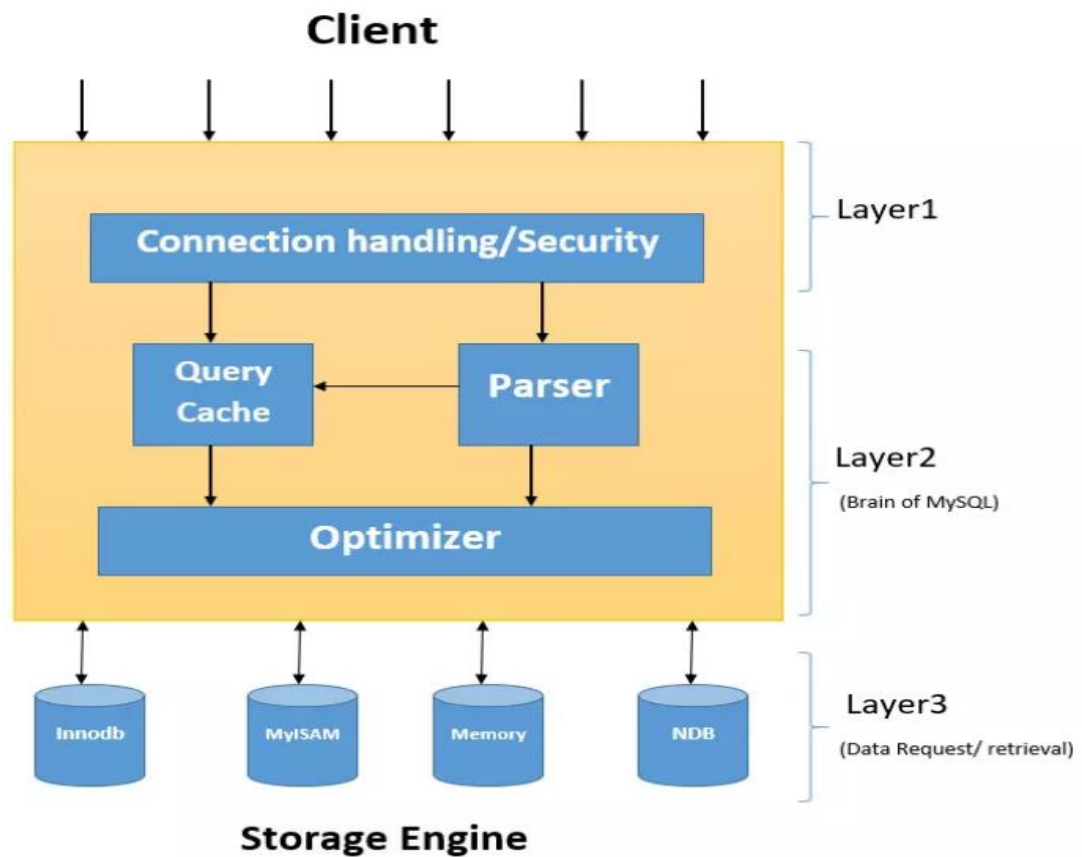
MongoDB là một cơ sở dữ liệu NoSQL mã nguồn mở, được thiết kế để lưu trữ và truy vấn dữ liệu phi cấu trúc. MongoDB sử dụng cấu trúc dữ liệu dạng tài liệu (document-oriented), thường được biểu diễn dưới dạng JSON (JavaScript Object Notation) hoặc BSON (Binary JSON). Điều này giúp hệ thống linh hoạt trong việc quản lý dữ liệu đa dạng và cho phép mở rộng quy mô theo chiều ngang (horizontal scaling). MongoDB phù hợp để sử dụng trong các hệ thống yêu cầu lưu trữ dữ liệu động, không đồng nhất, cần truy vấn nhanh chóng và khả năng mở rộng cao.



Hình 2.7: Kiến trúc MongoDB và các thành phần liên quan
(Nguồn ảnh: <https://www.geeksforgeeks.org/mongodb-architecture>)

2.1.6.2. MySQL

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở, được sử dụng rộng rãi trong các ứng dụng yêu cầu lưu trữ dữ liệu có cấu trúc. MySQL hoạt động dựa trên mô hình bảng, trong đó dữ liệu được tổ chức dưới dạng hàng và cột. Hệ quản trị cơ sở dữ liệu này cung cấp khả năng truy vấn mạnh mẽ thông qua ngôn ngữ SQL (Structured Query Language) và hỗ trợ các tính năng như quản lý giao dịch (transaction management), khóa ngoại (foreign key), và chỉ mục (index). MySQL thường được sử dụng trong các hệ thống yêu cầu tính toàn vẹn và nhất quán của dữ liệu cao trong hệ thống.



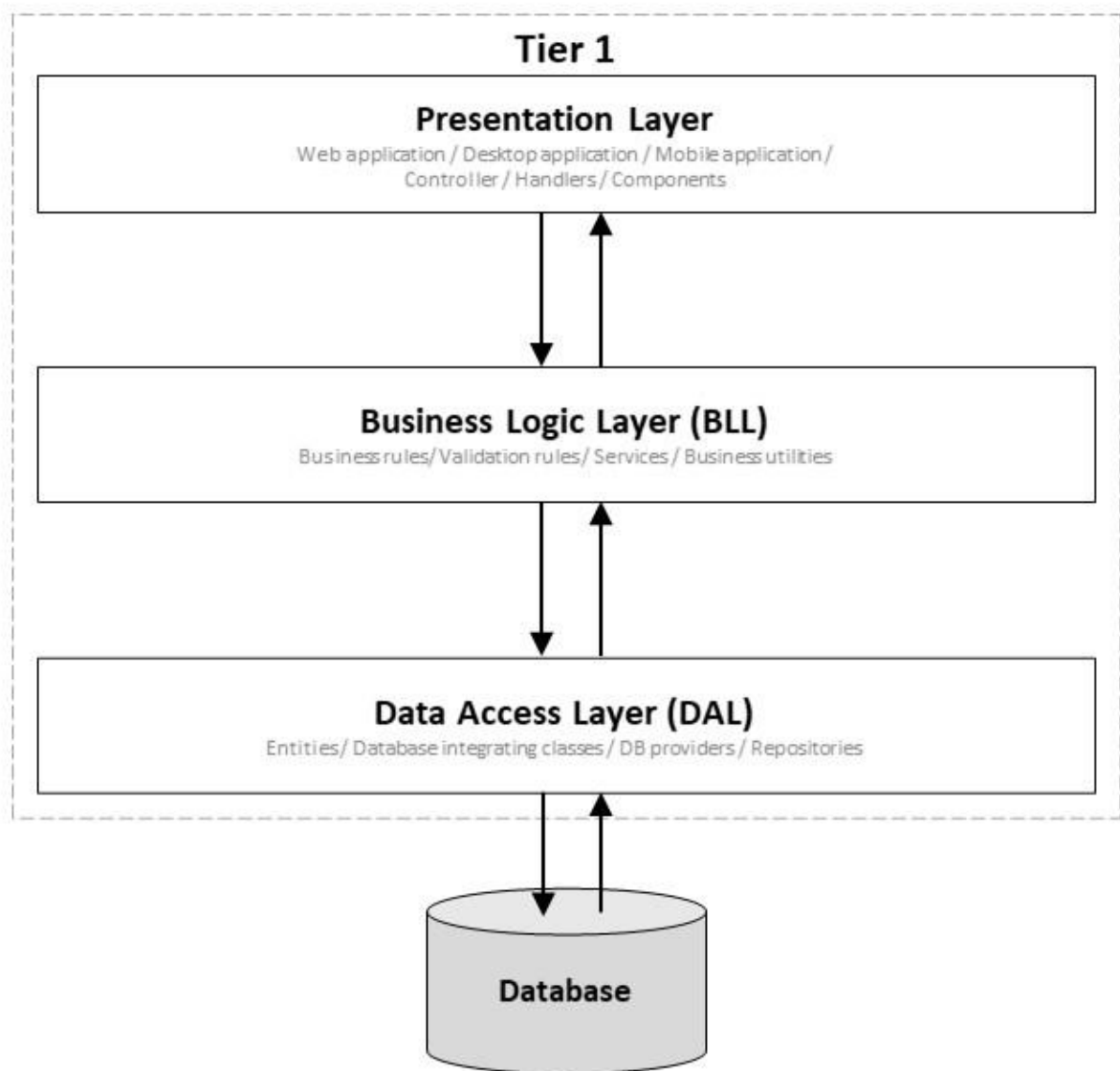
Hình 2.8: Kiến trúc MySQL và các thành phần liên quan

(Nguồn ảnh: <https://viblo.asia/p/mysql-architecture-and-history-RQqKLkd057z>)

2.1.7. Một số kiến trúc dùng cho hệ thống

2.1.7.1. Mô hình 3 lớp ở các service

Mô hình 3 lớp (Three-Tier Architecture) là một kiểu kiến trúc phần mềm được phân tách thành ba lớp chính, trong đó mỗi lớp thực hiện một nhiệm vụ cụ thể, giúp hệ thống dễ dàng phát triển, bảo trì và mở rộng. Mô hình này thường được áp dụng cho các ứng dụng lớn, nơi yêu cầu cao về tính độc lập giữa các thành phần và khả năng mở rộng.



Hình 2.9: Kiến trúc của mô hình 3 lớp

(Nguồn ảnh: <https://nextcodeblock.com/posts/multi-layer-software-architecture/>)

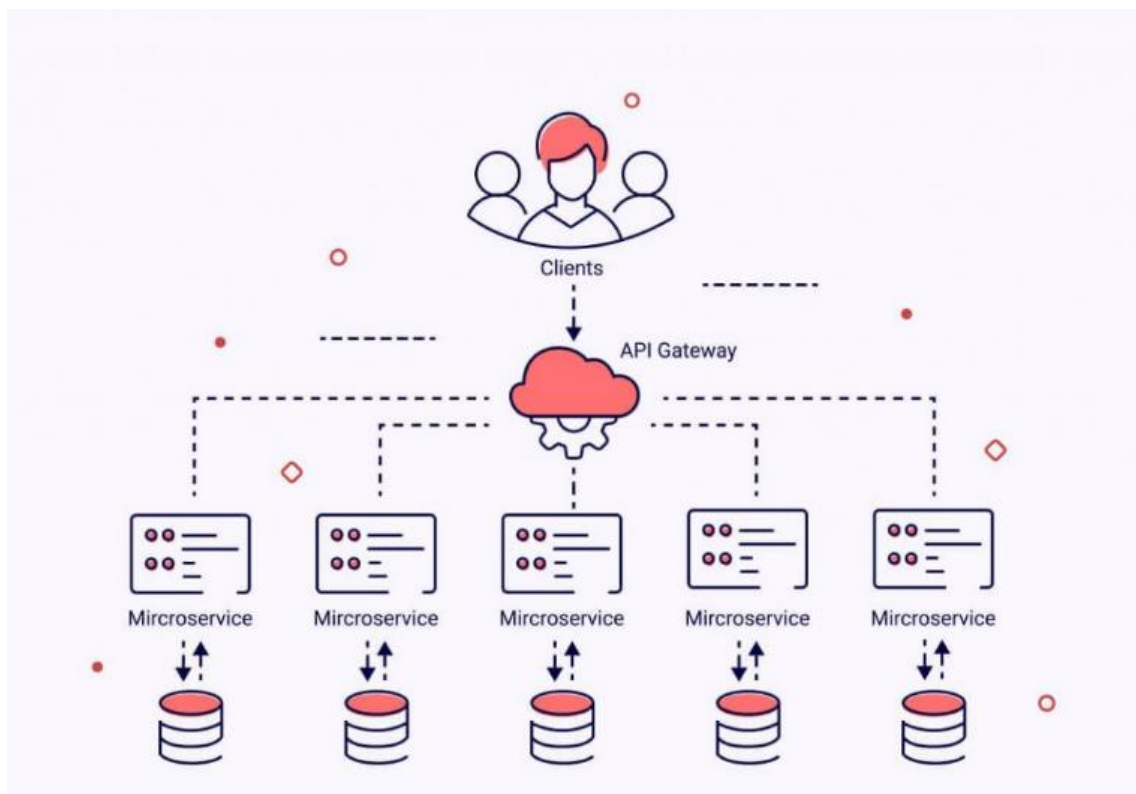
- Lớp Giao diện (Presentation Layer): Lớp này chịu trách nhiệm giao tiếp trực tiếp với người dùng (client), xử lý và nhận yêu cầu từ phía họ. Nó bao gồm các giao diện người dùng hoặc API mà client sử dụng để tương tác với hệ thống, gửi và nhận dữ liệu. Trong Spring Boot, lớp này thường được triển khai thông qua các Controller (@RestController), xử lý các yêu cầu HTTP (GET, POST, PUT, DELETE) từ người dùng.
- Lớp Logic Nghiệp vụ (Business Logic Layer): Đây là nơi quản lý toàn bộ các logic nghiệp vụ của hệ thống. Tất cả các quy tắc, thao tác liên quan đến nghiệp vụ đều được xử lý ở lớp này. Trong Spring Boot, lớp này thường được thực hiện thông qua các

Service (@Service). Các service nhận yêu cầu từ lớp Controller, thực hiện logic nghiệp vụ cần thiết, sau đó tương tác với Repository để lấy hoặc lưu trữ dữ liệu.

- Lớp Truy cập Dữ liệu (Data Access Layer): Lớp này đảm nhiệm việc quản lý, thao tác với cơ sở dữ liệu. Tất cả các hoạt động lưu trữ, truy xuất và cập nhật dữ liệu đều diễn ra tại đây. Trong Spring Boot, lớp này thường được hiện thực thông qua các Repository (@Repository), thường sử dụng Spring Data JPA hoặc các công nghệ tương tự để tương tác với cơ sở dữ liệu như MySQL, MongoDB, hoặc các cơ sở dữ liệu khác.

2.1.7.2. Kiến trúc Microservices trong hệ thống

Kiến trúc Microservices là một trong những xu hướng phát triển phần mềm hiện đại, trong đó hệ thống phần mềm được phân tách thành nhiều dịch vụ (services) nhỏ, mỗi dịch vụ hoạt động một cách độc lập với nhau. Mỗi dịch vụ trong kiến trúc Microservices đảm nhận một nhiệm vụ cụ thể, và các dịch vụ này tương tác với nhau để hoàn thành các chức năng lớn của hệ thống. Điều này giúp kiến trúc Microservices trở thành một giải pháp lý tưởng cho các hệ thống lớn và phức tạp, yêu cầu cao về khả năng mở rộng và bảo trì.



Hình 2.10: Kiến trúc của mô hình Microservices

(Nguồn ảnh: <https://middleware.io/blog/microservices-architecture/>)

Trong kiến trúc Microservices, mỗi dịch vụ có thể sử dụng các công nghệ, ngôn ngữ lập trình, cơ sở dữ liệu khác nhau, tạo nên một kiến trúc linh hoạt và dễ bảo trì. Nhờ vào đó đem lại những ưu điểm mạng mẽ so với kiến trúc truyền thống như :

- + **Độc lập:** Mỗi microservice là một đơn vị độc lập, có thể phát triển, kiểm thử và triển khai riêng lẻ mà không cần phải ảnh hưởng đến các dịch vụ khác.

- + **Khả năng mở rộng:** Các dịch vụ có thể được mở rộng độc lập theo yêu cầu về lưu lượng truy cập hoặc khối lượng dữ liệu của hệ thống.

- + **Khả năng chịu lỗi:** Nếu một dịch vụ gặp sự cố, nó không làm gián đoạn toàn bộ hệ thống mà chỉ ảnh hưởng đến phần chức năng mà dịch vụ đó phụ trách.

- + **Tích hợp linh hoạt:** Các dịch vụ có thể sử dụng các công nghệ khác nhau phù hợp với từng nhu cầu nghiệp vụ, ví dụ như cơ sở dữ liệu SQL hoặc NoSQL, tùy thuộc vào yêu cầu lưu trữ và xử lý dữ liệu.

Trong hệ thống hiện tại các hệ thống được chia thành các service như:

- + **API Gateway:** Là thành phần trung gian, giúp điều phối và định tuyến các yêu cầu từ người dùng đến các dịch vụ thích hợp trong hệ thống. API Gateway còn giúp che giấu các service phía sau, không cần phải lộ trực tiếp ra bên ngoài.

- + **Course Service:** Quản lý thông tin và chức năng liên quan đến các khóa học.

- + **Lecture Service:** Quản lý thông tin và chức năng của bài giảng trong khóa học.

- + **Assignment Service:** Quản lý thông tin và chức năng về bài tập trong hệ thống.

- + **Exam Result Service:** Quản lý các thông tin và chức năng liên quan đến kết quả bài làm của học viên.

- + **Enrollment Service:** Quản lý việc đăng ký khóa học.

- + **Chat Service:** Quản lý chức năng trò chuyện trực tuyến giữa các người dùng.

- + **AI Service:** Quản lý các dịch vụ tích hợp trí tuệ nhân tạo trong hệ thống.

+ **Blog Service:** Quản lý các thông tin và chức năng liên quan đến diễn đàn, nơi người dùng có thể thảo luận và chia sẻ kiến thức.

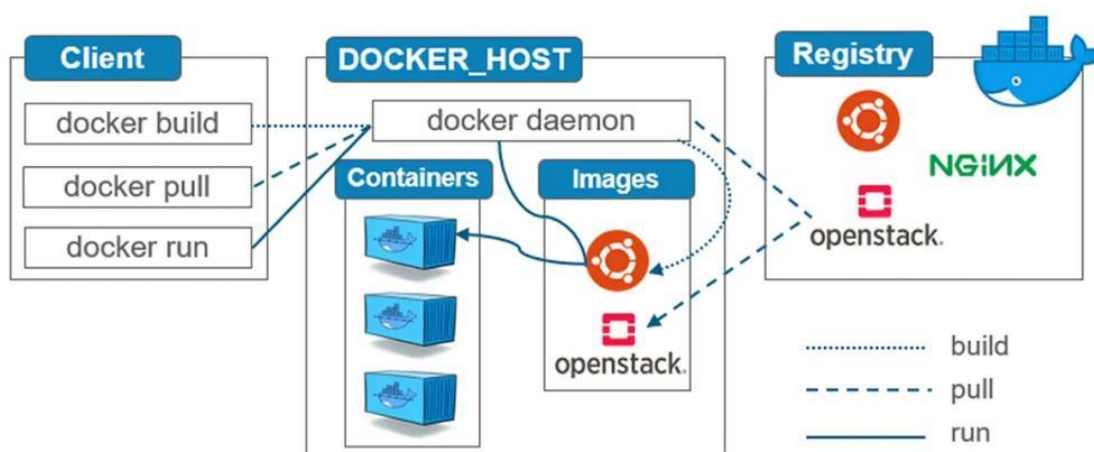
+ **Notification Service:** Quản lý các thông tin và chức năng liên quan đến việc thông báo đến người dùng, gửi mail.

Việc chia nhỏ các dịch vụ theo chức năng giúp hệ thống dễ dàng mở rộng, bảo trì và phát triển hơn, đồng thời tăng tính linh hoạt và khả năng chịu lỗi của hệ thống.

2.1.8. Các dịch vụ khác của hệ thống

2.1.8.1. Docker

Docker là một nền tảng hiện đại mã nguồn mở giúp tạo ra, triển khai và chạy các ứng dụng trong các container. Container là các gói phần mềm nhẹ, bao gồm toàn bộ mã nguồn, các phụ thuộc và môi trường thực thi cần thiết cho ứng dụng. Điều này giúp đảm bảo rằng ứng dụng luôn được chạy ổn định và nhất quán trên các môi trường khác nhau, từ máy phát triển cá nhân đến máy chủ sản xuất.



Hình 2.10: Mô hình kiến trúc của Docker

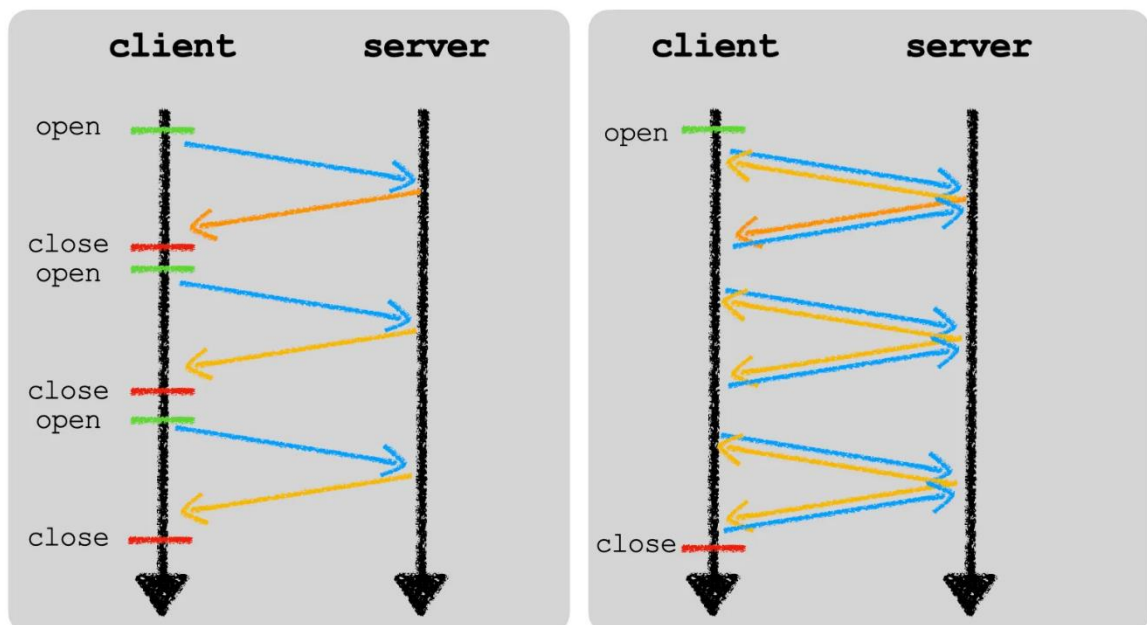
(Nguồn ảnh: <https://community.aws/content/2k3CYLom4uDNDzFBmXnxGfRpoOD>)

Docker giúp hệ thống dễ dàng quản lý và triển khai các microservices. Mỗi dịch vụ có thể được đóng gói thành một container riêng, đảm bảo tính linh hoạt và độc lập. Điều này cũng giúp giảm thiểu xung đột môi trường và tối ưu hóa tài nguyên hệ thống.

2.1.8.2. Web Socket

WebSocket là một giao thức truyền thông mạng cho phép thiết lập một kênh liên lạc hai chiều liên tục giữa máy chủ và client. Không giống như giao thức HTTP truyền thông, WebSocket cho phép dữ liệu được truyền liên tục giữa máy chủ và client mà không cần gửi lại yêu cầu. Điều này làm cho WebSocket trở thành một lựa chọn lý tưởng cho các ứng dụng thời gian thực như trò chuyện trực tuyến (chat), hệ thống theo dõi dữ liệu trực tiếp, và các game online.

WebSockets



Hình 2.11: Cơ chế hoạt động của Websocket

(Nguồn ảnh: <https://medium.com/swlh/how-to-build-a-websocket-applications-using-java-486b3e394139>)

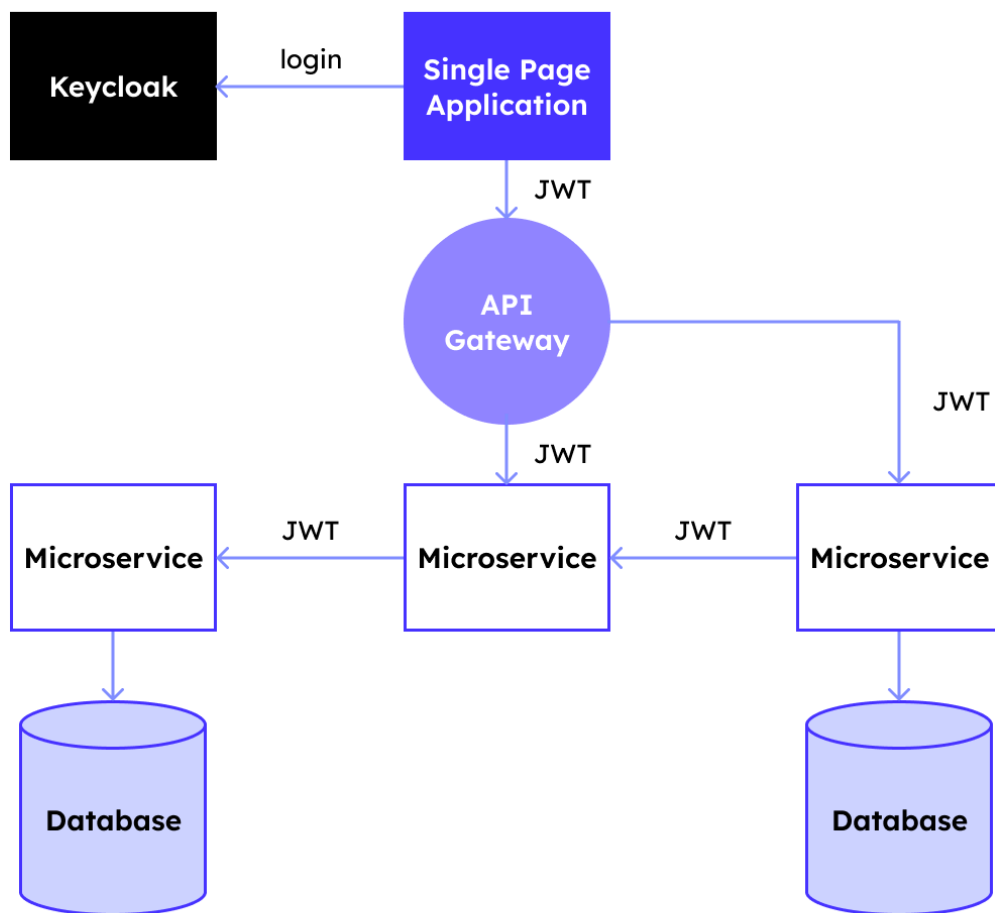
Trong hệ thống, WebSocket được tích hợp để hỗ trợ tính năng trò chuyện trực tuyến giữa người dùng, giúp cải thiện khả năng tương tác và trải nghiệm của người dùng.

2.1.8.3. KeyCloak

KeyCloak là một giải pháp mã nguồn mở dùng để quản lý xác thực và phân quyền. Nó cung cấp các tính năng Single Sign-On (SSO), xác thực người dùng, và phân quyền thông qua các tiêu chuẩn như OAuth2, OpenID Connect, và SAML 2.0. KeyCloak

giúp đơn giản hóa việc tích hợp cho các hệ thống quản lý xác thực, cho phép người dùng chỉ cần đăng nhập một lần và truy cập nhiều dịch vụ mà không cần phải đăng nhập lại.

Trong hệ thống, KeyCloak đảm nhận vai trò quản lý việc xác thực và phân quyền người dùng, giúp đảm bảo an ninh và tính bảo mật cho các dịch vụ và API.

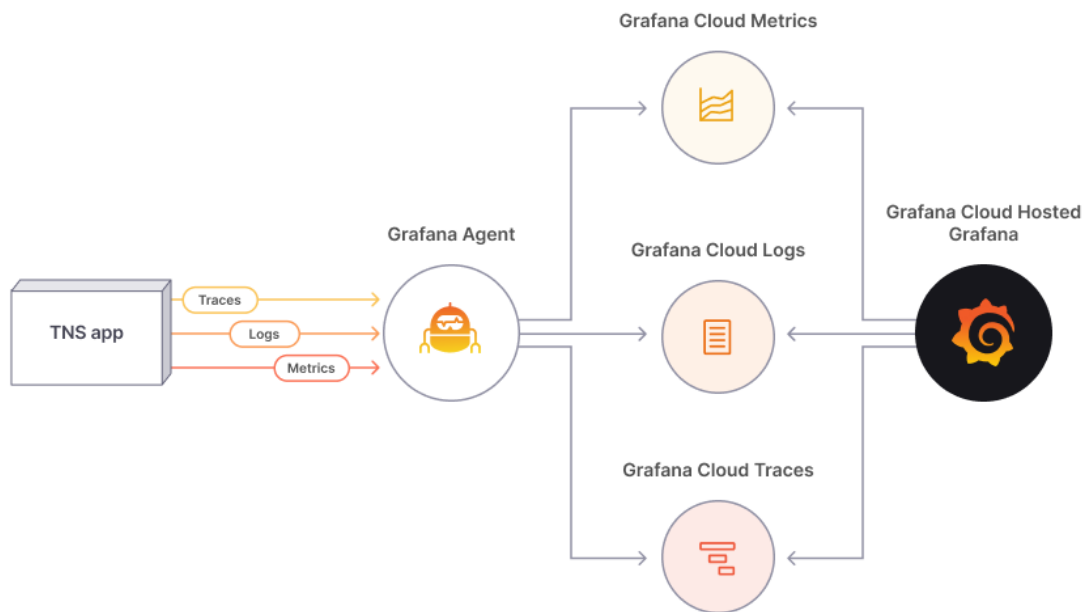


Hình 2.12: Cơ chế bảo mật của Keycloak trong hệ thống Microservices
(Nguồn ảnh: <https://www.altkomsoftware.com/blog/keycloak-security-in-microservices/>)

2.1.8.4. Grafana

Grafana là một công cụ mã nguồn mở dùng để giám sát và phân tích hệ thống được sử dụng rộng rãi trong các hệ thống lớn như microservice dùng để trực quan hóa dữ liệu và giám sát hiệu suất của hệ thống. Grafana cho phép người dùng tạo ra nhiều biểu đồ

và bảng điều khiển được tùy chỉnh từ nhiều nguồn dữ liệu khác nhau như Prometheus, Loki, Tempo và nhiều nguồn khác.



Hình 2.13: Cơ chế hoạt động của Grafana

(Nguồn ảnh: <https://grafana.com/blog/2022/02/23/introducing-exemplar-support-in-grafana-cloud-tightly-coupling-traces-to-your-metrics/>)

Trong hệ thống microservice, việc giám sát hiệu suất và các thông số của hệ thống là vô cùng quan trọng để đảm bảo sự ổn định và hiệu quả trong một hệ thống lớn. Grafana cung cấp một giao diện người dùng trực quan và dễ sử dụng, cho phép theo dõi các thông số như CPU, bộ nhớ, băng thông mạng, thời gian phản hồi API và các chỉ số khác trong thời gian thực. Điều này giúp các quản trị viên dễ dàng phát hiện ra các vấn đề về hiệu suất, tắc nghẽn hay sự cố xảy ra trong hệ thống.

2.1.8.5. Feign Client

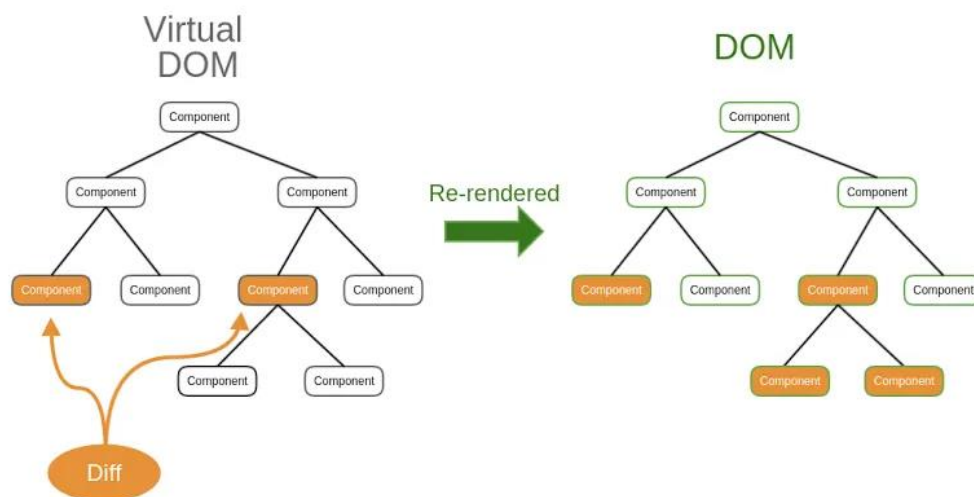
Feign Client là một thư viện Java được sử dụng để đơn giản hóa việc gọi các REST API giữa các dịch vụ trong hệ thống microservices. Feign cung cấp một cách tiện lợi để tạo ra các HTTP request mà không cần phải viết nhiều mã nguồn phức tạp. Nó giúp các dịch vụ giao tiếp của các service với nhau một cách dễ dàng thông qua các API mà không cần phải cấu hình phức tạp về giao thức HTTP.

Trong hệ thống microservices, các dịch vụ thường phải tương tác với nhau thông qua các API RESTful. Feign Client giúp quá trình này trở nên nhanh chóng và dễ bảo trì hơn, bằng cách tạo ra một giao diện đơn giản để thực hiện các yêu cầu HTTP và tự động ánh xạ dữ liệu phản hồi về các đối tượng Java. Điều này không chỉ giúp giảm thiểu mã nguồn cần viết mà còn cải thiện tính dễ đọc và dễ bảo trì của hệ thống.

2.2. Sơ lược về các công nghệ và kiến trúc phía Front-end

2.2.1. ReactJS

ReactJS là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook, giúp các lập trình viên xây dựng giao diện người dùng (UI) một cách hiệu quả và dễ dàng hơn. React được thiết kế để phát triển các ứng dụng web động, nơi giao diện người dùng có sự thay đổi liên tục mà không cần tải lại toàn bộ trang. Điểm nổi bật của React là khả năng tái sử dụng thành phần (component-based architecture), giúp chia nhỏ giao diện phức tạp thành các thành phần đơn lẻ, độc lập, và có thể quản lý dễ dàng. Mỗi thành phần trong React được phát triển riêng lẻ và có thể kết hợp lại với nhau để tạo thành giao diện hoàn chỉnh.



Hình 2.14: Cơ chế hoạt động của Virtual DOM trong React

(Nguồn ảnh: <https://medium.com/naukri-engineering/naukriengineering-virtual-dom-fa8019c626b>)

Một trong những đặc điểm nổi bật nhất của React là cơ chế Virtual DOM (Document Object Model ảo). Khi có thay đổi trong giao diện, React không cần cập nhật trực tiếp lên DOM thật, mà thay vào đó sử dụng Virtual DOM để tính toán và chỉ cập nhật

những phần thay đổi cần thiết. Điều này giúp tối ưu hóa hiệu suất của ứng dụng, đặc biệt là với các trang web có lượng tương tác, thay đổi lớn và nhiều thành phần động.

React cũng hỗ trợ các công nghệ hiện đại như JSX (JavaScript XML), giúp lập trình viên viết mã dễ dàng hơn bằng cách kết hợp HTML và JavaScript trong cùng một tệp, làm cho mã nguồn trở nên trực quan và dễ đọc hơn. Bên cạnh đó, React sử dụng Hooks như `useState`, `useEffect`, giúp quản lý trạng thái và vòng đời của các thành phần một cách đơn giản mà không cần phải sử dụng các class phức tạp như trước.

React không phải là một framework toàn diện như một số công cụ khác (ví dụ: Angular hay Vue), mà chỉ tập trung vào giao diện người dùng. Điều này mang lại sự linh hoạt cho các lập trình viên, cho phép họ tích hợp React với các thư viện hoặc công nghệ khác tùy theo nhu cầu của dự án. Nhờ vào kiến trúc thành phần và khả năng tùy biến cao, React đã trở thành lựa chọn hàng đầu cho việc phát triển các ứng dụng web hiện đại, giúp tăng tính linh hoạt, dễ bảo trì, và mở rộng.

2.2.2. React Router

React Router là một thư viện JavaScript được sử dụng rộng rãi để quản lý việc điều hướng giữa các trang trong ứng dụng React. Nó cho phép tạo ra các tuyến đường (routes) để kết nối các thành phần khác nhau với các URL cụ thể, từ đó giúp người dùng có thể điều hướng dễ dàng giữa các phần khác nhau của ứng dụng mà không cần tải lại toàn bộ trang.

React Router cung cấp các tính năng như:

- + **Dynamic routing:** Tạo ra các tuyến đường động, giúp tùy biến việc điều hướng dựa trên tham số URL.

- + **Nested routing:** Hỗ trợ việc tạo các tuyến đường lồng nhau, giúp tổ chức mã nguồn rõ ràng hơn khi có các giao diện phức tạp.

- + **Route protection:** Hỗ trợ bảo vệ các tuyến đường, chỉ cho phép truy cập khi người dùng đã xác thực, giúp tăng cường bảo mật cho các trang quan trọng.

React Router là một thành phần quan trọng trong việc xây dựng các ứng dụng web SPA (Single Page Application), nơi các nội dung được thay đổi động mà không phải tải lại toàn bộ trang.

2.2.3. Material-UI (MUI)

Material-UI là một thư viện UI component được phát triển theo chuẩn Material Design của Google. MUI cung cấp các thành phần giao diện đã được thiết kế sẵn như Button, TextField, Dialog, Snackbar, giúp tạo ra giao diện người dùng chuyên nghiệp và hiện đại một cách dễ dàng.

Những tính năng chính của Material-UI bao gồm:

- + **Responsive Design:** Các thành phần của Material-UI tự động điều chỉnh kích thước và bố cục theo kích thước màn hình, giúp giao diện web tương thích với nhiều loại thiết bị khác nhau.

- + **Customization:** Thư viện hỗ trợ khả năng tùy chỉnh cao, cho phép lập trình viên dễ dàng thay đổi giao diện của các thành phần sao cho phù hợp với yêu cầu của dự án.

- + **Icons and themes:** Material-UI tích hợp sẵn hệ thống theme và icons, giúp dễ dàng thay đổi giao diện toàn hệ thống chỉ với vài cấu hình.

2.2.4. Axios

Axios là một thư viện JavaScript nổi tiếng được sử dụng để thực hiện các yêu cầu HTTP trong các ứng dụng web. Nó hỗ trợ cả phía client (trình duyệt) và phía server (Node.js). Axios giúp đơn giản hóa việc gửi và nhận dữ liệu từ server, và hoạt động dựa trên cơ chế Promise, giúp xử lý các yêu cầu bất đồng bộ một cách hiệu quả và nhanh chóng.

Các tính năng chính của Axios:

- + **Promise-based:** Axios giúp xử lý các yêu cầu HTTP bất đồng bộ một cách dễ dàng thông qua cơ chế Promise, giúp việc quản lý luồng dữ liệu trở nên đơn giản.

- + **Tự động chuyển đổi JSON:** Khi gửi và nhận dữ liệu JSON, Axios tự động chuyển đổi mà không cần xử lý thủ công.

- + **Quản lý lỗi và Timeout:** Axios hỗ trợ tính năng timeout cho các yêu cầu, giúp ngăn ngừa việc ứng dụng bị treo do yêu cầu kéo dài quá lâu, và cung cấp cơ chế xử lý lỗi khi yêu cầu thất bại.

+ **Interceptors:** Axios cung cấp khả năng sử dụng interceptors để chỉnh sửa yêu cầu hoặc phản hồi trước khi xử lý, điều này rất hữu ích cho việc thêm token xác thực hoặc quản lý lỗi.

2.2.5. Fetch API

Fetch API là một chuẩn API được tích hợp sẵn trong các trình duyệt hiện đại, cho phép thực hiện các yêu cầu HTTP đến server. Fetch API cung cấp một cách đơn giản và hiệu quả để thực hiện các yêu cầu GET, POST, PUT, và DELETE từ front-end và cũng dựa trên cơ chế Promise, giúp xử lý các yêu cầu bất đồng bộ một cách rõ ràng và mạch lạc.

Các tính năng chính của Fetch API:

+ **Native support:** Fetch API được tích hợp sẵn trong các trình duyệt mà không cần cài đặt thêm bất kỳ thư viện nào, giúp dễ dàng sử dụng cho các ứng dụng nhỏ và đơn giản.

+ **Promise-based:** Fetch API hoạt động dựa trên Promise, giúp việc xử lý các thao tác bất đồng bộ trở nên đơn giản và dễ hiểu.

+ **Cấu trúc đơn giản:** Fetch có cú pháp đơn giản, dễ sử dụng, rất phù hợp cho những ứng dụng không yêu cầu nhiều tính năng nâng cao.

Fetch API là giải pháp tốt cho các ứng dụng nhỏ hoặc đơn giản, nơi không cần đến các tính năng nâng cao như quản lý lỗi phức tạp hay cài đặt timeout.

Chương 3. TRIỂN KHAI HỆ THỐNG

3.1. Giới thiệu hệ thống

Hệ thống website học tập trực tuyến được thiết kế để cung cấp một nền tảng học tập trực tuyến hiện đại, hiệu quả và linh hoạt. Mục tiêu của hệ thống là xây dựng một nền tảng học tập cho phép giảng viên và học viên tương tác, chia sẻ tài liệu, quản lý khóa học, bài tập, và đánh giá kết quả học tập một cách dễ dàng và hiệu quả.

Hệ thống được xây dựng dựa trên kiến trúc microservices, một mô hình kiến trúc phần mềm hiện đại, trong đó mỗi chức năng của hệ thống được chia thành các dịch vụ độc lập, có thể triển khai và hoạt động riêng biệt. Điều này giúp hệ thống linh hoạt, dễ bảo trì, mở rộng và đảm bảo tính ổn định khi cần phát triển thêm các tính năng mới.

Các tính năng chính của hệ thống bao gồm:

- + **Quản lý khóa học:** Giảng viên có thể tạo và quản lý các khóa học, cập nhật nội dung bài giảng, bài tập, và tài liệu học tập.

- + **Quản lý học viên:** Học viên có thể đăng ký khóa học, xem bài giảng, nộp bài tập và theo dõi kết quả học tập của mình.

- + **Diễn đàn thảo luận:** Học viên và giảng viên có thể tương tác qua các bài viết và bình luận trong diễn đàn trao đổi kiến thức.

- + **Trò chuyện trực tuyến:** Tính năng chat trực tuyến giữa các học viên và giảng viên, hỗ trợ giải đáp thắc mắc kịp thời.

- + **Tích hợp trí tuệ nhân tạo:** Hệ thống chatbot AI giúp học viên giải đáp các câu hỏi liên quan đến nội dung khóa học hoặc chức năng của hệ thống.

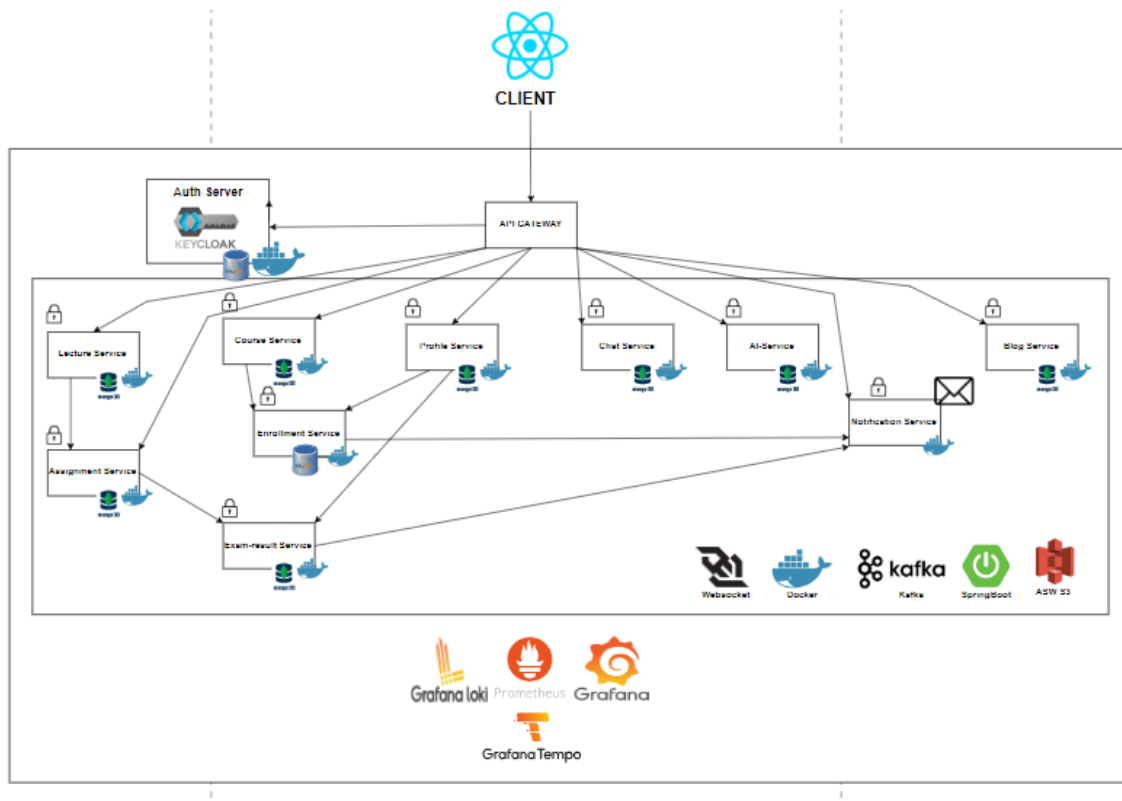
Với kiến trúc microservices, mỗi tính năng được triển khai dưới dạng một dịch vụ riêng biệt, có thể phát triển, bảo trì và mở rộng mà không ảnh hưởng đến toàn bộ hệ thống. Hệ thống sử dụng các công nghệ hiện đại như Spring Boot cho back-end, ReactJS cho front-end, cùng với các dịch vụ tích hợp như Docker, Kafka, Keycloak để quản lý xác thực, phân quyền, và bảo mật.

3.2. Kiến trúc hệ thống

Kiến trúc microservices là một phương pháp tiếp cận trong phát triển phần mềm, trong đó hệ thống được chia ra nhỏ thành nhiều dịch vụ nhỏ, độc lập và có thể triển khai một cách riêng lẻ. Mỗi microservice sẽ đảm nhận một chức năng cụ thể trong hệ thống và có thể tương tác với các dịch vụ khác thông qua các giao thức nhẹ như HTTP, RESTful API, hoặc các message broker như Kafka, RabbitMQ.

Hệ thống này được xây dựng dựa trên kiến trúc microservices, một mô hình kiến trúc phần mềm tiên tiến, trong đó mỗi thành phần chức năng của hệ thống được chia nhỏ thành các dịch vụ độc lập, có khả năng hoạt động và triển khai riêng biệt. Kiến trúc này giúp hệ thống trở nên linh hoạt, dễ bảo trì, mở rộng, và đảm bảo sự ổn định khi có nhu cầu phát triển thêm các tính năng mới.

3.2.1. Các thành phần chính của hệ thống



Hình 3.1: Kiến trúc Microservices hệ thống website học tập trực tuyến

- **API Gateway:** Là thành phần trung gian, giúp điều phối và định tuyến các yêu cầu từ người dùng đến các dịch vụ thích hợp. API Gateway còn giúp che giấu các microservices phía sau, không cần phải lộ trực tiếp ra bên ngoài.

- **Auth Server:** Là thành phần chịu trách nhiệm xác thực và phân quyền người dùng trong hệ thống. Nó đảm bảo chỉ những người dùng hợp lệ mới có thể truy cập vào các dịch vụ bằng cách kiểm tra thông tin đăng nhập và cấp phát token JWT sau khi xác thực thành công. Trong hệ thống, Keycloak được sử dụng làm Auth Server, hỗ trợ Single Sign-On (SSO), quản lý người dùng, vai trò (role-based access), và cấp phát token theo tiêu chuẩn OAuth 2.0 và OpenID Connect. Keycloak giúp bảo mật hệ thống, đơn giản hóa việc xác thực và phân quyền mà không cần phát triển từ đầu.

- **Course Service:** Quản lý toàn bộ thông tin và chức năng liên quan đến khóa học. Bao gồm việc tạo, cập nhật, xóa và hiển thị danh sách các khóa học, cùng với việc quản lý mô tả, thông tin giảng viên, tài liệu kèm theo của khóa học. Course Service cũng cung cấp API để truy xuất thông tin khóa học dựa trên các tiêu chí tìm kiếm khác nhau, giúp người học dễ dàng tìm kiếm và đăng ký khóa học phù hợp.

- **Lecture Service :** Quản lý các bài giảng trong từng khóa học. Bao gồm việc thêm mới, cập nhật, và xóa các bài giảng. Lecture Service cũng chịu trách nhiệm về nội dung chi tiết của bài giảng, như tài liệu giảng dạy, video hướng dẫn, và các tài nguyên đi kèm. Dịch vụ này giúp giảng viên tổ chức nội dung học tập một cách khoa học và giúp học viên tiếp cận bài giảng theo trình tự học tập.

- **Assignment Service:** Quản lý toàn bộ quy trình liên quan đến bài tập, từ việc tạo bài tập, cập nhật nội dung bài tập cho đến việc cung cấp chức năng nộp bài cho học viên.

- **Exam Result Service:** Chịu trách nhiệm quản lý kết quả bài làm của học viên, bao gồm việc lưu trữ, cập nhật và hiển thị kết quả sau khi chấm điểm. Exam Result Service còn cung cấp các báo cáo chi tiết về hiệu suất của từng học viên, cho phép giảng viên đánh giá kết quả học tập và học viên theo dõi sự tiến bộ của mình.

- **Enrollment Service:** Quản lý việc đăng ký khóa học của học viên. Dịch vụ này bao gồm việc xác minh thông tin đăng ký, kiểm tra tính hợp lệ và phân quyền truy cập vào các tài nguyên liên quan đến khóa học. Enrollment Service đảm bảo rằng chỉ những học viên đã đăng ký hợp lệ mới có thể truy cập vào nội dung khóa học. Ngoài ra,

service này cũng tích hợp công thanh toán Vnpay giúp người dùng có thể nạp tiền phục vụ cho việc mua các khóa học.

- **Chat Service:** Cung cấp chức năng trò chuyện trực tuyến giữa người dùng, sử dụng WebSocket để thực hiện giao tiếp theo thời gian thực. Tính năng này cho phép người dùng tương tác ngay lập tức với nhau, thảo luận và giải đáp các vấn đề liên quan đến khóa học một cách trực tiếp.

- **AI Service:** Quản lý các chức năng liên quan đến trí tuệ nhân tạo trong hệ thống, bao gồm chatbot hỗ trợ học tập. Dịch vụ này sử dụng mô hình AI từ OpenAI và tùy chỉnh để phù hợp với nhu cầu cụ thể của hệ thống, đồng thời thu thập thông tin từ các cuộc trò chuyện để phục vụ cho việc nghiên cứu và phát triển các tính năng trong tương lai.

- **Blog Service:** Quản lý diễn đàn nơi người dùng có thể thảo luận, chia sẻ kiến thức, đặt câu hỏi, và trả lời các câu hỏi từ các học viên khác. Blog Service hỗ trợ các tính năng như tạo bài viết, bình luận, thích, và phân loại theo chủ đề để người dùng dễ dàng tìm kiếm và tham gia thảo luận về các chủ đề quan tâm.

- **Notification Service:** Quản lý toàn bộ thông báo trong hệ thống, từ việc gửi thông báo trong hệ thống đến gửi email cho người dùng khi có sự kiện quan trọng .

- **AWS S3:** Được sử dụng để lưu trữ và quản lý các tài nguyên đa phương tiện trong hệ thống, như hình ảnh, video, và tài liệu học tập. AWS S3 cung cấp khả năng lưu trữ an toàn, mở rộng và có thể truy cập từ mọi nơi. Các dịch vụ như Course Service và Lecture Service có thể tương tác với AWS S3 để tải lên và tải xuống tài liệu học tập cho các khóa học.

3.2.2. Cơ chế xác thực và phân quyền của hệ thống

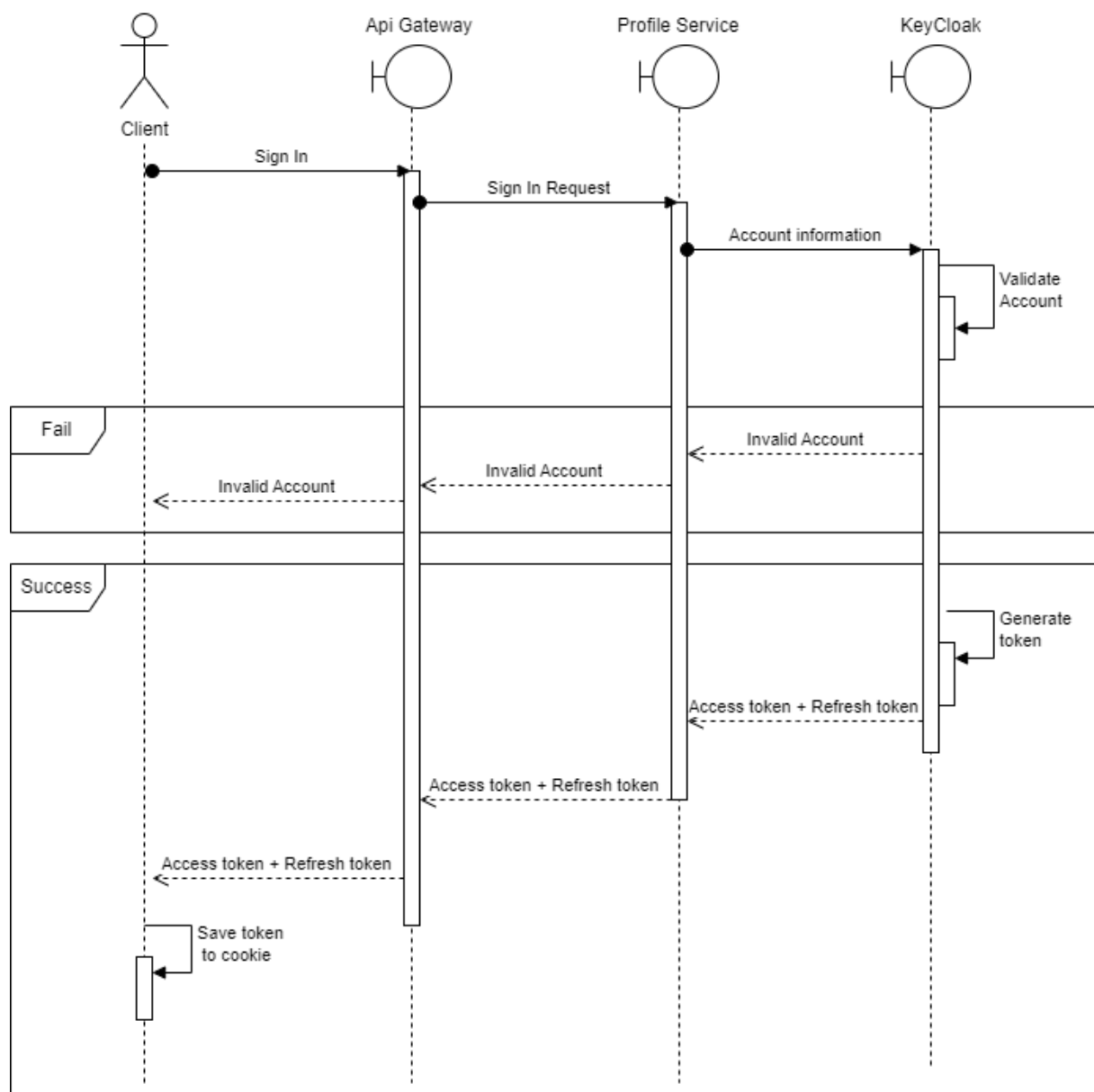
Hệ thống sử dụng cơ chế xác thực dựa trên chuỗi xác thực JWT (JSON Web Token). Khi người dùng tiến hành đăng nhập vào hệ thống, thông tin tài khoản sẽ được gửi từ phía người dùng lên hệ thống thông qua API Gateway. API Gateway có nhiệm vụ điều phối và định tuyến yêu cầu này đến Profile Service. Profile Service sẽ tiến hành gửi yêu cầu lên Keycloak để xác thực thông tin tài khoản của người dùng.

Keycloak, được tích hợp trong hệ thống, có nhiệm vụ kiểm tra tính hợp lệ của tài khoản, đảm bảo rằng người dùng có quyền truy cập vào hệ thống. Nếu thông tin tài khoản hợp lệ, Keycloak sẽ tạo và trả về hai loại chuỗi xác thực: Access Token và Refresh Token. Access Token sẽ được sử dụng để chứng thực và phân quyền cho phục vụ cho việc gọi các service khác cho người dùng

Khi Access Token hết hạn, hệ thống sẽ sử dụng Refresh Token để yêu cầu Keycloak cấp lại Access Token mới mà không cần người dùng đăng nhập lại. Chuỗi Refresh Token để giúp gia hạn phiên làm việc của người dùng mà không làm gián đoạn trải nghiệm sử dụng hệ thống.

Trong mỗi lần người dùng gửi yêu cầu lên hệ thống, Access Token sẽ được gửi kèm trong header của request. Phía server, thông qua API Gateway và Profile Service, sẽ kiểm tra tính hợp lệ của Access Token. Nếu hợp lệ, hệ thống sẽ xử lý dữ liệu và trả về thông tin phù hợp với quyền hạn của người dùng mà Access Token quy định.

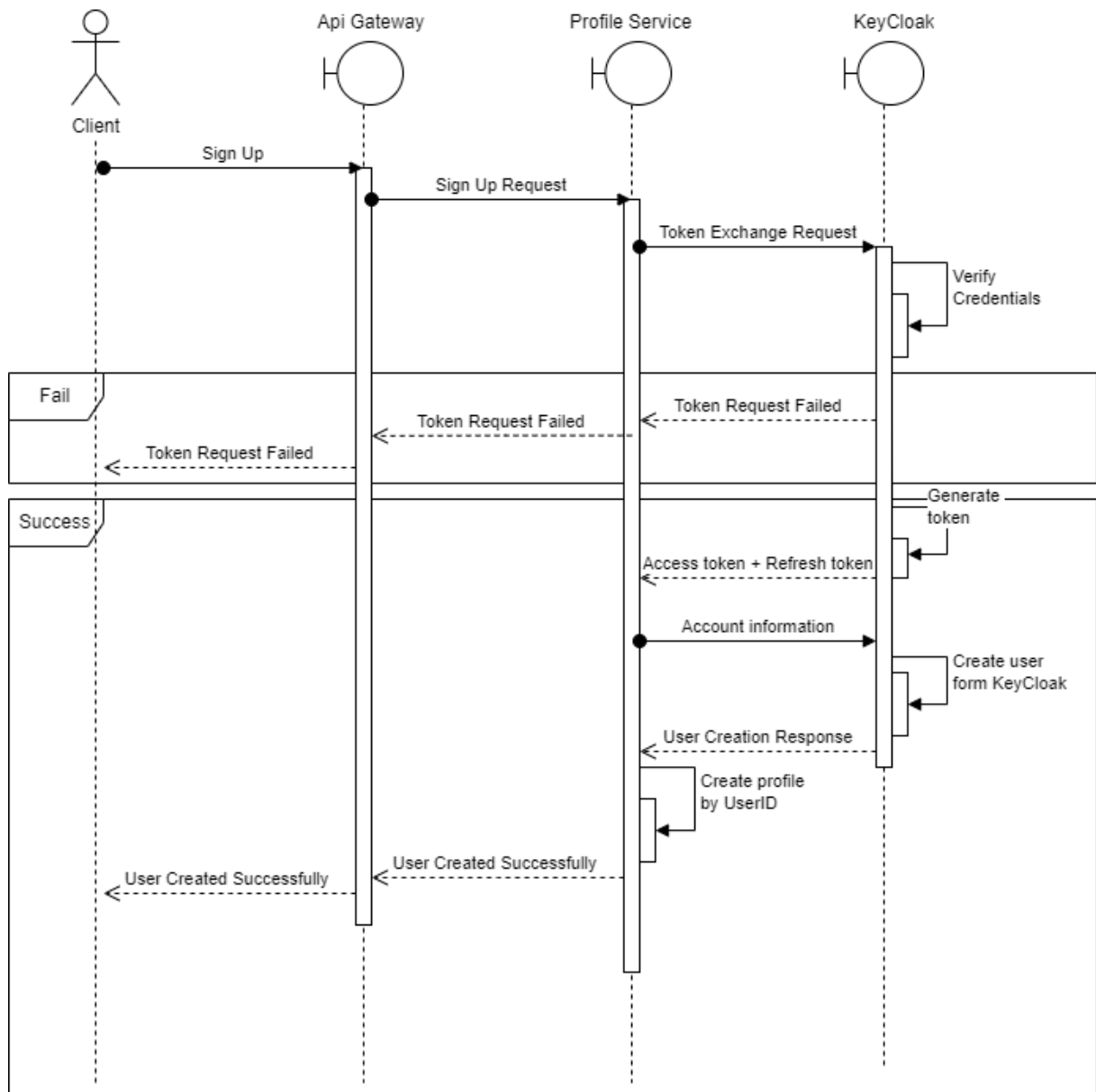
Keycloak giúp đảm bảo an ninh cho hệ thống và đơn giản hóa việc xác thực, phân quyền, hỗ trợ các tiêu chuẩn bảo mật hiện đại như OAuth 2.0 và OpenID Connect.



Hình 3.2: Cơ chế xác thực và phân quyền của hệ thống

3.2.3. Cơ chế tạo tài khoản cho người dùng của hệ thống

Cơ chế tạo tài khoản trong hệ thống là một cơ chế đặc biệt hoạt động dựa trên sự tương tác chặt chẽ giữa các service liên quan. Client gửi yêu cầu đăng ký tài khoản, thông tin đăng ký tài khoản sẽ đến API Gateway và được định tuyến đến Profile Service. Tại đây Profile Service sẽ gửi yêu cầu đến KeyCloak sau khi có token Profile Service sẽ tiến hành gửi thông tin tài khoản đến KeyCloak để tạo tài khoản. Sau khi tạo tài khoản thành công Profile Service sẽ lấy UserId từ phản hồi mà KeyCloak trả về để tiến hành tạo profile ở cơ sở dữ liệu và trả về thông báo tạo thành công cho client. KeyCloak chỉ lưu trữ tên đăng nhập, mật khẩu, email của người dùng, các thông tin còn lại sẽ được lưu ở Profile Service thông qua UserId, điều này sẽ giúp tách bạch việc lưu trữ, dễ dàng mở rộng, và hỗ trợ việc một người dùng có nhiều profile trong tương lai.



Hình 3.3: Cơ chế đăng kí tài khoản của hệ thống

3.2.4. Cơ chế giao tiếp đồng bộ của các service

Cơ chế giao tiếp đồng bộ trong hệ thống sử dụng Feign Client để thực hiện giao tiếp đồng bộ. Feign Client giúp đơn giản hóa việc gọi API giữa các service bằng cách cung cấp một API dễ sử dụng, tương tự như việc gọi một phương thức thông thường trong Java. Điều này giúp giảm tải công việc viết mã thủ công cho việc tạo ra và xử lý các yêu cầu HTTP. Hệ thống sử dụng cơ chế này cho các yêu cầu phải nhận phản hồi ngay để xử lý như, đăng ký khóa học, nạp tiền vào tài khoản.

3.2.5. Cơ chế giao tiếp bất đồng bộ của các service

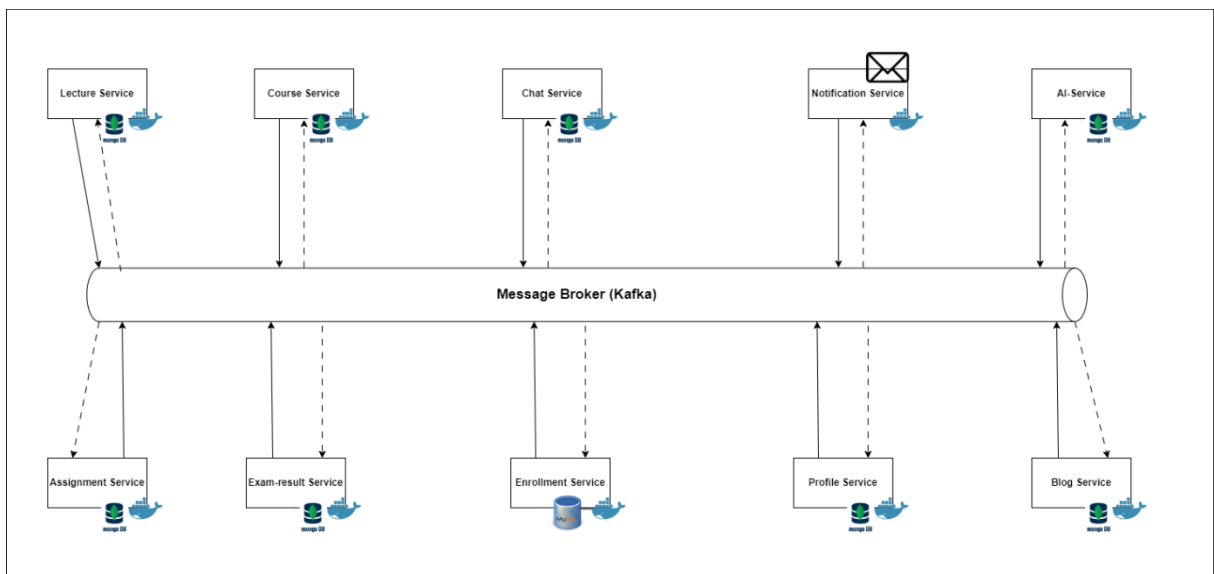
Cơ chế gửi bất đồng bộ trong hệ thống sử dụng message broker của Kafka để gửi hoặc nhận các thông điệp mà không cần phải phản hồi ngay lập tức. Các service có thể đóng vai trò là consumer (người tiêu dùng) hoặc producer (nhà sản xuất) để tương tác bất đồng bộ với nhau thông qua message broker. Điều này giúp các dịch vụ hoạt động độc lập với nhau, không cần phải chờ đợi các phản hồi.

Nhờ việc sử dụng cơ chế giao tiếp bất đồng bộ, hệ thống tăng tính chịu lỗi, tăng khả năng xử lý lượng yêu cầu lớn trong một thời điểm và dễ dàng mở rộng. Kafka giúp đảm bảo rằng các thông điệp được chuyển đi một cách tin cậy, đảm bảo không bị mất trong quá trình xử lý. Điều này rất quan trọng trong các hệ thống microservices, nơi có nhiều tác vụ và sự kiện cần được xử lý đồng thời.

Ví dụ về việc sử dụng cơ chế này trong hệ thống:

- + Khi một người dùng đăng ký tài khoản thành công, Profile Service sẽ gửi một thông điệp bất đồng bộ đến Notification Service để gửi mail chào mừng người dùng.

- + Khi một khóa học hay người dùng bị xóa, hệ thống sẽ gửi các thông điệp bất đồng bộ đến các service liên quan để xóa hoặc cập nhật trạng thái.



Hình 3.4: Cơ chế giao tiếp bất đồng bộ thông qua message broker của hệ thống

3.3. Cơ sở dữ liệu của các service

3.3.1. Cơ sở dữ liệu của Profile Service

STT	TÊN	Ý NGHĨA	KIỂU DỮ LIỆU
1	profileId	Lưu id người dùng sử dụng cho hệ thống	String
2	userId	Lưu id người dùng từ KeyCloak dùng cho việc xác thực và phân quyền	String
3	email	Lưu email mail của người dùng	String
4	username	Lưu tên đăng nhập của người dùng	String
5	firstName	Lưu tên của người dùng	String
6	lastName	Lưu họ của người dùng	String
7	avatarUrl	Lưu đường dẫn ảnh đại diện của người dùng	String
8	dob	Lưu ngày sinh của người dùng	LocalDate
9	coin	Lưu số tiền của người dùng	BigDecimal
10	roles	Lưu vai trò của người dùng	List<String>

Bảng 3.1: Bảng dữ liệu collection profile

3.3.2. Cơ sở dữ liệu của Course Service

STT	TÊN	Ý NGHĨA	KIỂU DỮ LIỆU
1	id	Lưu id của khóa học	String
2	name	Lưu tên của khóa học	String
3	description	Lưu thông tin mô tả khóa học	String
4	price	Lưu giá của khóa học	BigDecimal

5	category	Lưu loại của khóa học	String
6	imageUrl	Lưu đường dẫn của ảnh bìa khóa học	String
7	createdAt	Lưu ngày tạo khóa học	Date
8	updatedAt	Lưu ngày chỉnh sửa khóa học	Date
9	tags	Lưu các nhãn của khóa học	List<String>
10	teacherId	Lưu id của giáo viên	String

Bảng 3.2: Bảng dữ liệu collection course

3.3.3. Cơ sở dữ liệu của Lecture Service

STT	TÊN	Ý NGHĨA	Kiểu Dữ Liệu
1	id	Lưu id của bài giảng	String
2	courseId	Lưu id của khóa học chứa bài giảng	String
3	title	Lưu nội dung chính của bài giảng	String
4	content	Lưu nội dung bài giảng	String
5	fileUrl	Lưu đường dẫn đến file đính kèm của bài giảng	String
6	videoUrls	Lưu đường dẫn đến các video của bài giảng	List<String>
7	index	Lưu thứ tự bài giảng trong khóa học	Int
8	createdAt	Lưu ngày tạo bài giảng	Date
9	updatedAt	Lưu ngày chỉnh sửa của bài giảng	Date

Bảng 3.3: Bảng dữ liệu collection lecture

3.3.4. Cơ sở dữ liệu của Assignment Service

STT	TÊN	Ý NGHĨA	Kiểu Dữ Liệu
1	id	Lưu id của bài tập	String
2	lectureId	Lưu id của bài giảng chứa bài tập	String
3	title	Lưu tiêu đề của bài tập	String
4	questions	Lưu danh sách các câu hỏi của bài tập (tham chiếu đến bảng Question)	List<Question>

Bảng 3.4: Bảng dữ liệu collection assignment

STT	TÊN	Ý NGHĨA	Kiểu Dữ Liệu
1	questionText	Lưu id của bài tập	String
2	options	Lưu danh sách các lựa chọn	List<String>
3	correctAnswer	Lưu tiêu đề của bài tập	String

Bảng 3.5: Bảng dữ liệu collection question

3.3.5. Cơ sở dữ liệu của Exam Result Service

STT	TÊN	Ý NGHĨA	Kiểu Dữ Liệu
1	id	Lưu id của bài tập	String
2	userId	Lưu id của người dùng	String
3	assignmentId	Lưu id của bài tập	String
4	questionResults	Lưu danh sách kết quả của từng câu hỏi	List<QuestionResult>
5	score	Lưu điểm số của người dùng	Int

Bảng 3.6: Bảng dữ liệu collection assignmentResponse

STT	TÊN	Ý NGHĨA	KIỂU DỮ LIỆU
1	questionText	Lưu nội dung câu hỏi	String
2	userAnswer	Lưu câu trả lời của người dùng	String
3	correctAnswer	Lưu đáp án đúng	String
4	isCorrect	Lưu trạng thái đúng hay sai của câu trả lời	Boolean

Bảng 3.7: Bảng dữ liệu collection questionResult

3.3.6. Cơ sở dữ liệu của Enrollment Service

STT	TÊN	Ý NGHĨA	KIỂU DỮ LIỆU
1	id	Lưu id của lần đăng ký	Long
2	studentId	Lưu id của người dùng đăng ký	String
3	courseId	Lưu id của khóa học được đăng ký	String
4	enrollmentDate	Lưu thời gian đăng ký khóa học	LocalDateTime

Bảng 3.8: Bảng dữ liệu collection enrollment

3.3.7. Cơ sở dữ liệu của Chat Service

STT	TÊN	Ý NGHĨA	KIỂU DỮ LIỆU
1	id	Lưu id của tin nhắn	String
2	senderId	Lưu id của người gửi	String
3	receiverId	Lưu id của người nhận	String

4	content	Lưu nội dung của tin nhắn	String
5	Timestamp	Lưu thời gian gửi tin nhắn	LocalDateTime

Bảng 3.9: Bảng dữ liệu collection chat

3.3.8. Cơ sở dữ liệu của AI Service

STT	TÊN	Ý NGHĨA	KIỂU DỮ LIỆU
1	id	Lưu id lịch sử trò chuyện	String
2	profileId	Lưu id của người gửi	String
3	userMessage	Lưu nội dung tin nhắn của người dùng	String
4	gptResponse	Lưu nội dung phản hồi từ Ai	String
5	timestamp	Lưu thời gian gửi tin nhắn	LocalDateTime

Bảng 3.10: Bảng dữ liệu collection chatAi

3.3.9. Cơ sở dữ liệu của Blog Service

STT	TÊN	Ý NGHĨA	KIỂU DỮ LIỆU
1	postId	Lưu id của bài viết	String
2	authorId	Lưu id của người viết	String
3	title	Lưu tiêu đề của bài viết	String

4	content	Lưu nội dung của bài viết	String
5	imageUrl	Lưu đường dẫn ảnh của bài viết	String
6	createdAt	Lưu thời gian tạo bài viết	LocalDateTime
7	updatedAt	Lưu thời gian cập nhật bài viết	LocalDateTime

Bảng 3.11: Bảng dữ liệu collection blog

STT	TÊN	Ý NGHĨA	KIỂU DỮ LIỆU
1	commentId	Lưu id của bình luận	String
2	postId	Lưu ID của bài viết liên kết	String
3	commentParent	Lưu ID của bình luận cha (nếu có)	String
4	authorId	Lưu ID của người viết bình luận	String
5	authorName	Lưu tên của người viết bình luận	String
6	authorAvatar	Lưu đường dẫn đến ảnh đại diện của người viết	String
7	content	Lưu nội dung bình luận	String
8	createdAt	Lưu thời gian tạo bình luận	LocalDateTime

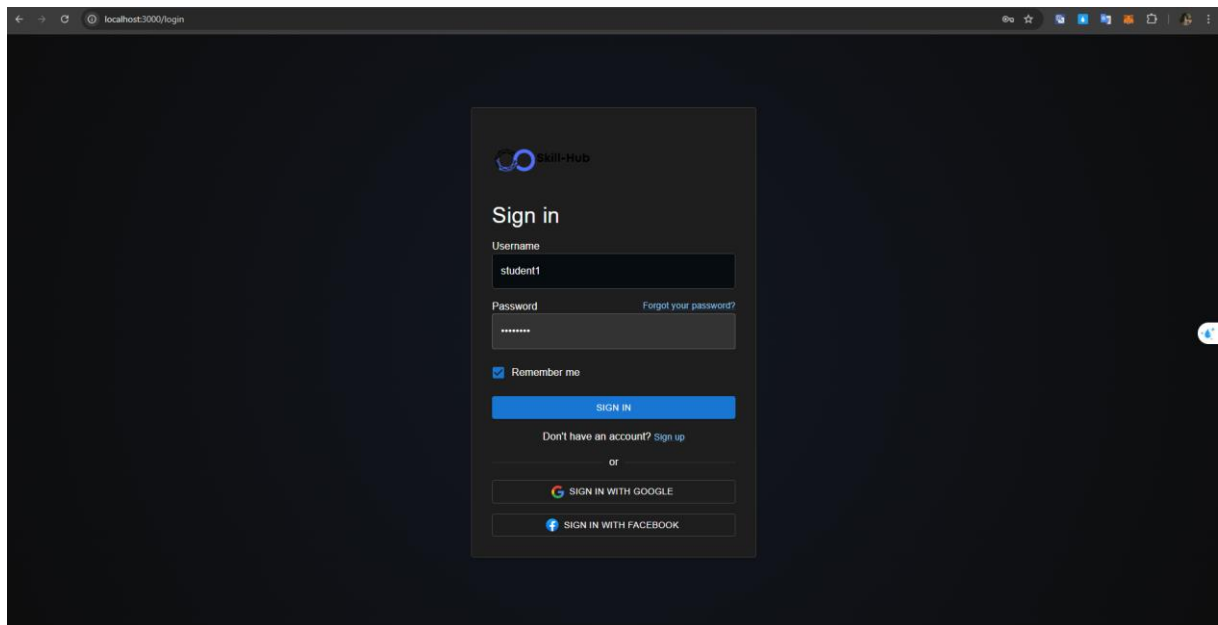
Bảng 3.12: Bảng dữ liệu collection comment

STT	TÊN	Ý NGHĨA	Kiểu dữ liệu
1	interactionId	Lưu id của tương tác	String
2	postId	Lưu id bài viết hoặc bình luận	String
3	userId	Lưu id của người tương tác	String
4	interactionType	Lưu loại tương tác	String

Bảng 3.13: Bảng dữ liệu collection interaction

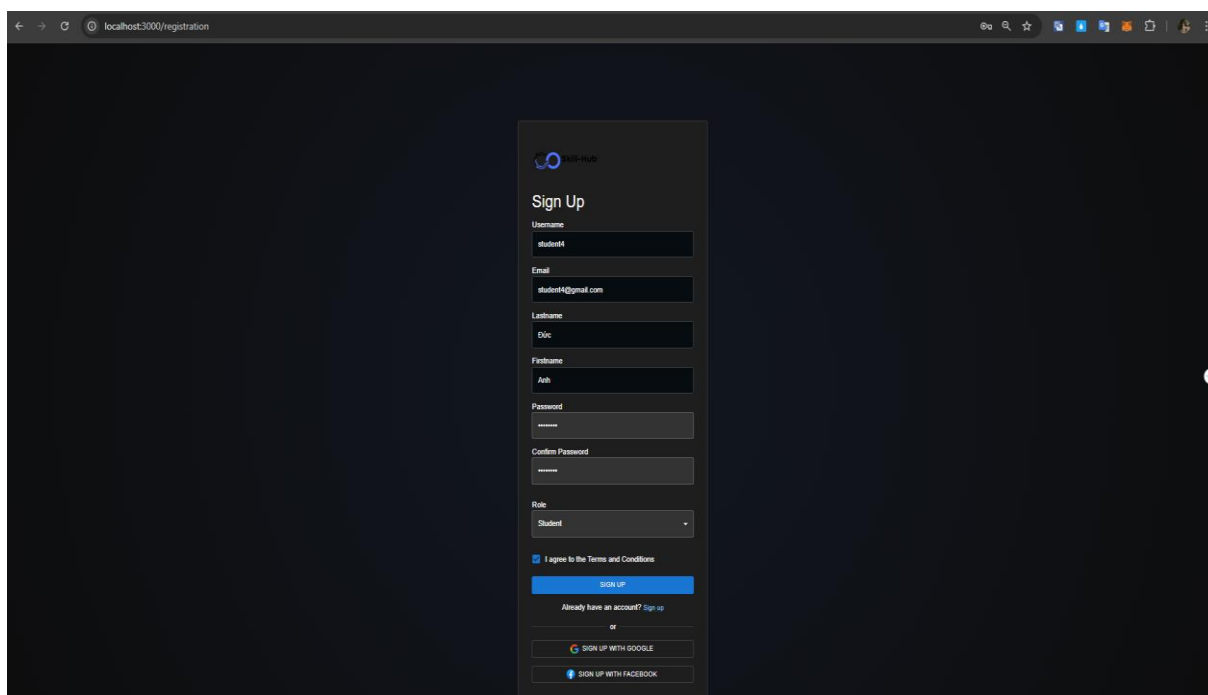
3.4. Giao diện của hệ thống

3.4.1. Giao diện trang đăng nhập



Hình 3.5: Giao diện trang đăng nhập

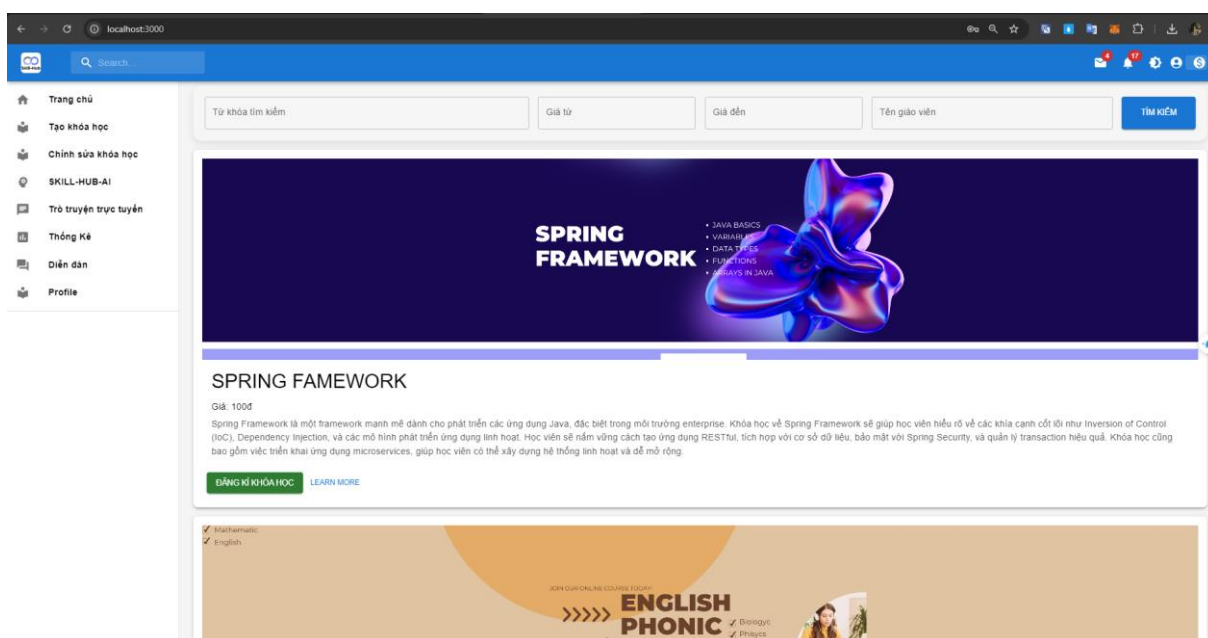
3.4.2. Giao diện trang đăng kí



The screenshot shows a web browser at localhost:3000/registration. The page has a dark background. A central white box contains the 'Sign Up' form. The form includes the following fields: Username (filled with 'student4'), Email (filled with 'student4@gmail.com'), Lastname (filled with 'Đức'), Firstname, Auth, Password (masked with dots), Confirm Password (masked with dots), Role (a dropdown menu with 'Student' selected), and a checkbox for 'I agree to the Terms and Conditions'. Below the form are buttons for 'SIGN UP', 'Already have an account? Sign up', and social login options: 'SIGN UP WITH GOOGLE' and 'SIGN UP WITH FACEBOOK'.

Hình 3.6: Giao diện trang đăng kí

3.4.3. Giao diện trang chủ



Hình 3.7: Giao diện trang chủ

3.4.4. Giao diện tạo khóa học cho giáo viên

TẠO KHÓA HỌC

Tên Khóa Học *
English Phonic

Mô tả khóa học *
Join our English Phonic course today and embark on an exciting journey to enhance your language skills. Our dedicated instructors and vibrant learning community will support you every step of the way, helping you improve your pronunciation, vocabulary, and overall

Giá của khóa học *
50

Tags (nhấn Enter or comma để thêm)
English

English X Tiếng Anh X Tiếng Anh cho người đi làm X

Giảng viên: Thùy Kim

TẢI ẢNH CHO KHÓA HỌC

TẠO KHÓA HỌC

Hình 3.8: Giao diện trang tạo khóa học cho giáo viên

3.4.3. Giao diện tạo khóa học cho admin

TẠO KHÓA HỌC

Tên Khóa Học *
SPRING FAMEWORK

Mô tả khóa học *
đựng RESTful, tích hợp với cơ sở dữ liệu, bảo mật với Spring Security, và quản lý transaction hiệu quả. Khóa học cũng bao gồm việc triển khai ứng dụng microservices, giúp học viên có thể xây dựng hệ thống linh hoạt và dễ mở rộng.

Giá của khóa học *
100

Tags (nhấn Enter or comma để thêm)
Microservice

Java X Thử Long X Spring Famework X Backend X

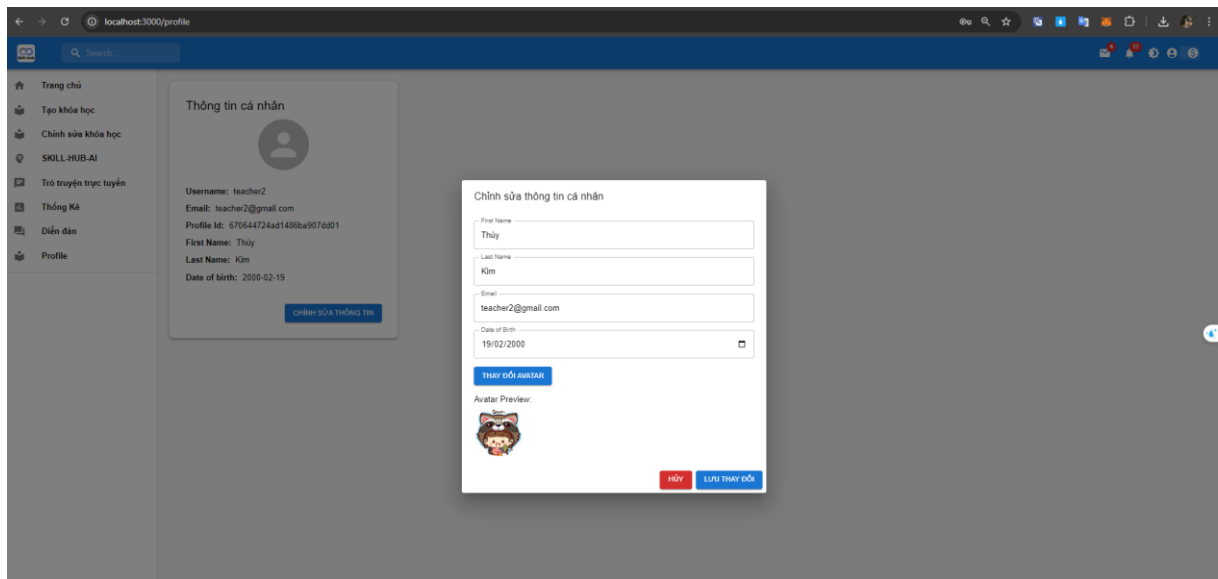
Chọn giảng viên *
Long Trần Quốc

TẢI ẢNH CHO KHÓA HỌC

TẠO KHÓA HỌC

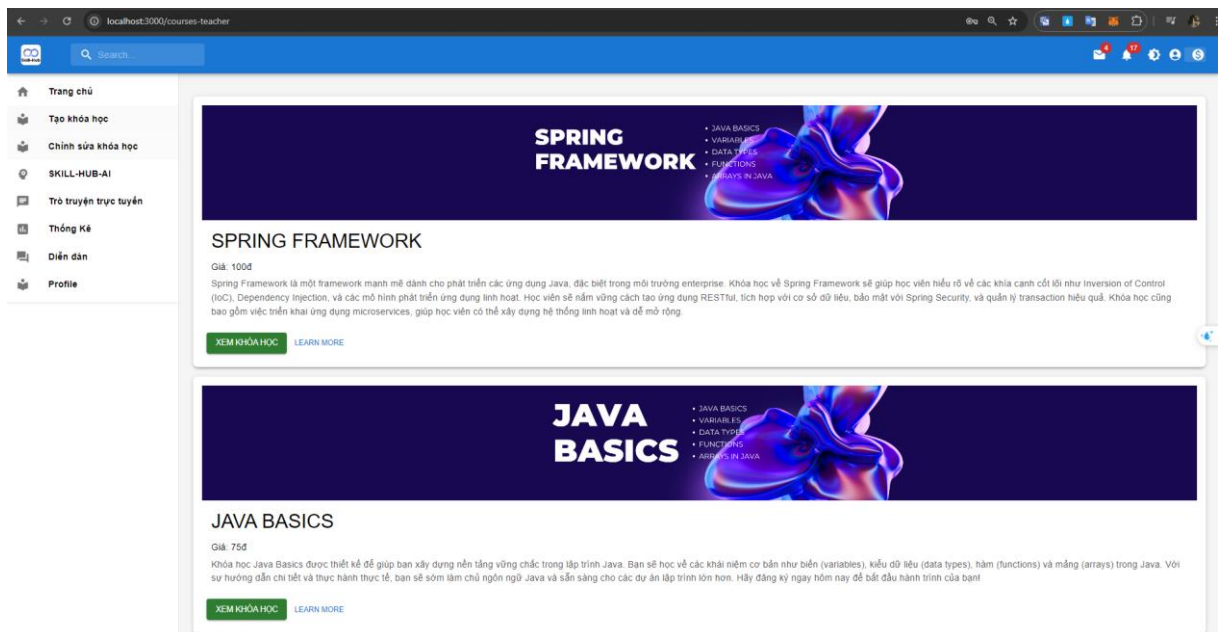
Hình 3.9: Giao diện trang tạo khóa học cho người quản trị

3.4.5. Giao diện chỉnh sửa thông tin cá nhân



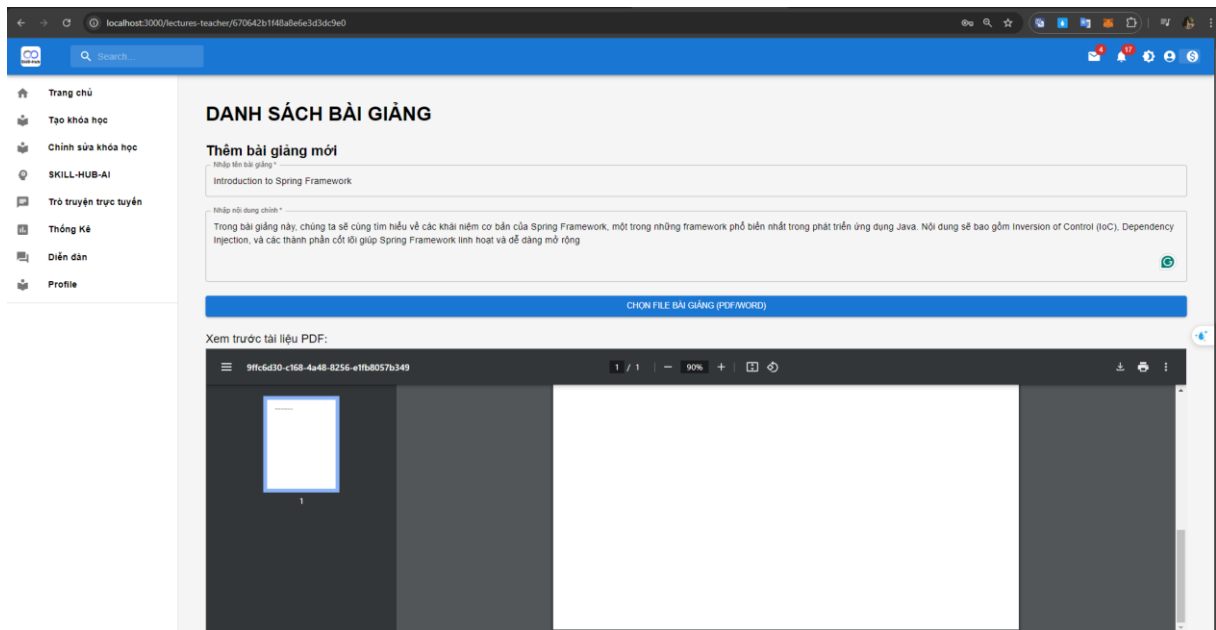
Hình 3.10: Giao diện trang chỉ sửa thông tin người dùng

3.4.6. Giao diện chỉnh sửa khóa học của giáo viên



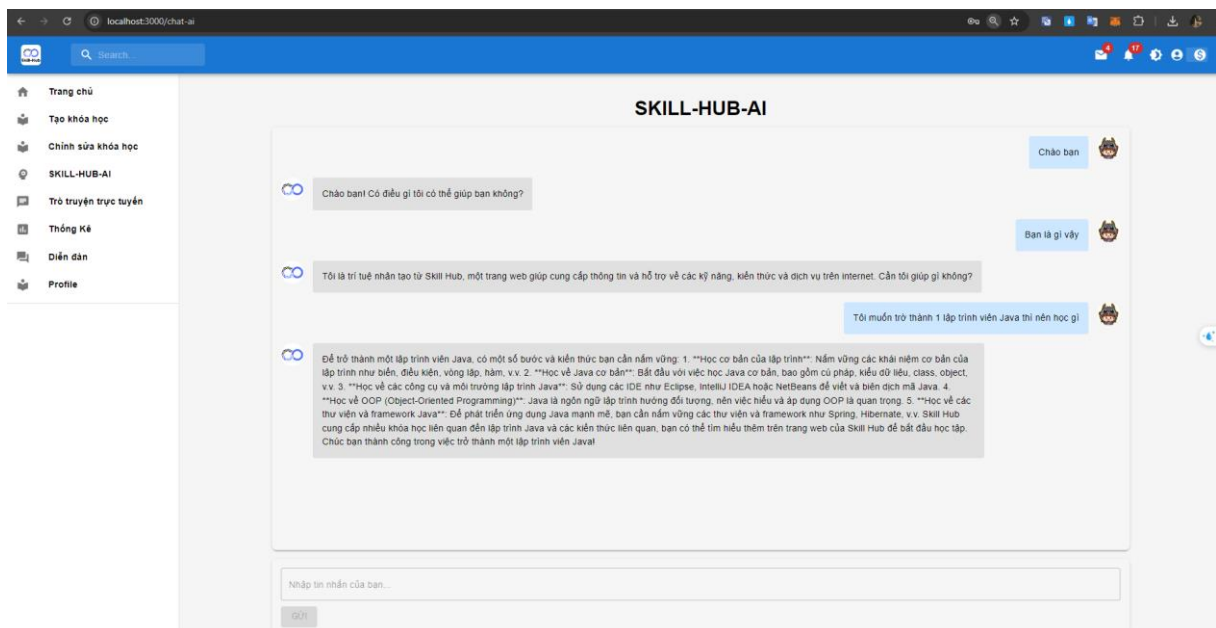
Hình 3.11: Giao diện trang chỉnh sửa khóa học cho giáo viên

3.4.7. Giao diện thêm bài giảng cho giáo viên



Hình 3.12: Giao diện trang thêm bài giảng cho giáo viên

3.4.8. Giao diện trò chuyện trực tuyến với AI



Hình 3.13: Giao diện trang trò chuyện với trí tuệ nhân tạo

3.4.9. Giao diện thêm bài tập trong bài giảng

The screenshot shows a web application interface for adding exercises. On the left is a sidebar with navigation links: Trang chủ, Tạo khóa học, Chính sửa khóa học, SKILL-HUB.AI, Trò chuyện trực tuyến, Thống kê, Diễn đàn, and Profile. The main content area is titled 'Bài Tập:' and contains a form for adding a new exercise. The form includes a title field, a description field, and two question sections. Each question section has a text input for the question and a list of multiple-choice options. The first question is about the main components of the Spring Framework, and the second is about the parts of Spring responsible for managing bean dependencies. At the bottom of the form is a blue button labeled 'THÊM CÂU HỎI' and a blue bar at the very bottom labeled 'THÊM MÀ TẬP'.

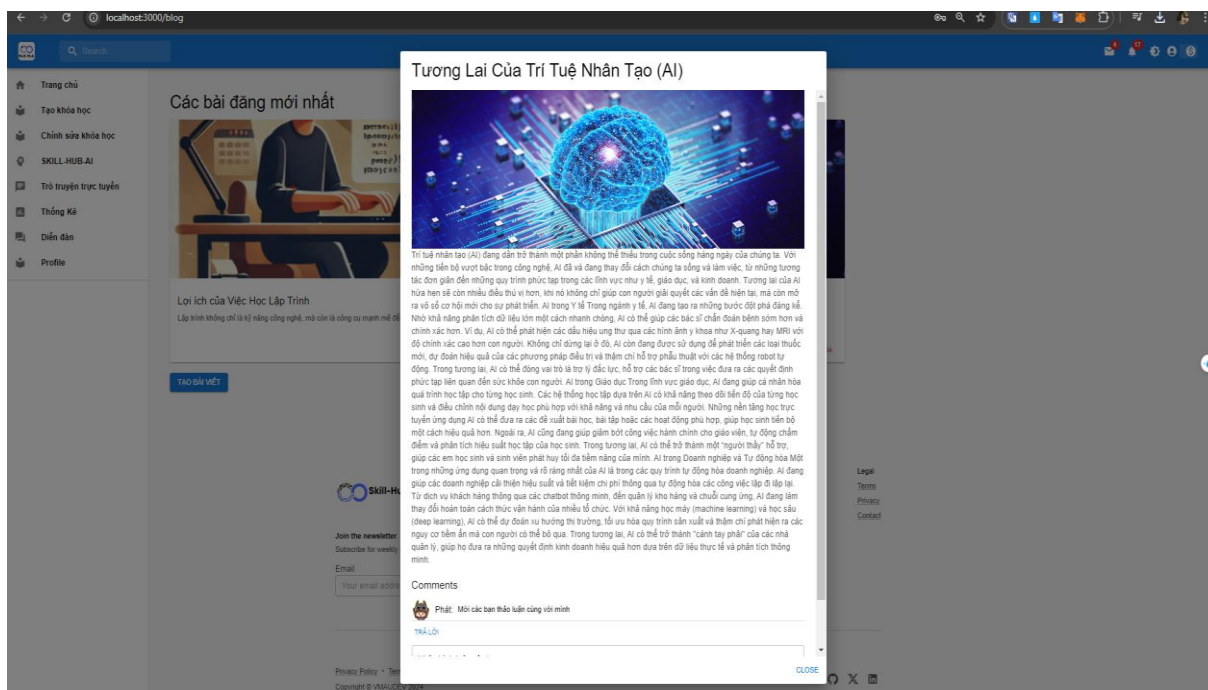
Hình 3.14: Giao diện trang thêm bài tập cho khóa học

3.4.10. Giao diện các bài giảng trong khóa học

The screenshot shows a web application interface for viewing a list of courses. On the left is a sidebar with navigation links: Trang chủ, Tạo khóa học, Chính sửa khóa học, SKILL-HUB.AI, Trò chuyện trực tuyến, Thống kê, Diễn đàn, and Profile. The main content area is titled 'DANH SÁCH BÀI GIẢNG' and displays a course titled 'Introduction to Spring Framework'. The course description states that it covers the basic concepts of the Spring Framework, including Inversion of Control (IoC), Dependency Injection, and the Spring Framework architecture. Below the description is a video player showing a video titled 'Learn Spring Framework #1-Introduction'. The video player has a progress bar and a play button. Below the video player is a section titled 'Bài tập:' (Exercises) which lists two questions. The first question is about the main components of the Spring Framework, and the second is about the parts of Spring responsible for managing bean dependencies. The questions are listed with their respective multiple-choice options.

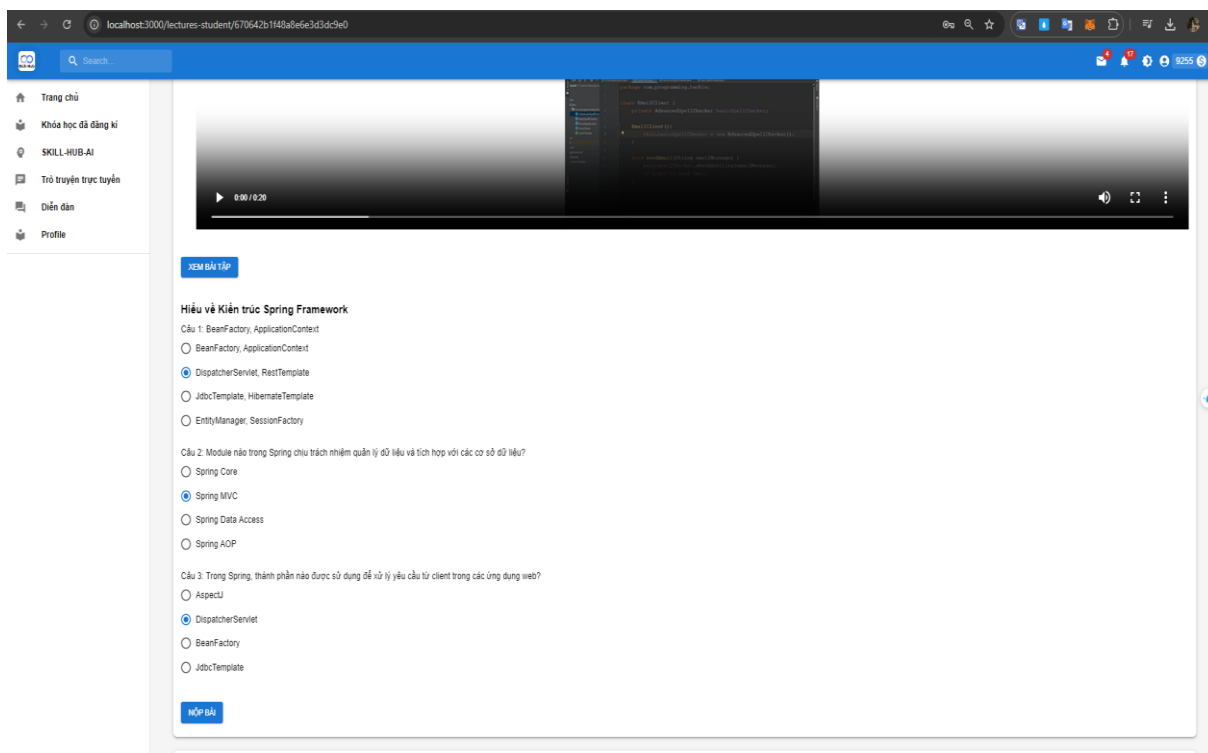
Hình 3.15: Giao diện trang bài giảng của khóa học

3.4.11. Giao diện diễn đàn



Hình 3.16: Giao diện trang diễn đàn

3.4.12. Giao diện làm bài tập của học viên



Hình 3.17: Giao diện trang làm bài tập của học viên

3.4.13. Giao diện nạp tiền qua VNPAY

The screenshot displays the VNPay payment interface. On the left, under 'Thông tin đơn hàng (Test)', the transaction amount is 1,000,000 VND, and the merchant is Công ty CTT HTT1. On the right, under 'Thanh toán qua Ngân hàng NCB', there is a form for card payment with fields for card number, cardholder name, and expiration date. A 'Tiếp tục' (Continue) button is at the bottom right.

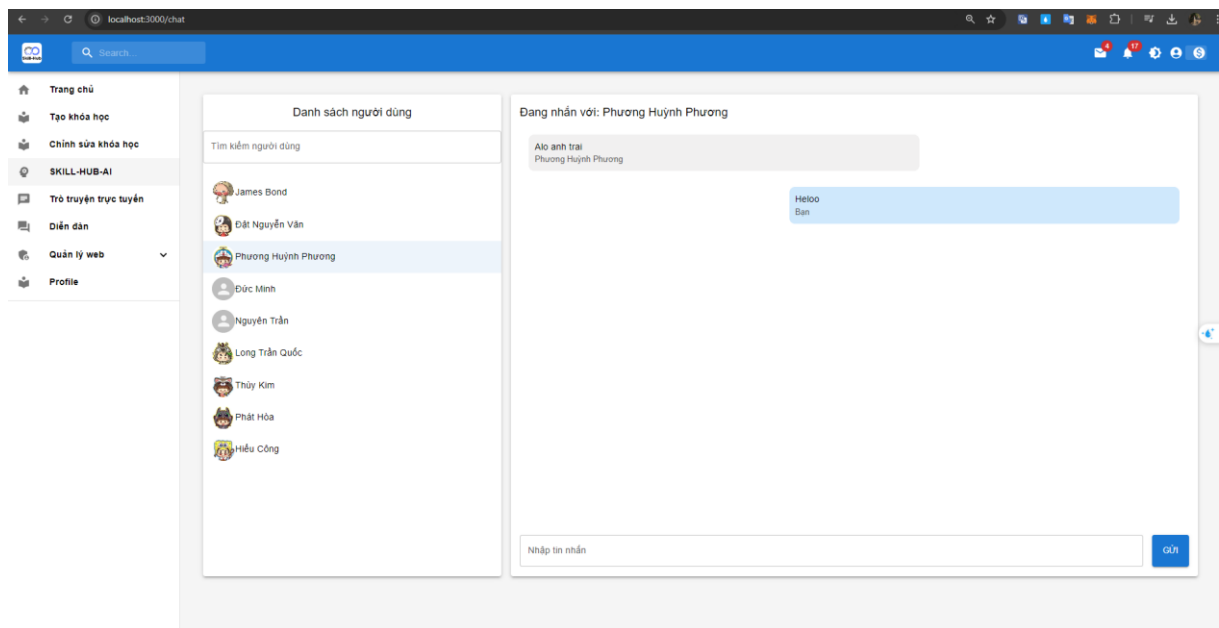
Hình 3.18: Giao diện trang nạp tiền qua Vnpay

3.4.14. Giao diện thống kê của người quản trị



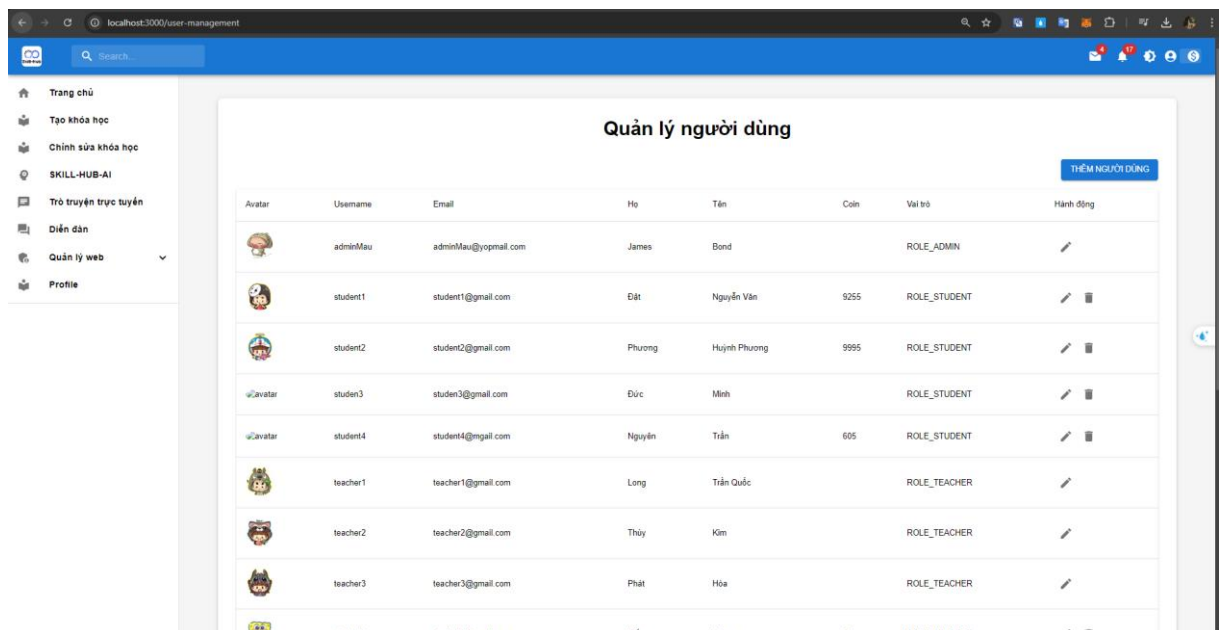
Hình 3.19: Giao diện trang thống kê cho người quản trị

3.4.15. Giao diện trò chuyện trực tuyến



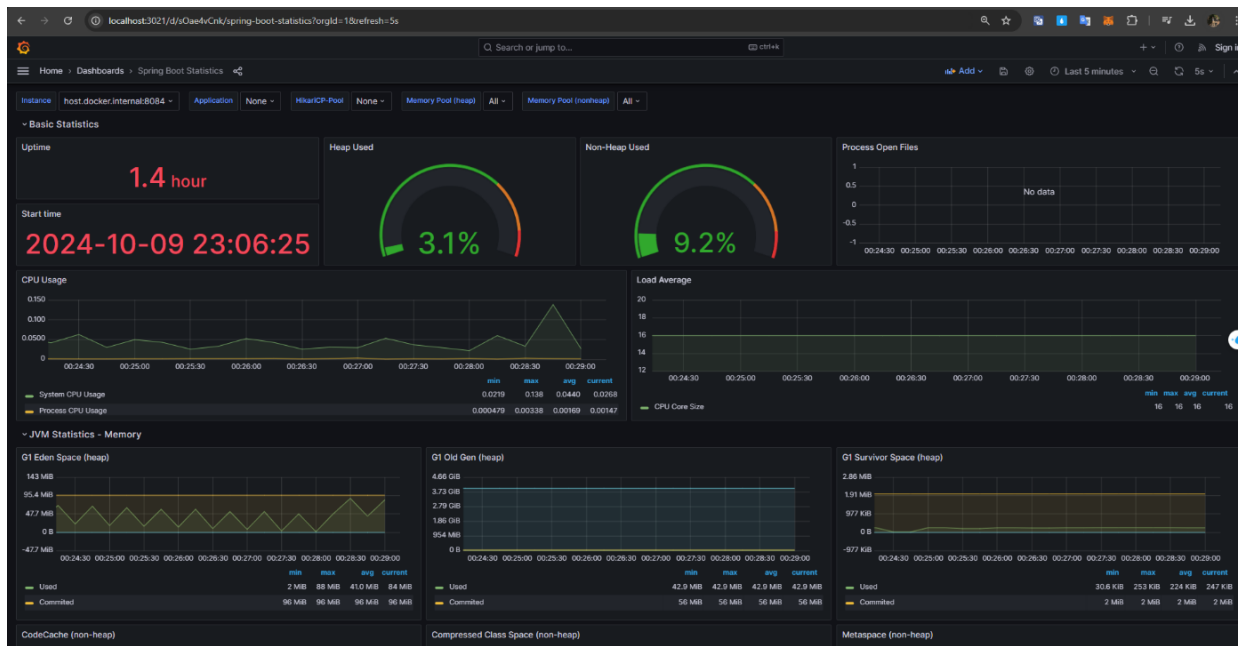
Hình 3.20: Giao diện trang trò chuyện trực tuyến cho người dùng

3.4.16. Giao diện quản lý người dùng của người quản trị



Hình 3.21: Giao diện trang quản lý người dùng

3.4.17. Giao diện thông kê giám sát hiệu suất của hệ thống



Hình 3.22: Giao diện trang giám sát hiệu suất hệ thống

Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1. Kết luận

Hệ thống được phát triển đã áp dụng thành công mô hình kiến trúc microservices vào thực tế, cho phép tách biệt các chức năng riêng lẻ của từng service, giúp tăng tính linh hoạt, khả năng mở rộng và dễ bảo trì hơn. Mỗi service trong hệ thống được phát triển và triển khai độc lập, giúp tối ưu hóa tài nguyên hệ thống và cải thiện hiệu suất xử lý các yêu cầu. Việc chia nhỏ hệ thống theo microservices còn giúp giảm thiểu sự phức tạp trong quản lý và phát triển hệ thống, đồng thời cho phép dễ dàng thêm mới hoặc thay đổi các dịch vụ mà không ảnh hưởng đến toàn bộ hệ thống.

Đặc biệt, việc tiếp cận các công nghệ tiên tiến như Docker, Kafka, và Keycloak đã góp phần lớn vào việc tối ưu hóa hệ thống. Docker giúp triển khai dịch vụ một cách nhanh chóng và nhất quán trên nhiều môi trường khác nhau. Kafka đóng vai trò quan trọng trong việc giao tiếp bất đồng bộ giữa các service, giúp tăng khả năng chịu lỗi và xử lý các tác vụ nặng một cách hiệu quả. Keycloak đảm bảo việc xác thực và phân quyền người dùng một cách an toàn và linh hoạt.

Việc ứng dụng trí tuệ nhân tạo vào hệ thống, như chatbot hỗ trợ người dùng, đã tạo ra trải nghiệm tương tác thông minh và tiện lợi, giúp nâng cao trải nghiệm của người dùng. Các công nghệ giám sát hệ thống như Prometheus, Grafana cũng đã được tích hợp nhằm theo dõi hiệu suất và tình trạng của các dịch vụ, giúp hệ thống năng cao độ ổn định và phản ứng kịp thời trước các vấn đề xảy ra.

Hơn nữa, khả năng phát triển và mở rộng của hệ thống được đảm bảo nhờ kiến trúc microservices, cho phép hệ thống dễ dàng tích hợp thêm các dịch vụ mới mà không làm gián đoạn dịch vụ hiện tại. Điều này không chỉ giúp tăng khả năng phát triển và mở rộng của hệ thống, mà còn đảm bảo hiệu suất hoạt động ổn định khi có sự biến đổi lớn về lượng truy cập trong thời gian ngắn.

Tổng kết lại, hệ thống đã triển khai và chứng minh được hiệu quả mà việc sử dụng kiến trúc microservices cùng với các công nghệ hiện đại trong việc xây dựng một nền tảng mạnh mẽ, linh hoạt và dễ dàng phát triển trong tương lai.

4.2. Hướng phát triển

Tiếp tục nghiên cứu và ứng dụng trí tuệ nhân tạo vào hệ thống để nâng cao trải nghiệm và hiệu quả học tập. Phân tích các tương tác giữa người dùng và trí tuệ nhân tạo nhằm tối ưu hóa chatbot, cải thiện khả năng tương tác và đáp ứng nhu cầu người học một cách hiệu quả hơn.

Tăng cường tính bảo mật cho hệ thống, bổ sung các phương thức đăng nhập qua mạng xã hội (như Google, Facebook) và quản lý chặt chẽ thông tin cá nhân của người dùng, đảm bảo an toàn và tuân thủ các quy định về bảo mật dữ liệu.

Tích hợp thêm nhiều phương thức thanh toán mới vào hệ thống, tạo sự tiện lợi cho người dùng khi thanh toán, bao gồm các ví điện tử, thẻ tín dụng quốc tế, và các hình thức thanh toán di động.

Nghiên cứu và phát triển các tính năng giám sát thời gian thực để theo dõi hành vi và thời gian hoạt động của người dùng. Điều này sẽ giúp cải thiện khả năng quản lý, nâng cao tính linh hoạt và hiệu suất của hệ thống trong việc đáp ứng các nhu cầu của người dùng.

TÀI LIỆU THAM KHẢO

- [1] Keycloak, " Open Source Identity and Access Management" 2024. [Online]. Available: <https://www.keycloak.org/>. [Accessed: 22-Sep-2024].
- [2] Geeksforgeeks, "Stream In Java" 2024. [Online]. Available: <https://www.geeksforgeeks.org/stream-in-java/>. [Accessed: 22-Oct-2024].
- [3] Confluent, "What is Apache Kafka?" 2024. [Online]. Available: <https://www.confluent.io/lp/apache-kafka>. [Accessed: 24-Oct-2023].
- [4] Docker, "What is Docker?," 2024. [Online]. Available: <https://docs.docker.com/get-started/docker-overview/>. [Accessed: 24-Oct-2024].
- [5] Spring, "Spring Boot Reference Documentation," 2024. [Online]. Available: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>. [Accessed: 24-Oct-2024].
- [6] AWS, "Amazon S3 - Cloud Object Storage" 2024. [Online]. Available: <https://aws.amazon.com/s3/>. [Accessed: 25-Oct-2024].
- [7] Prometheus, "Prometheus Monitoring System and Time Series Database" 2024. [Online]. Available: <https://prometheus.io/docs/introduction/overview/>. [Accessed: 26-Oct-2024].
- [8] Grafana, " Introducing exemplar support in Grafana Cloud, tightly coupling traces to your metrics" 2024. [Online]. Available: <https://grafana.com/blog/2022/02/23/introducing-exemplar-support-in-grafana-cloud-tightly-coupling-traces-to-your-metrics/>. [Accessed: 26-Oct-2024].

PHỤ LỤC

Dưới đây là cấu hình Docker Compose được sử dụng trong hệ thống để quản lý các dịch vụ microservices. Mỗi dịch vụ được triển khai độc lập với các cơ sở dữ liệu tương ứng, bao gồm các dịch vụ như MongoDB, MySQL, Kafka, Zookeeper, Keycloak và các dịch vụ giám sát hệ thống như Prometheus, Grafana.

Phiên bản Docker Compose:

```
1 version: '4'
```

Cấu hình các dịch vụ MongoDB:

```
15 mongodb_lecture:
16   image: mongo:7.0.5
17   container_name: mongodb_lecture
18   ports:
19     - "27019:27017"
20   environment:
21     MONGO_INITDB_ROOT_USERNAME: root
22     MONGO_INITDB_ROOT_PASSWORD: password
23     MONGO_INITDB_DATABASE: lecture-service
24   volumes:
25     - ./data/mongodb_lecture:/data/db
```

mongodb_lecture: Dịch vụ MongoDB cho lecture -service, sử dụng phiên bản MongoDB 7.0.5, với thông tin đăng nhập là root/password và cơ sở dữ liệu là lecture-service.

Tương tự, các dịch vụ MongoDB khác cho các microservices khác được cấu hình với các cổng và cơ sở dữ liệu tương ứng như mongodb_profile, mongodb_exam_results, mongodb_course,...

Cấu hình MySQL cho hệ thống:

```
96 mysql:
97   image: mysql:8.3.0
98   container_name: mysql
99   environment:
100     MYSQL_ROOT_PASSWORD: mysql
101   ports:
102     - "3306:3306"
103   volumes:
104     - ./mysql:/var/lib/mysql
105     - ./docker/mysql/init.sql:/docker-entrypoint-initdb.d/init.sql
```

mysql: Dịch vụ MySQL dùng để quản lý cơ sở dữ liệu quan hệ, sử dụng mật khẩu root và chạy trên cổng 3306.

Cấu hình Kafka và Zookeeper:

```

107  ▶  zookeeper:
108      image: confluentinc/cp-zookeeper:7.5.0
109      hostname: zookeeper
110      container_name: zookeeper
111  ▶  ports:
112      | - "2181:2181"
113  ▶  environment:
114      | ZOOKEEPER_CLIENT_PORT: 2181
115      | ZOOKEEPER_TICK_TIME: 2000
116

```

zookeeper: Dịch vụ Zookeeper quản lý thông tin và điều phối Kafka.

```

117  ▶  broker:
118      image: confluentinc/cp-kafka:7.5.0
119      container_name: broker
120  ▶  ports:
121      | - "9092:9092"
122      | - "29092:29092"
123  ▶  depends_on:
124      | - zookeeper
125  ▶  environment:
126      | KAFKA_BROKER_ID: 1
127      | KAFKA_ZOOKEEPER_CONNECT: 'zookeeper:2181'
128      | KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
129      | KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://broker:29092,PLAINTEXT_HOST://localhost:9092
130      | KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1

```

broker: Dịch vụ Kafka, giúp quản lý giao tiếp bất đồng bộ giữa các dịch vụ khác nhau.

```

132  ▶  schema-registry:
133      image: confluentinc/cp-schema-registry:7.5.0
134      hostname: schema-registry
135      container_name: schema-registry
136      depends_on:
137      | - broker
138      ports:
139      | - "8087:8081"
140      environment:
141      | SCHEMA_REGISTRY_HOST_NAME: schema-registry
142      | SCHEMA_REGISTRY_KAFKASTORE_BOOTSTRAP_SERVERS: 'broker:29092'
143      | SCHEMA_REGISTRY_LISTENERS: http://schema-registry:8081

```

kafka-ui: Đây là tên của container quản lý giao diện người dùng cho Kafka.

Keycloak và Keycloak-MySQL:

```

157 ► keycloak-mysql:
158     container_name: keycloak-mysql
159     image: mysql:8
160     restart: always
161     environment:
162         MYSQL_ROOT_PASSWORD: root
163         MYSQL_DATABASE: keycloak
164         MYSQL_USER: keycloak
165         MYSQL_PASSWORD: password
166     volumes:
167         - mysql_data:/var/lib/mysql
168     networks:
169         - keycloak-network

```

keycloak-mysql: Dịch vụ MySQL dùng cho Keycloak, giúp quản lý thông tin người dùng và quyền truy cập.

```

171 ► keycloak:
172     container_name: keycloak
173     image: quay.io/keycloak/keycloak:25.0.0
174     command: [ "start-dev", "--import-realm" ]
175     environment:
176         DB_VENDOR: MYSQL
177         DB_ADDR: keycloak-mysql
178         DB_DATABASE: keycloak
179         DB_USER: keycloak
180         DB_PASSWORD: password
181         KEYCLOAK_ADMIN: admin
182         KEYCLOAK_ADMIN_PASSWORD: admin
183     ports:
184         - "8180:8080"
185     volumes:
186         - keycloak_data:/opt/keycloak/data
187     depends_on:
188         - keycloak-mysql
189     networks:
190         - keycloak-network

```

keycloak: Dịch vụ quản lý danh tính và xác thực, sử dụng cơ sở dữ liệu MySQL.

Dịch vụ Giám sát Hệ thống:

```

196 ► prometheus:
197     image: prom/prometheus:v2.46.0
198     command:
199         - --enable-feature=exemplar-storage
200         - --config.file=/etc/prometheus/prometheus.yml
201     volumes:
202         - ./docker/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml:ro
203     ports:
204         - '9090:9090'

```

Prometheus: Dịch vụ giám sát hiệu suất của hệ thống.

```

191 ► loki:
192     image: grafana/loki:main
193     command: [ '-config.file=/etc/loki/local-config.yaml' ]
194     ports:
195     - '3100:3100'

```

Loki: Dịch vụ dùng để thu thập log của hệ thống.

```

205 ► tempo:
206     image: grafana/tempo:2.2.2
207     command: [ '-config.file=/etc/tempo.yaml' ]
208     volumes:
209     - ./docker/tempo/tempo.yml:/etc/tempo.yaml:ro
210     - ./docker/tempo/tempo-data:/tmp/tempo
211     ports:
212     - '3110:3100' # Tempo
213     - '9411:9411' # zipkin

```

Tempo: Dịch vụ dùng để giám sát và truy vết phân tán.

```

214 ► grafana:
215     image: grafana/grafana:10.1.0
216     volumes:
217     - ./docker/grafana:/etc/grafana/provisioning/datasources:ro
218     environment:
219     - GF_AUTH_ANONYMOUS_ENABLED=true
220     - GF_AUTH_ANONYMOUS_ORG_ROLE=Admin
221     - GF_AUTH_DISABLE_LOGIN_FORM=true
222     ports:
223     - '3021:3000'

```

Grafana: Dịch vụ cung cấp giao diện đồ họa để giám sát hệ thống qua Prometheus, Loki và Tempo.

Volumes và Networks:

```

224 volumes:
225     mysql_data:
226     | driver: local
227     keycloak_data:
228     | driver: local
229
230 networks:
231     keycloak-network:
232     | driver: bridge

```

Volumes: Định nghĩa các vùng lưu trữ dữ liệu cho các dịch vụ MySQL và Keycloak.

Networks: Mạng riêng biệt để kết nối các dịch vụ như Keycloak và MySQL.