KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

**THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH**

**HỌC KỲ I, NĂM HỌC 2023 - 2024**

**NGHIÊN CỨU RESTFUL API VỚI JAVA SPRINGBOOT ĐỂ XÂY DỰNG MODULE BACKEND CHO ỨNG DỤNG QUẢN LÝ QUÁN CAFE**

*Giáo viên hướng dẫn:*

TS. Nguyễn Bảo Ân

*Sinh viên thực hiện:*

Họ và tên - MSSV:

Lâm Ngọc Tài – 110120152

Lớp: DA20TTB

*Trà Vinh, tháng 12 năm 2023*

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

**A blue and white logo

Description automatically generated**

**THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH**

**HỌC KỲ I, NĂM HỌC 2023 - 2024**

**NGHIÊN CỨU RESTFUL API VỚI JAVA SPRINGBOOT ĐỂ XÂY DỰNG MODULE BACKEND CHO ỨNG DỤNG QUẢN LÝ QUÁN CAFE**

*Giáo viên hướng dẫn:*

TS. Nguyễn Bảo Ân

*Sinh viên thực hiện:*

Họ và tên - MSSV:

Lâm Ngọc Tài – 110120152

Lớp: DA20TTB

*Trà Vinh, tháng 12 năm 2023*

# NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

*Trà Vinh, ngày tháng năm 2023*

**Giáo viên hướng dẫn**

*(Ký tên và ghi rõ họ tên)*

# LỜI CẢM ƠN

# NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

*Trà Vinh, ngày tháng năm 2023*

**Thành viên hội đồng**

*(Ký tên và ghi rõ họ tên)*

Kính thưa quý thầy cô

Đầu tiên, em xin bày tỏ lòng tri ân sâu sắc đến các thầy cô giáo của Trường Đại học Trà Vinh, đặc biệt là quý thầy cô thuộc Khoa Kỹ thuật & Công nghệ, bộ môn Công nghệ Thông tin, đã tận tình hỗ trợ và tạo mọi điều kiện thuận lợi để con có thể hoàn thành bài đồ án chuyên ngành này một cách tốt nhất.

Em cũng xin gửi lời biết ơn chân thành đến thầy Nguyễn Bảo Ân - Giảng viên Khoa Kỹ thuật & Công nghệ, người đã hết lòng hướng dẫn, truyền đạt kiến thức và kinh nghiệm quý báu, giúp con hoàn thiện đồ án này.

Em nhận thức được rằng bản thân còn nhiều hạn chế và đồ án vẫn còn những sai sót nhất định do kinh nghiệm còn hạn chế. Em kính mong nhận được sự thông cảm và những lời góp ý quý báu từ quý thầy cô để em có thể rút kinh nghiệm và tiếp tục hoàn thiện mình trong những nghiên cứu sau này.

Cuối cùng, em xin kính chúc quý thầy cô sức khỏe dồi dào, hạnh phúc và thành công trong cuộc sống.

Với tất cả sự kính trọng và biết ơn

Ký tên

MỤC LỤC

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN 1](file:///D:\DACN\Mẫu%20-%20Quy%20định%20trình%20bày%20-%20thực%20hiện%20CN-CSN\110120152_LAMNGOCTAI_DOANCHUYENNGANH.docx#_Toc155552389)

[LỜI CẢM ƠN 2](#_Toc155552390)

[NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG 2](file:///D:\DACN\Mẫu%20-%20Quy%20định%20trình%20bày%20-%20thực%20hiện%20CN-CSN\110120152_LAMNGOCTAI_DOANCHUYENNGANH.docx#_Toc155552391)

[DANH MỤC VIẾT TẮC 6](#_Toc155552392)

[DANH MỤC HÌNH ẢNH 7](#_Toc155552393)

[DANH MỤC BẢNG 8](#_Toc155552394)

[TÓM TẮT ĐỒ ÁN 9](#_Toc155552395)

[Lí do chọn đề tài: 10](#_Toc155552396)

[Mục tiêu nghiêm cứu: 10](#_Toc155552397)

[Đối tượng nghiêm cứu: 10](#_Toc155552398)

[Phạm vị nghiêm cứu: 10](#_Toc155552399)

[CHƯƠNG 1: TỔNG QUAN 11](#_Toc155552400)

[1.1 Mô tả 11](#_Toc155552401)

[1.2 Phạm vi đề tài 11](#_Toc155552402)

[1.3 Hướng giải quyết 12](#_Toc155552403)

[CHƯƠNG 2: NGHIÊN CỨU VỀ LÝ THUYẾT 14](#_Toc155552404)

[2.1 Tìm hiểu về Java Spring Boot 14](#_Toc155552405)

[2.2 Json Web Token (JWT) 17](#_Toc155552406)

[2.3 Springdoc-openapi 22](#_Toc155552407)

[2.4 Tổng quan về MySQL 23](#_Toc155552408)

[CHƯƠNG 3: THỰC HÓA NGHIÊM CỨU 24](#_Toc155552409)

[3.1 Lược đồ use-case 24](#_Toc155552410)

[3.2 Mô hình thực thể kết hợp 25](#_Toc155552411)

[3.4 Mô hình mức dữ liệu logic 26](#_Toc155552412)

[3.5 Kiến trúc hệ thống 28](#_Toc155552413)

[3.6 Thiết kế API 29](#_Toc155552414)

[CHƯƠNG 4: KẾT QUẢ NGHIÊM CỨU 31](#_Toc155552415)

[4.1 Khởi tạo dự án 31](#_Toc155552416)

[4.2 Cấu hình dự án 32](#_Toc155552417)

[4.2 Bảo mật chương trình 33](#_Toc155552418)

[4.3 Xây dựng các API 40](#_Toc155552419)

[4.5 Triển khai 45](#_Toc155552420)

[CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 47](#_Toc155552421)

[5.1 Kết quả dạt được 47](#_Toc155552422)

[5.2 Hạn chế 47](#_Toc155552423)

[5.3 Hướng phát triển 47](#_Toc155552424)

[DANH MỤC TÀI LIỆU THAM KHẢO 49](#_Toc155552425)

[PHỤ LỤC 50](#_Toc155552426)

# DANH MỤC VIẾT TẮC

|  |  |
| --- | --- |
| Từ viết tắc | Nghĩa của từ |
| API | Application Programming Interface |
| RESTful | Representational State Transfer - ful |
| SQL | Structured Query Language |
| HTTP | HyperText Transfer Protocol |
| XML | eXtensible Markup Language |
| JWT | JSON Web Token |
| JSON | JavaScript Object Notation |

# DANH MỤC HÌNH ẢNH

[Hình 1. Mô tả quá trình tạo nên Spring Boot từ Spring Framework. 14](#_Toc155552427)

[Hình 2. Cơ chế Filter của Spring Security. 15](#_Toc155552428)

[Hình 3. Xác thực người dùng bằng json web token. 17](#_Toc155552429)

[Hình 4. Kiến trúc của JWT 18](#_Toc155552430)

[Hình 5.Page load của JWT 20](#_Toc155552431)

[Hình 6. Mô tả chung về springdoc-openai. 22](#_Toc155552432)

[Hình 7. Logo MySQL 23](#_Toc155552433)

[Hình 8. Sơ đồ use-case. 24](#_Toc155552434)

[Hình 9 Mô hình thực thể kết hợp 25](#_Toc155552435)

[Hình 10 Mô hình mức dữ liệu logic 26](#_Toc155552436)

[Hình 11. Kiến trúc dự án. 28](#_Toc155552437)

[Hình 12. Khởi tạo dự án Java Spring Boot. 31](#_Toc155552438)

[Hình 13. Một số dependencies của dự án. 31](#_Toc155552439)

[Hình 14. Cấu hình dự án 32](#_Toc155552440)

[Hình 15. Xây dựng security filters. 33](#_Toc155552441)

[Hình 16. WHITE\_LIST\_URL. 33](#_Toc155552442)

[Hình 17. Xây dựng JwtAuthenticationFilter 35](#_Toc155552443)

[Hình 18. Đoạn code cấu hình liên quan đến xác thực người dùng. 36](#_Toc155552444)

[Hình 19. Mô tả quá trình xác thực người dùng. 37](#_Toc155552445)

[Hình 20. Phương thức buildToken. 38](#_Toc155552446)

[Hình 21. Đoạn code kiểm tra JWT tokens. 39](#_Toc155552447)

[Hình 22. Lớp Entity. 40](#_Toc155552448)

[Hình 23. Lớp DAO. 41](#_Toc155552449)

[Hình 24. ProductRepository. 42](#_Toc155552450)

[Hình 25. Lớp service. 42](#_Toc155552451)

[Hình 26. EmployeeService. 43](#_Toc155552452)

[Hình 27. Đoạn code EmployeeController. 44](#_Toc155552453)

[Hình 28. Đoạn log hiện thị khi chạy dự án. 45](#_Toc155552454)

[Hình 29. Giao diện swagger-ui. 46](#_Toc155552455)

# DANH MỤC BẢNG

[Bảng 1. Bảng “productcategory” 26](#_Toc155514445)

[Bảng 2. Bảng “product” 26](#_Toc155514446)

[Bảng 3. Bảng “OrderItem” 27](#_Toc155514447)

[Bảng 4. Bảng “Invoice” 27](#_Toc155514448)

[Bảng 5. Bảng “productcategory” 27](#_Toc155514449)

[Bảng 6. Bảng “token” 28](#_Toc155514450)

# TÓM TẮT ĐỀ TÀI

**Vấn đề nghiên cứu**

Tích hợp dữ liệu không nhất quán: thông tin từ các nguồn khác nhau như nhà cung cấp tour, điểm du lịch, và khách sạn thường không nhất quán về định dạng và cấu trúc, làm tăng độ phức tạp trong quá trình tích hợp dữ liệu.

Cập nhật và bảo dưỡng nội dung: duy trì thông tin chính xác và cập nhật trên website, như giá tour, thông tin điểm đến, và đánh giá, cần nhiều công sức và nguồn lực.

Thích Nghi với Xu Hướng Thị Trường: Theo kịp với xu hướng thay đổi nhanh chóng của ngành du lịch, như du lịch bền vững, trải nghiệm cá nhân hóa, và công nghệ mới.

**Cách giải quyết vấn đề**

Một cách giải quyết vấn đề nghiên cứu là phát triển một website dựa trên ASP.NET web form, sử dụng MySQL để xử lý dữ liệu. Website này có thể giúp giải quyết các thách thức sau:

- Tích hợp dữ liệu không nhất quán: Sử dụng MySQL, một hệ quản trị cơ sở dữ liệu quan hệ, để lưu trữ và truy vấn dữ liệu từ các nguồn khác nhau. MySQL có thể hỗ trợ nhiều định dạng và cấu trúc dữ liệu, và cho phép thực hiện các thao tác như nối, lọc, và sắp xếp dữ liệu một cách linh hoạt và hiệu quả.

- Cập nhật và bảo dưỡng nội dung: Sử dụng ASP.NET web form, một công nghệ để xây dựng giao diện người dùng động cho website. ASP.NET web form cho phép tạo ra các trang web có thể tương tác với người dùng và cơ sở dữ liệu, và cung cấp các thành phần điều khiển để hiển thị và cập nhật nội dung một cách dễ dàng và nhanh chóng.

- Thích nghi với xu hướng thị trường: Sử dụng ASP.NET web form, một công nghệ có khả năng mở rộng và tùy biến cao. ASP.NET web form cho phép tạo ra các trang web có thể thay đổi theo yêu cầu của người dùng và xu hướng của ngành du lịch, bằng cách sử dụng các công cụ như master page, theme, skin, và user control.

**Một số kết quả đạt được:**

**MỞ ĐẦU**

## Lí do chọn đề tài:

Đầu tiên, chúng ta gặp phải vấn đề về tích hợp dữ liệu không nhất quán. Điều này xuất hiện khi thông tin từ nhiều nguồn khác nhau, như các nhà cung cấp tour, điểm du lịch, và khách sạn, không có sự đồng nhất về định dạng hoặc cấu trúc. Tình trạng này tạo ra rất nhiều khó khăn trong việc hợp nhất và quản lý dữ liệu một cách hiệu quả.

Tiếp theo là vấn đề cập nhật và bảo dưỡng nội dung trên website. Để đảm bảo thông tin luôn chính xác và cập nhật, như giá cả các tour du lịch, thông tin về các điểm đến, và đánh giá từ khách hàng, cần phải bỏ ra nhiều nỗ lực và nguồn lực. Việc này yêu cầu một hệ thống quản lý nội dung linh hoạt và một đội ngũ làm việc cẩn thận và chuyên nghiệp.

Cuối cùng, đối mặt với sự thay đổi không ngừng của thị trường du lịch, việc thích nghi với những xu hướng mới như du lịch bền vững, trải nghiệm cá nhân hóa cho khách hàng, và việc áp dụng các công nghệ mới trở thành yếu tố then chốt. Điều này yêu cầu doanh nghiệp phải không ngừng cập nhật và nâng cấp hệ thống của mình để không chỉ đáp ứng nhu cầu của thị trường mà còn tạo ra sự khác biệt trên thị trường cạnh tranh.

## Mục tiêu nghiêm cứu:

Để giải quyết các vấn đề trên, đề tài này đề xuất một hệ thống quản lý website du lịch thông minh, sử dụng các kỹ thuật khai thác dữ liệu, học máy, và trí tuệ nhân tạo để tự động hóa các quy trình liên quan đến tích hợp, cập nhật, và thích ứng nội dung. Hệ thống này sẽ giúp các doanh nghiệp du lịch tiết kiệm chi phí, thời gian, và nguồn lực, cũng như nâng cao chất lượng và hiệu quả của website du lịch

## Đối tượng nghiêm cứu:

ASP.NET Web forms: Các nguyên lý và xây dựng trang web du lịch theo cấu trúc client – server.

Trang web: Nghiên cứu về các yêu cầu và quy trình nghiệp vụ của website du lịch.

## Phạm vị nghiêm cứu:

Sử dụng môi trường phát triển dựa trên C# và các công cụ như Visual Studio, SQL server Manament Studio.

Các nghiên cứu và phát triển sẽ được thực hiện trong khuôn khổ thời gian và nguồn lực đã xác định trước.

# CHƯƠNG 1: TỔNG QUAN

## 1.1 Mô tả

## 1.2 Phạm vi đề tài

Đề tài sẽ tập trung trình bày kết quả nghiên cứu của tôi về các nội dung sau: Spring framework, MySQL, Maven. Mỗi phần tôi sẽ giới thiệu sơ lược và trình bày những nội dung cơ bản nhất, những điểm mạnh hay lợi ích mà nó mang lại cho các nhà phát triển phần mềm.

Cụ thể về Spring framework sẽ tập trung tìm hiểu và trình bày 2 module: Spring boot và Spring Security.

Sau khi tìm hiểu tôi sẽ vận dụng kết quả tìm hiểu được vào việc xây dựng một module được kiểm thử Postman.

## 1.3 Hướng giải quyết

* Khám Phá và Phân Tích:

Đầu tiên, tiến hành nghiên cứu về Spring framework để hiểu rõ kiến trúc cũng như cách thức hoạt động của nó. Điều này bao gồm việc tìm hiểu các khái niệm cơ bản, cấu hình, và quy trình vận hành.

Phân tích và đánh giá MySQL, một hệ quản trị cơ sở dữ liệu quan hệ, để hiểu cách nó có thể được tích hợp và sử dụng hiệu quả trong ứng dụng.

Nghiên cứu Maven như một công cụ quản lý và tự động hóa dự án, đặc biệt là cách quản lý phụ thuộc và xây dựng dự án.

* Tìm hiểu Spring Boot và Spring Security:

Tập trung nghiên cứu kỹ lưỡng về Spring Boot, một module của Spring framework, để tạo ra các ứng dụng độc lập với cấu hình mặc định tối ưu.

Nguyên cứu Spring Security, để bảo đảm tính năng bảo mật cho ứng dụng thông qua xác thực và ủy quyền người dùng.

* Ứng Dụng Thực Tiễn:

Áp dụng kiến thức thu được để thiết kế và xây dựng một module backend cho ứng dụng Quản lý Quán Cafe, sử dụng Spring Boot để cung cấp cơ sở vững chắc và Spring Security để đảm bảo tính bảo mật.

Tích hợp MySQL như là hệ quản trị cơ sở dữ liệu để quản lý dữ liệu liên quan đến đơn hàng, tồn kho, và thông tin người dùng.

* Kiểm Thử và Đánh Giá:

Tiến hành kiểm thử chức năng của module sử dụng Postman, một công cụ kiểm thử API, để đánh giá chức năng và tìm ra các vấn đề tiềm ẩn.

Tinh chỉnh và cải tiến module dựa trên kết quả kiểm thử để đảm bảo ứng dụng hoạt động hiệu quả và ổn định.

Các công cụ hỗ trợ nghiên cứu:

1. Microsoft Word: dùng để viết tài liệu báo cáo

2. Draw.io: dùng để vẽ các biểu đồ

3. IntelliJ IDEA: dùng để code back end

4. Postman: kiểm tra các API đã viết ra.

5. MySQL Workbench: dùng để lưu trữ cơ sở dữ liệu

Sau quá trình nghiêm cứu và xây dựng đã xây dựng thành công bộ API đủ phục vụ cho người lập trình viên xây dựng ứng dụng trên các nền tản khác nhau.

# CHƯƠNG 2: **NGHIÊN CỨU VỀ LÝ THUYẾT**

* ASP. Net
* Sql server
* C#
* Web forms

## 2.1 Tìm hiểu về Java Spring Boot

### 2.1.1 Java Spring Boot là gì?

Java Spring Boot là một dự án phụ của Spring Framework, được thiết kế để đơn giản hóa quá trình cài đặt và phát triển các ứng dụng dựa trên Spring. Đây là một framework mạnh mẽ, linh hoạt, được sử dụng rộng rãi trong việc phát triển ứng dụng web và microservices trong môi trường Java [1].

A close-up of a computer

Description automatically generated

Hình 1. Mô tả quá trình tạo nên Spring Boot từ Spring Framework.

Một số điểm nổi bậc của Java Spring Boot:

- Spring Boot tự động cấu hình ứng dụng của dựa trên các thư viện có trong classpath. Điều này giúp giảm thiểu công việc cấu hình và bootstrap ứng dụng.

- Nó tạo ra các ứng dụng có thể chạy độc lập, với embedded Tomcat, Jetty, hoặc các server web khác, không cần một servlet container riêng biệt.

- Các ứng dụng truyền thống của Spring Framework thường yêu cầu cấu hình XML phức tạp hoặc sử dụng annotation @Configuration để định nghĩa beans và các phụ thuộc của chúng. Spring Boot giảm bớt nhu cầu này bằng cách cung cấp tự động cấu hình và các cài đặt mặc định hợp lý [2].

Spring Boot được tạo nên bằng cách kết hợp Spring Framework với các server HTTP tích hợp và giảm bớt đi sự phức tạp của cấu hình XML hoặc annotation. Điều này dẫn đến một nền tảng dễ dàng thiết lập và sẵn sàng chạy trong môi trường riêng của nó, lý tưởng cho các microservices và ứng dụng web [1].

### 2.1.2 Tổng quan Spring Security

#### Spring Security là gì?

Spring Security là một framework bảo mật chuyên sâu cho các ứng dụng dựa trên Spring Framework. Nó nhằm mục đích cung cấp cả xác thực và phân quyền cũng như các cơ chế bảo vệ chống lại các tấn công thông dụng trong ứng dụng web. Spring Security đặc biệt chú trọng vào việc cung cấp một giải pháp bảo mật dễ dàng tích hợp nhưng cũng đủ mạnh mẽ để đáp ứng được các yêu cầu bảo mật phức tạp [3] [4].

#### Cơ chế hoạt động Spring Security

Spring Security hoạt động theo mô hình client-server. Khi một client gửi một request đến server, server sẽ xác thực người dùng và phân quyền để đảm bảo rằng người dùng chỉ có thể truy cập vào những tài nguyên mà họ được phép truy cập [5].

A diagram of a software company

Description automatically generated

Hình 2. Cơ chế Filter của Spring Security.

Spring Security là một framework bảo mật mạnh mẽ dành cho các ứng dụng Java, đặc biệt là những ứng dụng được xây dựng trên nền tảng Spring. Nó cung cấp một cơ chế bảo mật toàn diện thông qua một chuỗi các bộ lọc (filters) được xác định để kiểm soát và quản lý quyền truy cập vào ứng dụng. Dưới đây là quy trình hoạt động chi tiết của Spring Security khi xử lý xác thực thông tin người:

* Spring Security Filters

Khi một yêu cầu HTTP được gửi từ client đến server, chuỗi bộ lọc của Spring Security bắt đầu xử lý yêu cầu đó. Trong số các bộ lọc này, UsernamePasswordAuthenticationFilter đóng vai trò quan trọng trong việc lấy thông tin từ yêu cầu đăng nhập - thường là tên người dùng và mật khẩu - và tạo ra một đối tượng Authentication, thường là UsernamePasswordAuthenticationToken. Bộ lọc này kế thừa từ AbstractAuthenticationProcessingFilter, một lớp trừu tượng cung cấp cơ sở cho việc xử lý xác thực.

* AuthenticationManager và AuthenticationProvider

UsernamePasswordAuthenticationToken được tạo ra sau đó được chuyển đến AuthenticationManager, một giao diện trung tâm trong Spring Security quản lý quá trình xác thực. AuthenticationManager có một phương thức authenticate() được gọi để xử lý đối tượng Authentication. Trong thực tiễn, ProviderManager là một implementation của AuthenticationManager, quản lý một danh sách các AuthenticationProvider.

Mỗi AuthenticationProvider có một phương thức authenticate() riêng biệt và chịu trách nhiệm xác thực thông tin người dùng. Nếu một AuthenticationProvider không thể xác thực thông tin người dùng, quá trình xác thực sẽ tiếp tục với AuthenticationProvider tiếp theo trong danh sách, nếu có [3].

* UserDetailsService và PasswordEncoder

DaoAuthenticationProvider, một trong những AuthenticationProvider phổ biến, sử dụng UserDetailsService để tải thông tin chi tiết của người dùng dựa trên tên người dùng. UserDetailsService có một phương thức quan trọng là loadUserByUsername() để tìm kiếm và trả về một đối tượng UserDetails chứa thông tin như tên người dùng, mật khẩu và các quyền hạn (authorities).

PasswordEncoder là một thành phần quan trọng khác, nó chịu trách nhiệm mã hóa mật khẩu khi lưu trữ và so sánh mật khẩu đã mã hóa với mật khẩu được cung cấp bởi người dùng khi đăng nhập. Việc mã hóa này giúp bảo vệ thông tin mật khẩu người dùng ngay cả khi có sự cố dữ liệu bị rò rỉ.

* Security Context

Sau khi xác thực thành công, thông tin người dùng và các quyền hạn được lưu trong SecurityContextHolder, trong một đối tượng gọi là SecurityContext. Điều này cho phép thông tin người dùng dễ dàng được truy cập trong suốt quá trình xử lý yêu cầu mà không cần phải xác thực lại. SecurityContext tồn tại trong phạm vi của một yêu cầu và thường được liên kết với một phiên người dùng [5].

Spring Security xử lý xác thực người dùng thông qua một quy trình nhiều lớp, từ việc lấy thông tin người dùng từ yêu cầu đến xác định và xác thực thông tin đó, rồi cuối cùng lưu trữ trong môi trường an toàn để sử dụng sau này. Quy trình này không chỉ đảm bảo rằng chỉ những người dùng hợp lệ mới có thể truy cập vào ứng dụng, mà còn bảo vệ thông tin người dùng và ứng dụng khỏi các mối đe dọa bảo mật.

## 2.2 Json Web Token (JWT)

### 2.2.1 Tìm hiểu về JWT

JSON Web Token (JWT) là một tiêu chuẩn mở (RFC 7519) được sử dụng để truyền thông tin an toàn giữa hai bên dưới dạng một đối tượng JSON. Thông tin này có thể được xác thực và đáng tin cậy vì nó được ký số điện tử. JWT thường được sử dụng trong ứng dụng web để quản lý xác thực người dùng và trao đổi thông tin [6].

A diagram of security

Description automatically generated

Hình 3. Xác thực người dùng bằng json web token.

Dưới đây là giải thích từng bước của quy trình xác thực:

JWTAuthFilter: Đây là bộ lọc xác thực JWT, chịu trách nhiệm kiểm tra token JWT trong mỗi yêu cầu HTTP đến.

Validate JWT: JWTAuthFilter gọi JwtService để kiểm tra sự hợp lệ của token JWT.

UserDetailsService: Nếu token hợp lệ, JwtService sẽ liên lạc với UserDetailsService để lấy thông tin chi tiết của người dùng.

Security: Sau khi thông tin người dùng được xác nhận, hệ thống bảo mật của Spring (Security) sẽ cập nhật SecurityContextHolder với thông tin xác thực của người dùng.

DispatcherServlet: Khi thông tin xác thực được cập nhật, yêu cầu sẽ được chuyển đến DispatcherServlet, đây là một thành phần trung tâm trong Spring MVC điều hướng yêu cầu đến các Controller phù hợp.

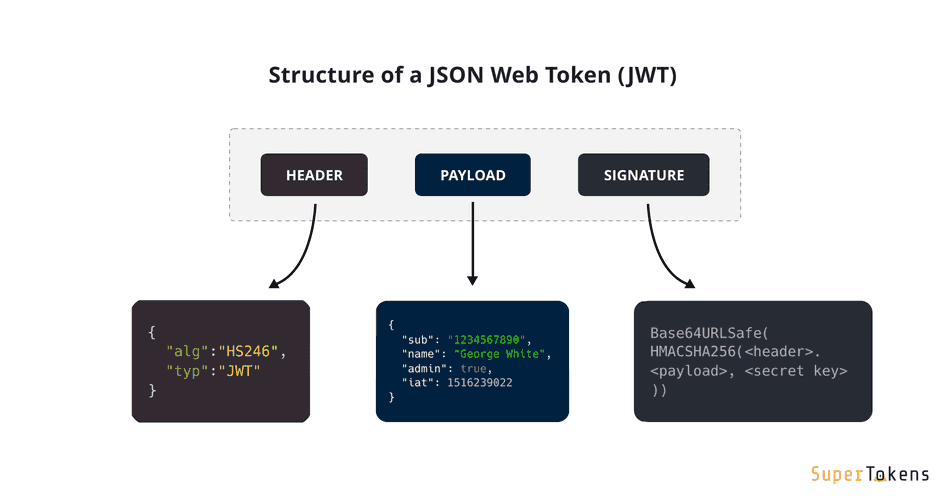
Controller: Controller sau đó xử lý yêu cầu và trả về phản hồi, thường là dạng JSON.

HTTP Response: Phản hồi sẽ được gửi trở lại cho client. Nếu xác thực thành công, client sẽ nhận được mã trạng thái HTTP 200. Nếu token không hợp lệ hoặc thiếu, client sẽ nhận được mã lỗi 403. Nếu thông tin người dùng không tồn tại, hệ thống cũng sẽ trả về mã lỗi 403.

Apache Tomcat: Là máy chủ web và servlet container mà Spring Boot sử dụng, được triển khai dưới dạng một container trong Docker, một nền tảng để triển khai ứng dụng dưới dạng containers, giúp việc triển khai và quản lý ứng dụng trở nên linh hoạt và dễ dàng hơn.

JWT gồm ba phần, tách biệt bởi dấu chấm (.):

* + Header (Tiêu đề): Header thường chứa loại token (JWT) và thuật toán mã hóa được sử dụng (như HMAC SHA256 hoặc RSA).
  + Payload (Dữ liệu): Payload chứa các tuyên bố (claims), như thông tin người dùng, thời gian hết hạn token, phát hành, v.v.
  + Signature (Chữ ký): Để tạo chữ ký, thuật toán mã hóa trong header được sử dụng cùng với base64-url encoded header và payload cùng với một 'secret' chỉ biết bởi người phát hành token [2].



Hình 4. Kiến trúc của JWT

### 2.2.2 Cơ chế ủy quyền

Ủy quyền JWT (JSON Web Token) là một quy trình quan trọng trong các hệ thống ứng dụng web và di động, cho phép kiểm soát quyền truy cập vào các tài nguyên và dịch vụ dựa trên các thông tin (claims) chứa trong token JWT. Dưới đây là mô tả cách thức JWT được sử dụng trong quá trình ủy quyền:

* Tạo JWT với Claims Ủy Quyền:
  + Khi JWT được tạo sau quá trình xác thực, nó có thể bao gồm các claims liên quan đến ủy quyền. Các claims này thường chứa thông tin về vai trò (role) của người dùng và các quyền (permissions) liên quan.
  + Ví dụ, một JWT có thể chứa thông tin như "role": "admin" hoặc "permissions": ["create", "delete"].
* Gửi JWT trong Mọi Yêu Cầu:
  + Sau khi đăng nhập, người dùng sẽ gửi JWT trong mỗi yêu cầu đến server, thường là thông qua header Authorization.
  + Điều này giúp server xác định danh tính (xác thực) cũng như quyền truy cập (ủy quyền) của người dùng.
* Server Kiểm Tra JWT:
  + Khi nhận yêu cầu, server sẽ xác minh JWT, không chỉ kiểm tra tính hợp lệ của nó mà còn xem xét các claims ủy quyền.
  + Server sau đó xác định xem người dùng có quyền truy cập vào tài nguyên hoặc thực hiện hành động yêu cầu hay không.
* Truy Cập Dựa trên Ủy Quyền:
  + Nếu JWT cho thấy người dùng có quyền thích hợp, yêu cầu sẽ được xử lý. Nếu không, server sẽ từ chối truy cập và có thể trả về một thông báo lỗi như 403 Forbidden.
* Ưu điểm của Ủy Quyền JWT
* Trung Tâm và Nhất Quán: Quản lý ủy quyền trở nên dễ dàng hơn khi mọi thông tin cần thiết đều được chứa trong JWT.
* Khả Năng Mở Rộng: JWT hỗ trợ việc mở rộng hệ thống mà không cần thay đổi cơ sở dữ liệu hoặc cấu trúc backend.
* Bảo Mật: Các claims trong JWT được bảo vệ bởi signature, giảm thiểu nguy cơ giả mạo hoặc thay đổi.
* Hạn Chế
* Quản Lý Token: Việc quản lý và hủy bỏ JWT có thể phức tạp, đặc biệt nếu token bị lộ hoặc cần được thu hồi trước thời hạn.
* Lưu Trữ Thông Tin Nhạy Cảm: Không nên lưu trữ thông tin nhạy cảm trong JWT vì chúng có thể bị đọc nếu token bị tiết lộ.

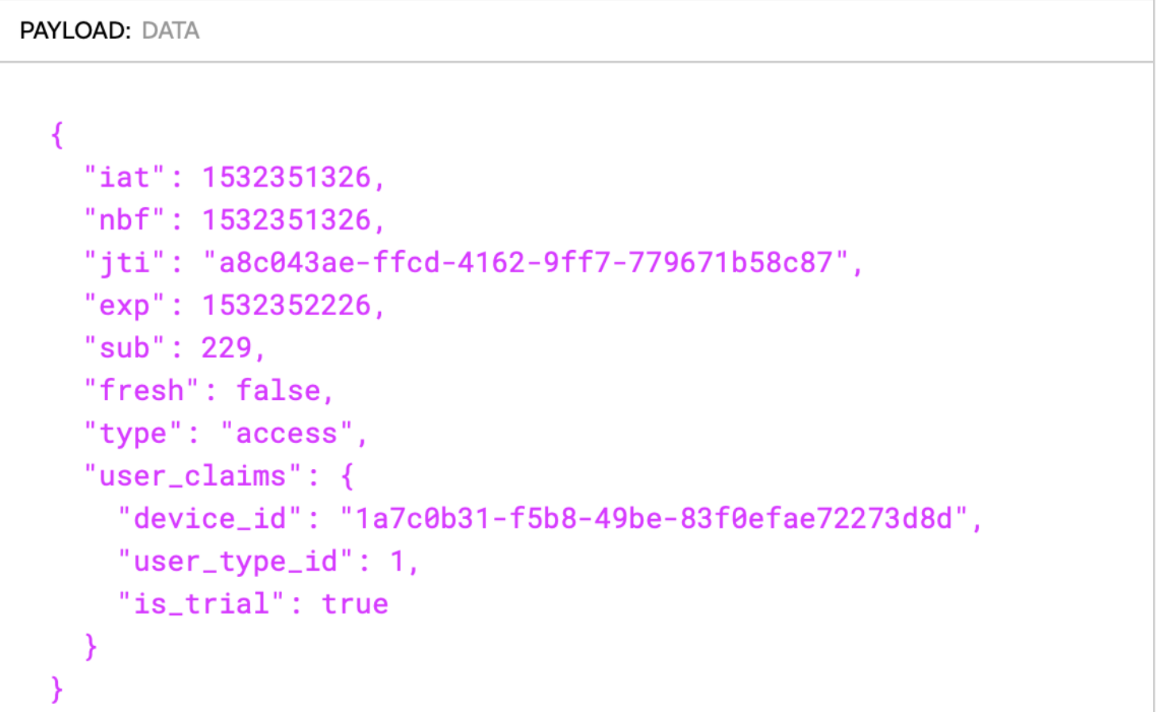
Ủy quyền dựa trên JWT là một phương pháp hiệu quả và linh hoạt trong việc quản lý quyền truy cập của người dùng trong ứng dụng, nhưng cần được xử lý cẩn thận để đảm bảo an ninh và hiệu quả [7].

### 2.2.3 Cơ chế xác thực

Xác thực thường sau khi người dùng đăng nhập, mỗi yêu cầu tiếp theo sẽ bao gồm JWT, cho phép người dùng truy cập các dịch vụ và tài nguyên được bảo vệ.

Quá Trình Xác Thực JWT sẽ diễn ra như sau:

* Đăng Nhập và Tạo Token:
  + Khi người dùng đăng nhập vào hệ thống bằng thông tin xác thực (Tên người dùng và mật khẩu), server sẽ xác minh thông tin này.
  + Nếu thông tin xác thực là hợp lệ, server tạo ra một JWT. JWT này chứa các thông tin (claims) về người dùng và một số metadata khác, như thời gian hết hạn của token.



Hình 5.Page load của JWT

* Gửi Token cho Người Dùng:
  + JWT sau khi được tạo sẽ được gửi trở lại cho client. Client này có thể là trình duyệt web, ứng dụng di động, hoặc bất kỳ loại client nào khác.
* Lưu Trữ Token:
  + Người dùng lưu trữ JWT ở phía client, thường là trong local storage của trình duyệt hoặc trong bộ nhớ của ứng dụng.
* Gửi Token trong các Yêu Cầu Tiếp Theo:
  + Trong mỗi yêu cầu đến server, JWT sẽ được đính kèm, thường là trong header Authorization của HTTP request.
  + Cách thức này giúp server biết rằng yêu cầu đang được thực hiện bởi người dùng đã xác thực [5].
* Xác Thực và Ủy Quyền từ Server:
  + Khi server nhận một yêu cầu với JWT, nó sẽ kiểm tra tính hợp lệ của token này.
  + Server xác minh JWT bằng cách kiểm tra signature để đảm bảo rằng nó không bị thay đổi hoặc giả mạo từ lúc được tạo.
  + Nếu JWT hợp lệ, yêu cầu được xử lý; nếu không, server sẽ trả về lỗi xác thực.

Ưu Điểm của JWT trong Xác Thực:

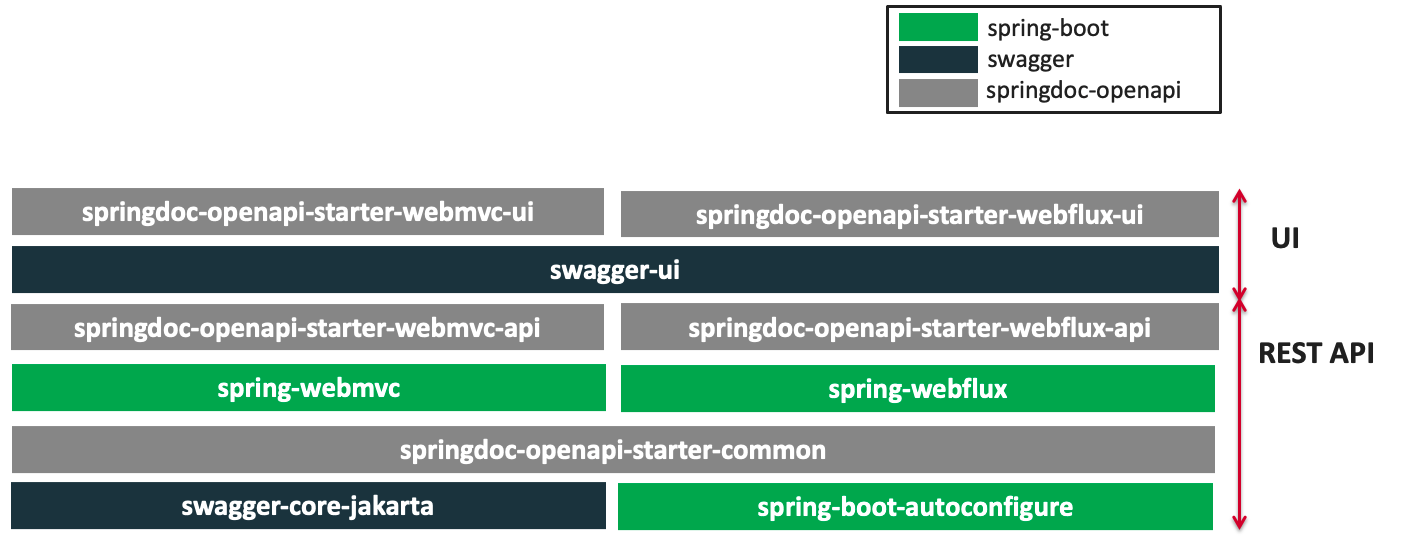
* Không Trạng Thái (Stateless): Server không cần lưu trữ thông tin xác thực hoặc phiên làm việc của người dùng, giúp giảm tải cho hệ thống.
* Bảo Mật và Hiệu Quả: JWT chứa signature để đảm bảo rằng nó không bị thay đổi. Nó cũng cho phép truyền thông tin một cách an toàn giữa client và server.

Hạn Chế:

* Bảo Mật: Thông tin trong JWT dễ bị đọc nếu token bị bắt giữ. Do đó, không nên lưu trữ thông tin nhạy cảm trong JWT.
* Quản Lý Phiên: JWT không thể bị hủy bỏ một cách dễ dàng trước khi hết hạn, vì vậy việc quản lý phiên có thể trở nên phức tạp nếu token bị chiếm đoạt [7].

## 2.3 Springdoc-openapi

Thư viện springdoc-openapi trong Java là một công cụ hữu ích giúp tự động hóa việc tạo tài liệu API cho các dự án Spring Boot. Thư viện này hoạt động bằng cách phân tích ứng dụng ở thời gian chạy để suy luận về ngữ nghĩa của API dựa trên cấu hình Spring, cấu trúc lớp và các annotation khác nhau [8].



Hình 6. Mô tả chung về springdoc-openai.

Springdoc-openapi là một thư viện được sử dụng trong các ứng dụng Spring Boot để tự động tạo ra tài liệu cho các API dựa trên OpenAPI 3 Specification. OpenAPI (trước đây được gọi là Swagger) là một tiêu chuẩn ngành công nghiệp cho việc mô tả, sản xuất và sử dụng RESTful services [8].

Tự Động Tạo Tài Liệu: springdoc-openapi có khả năng tự động phát hiện các endpoint trong ứng dụng và tạo ra tài liệu API tương ứng.

Hỗ Trợ OpenAPI 3: thư viện hỗ trợ phiên bản mới nhất của OpenAPI, cho phép mô tả API một cách chi tiết và đầy đủ hơn so với các phiên bản trước đây.

Tích Hợp UI Để Xem Tài Liệu: springdoc-openapi cung cấp tích hợp với Swagger UI, cho phép ta xem tài liệu API đã được tạo ra qua một giao diện người dùng trực quan và thân thiện.

Hỗ Trợ Đa Dạng: có khả năng hỗ trợ cả các ứng dụng WebFlux và Servlet, cũng như hỗ trợ cho các dịch vụ Reactive.

Cấu Hình Dễ Dàng: springdoc-openapi có thể được cấu hình một cách linh hoạt thông qua properties hoặc cấu hình Java.

Tích Hợp với Spring Security: có khả năng tích hợp với Spring Security để đưa ra các thông tin về xác thực và ủy quyền trong tài liệu API.

Tùy Chỉnh Tài Liệu: có thể tùy chỉnh tài liệu của mình thông qua việc sử dụng các annotation hoặc cung cấp thông tin bổ sung trong tài liệu OpenAPI.

Để sử dụng springdoc-openapi trong một ứng dụng Spring Boot, chỉ cần thêm dependency của springdoc-openapi vào file pom.xml (nếu sử dụng Maven) hoặc build.gradle (nếu sử dụng Gradle) của dự án. Sau khi thêm thành công, Spring Boot sẽ tự động cấu hình springdoc-openapi và có thể truy cập tài liệu API qua một URL mặc định, thường là /v3/api-docs cho tài liệu JSON và /swagger-ui.html để xem Swagger UI [8].

## 2.4 Tổng quan về MySQL

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở phổ biến, sử dụng ngôn ngữ truy vấn cơ sở dữ liệu SQL.

A logo with a dolphin and text

Description automatically generated

Hình 7. Logo MySQL

MySQL có mã nguồn mở, cho phép người dùng tự do sử dụng và sửa đổi theo nhu cầu của họ.

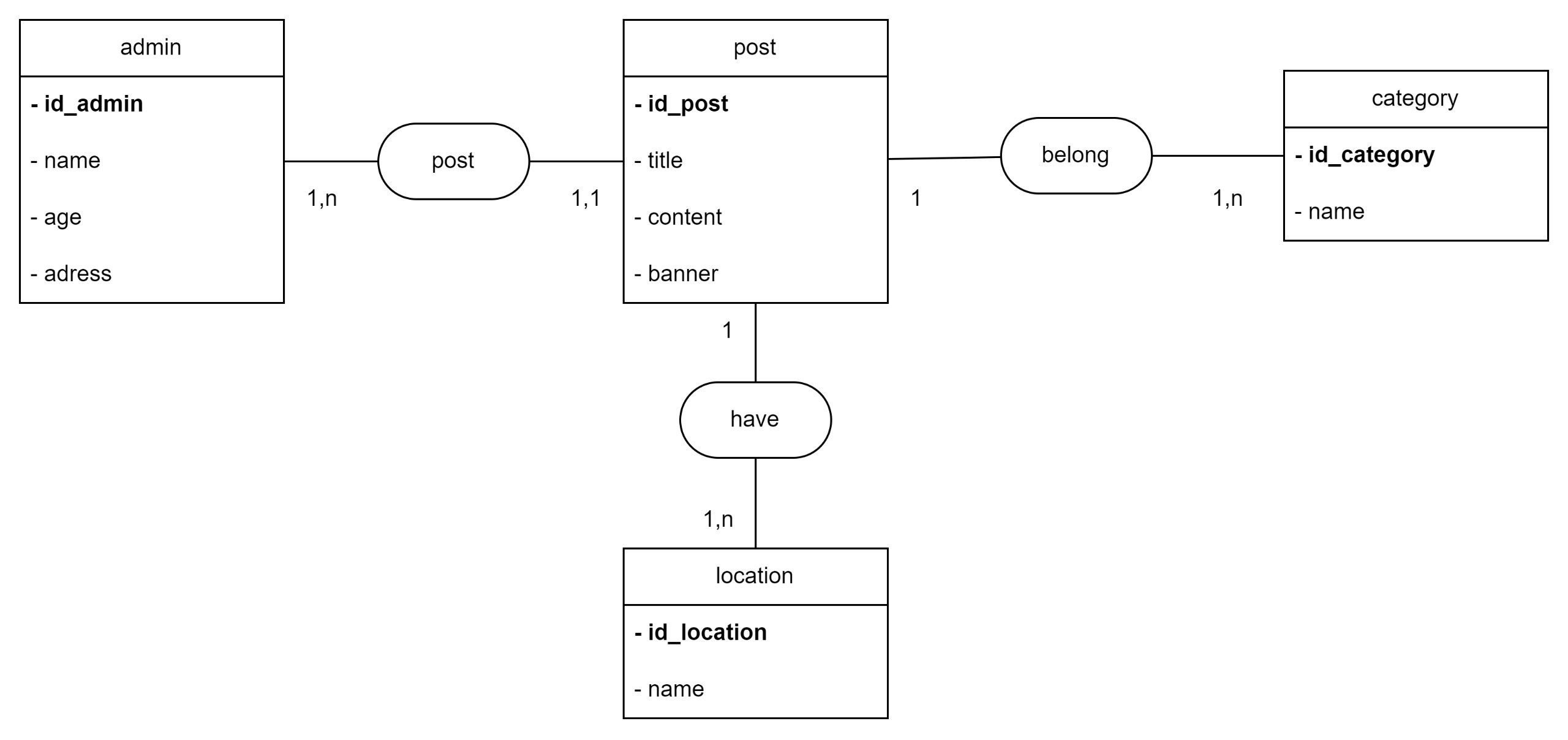
Dễ sử dụng: MySQL dễ cài đặt và sử dụng. Nó cung cấp một loạt các công cụ và giao diện để quản lý dữ liệu và cấu hình cơ sở dữ liệu.

Hiệu suất cao: MySQL được tối ưu hóa cho hiệu suất với các tính năng như bộ nhớ đệm truy vấn và bảng hash.

Bảo mật: MySQL cung cấp các tính năng bảo mật như xác thực, phân quyền và mã hóa để giúp bảo vệ dữ liệu.

Khả năng mở rộng: MySQL có thể xử lý một lượng lớn dữ liệu và được thiết kế để dễ dàng mở rộng [9] [9] [10].

# CHƯƠNG 3: THỰC HÓA NGHIÊM CỨU

Mô hình mức quan niệm:  


Mô tả thực thể:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Thuộc tính | Diễn giải | Loại già trị | Kiểu dữ liệu | Miền giá trị | Chiều dài | Ghi chú |
| MaNCC | Mã nhà cung cấp | BB | Nvarchar | Primary key | 50 |  |
| TenNCC | Tên nhà cung cấp | BB | Nvarchar |  | 50 |  |
| DiaChiNCC | Địa chỉ nhà cung cấp | BB | Nvarchar |  | 100 |  |
| SDTNCC | Số điện thoại nhà cung cấp | BB | Varchar (10) |  | 10 |  |

Mô hình dữ liệu mức logic

A diagram of a computer

Description automatically generated

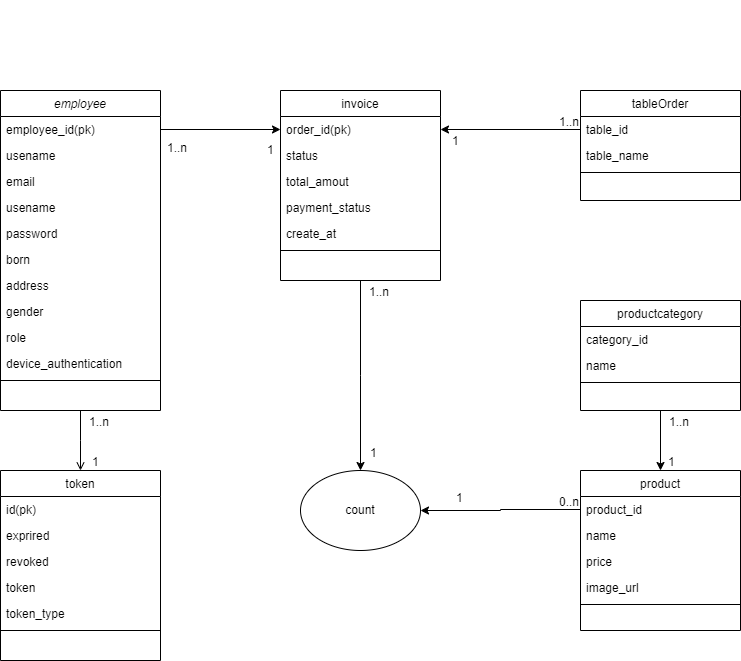
**Quan hệ dữ liệu: (tượng chưng)**

NhanVien (MaNVien, Hoten, NgaySinh, GioiTinh, DiaChiNV, SDTNV, TenDangNhap, MatKau, Quyen)

NhaCungCap (MaNCC, TenNCC, DiaChiNCC, SDTNNCC)

KhachHang (MaKH, TenKH,GioiTinhKH, SDTKH, DiaChiKH)

## 3.2 Mô hình thực thể kết hợp



Hình 9 Mô hình thực thể kết hợp

Hình ảnh trên mô tả một mô hình cơ sở dữ liệu quan hệ, thường được sử dụng trong thiết kế và lập trình cơ sở dữ liệu.

Mỗi mối quan hệ giữa các bảng được thể hiện bằng các đường nối có chú thích chỉ ra mối quan hệ số lượng (ví dụ: 1-n tức là một đến nhiều). Điều này chỉ ra cách các bảng liên kết với nhau.

## 3.4 Mô hình mức dữ liệu logic

A screenshot of a computer

Description automatically generated

Hình 10 Mô hình mức dữ liệu logic

Bảng 1. Bảng “productcategory”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên thuộc tính | Loại | Ràng buộc | Ý nghĩa/ghi chú |
| 1 | Category\_id | Int | Khóa chính | Mã thể loại |
| 2 | name | Varchar(50) |  | Tên thể loại |

Bảng 2. Bảng “product”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên thuộc tính | Loại | Ràng buộc | Ý nghĩa/ghi chú |
| 1 | product\_id | Int | Khóa chính | Mã sản phẩm |
| 2 | name | Varchar(50) |  | Tên sản phẩm |
| 3 | price | Decimal(10, 2) |  | Giá của sản phẩm |
| 4 | image\_url | Varchar(255) |  | Đường dẫn của ảnh |
| 5 | Category\_id | int | Khóa ngoại | Mã thể loại |

Bảng 3. Bảng “OrderItem”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên thuộc tính | Loại | Ràng buộc | Ý nghĩa/ghi chú |
| 1 | Category\_id | Int | Khóa ngoại | Mã thể loại |
| 2 | product\_id | int | Khóa ngoại | Mã sản phẩm |
| 3 | count | Int |  | Số lượng sản phẩm |

Bảng 4. Bảng “Invoice”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stt | Tên thuộc tính | Loại | Ràng buộc | Ý nghĩa/ghi chú |
| 1 | Invoice\_id | Int | Khóa chính | Mã hóa đơn |
| 2 | Tableorder\_id | Int | Khóa ngoại | Mã bàn |
| 3 | Employee\_id | Int | Khóa ngoại | Mã nhân viên |
| 4 | Payment\_status | Varchar(20) |  | Trạng thái hóa đơn |
| 5 | Total\_amount | Decimal(10, 2) |  | Tổng giá trị đơn hàng |
| 6 | Create\_at | Timetemp |  | Ngày lập hóa đơn |

Bảng 5. Bảng “productcategory”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên thuộc tính | Loại | Ràng buộc | Ý nghĩa/ghi chú |
| 1 | employee\_id | Int | Khóa chính | Mã người dùng |
| 2 | usename | Varchar(50) |  | Tên người dùng |
| 3 | email | Varchar(50) |  | Email của người dùng |
| 4 | password | Varchar(255) |  | Mật khẩu của người dùng |
| 5 | Born | Date |  | Ngày sinh cảu người dùng |
| 6 | address | Varchar(255) |  | Địa chỉ của người dùng |
| 7 | gender | Tinyint |  | Giới tính của người dùng |
| 8 | role | Varchar(10) |  | Vai trò của người dùng |
| 9 | device\_authentication | Varchar(10) |  | Mã xác thực |

Bảng 6. Bảng “token”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên thuộc tính | Loại | Ràng buộc | Ý nghĩa/ghi chú |
| 1 | id | Int | Khóa chính | Mã của token |
| 2 | expired | Tinyint |  | Token đã hết hạn chưa |
| 3 | revoked | Tinyint |  | Token đã thu hồi chưa |
| 4 | token\_type | Varchar(20) |  | Loại của token |
| 5 | employee\_id | int | Khóa ngoại | Mã của người dùng |

# **CHƯƠNG 4: KẾT QUẢ NGHIÊM CỨU**

# CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 5.1 Kết quả dạt được

Đề tài đã khảo sát các khái niệm cơ bản của RESTful API và cách sử dụng Java SpringBoot để thiết kế và triển khai một API cho module backend của ứng dụng Quản lý Quán Cafe. Tôi đã tạo ra các endpoint để thực hiện các thao tác CRUD trên các đối tượng như sản phẩm, đơn hàng, khách hàng và nhân viên. Kiểm tra và kiểm thử các endpoint bằng công cụ Postman hoặc Swagger. Tôi cũng đã áp dụng các kỹ thuật như phân trang, bảo mật và kiểm thử để nâng cao chất lượng và hiệu suất của API. Bằng cách sử dụng RESTful API, có thể tạo ra một module backend linh hoạt, mở rộng và dễ tích hợp với các module frontend khác nhau.

## 5.2 Hạn chế

Nhiên cứu này cũng chỉ ra một số hạn chế và khuyến nghị của việc xây dựng module backend cho ứng dụng Quản lý Quán Cafe bằng Java SpringBoot, như sau:

- Tính năng chưa hoàn thiện: ứng dụng chỉ hỗ trợ một số chức năng cơ bản như đặt bàn, thanh toán, kiểm tra doanh thu, chưa có tính năng quản lý nhân viên, quản lý kho hàng, quản lý khuyến mãi, etc. Cần phát triển thêm các tính năng này để tăng khả năng ứng dụng hóa của ứng dụng.

- Bảo mật chưa cao: ứng dụng chưa có tính năng phân quyền người dùng, cho phép bất kỳ ai cũng có thể truy cập vào module backend qua RESTful API.

- Còn một số lỗi: ứng dụng có thể gặp phải một số lỗi khi xử lý các yêu cầu HTTP từ module frontend. Cần kiểm tra và sửa lỗi kỹ lưỡng trước khi triển khai ứng dụng lên môi trường thực tế.

## 5.3 Hướng phát triển

Tích Hợp Công Nghệ Mới: Khám phá và tích hợp các công nghệ mới vào hệ thống hiện tại, như Spring WebFlux cho lập trình phản ứng, hoặc container hóa ứng dụng với Docker và Kubernetes để dễ dàng triển khai và quản lý.

Tối Ưu Hóa Hiệu Suất: Đánh giá và tối ưu hóa hiệu suất của các API, đảm bảo việc xử lý đồng thời số lượng lớn yêu cầu một cách nhanh chóng và ổn định.

Bảo Mật Cao Cấp: Tăng cường các biện pháp bảo mật, bao gồm việc cập nhật thường xuyên các phụ thuộc và sử dụng các kỹ thuật mã hóa tiên tiến hơn.

Phát Triển Dựa Trên Đám Mây: Di chuyển hệ thống backend lên môi trường đám mây để cải thiện khả năng mở rộng và độ tin cậy.

Phân Tích Dữ Liệu và AI: Áp dụng phân tích dữ liệu lớn và học máy để cung cấp thông tin chi tiết từ dữ liệu, cải thiện quyết định kinh doanh và tạo trải nghiệm cá nhân hóa cho khách hàng.

Giao Diện Người Dùng và UX: Phát triển và cải tiến giao diện người dùng phía client để đảm bảo trải nghiệm người dùng mượt mà và trực quan.

Tích Hợp Thanh Toán Điện Tử: Nhúng các giải pháp thanh toán điện tử để cung cấp một quy trình thanh toán liền mạch cho khách hàng.

Tích Hợp Thêm Các Mô-đun: Như quản lý lịch đặt bàn, chương trình khách hàng thân thiết, và tự động hóa quản lý tồn kho.

Mỗi hướng phát triển này đều nhằm mục đích tăng cường khả năng cạnh tranh, cải thiện hiệu quả hoạt động và đảm bảo ứng dụng Quản lý Quán Cafe có thể thích ứng linh hoạt với các yêu cầu thay đổi của thị trường.

# DANH MỤC TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | M. Heckler, Spring Boot: Up and Running, 2021. |
| [2] | S. Musib, Spring Boot in Practice, 2022. |
| [3] | A. Mankale, Spring Security 3.x Cookbook, Packt, 2013. |
| [4] | L. Spilca, Spring Security in Action, 2020. |
| [5] | M. Knutson, Spring Security, 2017. |
| [6] | N. Madden, API Security in Action, 2020. |
| [7] | D. Moore, Breaking down JSON Web Tokens: From pros and cons to building and revoking, 2022. |
| [8] | "swagger," [Online]. Available: https://swagger.io/. [Accessed 2 1 2024]. |
| [9] | T. N. N. Cương, Lý Thuyết Cơ Sở Dữ Liệu Quan Hệ Và Ứng Dụng, 2019. |
| [10] | G. Reese, MySQL Pocket Reference, 2003. |
|  |  |
|  |  |

# PHỤ LỤC