|  |
| --- |
| **TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI TP. HỒ CHÍ MINH**  **KHOA CÔNG NGHỆ THÔNG TIN** |



**WEBSITE BLOG CHIA SẺ BÀI VIẾT**

|  |  |
| --- | --- |
| **Sinh viên thực hiện** | **Mã số sinh viên** |
| Nguyễn Trần Đình Quân | 20h1120161 |
| Nguyễn Văn Hiếu | 20h1120126 |
| Lê Thành Đạt | 20h1120116 |
| Nguyễn Văn Tuấn | 20h1120167 |
| Nguyễn Quang Trường | 20h1120182 |

TP Hồ Chí Minh, 2023

**MỤC LỤC**

[**DANH MỤC HÌNH ẢNH** iii](#_Toc151767641)

[**DANH MỤC BẢNG** iv](#_Toc151767642)

[**DANH MỤC CÁC TỪ VIẾT TẮT** v](#_Toc151767643)

[**LỜI MỞ ĐẦU** 1](#_Toc151767644)

[**1.** **Tính cấp thiết của đề tài** 1](#_Toc151767645)

[**2.** **Tình hình nghiên cứu** 2](#_Toc151767646)

[**3.** **Mục đích nghiên cứu** 3](#_Toc151767647)

[**4.** **Phương pháp nghiên cứu** 3](#_Toc151767648)

[**5.** **Các kết quả đạt được của đề tài** 4](#_Toc151767649)

[**6.** **Kết cấu của TTTN** 4](#_Toc151767650)

[**CHƯƠNG 1: CƠ SỞ LÝ THUYẾT** 5](#_Toc151767651)

[**1.1.** **Spring Framework ?** 5](#_Toc151767652)

[**1.1.1.** **Spring Boot?** 5](#_Toc151767653)

[**1.1.2.** **Spring Security?** 6](#_Toc151767654)

[**1.1.3.** **JWT Authentication?** 7](#_Toc151767655)

[**1.2.** **ReactJs ?** 8](#_Toc151767656)

[**1.3.** **Database MySql ?** 9](#_Toc151767657)

[**CHƯƠNG 2: XÂY DỰNG WEBSITE** 11](#_Toc151767658)

[**2.1.** **Phân tích tổng quan** 11](#_Toc151767659)

[**2.1.1.** **Mô tả hệ thống** 11](#_Toc151767660)

[**2.2.** **Đặc tả các chức năng** 11](#_Toc151767661)

[**2.2.1.** **Chức năng cơ bản** 11](#_Toc151767662)

[**2.2.2.** **Chức năng quản lý bài đăng cá nhân** 12](#_Toc151767663)

[**2.2.3.** **Chức năng quản lý bài đăng của Admin** 14](#_Toc151767664)

[**2.2.4.** **Chức năng quản lý người dùng của Admin** 15](#_Toc151767665)

[**2.3.** **Xây dựng sơ đồ hệ thống** 16](#_Toc151767666)

[**2.3.1.** **Sơ đồ usecase** 16](#_Toc151767667)

[**2.3.2.** **Sơ đồ lớp hệ thống** 21](#_Toc151767668)

[**2.4.** **Thiết kế cơ sở dữ liệu** 21](#_Toc151767669)

[**2.4.1.** **Bảng người dùng** 21](#_Toc151767670)

[**2.4.2.** **Bảng bài đăng** 22](#_Toc151767671)

[**2.5.** **Thiết kế giao diện** 23](#_Toc151767672)

[**2.5.1.** **Giao diện đăng nhập** 23](#_Toc151767673)

[**2.5.2.** **Giao diện đăng ký** 23](#_Toc151767674)

[**2.5.3.** **Giao diện màn hình chính** 24](#_Toc151767675)

[**2.5.4.** **Giao diện đọc xem bài viết** 24](#_Toc151767676)

[**2.5.5.** **Giao diện đăng bài** 25](#_Toc151767677)

[**2.5.6.** **Giao diện quản lý bài đăng của người dùng** 25](#_Toc151767678)

[**2.5.7.** **Giao diện quản lý của Admin** 26](#_Toc151767679)

[**CHƯƠNG 3: TRIỂN KHAI WEBSITE** 27](#_Toc151767680)

[**KẾT LUẬN** 30](#_Toc151767681)

[**TÀI LIỆU THAM KHẢO** 31](#_Toc151767682)

**DANH MỤC HÌNH ẢNH**

[Thống kê 1 spring boot 2](#_Toc154335075)

[Thống kê 2 reactjs framework 3](#_Toc154335076)

[Hình 2.1 Sơ đồ usecase của website 16](#_Toc154335061)

[Hình 2.2 Sơ đồ usecase quản lý bài đăng người dùng 17](#_Toc154335062)

[Hình 2.3 Sơ đồ usecase quản lý bài đăng của Admin 18](#_Toc154335063)

[Hình 2.4 Sơ đồ usecase quản lý người dùng của Admin 19](#_Toc154335064)

[Hình 2.5 Sơ đồ usecase của người dùng vãng lai 20](#_Toc154335065)

[Hình 2.6 Sơ đồ lớp hệ thống 21](#_Toc154335066)

[Hình 2.7 Giao diện đăng nhập 23](#_Toc154335067)

[Hình 2.8 Giao diện đăng ký 23](#_Toc154335068)

[Hình 2.9 Giao diện màn hình chính 24](#_Toc154335069)

[Hình 2.10 Giao diện đọc xem bài viết 24](#_Toc154335070)

[Hình 2.11 Giao diện đăng bài 25](#_Toc154335071)

[Hình 2.12 Giao diện quản lý người dùng 25](#_Toc154335072)

[Hình 2.13 Giao diện quản lý của Admin 26](#_Toc154335073)

[Hình 3.1 Xác định api endpoints 27](#_Toc154335378)

[Hình 3.2 Sử dụng http request fetch 27](#_Toc154335379)

[Hình 3.3 Xử lý bằng useState 28](#_Toc154335380)

[Hình 3.4 Sử dụng Jsx 28](#_Toc154335381)

[Hình 3.5 Xử lý sự kiện 29](#_Toc154335382)

**DANH MỤC BẢNG**

[Bảng 2. 1 Chức năng cơ bản 18](#_Toc151766541)

[Bảng 2. 2 Chức năng quản lý bài đăng cá nhân 20](#_Toc151766542)

[Bảng 2. 3 Chức năng quản lý bài đăng của Admin 21](#_Toc151766543)

[Bảng 2. 4 Chức năng quản lý người dùng của Admin 21](#_Toc151766544)

[Bảng 2. 5 Bảng người dùng 27](#_Toc151766545)

[Bảng 2. 6 Bảng bài đăng 28](#_Toc151766546)

**DANH MỤC CÁC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **STT** | **Chữ viết tắt** | **Tên đầy đủ** |
| **1** | API | Application Programming Interface |
| **2** | JSON | JavaScript Object Notation |
| **3** | XML | Extensible Markup Language |
| **4** | HTML | HyperText Markup Language |
| **5** | HTTP | HyperText Transfer Protocol |
| **6** | DOM | Document Object Model |
| **7** | UI | User Interface |

**LỜI MỞ ĐẦU**

1. **Tính cấp thiết của đề tài**
   1. **Sự phổ biến của Spring Boot và ReactJs trong phát triển website**

Spring boot và ReactJs là hai công nghệ phổ biến được ứng dụng rộng rãi trong lĩnh vực phát triển web

* + 1. **Spring Boot**

Spring boot, là một framework mã nguồn mở được phát triển dựa trên nền tảng Java, cung cấp một cách nhanh chóng và tiện lợi để xây dựng các ứng dụng web Java-based, nó cung cấp mô hình lập trình và cấu hình toàn diện cho các ứng dụng doanh nghiệp. Tính năng chính và nổi bật của Spring Framework đó là Dependency Injection (DI) và Inversion of Control (IOC) giúp phát triển các ứng dụng với khả năng “loosely coupled”

* + 1. **ReactJs**

ReactJs, một thư viện JavaScript được tạo ra để xây dựng giao diện người dùng có khả năng tương tác tốt và nhanh chóng cho các ứng dụng web ngoài ra còn có di động. Nó được Facebook thiết kế nhằm giúp tạo ra các ứng dụng web hấp dẫn trong thời gian ngắn và hiệu quả mà không cần code quá nhiều. ReactJs tối ưu hiệu suất làm việc cao nhất, thông qua việc tập chung các component riêng lẻ lại với nhau. ReactJs sử dụng các các thành phần của giao diện tạo thành những component đơn giản hơn thay vì phải làm việc trên toàn bộ ứng dụng web.

* 1. **Sự cần thiết của việc tìm hiểu và sử dụng Spring Boot và ReactJs để xây dựng website blog**

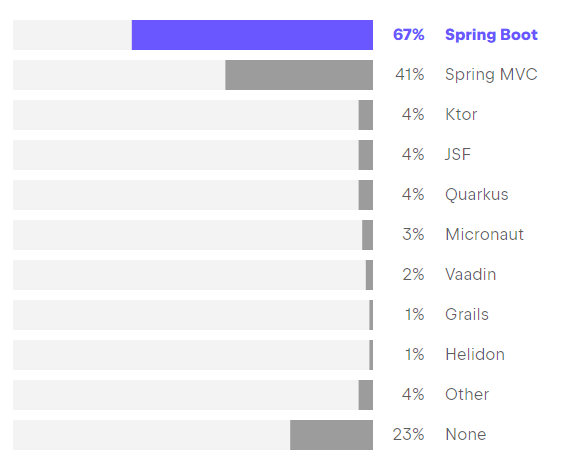
Chính vì sự phổ biến của 2 framework này mà lý do tại sao phải tìm hiểu Spring và React lại càng cần thiết, để đáp ứng được nhu cầu công việc và nhu cầu tuyển dụng của các công ty hiện nay. Qua đó thực hành và áp dụng vào việc xây dựng website blog.

Một số lợi ích của Spring Boot và ReactJs khi tìm hiểu và áp dụng:

* Tối ưu hóa hiệu suất: Spring Boot cung cấp một cấu trúc phát triển linh hoạt và tối ưu hóa, giúp tăng cường hiệu suất và mở rộng ứng dụng web.
* UI tương tác và linh hoạt: ReactJs cho phép xây dựng giao diện người dùng tương tác với khả năng tái sử dụng thành phần, tăng tính linh hoạt và trải nghiệm người dùng tốt hơn.
* Phát triển nhanh chóng: sử dụng Spring boot và ReactJs giúp tăng tốc quá trình phát triển, từ việc xây dựng giao diện đến việc quản lý backend, giảm thời gian và công sức.
* Hỗ trợ cộng động lớn: Cả Spring Boot và ReactJs vì sự phổ biến của nó mà đều có cộng động hỗ trợ lớn, cung cấp nhiều tài liệu, hướng dẫn và hỗ trợ từ cộng đồng, giúp tôi ưu hóa quá trình học tập và triển khai.

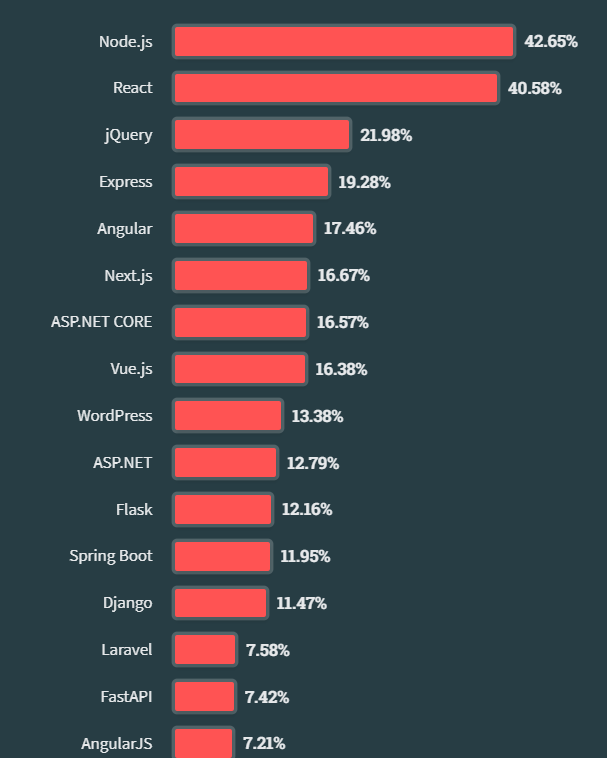
1. **Tình hình nghiên cứu**

**Spring Boot:** ngày nay spring boot vẫn mang lại sự đơn giản và hiệu quả cho việc phát triển Java. Không chỉ như vậy, điều thực sự àm nên sự khác biệt của Spring Boot so với các framework là khả năng phát triển mạnh mẽ của nó. Không đơn thuần là một công nghệ của quá khứ. Spring Boot luôn thích nghi để đáp ứng nhu cầu luôn thay đổi và phát triển của công nghệ hiện đại.



Thống kê spring boot

**ReactJs:** Qua nhiều năm phát triển và cạnh giữa vô số framework khác thì reactjs vẫn giữ được vị trí sử dụng phổ nhờ vào việc tái sử dụng các thành phần để tạo ra các giao diện cho tương tác và linh hoạt cao.



Thống kê reactjs framework

1. **Mục đích nghiên cứu**

Mục tiêu cuối cùng của việc tìm hiểu Spring Boot và Reactjs đó là để sử dụng và áp dụng vào các dự án sau này. Để thành thạo hơn thì mục tiêu trong dự án này là xây dựng một website blog để người dùng có thể chia sẻ bài viết của nhau và có thể đăng bài viết để chia sẻ tới mọi người giới hạn trong website dựa vào những mục tiêu như vậy có thể áp dụng spring boot làm các api để xử lý backend cho trang web và sử dụng reactjs để tạo giao diện tương tác với người dùng cuối.

Đây là một cơ hội để học hỏi và sau khi hoàn thành website có thể tự tin sử dụng spring boot cũng như reactjs cho các dự án khác để xây dựng hệ thống trang web có tính ổn định và dễ dàng bảo trì hơn.

1. **Phương pháp nghiên cứu**
   1. **Tìm hiểu công nghệ**

Tìm hiểu về cách Spring Boot và ReactJs vận hành, cách sử dụng, kết cấu ứng dụng. Tìm kiếm những ứng dụng hỗ trợ cho các framework này như jwt để tạo token, postman để kiểm thử api, boostrap để tạo giao diện,…

* 1. **Lên ý tưởng xây dựng trang web**

Ý tưởng ban đầu là xây dựng một website có thể upload các bài post đơn giản và tạo giao diện cho người dùng nhưng sau khi phát triển hơn thì đã các chức năng và tạo nên 1 trang web để người xem có thể đăng ký và đăng nhập để đăng bài viết mà mình thích cũng như quản lý bài viết của mình bên cạnh đó còn phần quyền cho admin để có thể quản lý user và quản lý các bài đăng trên trang của mình.

* 1. **Xây dựng backend Spring Boot**

Sau khi tìm hiểu, áp dụng để xây dựng các api dựa trên cấu trúc rest và sử dụng spring security để phân quyền và xác thực người dùng.

* 1. **Áp dụng backend vào và xây dựng frontend ReactJs**

Áp dụng những api đã được tạo vào và xây dựng giao diện bằng reactjs dựa vào công nghệ jsx để tạo được giao diện cho người dùng.

* 1. **Kiểm tra chức năng người dùng cuối**

Sau khi hoàn thành trang web sẽ tiến hành kiểm tra các chức năng người dùng cuối sẽ sử dụng.

1. **Các kết quả đạt được của đề tài**

Sau khi hoàn thành đề tài, kết quả đạt được trước hết là một trang web mà người dùng có thể đăng xóa sửa bài viết của mình cũng như admin có thể quản lý người dùng và các bài viết trên trang. Sau đó là hiểu rõ công nghệ xây dựng lên website là spring boot và reactjs để có thể sử dụng thành thạo hơn về 2 framework này cho các dự án về sau.

1. **Kết cấu của TTTN**

Bài báo cáo này bao gồm 3 chương:

Cơ sở lý thuyết, Xây dựng webstie, Triển khai website

Và phần kết luận tổng kết báo cáo.

**CHƯƠNG 1: CƠ SỞ LÝ THUYẾT**

* 1. **Spring Framework ?**

Spring framework là một framework phát triển ứng dụng Java mạnh mẽ và phổ biến, được thiết kế để xây dựng các ứng dụng doanh nghiệp Java. Được phát triển ban đầu bởi Rod Johnson vào năm 2002, Spring cung cấp một cách tiếp cận linh hoạt và mô hình lập trình dễ hiểu, giúp giảm thiểu sự phức tạp khi phát triển ứng dụng.

Các thành phần chính của Spring framework bao gồm:

* Dependency Injection (DI) và Inversion of Control (IoC): Spring framework sử dụng DI và IoC để quản lý các thành phần, làm tăng tính linh hoạt và tái sử dụng mã.
* Modularity: Spring được chia thành các module, mỗi module chịu trách nhiệm về một phần cụ thể của việc phát triển ứng dụng như JDBC, Security, MVC, và một số tính năng khác.
* Lớp Abstraction: Spring cung cấp các lớp trừu tượng hóa cho việc làm với các công nghệ khác nhau như JDBC, JPA và các dịch vụ web khác.
* Hỗ trợ mạnh mẽ cho các công nghệ và tiêu chuẩn: Spring hỗ trợ tích hợp với nhiều công nghệ và tiêu chuẩn Hibernate, JPA, RESTful, và nhiều loại dịch vụ web khác.
  + 1. **Spring Boot?**

Spring Boot là một phần mở rộng của Spring Framework, tập trung vào việc giảm bớt sự phức tạp trong việc phát triển ứng dụng Java dựa trên Spring. Điểm mạnh của nó nằm ở việc cung cấp một cách tiếp cận tự động hóa, tiết kiệm thời gian và tối ưu hóa công việc cấu hình để tập trung hoàn toàn vào việc viết code.

Các điểm nổi bật của spring boot:

* Auto-Configuration (Tự động cấu hình): Spring boot sử dụng cơ chế tự động cấu hình, dựa trên các convention và các tiêu chuẩn mặc định, giúp tự động cấu hình ứng dụng một cách thông minh. Điều này giảm thiểu việc phải cấu hình thủ công và tiết kiệm thời gian cho việc triển khai.
* Starter Dependencies: Spring Boot cung cấp các “starter” dependencies, là các công nghệ khác. Các starter này giúp dễ dàng bắt đầu và tích hợp các tính năng vào ứng dụng một cách nhanh chóng chỉ qua một vài cấu hình cơ bản.
* Embedded Servers: Spring Boot đi kèm với các máy chủ nhúng như Tomcat, Jetty hoặc Undertow, cho phép triển khai ứng dụng mà không cần phải cài đặt và cấu hình máy chủ riêng biệt.
  + 1. **Spring Security?**

Sring security là một phần mở rộng quan trọng của spring framework, cung cấp các tính năng bảo mật mạnh mẽ để bảo vệ ứng dụng Java. Nó cho phép xác thực (authentication) và ủy quyền (authorization) trong ứng dụng, bảo vệ các tài nguyên, endpoints và các hoạt động khác.

Các điểm chính của Spring Security:

* Xác thực (Authentication): Spring Security hỗ trợ nhiều phương pháp xác thực như xác thực dựa trên form, xác thực HTTP Basic/Digest, xác thực OAuth, JWT. Nó cho phép tích hợp với cơ sở dữ liệu người dùng hoặc các hệ thống xác thực bên ngoài.
* Ủy quyền (Authorization): Nó cung cấp cơ chế để quản lý và kiểm soát quyền truy cập vào tài nguyên và hoạt động trong ứng dụng, Bạn có thể xác định các quyền truy cập cho từng người dùng hoặc nhóm người dùng cụ thể.
* Session Management (Quản lý phiên): Spring Security quản lý các phiên người dùng, cho phép tuỳ chỉnh cách quản lý và xử lý phiên đăng nhập, bao gồm cách thức xác định phiên, hết hạn phiên, và cơ chế bảo vệ phiên tránh khỏi các cuộc tấn công.
* CSRF Protection (Bảo vệ chống tấn công CSRF): Nó cung cấp cơ chế bảo vệ chống tấn công giả mạo yêu cầu từ trang web khác (Cross – site request forgery) bằng cách sử dụng token.
  + 1. **JWT Authentication?**

JWT (JSON Web Token) Authentication là một phương thức xác thực phổ biến được sử dụng trong việc xác thực người dùng và ủy quyền truy cập vào các tài nguyên trong ứng dụng web hoặc dịch vụ web.

Cơ chế JWT Authenticaiton:

* Token based authentication: JWT sử dụng token để xác thực người dùng. Khi người dùng đăng nhập thành công, hệ thống sẽ tạo một JWT chứa thông tin xác thực như user ID, quyền truy cập, thời gian hết hạn, và các thông tin khác.
* Token structure: JWT bao gồm ba phần: Header, Payload, và Signature. Header chứa thông tin về thuật toán mã hóa và kiểu token. Payload lưu trữ các thông tin xác thực. Signature sử dụng chữ ký để xác thực tính hợp lệ của token.
* Stateless và phân phối: JWT là một giải pháp không lưu trạng thái (stateless), nghĩa là server không cần lưu trữ thông tin phiên hay token. Mỗi yêu cầu từ client đều đi kèm với token, giúp cho hệ thống dễ dàng mở rộng và không phụ thuộc vào state của server.
* Sử dụng trong API Authorization: Khi người dùng gửi yêu cầu đến server, họ cần gửi token JWT kèm theo. Server sẽ kiểm tra và giải mã token, sau đó kiểm tra xem nó có hợp lệ hay không. Nếu hợp lệ, yêu cầu sẽ được xử lý và trả về kết quả.
* An toàn và bảo mật: JWT được ký và mã hóa, đảm bảo tính toàn vẹn và bảo mật của dữ liệu. Chữ ký đảm bảo rằng token không bị thay đổi trên đường truyền.
* Thời gian hết hạn: JWT có thể được thiết lập thời gian hết hạn, giúp đảm bảo an toàn thông tin khi token không còn hiệu lực.
  1. **ReactJs ?**

ReactJs là một thư viện JavaScript phổ biến và mạnh mẽ được sử dụng rộng rãi để phát triển giao diện người dùng (UI) trong các ứng dụng web hiện đại. Được Facebook tạo ra và duy trì, ReactJS tập trung vào việc tạo ra giao diện người dùng linh hoạt và dễ bảo trì, đặc biệt trong các ứng dụng web có tương tác cao và các ứng dụng đơn trang (single-page applications - SPAs).

Một trong những đặc điểm quan trọng nhất của ReactJS là mô hình component-based. React cho phép phát triển giao diện bằng cách chia UI thành các thành phần (components) độc lập, mỗi thành phần quản lý một phần nhỏ của giao diện. Điều này tạo điều kiện thuận lợi cho việc tái sử dụng và bảo trì mã nguồn, vì mỗi component có thể được sử dụng lại hoặc thay thế mà không làm ảnh hưởng đến các phần khác trong ứng dụng.

Những đặc điểm quan trọng của ReactJs:

* Component – Based: ReactJS sử dụng cấu trúc dựa trên thành phần (component-based structure), cho phép phát triển giao diện bằng cách tạo ra các thành phần độc lập, có khả năng tái sử dụng.
* Virtual DOM: React sử dụng Virtual DOM để tối ưu hóa hiệu suất của ứng dụng. Thay vì cập nhật toàn bộ DOM khi có thay đổi, React sử dụng Virtual DOM để xác định các thay đổi cần áp dụng vào DOM thực tế, giúp tăng tốc độ hiển thị và tương tác.
* JSX: JSX là một phần mở rộng của JavaScipt, cho phép viết HTML giống như cú pháp JavaScript, giúp việc tạo UI trở nên dễ dàng và rõ rang hơn.
* Single page Application (SPAs): React thường được sử dụng để xây dựng các ứng dụng đơn trang (single-page applications), nơi mà toàn bộ nội dung của trang web không được tải lại khi người dùng tương tác.
* Data Binding: React hỗ trợ mô hình dữ liệu một chiều (unidirectional data flow), giúp dễ dàng quản lý và theo dõi dữ liệu khi có thay đổi.
* Community và Ecosystem phong phú: React có một cộng đồng lớn, cung cấp nhiều thư viện và công cụ hỗ trợ để xây dựng và phát triển ứng dụng một các hiệu quả.

**React hooks** là một tính năng mới được giới thiệu từ phiên bản React 16.8. Hooks cho phép bạn sử dụng các tính năng của React trước đây chỉ có thể sử dụng trong class components, như state, lifecycle methods, và các tính năng của React mà không cần sử dụng class.

Hooks được thiết kế để giải quyết những vấn đề phổ biến khi sử dụng class components, như logic lặp lại (repetitive logic), khó khăn trong việc chia sẻ logic giữa các component không liên quan, hay khó khăn khi tái sử dụng logic giữa các lifecycle methods.

Một số hooks phổ biến trong React bao gồm:

* useState: Cho phép bạn sử dụng state trong functional components. Ví dụ, const [count, setCount] = useState(0); để quản lý state của một biến count.
* useEffect: Thực thi side effects trong functional components. Ví dụ, có thể sử dụng useEffect để gọi một API khi component được render.
* useContext: Cho phép sử dụng Context API của React mà không cần wrapping trong class components.
* useReducer: Cung cấp một cách thức để quản lý state phức tạp hơn bằng cách sử dụng một reducer.
* Custom hooks: Có thể tạo ra các hooks tùy chỉnh, cho phép tái sử dụng logic giữa các component.
  1. **Database MySql ?**

MySQL là một hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở phổ biến, phát triển và duy trì bởi Oracle Corporation. Nó là một phần mềm mã nguồn mở, cung cấp tính năng quản lý cơ sở dữ liệu quan hệ mạnh mẽ, linh hoạt và ổn định.

Một số điểm nổi bật của MySQL:

* Mã nguồn mở: MySQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở, điều này có nghĩa là có thể sử dụng, chỉnh sửa và phân phối nó miễn phí. Điều này giúp hỗ trợ một cộng đồng lớn và phong phú về kiến thức và tài liệu.
* Đa nền tảng: MySQL hỗ trợ nhiều nền tảng khác nhau, từ Windows, Linux đến macOS, cung cấp sự linh hoạt cho các nhà phát triển.
* Tốc độ và hiệu suất: MySQL cung cấp hiệu suất cao và tốc độ xử lý nhanh, đặc biệt là trong các ứng dụng yêu cầu truy cập dữ liệu nhanh và xử lý lượng dữ liệu lớn.
* An toàn và bảo mật: MySQL cung cấp các tính năng bảo mật mạnh mẽ bao gồm quản lý quyền truy cập, mã hóa dữ liệu và kiểm soát truy cập chi tiết.
* Cộng đồng và tài liệu phong phú: Có một cộng đồng lớn của các nhà phát triển, chuyên gia và người dùng MySQL trên toàn thế giới. Bạn có thể dễ dàng tìm thấy các tài liệu, hướng dẫn, và hỗ trợ từ cộng đồng này.
* Hỗ trợ cho nhiều loại dữ liệu: MySQL hỗ trợ nhiều loại dữ liệu khác nhau như số, chuỗi, ngày tháng, đồng thời cũng hỗ trợ các loại dữ liệu phức tạp như JSON, XML, và nhiều loại dữ liệu khác.

Trong Spring Boot, MySQL thường được sử dụng làm cơ sở dữ liệu phổ biến để lưu trữ và quản lý dữ liệu cho ứng dụng. Spring Boot cung cấp cách tiếp cận dễ dàng để kết nối và làm việc với MySQL thông qua các công cụ như spring-boot-starter-data-jpa và các cấu hình liên quan.

Thông qua spring-boot-starter-data-jpa, Spring Boot cho phép tạo các Repository và Entity để tương tác với cơ sở dữ liệu MySQL một cách dễ dàng và linh hoạt. Cấu hình kết nối cơ sở dữ liệu, như thông tin về URL, username, password, và các properties khác, thường được đặt trong file cấu hình application.properties.

Khi tích hợp MySQL vào Spring Boot, việc thực hiện các thao tác CRUD (Create, Read, Update, Delete) trở nên thuận tiện và mạnh mẽ, giúp xây dựng ứng dụng có khả năng lưu trữ và xử lý dữ liệu hiệu quả.

**CHƯƠNG 2: XÂY DỰNG WEBSITE**

* 1. **Phân tích tổng quan**
     1. **Mô tả hệ thống**

**Các chức năng cơ bản:** tìm kiếm, đọc bài, đăng ký hay đăng nhập tài khoản thì các tác nhân đều có thể thực hiện được.

**Tác nhân là Vistor (Người dùng vãng lai):** là người dùng chưa từng đăng ký tài khoản thì chỉ có thể tìm kiếm, xem bài viết và đăng ký tài khoản.

**Người dùng:** Khi đã có tài khoản đã được đăng ký thì đăng nhập tài khoản để có thể thực hiện chức năng quan trọng là đăng bài, quản lý bài viết của bản thân.

**Đối với Admin (Quản trị viên):** có quyền cao nhất phải thực hiện các hoạt động như:

* Duy trì, bảo trì hệ thống: Admin hỗ trợ người dùng ngay khi hệ thống có lỗi hay các vấn đề ảnh hưởng tới quá trình vận hành.
* Quản lý các người dùng: Admin sẽ quản lý tất cả các thông tin về người dùng, có thể cập nhật cho người dùng trong trường hợp bị lỗi hoặc có thể xóa trực tiếp tài khoản đó nếu tài khoản đó vi phạm chính sách của Website
* Quản lý các bài đăng của tất cả người dùng: Tương tự như quản lý người dùng thì Admin cũng có quyền quản lý cả bài viết của tất cả các cá nhân đã đăng ký tài khoản tại Web và đã đăng bài Admin chỉ có thể xem bài và xóa bài viết không có quyền chỉnh sửa bài viết.
  1. **Đặc tả các chức năng**
     1. **Chức năng cơ bản**

|  |  |  |  |
| --- | --- | --- | --- |
| **Chức năng** | **Yêu cầu chức năng** | **Mô tả chức năng chi tiết** | **Yêu cầu phi chức năng** |
| **Cơ bản** | Đăng ký | Chức năng này xử lý khi một người dùng mới muốn tham đăng bài thì thực hiện đăng ký để tạo tài khoản | * Giao diện trực quan. * Khả năng tương tác với người dùng dễ dàng. * Bảo mật * Trả dữ liệu tìm kiếm với độ trễ thấp nhất |
|  | Đăng nhập | Xử lý cho những người đã có tài khoản, người dùng cần phải đăng nhập mới có thể dùng các chức năng sau này |
|  | Đăng xuất | Xử lý đăng xuất khi người dùng muốn đăng nhập hay đăng ký 1 tài khoản khác. |
|  | Tìm kiếm | Chức hỗ trợ người dùng thực hiện các tìm kiếm về các bài đăng ngay trong trang chủ. |
|  | Trang đọc bài | Đây trang hiển thị các thông tin của 1 bài đăng: Tiêu đề, nội dung, ngày đăng, người đăng. |

Bảng 2. Chức năng cơ bản

* + 1. **Chức năng quản lý bài đăng cá nhân**

|  |  |  |  |
| --- | --- | --- | --- |
| **Chức năng** | **Yêu cầu chức năng** | **Mô tả chức năng chi tiết** | **Yêu cầu phi chức năng** |
| **Cơ bản** | Hiển thị thông tin của bài đăng | Hiển thị các bài đăng hiện đã được đăng của người dùng chức năng này xử lý khi người dùng ấn vừa ấn vào trang quản lý bài đăng của người dùng. | * Giao diện trực quan. * Toàn vẹn dữ liệu. * Khả năng tương tác với người dùng dễ dàng. * Nhanh chóng cập nhật lại giao diện sau khi người dùng thực hiện đăng, xóa sửa. * Hỗ trợ lưu trữ thông tin lớn * Trả về tìm kiếm nhanh. |
|  | Thông kê số lượng bài đăng | Xử lý hỗ trợ người dùng khi quản lý các bài đăng xem số lượng đã đăng hiện tại. |
|  | Tìm kiếm | Chức năng hỗ trợ người dùng thực hiện các tìm kiếm về các bài đăng trong quá trình quản lý. |
|  | Đăng bài | Chức năng cho phép người dùng đăng các bài đăng với các text, hình ảnh. |
|  | Cập nhật bài đăng | Cập nhật các thông tin như: Tiêu đề, nội dung. |
|  | Xóa bài đăng | Xóa 1 bài đăng khi người dùng chọn và ấn vào nút xóa. |
|  | Cập nhật trạng thái giao diện | Cập nhật trạng thái sau khi đăng hay chỉnh sửa, xóa. |

Bảng 2. Chức năng quản lý bài đăng cá nhân

* + 1. **Chức năng quản lý bài đăng của Admin**

|  |  |  |  |
| --- | --- | --- | --- |
| **Chức năng** | **Yêu cầu chức năng** | **Mô tả chức năng chi tiết** | **Yêu cầu phi chức năng** |
| **Cơ bản** | Tìm kiếm | Chức năng hỗ trợ Admin thực hiện các tìm kiếm về các bài đăng trong quá trình quản lý. | * Giao diện trực quan. * Khả năng tương tác với người dùng dễ dàng. * Bảo mật * Trả dữ liệu tìm kiếm với độ trễ thấp nhất |
|  | Hiển thị thông tin của bài đăng | Admin có thể xem được các bài đăng hiện đã được đăng của các người dùng. Để có thể thực hiện các nghiệp vụ khác. |
|  | Đăng xuất | Xử lý đăng xuất khi người dùng muốn đăng nhập hay đăng ký 1 tài khoản khác. |
|  | Tìm kiếm | Chức hỗ trợ người dùng thực hiện các tìm kiếm về các bài đăng ngay trong trang quản lý của Admin |
|  | Thông kê số lượng bài đăng | Xử lý hỗ trợ Admin khi quản lý các bài đăng xem số lượng các bài đăng hiện có. |
|  | Xóa bài đăng | Admin có quyền xóa bất kì một bài đăng. |
|  | Cập nhật trạng thái giao diện | Cập nhật trạng thái sau khi xóa. |

Bảng 2. Chức năng quản lý bài đăng của Admin

* + 1. **Chức năng quản lý người dùng của Admin**

|  |  |  |  |
| --- | --- | --- | --- |
| **Chức năng** | **Yêu cầu chức năng** | **Mô tả chức năng chi tiết** | **Yêu cầu phi chức năng** |
| **Cơ bản** | Tìm kiếm | Chức hỗ trợ người dùng thực hiện các tìm kiếm về các tài khoản của người dùng ngay trong trang quản lý của Admin | * Giao diện trực quan. * Khả năng tương tác với người dùng dễ dàng. * Bảo mật * Trả dữ liệu tìm kiếm với độ trễ thấp nhất * Xử lý nhanh chóng khi thực hiện các thao tác. * Đảm bảo ổn định trong quá trình cập nhật hay xóa. |
|  | Xóa tài khoản | Admin có quyền xóa, hoặc ban bất kì một tài khoản trong trường hợp người dùng đó vi phạm chính sách của website. |
|  | Cập nhật | Xử lý cập nhật lại tài khoản trong trường hợp người dùng đó có thể gặp lỗi hay thao tác thực hiện từ người dùng đó không được. |

Bảng 2. Chức năng quản lý người dùng của Admin

* 1. **Xây dựng sơ đồ hệ thống**
     1. **Sơ đồ usecase**
        1. **Sơ đồ usecase của website**

*A diagram of a diagram

Description automatically generated*

Hình 2. Sơ đồ usecase của website

* + - 1. **Sơ đồ usecase quản lý bài đăng người dùng**

A diagram of a diagram

Description automatically generated

Hình 2. Sơ đồ usecase quản lý bài đăng người dùng

**Tác nhân:** User (Người dùng)

**Mục đích:** Quản lý bài đăng của người dùng

**Mô tả:** Người dùng có thể thêm, xóa, sửa bài đăng.

* + - 1. **Sơ đồ usecase quản lý bài đăng của Admin**

A diagram of a diagram with Ice hockey rink in the background

Description automatically generated

Hình 2. Sơ đồ usecase quản lý bài đăng của Admin

**Tác nhân:** Admin (Quản trị viên)

**Mục đích:** Quản lý bài đăng của tất cả người dùng

**Mô tả:** Admin có thể xem hoặc xóa bài đăng.

* + - 1. **Sơ đồ usecase quản lý người dùng của Admin**

*A diagram of a diagram with Ice hockey rink in the background

Description automatically generated*

Hình 2. Sơ đồ usecase quản lý người dùng của Admin

**Tác nhân:** Admin (Quản trị viên)

**Mục đích:** Quản lý người dùng

**Mô tả:** Admin có thể xem hoặc cập nhật lại thông tin của người dùng hoặc xóa tài khoản của người dùng đó .

* + - 1. **Sơ đồ usecase của người dùng vãng lai**

*A diagram of a person

Description automatically generated*

Hình 2. Sơ đồ usecase của người dùng vãng lai

**Tác nhân:** Vistor (Khách vãng lai người dùng chưa từng đăng ký tài khoản)

**Mục đích:** Các chức năng của người dùng chưa đăng ký tài khoản.

**Mô tả:** Người dùng chưa đăng ký tài khoản có thể đăng ký sau đó đăng nhập,xem tất cả các bài đăng và có thể tìm kiếm các bài đăng muốn xem

* + 1. **Sơ đồ lớp hệ thống**

**A diagram of a computer

Description automatically generated**

Hình 2. Sơ đồ lớp hệ thống

* 1. **Thiết kế cơ sở dữ liệu**
     1. **Bảng người dùng**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** | **Khóa chính** | **Khóa ngoại** | **Mô tả** |
| **1** | id | Long | X |  | id định danh người dùng |
| **2** | username | String |  |  | Tên tài khoản người dùng |
| **3** | email | String |  |  | Email của người dùng |
| **4** | password | String |  |  | Mật khẩu đăng nhập |
| **5** | roles | String |  |  | Quyền hạn của người dùng |

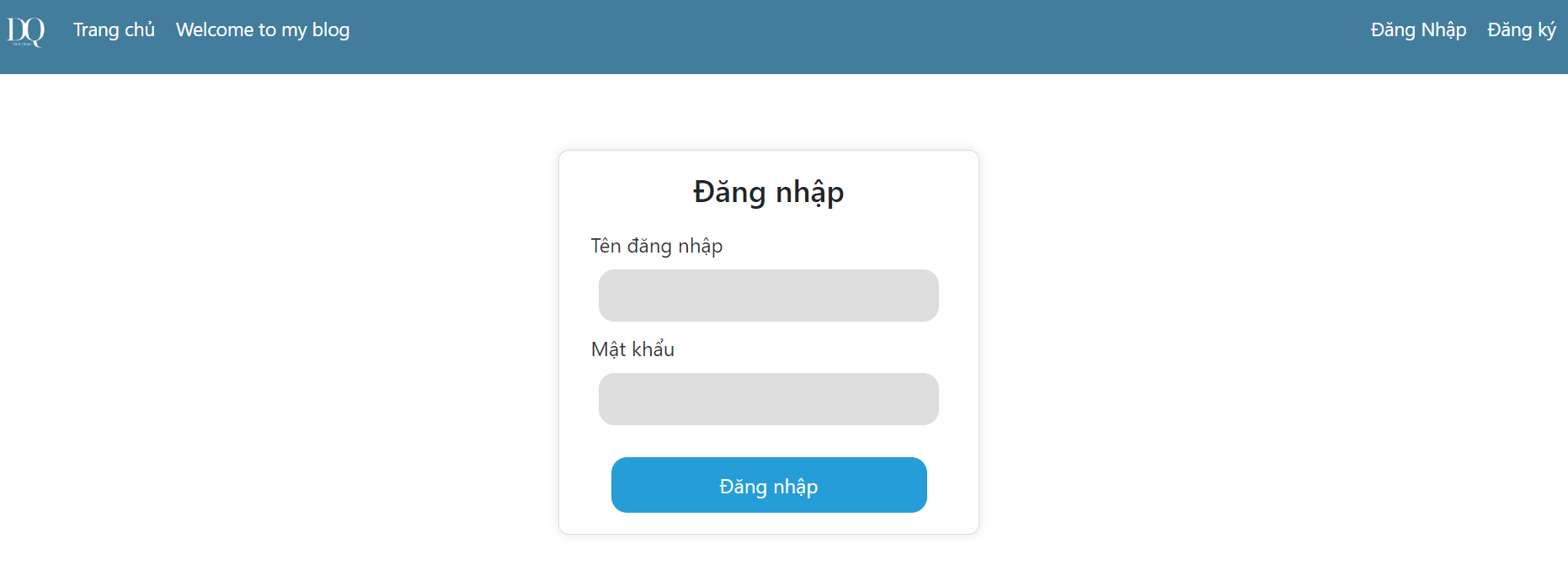
Bảng 2. Bảng người dùng

* + 1. **Bảng bài đăng**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** | **Khóa chính** | **Khóa ngoại** | **Mô tả** |
| **1** | id | Long | X |  | id của bài viết |
| **2** | postTitle | Text |  |  | Tiêu đề bài viết |
| **3** | postBody | Text |  |  | Nội dung bài viết |
| **4** | postImg | Text |  |  | Ảnh tiều đề bài viết |
| **5** | postDescribe | Text |  |  | Nội dung nổi bật của bài viết |
| **6** | nameAuthor | Text |  |  | Tên tác giả bài viết |
| **7** | datePost | Text |  |  | Ngày đăng bài |
| **8** | userid | Long |  |  | Gán id của người dùng vào để xác thực bài viết của ai |

Bảng 2. Bảng bài đăng

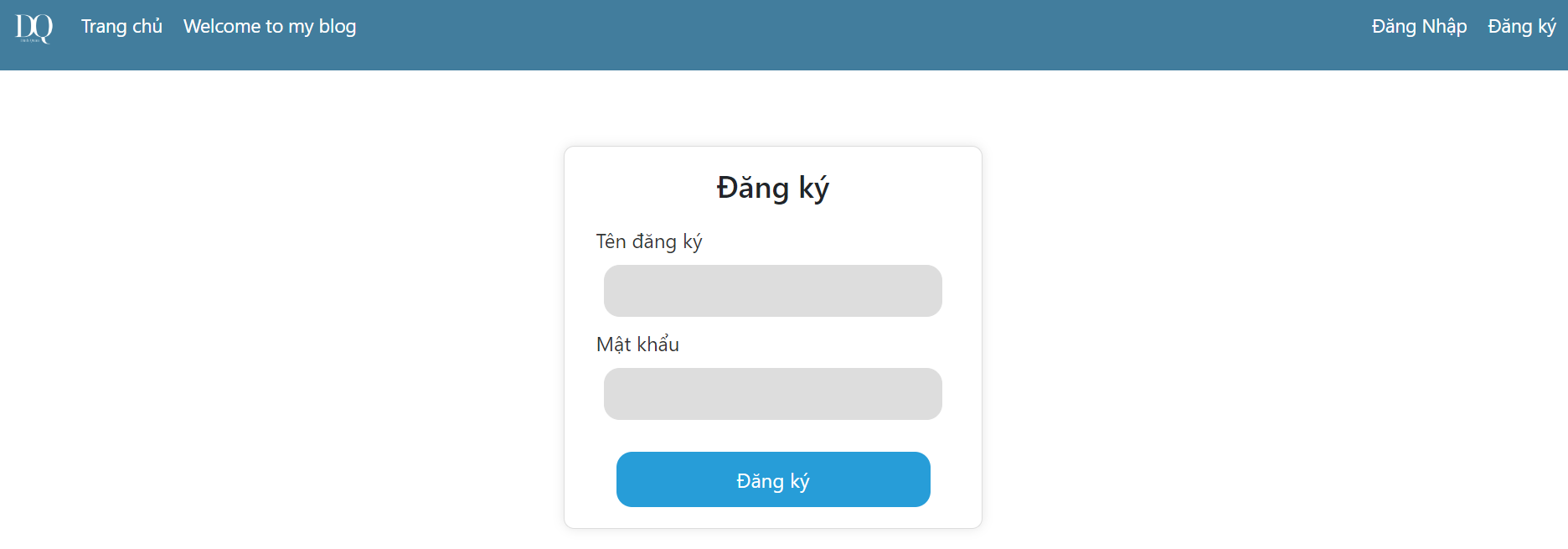
* 1. **Thiết kế giao diện**
     1. **Giao diện đăng nhập**

****

Hình 2. Giao diện đăng nhập

Giao diện được thiết kế dành cho người dùng đăng nhập khi có tài khoản yêu cầu đăng nhập gồm có Tên đăng nhập và Mật khẩu

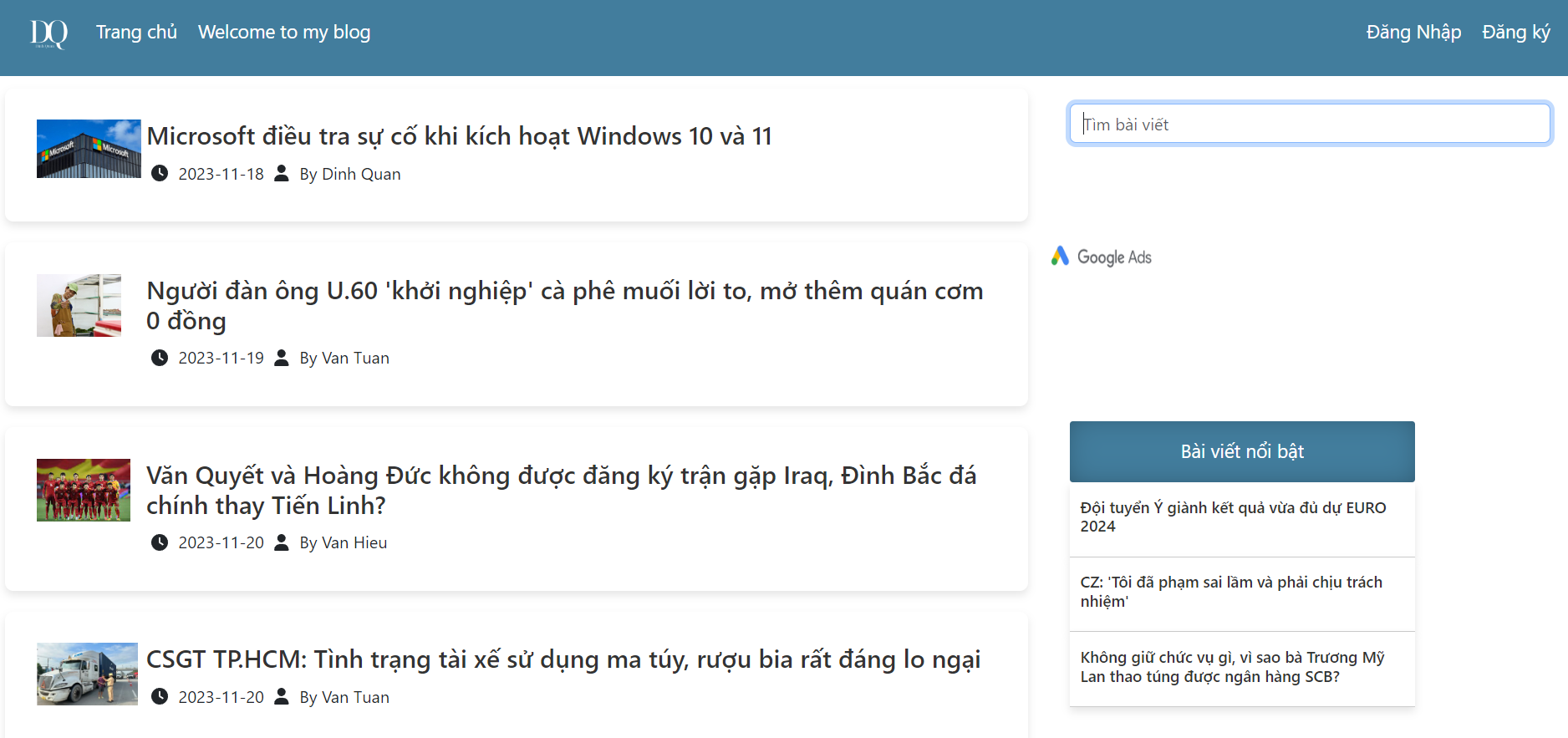
* + 1. **Giao diện đăng ký**

****

Hình 2. Giao diện đăng ký

Giao diện đăng ký gồm có 3 thông tin cần được nhập vào: Tên đăng nhập, Email, Mật khẩu

* + 1. **Giao diện màn hình chính**

****

Hình 2. Giao diện màn hình chính

Giao diện được thiết kế gốm có thanh điều hướng hiển thị trên cùng gồm các nút điều hướng và thanh tìm kiếm

Dưới thanh điều hướng gồm có 2 thành phần nội dung được chia bên trái hiển thị các bài đăng hiện tại đang có và bên phải là 1 phần hiển thị các nội dung bài nổi bật

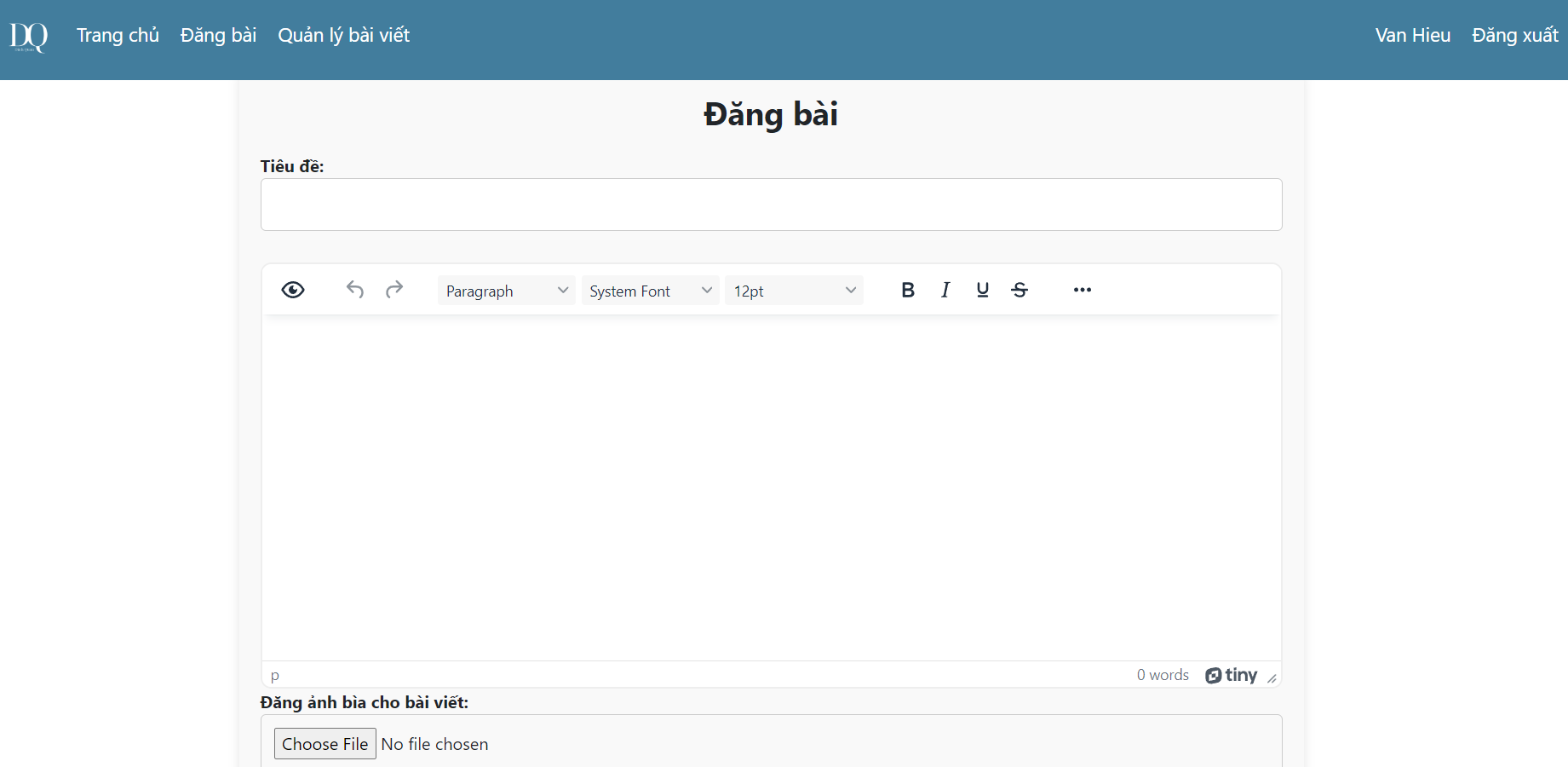
* + 1. **Giao diện đọc xem bài viết**

****

Hình 2. Giao diện đọc xem bài viết

Đối với giao diện đọc bài thì tương tự như màn hình chính thì phần nội dung trái được thay thế và hiển thị nội dung chi tiết của 1 bài viết mà người dùng đã truyền tải.

* + 1. **Giao diện đăng bài**

****

Hình 2. Giao diện đăng bài

Giao diện đăng bài thì gồm các ô để nhập các thông tin 1 bài viết như: Tiêu đề, Nội dung nổi bật là phần nội dung sẽ hiển thị bên ngoài của màn hình chính giúp thu hút người đọc bài, Phần nội dung chính được thiết kế để người dùng có thể thêm các kiểu chữ viết hay định dạng tùy ý

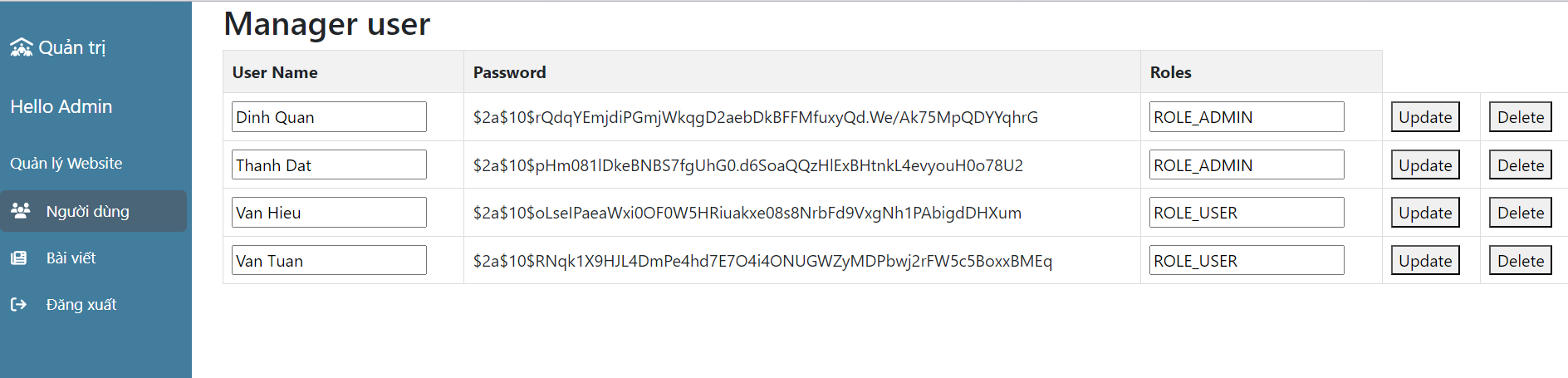
* + 1. **Giao diện quản lý bài đăng của người dùng**

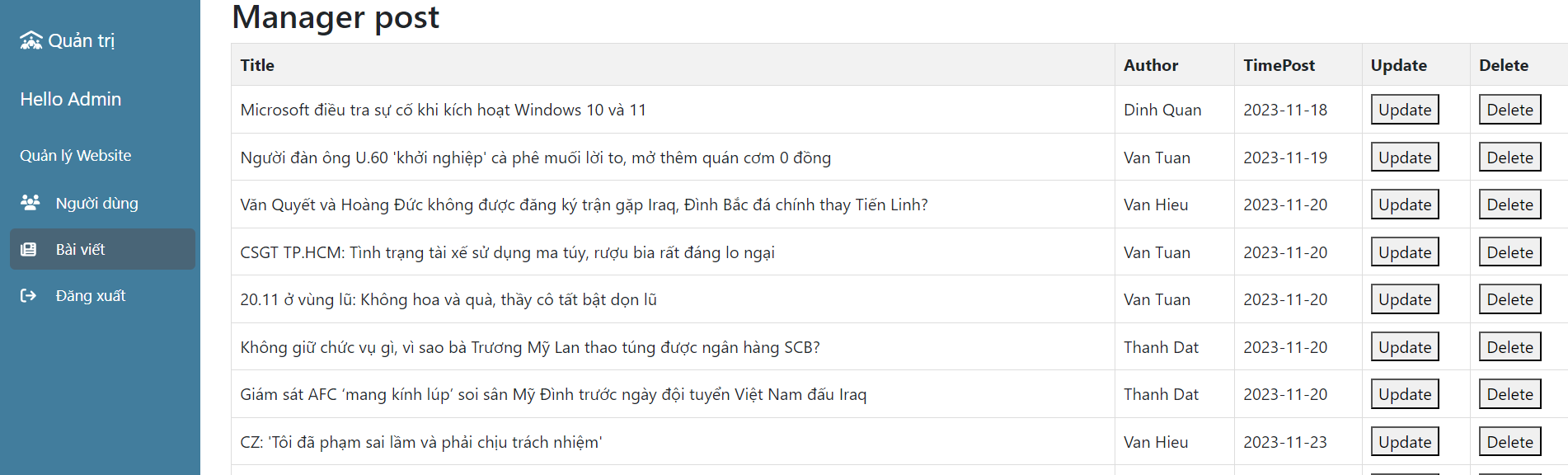
****

Hình 2. Giao diện quản lý người dùng

Phần giao diện dành cho các người dùng khi đã thực hiện đăng các bài viết và người dùng có thể quản lý trong phần này với các chức năng như: chỉnh sửa, xóa.

* + 1. **Giao diện quản lý của Admin**

****

****

Hình 2. Giao diện quản lý của Admin

Đây là 1 giao diện dành riêng cho Admin trong quá trình quản lý tài khoản người dùng và các bài đăng.

Được thiết kế dạng bảng để người dùng có thể dễ dang quan sát tìm kiếm người dùng khi thực hiện các chức năng quản lý.

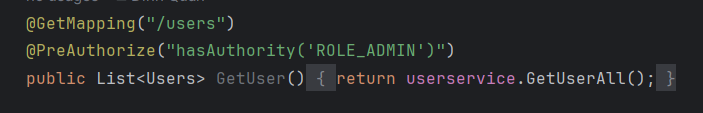
**CHƯƠNG 3: TRIỂN KHAI WEBSITE**

Để hệ thống có thể hoạt động thì chúng cần phải có sự tương tác giữa Front-end và Back-end. Bên phía Front-end trong hệ thống này chính là ReactJs sẽ xử lý xây dựng giao diện cho người dùng. Bên phía Back-end trong hệ thống này là Spring Boot dùng để xử lý các vấn đề như kết nối với cơ sở dữ liệu , Thêm xóa sửa bài viết…

Dưới đây là các giai đoạn sự tương tác giữa Front-end và Back-end.

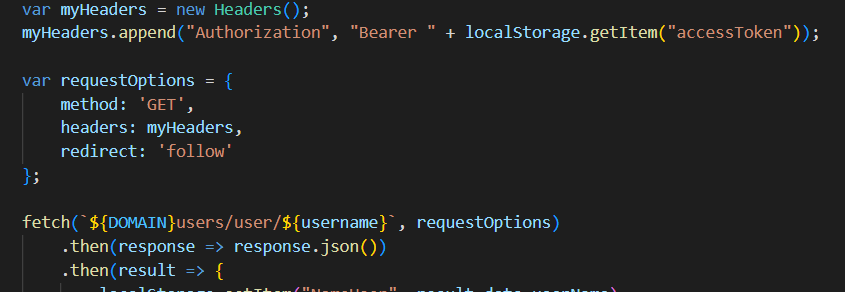
* 1. **Xác định api endpoints của backend và sử dụng http request từ frontend**

Backend cung cấp các API endpoints để frontend có thể giao tiếp và truy xuất dữ liệu từ server. Đây bao gồm các phương thức như GET, POST, PUT, DELETE để lấy dữ liệu, thêm, cập nhật, xóa dữ liệu.



Hình 3. Xác định api endpoints

Sau đó sẽ frontend tức là react sẽ sử dụng những api này để xử lý dữ liệu



Hình 3. Sử dụng http request fetch

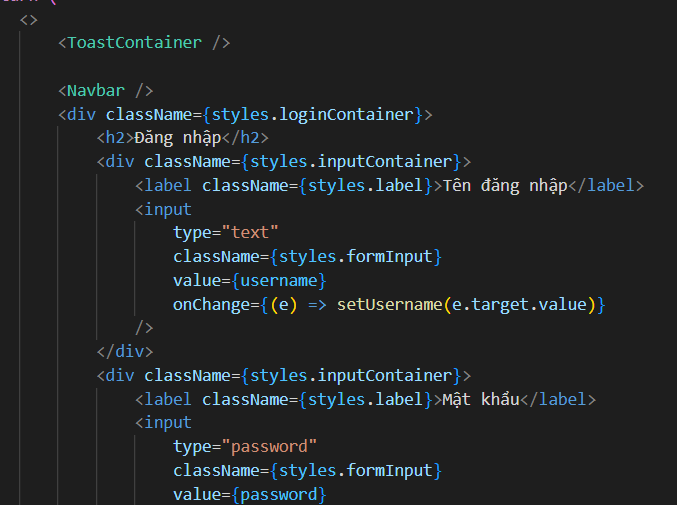
* 1. **Xử lý và hiển thị dữ liệu từ Backend**

Xử lý dữ liệu được trả về từ backend thông qua các request Http có thể được xử lý và hiển thị trên giao diện frontend. Sử dụng các state để quản lý và sử dụng dữ liệu được fetch.



Hình 3. Xử lý bằng useState

Sử dụng jsx để dùng dữ liệu tạo giao diện cho người dùng tương tác. Sử dụng boostrap và css module để hiển thị thẩm mỹ hơn.



Hình 3. Sử dụng Jsx

* 1. **Xử lý các sự kiện và tương tác của người dùng**

Các sự kiện như onclick hoặc một số event khác được xử lý bằng javascript .



Hình 3. Xử l ý sự kiện

**KẾT LUẬN**

Sau khi hoàn thành bài báo cáo về quá trình xây dựng website cũng như tìm hiểu về framework spring boot và reactjs, em nhận thấy đã vận dụng được những kiến thức để xây dựng được một trang web tương đối mà áp dụng cả 2 framework. Qua đó sau khi kết thúc kì thực tập tốt nghiệp này nhờ vào bài báo cáo em đã tự tin có thể vận dụng và phát triển kiến thức để có thể xây dựng những dự án lớn hơn.

Đây là những điểm còn hạn chế em tự nhận thấy sau khi hoàn thành bài báo cáo:

* Chưa thể sửa bài viết dành cho người dùng.
* Thiếu tính nắng tìm kiếm trong quản lý người dùng và bài viết.
* Giao diện chưa bắt mắt.
* Ngoài ra code cũng chưa được tối ưu vẫn còn xảy ra lỗi khi sử dụng.
* Chưa sử dụng tối ưu framework.

**Hướng phát triển**

Trong tương lai em phải xử lý các logic về chức năng cho hợp lý và tối ưu cấu trúc hệ thống tránh xảy ra các lỗi gây khó chịu cho người dùng. Đồng thời cải tiến thêm những tính năng hay hơn và thiết kế ứng dụng đẹp, sinh động và trực quan hơn/

Phát triển các tính năng như là có thể tự động thu nhỏ dựa trên các thiết bị khi sử dụng web trên trình duyệt.

Thêm các tính năng như thay đổi font chữ kích thước màu sắc cho người dùng túy ý về cách sử dụng khi đọc các bài đăng của mỗi cá nhân.

Cập nhật kiến thức liên quan đến framework đang xử dụng cũng như tối ưu code hơn.

**TÀI LIỆU THAM KHẢO**

* 1. 10/2023, <https://survey.stackoverflow.co/2023/>
  2. 10/2023, <https://www.jetbrains.com/lp/devecosystem-2022/java>
  3. 10/2023, <https://react.dev/learn>
  4. 9/2023, <https://spring.io/projects/spring-boot>
  5. 9/2023, <https://loda.me/courses/spring-boot>
  6. 9/2023, <https://spring.io/projects/spring-security>
  7. 9/2023, <https://www.geeksforgeeks.org/spring-boot-3-0-jwt-authentication-with-spring>