Shape, square

Description automatically generatedTRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT

**THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

---------------------------

Logo

Description automatically generated

**BÁO CÁO ĐỒ ÁN**

**MÔN HỌC:LẬP TRÌNH WEBSITE**

**ĐỀ TÀI: XÂY DỰNG WEBSITE THƯƠNG MẠI ĐIỆN TỬ QUẢN LÍ HỆ THỐNG BÁN ĐỒ THỂ THAO**

**Mã lớp học phần: WEPR330479\_22\_1\_08**

**GVHD: ThS. Trương Thị Khánh Dịp**

**Nhóm 25:**

**Đoàn Quốc Bảo MSSV:20110612**

**Hồ Thành Danh MSSV: 20110207**

TP Hồ Chí Minh, tháng 10 năm 2022

**MỤC LỤC HÌNH ẢNH**

Hình 1: Cách thức hoạt động của MySQL (Mô hình Client – server) 8

Hình 2: Kiến trúc của Spring Framework 12

Hình 3: Spring boot 17

Hình 4: Spring Security FilterChainProxy 21

Hình 5: Minh họa SecurityContext 22

Hình 6: Session-based authentication 24

Hình 7: Token-based Authentication via JWT 25

Hình 8: Ví dụ cấu hình ủy quyền bằng URL 27

Hình 10: Cấu trúc một JWT 29

Hình 11: Hibernate là gì 32

Hình 12: Kiến trúc JPA 33

Hình 13: Kiến trúc Hibernate 34

Hình 14: Một số ORM framework khác Hibernate 34

Hình 15: API là gì 36

Hình 16: So sánh độ phổ biến giữa ReactJS, AngularJS, VueJS 40

Hình 17: Cấu trúc Redux 41

Hình 18: Nguyên tắc vận hành của Redux 43

Hình 19: Ví dụ hàm useState 45

Hình 20: Ví dụ hàm useState 45

Hình 21: Ví dụ hàm useEffect 46

Hình 22: Ví dụ dùng useEffect tương tự chức ComponentDidMount 47

Hình 23: Tóm tắt hàm useEffect 47

Hình 24: Ví dụ hàm useReducer 48

Hình 25: Ví dụ hàm useMemo 48

Hình 26: Ví dụ có sử dụng hàm useMemo 49

Hình 27: Ví dụ hàm useCallBack 49

Hình 28: Chức năng của ứng dụng Web 52

Hình 30: Sơ đồ của hệ thống 72

**MỤC LỤC BẢNG BIỂU**

Bảng 1: So sánh SQL và MySQL 8

Bảng 2: Xếp hạng các engine database 9

Bảng 3: Mô tả chức năng 55

Bảng 4: Mô tả chức năng đăng nhập 57

Bảng 6: Mô tả chức năng cập nhật ảnh đại diện 57

Bảng 5: Mô tả chức năng xem trang cá nhân 59

Bảng 7: Mô tả chức năng đổi mật khẩu 59

Bảng 8: Mô tả chức năng giỏ hàng 60

Bảng 9: Mô tả chức năng xem và lọc danh sách sản phẩm 60

Bảng 10: Mô tả chức năng Thanh toán giỏ hàng 61

Bảng 11: Mô tả chức năng xem danh sách sản phẩm 61

Bảng 12: Mô tả chức năng đánh giá sản phẩm 62

Bảng 13: Mô tả chức năng quản lý người bán 65

Bảng 14: Mô tả chức năng thêm sản phẩm mới 65

Bảng 15: Mô tả chức năng xem và lọc danh sách sinh viên 66

Bảng 16: Mô tả chức năng xem danh sách đội nhóm 66

Bảng 17: Mô tả chức năng chỉnh sửa thông tin cá nhân 67

Bảng 18: Mô tả chức năng thích bài viết 68

Bảng 19: Mô tả chức năng phê duyệt hoặc từ chối duyệt đề xuất 68

Bảng 20: Mô tả chức năng giao việc 70

Bảng 21: Mô tả chức năng nhận hoặc từ chối công việc 71

**MỤC LỤC**

[**LỜI CẢM ƠN** 1](#_Toc121957410)

[**DANH MỤC VIẾT TẮT** 2](#_Toc121957411)

**LỜI CẢM ƠN**

Để có thể hoàn thành đồ án một cách hoàn chỉnh, bên cạnh sự nỗ lực cố gắng của các thành viên trong nhóm còn có sự hướng dẫn nhiệt tình của quý thầy cô, sự động viên ủng hộ của gia đình và bạn bè trong suốt thời gian học tập nghiên cứu và thực hiện đồ án.

Trước hết, nhóm em xin gửi lời cảm ơn chân thành nhất đến cô Mai Anh Thơ, vô cùng bận bịu nhưng vẫn luôn giành thời gian để giải đáp những thắc mắc cũng như đã hết lòng giúp đỡ, chỉ bảo, tạo mọi điều kiện tốt nhất cho nhóm em hoàn thành đồ án này.

Bên cạnh đó, cũng xin chân thành bày tỏ lòng biết ơn đến toàn thể quý thầy cô trong Khoa Công Nghệ Thông Tin, Trường Đại học Sư phạm Kỹ thuật TP.HCM đã tận tình truyền đạt những kiến thức quý báu cũng như tạo mọi điều kiện thuận lợi nhất cho nhóm em trong suốt quá trình học tập và cho đến khi thực hiện đồ án.

Với kiến thức và trình độ lý luận còn hạn chế, đồ án này không thể tránh khỏi những điều thiếu sót. Vì vậy nhóm em rất mong nhận được những ý kiến đóng góp, nhận xét và sự chỉ bảo của cô để nhóm em có thể học hỏi hêm những kiến thức và kinh nghiệm cho bản thân, phục vụ cho quá trình học tập sau này.

Xin chân thành cảm ơn!

**DANH MỤC VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| STT | Tên viết tắt | Tên đầy đủ |
| 1 | API | Application Programming Interface |
| 2 | HTTP | Hypertext Transfer Protocol |
| 3 | SQL | Structured Query Language |
| 4 | RDBMS | Relational Database Management System |
| 5 | JWT | Json Web Token |
| 6 | JPA | Java Persistence API |
| 7 | CSDL | Cơ sở dữ liệu |
| 8 | GPL | General Public License |
| 9 | MVC | Model View Controller |
| 10 | POJO | Plain old Java object |
| 11 | AOP | Aspect Oriented Programming |
| 12 | CSS | Cascading Style Sheets |
| 13 | HTML | Hypertext Markup Language |
| 14 | GWT | Google Web Toolkit |
| 15 | IoC | Inversion of control |
| 16 | JDO | Java Data Objects |
| 17 | STOMP | Streaming Text Oriented Messaging Protocol |
| 18 | JNDI | Java Naming and Directory Interface |
| 19 | CLI | Command Line Interface |
| 20 | SSO | Single Sign-On |
| 21 | LDAP | Lightweight Directory Access Protocol |
| 22 | CORS | Cross-origin resource sharing |
| 23 | ORM | Object Relational Mapping |
| 24 | CRUD | Create, Read, Update, Delete |
| 25 | SOAP | Simple Object Access Protocol |
| 26 | HTTP | HyperText Transfer Protocol |
| 27 | JSON | JavaScript Object Notation |
| 28 | SEO | Search Engine Optimization |
| 29 | VPS | Virtual Private Server |
| 30 | CI | Contnuous Integration |

**CHƯƠNG 1: PHẦN MỞ ĐẦU**

## **Tên đề tài**

Xây dựng Ứng dụng Web thương mại điện tử giúp tổ chức và quản lý bán quần áo sử dụng sử dụng Spring Boot và ReactJS.

## **Nhóm sinh viên thực hiện**

* + **Sinh viên 1:**
    - **Họ và tên:** Đoàn Quốc Bảo
    - **Mã số sinh viên:** 20110612
    - **Lớp:** 201102C
    - **Ngành:** Công nghệ thông tin
    - **Khoa:** Công nghệ thông tin
    - **Khóa:** K20
    - **SĐT:** 0358513566
  + **Sinh viên 2:**
    - **Họ và tên:** Hồ Thành Danh
    - **Mã số sinh viên:** 20110207
    - **Lớp:** 201102C
    - **Ngành:** Công nghệ thông tin
    - **Khoa:** Công nghệ thông tin
    - **Khóa:** K20
    - **SĐT:**

## **Lý do chọn đề tài**

Ngày nay công nghệ thông tin ngày càng phát triển và được ứng dụng trong nhiều lĩnh vực. Các hệ thống tổ chức và quản lý trong thương mại điện tử như quản lí cửa hàng kinh doanh online này càng là xu thế phát triển.

Vì thế một hệ thống quản lý kinh doanh online quần áo thể thao cần đáp ứng được các nhu cầu về thông tin sản phẩm, các thương hiệu, các danh mục xu hướng hiện nay.

Nhận thấy được mức độ cần trong công cuộc đổi mới, chuyển đổi số hiện hay thì việc kinh doanh online ngày càng phổ biến, nên em quyết định chọn đề tài: “Xây dựng ứng dụng Web quản lý kinh doanh quần áo thể thao online sử dụng Spring boot và ReactJS”.

## **Mục tiêu của đề tài**

* **Mục tiêu tổng quát:** Xây dựng ứng dụng Web thương mại điện tử quản lý kinh doanh quần áo thể thao online.
* **Mục tiêu cụ thể:**
  + Người quản lý có thể xây dựng một trang bán hàng gồm các danh mục, thương hiệu, sản phẩm theo ý muốn, quản lý hiệu quả, dễ dàng thống kê các hoạt động của trang bán hàng thông qua chức năng thống kê cửa hàng.
  + Thông tin cá nhân, đơn hàng được minh bạch trên hệ thống nội bộ, dễ dàng quản lý và tìm kiếm. Người dùng có thể liên lạc với người quản lí.

## **Nội dung và phạm vi nghiên cứu**

* **Nội dung nghiên cứu:**
  + Nghiên cứu quy trình xử lý các chức năng trong trang web quản lí bán hàng quần áo thể thao online.
  + Tìm hiểu và nghiên cứu về HTML, CSS, Javascript, ReactJS, xây dựng phần Frontend.
  + Nghiên cứu Spring Framework cụ thể là Spring MVC, Spring Boot, Spring Data JPA, nghiên cứu RESTful web service để xây dựng phần Backend.
  + Dùng postman để hỗ trợ kiểm tra API phía Backend
  + Nghiên cứu Axios để Frontend gọi đến API của Backend.
* **Phạm vi nghiên cứu:**
* Ứng dụng web thương mại điện tử quản lý cửa hàng kinh doanh quần áo thể thao.

## **Cơ sở khoa học và thực tiễn**

**Đề tài được xây dựng mới hoàn toàn, ứng dụng các kiến thức sau:**

* HTML, CSS, Javascript cơ bản
* Java
* ReactJS
* Spring Framework (Spring Boot, Spring Data JPA, Spring Security)
* Mô hình MVC
* RESTfull web service

**Giải quyết được bài toán tổ chức quản lý hệ thống thương mại điện tử kinh doanh quần áo thể thao**

## **Thời gian thực hiện**

Kéo dài suốt học kì 1 năm học 2022 – 2023.

## **Sản phẩm của đề tài**

Ứng dụng Web giúp tổ chức và quản lý hệ thống thương mại điện tử kinh doanh quần áo thể thao.

* **Hệ thống có 3 đối tượng chính là:**

Người dùng của hệ thống (người bán hàng) và người quản trị hệ thống (admin). Mỗi người sẽ đăng kí tài khoản mua hàng hoặc bán hàng (người bán hàng). Người dùng truy cập với chức năng bán hàng sẽ có quyền xem đơn hàng thêm, xóa chỉnh sửa thông tin sản phẩm và người dùng đăng nhập với chức năng mua hàng sẽ có chức năng mua hàng, xem giỏ hàng, thanh toán đơn hàng

**Đối với người dùng cho việc mua hàng:**

* Xem sản phẩm theo mức giá, theo tên sản phẩm , theo danh mục sản phẩm
* Thay đổi giao diện xem sản phẩm
* Đánh giá sản phẩm
* Lựa chọn sản phẩm và mua sản phẩm
* Xem thông tin lịch sử đặt hàng
* Xem thông tin đơn hàng
* Xem thông người bán đang theo dõi
* Xem thông tin cá nhân, chỉnh sửa thông tin cá nhân, địa chỉ nhận hàng, địa chỉ thanh toán, thuộc tính thanh toán.

**Đối với người dùng cho việc bán hàng:**

* Xem thông tin chi tiết đơn hàng
* Xem thông tin cá nhân
* Xem sản phẩm theo mức giá, theo tên sản phẩm , theo danh mục sản phẩm
* Chỉnh sửa, thêm, xóa sản phẩm
* Thay đổi giao diện xem sản phảm
* Xem thông tin sản phẩm đang bán chạy
* Xem các đơn đặt hàng

**Đối với người quản trị hệ thống (admin):**

* Xem sản phẩm theo mức giá, theo tên sản phẩm , theo danh mục sản phẩm
* Xem các lượt đánh giá sản phẩm
* Xem và cho phép các người bán hoạt động
* **Về quản lí danh mục:**
* Quản lý tất cả các danh mục sản phẩm để phục vụ cho việc tra cứu thông tin sản phẩm phù hợp.
* **Về quản lí sản phẩm:**
* Quản lí tất cả các sản phẩm để phục vụ cho việc mua hàng.
* **Về quản lí giỏ hàng:**
* Quản lí tất sản sản phâm đã thêm vào giỏ hàng

**CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT**

1. **MySQL**
2. **Tổng quan về MySQL**

MySQL là một hệ thống quản trị cơ sở dữ liệu mã nguồn mở (Relational Database Management System, viết tắt là RDBMS) hoạt động theo mô hình client-server, dựa trên ngôn ngữ truy vấn có cấu trúc (SQL) và được phát triển, phân phối, hỗ trợ bởi Tập Đoàn Oracle. RDBMS là một phần mềm hay dịch vụ dùng để tạo và quản lý các cơ sở dữ liệu (Database) theo hình thức quản lý các mối liên hệ giữa chúng.

MySQL là một trong số các phần mềm RDBMS. RDBMS và MySQL thường được cho là một vì độ phổ biến quá lớn của MySQL. Các ứng dụng web lớn nhất như Facebook, Twitter, YouTube, Google, và Yahoo! đều dùng MySQL cho mục đích lưu trữ dữ liệu. Kể cả khi ban đầu nó chỉ được dùng rất hạn chế nhưng giờ nó đã tương thích với nhiều hạ tầng máy tính quan trọng như Linux, macOS, Microsoft Windows, và Ubuntu.

1. **Các khái niệm liên quan**

**Database** là cơ sở dữ liệu, là một bộ sưu tập dữ liệu được tổ chức bày bản và thường được truy cập từ hệ thống máy tính hoặc tồn tại dưới dạng tập tin trong hệ quản trị cơ sở dữ liệu. Database còn có thể được lưu trữ trên thiết bị có chức năng ghi nhớ như: thẻ nhớ, đĩa cứng, CD…

Vai trò của Database vô cùng quan trọng trong một hệ thống. Nó chính là nơi lưu trữ dữ liệu, cũng có thể liên quan đến tài khoản và phân quyền của hệ thống. Vì vậy nên đã có rất nhiều hệ quản trị cơ sở dữ liệu ra đời, thậm chí có nhiều hệ thống đánh đổi sự phức tạp và chậm chạp để đổi lấy sự bảo mật.

**Mô hình client-server**

Diagram

Description automatically generated

Hình 1: Cách thức hoạt động của MySQL (Mô hình Client – server)

(Nguồn: mauwebsite.vn)

Cốt lõi của MySQL là máy chủ MySQL, xử lý tất cả các hướng dẫn CSDL hoặc các lệnh. Máy chủ SQL có sẵn như một chương trình riêng biệt để sử dụng trong môi trường mạng client – server. Nó như một thư viện có thể được nhúng hoặc liên kết vào các ứng dụng riêng biệt. Nhờ sự hoạt động riêng biệt này nên ban đầu MySQL được phát triển nhằm mục đích là xử lý CSDL lớn, có thể gửi dữ liệu đi nhiều vị trí mặc dù chỉ được cài đặt trên một máy. Người khác ở vị trí khác cũng có thể thông qua nhiều giao diện máy khách MySQL khác nhau để gửi request đến server lấy response.

**SQL** vừa đọc qua thì nhiều người lại cho rằng MySQL và SQL giống nhau, nhưng thật sự thì nó khác nhau hoàn toàn.

Bảng 1: So sánh SQL và MySQL

|  |  |  |
| --- | --- | --- |
| Tiêu chí | SQL | MySQL |
| Khái niệm | Là ngôn ngữ truy vấn có cấu trúc Structured Query Language được phát triển bởi Microsoft. Đây là một công cụ hữu ích để quản lý cơ sở dữ liệu quan hệ. | Là một RDBMS dùng để lưu trữ, truy vấn, sửa đổi và quản trị cơ sở dữ liệu. |
| Kiểu | Là một ngôn ngữ truy vấn. | Là phần mềm cơ sở dữ liệu sử dụng ngôn ngữ SQL để truy vấn cơ sở dữ liệu. |
| Nhiệm vụ | Truy vấn và vận hành hệ thống | Cho phép xử lý dữ liệu, sửa đổi, xóa hay lưu trữ theo định dạng bảng. |
| Hỗ trợ kết  nối | Không có trình kết nối | Cung cấp một công cụ tích hợp (MySQL workbench) để thiết kế và phát triển cơ sở dữ liệu. |
| Môi trường | Hoạt động tốt với .NET | Có thể kết hợp nhiều ngôn ngữ lập trình khác nhau, thông thường là PHP |
| Bảo mật | Công cụ bảo mật riêng với Microsoft Baseline Security Analyzer, giúp tăng tính bảo mật cho SQL một cách triệt để. | Các chuyên gia về cơ sở dữ liệu giúp MySQL có tính năng bảo mật, đảm bảo dữ liệu được bảo mật tuyệt đối. |
| Cập nhật phiên bản | Ngôn ngữ cố định, lệnh vẫn giữ nguyên. | Cập nhật thường xuyên. |

1. **Ưu nhược điểm**

Vì sao MySQL lại phổ biến đến vậy?

Theo như thống kê từ trang *db-engines.com (Năm 2022),* MySQL không phải là hệ quản lý cơ sở dữ liệu (RDBMS) duy nhất trên thị trường, nhưng nó đích thực phổ biến nhất và chỉ xếp sau Oracle Database khi xét đến những thông số chính như số lượng tìm kiếm, profile người dùng trên LinkedIn, và lượng thảo luận trên các diễn đàn internet.

Bảng 2: Xếp hạng các engine database

Graphical user interface, text, application, email

Description automatically generated

*(Nguồn: db-engines.com)*

Lý do chính vì sao rất nhiều ông lớn công nghệ phụ thuộc vào MySQL là gì?

Việc hiểu rõ ưu và nhược điểm sẽ giúp khai thác và sử dụng MySQL một cách tối ưu và hiệu quả nhất.

* **Ưu điểm:**
* *Miễn phí*: MySQL được phát hành theo giấy phép nguồn mở. Bởi vậy, không phải trả tiền để sử dụng nó.
* *Dễ sử dụng*: Nó hoạt động trên nhiều hệ điều hành với nhiều ngôn ngữ bao gồm Java, C, C++, PHP,… Bởi vậy, nó cung cấp một hệ thống các hàm tiện ích mạnh mẽ và tiện lợi.
* *Tốc độ nhanh chóng*:MySQL là hệ cơ sở dữ liệu dễ dùng, có tốc độ nhanh và hoạt động ổn định ngay cả với các tập dữ liệu lớn.
* *Hỗ trợ cơ sở dữ liệu lớn*:MySQL có thể hỗ trợ cơ sở dữ liệu lên tới 50 triệu hoặc nhiều hơn trong một bảng. Giới hạn kích thước tệp mặc định cho 1 bảng là 4GB nhưng có thể tăng hạn mức nếu hệ điều hành có xử lý được. Giới hạn lý thuyết có thể lên tới 8 triệu TB.
* *Chương trình mạnh mẽ*:MySQL là một chương trình mạnh mẽ theo đúng nghĩa. Nó có thể xử lý một tập hợp lớn các chức năng của các gói cơ sở dữ liệu mạnh mẽ và đắt tiền nhất.
* *Tính bảo mật cao*: Sở hữu nhiều tính năng bảo mật cấp cao. Bởi vậy, nó cực kỳ thích hợp cho các ứng dụng có truy cập cơ sở dữ liệu trên internet.
* *Đa tính năng*: MySQL hỗ trợ nhiều chức năng SQL được mong chờ từ một hệ quản trị CSDL quan hệ cả trực tiếp và gián tiếp.
* *Khả năng tùy biến cao*: Giấy phép GPL mã nguồn mở cho phép các lập trình viên sửa đổi phần mềm MySQL sao cho phù hợp với môi trường sử dụng của riêng họ.
* **Nhược điểm:**

Nhưng song song với đó, hệ thống quản trị cơ sở dữ liệu này cũng tồn đọng nhiều nhược điểm.

* *Độ tin cậy chưa cao*: Do các chức năng cụ thể được xử lý với MySQL (giao dịch, kiểm toán, tài liệu tham khảo,…) khiến cho nó kém tin cậy hơn so với một số hệ quản trị CSDL khác.
* *Giới hạn*: MySQL sẽ không làm tất cả và nó sẽ đi kèm một số hạn chế nhất định về chức năng mà một ứng dụng có thể cần đến.
* *Dung lượng hạn chế*: Nếu bản ghi của bạn lớn dần lên thì việc truy xuất dữ liệu sẽ khó khăn hơn. Khi đó, bạn phải áp dụng nhiều biện pháp nhằm tăng tốc độ truy xuất dữ liệu (ví dụ như: chia tải database ra nhiều server, tạo cache MySQL,…).

1. **Kết luận**

Bất kì một RDBMS nào thì cũng có ưu nhược điểm riêng của nó. Tuy nhiên đối với những ưu điểm kể trên thì MySQL rất đáng để ưu tiên lựa chọn. Đặc biệt là ưu điểm miễn phí khiến cho những người mới như em quyết định lựa chọn sử dụng.

1. **Spring Framework**
2. **Tổng quan về Spring Framework**

Là một lập trình viên java thì chắc chắn ai cũng đã ít nhất một lần được nghe nhắc qua các framework như là:

* Spring MVC – Model View Controller
* Struts
* Hibernate
* GWT – Google Web Toolkit
* Vaadin

Spring luôn là một thứ gì đó luôn được nhắc đến đầu tiên với vai trò là một Java Web Frameworks lâu đời nhất, nhưng như người ta nói: Old but Gold. Đến nay, nó vẫn được tận dụng và không ngừng biến đổi, phát triển thêm nhiều thay đổi cùng Java. Trong đó thì Spring MVC được nhắc đến chính là việc triển khai MVC cho các ứng dụng web được cung cấp bởi mô-đun MVC của khuôn khổ Spring.

1. **Kiến trúc của Spring Framework**



Hình 2: Kiến trúc của Spring Framework

Spring là một khuôn khổ mạnh mẽ cho các ứng dụng J2EE và lý do đằng sau sự thành công của nó là kiến ​​trúc của nó. Kiến trúc Spring bao gồm bảy mô-đun lớn và gần 20 module nhỏ. Nó giúp người dùng có thể lựa chọn module phù hợp với nhu cầu mà không bị thừa thãi.

**Core Container**

* TầngCore Container chứa Core, Bean, Context và Expression Language.
* Module Core cung cấp các thành phần cơ bản của Framework, bao gồm các tính năng IoC và Dependency Injection.
* Module Bean cung cấp BeanFactory là một implementation phức tạp của pattern factory.
* Module Context được xây dựng trên nền tảng vững chắc được cung cấp bởi các module Core và Bean và nó là một phương tiện để truy cập vào bất kỳ đối tượng được quy định và cài đặt trước. Interface ApplicationContext là tâm điểm của các module Context.
* Module SpEL cung cấp một ngôn ngữ biểu hiện mạnh mẽ để truy vấn và thao tác đối tượng đồ thị trong thời gian chạy.

**Data Access/Integration**

Tầng này chứa các module sử dụng cho việc truy cập dữ liệu / lớp Integration bao gồm các module JDBC, ORM, OXM, JMS và module Transaction có chi tiết như sau:

* Module JDBC cung cấp một lớp JDBC-trừu tượng loại bỏ sự cần thiết phải làm JDBC.
* Module ORM cung cấp các lớp integration cho các API mapping liên quan đến đối tượng, bao gồm cả JPA, JDO, Hibernate, và iBatis.
* Module OXM cung cấp một lớp trừu tượng có hỗ trợ việc triển khai lập mapping imlementations Object / XML cho JAXB, Castor, XMLBeans, JiBX và XStream.
* Module Java Messaging Service JMS chứa các tính năng cho tạo và xóa các message.
* Module Transaction hỗ trợ quản lý giao dịch chương trình và khai báo cho các lớp implement những interface đặc biệt và cho tất cả các POJOs.

**Web**

Tầng web sẽ bao gồm các module Web, Web-MVC, Web-Socket, và các module Web-Portlet có chi tiết như sau:

* Module Web cung cấp các tính năng tích hợp web cơ bản như chức năng upload tập tin nhiều phần dữ liệu và khởi tạo của các container IoC sử dụng để lắng nghe servlet và context ứng dụng web.
* Module Web-MVC chứa MVC của Spring cho các ứng dụng web.
* Module Web-Socket cung cấp hỗ trợ cho WebSocket, giao tiếp hai chiều giữa máy khách và máy chủ trong các ứng dụng web.
* Module Web-Portlet cung cấp việc xử lý MVC sẽ được sử dụng trong một môi trường portlet và mô phỏng các chức năng của module Web-Servlet.

**Miscellaneous**

Ngoài những module kể trên, thì những module quan trọng còn lại được nhóm lại với nhau thành một nhóm với nhau bao gồm AOP, Aspects, Instrumentation và Test có chi tiết như sau:

* Module AOP cung cấp thực hiện lập trình hướng khía cạnh cho phép xác định method-interceptors, pointcuts để tách mã thực hiện chức năng.
* Module Aspects cung cấp tích hợp với AspectJ - là một framework (AOP) lập trình hướng khía cạnh mạnh mẽ.
* Module Instrumentation cung cấp lớp hỗ trợ instrumentation và lớp loader instrumentation để sử dụng trong các máy chủ ứng dụng nhất định.
* Module Messaging cung cấp hỗ trợ cho STOMP (Streaming Text Oriented Messaging Protocol) như WebSocket sub-protocol để sử dụng trong các ứng dụng. Nó cũng hỗ trợ model chú thích cho việc định tuyến và xử lý tin nhắn STOMP từ WebSocket.
* Module Test hỗ trợ việc test các thành phần Spring với JUnit hoặc framework TestNG.

1. **Ưu nhược điểm của Spring Boot**

**Ưu điểm**

* Dependency Injection hoặc Inversion of Control được sử dụng để giúp các component tách rời, độc lập với nhau. Spring container sẽ giúp gắn kết những components này lại với nhau theo đặc tả business của hệ thống.
* Spring IoC container quản lý vòng đời của Spring Bean và các cấu hình của project chẳng hạn như JNDI lookup.
* Spring IoC container quản lý vòng đời của Spring Bean và các cấu hình của project chẳng hạn như JNDI lookup.
* Spring MVC framework được sử dụng cho phát triển ứng dụng web rất dễ dàng với việc hỗ trợ rất tốt các tính năng web services, json,…
* Hỗ trợ quản lý transaction, JDBC operations, File uploading, Exception Handling,… rất dễ dàng bằng cách cấu hình được rút gọn, thay vào đó là sử dụng annotation hoặc spring bean configuration file.
* Làm giảm đi sự phụ thuộc giữa các components khác nhau của ứng dụng, Spring IoC container làm nhiệm vụ khởi tạo resources hoặc beans và “tiêm – inject” chúng theo sự phụ thuộc khác nhau.
* Thực hiện unit test case rất dễ bởi vì business logic của bạn không có sự phụ thuộc trực tiếp. Việc thực hiện chỉ là viết test configuration và inject mock bean cho các mục đích test khác nhau.
* Thực hiện unit test case rất dễ bởi vì business logic của bạn không có sự phụ thuộc trực tiếp. Việc thực hiện chỉ là viết test configuration và inject mock bean cho các mục đích test khác nhau.
* Làm giảm đi khối lượng code rất nhiều, chẳng hạn như việc khởi tạo đối tượng, open/close các resources,…
* Spring framework chia thành nhiều module riêng biệt, do đó việc sử dụng các features trong Spring framework rất tự do… Ví dụ như ứng dụng không sử dụng tính năng transaction, thì không cần thiết phải thêm dependency này vào.

**Nhược điểm:**

Bên cạnh những ưu điểm được nhắc đến ở trên, Spring Framework cũng tồn tại một số nhược điểm có thể kể đến như:

* Độ phức tạp – Khung công tác Spring có rất nhiều biến và phức tạp. Do đó, bạn chỉ nên sử dụng nó nếu bạn có một đội ngũ lập trình viên có kinh nghiệm đã sử dụng framework này trước đó. Learning curve của bạn sẽ khó khăn, vì vậy nếu bạn hoặc nhóm của bạn không có nhiều kinh nghiệm phát triển, tốt hơn là bạn nên cân nhắc việc sử dụng nó.
* Cơ chế song song – Một trong những ưu điểm lớn nhất của Spring là nó cung cấp cho các nhà phát triển nhiều lựa chọn, nhưng đây cũng có thể là một nhược điểm vì nó gây ra sự nhầm lẫn. Các nhà phát triển phải biết tính năng nào sẽ hữu ích và việc đưa ra quyết định sai lầm có thể dẫn đến sự chậm trễ đáng kể.
* Không có nguyên tắc cụ thể – Trong tài liệu Spring không đề cập tới việc đối phó với các mối đe dọa như XSS hoặc giả mạo yêu cầu trên nhiều trang web. Với suy nghĩ này, bạn và nhóm của bạn sẽ cần phải tự tìm ra cách ngăn chặn tin tặc xâm nhập vào ứng dụng của bạn.

1. **Hệ sinh thái của Spring – Spring Ecosytem**

Spring framework sở hữu những ưu điểm nổi trội hướng tới những nghiệp vụ chuyên biệt của một khung phần mềm. Tuy nhiên, điều thực sự làm Spring Framework trở nên có giá trị là hệ sinh thái Spring Ecosystem phát triển mạnh mẽ trong những năm qua và vẫn đang được hỗ trợ để phát triển nhiều hơn trong tương lai.

* Từ cấu hình đến bảo mật, từ phát triển ứng dụng web (web apps) đến dữ liệu lớn (big data), bất kể nhu cầu cơ sở hạ tầng của ứng dụng là gì, đều có các Spring Project giúp bạn xây dựng nó.
* Spring được thiết kế theo mô-đun. Vậy nên lập trình viên có thể bắt đầu với quy mô nhỏ và chỉ sử dụng những module họ cần. Đến khi mở rộng hệ thống, họ cũng dễ dàng sử dụng các module anh em trong hệ sinh thái.
* Mặc dù có một danh sách dài các module hỗ trợ trong hệ sinh thái của Spring framework và nó liên tục thay đổi, nhưng có một số module đáng được nhắc tới mà bạn nên biết:
  + **Spring Boot**: Đưa ra một bộ thư viện để xây dựng ứng dụng Spring nhanh chóng và đáng tin cậy.
  + **Spring Cloud**: Cung cấp một tập hợp các công cụ hỗ trợ các hệ thống phân tán, hữu ích để xây dựng và triển khai các hệ thống microservice.
  + **Spring Security**: Bảo vệ ứng dụng của bạn với việc triển khai xác thực và ủy quyền có thể mở rộng.
  + **Spring Data**: Cung cấp một cách truy cập dữ liệu nhất quán để truy cập tới các cơ sở dữ liệu quan hệ SQL và NoSQL và hơn thế nữa.
  + **Spring REST Docs**: Cho phép lập trình viên mô tả các API RESTful bằng các tài liệu tự viết với các tài liệu chính xác được tạo tự động.
* Ngoài ra, hệ sinh thái của Spring Framework cũng có rất nhiều bên thứ 3 hỗ trợ. Nhờ vậy, việc sử dụng Spring Framework cũng rất tiện lợn và dễ dàng.

1. **Kết luận**

Vậy thì tại sao Spring Framework được sử dụng rộng rãi nhất?

Với bối cảnh được nêu ở trên, sẽ dễ dàng trả lời được câu hỏi “Lý do chọn Spring Framework để phát triển ứng dụng web” hay “Vì sao Spring là framework được sử dụng rộng rãi nhất?”.

Khi xác định được mối quan tâm cụ thể của bản thân để tìm ra một framework phù hợp với nhiều khía cạnh như web, truy cập dữ liệu, tích hợp nhiều dịch vụ,… Ta sẽ trả lời được Spring thực sự tỏa sáng ở đâu để đáp ứng yêu cầu của doanh nghiệp.

Spring luôn giữ được vị thế của mình từ lúc ra đời cho đến nay, Spring Framework có một lịch sử lâu dài về đổi mới, áp dụng và tiêu chuẩn hóa. Qua nhiều năm, nó đã đủ trưởng thành để trở thành giải pháp mặc định cho hầu hết các vấn đề phổ biến gặp phải trong quá trình phát triển các ứng dụng doanh nghiệp quy mô từ nhỏ đến lớn.

Điều thú vị hơn nữa là Spring đang được phát triển và duy trì tích cực. Hỗ trợ cho các tính năng ngôn ngữ mới và các giải pháp tích hợp doanh nghiệp đang được phát triển mỗi ngày. Cùng với đó là sự đóng góp không ngừng của cộng đồng spring khiến Spring đã phát triển lại càng phát triển hơn.

Song đó thì Spring còn có những dự án khác mang tầm ảnh hưởng lớn như là Spring MVC, Sping Security, Spring Boot.

1. **Spring Boot**
2. **Tổng quan về Spring Boot**

Diagram

Description automatically generated

Hình 3: Spring boot

Spring Boot là một framework dựa trên Java mã nguồn mở được giới thiệu bởi bởi Pivotal Software, Inc. Nó đã trở thành một giải pháp phổ biến để xây dựng cả triển khai Kho lưu trữ ứng dụng web (WAR) và các ứng dụng Java độc lập do quá trình phát triển hiệu quả về thời gian và không phức tạp.

Spring Boot nổi bật trong số các framework khác vì nó cung cấp cho các kỹ sư phần mềm cấu hình nhanh, xử lý hàng loạt, quy trình làm việc hiệu quả và nhiều công cụ khác nhau, giúp phát triển các ứng dụng dựa trên Spring mạnh mẽ và có thể mở rộng.

1. **Tại sao phải sử dụng Spring Boot**

Spring Boot là một mô-đun nằm trong Spring Framework nổi tiếng của Java. Sự ra đời của nó năm 2014 tuy muộn so với các mô-đun khác nhưng là một sáng kiến tuyệt vời của Spring Team, nó giúp giảm tải công việc cấu hình(configuration) của các lập trình viên cho việc phát triển dự án.

Spring là toàn bộ hệ sinh thái để phát triển Java bao gồm một số lượng lớn các mô-đun có sẵn như Spring MVC, Spring JDBC, Spring Security và các mô-đun khác.

Ngược lại, Spring Boot là một phần mở rộng của Spring, được sử dụng để xây dựng các ứng dụng sử dụng microservices. Nó tạo điều kiện và tăng tốc quá trình phát triển, làm cho nó hiệu quả hơn do các tính năng cốt lõi của nó.

**Tự động cấu hình**

Tự động cấu hình là một cách tiếp cận Spring Boot cốt lõi để giảm số lượng hành động mà nhà phát triển phải thực hiện. Nó tự động cấu hình một ứng dụng Spring dựa trên các phụ thuộc bạn đã thêm trước đó. Tự động cấu hình Spring Boot cung cấp các tính năng mặc định mạnh mẽ đồng thời tính linh hoạt cao.

**Opinionated Dependencies**

Opinionated có nghĩa là Spring Boot tự quyết định tập hợp các bean được định cấu hình mặc định mà bạn có thể ghi đè nếu cần. Hơn nữa, framework này chọn các gói để cài đặt cho các phần dependencies mà ta có thể cần. Bằng cách này, các nhà phát triển Spring Boot bắt đầu xây dựng ứng dụng của họ ngay lập tức, tập trung nhiều hơn vào logic nghiệp vụ vì phần lớn công việc được thực hiện bởi chính framework.

**Embedded Servers**

Máy chủ nhúng là một phần của ứng dụng. Điều đó có nghĩa là ta không phải cài đặt trước nó trong môi trường triển khai. Spring Boot cung cấp một máy chủ nhúng Tomcat theo mặc định, nhưng ta có thể thay đổi nó thành Jetty hoặc Undertow. Máy chủ được nhúng giúp triển khai hiệu quả hơn và giảm thời gian khởi động lại ứng dụng.

**Độc lập**

Spring Boot cho phép các nhà phát triển thiết lập và chạy các ứng dụng Spring độc lập, cấp sản xuất mà không cần triển khai chúng lên máy chủ web. Để chạy một ứng dụng Java, bạn cần phải đóng gói ứng dụng đó, chọn, tải xuống và định cấu hình máy chủ web cũng như tổ chức quá trình triển khai. Ngược lại, một ứng dụng Java Spring Boot chỉ cần đóng gói, sau đó nó đã sẵn sàng để chạy, sử dụng các lệnh đơn giản.

1. **Ưu nhược điểm của Spring Boot**

Dưới đây là những ưu và nhược điểm của Spring Boot framework sẽ giúp bạn quyết định xem công cụ phát triển này có đáp ứng được nhu cầu của bạn hay không.

**Ưu điểm**

Spring Boot được thiết kế để giúp các kỹ sư phần mềm đẩy nhanh quá trình phát triển, cho phép họ loại bỏ việc thiết lập và cấu hình ban đầu tốn thời gian của môi trường triển khai. Các lợi ích chính của Spring Boot bao gồm:

* Phát triển các ứng dụng dựa trên Spring một cách tiết kiệm thời gian và dễ dàng.
* Tự động cấu hình tất cả các components cho một ứng dụng Spring cấp sản xuất.
* Các máy chủ nhúng được tạo sẵn (Tomcat, Jetty và Undertow), dẫn đến việc triển khai ứng dụng được tăng tốc và hiệu quả hơn.
* Điểm cuối HTTP, cho phép nhập các tính năng bên trong ứng dụng như chỉ số, tình trạng sức khỏe, v.v.
* Không có cấu hình XML.
* Nhiều lựa chọn bổ sung, hỗ trợ nhà phát triển làm việc với cơ sở dữ liệu được nhúng và trong bộ nhớ.
* Dễ dàng truy cập cơ sở dữ liệu và các dịch vụ hàng đợi như MySQL, Oracle, MongoDB, Redis, ActiveMQ và các dịch vụ khác.
* Tích hợp trơn tru với hệ sinh thái Spring.
* Cộng đồng lớn và rất nhiều hướng dẫn, tạo điều kiện cho giai đoạn làm quen.

Một số ưu điểm khác

* Giảm thiểu thời gian phát triển code, tăng hiệu suất phát triển chung của cả dự án.
* Dễ dàng tích hợp các mô-đun liên quan như Sping-MVC, Spring Data, Spring Sercurity, Spring Cloud,v.v…
* Nó cung cấp các HTTPs servers như Tomcat, Jety,.. để phát triển, kiểm thử, deploy một cách dễ dàng.
* Cung cấp công cụ CLI(Command Line Interface) cho việc phát triển và test ứng dụng nhanh chóng từ command line
* Ngoài ra còn có nhiều plugins để phát triển nhanh chóng bằng các công cụ như Build như Maven hoặc Gradle.

**Nhược điểm**

* ***Thiếu kiểm soát***: Do style cố định, Spring Boot tạo ra nhiều phụ thuộc không được sử dụng dẫn đến kích thước tệp triển khai lớn.
* *Quá trình chuyển đổi*: Dự án Spring cũ hoặc hiện có thành các ứng dụng Spring Boot nhiều khó khăn và tốn thời gian.
* ***Không thích hợp cho các dự án quy mô lớn***: Hoạt động liên tục với các microservices, theo nhiều nhà phát triển, Spring Boot không phù hợp để xây dựng các ứng dụng nguyên khối.

1. **Kết luận**

Spring Boot đã trở thành một yếu tố không thể thiếu của hệ sinh thái Java, cung cấp một bộ công cụ hiệu quả và có thể mở rộng để xây dựng các ứng dụng Spring với kiến trúc microservices. Do thiết lập mặc định cho các bài kiểm tra đơn vị và tích hợp, nó cho phép các nhà phát triển tận hưởng quá trình triển khai và phát triển được tăng tốc. Hơn nữa, Spring Boot giúp các nhà phát triển xây dựng các ứng dụng mạnh với cấu hình rõ ràng và an toàn mà không tốn nhiều thời gian và công sức để có thêm kiến thức về Spring.

Để quyết định xem giải pháp phát triển Java này có đáp ứng các yêu cầu dự án hay không, hãy xem những lợi ích và hạn chế của Spring Boot, các tính năng cốt lõi của nó và điều chỉnh chúng phù hợp với mục tiêu của bạn. Bằng cách này, ta sẽ có thể giải tỏa mọi nghi ngờ và chọn giải pháp tốt nhất cho công ty của mình.

1. **Spring Security**
2. **Tổng quan**

Spring Security là một trong những core feature quan trọng của Spring Framework, nó giúp chúng ta phân quyền và xác thực người dùng trước khi cho phép họ truy cập vào các tài nguyên của chúng ta.

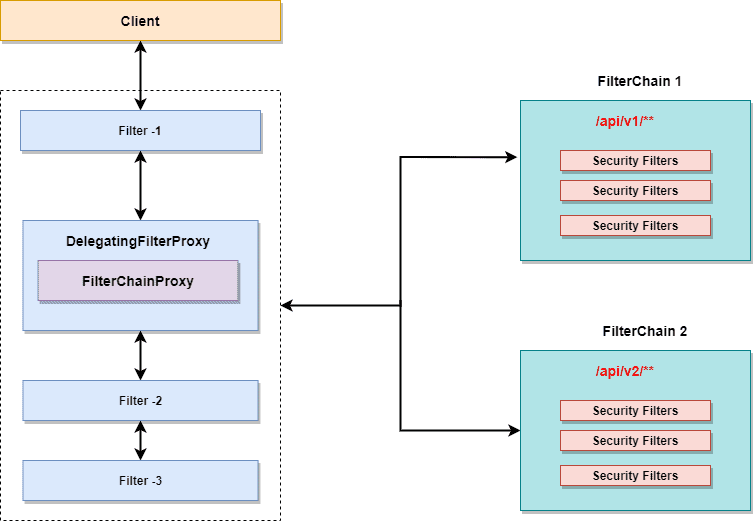
Khi nói đến các ứng dụng cần bảo mật, thì trước tiên cần biết đến hai khái niệm: Authentication và Autorization. Hiểu đơn giản là Xác thực và Phân quyền.

Spring Security rất mạnh mẽ và có khả năng tùy biến cao, lập trình viên có thể sử dụng cấu hình có sẵn của framework hoặc tùy chỉnh theo từng bài toán của hệ thống.

1. **Cơ chế hoạt động chính**

Kiến trúc của Spring Security trong các ứng dụng web hoàn toàn dựa trên các Servlet Filter, nó không phụ thuộc sử dụng servlet cũng như không phụ thuộc vào bất kì một công nghệ web nào.

Về cơ bản, Spring Security gồm nhiều filter khác nhau, được quản lý thông qua FilterChainProxy. Các filter này có trách nhiệm chặn yêu cầu, kiểm tra xác thực, kiểm tra uỷ quyền, điều hướng tới Controller để xử lý request. Sau đó ném ra ngoại lệ trong trường hợp thông tin xác thực không hợp lệ.



Hình 4: Spring Security FilterChainProxy

Spring Security cung cấp class SecurityContext, quản lý tất cả thông tin xác thực của ứng dụng. Thông tin xác thực của người dùng sẽ được lưu trong đối tượng Authentication, và được lấy thông qua SecurityContextHolder. SecurityContext sử dụng ThreadLocal để quản lý thông tin, do đó đảm bảo các phương thức thực thi trên cùng một luồng sẽ được chia sẻ thông tin xác thực như nhau.

Graphical user interface, text, application

Description automatically generated

Hình 5: Minh họa SecurityContext

1. **Về cơ chế xác thực**

Xác thực là gì?

Hiểu một cách đơn giản, xác thực là quá trình hệ thống nhận dạng và xác minh một người nào đó khi họ muốn truy cập tài nguyên của hệ thống. Nó giống như lúc chúng ta muốn vào facebook, cần đăng nhập username và password của mình trước tiên. Nếu thông tin đăng nhập đúng, hệ thống sẽ tiếp tục xử lý yêu cầu. Nếu không, sẽ không có quyền truy cập vào ứng dụng.

Spring hỗ trợ xác thực như thế nào?

Trước khi truy cập các tài nguyên giới hạn quyền của hệ thống, người dùng phải đăng nhập. Spring Security có cơ chế xử lý đăng nhập cho các bài toán từ đơn giản tới phức tạp, ví dụ:

* Thông tin đăng nhập được quản lý bởi chính hệ thống;
* Thông tin đăng nhập được quản lý bởi bên thứ 3;
* Cơ chế xác thực SSO (Oauth2, SAML);
* Cơ chế xác thực LDAP;
* JWT (Json Web Token)

Spring Security cung cấp interface AuthenticationProvider để xử lý các bài toán nêu trên một cách đơn giản, hiệu quả, an toàn và dễ mở rộng. Lập trình viên có thể sử dụng các phương thức có sẵn của Spring Security, hoặc là triển khai các tuỳ chỉnh AuthenticationProvider cho từng bài toán cụ thể.

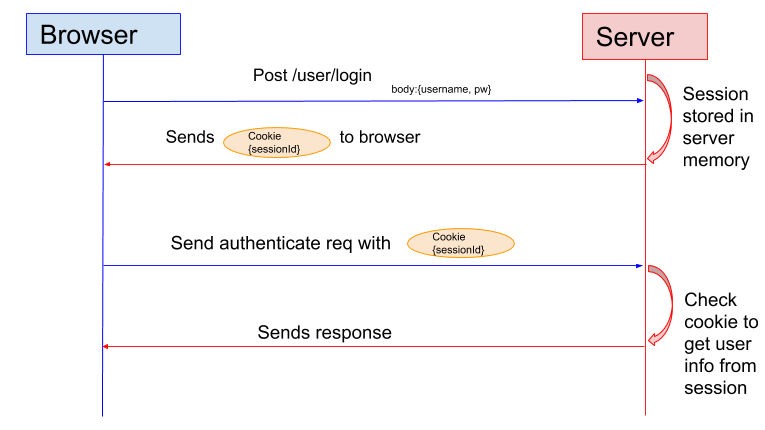
1. **Trạng thái và phi trạng thái (Stateful vs Stateless)**

Trong các ứng dụng web, việc tương tác giữa client và server có 2 mô hình: trạng thái và phi trạng thái.

**Trạng thái** nghĩa là server lưu thông tin của client để có thể xác định danh tính của client. Điển hình là giống như Session trong Tomcat của Java. Cơ chế xác thực theo mô hình này này được gọi là xác thực dựa trên phiên (session-based authentication).

Ví dụ, sau khi người dùng đăng nhập, ứng dụng lưu thông tin của người dùng trong phiên server và cung cấp cho client một giá trị cookie ghi lại phiên tương ứng. Với các yêu cầu tiếp theo, client đính kèm giá trị cookie đó (bước này là do trình duyệt tự động hoàn thành). Server sẽ đối chiếu cookie với session để có thể xác định được được yêu cầu đó là từ đâu và do ai.

Xác thực dựa trên phiên là đơn giản và thuận tiện nhất. Nó phù hợp với các dự án quy mô nhỏ. Tuy nhiên mô hình này khiến server phải lưu trữ một lượng lớn dữ liệu để quản lý phiên, cũng khi gặp khó khăn trong các trường hợp cần mở rộng ứng dụng theo cụm cũng như khó đáp ứng lượng người dùng đồng thời lớn



Hình 6: Session-based authentication

*(Nguồn: sherryhsu.medium.com)*

Spring Security hỗ trợ rất tốt cơ chế xác thực dựa trên phiên nêu trên. Ngoài ra Spring Security cũng hỗ trợ các thẻ tùy chỉnh với các template engine của Java như Thymeleaf, JSP,... Trong trường hợp triển khai ứng dụng theo cụm, Spring Security cũng có thể kết hợp với Spring Session để quản lý phiên phân tán.

**Phi trạng thái** hiểu đơn giản là server không lưu trữ thông tin của client. Các yêu cầu từ client tới server là độc lập với nhau. Để server có thể xác định được những yêu cầu này đến từ đâu và do ai, client sẽ phải đính kèm thông tin mô tả cho các yêu cầu đó.

Cơ chế xác thực dựa trên mô hình phi trạng thái còn được gọi là xác thực dựa trên mã thông báo (token-based authentication).

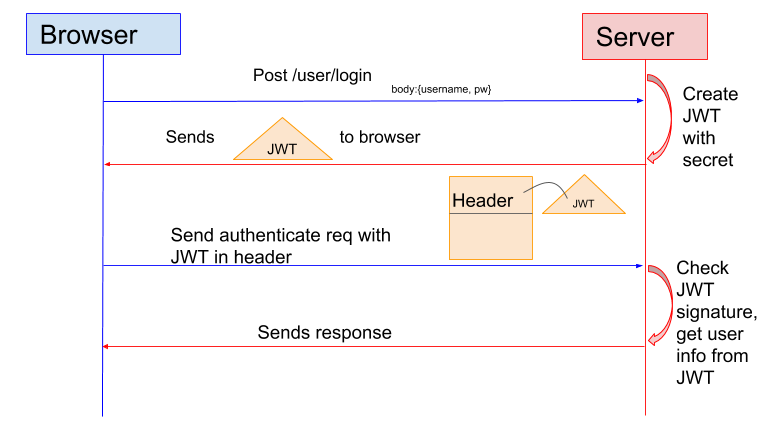
Ví dụ, bước đầu tiên, người dùng gửi username/password đến server để xác thực. Nếu thông tin chính xác, server sẽ mã hóa thông tin và xử lý, sau đó trả về một mã thông báo cho client. Client sẽ lưu trữ mã thông báo, với mỗi yêu cầu sau đó, client đính kèm mã thông báo đó trên header. Server sẽ xử lý xác thực dựa trên mã thông báo đó.

Cơ chế xác thực qua mã thông báo khắc phục được những nhược điểm của cơ chế xác thực qua phiên. Giảm một lượng lớn dữ liệu cần lưu trữ ở server, phù hợp với các hệ thống sử dụng RESTful API, dễ dàng mở rộng và phân cụm. Tuy nhiên phương pháp này cũng cần xử lý phức tạp hơn.

Có một số kiểu mã thông báo phổ biến như:

**Basic token:** Khi người dùng đăng nhập, server sẽ trả về cho client mã thông báo. Server sẽ lưu mã thông báo trong database hoặc cache storage. Với mỗi request gửi kèm token, server sẽ kiểm tra token đó còn hiệu lực hay không và đang được gắn với người dùng nào trong database. Với kiểu này, server vẫn phải hoạt động khá nhiều.

**Json web token (JWT):** Giống như basic token, server sẽ trả về mã JWT cho client khi người dùng đăng nhập. Nhưng thay vì lưu ở cả server và client, JWT này sẽ lưu trữ ở client, và được server giải mã thông qua secret key. JWT có thể được sử dụng như một cơ chế xác thực không yêu cầu database.



Hình 7: Token-based Authentication via JWT

*(Nguồn: Token-based Authentication via JWT)*

Spring cung cấp thư viện JWT cũng như các thư viện mã hoá phục vụ cho cơ chế xác thực dựa trên mã thông báo. Với việc xử lý mã thông báo, để đảm bảo việc xử lý chỉ diễn ra một lần duy nhất cho từng yêu cầu, Spring cung cấp filter OncePerRequestFilter để lập trình viên có thể triển khai cơ chế trên một cách nhanh chóng và chính xác.

1. **Cơ chế uỷ quyền: Authorization**

**Ủy quyền là gì?**

Nếu xác thực là quá trình kiểm tra người dùng hợp lệ thì ủy quyền là quá trình kiểm tra người dùng có được phép tiếp cận tài nguyên hay không. Đối với ứng dụng đơn giản, người dùng chỉ cần đăng nhập là có thể truy cập hết tài nguyên của ứng dụng. Tuy nhiên, với các ứng dụng lớn và phức tạp, dữ liệu chia theo từng nhóm người dùng với nhiều vai trò khác nhau. Ủy quyền giúp đảm bảo kiểm soát truy cập các tài nguyên an toàn và bảo mật hơn.

Ví dụ trong một ứng dụng doanh nghiệp, tài khoản của mỗi phòng ban sẽ chỉ có thể quản lý thông tin của phòng ban đó. Có những tài khoản chỉ được cấp quyền xem và không được thêm, sửa, xóa thông tin. Có những tài khoản được cấp quyền thao tác tất cả.

**Phân quyền ứng dụng web với Spring Security**

Spring Security hỗ trợ rất tốt trong việc phân quyền ứng dụng, cả ở 2 cấp độ: request level và method invocation level.

Spring Security cung cấp interface GrantedAuthority để đại diện cho các quyền của người dùng. Nếu một hệ thống phức tạp, cần kiểm tra ủy quyền dựa trên nhiều yếu tố khác, Spring Security cũng cung cấp thêm đối tượng User để lập trình viên có thể tùy chỉnh theo ý muốn.

A picture containing table

Description automatically generated

Hình 8: Ví dụ cấu hình ủy quyền bằng URL

Spring Security cung cấp tính năng ủy quyền ở cấp độ request bằng việc kế thừa class WebSecurityConfigurerAdapter. Như ví dụ trên, các url bắt đầu với */api/auth/* hoặc ***/****api/get-image/* hoặc ***/****api/test/* thì không giới hạn quyền, trong khi các url khác thì phải được xác thực. Việc cấu hình khá dễ hiểu và ngắn gọn. Tuy nhiên tùy vào nhu cầu phân quyền của hệ thống theo chức năng hay theo nhóm người dùng, mà chúng ta sẽ có những cách tùy biến khác. Chẳng hạn như đối với một hệ thống phức tạp, việc kiểm soát truy cập đến tận cấp độ phương thức (method) là điều cần thiết cho nên Spring cũng đã hỗ trợ ta thêm một số Annontation: @PreAuthorize, @PostAuthorize để áp dụng tính năng bảo mật ở cấp độ Class và Method.

Phạm vi ảnh hưởng của annotation phụ thuộc vào vị trí mà nó được đánh dấu. Nếu đánh dấu các annotation ở đầu class, nó sẽ áp dụng cho tất cả các phương thức của class đó. Trong trường hợp sử dụng riêng lẻ, nó chỉ ảnh hưởng với phương thức được đánh dấu. Đối với các phương thức bên trong annotation, lập trình viên có thể sử dụng các phương thức có sẵn của Spring Security, cũng như có thể tùy chỉnh theo các bài toán cụ thể.

Ngoài các tính năng chính nêu trên, Spring Security còn hỗ trợ nhiều tính năng khác như:

* Hỗ trợ nhiều password encoder.
* Dễ dàng xử lý các exceptions.
* Cấu hình CSRF, CORS đơn giản.
* Tương thích dễ dàng với các template engine của Java như Thymeleaf.

1. **Kết luận**

Spring security còn rộng lớn rất và nhiều thứ hay ho, chúng ta đều có thể tùy chỉnh để đạt được mục đích của từng bài toán cụ thể, có thể xử lý lỗi theo ý muốn, có thể phần quyền theo từng chức năng, phân quyền theo từng nhóm người.

Với Spring Security, lập trình viên có thể tùy chỉnh bảo mật cho hệ thống của mình theo cách đơn giản nhất, thông qua các cấu hình và annotation. Điều này giúp lập trình viên có thể dễ dàng tiếp cận những khía cạnh bảo mật, rút ngắn thời gian phát triển cũng như chi phí vận hành và bảo trì hệ thống.

1. **JWT (Json Web Token)**
2. **Tổng quan**

JWT là gì mà tại sao lại được nhắc đến việc xác thực cùng với Spring Security

JWT về mặt kỹ thuật là một cơ chế để xác minh chính chủ một số dữ liệu JSON. Nó là một chuỗi biến đổi, có thể chứa một lượng dữ liệu không giới hạn (không giống như một cookie) và nó đã được mã hóa bằng chữ ký.

Khi một máy chủ nhận được JWT, nó có thể đảm bảo dữ liệu mà nó chứa có thể được tin cậy bởi vì nó đã được xác thực với chữ ký đã được lưu trữ. Không yếu tố trung gian nào có thể sửa đổi JWT sau khi nó được gửi.

Điều quan trọng cần lưu ý là JWT đảm bảo quyền sở hữu dữ liệu nhưng không mã hóa; bất kỳ ai cũng có thể xem dữ liệu JSON chúng ta lưu trữ trong JWT một khi họ có được JWT đấy, vì nó chỉ được tuần tự hóa, không được mã hóa. Vì lý do này, nên sử dụng JWT với HTTPS. JWT là một công nghệ tuyệt vời để xác thực API và ủy quyền từ máy chủ đến máy chủ. Nó không phải là một lựa chọn tốt cho các session.

1. **Ba thành phần chính của JWT**

JWTs có thể được ký bằng một thuật toán bí mật (với thuật toán HMAC) hoặc một public / private key sử dụng mã hoá RSA.

**Một ví dụ về JWT Token:**

eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbkBnbWFpbC5jb20iLCJpYXQiOjE2NjA1ODc5OTQsImV4cCI6MTY2MTE5Mjc5NH0.qaxpkjNudHL\_B9017B8fPxZsr5r8lTkNO1SPa31ZC0wzVK4s3byboT4T8oHwJRiL1EIfHN6Efxh5XmzXv9-vOg

Thoạt trông phức tạp là thế nhưng nếu hiểu, cấu trúc của một JWT chỉ đơn giản như sau:

A picture containing graphical user interface

Description automatically generated

Hình 10: Cấu trúc một JWT

Nói một cách khác, JWT là sự kết hợp (bởi dấu .) một Object Header dưới định dạng JSON được encode base64, một payload object dưới định dạng JSOn được encode base64 và một Signature cho URI cũng được mã hóa base64 nốt.

**Header**

Header bao gồm hai phần chính: loại token (mặc định là JWT - Thông tin này cho biết đây là một Token JWT) và thuật toán đã dùng để mã hóa (HMAC SHA256 - HS256 hoặc RSA).

Shape

Description automatically generated with medium confidence

**Payload**

**Payload chứa các claims**. Claims là một các biểu thức về một thực thể (chẳng hạn user) và một số metadata phụ trợ. Có 3 loại claims thường gặp trong Payload: reserved, public và private claims.

**Reserved claims**: Đây là một số metadata được định nghĩa trước, trong đó một số metadata là bắt buộc, số còn lại nên tuân theo để JWT hợp lệ và đầy đủ thông tin: iss (issuer), iat (issued-at time) exp (expiration time), sub (subject), aud (audience), jti (Unique Identifier cho JWT, Can be used to prevent the JWT from being replayed. This is helpful for a one time use token.) ... Ví dụ:

Graphical user interface, text, application

Description automatically generated

Hoặc ta có thể thay đổi

A picture containing shape

Description automatically generated

**Signature**

Chữ ký Signature trong JWT là một chuỗi được mã hóa bởi header, payload cùng với một chuỗi bí mật theo nguyên tắc sau:

A picture containing application

Description automatically generated

Do bản thân Signature đã bao gồm cả header và payload nên Signature có thể dùng để kiểm tra tính toàn vẹn của dữ liệu khi truyền tải.

1. **Xử dụng JWT để xác thực API**

Ứng dụng nhiều nhất của JWT Token, và mục đích duy nhất bạn nên sử dụng JWT là dùng nó như một cơ chế xác thực API.

Điều này hiện nay là quá phổ biến và được sử dụng rộng rãi, kể cả Google cũng sử dụng JWT để xác thực các APIs của họ.

1. **Vô hiệu hoá 1 single token**

Làm thế nào bạn có thể làm mất hiệu lực môt single token? Một cách đơn giản là thay đổi secret key của server, làm mất hiệu lực tất cả các token. Nhưng làm vậy sẽ khiến khách hàng choáng váng, không hiểu vì sao mà token của họ “toang” như vậy.

Một cách khác để xử lý vấn đề này là thêm yếu tố thời gian vào đối tượng người dùng. Mã token tự động lưu trữ giá trị created at trong thuộc tính iat. Mỗi lần bạn kiểm tra token, bạn chỉ cần so sánh giá trị created at với giá trị thời gian ở đối tượng user.

Để vô hiệu hóa token, chỉ cần cập nhật giá trị phía máy chủ và nếu iat cũ hơn, bạn có thể reject token.

1. **Lưu trữ JWTs an toàn**

JWT cần được lưu trữ ở nơi an toàn bên trong trình duyệt của người dùng.

Có thể lưu ở localStorage hoặc session. Nhưng vì độ phức tạp nên nếu như hệ thống không thật sự cần quá nhiều bảo mật thì chúng ta vẫn có thể lưu chúng ở localStorage

Nếu bạn lưu trữ nó bên trong localStorage, nó có thể truy cập bằng bất kỳ script nào trong trang của chúng ta (điều này thực sự tệ, vì một cuộc tấn công XSS có thể cho phép kẻ tấn công bên ngoài lấy được token).

Khi lưu token trong localStorage (hoặc sessionStorage). Nếu bất kỳ mã script bên thứ ba nào bạn đưa vào trang của mình bị xâm phạm, nó có thể truy cập tất cả các token của người dùng.

JWT muốn an toàn thì cần được lưu trữ bên trong httpOnly cookie, một loại cookie đặc biệt mà chỉ có thể gửi trong các yêu cầu HTTP đến server và nó không bao giờ có thể truy cập (cả để đọc hoặc viết) từ JavaScript chạy trên trình duyệt.

Vì JWT đã được ký, người nhận có thể chắc chắn rằng khách hàng thực sự là người mà họ nghĩ.

1. **Sử dụng JWT cho xác thực SPA**

JWTs có thể được sử dụng như một cơ chế xác thực không cần đến cơ sở dữ liệu. Server có thể tránh sử dụng cơ sở dữ liệu vì dữ liệu lưu trữ trong JWT được gửi cho khách hàng là an toàn.

1. **Sử dụng JWT để uỷ quyền hoạt động trên các máy chủ**

Giả sử bạn đăng nhập vào một máy chủ, SERVER1, sau khi đăng nhập thành công thì được chuyển hướng đến một máy chủ khác SERVER2 để thực hiện một số xử lý.

SERVER1 có thể cấp cho bạn một JWT để ủy quyền cho phép bạn kết nối đến SERVER2. Hai máy chủ không cần chia sẻ cùng một session hoặc bất cứ điều gì để xác thực bạn. Sử dụng token để xử lý trường hợp này là rất hoàn hảo.

1. **Hibernate**
2. **Tổng quan**

Hibernate đến thời điểm tại có thể xem là một ORM có tầm ảnh hưởng lớn trong cộng đồng Java. Đóng vai trò là một ORM framework đi đầu trong lĩnh vực, đa số các dự án đều tin cậy và sử dụng Hibernate.

Nó dùng để thực hiện mapping cơ sở dữ liệu quan hệ sang các Object trong phần ngôn ngữ hướng đối tượng.

Hibernate là một Framework cho persistence layer thực hiện giao tiếp giữa các tầng dữ liệu với các phần ứng dụng như kết nối, truy xuất, lưu trữ...

Cái tên Hibernate được ra mắt vào năm 2001 do nhà sáng lập có tên Gavin King đưa ra. Nó giống như một sự thay thế cho EJB2 kiểu thực hể bean.

Hibernate Framework được biết đến là một giải pháp ORM (Object Relational Mapping) mã nguồn mở và gọn nhẹ. Nó hỗ trợ việc đơn giản hóa sự phát triển của ứng dụng Java nhằm thực hiện tương tác với cơ sở dữ liệu. Bên cạnh đó, Tool ORM giúp bạn đơn giản hóa việc tạo ra dữ liệu, thao tác dữ liệu cũng như truy cập dữ liệu. Đây là một trong những kỹ năng lập trình được sử dụng nhằm làm cho ánh xạ đối tượng vào dữ liệu được lưu trữ trong cơ sở dữ liệu.



Hình 11: Hibernate là gì

1. **Phân biệt Hibernate và JPA**

Khi lập trình với Spring Boot, sẽ sử dụng Spring Data JPA để kết nối với cơ sở dữ liệu. Và cần thiết phải phân biệt giữa JPA và Hibernate khi sử dụng Spring Boot.

**JPA là gì?**

JPA (Java Persistence API) là một giao diện lập trình ứng dụng Java, nó mô tả cách quản lý các mối quan hệ dữ liệu trong ứng dụng sử dụng Java Platform.

JPA cung cấp một mô hình POJO persistence cho phép ánh xạ các table/các mối quan hệ giữa các table trong database sang các class/mối quan hệ giữa các object.

**Ví dụ:** table Users với các column (Id, name, age…) sẽ tương ứng với class Users.java với các field Id, name, age… từ đó mỗi khi truy vấn table hay các column ta sẽ truy vấn trực tiếp trên các class, các field của class mà không cần quan tâm tới việc đang dùng loại database nào, dữ liệu database ra sao…

**Một số khái niệm trong JPA**

**Entity**: Entity là các đối tượng thể hiện tương ứng một table trong cơ sở dữ liệu. Khi lập trình, entity thường là các class POJO đơn giản, chỉ gồm các method getter, setter.

**EntityManager**: EntityManager là một giao diện (interface) cung cấp các API cho việc tương tác với các Entity như Persist (lưu một đối tượng mới), merge (cập nhật một đối tượng), remove (xóa 1 đối tượng).

**EntityManagerFactory**: EntityManagerFactory được dùng để tạo ra một thể hiện của EntityManager.

Diagram

Description automatically generated

Hình 12: Kiến trúc JPA

**Hibernate là gì?**

Hibernate là một ORM (Object Relational Mapping) framework cho phép người lập trình thao tác với database một cách hoàn toàn tự nhiên thông qua các đối tượng. Lập trình viên hoàn toàn không cần quan tâm đến loại database sử dụng, SQL…

Hay nói cách khác, Hibernate chính là cài đặt của JPA (JPA là 1 tập các interface, còn Hibernate implements các interface ấy 1 cách chi tiết).

Diagram

Description automatically generated

Hình 13: Kiến trúc Hibernate

**Một số ORM framework khác.**

Ngoài Hibernate ra có một số framework khác như Open JPA, Eclipselink cũng thực hiện implements JPA nhưng Hibernate được sử dụng phổ biến hơn cả.

Diagram

Description automatically generated

Hình 14: Một số ORM framework khác Hibernate

1. **Ưu nhược điểm của Hibernate**

**Ưu điểm:**

* *Dễ dàng sử dụng*: Hibernate Framework giúp bạn quản lý các kết nối database và dễ dàng fix bug. Nó cung cấp rất nhiều API truy vấn có sẵn.
* *Mang tính độc lập*: Khi dùng Hibernate Framework bạn không cần quan tâm quá nhiều đến cơ sở dữ liệu khi muốn dùng để viết câu lệnh SQL.
* *Mang tính hướng đối tượng*: Nó hỗ trợ tập trung xử lý theo hướng đối tượng sao cho phù hợp với việc sử dụng trong các case CRUD như Create, Read, Update, Delete...
* *Mang tính tin cậy cao*: Hibernate đã trải qua công đoạn kiểm thử và kết quả đem tới là nó khá an toàn trong các truy vấn.

**Nhược điểm:**

* Hibernate Framework không tham gia hỗ trợ các câu truy vấn phức tạp.
* Bắt nguồn từ lý do Hibernate Framework không thể cover hết tất cả các cú pháp của các hệ quản trị cơ sở dữ liệu, vì thế một số trường hợp bắt buộc phải dùng Native SQL.
* Do Hibernate Framework được tự động sinh ra nên nó bị hạn chế sự can thiệp vào câu lệnh SQL.

1. **Kết luận**

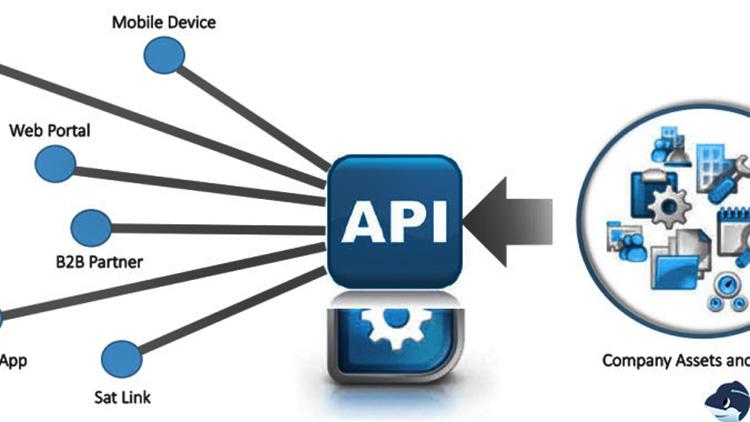
Vậy JPA là tập các giao diện (interfaces), qui chuẩn, định nghĩa. còn Hibernate thực hiện cài đặt (implements) các giao diện đó, định nghĩa đó.

1. **Sự khác nhau giữa API và Web Service**

Hệ thống của em sẽ sử dụng phần java Backend như một Web service để cung cấp API cho phần ReactJS frontend sử dụng.

API và Web service hoạt động như các phương tiện giao tiếp. Điểm khác biệt duy nhất giữa API và Web service chính là Web service tạo thuận lợi cho sự tương tác giữa hai máy qua mạng. API hoạt động như một giao diện giữa hai ứng dụng khác nhau để chúng có thể giao tiếp với nhau.

API là phương pháp mà các nhà cung cấp bên thứ 3 sử dụng để có thể viết các chương trình có giao diện dễ dàng giao tiếp với các chương trình khác. Web service được thiết kế để có giao diện được mô tả ở một định dạng mà máy tính có thể xử lý được thường được chỉ định trong Web Service Description Language (WSDL).



Hình 15: API là gì

Thông thường, "HTTP" là giao thức được sử dụng phổ biến nhất trong quá trình giao tiếp. Web service sử dụng 3 chuẩn chính là SOAP, REST và XML-RPC để làm phương tiện giao tiếp. API có thể sử dụng bất kỳ phương tiện giao tiếp nào để bắt đầu tương tác giữa các ứng dụng. Ví dụ, các cuộc gọi hệ thống được gọi bằng cách làm gián đoạn bởi các kernel Linux API.

API xác định chính xác các phương thức cho một chương trình phần mềm để tương tác với các chương trình khác. Khi hành động này liên quan đến việc gửi dữ liệu qua mạng, Web service sẽ xuất hiện trong đó. Một API nói chung bao gồm việc “gọi” các chức năng bên trong một chương trình phần mềm.

Trong trường hợp các ứng dụng Web, API được sử dụng dựa trên web. Các ứng dụng máy tính như bảng tính và tài liệu Word sử dụng API dựa trên VBA và COM mà không liên quan đến Web service. Một ứng dụng máy chủ như Joomla có thể sử dụng một API dựa trên PHP hiện diện trong máy chủ mà không yêu cầu Web service.

Web service chỉ đơn thuần là một API được gói trong HTTP. API không phải lúc nào cũng cần phải dựa trên web. Một API bao gồm một rule và các thông số kỹ thuật đầy đủ của một chương trình phần mềm để tạo thuận lợi cho việc tương tác. Một Web service có thể không chứa đầy đủ các thông số kỹ thuật và đôi khi không thể thực hiện tất cả các tác vụ mà một API hoàn chỉnh có thể làm được.

Các API có thể được tiếp xúc theo nhiều cách khác nhau, trong đó bao gồm: COM, file DLL và file .H trong ngôn ngữ lập trình C/C ++, file JAR hoặc RMI trong Java, XML qua HTTP, JSON qua HTTP, …. Phương thức mà Web service sử dụng để lộ API là hoàn toàn thông qua mạng.

**Kết luận**

* Tất cả Web services là APIs nhưng tất cả các APIs không phải là Web services.
* Web services không thể thực hiện được tất cả các thao tác mà API sẽ thực hiện.
* Một Web service sử dụng 3 chuẩn chính: SOAP, REST và XML-RPC trong quá trình giao tiếp, ngược lại API có thể sử dụng bất kỳ chuẩn nào để giao tiếp.
* Một Web service đòi hỏi luôn luôn phải có mạng để nó hoạt động nhưng API thì không cần.
* API tạo điều kiện liên kết trực tiếp với một ứng dụng trong khi Web service thì không.

1. **Thư viện jackson**

Để xây dựng được một Rest full API thì tiên quyết không thể không nhắc đến Jackson.

Jackson là một thư viện Java chứa rất nhiều chức năng để đọc và xây dựng JSON. Nó có khả năng ràng buộc dữ liệu rất mạnh mẽ và cung cấp một framework để tùy chỉnh quá trình chuyển đối tượng Java sang chuỗi JSON và chuỗi JSON sang đối tượng Java.

**Tạo JSON từ đối Java**

Có 3 cách để tạo JSON từ Java:

* Từ một đối tượng Java
* Từ cây JsonNode
* Từ Json Stream

**Phân tích chuỗi JSON**

Có 3 cách thường dùng để phân tích chuỗi JSON sang Java:

* **Streaming**: sử dụng JsonParser để phân tích cú pháp json. Nó cung cấp các phần tử json như là các token. Sử dụng JsonGenerator để tạo ra json từ chuỗi, số nguyên, boolean, …
* **Tree Traversing**: json có thể được đọc thành một JsonNode. Node sau đó có thể đi qua để có được thuộc tính cần thiết. Một cây cũng có thể được tạo ra và sau đó được viết như chuỗi json.
* **Data Binding** (liên kết dữ liệu): sử dụng Annotation để đánh dấu các thuộc tính trên đối tượng (POJO) liên kết với các phần tử của chuỗi JSON.

1. **ReactJS**

Những lý thuyết nêu trên là cơ bản đã xong phần core, phần Backend xử lý phía sau Browser. Ta cần có một hệ thống phía FrontEnd để sử dụng dữ liệu đó và hiển thị ra cho người dùng.

1. **Tổng quan**

ReactJS là thư viện JavaScript được sử dụng để xây dựng các thành phần UI có thể tái sử dụng. Theo tài liệu chính thức của React, tạm dịch “React là một thư viện để xây dựng giao diện người dùng có thể kết hợp. Nó khuyến khích việc tạo ra các thành phần giao diện người dùng có thể tái sử dụng, hiển thị dữ liệu thay đổi theo thời gian. Rất nhiều người sử dụng React làm V trong MVC. React trừu tượng hóa DOM khỏi bạn, cung cấp mô hình lập trình đơn giản hơn và hiệu suất tốt hơn. React cũng có thể hiển thị trên máy chủ bằng Node và nó có thể cung cấp năng lượng cho các ứng dụng gốc bằng React Native. React triển khai luồng dữ liệu phản ứng một chiều, điều này giúp giảm bớt phần soạn sẵn và dễ lý luận hơn so với ràng buộc dữ liệu truyền thống.”

1. **Một số tính năng nổi bật của React**

* *Sử dụng JSX*: Trong React, thay vì thường xuyên sử dụng JavaScript để thiết kế bố cục trang web thì sẽ dùng JSX. JSX được đánh giá là sử dụng đơn giản hơn JavaScript và cho phép trích dẫn HTML cũng như việc sử dụng các cú pháp thẻ HTML để render các subcomponent. JSX tối ưu hóa code khi biên soạn, vì vậy nó chạy nhanh hơn so với code JavaScript tương đương.
* *Single-way data flow (Luồng dữ liệu một chiều)*: ReactJS không có những module chuyên dụng để xử lý data, vì vậy ReactJS chia nhỏ view thành các component nhỏ có mỗi quan hệ chặt chẽ với nhau. Luồng truyền dữ liệu trong ReactJS là luồng dữ liệu một chiều từ cha xuống con. Việc ReactJS sử dụng one-way data flow có thể gây ra một chút khó khăn cho những người muốn tìm hiểu và ứng dụng vào trong các dự án. Tuy nhiên, cơ chế này sẽ phát huy được ưu điểm của mình khi cấu trúc cũng như chức năng của view trở nên phức tạp thì ReactJS sẽ phát huy được vai trò của mình.
* *Virtual DOM*: Những Framework sử dụng Virtual-DOM như ReactJS khi Virtual-DOM thay đổi, chúng ta không cần thao tác trực tiếp với DOM trên View mà vẫn phản ánh được sự thay đổi đó. Do Virtual-DOM vừa đóng vai trò là Model, vừa đóng vai trò là View nên mọi sự thay đổi trên Model đã kéo theo sự thay đổi trên View và ngược lại. Có nghĩa là mặc dù chúng ta không tác động trực tiếp vào các phần tử DOM ở View nhưng vẫn thực hiện được cơ chế Data-binding. Điều này làm cho tốc độ ứng dụng tăng lên đáng kể – môt lợi thế không thể tuyệt vời hơn khi sử dụng Virtula-DOM.

1. **Ưu nhược điểm của REACTJS**

**Ưu điểm**

* *Phù hợp với đa dạng thể loại website*: ReactJS khiến cho việc khởi tạo website dễ dàng hơn bởi vì bạn không cần phải code nhiều như khi tạo trang web thuần chỉ dùng JavaScript, HTML và nó đã cung cấp cho bạn đủ loại “đồ chơi” để bạn có thể dùng cho nhiều trường hợp.
* *Tái sử dụng các Component***:** Nếu bạn xây dựng các Component đủ tốt, đủ flexible để có thể thoả các “yêu cầu” của nhiều dự án khác nhau, bạn chỉ tốn thời gian xây dựng ban đầu và sử dụng lại hầu như toàn bộ ở các dự án sau. Không chỉ riêng mỗi ReactJS mà các framework hiện nay cũng đều cho phép chúng ta thực hiện điều đó, ví dụ Flutter chẳng hạn.
* *Có thể sử dụng cho cả Mobile application:* Hầu hết chúng ta đều biết rằng ReactJS được sử dụng cho việc lập trình website, nhưng thực chất nó được sinh ra không chỉ làm mỗi đều đó. Nếu bạn cần phát triển thêm ứng dụng Mobile, thì hãy sử dụng thêm React Native – một framework khác được phát triển cũng chính Facebook, bạn có thể dễ dàng “chia sẻ” các Component hoặc sử dung lại các Business Logic trong ứng dụng.
* *Thân thiện với SEO:* SEO là một phần không thể thiếu để đưa thông tin website của bạn lên top đầu tìm kiếm của Google. Bản chất ReactJS là một thư viện JavaScript, Google Search Engine hiện nay đã crawl và index được code JavaScript, tuy nhiên cũng cần thêm một vài thư viện khác để hỗ trợ điều này.
* *Debug dễ dàng:* Facebook đã phát hành một Chrome extension dùng trong việc debug trong quá trình phát triển ứng dụng. Điều đó giúp tăng tốc quá trình release sản phẩm cung như quá trình coding.
* *Công cụ phát triển web hot nhất hiện nay:* Nếu nhìn vào số liệu thống kê từ Google Trend ở Việt Nam ở hình 2.16 bên dưới, dạo lướt qua các trang tuyển dụng hàng đầu ở Việt Nam như Topdev, Itviec, v.v ta sẽ thấy số lượng tuyển dụng cho vị trí React Developer là cực kỳ lớn cùng với mức lương vô cùng hấp dẫn và độ phổ biến hiện tại của ReactJS trên thị trường Việt Nam là như thế nào.

Graphical user interface, text, application, email

Description automatically generated

Hình 16: So sánh độ phổ biến giữa ReactJS, AngularJS, VueJS

**Nhược điểm**

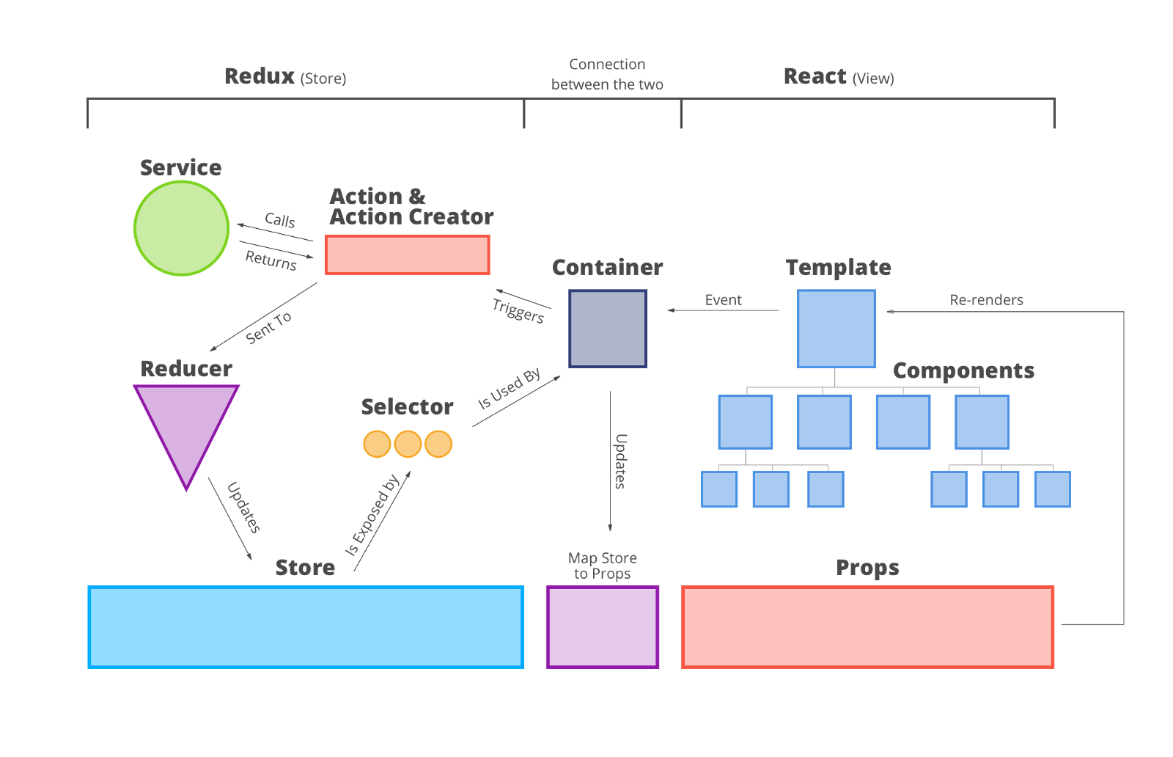
* *Sự cố với JSX*:JSX, hay thường gọi là cú pháp mở rộng JavaScript, đôi khi dường như không phải là một ý tưởng tuyệt vời để phát triển ứng dụng web vì nó liên quan đến nhiều phức tạp.
* *Tài liệu ít*:Tích hợp các tính năng mới có thể trở thành vấn đề đau đầu đối với một số nhà phát triển JavaScript. Một điều gây ra nhiều khó khăn là việc tạo tài liệu và hướng dẫn cho chức năng mới này. Để làm cho tình hình tốt hơn một chút, một số nhà phát triển JavaScript viết hướng dẫn của riêng họ cho các công cụ cụ thể được sử dụng trong các dự án.
* *Tốc độ phát triển cao*:Tốc độ phát triển cao có cả ưu và nhược điểm. Về khuyết điểm, thật không thoải mái khi thường xuyên học lại những cách làm mới mỗi khi có những thay đổi mới. Có thể khó cho các nhà phát triển áp dụng tất cả các thay đổi với tất cả các bản cập nhật liên tục.
* *Giao diện*:REACTJS chỉ bao gồm các Lớp giao diện người dùng của một ứng dụng và không bao gồm gì khác. Do đó, bạn vẫn cần lựa chọn các công nghệ khác để có được bộ công cụ đầy đủ để phát triển trong dự án.

1. **Kết luận**

React là một thư viện rất thú vị và được phát triển dựa trên rất nhiều cấu trúc phức tạp. Tuy nhiên thư viện này lại rất dễ sử dụng và thêm vào trong nhiều application khác nhau. Ngoài ra React còn có thư viện React Native được dùng để thiết kế native apps.

1. **Redux**

Phát sinh từ việc các ứng dụng single-page yêu cầu sử dụng Javascript ngày càng phức tạp. Khi đó, code sẽ phải quản lý số lượng state nhiều hơn. Trong khi đó, với Redux thì state của ứng dụng đều được lưu giữ trong store. Và mỗi component rất có thể truy access với bất kỳ state nào khi chúng muốn từ store này.



Hình 17: Cấu trúc Redux

1. **Tổng quan**

Redux được định nghĩa là một predictable state management tool dành riêng cho các ứng dụng của Javascript. Redux sẽ hỗ trợ viết ứng dụng hoạt động một cách nhất quán, chạy trong môi trường khác nhau và dễ hơn khi test. Redux được ra đời từ việc lấy cảm hứng bởi các tư tưởng ngôn ngữ Elm và kiến trúc Flux từ Facebook. Chình vì vậy, Redux luôn được sử dụng với mục đích kết hợp cùng với React.

1. **Cách hoạt động**

Redux sẽ hoạt động dựa vào 3 thành phần cơ bản là: Actions, Reducers và Store.

* **Với Actions**: Bạn có thể hiểu Actions là các events và chúng là cách cần thiết mà bạn cần làm để send data từ app đến Redux store. Các data xuất hiện thông qua các tương tác của user hoặc qua app, API call hoặc từ form submission.
* **Về Reducers**: Reducers là những function dạng nguyên thủy và chúng thường lấy state hiện tại của app. Từ đó, thực hiện một action rồi trả về dưới dạng một state mới. Các states này sẽ được lưu trữ như objects và chúng sẽ định rõ các state của ứng dụng thay đổi khi phản hồi một action gửi đến store.
* **Store**: Store là loại lưu trạng thái ứng dụng và là duy nhất trong bất kỳ ứng dụng Redux nào. Người dùng có thể access các state đã được lưu, update hoặc đăng ký cũng như hủy đăng ký các listeners thông qua helper methods.

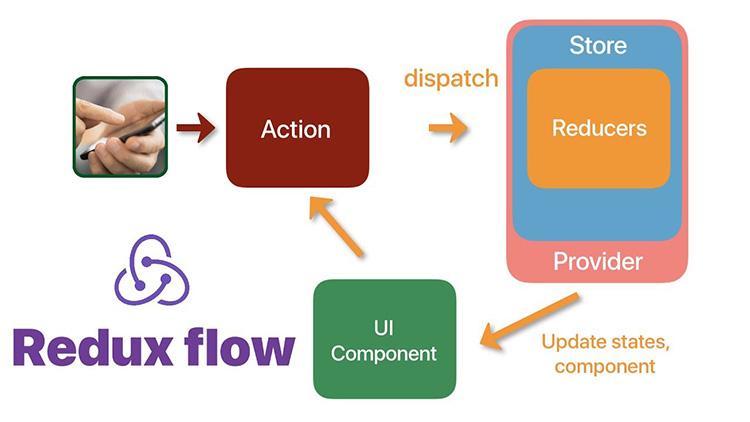
1. **Nguyên lý vận hành**

Quá trình xây dựng Redux các chuyên gia đã dựa vào 3 nguyên lý cơ bản sau:

Sử dụng nguồn dữ liệu đáng tin cậy duy nhất. Các State của tất cả ứng dụng đều nằm trong một object tree nằm trong một Store duy nhất.

Chỉ được phép đọc trạng thái: Có nghĩa rằng, để thay đổi State của ứng dụng thì chỉ có cách duy nhất là phát một Action.

Sử dụng hàm thuần túy để thay đổi. Với mục đích chỉ ra cách State được biến đổi từ Action. Vì thế, chúng ta cần sử dụng các pure function gọi là Reducer.



Hình 18: Nguyên tắc vận hành của Redux

1. **Một số khái niệm liên quan**

* **Redux Thunk**: Khi nhắc đến Redux thì không thể không nhớ đến Redux Thunk. Và định nghĩa này đã được các chuyên gia nhận định như sau: Redux Thunk là một Middleware có thể cho phép người dùng viết các Action trả về một function. Thay vì phải sử dụng một plain javascript object bằng cách trì hoãn quá trình đưa action đến reducer. Ngoài ra, Redux Thunk còn được sử dụng nhằm mục đích xử lý các logic bất đồng bộ phức tạp. Những đồng bộ này cần truy cập đến store hoặc lấy dữ liệu như Ajax request.
* **Redux Persist:** Redux Persist là dạng gói tự động hóa cho quy trình duy trì trạng thái từ cửa hàng Redux của bạn đến với bộ nhớ của thiết bị cục bộ. Ví dụ như: Redux Persist có nhiệm vụ thực hiện tái tạo cửa hàng Redux trong các lần khởi chạy ứng dụng. Các tiện ích này sẽ giúp người dùng giảm thiểu công việc cũng như quá trình để duy trì được dữ liệu trên thiết bị. Có thể là mã thông báo xác thực hoặc các cài đặt tài khoản.

Việc sử dụng Redux Persist sẽ giúp bạn thực hiện công việc hoàn toàn tự động mà chỉ cần lượng nhỏ bản ghi sẵn cho quá trình khởi tạo. Gói tự động này hoạt động rất hiệu quả và sở hữu nhiều bộ giảm được thiết kế vô cùng tốt. Ngoài ra, nó còn có thể giúp cho các bộ giảm bớt chi tiết khi cần thiết hoặc khi ứng dụng đang ngày càng phức tạp. Nhờ vậy mà quy trình quản lý cửa hàng Redux sẽ trở nên đơn giản mà hiệu quả hơn.

1. **Lợi ích mà Redux mang lại**

* **Output đồng nhất, dễ đoán**:Với chỉ một "Source of Truth" (store), chúng ta sẽ gặp ít vấn đề trong việc sync state giữa các component với nhau hơn.
* **Khả năng maintain**: Redux có bộ guideline hết sức chặt chẽ về cách tổ chức code, action làm gì, reducer làm việc gì,... Mọi thứ đều cụ thể và rõ ràng nên việc maintain sẽ dễ dàng hơn nhiều.
* **Khả năng scale**:Như đã nói ở trên, với việc tổ chức code nghiêm ngặt và rõ ràng, việc scale project sẽ trở nên dễ dàng hơn nhiều.

1. **Kết luận**

React là đứng một mình thì nó vẫn đỉnh. Chúng ta hoàn toàn có thể build một project hoàn chỉnh với chỉ React. Nhưng khi project ngày càng lớn với ngày càng nhiều component, việc chỉ sử dụng duy nhất React để quản lý state sẽ trở nên rất phức tạp. Đây chính là lúc Redux toả sáng, nó làm giảm bớt sự phức tạp trong các ứng dụng lớn như vậy. Nhưng không phải lúc nào cũng nên dùng Redux. Chỉ nên dùng redux ở một số trường hợp như sau:

* Có một cấu trúc các thành phần như trên  - dữ liệu được luân chuyển xuống nhiều tầng  - hãy cân nhắc dùng Redux.
* Cần lưu  - cache dữ liệu giữa các view. Ví dụ, tải về dữ liệu khi người dùng click vào trang chi tiết, và lưu dữ liệu lại để lần sau vào lại trang đó nhanh hơn -  cân nhắc dùng Redux.
* Nếu ứng dụng đang lớn lên, quản lí nhiều dữ liệu -  cân nhắc dùng Redux. Nhưng, tránh lạm dụng sử dụng nó, cân nhắc thêm nó vào khi thích hợp. Vì nếu quá lạm dụng Redux thì ngược lại nó lại khiến cho ứng dụng nặng nề hơn.

1. **React hook**
2. **Tổng quan**

Ở bản React 16.8 những nhà phát triển React đã kịp nhận ra vấn đề khó khăn của các lập trình viên khi phải loay hoay không biết nên dùng statless (functional) component hay là statefull component và quá nhiều LifeCycle, nên đã cho ra mắt một tính năng mới tên gọi là React hooks – Điều này cho phép sử dụng **state** và các tính năng khác của React mà không phải dùng đến **Class.**

Có thể thấy, các nhà phát triển React họ đang muốn hướng đến một tương lai Functional Programming thay vì sử dụng những Class mà chỉ nghe cái tên thôi là ta đã nghĩ ngay đến OOP. Cộng với việc không sử dụng Class kế thừa từ React Component nữa nên giờ đây kích thước bundle sẽ được giảm đáng kể bởi code sử dụng Hooks.

Để có thể thay thế được Class thì React Hooks cung cấp cho chúng ta một bộ các built-in Hooks, giúp chúng ta sử dụng được các thành phần tạo nên React, có hai loại built-in đó là: Basic Hooks và Additional Hooks

1. **Về Basic Hooks**

**useState**

Hàm này nhận đầu vào là giá trị khởi tạo của một state và trả ra một mảng gồm có hai phần tử, phần tử đầu tiên là state hiện tại, phần tử thứ hai là một function dùng để update state (giống như hàm setState cũ vậy). Ví dụ:

Khi dùng với Class (kiểu cũ)

Graphical user interface, text, application

Description automatically generated

Hình 19: Ví dụ hàm useState

Khi dùng hooks

Graphical user interface

Description automatically generated with medium confidence

Hình 20: Ví dụ hàm useState

Khi muốn update state cho isLoading là true thì chỉ cần gọi đến hàm setLoading(true).

**useEffect**

Hàm useEffect ra đời để giải quyết được sự phức tạp trong các hàm LifeCycle. Nó giúp xử lý các side effects, useEffect sẽ tương đương với các hàm componentDidMount, componentDidUpdate và componentWillUnMount trong LifeCycle. Ví dụ:

Graphical user interface, text, application

Description automatically generated

Hình 21: Ví dụ hàm useEffect

Có thể thấy trong useEffect ta cũng có thể thực hiện công việc call API giống như hàm ComponentDidMount ngày trước. Để tránh việc hàm useEffect luôn chạy vào mỗi khi có thay đổi State thì ta có thể truyền vào tham số thứ hai trong useEffect đó là một array, trong array này ta có thể truyền vào đó những giá trị mà useEffect sẽ subcribe nó, tức là chỉ khi nào những giá trị đó thay đổi thì hàm useEffect mới được thực thi. Hoặc bạn cũng có thể truyền vào một array rỗng thì khi đó nó sẽ chỉ chạy một lần đầu tiên sau khi render giống với hàm ComponentDidMount Ví dụ:

Graphical user interface, text, application

Description automatically generated

Hình 22: Ví dụ dùng useEffect tương tự chức ComponentDidMount

**Lưu ý:** trong hàm useEffect bắt buộc phải return về function thì khi làm điều này nó sẽ tương đương với việc ta sử dụng hàm LifeCycle componentWillUnMount

Tóm lại, những gì ta cần nhớ khi sử dụng useEffect là:

Graphical user interface, text, application

Description automatically generated

Hình 23: Tóm tắt hàm useEffect

1. **Về Additional Hooks**

**useReducer**

Thực tế khi sử dụng useState thì nó sẽ trả về một phiên bản đơn giản của useReducer, vậy nên chúng ta có thể coi useReducer như một phiên bản nâng cao hơn dùng để thay thế cho việc sử dụng useState. Nếu đã làm việc với React-Redux thì chắc hẳn ta sẽ dễ dàng nhận ra flow quen thuộc này. Giống như reducer trong Redux thì useReducer cũng nhận vào một reducer dạng (state, action) và trả ra một newState. Khi sử dụng sẽ nhận được một cặp bao gồm current state và dispatch function. Ví dụ:

Text

Description automatically generated

Hình 24: Ví dụ hàm useReducer

**useMemo**

useMemo giúp ta kiểm soát việc được render dư thừa của các component con, nó khá giống với hàm shouldComponentUpdate trong LifeCycle. Bằng cách truyền vào một tham số thứ hai thì chỉ khi tham số này thay đổi thì thằng useMemo mới được thực thi. Ví dụ:

Không sử dụng useMemo:

A picture containing text

Description automatically generated

Hình 25: Ví dụ hàm useMemo

Có sử dụng useMemo:

Text

Description automatically generated

Hình 26: Ví dụ có sử dụng hàm useMemo

**useCallback**

useCallback có nhiệm vụ tương tự như useMemo nhưng khác ở chỗ function truyền vào useMemo bắt buộc phải ở trong quá trình render trong khi đối với useCallback đó lại là function callback của một event nào đó như là onClick chẳng hạn.

Ví dụ:

Graphical user interface, text, application, email

Description automatically generated

Hình 27: Ví dụ hàm useCallBack

Trong ví dụ trên ta sử dụng useCallback cho sự kiện onClick, điều này có nghĩa là việc thay đổi giá trị text trong ô Input sẽ không làm component Checkbox bị re-render.

1. **Kết luận**

Ngoài những hook cơ bản hay được sử dụng mà mình đã giới thiệu ở trên thì vẫn còn một số hook khác như là useContext, useRef, useLayoutEffect, useDebugValue, useImperativeHandle các bạn có thể vào trang chủ của react hooks để tìm hiểu thêm nhé.

Để có thể sử dụng hooks một cách hiệu quả và tối ưu nhất thì mọi người hãy lưu ý rằng hạn chế sử dụng hooks trong các vòng lặp hay nested function bởi vì như vậy nó sẽ làm mất đi tính đúng đắn của hooks. Một điều nữa là chỉ nên gọi hooks trong React component, đừng gọi hooks trong một function javasript thông thường.

1. **Axios**
2. **Tổng quan**

Axios là một thư viện HTTP Client dựa trên Promise dành cho node.js và trình duyệt. Nó có tính đẳng hình (tức là cùng codebase có thể chạy trong cả trình duyệt và node.js). Ở phía server thì nó sử dụng native module http trong node.js, còn ở phía client (trình duyệt) thì nó sử dụng XMLHttpRequest.

1. **Tính năng**

* Tạo request từ trình duyệt bằng XMLHttpRequest
* Tạo request từ node.js bằng http
* Hỗ trợ Promise API
* Đón chặn request và response
* Biến đổi dữ liệu request và response
* Bãi bỏ request
* Tự động chuyển đổi cho dữ liệu JSON
* Hỗ trợ phía client bảo vệ chống lại XSRF

1. **Phân biệt Axios và Ajax**

Hiện nay có quá nhiều thư viện có thể dùng để call API, lúc mới bắt đầu em cũng hoang mang và hay nhầm lẫn giữa Ajax và Axios.

Thực sự thì Axios chỉ là một thư viện JavaScript, nó giúp sử dụng Ajax một cách dễ dàng hơn. Ngoài ra thì cũng có những thư viện khác cũng làm được điều tương tự như là Jquery, Request, Fetch…

1. **Kết luận**

Là một thư viện phổ biến, Axios được hưởng lợi rất nhiều từ một hệ sinh thái các thư viện bên thứ ba thừa hưởng từ tính năng của Axios.

Có rất nhiều lý do để ta chọn sử dụng Axios. Thư viện này có một API rõ ràng, với các method hữu dụng quen thuộc với bất kì ai đã từng sử dụng jQuery trước đây. Sự phổ biến của nó và việc có sẵn các thư viện bên thứ ba khiến Axios trở thành lựa chọn chắc chắn cho ứng dụng cho dù là front-end hay back-end hay cả hai.

**CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ**

1. **Xác định và mô tả chức năng**
2. **Xác định chức năng**
3. **Mô hình Use Case**
4. **Diagram

   Description automatically generatedCác chức năng:**
5. **Đặc tả Use Case**

Bảng 3: Mô tả chức năng

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Mã chức năng** | **Tên chức năng** | **Mô tả chức năng** |
| 1 | UC01 | Đăng nhập | Cho phép người dùng đăng nhập vào hệ thống. |
| 2 | UC02 | Đăng kí | Cho phép người dùng đăng kí người mua hàng, hoặc người bán hàng. |
| 3 | UC03 | Xem trang cá nhân | Cho phép người dùng xem thông tin cá nhân. |
| 4 | UC04 | Quản lý danh mục | Cho phép người dùng xem và lọc danh sách danh mục. |
| 5 | UC05 | Quản lý giỏ hàng | Cho phép người dùng quản lý sản phẩm trong giỏ hàng. |
| 6 | UC06 | Xem và lọc danh sách sản phẩm | Cho phép người dùng xem và lọc danh sách sản phẩm |
| 7 | UC07 | Thanh toán giỏ hàng | Cho phép người dùng thanh toán sản phẩm trong giỏ hàng. |
| 8 | UC08 | Xem danh sách sản phẩm | Cho phép người dùng xem tất cả sản phẩm. |
| 9 | UC09 | Đánh giá sản phẩm | Cho phép người dùng đánh giá và bình luận sản phẩm. |
| 10 | UC10 | Phê duyệt hoặc từ chối duyệt người bán | Cho phép quản trị viên phê duyệt hoặc từ chối duyệt tài khoản người bán. |
| 11 | UC11 | Thêm sản phẩm | Cho phép người bán thực hiện hành động thêm sản phẩm. |
| 12 | UC12 | Xóa sản phẩm | Cho phép người bán thực hiện hành động thêm sản phẩm. |
| 13 | UC13 | Chỉnh sửa sản phẩm | Cho phép người bán thực hiện hành động chỉnh sửa sản phẩm. |
| 14 | UC14 | Chỉnh sửa thông tin cá nhân người mua hàng | Người mua hàng có thể xem và chỉnh sửa thông tin cá nhân như: địa chỉ nhận hàng, địa chỉ thanh toán, phương thức thanh toán |

1. **Chức năng “Đăng nhập”**

Bảng 4: Mô tả chức năng đăng nhập

|  |  |
| --- | --- |
| Use Case ID | **UC01** |
| Use Case Name | Đăng nhập |
| Description | Cho phép người dùng đăng nhập vào hệ thống bằng tài khoản đã được tạo trước đó. |
| Actor | Khách |
| Trigger | Người dùng truy cập vào trang Web lần đầu tiên. |
| Pre-condition | * Thiết bị của người dùng đã được kết nối internet khi thực hiện đăng nhập. |
| Post-condition | * Người dùng đăng nhập thành công. * Hệ thống ghi nhận thông tin cho các thao tác tiếp theo. * Hệ thống hiển thị màn hình trang chủ. |
| Basic Flow | 1. Người dùng nhập tên người dùng và mật khẩu. 2. Người dùng click vào button **Đăng nhập**. 3. Hệ thống xác thực đăng nhập hợp lệ và chuyển hướng sang trang chính. |
| Alternate Flow | Không có. |
| Exception Flow | 1. Nếu tài khoản không hợp lệ thì sẽ hiển thị cảnh báo “Incorrect username or password”   *Use Case quay về bước 1.* |

1. **Chức năng “Đăng kí tài khoản”**

Bảng 6: Mô tả chức năng cập nhật ảnh đại diện

|  |  |
| --- | --- |
| Use Case ID | **UC02** |
| Use Case Name | Đăng kí tài khoản |
| Description | Cho phép người dùng khách đăng kí tài khoản dùng để bán hàng hoặc mua hàng. |
| Actor | Khách |
| Trigger | Người dùng click vào button **Đăng kí tài khoản** ở giao diện chính |
| Pre-condition | * Đang ở Use Case **UC02**. |
| Post-condition | Hệ thống cập tài khoản người dùng lên giao diện cho người dùng. |
| Basic Flow | 1. Người dùng click vào button **Đăng kí tài khoản** ở trang chủ 2. Hệ thống hiển thị Popup để tải ảnh đại diện. 3. Người dùng chọn ảnh đại diện từ máy và click vào button **OK**. 4. Hệ thống cập nhật ảnh đại diện mới và hiển thị lên giao diện cho người dùng. |
| Alternate Flow | Không có. |
| Exception Flow | Không có. |

1. **Chức năng “Xem trang cá nhân”**

Bảng 5: Mô tả chức năng xem trang cá nhân

|  |  |
| --- | --- |
| Use Case ID | **UC03** |
| Use Case Name | Xem trang cá nhân |
| Description | Cho phép người dùng sau khi đăng nhập có thể xem được thông tin cá nhân của chính bản thân |
| Actor | Người dùng |
| Trigger | Người dùng click vào “My Profile” ở thanh Header |
| Pre-condition | Người dùng đã đăng nhập. |
| Post-condition | Hệ thống hiển thị thông tin của người được chọn. |
| Basic Flow | * + - 1. Người dùng click vào “My Profile” ở thanh Header.       2. Hệ thống hiển thị thông tin cá nhân của người dùng |
| Alternate Flow | Không có. |
| Exception Flow | 2a. Nếu người dùng là người bán hàng:  + Dashboard: thông tin sản phẩm bán chạy và đơn hàng  + Product: các sản phẩm của người bán  + Order: các đơn đặt hàng của người bán  + Logout  2b. Nếu người dùng là người mua hàng: |

1. **Chức năng “Quản lí danh mục”**

Bảng 7: Mô tả chức năng đổi mật khẩu

|  |  |
| --- | --- |
| Use Case ID | **UC04** |
| Use Case Name | Quản lí danh mục |
| Description | Cho phép người dùng sau khi đăng nhập có thể xem danh mục sản phẩm |
| Actor | Người dùng |
| Trigger | Người dùng click vào button **Category** ở giao diện trang chủ |
| Pre-condition | * Người dùng đã đăng nhập. * Đang ở Use Case **UC01**. |
| Post-condition | Hệ thống cập nhật Category cho người dùng. |
| Basic Flow | 1. Người dùng click vào button **Category** và lựa chọn danh mục**.** 2. Người dùng tiến hành chọn vào button **Xem giao diện sản phẩm** để thay đổi giao diện xem sản phẩm. 3. Người dùng tiến hành chọn vào button Sắp xếp sản phẩm để sắp xếp sản phẩm theo thứ tự, hoặc theo giá, hoặc mới nhất 4. Hệ thống tiến hành trả về các sản phẩm cho người dùng nếu có. |
| Alternate Flow | Không có. |
| Exception Flow | Không có |

1. **Chức năng “Quản lí giỏ hàng”**

Bảng 8: Mô tả chức năng giỏ hàng

|  |  |
| --- | --- |
| Use Case ID | **UC05** |
| Use Case Name | Xem tất cả sản phẩm trong giỏ hàng |
| Description | Cho phép người dùng xem tất cả sản phẩm đã thêm vào giỏ hàng. |
| Actor | Người mua hàng |
| Trigger | Nguời dùng click vào biểu tượng giỏ hàng ở thanh Header. |
| Pre-condition | Người dùng mua hàng đã đăng nhập. |
| Post-condition | Hiển thị tất cả sản phẩm trong giỏ hàng. |
| Basic Flow | 1. Người dùng click vào biểu tượng giỏ hàng ở thanh Header. 2. Hệ thống hiển thị những sản phẩm đã thêm vào giỏ hàng. 3. Người dùng click vào **check out** đểtiến hành thanh toán 4. Hệ thống chuyển hướng sang trang **Thanh toán**. 5. Người dùng nhập vào các thông tin giao hàng và phương thức thanh toán 6. Hệ thống sẽ thông báo thông tin đơn hàng cho người bán sản phẩm. |
| Alternate Flow | Không có. |
| Exception Flow | Không có |

1. **Chức năng “Xem và lọc danh sách sản phẩm”**

Bảng 9: Mô tả chức năng xem và lọc danh sách sản phẩm

|  |  |
| --- | --- |
| Use Case ID | **UC06** |
| Use Case Name | Xem sản phẩm |
| Description | Cho phép người dùng xem sản phẩm của các người bán |
| Actor | Người dùng hệ thống |
| Trigger | * Người dùng click vào biểu tượng **Category** trên thanh **Header**. * Người dùng chọn các loại sản phẩm để xem các sản phẩm |
| Pre-condition | * Người dùng đã đăng nhập. * Đang ở Use Case **UC05**. |
| Post-condition | Hệ thống lấy các sản phẩm danh sách thông báo của người dùng. |
| Basic Flow | 1. Người dùng tích vào những danh mục sản phẩm. 2. Người dùng tiến hành chọn vào button Sắp xếp sản phẩm để sắp xếp sản phẩm theo thứ tự, hoặc theo giá, hoặc mới nhất 3. Hệ thống tiến hành trả về các sản phẩm cho người dùng nếu có. |
| Alternate Flow | Không có |
| Exception Flow | Không có |

1. **Chức năng “Thanh toán giỏ hàng”**

Bảng 10: Mô tả chức năng Thanh toán giỏ hàng

|  |  |
| --- | --- |
| Use Case ID | **UC07** |
| Use Case Name | Thanh toán giỏ hàng |
| Description | Cho phép người dùng thanh toán các sản phẩm có trong giỏ hàng |
| Actor | Người mua hàng |
| Trigger | Người dùng click vào button **Thanh toán** ở giao diện giỏ hàng |
| Pre-condition | Người dùng mua hàng đã đăng nhập. |
| Post-condition | Đơn hàng thanh toán thành công |
| Basic Flow | 1. Người dùng click vào button **Check out** ở giao diện giỏ hàng 2. Hệ thống hiển thị những thông tin người mua và những thông tin còn thiếu. 3. Người dùng nhập vào các thông tin giao hàng và phương thức giao hàng, phương thức thanh toán |
| Alternate Flow | 1. Đang ở Use Case **UC05**. |
| Exception Flow | Không có. |

1. **Chức năng “Xem danh sách sản phẩm”**

Bảng 11: Mô tả chức năng xem danh sách sản phẩm

|  |  |
| --- | --- |
| Use Case ID | **UC08** |
| Use Case Name | Xem sản phẩm |
| Description | Cho phép người bán hàng xem những sản phẩm đang kinh doanh |
| Actor | Người bán hàng |
| Trigger | Người dùng bán hàng chọn **Product** ở trang **Profile** |
| Pre-condition | Người dùng bán hàng xem sản phẩm |
| Post-condition |  |
| Basic Flow | 1. Người dùng nhấn vào Tài khoản và nhấn vào button **My Profile** 2. Người dùng click vào button Product 3. Hệ thống tiến hành kiểm tra sản phẩm và gửi thông tin sản phẩm của người bán |
| Alternate Flow | Không có |
| Exception Flow | Không có |

1. **Chức năng “Đánh giá sản phẩm”**

Bảng 12: Mô tả chức năng đánh giá sản phẩm

|  |  |
| --- | --- |
| Use Case ID | **UC09** |
| Use Case Name | Đánh giá sản phẩm |
| Description | Cho phép người dùng xem và đánh giá sản phẩm |
| Actor | Người dùng |
| Trigger | Người dùng chọn **thêm đanh giá** ở trang chi tiết sản phẩm |
| Pre-condition | Người dùng đã đăng nhập. |
| Post-condition |  |
| Basic Flow | Người dùng chọn vào sản phẩm để xem chi tiết sản phẩm  Người dùng chọn vào **WRITE REVIEW** để viết bình luận  Nhấn vào button **SEND REVIEW** |
| Alternate Flow | Không có. |
| Exception Flow | Không có |

1. **Chức năng “Phê duệt hoặc từ chối phê duyệt người bán”**

Bảng 13: Mô tả chức năng quản lý người bán

|  |  |
| --- | --- |
| Use Case ID | **UC10** |
| Use Case Name | Quản lý đăng kí tài khoản người bán |
| Description | Cho phép người dùng quản trị quản lý các tài khoản người bán |
| Actor | Quản trị viên |
| Trigger | Người dùng quản trị viên chọn **Pending Sellers** ở Profile admin |
| Pre-condition | * Người dùng đã đăng nhập. * Tài khoản người dùng có quyền quản trị viên (admin). |
| Post-condition | Hệ thống hiển thị danh sách người bán cần phê duyệt |
| Basic Flow | 1. Người dùng chọn **Pending Sellers** ở Profile admin 2. Hệ thống hiển thị ra danh sách người bán cần phê duyệt 3. Người quản trị có thể nhấn vào icon phê duyệt người bán |
| Alternate Flow | Không có. |
| Exception Flow | Không có. |

1. **Chức năng “Thêm sản phẩm mới”**

Bảng 14: Mô tả chức năng thêm sản phẩm mới

|  |  |
| --- | --- |
| Use Case ID | **UC11** |
| Use Case Name | Thêm sản phẩm mới |
| Description | Cho phép người bán hàng thêm sản phẩm mới |
| Actor | Người bán hàng |
| Trigger | Người dùng bán hàng chọn Add Product ở trang sản phẩm |
| Pre-condition | Người dùng bán hàng đã đăng nhập. |
| Post-condition | Hệ thống hiển thị danh sách sản phẩm theo điều kiện lọc |
| Basic Flow | 1. Người dùng bán hàng chọn **Add Product** ở thanh trang Product. 2. Hệ thống hiển thị Trang thêm sản phẩm 3. Người dùng nhập các thông tin sản phẩm cần thêm:  * Product Name. * Product price * Product description * Category. * Image.  1. Người dùng click vào button **Add Product.**   *Use Case quay lại bước 1.* |
| Alternate Flow | Không có. |
| Exception Flow | Không có |

1. **Chức năng “Xóa sản phẩm”**

Bảng 15: Mô tả chức năng xem và lọc danh sách sinh viên

|  |  |
| --- | --- |
| Use Case ID | **UC12** |
| Use Case Name | Xóa sản phẩm |
| Description | Cho phép người dùng bán hàng xóa sản phẩm. |
| Actor | Người bán hàng |
| Trigger | Người dùng chọn icon **Xóa** ở thanh mỗi dòng sản phẩm. |
| Pre-condition | Người dùng đã đăng nhập. |
| Post-condition | Hệ thống hiển thị danh sách sinh viên theo điều kiện lọc (nếu có, không thực hiện lọc sẽ lấy lên tất cả, có phân trang). |
| Basic Flow | 1. Người dùng chọn icon **xóa** ở mỗi cuối dòng 2. Hệ thống hiển thị danh sách sản phẩm còn lại. |
| Alternate Flow | Không có. |
| Exception Flow | Không có |

1. **Chức năng “Chỉnh sửa sản phẩm”**

Bảng 16: Mô tả chức năng xem danh sách đội nhóm

|  |  |
| --- | --- |
| Use Case ID | **UC13** |
| Use Case Name | Chỉnh sửa sản phẩm |
| Description | Cho phép người dùng bán hàng chỉnh sửa sản phẩm |
| Actor | Người bán hàng |
| Trigger | Người dùng chọn icon **Chỉnh sửa** ở thanh mỗi dòng sản phẩm. |
| Pre-condition | Người dùng bán hàng đã đăng nhập. |
| Post-condition | Hệ thống hiển thị danh sách đội nhóm. |
| Basic Flow | 1. Người dùng bán hàng chọn icon **Chỉnh sửa** ở cuối dòng sản phẩm 2. Hệ thống hiển thị Trang chỉnh sửa sản phẩm 3. Người dùng nhập chỉnh sửa các thông tin sản phẩm:  * Product Name. * Product price * Product description * Category. * Image.  1. Người dùng click vào button **Edit Product.**   *Use Case quay lại bước 1.* |
| Alternate Flow | Không có. |
| Exception Flow | Không có. |

1. **Chức năng “Chỉnh sửa thông tin cá nhân”**

Bảng 17: Mô tả chức năng chỉnh sửa thông tin cá nhân

|  |  |
| --- | --- |
| Use Case ID | **UC14** |
| Use Case Name | Chỉnh sửa thông tin cá nhân |
| Description | Cho phép người dùng mua hàng chỉnh sửa thông tin cá nhân đã đăng kí trước đó. |
| Actor | Người dùng mua hàng |
| Trigger | Người dùng chọn **Profile** ở trong trang **My Profile.** |
| Pre-condition | Người dùng mua hàng đã đăng nhập. |
| Post-condition | Hệ thống hiển thị danh sách thông tin cá nhân, thông tiên vận chuyển, thông tin thanh toán. |
| Basic Flow | 1. Người dùng chọn vào thông tin cần chỉnh sửa và nhất nút **Save Setting** 2. Hệ thống sẽ cập nhật thông tin cá nhân:  * Chi tiết cá nhân * Địa chỉ giao hàng * Thông tin thanh toán * Phương thức thanh toán |
| Alternate Flow | Không có. |
| Exception Flow | Không có. |

1. **Sơ đồ của hệ thống**

Diagram

Description automatically generated

Hình 30: Sơ đồ của hệ thống

Hệ thống được xây dựng bao gồm:

* **Front-end**: ReactJS, request API từ back-end, xử lý và trả lên giao diện cho người dùng.
* **Back-end**: Spring boot chạy dưới dạng jar file, kết nối thao tác trực tiếp với database, cung cấp API cho front-end.
* **Database**: Mysql, lưu trữ dữ liệu, nơi xây dựng quan hệ giữa các thực thể, mang tư duy của hệ thống.

Cả hai source sẽ được quản lý bởi Jenkins (Jenkins là một opensource dùng để thực hiện chức năng tích hợp liên tục (gọi là CI – Continuous Integration) và xây dựng các tác vụ tự động hóa.). Mục đích giúp tự động hóa phía server. Dễ dàng giúp start, stop hoặc đồng bộ code, build code và deploy, tất cả chỉ bằng một nút nhấn. Xem được cả log build và gửi email thông báo build thành công.

Và tất cả của hệ thống hiện tại đang được quản lý trên cùng một VPS server.

1. **Database Diagram**

**Hệ thống được chia làm ba phần lớn:**

**Vai trò, Chức vụ:**

Là cơ sở để tổ chức phân quyền theo chức năng trong hệ thống. Mỗi một vai trò (role) được tạo sẽ chỉ định các chức năng (option) và quyền (permission) cho vai trò đó.

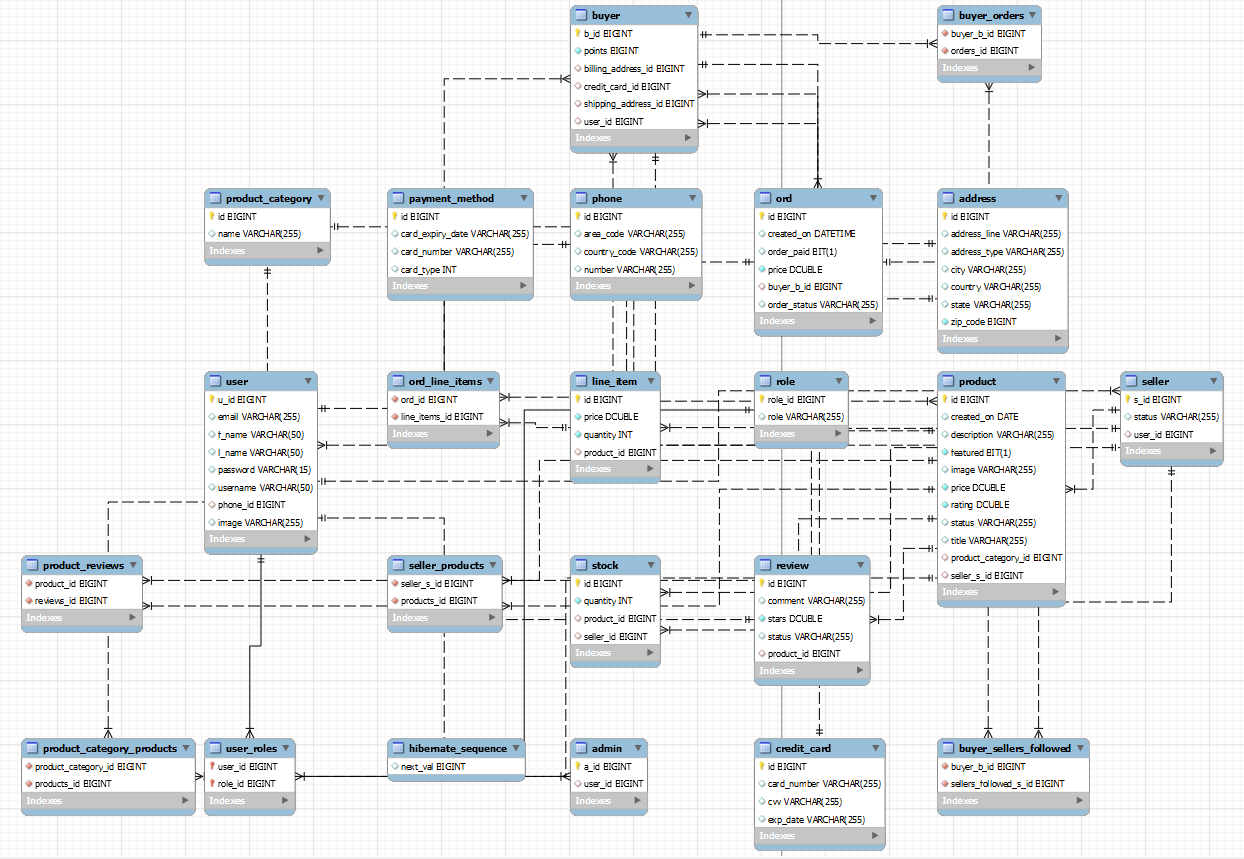
Khi tạo phòng ban thì sau đó sẽ tiến hành tạo các chức vụ trong phòng và chọn lấy một vai trò. Từ đó thành viên giữ chức vụ đó sẽ có được các quyền từ vai trò được chọn. Danh sách vai trò sẽ được dùng lại với nhiều chức vụ khác nhau.

**Sản phẩm, danh mục sản phẩm**

Mục đích thiết kế ra để người dùng bán hàng có thể tự thêm sản phẩm mà mình muốn. Mẫu này sẽ được người quản trị tạo, và các người bán khi thực hiện thêm sản phẩm sẽ chọn danh mục mà mình muốn thêm sản phẩm.

**Tổng thể:**

Như đã trình bày ở phía trên, hệ thống chia ra từng phần nhỏ, ngoài ra thì còn có các table bổ trợ cho một số chức năng khác, và mối quan hệ của từng phần nhỏ này với nhau sẽ được thể hiện ở sơ đồ bên dưới (quét mã QR để xem chi tiết)

****

**TÀI LIỆU THAM KHẢO**

**Tài liệu tiếng Việt**

* 1. Flinters (03/2021), Sử dụng JWT Authentication khi nào và như thế nào?, <https://labs.flinters.vn/security/su-dung-jwt-authentication-khi-nao-va-nhu-the-nao/>, xem 08/2022.
  2. GPCoder (01/2018), Hướng dẫn sử dụng thư viện Jackson, <https://gpcoder.com/3326-huong-dan-su-dung-thu-vien-jackson/>, xem 08/2022.
  3. Mauwebsite (10/2021), MySQL là gì? Tổng hợp kiến thức cần biết về MySQL, <https://mauwebsite.vn/mysql-la-gi/>, xem 08/2022.
  4. NguyenDuong (09/2016), Spring Framework - Architecture (Part 2), <https://viblo.asia/p/spring-framework-architecture-part-2-ZDEvLYjbGJb>, xem 08/2022.
  5. Wikipedia (2021), MySQL, <https://vi.wikipedia.org/wiki/MySQL>, xem 08/2022.

**Tài liệu tiếng Anh**

1. Baeldung (07/2021), Security with Spring, <https://www.baeldung.com/security-spring>, xem 08/2022.
2. Mysql, Mysql, <https://dev.mysql.com/doc/>, xem 08/2022.
3. reactjs, Getting Started, <https://reactjs.org/docs/getting-started.html>, xem 08/2022.
4. reactjs, Introducing Hooks, <https://reactjs.org/docs/hooks-intro.html>, xem 08/2022.
5. Redux, Redux, <https://redux.js.org/>, xem 08/2022.
6. spring, Spring Boot, <https://docs.spring.io/spring-boot/docs/current/reference/html/>, xem 08/2022.
7. spring, Spring Data, <https://reactjs.org/docs/getting-started.html>, xem 08/2022.
8. spring, Spring Security, <https://docs.spring.io/spring-security/reference/index.html>, xem 08/2022.