**TECHNICAL RESEARCH**

**REACT JS AND JAVA SPRING BOOT**

**Thành viên:** 18520335 Đinh Ngọc Uyên Phương

18520368 Phạm Ngọc Thịnh

**GVHD:** Thầy Nguyễn Công Hoan

**MỤC LỤC**

[**I. REACT JS**](#_c1m5nftcvjby) **1**

[1. Khái niệm React JS](#_uprj2rsuwerd) 1

[2. Tại sao lại chọn ReactJS? ReactJS giúp ích gì trong việc xây dựng giao diện?](#_42thcrf211pw) 1

[3. Tìm hiểu các thư viện - tính năng - package được áp dụng trong đồ án](#_hqvo5hvwsysx) 2

[3.1. React Hook](#_vnadfoox7qk) 2

[3.2. React Router](#_ces73gcwu16w) 3

[3.3. React Redux - Redux toolkit](#_60rga5zbfuwq) 3

[3.4. Redux Thunk](#_oe77ncel42xa) 4

[4. Mô tả tổng quát codebase](#_t3jm0z1w985b) 5

[**II. SPRING BOOT**](#_3gu5fwneldsx) **7**

[1. Khái niệm Spring Boot:](#_vchiw51imgb9) 7

[2. Ưu điểm của Spring Boot:](#_40zmdpeidbpn) 7

[3. Các khái niệm cơ bản dependencies - library - framework tìm hiểu và tích hợp sử dụng:](#_mn5xs8ox605s) 7

[4. Codebase:](#_exg12upxhjpw) 9

[**III. DANH SÁCH CÁC TÍNH NĂNG**](#_57fxfsqhqheb) **12**

# **I. REACT JS**

## **1. Khái niệm React JS**

React là một thư viện Javascript, được phát triển bởi Facebook, hỗ trợ xây dựng giao diện người dùng.

Một ứng dụng có giao diện xây dựng bằng React JS sẽ được tạo bởi nhiều thành phần nhỏ (component), trong đó, output của mỗi component là một đoạn code HTML nhỏ và có thể tái sử dụng xuyên suốt ứng dụng. Component được xem là từ khoá chính (key concept) khi nhắc đến React. Mỗi component có thể chứa các component nhỏ hơn, do đó, một ứng dụng với cấu trúc phức tạp sẽ được hình thành từ nhiều component con. Đồng thời, chỉ cần định nghĩa component con 1 lần và có thể sử dụng ở nhiều nơi.

## **2. Tại sao lại chọn ReactJS? ReactJS giúp ích gì trong việc xây dựng giao diện?**

Một số lợi ích của ReactJS khiến nhóm quyết định chọn thư viện này để tìm hiểu trong đồ án

* So với đối thủ cạnh tranh là Angular, React có một số điểm nổi trội hơn khiến nhóm quyết định tìm hiểu và áp dụng React trong đồ án này
  + ***Về mặt hiệu suất:*** React sử dụng công nghệ DOM ảo còn Angular thì sử dụng DOM thường. Với DOM thường, Angular sử dụng DOM API để tìm và cập nhật các element, hiệu suất của ứng dụng sẽ giảm đi đáng kể khi phải liên tục tìm và cập nhật DOM. Chính vì thế, DOM ảo ra đời giải quyết bài toán cập nhật DOM sao cho hiệu năng tốt nhất. Tất cả thay đổi trong React sẽ được cập nhật trong DOM ảo, React sử dụng thuật toán Diff để kiểm tra sự khác biệt giữa DOM ảo và DOM gốc rồi cập nhật những thay đổi tại những vị trí cần thiết.
  + ***Về mặt cộng đồng hỗ trợ:*** Mặc dù còn non trẻ, [React](https://github.com/facebook/react).js đã đạt được 163 nghìn sao và 32,7 nghìn lần Fork trên Github và vượt qua Angular.js một cách rõ ràng khi nó xoay quanh 59,5 nghìn Sao và 1,578 nghìn Người đóng góp. Do đó, cộng đồng hỗ trợ ReactJS rất mạnh, điều này có thể hỗ trợ nhóm trong việc tìm và sửa chữa lỗi
* JSX: trước khi thực hiện đồ án 1, nhóm đã có kiến thức căn bản về HTML, do đó khi tiếp xúc với JSX sẽ dễ hiểu hơn
* Để khởi tạo 1 ứng dụng React chỉ cần dùng lệnh “create-react-app" mà không cần tốn quá nhiều thời gian cho việc config
* Component: các component được tách thành từng file nhỏ và có thể tái sử dụng, vì vậy có thể tránh được việc code bị trùng nhau, dễ sửa chữa, duy trì và phát triển ứng dụng

## **3. Tìm hiểu các thư viện - tính năng - package được áp dụng trong đồ án**

### **3.1. React Hook**

3.1.1. Khái niệm: Hooks là những hàm cho phép kết nối React state và lifecycle vào các components sử dụng hàm. Với Hooks có thể sử dụng state và lifecycles mà không cần dùng ES6 class. Sự ra đời này đã giúp các lập trình viên tránh sự nhập nhằng về con trỏ this

3.1.2. Các hook phổ biến: useState, useEffect, useRef.

***a. useState:*** cập nhật giá trị của state, trong class component để thay đổi state phải dùng cú pháp this.state.user. Tuy nhiên với cú pháp này sẽ bị nhập nhằng con trỏ this, do đó, hook useState ra đời để giảm bớt sự nhập nhằng trên. Để thay đổi state trong functional component chỉ cần dùng hàm setState()

***b. useEffect:*** có tác dụng tương tự như componentDidMount, componentDidUpdate, và componentWillUnmount trong class component

***c. useRef:*** dùng để lưu trữ giá trị của một biến qua các lần render. Sau mỗi lần render, giá trị của một số biến trong component sẽ quay lại giá trị ban đầu, sử dụng useRef để lưu trữ các giá trị này

### **3.2. React Router**

3.2.1. Khái niệm: là một thư viện định tuyến, với Url này sẽ tương đương với Route này và render ra giao diện tương ứng.

3.2.2. Các tính năng - tiện ích được hỗ trợ bởi react router: <Link>, <NavLink>, <Redirect>, useLocation, useParams

***a. <Link> <NavLink>:*** đối với HTML, cặp thẻ để chuyến hướng là <a></a>, còn ở React thì dùng <Link>, <NavLink>

***b.<Redirect>:*** dùng để chuyển trang, có thể lấy thông tin các trang trước để sử dụng thông qua useLocation

***c. useParams*** (chỉ có ở React router v5): dùng để lấy thông tin các params trên url

***d. useLocation*** (chỉ có ở React router v5): với react router v4, để lấy được thông tin từ url (params, pathname, …) thì phải dùng đối tượng location được truyền như props vào component. Tuy nhiên, ở v5, chỉ cần dùng hook useLocation thì vấn đề này có thể giải quyết mà không cần truyền đối tượng location

### **3.3. React Redux - Redux toolkit**

3.3.1. Khái niệm Redux và Redux Toolkit

***a. Redux***: là công cụ dùng để quản lý state của ứng dụng. Ứng dụng có thể truy xuất state này ở mọi nơi thông qua store mà không cần truyền state từ component cha sang con.

***b. Redux toolkit:*** là một package được sinh ra để tiết kiệm thời gian trong việc cấu hình store, các file action hay reducer trong redux khá tách biệt và code lặp lại khá nhiều. Tuy nhiên, đến với redux toolkit, việc config store, action, reducer trở nên dễ dàng, nhanh chóng và ngắn gọn.

3.3.2. Các tính năng - tiện ích được hỗ trợ bởi redux: store, action, reducer và nguyên lý hoạt động

***a. Store*** là nơi lưu trạng thái của ứng dụng và là duy nhất, có thể lưu trữ, truy xuất hoặc cập nhật giá trị state trong store thông qua các action

***b. Action:*** là các event, các event này bao gồm type (để reducer biết đây là loại action gì, từ đó thực hiện các hành động cập nhật state thích hợp) và payload (chứa thông tin state mới)

***c. Reducer:*** là các pure function, các hàm này lấy state hiện tại, kết hợp với loại action, state mới và logic được khai báo trong reducer để cho một state có giá trị khác state ban đầu.

3.3.3. Các tính năng - tiện ích của redux được thay thế bởi redux toolkit:

***a. Slice:*** Với redux, khi định nghĩa action và reducer phải tách ra thành các file khác nhau, trong khi đó đối với slice, reducer và action được kết hợp lại trong cùng 1 file

### **3.4. Redux Thunk**

3.4.1. Khái niệm

Đối với Redux, Action chỉ nhận vào POJO rồi chuyển qua cho reducer. Tuy nhiên, 1 số trường hợp action cần truyền vào function (Async Action) cho các tác vụ bất đồng bộ (như gọi api, set time out)

Khi cần thực hiện các thao tác kể trên, middleware giúp action thực hiện thao tác bất động bộ xong mới chuyển cho reducer. Một số loại middleware có thể giải quyết bài toán trên như redux thunk, redux saga. Nhóm quyết định chọn redux thunk vì dễ sử dụng, cấu hình dễ dàng và phù hợp với những ứng dụng vừa và nhỏ

Redux Thunk là một Middleware cho phép viết các Action trả về một function thay vì một plain javascript object bằng cách trì hoãn việc đưa action đến reducer.

Redux Thunk được sử dụng để xử lý các logic bất đồng bộ phức tạp cần truy cập đến Store hoặc đơn giản là việc lấy dữ liệu như Ajax request.

## **4. Mô tả tổng quát codebase**

**4.1. Component** là nơi chứa các dump component - các component này sẽ không quan tâm đến việc xử lý dữ liệu, nó sẽ chú trọng vào xây dựng giao diện cho ứng dụng, còn xử lý logic nó sẽ không tham gia vào mà chỉ nhận dữ liệu từ các Component hay Service khác

**4.2. Pages** là nơi chưa các smart component - Component này sẽ chứa logic xử lý chức năng, truyền dữ liệu xuống các dump component

**4.3. Api**

* ***axiosClient:*** là nơi cấu hình cho các http request đến server và các request này không yêu cầu token kèm theo ở header
* ***axiosAuthenClient:*** là nơi cấu hình cho các http request đến server, và các request này yêu cầu gửi kèm token ở header. Việc gửi kèm token được cấu hình ở request interceptor - mỗi request trước khi tới server đều phải đi qua request interceptor này để đính kèm token. Đối với những response bị lỗi do token hết hạn, response interceptor sẽ gửi request refresh token hoặc điều hướng người dùng về trang đăng nhập
* ***xApi:*** là nơi thực hiện gọi và nhận dữ liệu từ các http request

**4.4. App/store:** là nơi cấu hình redux store

**4.5. Assets:** là nơi chứa các file ảnh hoặc style (CSS) chung cho toàn bộ project

**4.6. Utilities:** ngoài các file khai báo hằng số (Url, thời gian cookies hết hạn) thì utilities còn chứa các slice. Slice là nơi khai báo các reducer và action trong redux toolkit

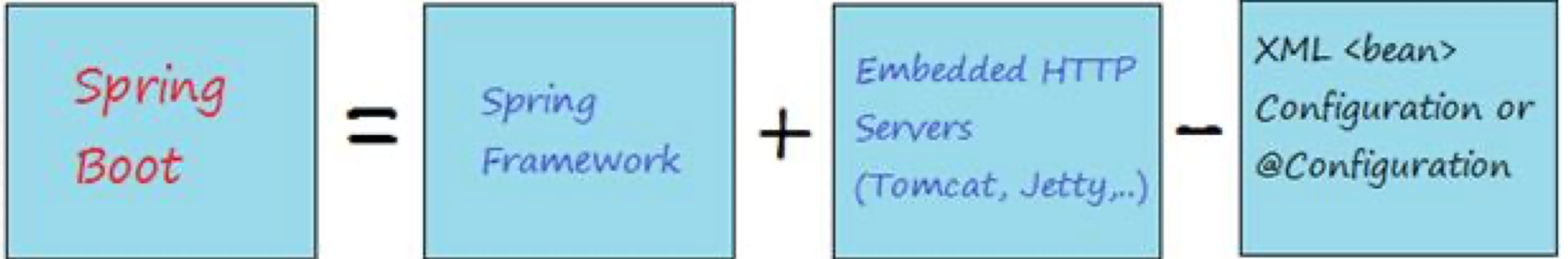
**4.7. Helpers:**

* ***authHeader:*** trả về định dạng token để gắn vào header cho mỗi http request
* ***cookiesService:*** là nơi tạo, thay đổi hoặc xóa các cookies
* ***refreshToken:*** là service dùng để gọi lên server lấy một token mới, nếu request này xảy ra lỗi, service sẽ điều hướng người dùng về trang đăng nhập

# **II. SPRING BOOT**

## 1. Khái niệm Spring Boot:

* Spring Boot là một module của Spring Framework, cung cấp tính năng RAD (Rapid Application Development - Phát triển ứng dụng nhanh).
* Spring Boot ra đời nhằm rút ngắn thời gian cài đặt và cấu hình Spring MVC Project. Giúp các lập trình viên tập trung vào nghiệp vụ hơn là cấu hình.
* Spring Boot dễ dàng trong việc tích hợp các hệ sinh thái của Spring như: Spring JDBC, Spring ORM, Spring Security,...



## 2. Ưu điểm của Spring Boot:

* Nhúng trực tiếp Tomcat, Jetty hoặc Undertow (Không cần phải deploy ra file war).
* Tự động cấu hình Spring khi cần thiết.
* Không sinh code cấu hình và không cần cấu hình bằng XML.
* Chuẩn cho Microservices (Cloud support, giảm việc setup, config các thư viện hỗ trợ).

## 3. Các khái niệm cơ bản dependencies - library - framework tìm hiểu và tích hợp sử dụng:

**3.1. Hibernate:**

* **ORM là gì ?** ORM là viết tắt của object-relational-mapping, công nghệ cho phép chuyển đổi từ các object trong ngôn ngữ hướng đối tượng sang database quan hệ và ngược lại.
* Hibernate là một ORM framework mã nguồn mở giúp các lập trình viên viết các ứng dụng Java có thể tham chiếu các đối tượng (entity) với bảng (table) trong hệ quản trị cơ sở dữ liệu.
* Hibernate giúp thực hiện giao tiếp giữa tầng ứng dụng với tầng dữ liệu (kết nối, truy xuất, lưu trữ,...).
* ***Tại sao chọn tìm hiểu và sử dụng Hibernate?*** Vì Hibernate giúp cải thiện một số vấn đề khi sử dụng JDBC:
  + Hạn chế lặp đi lặp lại những dòng code giống nhau trong ứng dụng chỉ để lấy dữ liệu từ database.
  + Giúp các lập trình viên đỡ phải vất vả với việc map giữa Object Java với các table tương ứng trong database.
  + Giúp giảm thiểu công sức khi thay đổi từ hệ quản trị cơ sở dữ liệu này sang hệ quản trị cơ sở dữ liệu khác(Hibernate viết một lần chạy mọi loại cơ sở dữ liệu).

**3.2. Spring Data JPA:**

* JPA (Java Persistence API) là một module nhỏ của Spring Data.
* JPA được tạo ra với mục đích giảm thiểu các đoạn code lặp đi lặp lại nhiều lần khi tương tác với cơ sở dữ liệu.
* JPA chỉ là một API định nghĩa các đặc tả cần thiết và không có code hiện thực từ những đặc tả đó.
* Cần phải có một cài đặt ORM để hoạt động. Các ORM framework có thể sử dụng cho JPA như: Hibernate, iBatis, Eclipse Link, OpenJPA,... Trong đồ án 1, chúng em tìm hiểu và sử dụng Hibernate là ORM framework cho đồ án.
* Có thể hiểu nôm na JPA là một interface và Hibernate chính là người hiện thực cái interface đó.

**3.3. Spring Security:**

* Là một framework của Spring được cung cấp để xử lý các vấn đề liên quan đến bảo mật, đặc biệt là vấn đề xác thực (authentication) và phân quyền(authorization) cho các ứng dụng web.

**3.4. Swagger:**

* Là một phần mềm mã nguồn mở được sử dụng để phát triển, xây dựng, thiết kế và làm tài liệu cho các RESTful Web Service.
* Swagger UI được tìm hiểu và tích hợp trong đồ án để hỗ trợ việc tự động làm các tài liệu, mô tả các API của đồ án.

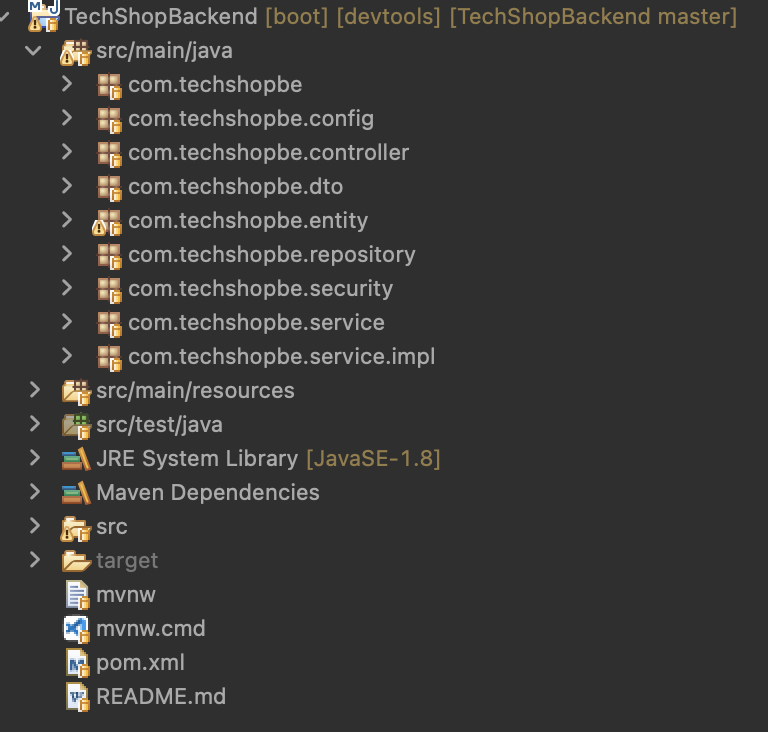
**3.5. Jbcrypt:**

* Là một thư viện hỗ trợ mã hoá các thông tin quan trọng (Vd: password,...)

**3.6. Jjwt:**

* Là một thư viện hỗ trợ các tác vụ và thao tác với Json Web Token như việc tạo ra các token, cấu hình token, đọc các token,.....
  + **Json Web Token** là một chuẩn để truyền tải thông tin một cách an toàn giữa các bên bằng đối tượng Json.
  + **Json Web Token** thường được dùng để xác thực và phân quyền người dùng( authentication và authorization), chuỗi token sẽ được gửi kèm trong phần header của request và server sẽ thông qua token đó xác thực request.

## 4. Codebase:

****

**Mô tả các packages trong Codebase:**

* **com.techshopbe**: chứa file SpringBootApplication để chạy ứng dụng Spring Boot.
* **com.techshopbe.config**: chứa các class dùng để Config hệ thống (VD: CorsConfig,..) hoặc Config những thư viện, framework sử dụng (VD: SwaggerConfig,..).
* **com.techshopbe.controller**: chức các class Controller ( ở đây là các RestController trong Spring Boot), các Controller giúp điều hướng và mapping các request đến server.
* **com.techshopbe.dto**: chứa các class được định nghĩa là dto (data transfer object). (Vd: AuthenticationDTO là một data transfer object chứa những thông tin nhỏ của user dùng để xác thực đăng nhập,...).
* **com.techshopbe.entity**: chứa các class là entity của dự án, các entity có khả năng mapping với các table dưới cơ sở dữ liệu (sử dụng ORM framework).
* **com.techshopbe.repository**: chứa các repository của dự án, các repository này được kế thừa từ JPARepository (được cung cấp bởi Spring JPA).
* **com.techshopbe.security**: chứa các class dùng để secure hệ thống, các class có tính năng bảo mật như xác thực và phân quyền và các cấu hình security cho ứng dụng.
* **com.techshopbe.service**: chứa các Service Interface của ứng dụng, các Service Interface này sinh ra giúp cho việc định nghĩa và liệt kê các phương thức trở nên linh hoạt hơn. Giúp cho dự án dễ bảo trì và nâng cấp về sau.
* **com.techshopbe.service.impl**: chứa các class Service, các class Service này sẽ implements những Service Interface và định nghĩa những phương thức của Service Interface đó.

# 

# **III. DANH SÁCH CÁC TÍNH NĂNG**

1. Hỗ trợ tìm kiếm trực tiếp (Live search)
2. Thanh điều hướng đến các loại sản phẩm mà TechShop cung cấp
3. Gợi ý các dòng sản phẩm liên quan
4. Đánh giá từ người dùng sau khi trải nghiệm dịch vụ.
5. Quảng cáo các chương trình khuyến mãi
6. Tạo giỏ hàng và đặt hàng nhanh chóng
7. Tạo danh sách sản phẩm yêu thích
8. Xem thông tin chi tiết của sản phẩm
9. Cập nhật các loại thông tin cá nhân (thông tin giao hàng, thanh toán)
10. Đăng ký tài khoản để nhận các thông tin ưu đãi, theo dõi đơn hàng