TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

-----🙞🙜🕮🙞🙜-----



**BÁO CÁO PROJECT 2**

**Đề tài**

**XÂY DỰNG TRANG WEB BÁN QUẦN ÁO ONLINE**

Sinh viên thực hiện:

Phạm Duy Tiên 20164038

Nguyễn Anh Phương 20163228

Hồ Xuân Cường 20160537

GVHD: TS. Trần Việt Trung

Hà Nội, ngày 18 tháng 5 năm 2019

MỤC LỤC

[Lời nói đầu 4](#_Toc9233891)

[Phần 1: ReacJS 5](#_Toc9233892)

[1. Giới thiệu về ReactJS 5](#_Toc9233893)

[1.1 ReactJS là gì ? 5](#_Toc9233894)

[1.2. Tại sao phải sử dụng ReactJS ? 5](#_Toc9233895)

[1.3. Để học tốt ReactJS cần những kiến thức nền tảng gì ? 5](#_Toc9233896)

[1.4. Lịch sử hình thành ReactJS 5](#_Toc9233897)

[2. JS ES6 6](#_Toc9233898)

[2.1 Arrow function 6](#_Toc9233899)

[2.2. Callback function 7](#_Toc9233900)

[2.3. Hướng đối tượng trong JS 7](#_Toc9233901)

[3. Những kiến thức trọng tâm trong ReactJS 7](#_Toc9233902)

[3.1. Cấu trúc thư mục ReactJS 7](#_Toc9233903)

[3.2. Components 8](#_Toc9233904)

[3.3. Props trong ReacJS 9](#_Toc9233905)

[3.4. State trong ReactJS 9](#_Toc9233906)

[3.5. Ref trong ReactJS 10](#_Toc9233907)

[3.6. Components Life Cycle (Vòng đời của components) 11](#_Toc9233908)

[3.7. React Router 11](#_Toc9233909)

[Phần 2: Redux 14](#_Toc9233910)

[1. Nguyên lý 14](#_Toc9233911)

[2. Cấu trúc 14](#_Toc9233912)

[Phần 3: API Spring Boot 17](#_Toc9233913)

[1. Giới thiệu 17](#_Toc9233914)

[2. Cài đặt và chạy chương trình Spring Boot 17](#_Toc9233915)

[2.1. Cài đặt Spring Tool Suite (STS) 17](#_Toc9233916)

[2.2. Tạo Spring Boot Project và chạy chương trình 17](#_Toc9233917)

[3. Xây dựng Restful API bằng Spring Boot 18](#_Toc9233918)

[3.1. Thiết lập cấu hình các file 18](#_Toc9233919)

[3.2. Giải thích nội dung các file 19](#_Toc9233920)

[3.3. Kết quả 24](#_Toc9233921)

[Phần 4: Kết hợp ReactJS với JavaSpring 25](#_Toc9233922)

[1 . Fetch 25](#_Toc9233923)

[2. Axios 25](#_Toc9233924)

[3. So sánh axios và fetch 25](#_Toc9233925)

[Phần 5. Các biểu đồ của trang web 26](#_Toc9233926)

[1. Biểu đồ Usecase tổng quan 26](#_Toc9233927)

[2. Biểu đồ cơ sở dữ liệu 27](#_Toc9233928)

[Phần 6. Tài liệu tham khảo 28](#_Toc9233929)

# Lời nói đầu

Hiện nay, Web Application có vô vàn công nghệ để xây dựng. Về phía sever-side chúng ta có các ngôn ngữ PHP, JAVA, Python, NodeJS,...và cả về phía client-side chúng ta có AngularJS, ReactJS, Bootrap, JQuery,.... Hay chỉ nói riêng JS framework đã có khoảng hơn 63 framework.



Việc này có vẻ như sẽ giúp ích cho lập trình viên có thể tạo một trang Web nhanh chóng hơn, tuy nhiên có một vấn đề là việc học gì cũng đã là một vấn đề đau đầu với những người mới bắt đầu như chúng ta. Trong báo cáo, bọn em đã lựa chọn công nghệ ReacJS để làm front-end và Java Spring boot để tạo API giao tiếp với sever-side. Vậy ReactJS có gì khác với những framework còn lại ? ReactJS có gì hơn ? Tại sao nên chọn ReactJS ? ... và nhiều câu hỏi khác. Tất cả sẽ được giải đáp trong báo cáo này.

Phần 1: Giới thiệu về ReacJS

Phần 2: Quản lý state với Redux

Phần 3: API Spring Boot

Phần 4: ReacJS kết nối Spring Boot

Phần 5: Các biểu đồ của trang web

Phần 6: Tài liệu tham khảo

# Phần 1: ReacJS

## 1. Giới thiệu về ReactJS

### 1.1 ReactJS là gì ?

React (còn được gọi là Reactjs hay React.js) là một Thư viện javascript được tạo ra bởi sự cộng tác giữa Facebook và Instagram. Nó cho phép những nhà phát triển web tạo ra giao diện người dung nhanh chóng. Phần Views của Reactjs thường được hiển thị bằng việc chủ yếu dung các component mà chứa các component cụ thể hoặc các thẻ HTML. Một trong những đặc trưng duy nhất của Reactjs là việc render dữ liệu không những có thể thực hiện ở tầng server mà còn ở tầng client.

Nó cũng sử dụng khái niệm là Virtual DOM (DOM ảo). Virtual DOM tạo ra bản cache cấu trúc dữ liệu của ứng dụng trên bộ nhớ. Sau đó, ở mỗi vòng lặp, nó liệt kê những thay đổi và sau đó là cập nhật lại sự thay đổi trên DOM của trình duyệt một cách hiệu quả. Điều này cho phép ta viết các đoạn code như thể toàn bộ trang được render lại dù thực tế là Reactjs chỉ render những component hay subcomponent nào thực sự thay đổi.

### 1.2. Tại sao phải sử dụng ReactJS ?

Trước tiên là sử nổi lên của single-page (tức tất mọi thao tác hiển thị trên một trang) việc này tạo giúp nâng cao trải nghiệm người dùng UX, tuy nhiên việc xử lý đó có thể tạo áp lực lớn cho Sever vì phải load trang web nhiều lần. ReactJS sinh ra cho việc này

Để là được việc đó ReactJS cung cấp hai khái niệm Component và VitrualDOM như đã nhắc ở trên. Component cho ta khả năng re-use (tái sử dụng) code rất tốt vì những thành phần giống nhau ta chỉ cần viết một lần điều này cũng cho ta khả năng chỉnh sửa và maintance (bảo trì) tốt.

VitrualDOM (bình thường Sever sẽ tạo ra cây DOM thật gửi lên phái View), nếu có bất kì sự thay đổi nào DOM ảo chỉ render lại những thay đổi đó không cần phải render cả trang Web như DOM thật, tuyệt vời hơn ReactJS có thể làm điều này ở phía Client-side giúp giảm tải khá nhiều cho phía sever-side

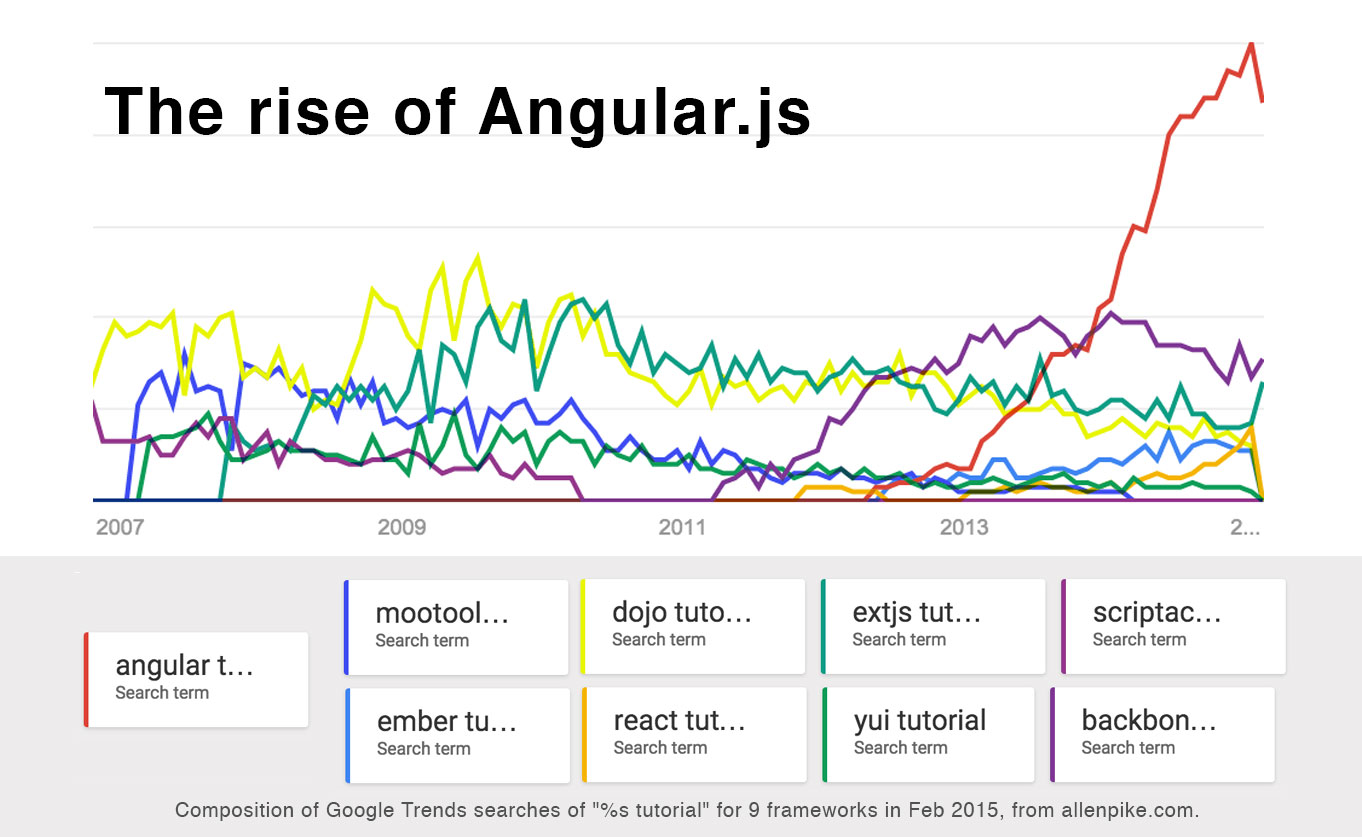
### 1.3. Để học tốt ReactJS cần những kiến thức nền tảng gì ?

Vì ReactJS là một library của JS cho nên kiến thức nền tảng về JS là điều chắc chắn, đặc biệt là kiến thức JS ES6 vì ReactJS sử dụng cú pháp JS ES6.

Là một front-end library cho nên những kiến thức về HTML, CSS, Bootrap là điều bắt buộc

### 1.4. Lịch sử hình thành ReactJS

Năm 2010, Google ra đời AngularJS khá mạnh mẽ với hỗ trợ double-binding dữ liệu, [dependency injection](https://toidicodedao.com/2015/11/03/dependency-injection-va-inversion-of-control-phan-1-dinh-nghia/), ajax, giúp công việc lập trình web trở nên rất nhanh chóng và hiệu quả



Năm 2011, Facebook tuyến bố angularJS chậm chạp và nặng nề và cho ra đời ReactJS. và với Redux đã tạo nên một công cụ mạnh mẽ phía view

*(Để biết thêm các bạn có thể đọc ở series JS truyền kì ở link:*

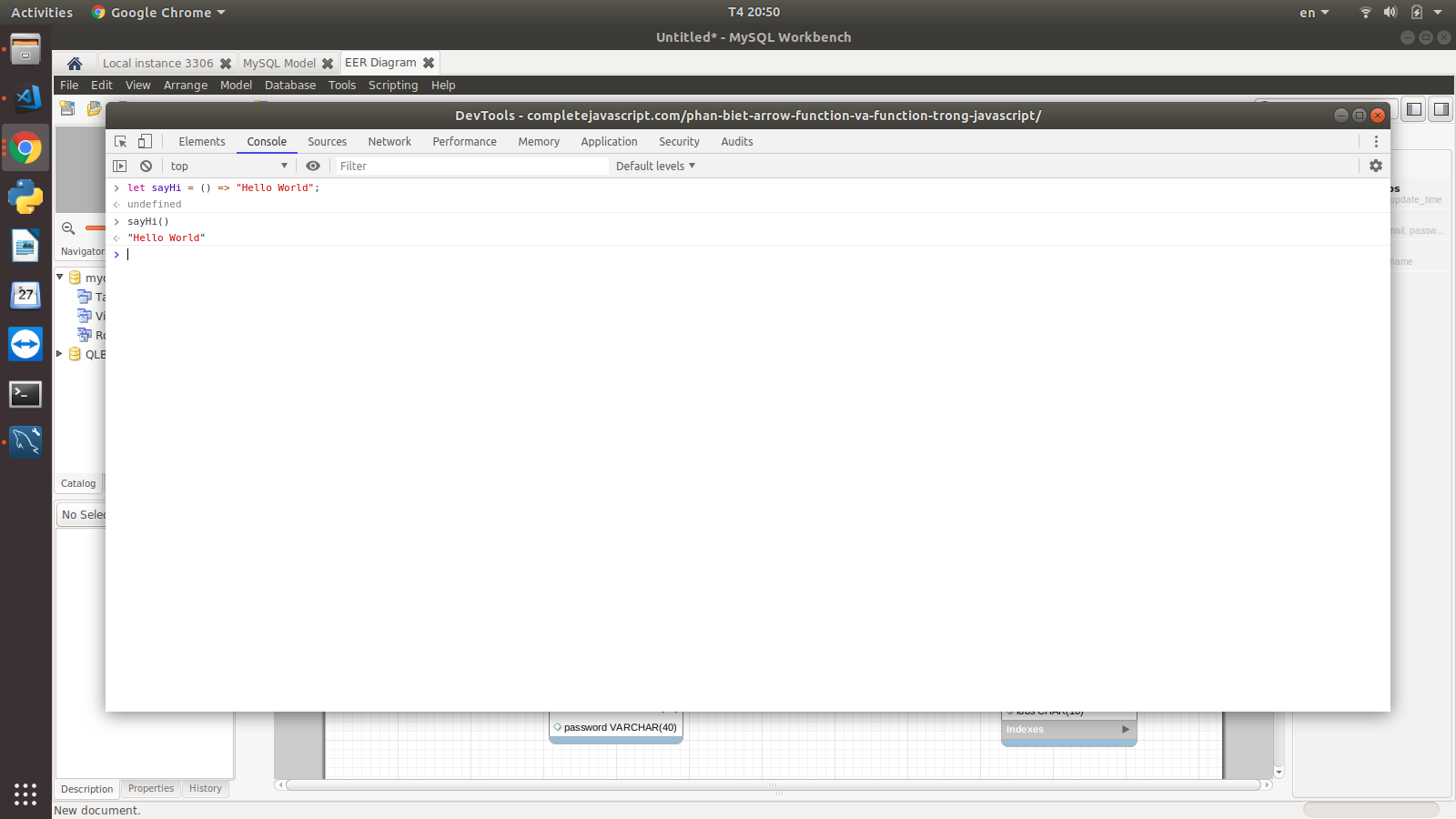
[*https://toidicodedao.com/2016/08/18/js-truyen-ki-chuong-2-vo-lam-day-song/*](https://toidicodedao.com/2016/08/18/js-truyen-ki-chuong-2-vo-lam-day-song/)*)*

## 2. JS ES6

Ngoài việc sử dụng cú pháp JSX *(sẽ đề cập ở phần sau)* thì ReactJS sử dụng chủ yếu sử dụng cú pháp JS6 vậy nên việc biết và sử dụng JS ES6 là điều thiết yếu, sau đây mình xin điểm qua một số kiến thức JS ES6 mình thấy là quan trọng nhất.

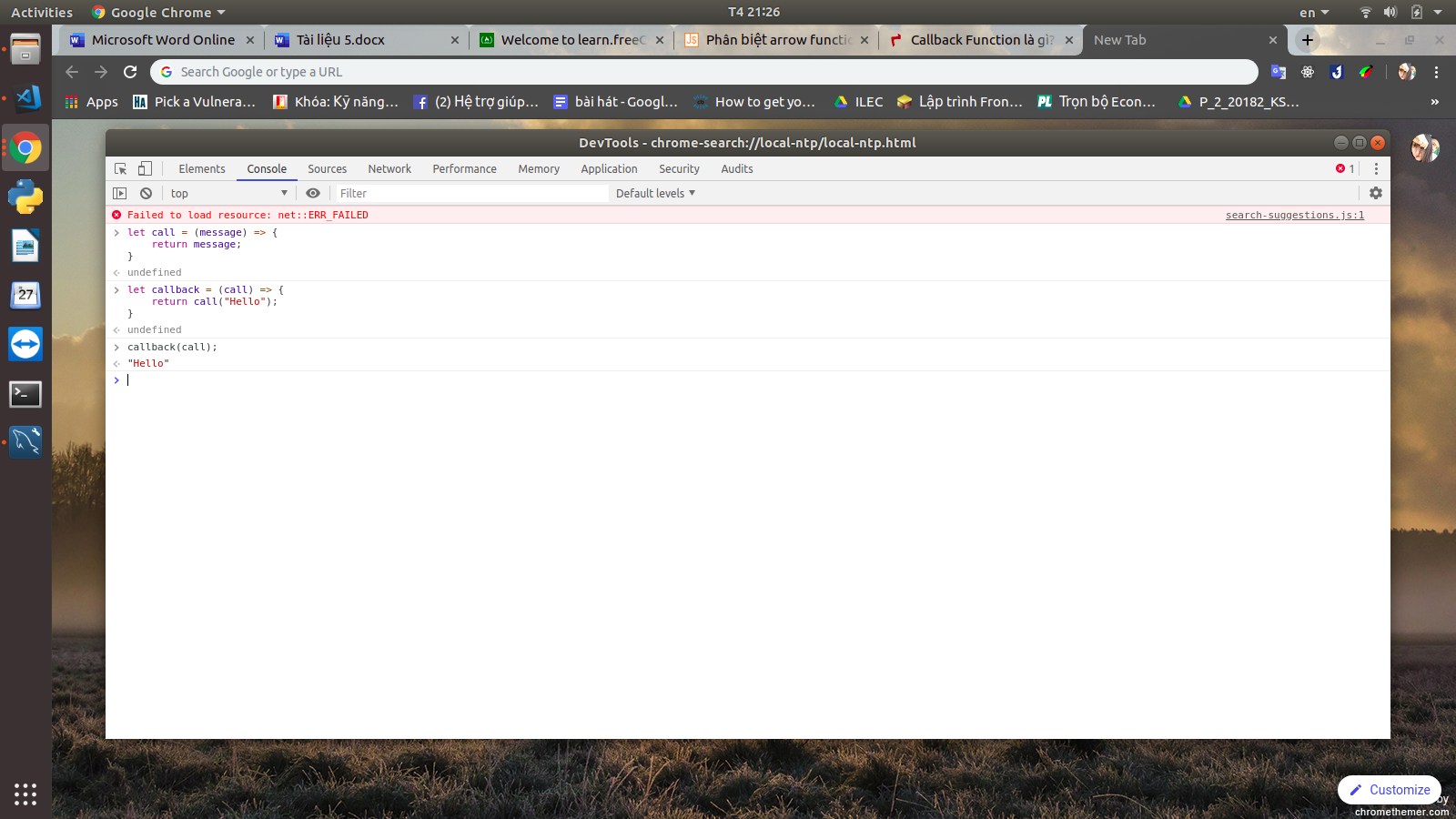
### 2.1 Arrow function

Arrow function cho khả năng viết code ngắn gọn hơn, là điều kiện để viết callback function dễ dàng hơn.



### 2.2. Callback function

Callback function được sử dụng cực kì phổ biết trong reactjs, vậy nó là gì?? Là cách chúng ta truyền vào function một function khác như một parameter



Callback hay được sử dụng trong các phương thức map, filter, reduce hay được ứng dụng rất nhiều trong truyền function dưới dạng một props

Như ta đã biết JS là ngôn ngữ bất đồng bộ (async) nên việc hàm trước chưa chạy xong hàm sau đã chạy có thể sinh ra nhưng lỗi không đoán trước được nên callback sinh ra để giải quyết vấn đề trên

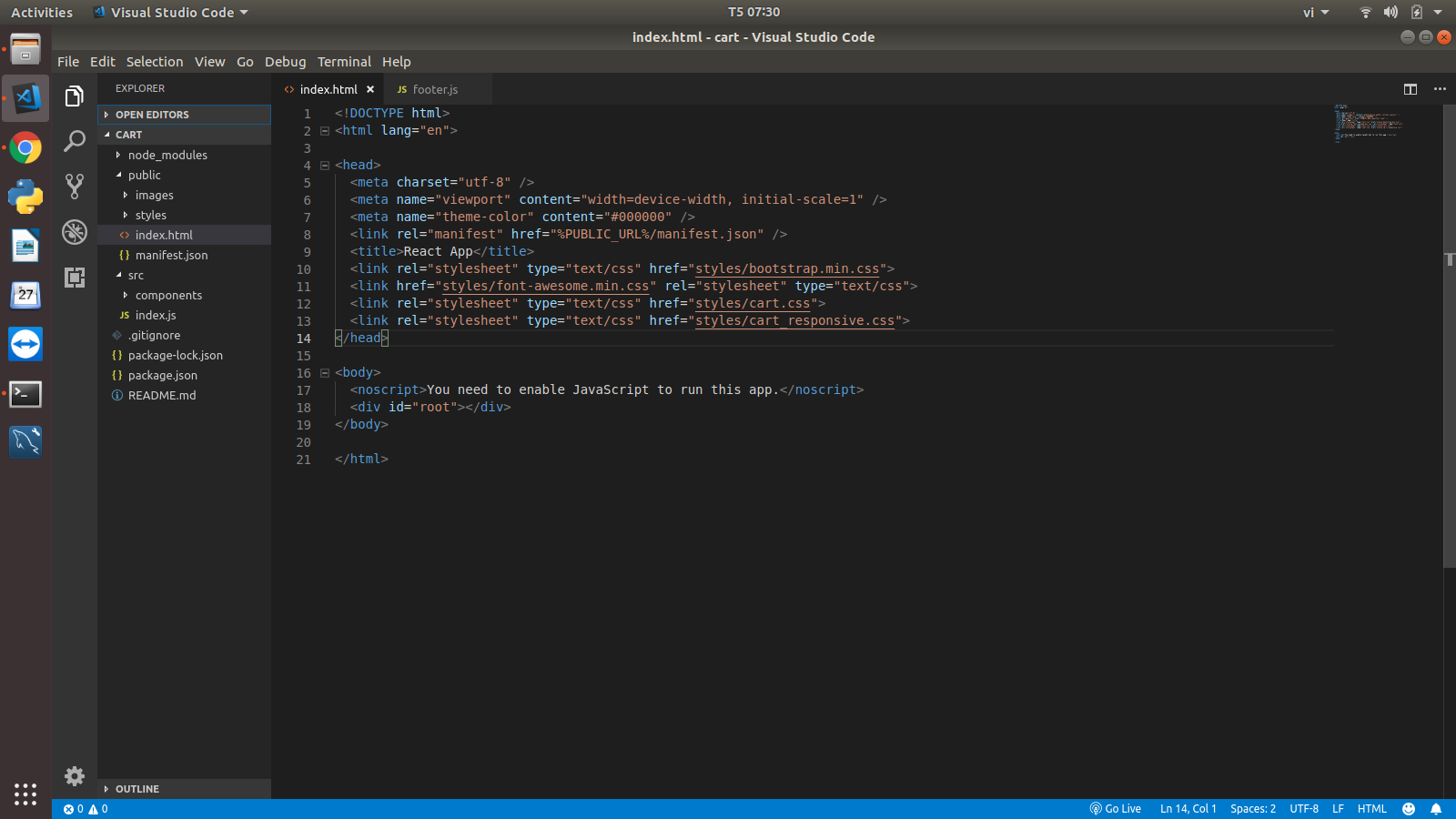
### 2.3. Hướng đối tượng trong JS

Như ta đã biết JS là ngôn ngư hướng đối intansce không như Java hay các ngôn ngữ hướng đối tượng khác trước đây nó không có class cụng như contructor. Tuy nhiên ở phiên bản ES6 đã tích hợp thêm với một số ưu điểm hơn

Hướng đuối tượng trong JS có đôi chút khác biết do đó việc học nắm vững kiến thức này là hết sức quan trọng đối với việc tạo Components sau này

## 3. Những kiến thức trọng tâm trong ReactJS

### 3.1. Cấu trúc thư mục ReactJS



Node\_modul : để biên dịc mã JS của ReactJS thì ta cần sever NodeJS biên dịch câu lệnh ReactJS, đây cũng là thư mục chưa các thư viện của ReactJS

Public: các folder ảnh. style hay JS của trang Web, bên cạnh đó có chưa file index.html (các component đc render ra đây)

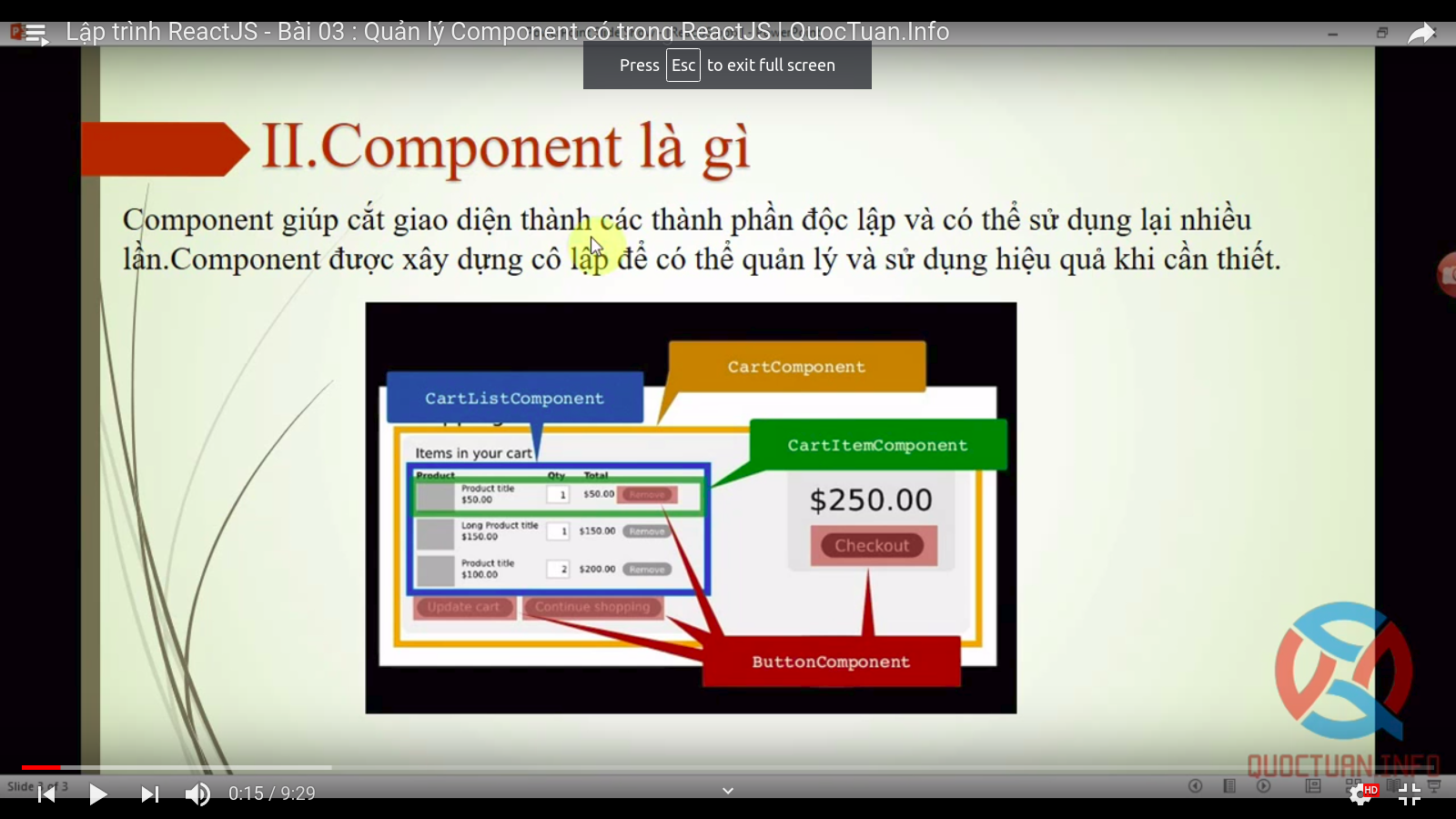
Src : quan trọng nhất, chưa các file components của ReacJS

package.json: chưa thông tin về app theo dạng json

### 3.2. Components

Component là khái niệm quan trọng nhất trong ReactJS nó giúp ReactJS trở nên mạnh mẽ. Bình thường với những giao diện truyền thống ta phải copy paste những đoạn HTML giống nhau ví dụ như một item trong Web bán hàng, vậy điều gì xảy ra khi ta phải sửa một thẻ trong đoạn code đó, ta phải sửa 1000 lần nếu có 1000 đoạn mã giống nhau đó.

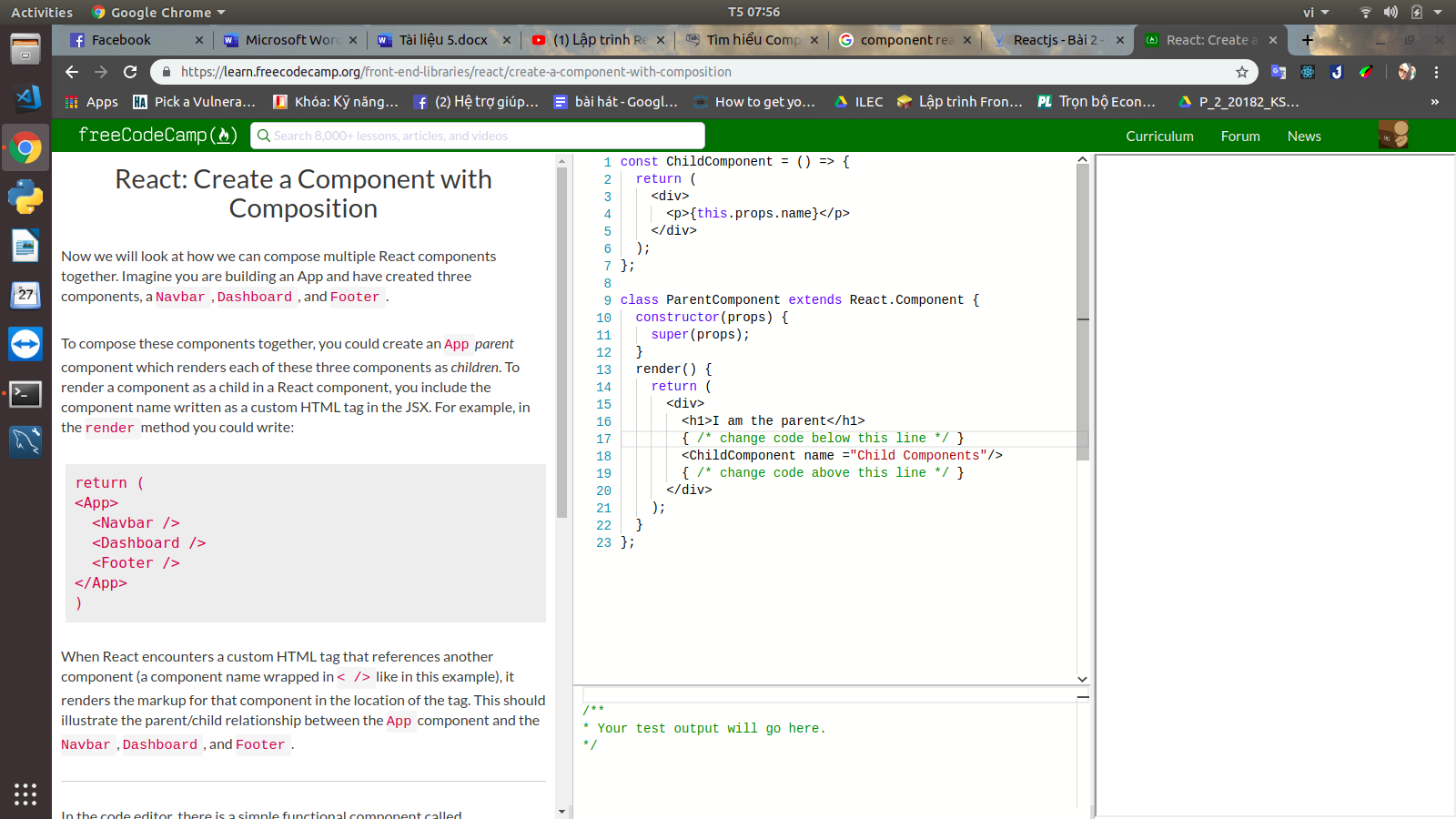
Compnent giúp ta reuse code và mantaince tốt với chỉ 1 component cho tất cả.



### 3.3. Props trong ReacJS

Props là thuộc tính của component. Với các thành phần được xây dựng dựa trên một components điều làm chúng trở nên khác biệt là cách ta truyền props vào khác nhau

Props được truyền từ bên ngoài vào và nó có thể là bất cử thứ gì (string, number, boolean, function, object,...)



### 3.4. State trong ReactJS

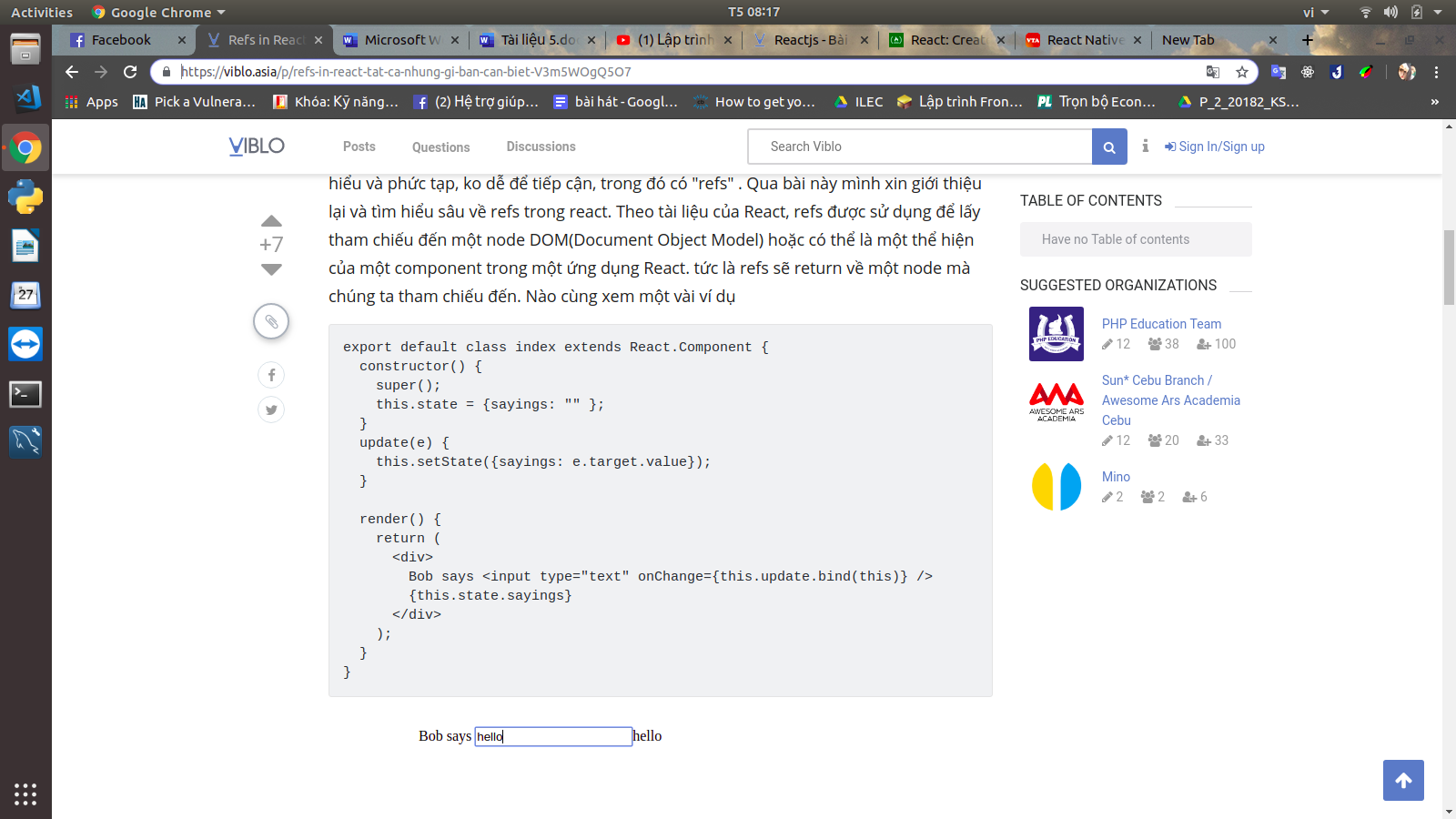
State là trạng thái của components. Ví dụ một state tên là name trong components app



Khi mới bắt đầu chúng ta dễ nhầm lẫn giữa hai khái niệm state và props vì công dụng của nó gần như tương đương. Tuy nhiên Props được truyền bên ngoài (outscope), còn state là trạng thái của riêng component và nó là private đối với components đó (inscope). Vậy nên muốn thay đổi state ta chỉ có thể thay thế bên trong nó.

### 3.5. Ref trong ReactJS

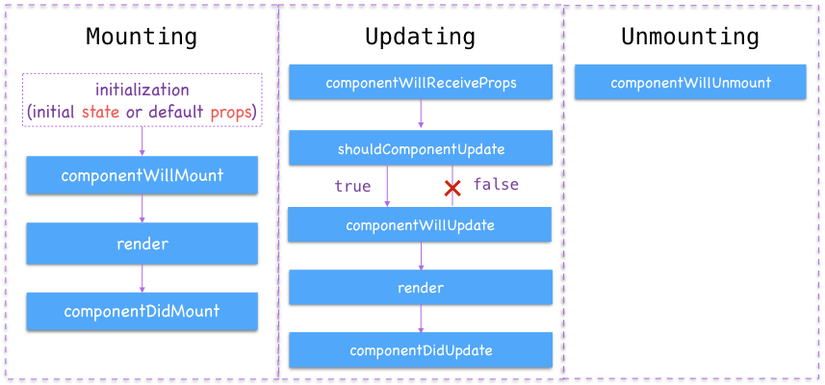
Ref tham chiếu tới một node DOM trong trang web. Vậy nó có gì hay ? Ứng dụng dễ thấy nhất là ta bắt sự kiện cho trang Web ReactJS



Do kĩ thuật Component nên sẽ có thể xảy ra trường hợp 2 node cùng id. Làm cách nào ta thay đổi node đó mà không làm ảnh hưởng tới node kia, câu trả lời đó là Refs.

### 3.6. Components Life Cycle (Vòng đời của components)

Như ta đã biết Components là khái niệm cốt lõi trong reactJS vậy nên việc nắm vòng đời của nó cụng quan trọng tương tự. Khi nào nó tạo ra? Khi nào nó kết thúc? Cái gì ảnh hưởng đến nó ?



- Mouting : Quá trình một component sinh ra

- Updating : Quá trình một component được update (khi thay đổi state, props..)

- Unmounting : Quá trình một component chết đi

Trong đây có một số hàm quan trọng cần nắm:

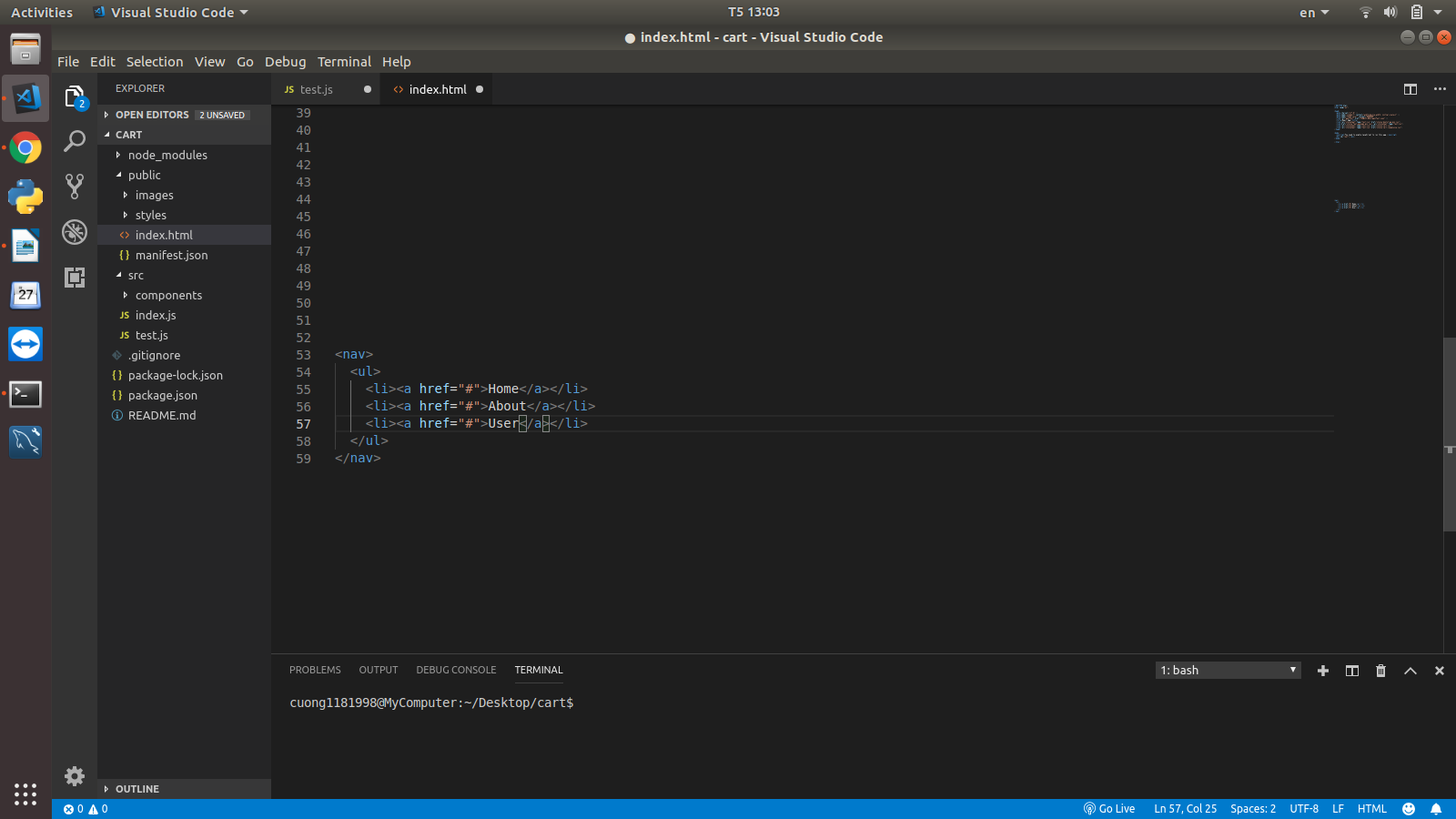
* Contructor() : Hàm khởi tạo components
* ComponentWillMount() : Làm gì đó trước khi render, cập nhật state, tính toán props
* Render() : Quá trình vẽ ra Vitrual DOM và hiển thị lên
* ComponentDidMount() : Làm gì đó sau khi render, fetch dữ liệu từ sever, viết mã JS tương tác với giao diện
* ComponentUpdate(): Là gì trước và sau update

Thứ tự thực hiện và các hàm được gọi sẽ là:

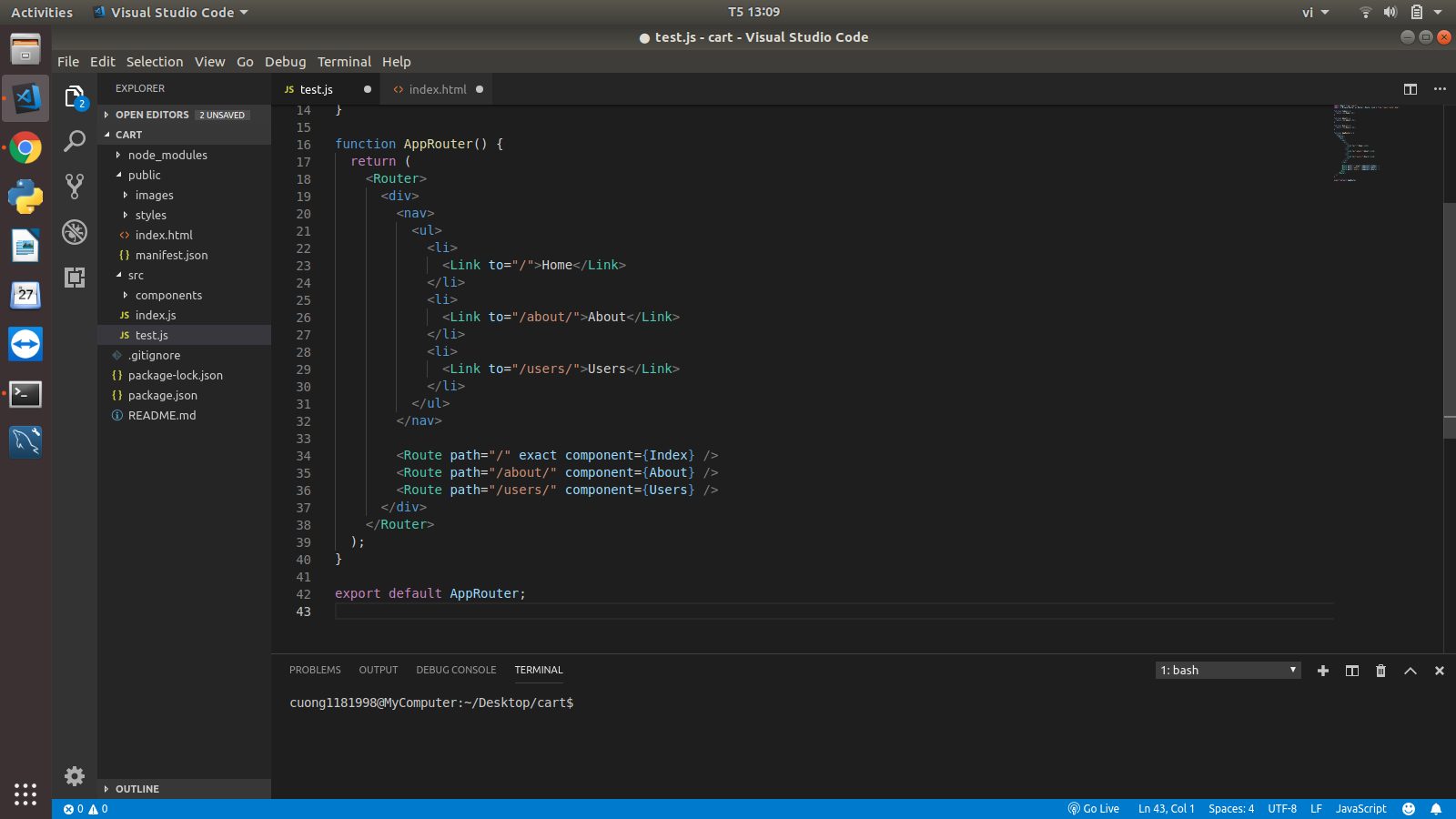
Contructor -> ComponentWillMount -> Render -> ComponentDidMount

### 3.7. React Router

Điều hướng là cách dẫn một link trang Web tới một link trang Web khác. Vậy điều hướng trong React có gì khác so với điều hướng truyền thống?



Khi ta click vào một đường link trình duyệt điều hướng ta đến một trang Web mới rất đơn gian, tuy nhiên nếu trang Web lớn với nhiều đường link hay ta muốn làm single page điều này không có nhiều lợi ích. Hơn nữa mỗi lần click ta lại phải load trang Web 1 lần nữa.



Cú pháp React có vẻ như là khá phức tạp, tuy nhiên nó lại hết sức hữu hiệu. Nó cho khả năng điều hướng không load trang tăng trải nghiệm người dùng.

Để là được điều đó ReactJS lưu các components có Link trong một History, Khi người dùng điều hướng ReactJS chỉ cần tìm trong History và lôi ra.

Để nắm rõ React Router ta cần tìm hiểu thêm về các khái niệm Route, Link, Match....

# 

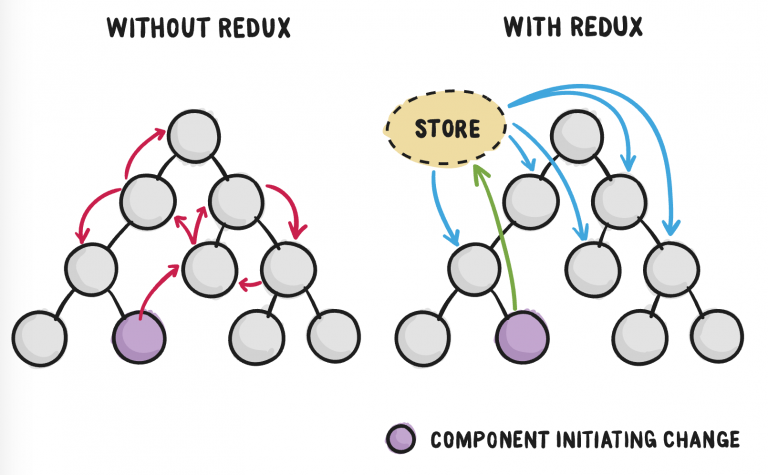
# Phần 2: Redux

Khi làm việc với React hay các dự án ứng dụng Single Page nói chung, có một vấn đề khá đau đầu là làm sao quản lý được trạng thái của ứng dụng đó**.**Redux js là một thư viện Javascript giúp tạo ra thành một lớp quản lý trạng thái của ứng dụng**.**

## 1. Nguyên lý

Redux được xây dựng dựa trên 3 nguyên lý:

* Nguồn dữ liệu tin cậy duy nhất: State của toàn bộ ứng được chứa trong một object tree nằm trong Store duy nhất
* Trạng thái chỉ được phép đọc: Cách duy nhất để thay đổi State của ứng dụng là phát một Action (là 1 object mô tả những gì xảy ra)
* Thay đổi chỉ bằng hàm thuần túy: Để chỉ ra cách mà State được biến đổi bởi Action chúng ta dùng các pure function gọi là Reducer



## 2. Cấu trúc

Về cơ bản Redux có 4 thành phần như sau:

* Action: Là nơi mang các thông tin dùng để gửi từ ứng dụng đến Store. Các thông tin này là 1 object mô tả những gì đã xảy ra.

export const INCREASE = 'INCREASE'

export function increase() {

return {

type: INCREASE

}

}

* Reducer: Là nơi xác định State thay đổi như thế nào.

export default function counterApp (state = initialState, action) {

switch (action.type) {

case INCREASE:

return {

increase: ++state.increase,

decrease: state.decrease

}

case DECREASE:

return {

increase: state.increase,

decrease: ++state.decrease

}

default:

return state

}

}

* Store: Là nơi quản lý State, cho phép truy cập State qua getState(), update State qua dispatch(action), đăng kí listener qua subscribe(listener).

import { createStore } from 'redux'

import counterApp from './reducers'

let store = createStore(counterApp)

* View: Hiển thị dữ liệu được cung cấp bởi Store

# Phần 3: API Spring Boot

## 1. Giới thiệu

Spring là một Framwork phát triển các ứng dụng Java được sử dụng bởi hàng triệu lập trình viên. Nó giúp tạo các ứng dụng có hiệu năng cao, dễ kiểm thử, tái sử dụng code…

Spring là một mã nguồn mở, được chia sẻ và có cộng đồng người dùng rất lớn.

Spring Boot là một Project nằm trên tầng IO Execution (tầng thực thi) của Spring IO Framework.

Spring Boot là một bước tiếp theo của Spring, để là cho Spring dễ dàng hơn trong việc thiết lập và phát triển ứng dụng. Với Spring Boot các cấu hình Spring được giảm thiểu tối đa. Spring Boot hỗ trợ các bộ chứa nhúng (embedded contrainers) điều này cho phép các ứng dụng web có thể chạy độc lập mà không cần phải triển khai lên các web server.

Ở bài báo cáo này, nhóm chúng em sẽ trình bày cụ thể việc sử dụng Spring Boot để xây dựng **Restful AP**I.

## 2. Cài đặt và chạy chương trình Spring Boot

### 2.1. Cài đặt Spring Tool Suite (STS)

Spring Tool Suite là một Plugin mở rộng hỗ trợ lập trình Spring trên IDE Eclipse.

Các bước cài đặt:

* B1: Vào Help chọn Eclipse Marketplace…
* B2: Nhập vào STS để tìm kiếm. Cài đặt Spring Tool Suite cho Eclipse.
* B3: Sau khi cài đặt xong khởi động lại IDE

### 2.2. Tạo Spring Boot Project và chạy chương trình

#### 2.2.1. Các bước tiến hành:

* B1: Chọn File/New/Other…
* B2: Đặt tên Name, Group, Arifact và Package cho Project
* B3: Do ta đang muốn xây dựng web bằng Sprin Boot nên ta chọn đầu vào “Web”
* B4: Chọn Next sau đó ấn Finish để kết thúc cài đặt

Ở các bước trên ta đã vừa tạo ra một Spring Boot Project, hãy xem nội dung file pom.xml, đây là file cấu hình các thư viện, các phụ thuộc của Project.

**(1) spring-boot-starter-parent**

spring-boot-starter-parent là một project sẵn có trong Spring Boot. Các thư viện phụ thuộc cơ bản đã được khai báo trong spring-boot-starter-parent, project của chúng ta chỉ cần kế thừa nó.

**(2) spring-boot-starter-web**

Các “Starter” khác chỉ đơn giản là cung cấp phụ thuộc mà chúng ta có khả năng cần thiết khi phát triển một loại hình cụ thể của ứng dụng. Chẳng hạn khi chúng ta phát triển một ứng dụng web, chúng ta cần một phụ thuộc spring-boot-starter-web.

**(3) spring-boot-maven-plugin**

spring-boot-maven-plugin là plugin cung cấp các thư viện cần thiết giúp project của bạn có thể chạy trực tiếp mà không cần triển khai một web server. Nó giúp tạo ra một file **jar** có thể thực thi (Execitable)

#### 2.2.2. Chạy chương trình Spring Boot Project

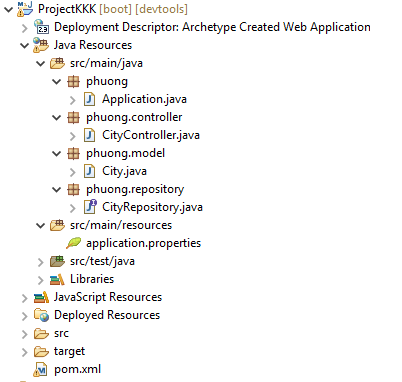
Khi ta tạo một “Spring Boot Web App”, có một class Application.java được tạo ra bởi công cụ “Spring Tool Suite”. Chạy class với tùy chọn “Spring Boot App”.

Khi này kết quả được phản hồi ở đường dẫn “localhost:8080”. Ở phần sau, xây dựng Restful API bằng Spring Boot ta sẽ rõ hơn về kết quả.

## 3. Xây dựng Restful API bằng Spring Boot

### 3.1. Thiết lập cấu hình các file

Khởi tạo một Spring Boot Project mới có các packge và các file như sau:



- package “phuong” chứa file thực thi chương trình Application.java

- package “phuong.controller” chứ file CityController.java, file này chịu trách nhiệm điều hướng và thực hiện các Request mà client gửi lên server

- package “phuong.model” chứa file City.java, file này cấu hình một thực thể xác định, ở đây là City

- package “phuong.repository” chứa file CityRepository.java đây là một interface được kết thừa từ class JpaRepository để chứa các phương thức như thêm, sửa, xóa dữ liệu, sau đó Controller có thể dùng.

- file appication.properties cấu hình cơ sở dữ liệu mà ta sử dụng. Cho ta biết ta đang sử dụng cơ sở dữ liệu nào, tài khoản, password của cơ sở dữ liệu muốn truy cập.

### 3.2. Giải thích nội dung các file

- Nội dung file pom.xml (đã được giải thích ở trên)

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0* [*http://maven.apache.org/xsd/maven-4.0.0.xsd*](http://maven.apache.org/xsd/maven-4.0.0.xsd)*"*>  <modelVersion>4.0.0</modelVersion>  <groupId>anhphuong</groupId>  <artifactId>ProjectKKK</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>  <name>spring-boot-rest-api-tutorial</name>  <description>Sample REST CRUD API project using Spring Boot, Mysql, JPA and Hibernate</description>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.5.3.RELEASE</version>  <relativePath/> <!-- lookup parent from repository -->  </parent>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  <java.version>1.8</java.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-actuator</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <scope>runtime</scope>  </dependency>  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.45</version>  <scope>runtime</scope>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

- Nội dung file properties.appilcation

|  |
| --- |
| ## Database Properties  spring.datasource.url = jdbc:mysql://localhost:3306/foodyv2?useSSL=false  spring.datasource.username = root  spring.datasource.password = 123456  ## Hibernate Properties  # The SQL dialect makes Hibernate generate better SQL for the chosen database  spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect  # Hibernate ddl auto (create, create-drop, validate, update)  spring.jpa.hibernate.ddl-auto = update |

* spring.datasource.url: đường dẫn đến cơ sở dữ liệu, ở đây là cơ sở dữ liệu: foodyv2
* username, password: là tên người dùng và mật khẩu của sql, mặc định theo máy từng người
* cơ sở dữ liệu foodyv2: chứ bảng **City** có hai thuộc tính là **id** và **name**

- Nội dung file City.java

|  |
| --- |
| **package** phuong.model;  **import** javax.persistence.Column;  **import** javax.persistence.Entity;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.GenerationType;  **import** javax.persistence.Id;  **import** javax.persistence.Table;  @Entity  @Table(name="city")  **public** **class** City {  @Id  @GeneratedValue(strategy=GenerationType.***AUTO***)  @Column(name = "id", nullable = **false**)  **private** Integer id;  @Column(name = "name", nullable = **false**)  **private** String name;  **public** City() {  **super**();  }  **public** City(Integer id, String name) {  **super**();  **this**.id = id;  **this**.name = name;  }  **public** Integer getId() {  **return** id;  }  **public** **void** setId(Integer id) {  **this**.id = id;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  } |

* @Entity

Là một **annotation** trong Spring Boot để giải thích City là một thực thể

* @Table(name="city")

Entity tham chiếu đến bảng City trong cơ sở dữ liệu

* @Id

@GeneratedValue(strategy=GenerationType.***AUTO***)

@Column(name = "id", nullable = **false**)

**private** Integer id;

@Id: thể hiện thuộc tính này là khóa chính trong bảng cơ sở dữ liệu

@GeneratedValue: Là một annotation trong JPA, khóa chính tự động được tăng lên khi ta insert thêm một bảnh ghi vào cơ sở dữ liệu

@Column(name = “id”, nullbale = false): Thuộc tính này tham chiếu đến cột “id” trong bảng cơ sở dữ liệu, nếu không có dòng này thì tên thuộc tính được mặc định là tên của cột trong bảng city.

- Nội dung file CityRepository.java

|  |
| --- |
| **package** phuong.repository;    **import** org.springframework.data.jpa.repository.JpaRepository;  **import** phuong.model.City;  **public** **interface** CityRepository **extends** JpaRepository<City, Integer> {  } |

* File định nghĩa một interface kế thừa class JpaRepository của Spring Boot. Class này chứa các thao tác thêm, sửa, xóa, hiển thị thực thể

- Nội dung file CityController.java

|  |
| --- |
| **package** phuong.controller;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.web.bind.annotation.DeleteMapping;  **import** org.springframework.web.bind.annotation.PathVariable;  **import** org.springframework.web.bind.annotation.PutMapping;  **import** org.springframework.web.bind.annotation.RequestBody;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RestController;  **import** phuong.model.City;  **import** phuong.repository.CityRepository;  @RestController  @RequestMapping("/api/city")  **public** **class** CityController {  @Autowired  **private** CityRepository cityRepository;  // lay tat ca  @RequestMapping("/getall")  **public** List<City> getAll(){  **return** (List<City>) cityRepository.findAll();  }  // lay theo id  @RequestMapping("/{id}")  **public** City getByID(@PathVariable("id") **int** id) {  System.***out***.println("Fetching Id" + id);  City c = cityRepository.findOne(id);  **if**(c == **null**) {  System.***out***.println("Id" + id + "not found");  }  **return** c;  }  // Them  @RequestMapping("/add")  **public** City add(@RequestBody City c) {  // ID tu tang nen se ko trung  System.***out***.println("Added:: " + c);  cityRepository.save(c);  **return** c;  }  // Xoa  @DeleteMapping("/delete/{id}")  **public** **void** deleteByID(@PathVariable("id") **int** id) {  City hn = cityRepository.findOne(id);  **if**(hn == **null**) {  System.***out***.println("Khong ton tai id. khong xoa duoc");  **return**;  }  cityService.delete(id);  System.***out***.println("delete id: " + id);  }  // Cap nhat  @PutMapping("/update")  **public** City update(@RequestBody City c) {  City hn = cityRepository.findOne(c.getId());  **if**(hn != **null**) {  System.***out***.println("update id" + hn.getId());  cityRepository.save(c);  **return** c;  }  System.***out***.println("Khong ton tai thanh pho");  **return** hn;  }  } |

* @RestController

Đây là nội dung của một file viết các Restful Api

* @RequestMapping("city")

Đường dẫn của các api đều chứa: “localhost:8080/city”

* @RequestMapping("/getall")

**public** List<City> getAll(){

**return** (List<City>) cityRepository.findAll();

}

Đường dẫn của api lấy ra tất cả các thành phố “localhost:8080/api/city/getall”

Hàm trả về một List các thành phố, cityRepository sử dụng các phương thức findAll(), findOne, delete… là các phương thức của JPA (thêm, sửa, xóa,….) trong Spring boot.

- Nội dung file Application.java

|  |
| --- |
| **package** phuong;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  /\*\*  \* The type Application.  \*  \* **@author** Nguyen Anh Phuong  \*/  @SpringBootApplication  **public** **class** Application {    /\*\*  \* The entry point of application.  \*  \* **@param** args the input arguments  \*/  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(Application.**class**, args);  }  } |

* @SpringBootApplication

Annotation thể hiện đây là một ứng dụng Spring. Khi chạy file này ta sẽ có kết quả trả về trên localhost:8080.

### 3.3. Kết quả

Kết quả lấy ra tất cả các thành phố trong bảng City bằng api: “localhost:8080/api/city/getall”.

# Phần 4: Kết hợp ReactJS với JavaSpring

Trong báo cáo này sẽ đề cập đến việc kết hợp Spring và ReactJs bằng 2 cách là fetch và axios

## 1 . Fetch

Fetch API là một API đơn giản cho việc gửi và nhận request bằng js. Với fetch thì việc thực hiện các yêu cầu web và xử lý phản hồi dễ dàng hơn so với XMLHttpRequest cũ.

Phương thức fetch() trả về một promise có trạng thái resolves với giá trị là response cho request đó. Khi promise resolves, response được truyền tới .then. Đây là nơi mà có thể sử dụng response. Nếu request không thành công, thì sẽ chuyển tới .catch với tham số là lỗi tương ứng. Trong trường hợp có nhiều .then thì sẽ thực hiện lần lượt các .then.

## 2. Axios

Axios là một HTTP client được viết dựa trên Promises được dùng để hỗ trợ cho việc xây dựng các ứng dụng API từ đơn giản đến phức tạp và có thể được sử dụng cả ở trình duyệt hay Node.js.

Cài đặt : tìm hiểu theo từ khóa axios.

## 3. So sánh axios và fetch

Khi sử dụng Fetch, nếu khi server trả về các mã lỗi 4xx hay 5xx, thì hàm catch() của bạn sẽ không được gọi đến và người lập trình viên sẽ có nhiệm vụ phải tự kiểm tra trạng thái của mã trả về để xác định xem liệu request đó có thành công hay không. Trong khi đó, **Axios** sẽ reject tất cả các promise của request nếu một trong các mã lỗi trên được trả về.

**Fetch** không tự động gủi trả cookies về cho server khi tạo một request. Ta sẽ cần phải truyền một cách trực tiếp các option để cho cookies có thể được include. Còn với **Axios** thì bạn không hề phải lo về vấn đề này.

**Axios** có thể được xử dụng ở cả browser và Node.js và chính điều này đã tạo cơ hội cho việc chia sẻ code Javascript giữa các trình duyệt và phần back-end hay việc thực hiện render cho ứng dụng front-end của bạn ở phía server

# Phần 5. Các biểu đồ của trang web

## 1. Biểu đồ Usecase tổng quan

A picture containing text, map

Description automatically generated

## 2. Biểu đồ cơ sở dữ liệu

A screenshot of a cell phone

Description automatically generated

# Phần 6. Tài liệu tham khảo

Link github của web site xây dựng được:

<https://github.com/duytien241/project2_webbanhang>

Phần 1:

Phần 2:

Phần 3:

<https://www.youtube.com/watch?v=lP5hDq0rVjA>

<https://o7planning.org/vi/11645/vi-du-crud-restful-web-service-voi-spring-boot>

Phần 4: