**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**⬩⬩🙠🕮🙢⬩⬩**



**CÔNG NGHỆ PHẦN MỀM MỚI**

**XÂY DỰNG WEBSITE BÁN DỤNG CỤ THỂ THAO VỚI CÔNG NGHỆ MEAN STACK**

**GVHD: ThS. Lê Vĩnh Thịnh**

**Sinh viên thực hiện MSSV**

**Trần Tấn Quý 16110196**

**Phạm Hoàng Việt 16110547**

**Thành phố Hồ Chí Minh, tháng 11 năm 2019**

**NHẬN XÉT CỦA GIẢNG VIÊN**

Chữ ký

**ThS. Lê Vĩnh Thịnh**

**MỤC LỤC**

**DANH SÁCH TỪ VIẾT TẮT**

**DANH SÁCH HÌNH ẢNH**

**DANH SÁCH BẢNG**

# **NỘI DUNG**

# ***1. Mô tả đồ án***

## **Mục tiêu**

## **1.2. Yêu cầu người dùng**

## **1.3. Lược đồ usecase**

## **1.4. Mô tả usecase**

**2. Thiết kế hệ thống**

## **2.1. Công nghệ**

***2.1.1. OAuth***

**2.1.1.1. OAuth là gì?**

OAuth là một phương thức chứng thực các ứng dụng có thể chia sẻ tài nguyên với nhau mà không cần chia sẻ các thông tin username và mật khẩu. Trong OAuth, O tượng trưng cho ‘Open’ và ‘Auth’ tượng trưng cho Authentication (xác thực người dùng) & Authorization (ủy quyền cho ứng dụng truy cập tài nguyên của người dùng).

OAuth có 2 phiên bản là OAuth 1.0 và OAuth 2.0, trong đó OAuth 2.0 được sử dụng rất phổ biến trong các ứng dụng/website công nghệ thông tin ngày nay. Một số thuật ngữ trong OAuth 2.0:

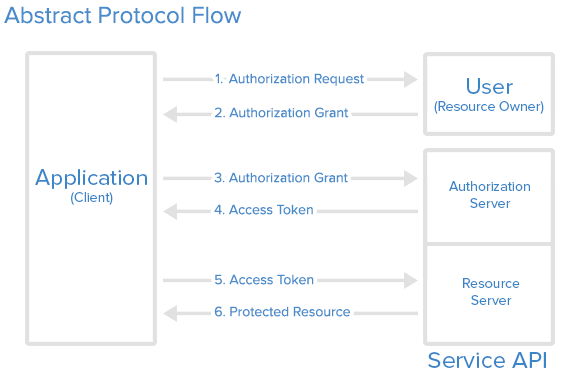
- **Resource owner:** có thể hiểu là người dùng.

**- Client:** ứng dụng của bên thứ 3. VD: một web app sử dụng Google API.

**- Authorization server:** là một đối tượng cấp quyền cho client để truy cập vào dữ liệu của server.

**- Resource server:** nơi chứa thông tin, dữ liệu cần chia sẻ.

**2.1.1.2. Cách hoạt động của OAuth 2.0**



Hình 2. 1. Luồng hoạt động của OAuth 2.0 [1]

Luồng hoạt động có thể được tóm gọn thành 3 luồng chính:

- Client gửi yêu cầu authorization đến cho user (resource owner), nếu user chấp nhận thì sẽ tiến hành cấp quyền cho client.

- Sau khi có authorization grant, client tiến hành gửi nó đến Authorization Server và sau khi xác nhận xong, server sẽ trả về một Access Token.

- Client sẽ sử dụng access token trong suốt quá trình làm việc với resource server. Resource server đóng vai trò kiểm tra tính hợp lệ của token mỗi khi nó được gửi lên và trả về dữ liệu cho client.

Trong OAuth, với các thông tin được gửi đi sẽ luôn đi kèm với 2 tham số quan trọng nhất:

- Client identifier (ClientID): là một chuỗi ký tự được sử dụng bởi service api để định danh client.

- Client secret: là chuỗi ký tự dùng để xác thực định danh ứng dụng khi ứng dụng yêu cầu truy cập vào thông tin tài khoản của user. Đây là chuỗi được giữ bí mật giữa client và authorization server.

**2.1.1.3. Ứng dụng của OAuth**

OAuth được ứng dụng nhiều nhất trong việc đăng nhập bằng cách sử dụng credential của các bên thứ ba. Nó giúp cho việc tiếp cận đến một trang web dễ dàng và nhanh chóng hơn mà không cần phải tiến hành đăng ký một tài khoản với nhiều bước rườm rà từ đó có quá nhiều tài khoản cần phải ghi nhớ bởi người dùng.

Những doanh nghiệp lớn như Facebook, Google, Twitter, Github,... là những doanh nghiệp cho phép người dùng sử dụng API để thiết lập OAuth cho riêng mình.

**2.1.2. PassportJS**

**2.1.2.1. PassportJS là gì?**

Passport.js là một module của NodeJS được ứng dụng trong việc authentication cho ứng dụng. Nó linh hoạt ở chỗ chúng ta có thể sử dụng để xác thực bằng nhiều trường hợp khác nhau như Facebook, Google, … hay dựa vào Token.



**2.1.2.2. Các hàm và middleware trong Passport.js**

- passport.initialize: kiểm tra session lấy ra passport.user nếu chưa có thì tạo rỗng.  
 - passport.session: sử dụng session lấy thông tin user rồi gắn vào req.user.  
 - passport.deserializeUser: giúp ta lấy dữ liệu user dựa vào thông tin lưu trên session và gắn vào req.user.  
 - passport.authenticate: middleware giúp ta gắn kịch bản local vào route.  
 - passport.serializeUser: hàm được gọi khi xác thực thành công để lưu thông tin user vào session

**2.1.2.3. Các bước xác thực tài khoản sử dụng Passport.js**

- Khi người dùng submit form đăng nhập, một request POST được tạo ra khởi chạy middleware passport.authenticate.

- Kịch bản Local đã được thiết lập cho passport.authenticate nên nó sẽ gọi đến cái kịch bản ta đã cài đặt.

- Nó lấy dữ liệu req.body.username và req.body.passport rồi gán cho hàm verify local.

- Truy vấn database rồi kiểm tra xem passport của người dùng đưa lên có đúng không.

- Trong trường hợp Error từ database ta gọi đến callback là done với param là err( done(err)). Khi mà nó không tìm thấy được người dùng hợp lý ta gọi đến done(null,false). Còn nếu thông tin đăng nhập đúng ta gọi done(null,user).

- Khi callback done được gọi, nó sẽ lấy dữ liệu err, user và dữ liệu bạn custom thêm nếu có trả lại cho passport.authenticate.

- Nếu dữ liệu trả về của callback là null, true, xác thực thành công passport tiếp tục gọi hàm req.login( cái này tự gắn vào từng request khi bạn cài đặt passport ở bước đầu tiên)

- Hàm req.login gọi đến thằng passport.serializeUser mà ta đã định nghĩa trước đó. Hàm này truy cập vào đối tượng user mà ta trả về cho middleware passport.authenticate và xác định xem thành phần nào của đối tượng sẽ lưu vào trong session. Kết quả của hàm này là ta sẽ có đối tượng req.session.passport.user = các thông tin ta truyền vào trong serializeUser. Trong ví dụ bên trên thì nó là [user.id](http://user.id/).

- Đồng thời với trên passport cũng có gắn thông tin user vào req.user.

- Việc xác thực kết thục, hàm requestHandler sẽ được gọi đưa chúng tra đến trang đã thiết lập.

**2.1.2.4. Một số hàm hỗ trợ cho request**

- req.login(): thiết lập phiên đăng nhập.

- req.logout(): đăng xuất khỏi phiên đăng nhập.

- req.isAuthenticated(): kiểm tra user đã được xác thực hay chưa. Hàm trả về true nếu user đã đăng nhập.

- req.isUnauthenticated()

**MEAN STACK**

***2.1.3.* MongoDB**

**2.1.3.1. MongoDB là gì?**



MongoDB là một hệ quản trị cơ sở dữ liệu mã nguồn mở thuộc học NoSQL. Nó được thiết kế theo kiểu hướng đối tượng, các bảng trong MongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ trên bảng không cần tuân theo một cấu trúc nhất định nào cả (điều này rất thích hợp để làm big data). MongoDB lưu trữ dữ liệu theo hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON nên truy vấn sẽ rất nhanh.

**2.1.3.2. Ưu điểm của MongoDB**

- Schema linh hoạt: Do MongoDB sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi một collection sẽ các các kích cỡ và các document khác nhau.

- Cấu trúc đối tượng rõ ràng: Tuy rằng cấu trúc của dữ liệu là linh hoạt nhưng đối tượng của nó được xác định rất rõ ràng.

- Sử dụng bộ nhớ nội tại, nên truy vấn sẽ rất nhanh.

- MongoDB rất dễ mở rộng. Không có các join: Điều này cũng góp phần tạo nên tốc độ truy vấn cực nhanh trên mongoDB.

- MongoDB phù hợp cho các ứng dụng realtime.

**2.1.3.3. Nhược điểm của MongoDB**

- Điều đầu tiên phải kể đến ở đây là MongoDB không có các tính chất ràng buộc như trong RDBMS nên khi thao tác với mongoDB thì phải hết sức cẩn thận.

- MongoDB sử dụng sẽ hao tốn tài nguyên của hệ thống nhiều hơn RDBMS. Nhưng đến thời điểm hiện tại thì vấn đề này không còn là điều lo ngại.

**2.1.4. ExpressJS**

**2.1.4.1. ExpressJS là gì?**



ExpressJS là một framework được xây dựng trên nền tảng của NodeJS. Đây là một framework cung cấp nhiều tính năng mạnh mẽ trên nền tảng web cũng như các ứng dụng di động, nó hỗ trợ các phương thức HTTP và middleware để tạo ra một API vô cùng mạnh mẽ và dễ sử dụng.

**2.1.4.2. Ưu điểm của ExpressJS**

- Giúp cho việc phát triển web application dễ dàng và nhanh chóng.

- Dễ dàng cấu hình và tùy chỉnh.

- Cho phép người dùng định nghĩa các routes cho ứng dụng dựa trên các phương thức HTTP và URLs.

- Cho phép người dùng khởi tạo các REST API server.

- Dễ dàng kết nối với các cơ sở dữ liệu như MongoDB, MySQL,…

- Bao gồm nhiều middleware modules cho phép người dùng sử dụng để thực thi các task trên request và response.

**2.1.4.3. Nhược điểm của ExpressJS**

- ExpressJS chỉ phù hợp cho các dự án vừa và nhỏ.

- Tốn nhiều công sức do người dùng cần phải tạo tất cả các endpoints

- Khi codebase phát triển lớn hơn thì việc refactor cũng trở nên khó khăn hơn với ExpressJS

**2.1.5. AngularJS**

**2.1.5.1. Angular JS là gì?**



AngularJS là một framework có cấu trúc cho các ứng dụng web động. Nó cho phép người dùng sử dụng HTML như một ngôn ngữ mẫu và cho phép người dùng mở rộng cú pháp của HTML để diễn đạt các thành phần ứng dụng một cách rõ ràng và súc tích. Hai tính năng cốt lõi: Data binding và Dependency injection của AngularJS loại bỏ phần lớn code mà người dùng thường phải viết. Nó xảy ra trong tất cả các trình duyệt, làm cho nó trở thành đối tác lý tưởng của bất kỳ công nghệ Server nào [2].

**Tính năng chính của Angular JS**

**- Data-binding**: Tự động đồng bộ hóa dữ liệu giữa Model và View.

**- Module**: dùng để định nghĩa một ứng dụng AngularJS. Module giống như một chiếc hộp để chứa các thành phần khác của ứng dụng bên trong nó. Module còn được gọi là một container của ứng dụng.

**- Scope**: Là những đối tượng hướng đến model. Nó là cầu nối giữa Controller và View.

**- Controller**: Là những hàm JavaScrip xử lí kết hợp với Scope.

**- Service**: AngularJS sử dụng các API được xây dựng từ các web service để thao tác với Database.

**- Directives**: Đánh dấu vào các yếu tố của DOM, nghĩa là sẽ tạo ra các thẻ HTML tùy chỉnh.

**- Template**: Các rendered view với các thông tin từ controller và model có thể sử dụng trong các file riêng biệt hoặc với view với trang sử dụng Partials.

**- Routing**: Chuyển đổi giữa các action trong Controller.

**- Model View Controller**: Mô hình chia thành phần riêng biệt thành Model, View, Controller. Đây là một mô hình khá hay nhưng trong Angular thì nó được chế biến lại một chút thành Model View View Model.

**- Dependency Injection**: Giúp các nhà phát triển tạo ứng dụng dễ dàng hơn.

**2.1.5.2. Ưu điểm của Angular JS**

**-** AngularJS có khung dựa theo MVC và mã nguồn mở.

**-** Sử dụng Injection Dependency

**-** Dễ dàng mở rộng và tùy chỉnh.

**-** Ngôn ngữ HTML và JavaScript thuần tý.

**-** Developer sẽ viết ít code nhưng lại chức năng lại đầy đủ hơn với Angular JS.

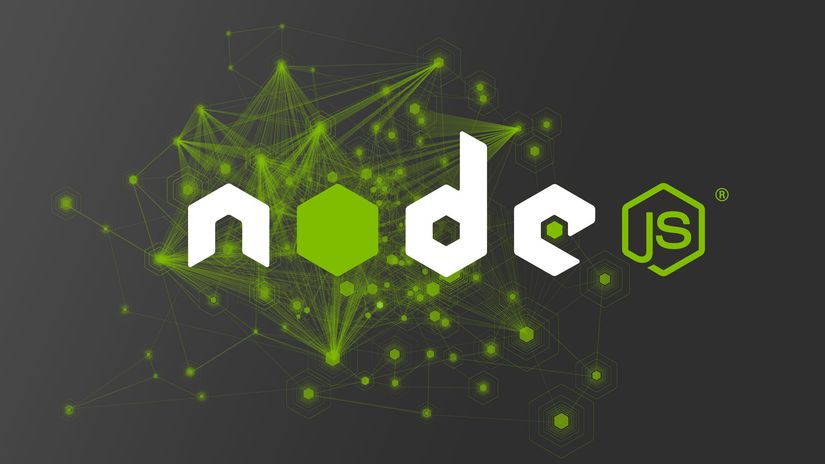
**-** REST services được hỗ trợ đầy đủ

**2.1.5.3. Nhược điểm của Angular JS**

**-** Do bản thân là một JavaScript framework, ứng dụng được viết bởi Angular JS có độ an toàn được xem là thấp do đó cần phải có các tính năng bảo mật và xác thực từ phía server để nâng cao độ tin cậy.

**-** Khi người dùng vô hiệu hóa JavaScript thì họ không thể thấy gì khác ngoài một giao diện cơ bản.

**2.1.6. NodeJS**

**2.1.6.1. NodeJS là gì?**

- Nodejs là một nền tảng (Platform) phát triển độc lập được xây dựng ở trên Javascript Runtime của Chrome mà chúng ta có thể xây dựng được các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng.

- Nodejs được xây dựng và phát triển từ năm 2009, bảo trợ bởi công ty Joyent, trụ sở tại California, Hoa Kỳ.

- Phần Core bên dưới của Nodejs được viết hầu hết bằng C++ nên cho tốc độ xử lý và hiệu năng khá cao.

- Nodejs tạo ra được các ứng dụng có tốc độ xử lý nhanh, realtime thời gian thực.

- Nodejs áp dụng cho các sản phẩm có lượng truy cập lớn, cần mở rộng nhanh, cần đổi mới công nghệ, hoặc tạo ra các dự án Startup nhanh nhất có thể.

**2.1.6.2. Ưu điểm của NodeJS**

- Node.js có tốc độ rất nhanh.

- Các ứng dụng Nodejs được viết bằng **javascript**, ngôn ngữ này là một ngôn ngữ khá thông dụng.

- Nodejs chạy đa nền tảng phía Server, sử dụng kiến trúc hướng sự kiện Event-driven, cơ chế non-blocking I/O làm cho nó nhẹ và hiệu quả.

- Có thể chạy ứng dụng Nodejs ở bất kỳ đâu trên máy Mac – Window – Linux, hơn nữa cộng đồng Nodejs rất lớn và hoàn toàn miễn phí.

- Các ứng dụng NodeJS đáp ứng tốt thời gian thực và chạy đa nền tảng, đa thiết bị.

**2.1.6.3. Nhược điểm của NodeJS**

**-** Ứng dụng nặng tốn tài nguyên

- Việc triển khai Node.js trên host không phải là điều dễ dàng.

- Node.js đó là nó vẫn đang trong giai đoạn phát triển ban đầu, điều này có nghĩa là một số đặc trưng sẽ thay đổi trong quá trình phát triển tiếp theo.

## **2.2. Thiết kế kiến trúc**

## **2.3. Thiết kế giao diện**

## **2.4. Class diagram**

## **2.5. Sequence diagram**

## **2.6. Mô tả cơ sở dữ liệu MongoDB**

## **2.7. Configuration**

.

# ***3. Cài đặt đồ án***

## **3.1. Môi trường**

## **3.2. Phân công công việc**

## **3.3. Kế hoạch công việc**

# ***4. Hướng dẫn sử dụng***

# ***5. Tổng kết***

## **5.1. Kết quả**

|  |  |
| --- | --- |
| **Áp dụng Google OAuth cho website** |  |
| **Thiết kế giao diện với AngularJS** |  |
| **1. Giao diện trang chủ** | **C:\Users\User\Desktop\home.png** |
| **2. Giao diện sản phẩm** | **C:\Users\User\Desktop\product.png** |
| **3. Giao diện đăng nhập** | **C:\Users\User\Desktop\login.png** |
| **4. Giao diện giỏ hàng** | **C:\Users\User\Desktop\cart.png** |
| **5. Giao diện chi tiết sản phẩm** | **C:\Users\User\Desktop\detail.png** |
| **6. API get dữ liệu product** |  |

## **5.2. Khó khăn**

## **5.3. Giải pháp**

## **5.4. Ưu điểm**

## **5.5. Nhược điểm**

## **5.6. Hướng phát triển**

## **5.7. Source code**

# **TÀI LIỆU THAM KHẢO**

[1] Viblo, OAuth là gì và tại sao nên sử dụng nó?, <https://viblo.asia/p/oauth-la-gi-va-tai-sao-nen-su-dung-no-1Je5E4n4lnL>, truy cập ngày 18 tháng 10 năm 2019.

[2] Viblo, AngularJS là gì?, <https://viblo.asia/p/angularjs-toan-tap-phan-1-angularjs-la-gi-MgNvWYZEeYx>, truy cập ngày 8 tháng 11 năm 2019