**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

🞠◊🞠◊🞠◊🞠◊🞠



**Tiểu luận chuyên ngành**

**TÌM HIỂU VÀ XÂY DỰNG ỨNG DỤNG SPRING RESTFUL**

**GVHD:** **NGUYỄN MINH ĐẠO**

**NHÓM THỰC HIỆN:**

**1. Bùi Phan Viết Cường 15110173**

**2. Phạm Ngọc Huy 15110218**

**TP.HỒ CHÍ MINH – 01/2019**

ĐH SƯ PHẠM KỸ THUẬT TP.HCM XÃ HỘI CHỦ NGHĨA VIỆT NAM KHOA CNTT Độc lập – Tự do – Hạnh Phúc \*\*\*\*\*\* \*\*\*\*\*\*\*

**PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỠNG DẪN**

Họ và tên Sinh viên 1 : .......... Bùi Phan Viết Cường........ MSSV 1: .....15110173......  
Họ và tên Sinh viên 2 : .......... Phạm Ngọc Huy................ MSSV 2: .....15110218......  
Ngành: Công nghệ Thông tin  
Tên đề tài: TÌM HIỂU VÀ XÂY DỰNG ỨNG DỤNG SPRING RESTFUL.  
Họ và tên Giáo viên hướng dẫn: ................ NGUYỄN MINH ĐẠO ............................   
**NHẬN XÉT**1. Về nội dung đề tài khối lượng thực hiện:  
....................................................................................................................................................  
....................................................................................................................................................  
....................................................................................................................................................  
....................................................................................................................................................  
....................................................................................................................................................  
2. Ưu điểm:  
....................................................................................................................................................   
....................................................................................................................................................   
....................................................................................................................................................  
....................................................................................................................................................  
3. Khuyết điểm  
....................................................................................................................................................   
....................................................................................................................................................  
....................................................................................................................................................  
4. Đề nghị cho bảo vệ hay không ?  
5. Đánh giá loại :  
6. Điểm :  
 *Tp. Hồ Chí Minh, 4 ngày 1 tháng năm 2019* Giáo viên hướng dẫn  
 *(Ký & ghi rõ họ tên)*

ĐH SƯ PHẠM KỸ THUẬT TP.HCM XÃ HỘI CHỦ NGHĨA VIỆT NAM KHOA CNTT Độc lập – Tự do – Hạnh Phúc \*\*\*\*\*\* \*\*\*\*\*\*\*

**PHIẾU NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN**

Họ và tên Sinh viên 1 : ..........Bùi Phan Viết Cường........ MSSV 1: .....15110173.......  
Họ và tên Sinh viên 2 : ..........Phạm Ngọc Huy................ MSSV 2: .....15110218.......   
Ngành: Công nghệ Thông tin  
Tên đề tài: TÌM HIỂU VÀ XÂY DỰNG ỨNG DỤNG SPRING RESTFUL.  
Họ và tên Giáo viên phản biện: ................... LÊ VĂN VINH .......................................   
**NHẬN XÉT**1. Về nội dung đề tài khối lượng thực hiện:  
....................................................................................................................................................  
....................................................................................................................................................  
....................................................................................................................................................  
....................................................................................................................................................  
....................................................................................................................................................  
2. Ưu điểm:  
....................................................................................................................................................   
....................................................................................................................................................   
....................................................................................................................................................  
....................................................................................................................................................  
3. Khuyết điểm  
....................................................................................................................................................   
....................................................................................................................................................  
....................................................................................................................................................  
4. Đề nghị cho bảo vệ hay không ?  
5. Đánh giá loại :  
6. Điểm :  
 *Tp. Hồ Chí Minh*, *4 ngày 1 tháng năm 2019* Giáo viên phản biện  
 *(Ký & ghi rõ họ tên)*

MỤC LỤC

[MỤC LỤC](#_Toc534161245)

[DANH MỤC BẢNG](#_Toc534161246)

[DANH MỤC HÌNH ẢNH](#_Toc534161247)

[GIỚI THIỆU ĐỀ TÀI 1](#_Toc534161248)

[1. Tính cấp thiết của đề tài (Lý do chọn đề tài) 1](#_Toc534161249)

[2. Mục tiêu đề tài 1](#_Toc534161250)

[3. Đối tượng và phạm vi nghiên cứu 2](#_Toc534161251)

[4. Phương pháp nghiên cứu 2](#_Toc534161252)

[5. Cấu trúc và nội dung đề tài 3](#_Toc534161253)

[CHƯƠNG 1: KHẢO SÁT HIỆN TRẠNG 4](#_Toc534161254)

[1.1. Hiện trạng 4](#_Toc534161255)

[1.2. Khảo sát 5](#_Toc534161256)

[1.2.1. Diễn đàn Viblo 5](#_Toc534161257)

[1.2.2. Diễn đàn Stackoverflow 7](#_Toc534161258)

[1.2.3. Diễn đàn Github Community 9](#_Toc534161259)

[1.2.4. Diễn đàn Kipalog 11](#_Toc534161260)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 13](#_Toc534161261)

[2.1. Tìm hiểu về Web Service 13](#_Toc534161262)

[2.1.1. Khái niệm về Web Service 13](#_Toc534161263)

[2.1.2. Đặc điểm của Web Service 13](#_Toc534161264)

[2.1.3. Ưu điểm của Web Service 13](#_Toc534161265)

[2.1.4. Nhược điểm của Web Service 14](#_Toc534161266)

[2.1.5. Kiến trúc của Web Service 14](#_Toc534161267)

[2.1.6. Các thành phần của Web Service 15](#_Toc534161268)

[2.2. Tìm hiểu về RESTful Web Service và RESTful API 19](#_Toc534161269)

[2.2.1. Tổng quan về RESTful Web Service 19](#_Toc534161270)

[2.2.2. Tổng quan về API 20](#_Toc534161271)

[2.2.3. Định nghĩa RESTful API 21](#_Toc534161272)

[2.2.4. Nhu cầu và mục đích sử dụng 21](#_Toc534161273)

[2.2.5. Quy tắc chuẩn trong thiết kế RESTful API 22](#_Toc534161274)

[2.2.6. Các thành phần trong RESTful API 24](#_Toc534161275)

[2.2.7. Cách làm việc với RESTful API 26](#_Toc534161276)

[2.2.8. Ưu nhược điểm của RESTful API 26](#_Toc534161277)

[2.3. Tìm hiểu RESTful API trong ứng dụng Spring Boot 27](#_Toc534161278)

[2.3.1. Khái niệm về ứng dụng Spring Boot 27](#_Toc534161279)

[2.3.2. Spring Boot REST 27](#_Toc534161280)

[2.3.3. RESTful API trong mô hình ba lớp Spring Boot 29](#_Toc534161281)

[2.3.4. RESTful API phía Backend 30](#_Toc534161282)

[CHƯƠNG 3: PHÂN TÍCH YÊU CẦU ỨNG DỤNG 32](#_Toc534161283)

[3.1. Sơ đồ Use Case 32](#_Toc534161284)

[3.2. Phân tích nghiệp vụ 37](#_Toc534161285)

[3.2.1. Nghiệp vụ Đăng ký 37](#_Toc534161286)

[3.2.2. Nghiệp vụ Đăng nhập 38](#_Toc534161287)

[3.2.3. Nghiệp vụ Tạo bài viết mới 40](#_Toc534161288)

[3.2.4. Nghiệp vụ Đánh giá bài viết 40](#_Toc534161289)

[3.2.5. Nghiệp vụ Lưu trữ bài viết (Bookmark) 42](#_Toc534161290)

[3.2.6. Nghiệp vụ Bình luận 42](#_Toc534161291)

[3.2.7. Nghiệp vụ Báo cáo bài viết 44](#_Toc534161292)

[3.2.8. Nghiệp vụ Theo dõi tag/ Bỏ theo dõi tag 44](#_Toc534161293)

[3.2.9. Nghiệp vụ Duyệt bài viết 46](#_Toc534161294)

[3.2.10. Nghiệp vụ Gán vai trò cho người dùng 47](#_Toc534161295)

[CHƯƠNG 4: THIẾT KẾ CƠ SỞ DỮ LIỆU 49](#_Toc534161296)

[4.1. Lược đồ ERD 49](#_Toc534161297)

[4.2. Bảng mô tả thuộc tính 49](#_Toc534161298)

[CHƯƠNG 5: CÀI ĐẶT ỨNG DỤNG 60](#_Toc534161299)

[5.1. Ngôn ngữ lập trình 60](#_Toc534161300)

[5.1.1. Ngôn ngữ lập trình Java 60](#_Toc534161301)

[5.1.2. Ngôn ngữ lập trình TypeScript 60](#_Toc534161302)

[5.1.3. Ngôn ngữ truy vấn HQL trong JPA Data 61](#_Toc534161303)

[5.2. Cài đặt ứng dụng phía backend 61](#_Toc534161304)

[5.2.1. Khởi tạo ứng dụng Spring Boot 61](#_Toc534161305)

[5.2.2. Thêm các dependency cho ứng dụng 63](#_Toc534161306)

[5.2.3. Cấu hình application.properties cho ứng dụng Spring Boot 64](#_Toc534161307)

[5.2.4. Cấu hình CORS cho ứng dụng làm việc với API 65](#_Toc534161308)

[5.2.5. Cấu hình Spring Security 67](#_Toc534161309)

[5.3. Cài đặt ứng dụng phía Front-end 69](#_Toc534161310)

[5.3.1. Cài đặt Yarn quản lý các thư viện node package 69](#_Toc534161311)

[5.3.2. Khởi tạo ứng dụng Angular 6 70](#_Toc534161312)

[5.3.4. Thêm các thư viện cần thiết cho ứng dụng trong package.json 72](#_Toc534161313)

[5.3.5. Cấu hình preload cho angular.json 73](#_Toc534161314)

[CHƯƠNG 6: QUẢN LÝ MÃ NGUỒN VÀ PHIÊN BẢN 74](#_Toc534161315)

[6.1. Tạo dự án tại Github 74](#_Toc534161316)

[6.2. Khởi tạo dự án 75](#_Toc534161317)

[6.3. Làm việc với dự án 75](#_Toc534161318)

[CHƯƠNG 7: XÂY DỰNG RESTFUL API CHO ỨNG DỤNG 78](#_Toc534161319)

[7.1. Mô hình Service trong ứng dụng 78](#_Toc534161320)

[7.2. Cấu trúc REST Controller trong ứng dụng 79](#_Toc534161321)

[7.2. Xác thực trong ứng dụng 81](#_Toc534161322)

[CHƯƠNG 8: KIỂM THỬ ỨNG DỤNG 84](#_Toc534161323)

[8.1 Kế hoạch kiểm thử chung 84](#_Toc534161324)

[8.2. Chi tiết testcase đã kiểm thử 84](#_Toc534161325)

[8.2.1. Kiểm thử vai trò người dùng 84](#_Toc534161326)

[8.2.2. Kiểm thử vai trò Admin 88](#_Toc534161327)

[8.2.3. Kiểm thử vai trò Moderator 89](#_Toc534161328)

[KẾT LUẬN 91](#_Toc534161329)

[1. Kết quả đạt được 91](#_Toc534161330)

[2. Kinh nghiệm rút ra từ quá trình thực hiện đề tài 91](#_Toc534161331)

[3. Hướng phát triển cho đề tài 92](#_Toc534161332)

[TÀI LIỆU THAM KHẢO 93](#_Toc534161333)

**DANH MỤC BẢNG**

[**Bảng 2.1:** Danh mục MINE phổ biến sử dụng với REST Service 24](#_Toc534161449)

[**Bảng 4.1:** Bảng mô tả thuộc tính bảng user 49](#_Toc534161450)

[**Bảng 4.2:** Bảng mô tả thuộc tính bảng user\_profile 51](#_Toc534161451)

[**Bảng 4.3:** Bảng mô tả thuộc tính bảng article 52](#_Toc534161452)

[**Bảng 4.4:** Bảng mô tả thuộc tính bảng article\_interact 53](#_Toc534161453)

[**Bảng 4.5:** Bảng mô tả thuộc tính bảng comment 54](#_Toc534161454)

[**Bảng 4.6:** Bảng mô tả thuộc tính bảng user\_interact 55](#_Toc534161455)

[**Bảng 4.7:** Bảng mô tả thuộc tính bảng tag 55](#_Toc534161456)

[**Bảng 4.8:** Bảng mô tả thuộc tính bảng tag\_article 56](#_Toc534161457)

[**Bảng 4.9:** Bảng mô tả thuộc tính bảng report\_article 56](#_Toc534161458)

[**Bảng 4.10:** Bảng mô tả thuộc tính bảng report\_comment 57](#_Toc534161459)

[**Bảng 4.11:** Bảng mô tả thuộc tính bảng session 57](#_Toc534161460)

[**Bảng 4.12:** Bảng mô tả thuộc tính bảng follow\_tag 58](#_Toc534161461)

[**Bảng 4.13:** Bảng mô tả thuộc tính bảng notification 59](#_Toc534161462)

[**Bảng 8.1:** Bảng testcase kiểm thử vai trò người dùng 84](#_Toc534161463)

[**Bảng 8.2:** Bảng testcase kiểm thử vai trò Admin 88](#_Toc534161464)

[**Bảng 8.3:** Bảng testcase kiểm thử vai trò Moderator 89](#_Toc534161465)

**DANH MỤC HÌNH ẢNH**

[**Hình 1.1:** Giao diện diễn đàn Viblo 5](#_Toc534161351)

[**Hình 1.2:** Giao diện diễn đàn Stackoverflow 7](#_Toc534161352)

[**Hình 1.3:** Giao diễn diễn đàn Github Community 9](#_Toc534161353)

[**Hình 1.4:** Giao diện diễn đàn Kipalog 11](#_Toc534161354)

[**Hình 2.1:** Mô hình giao tiếp giữa Client và Server sử dụng RESTful API 24](#_Toc534161355)

[**Hình 2.2:** Ví dụ về các Annotation trong Spring Boot RESTful 28](#_Toc534161356)

[**Hình 2.3:** RESTful API trong mô hình ba lớp Spring Boot 29](#_Toc534161357)

[**Hình 2.4:** Cấu trúc backend của một ứng dụng Spring Boot theo chuẩn RESTful API 30](#_Toc534161358)

[**Hình 3.1:** Sơ đồ Use Case Tổng quát diễn đàn 32](#_Toc534161359)

[**Hình 3.2:** Sơ đồ Use Case Tương tác người dùng 33](#_Toc534161360)

[**Hình 3.3:** Sơ đồ Use Case Quản lý bài viết 33](#_Toc534161361)

[**Hình 3.4:** Sơ đồ Use Case Quản lý Tag 33](#_Toc534161362)

[**Hình 3.5:** Sơ đồ Use Case Quản lý thông báo 34](#_Toc534161363)

[**Hình 3.6:** Sơ đồ Use Case Quản lý bình luận 34](#_Toc534161364)

[**Hình 3.7:** Sơ đồ Use Case Quản lý thông tin 35](#_Toc534161365)

[**Hình 3.8:** Sơ đồ Use Case Quản lý bài viết diễn đàn 35](#_Toc534161366)

[**Hình 3.9:** Sơ đồ Use Case Quản lý báo cáo diễn đàn 36](#_Toc534161367)

[**Hình 3.10:** Sơ đồ Use Case Quản lý tag diễn đàn 36](#_Toc534161368)

[**Hình 3.11:** Sơ đồ Use Case Quản lý thành viên diễn đàn 37](#_Toc534161369)

[**Hình 3.12:** Lược đồ hoạt động cho nghiệp vụ Đăng ký 37](#_Toc534161370)

[**Hình 3.13:** Lược đồ tuần tự cho nghiệp vụ Đăng ký 38](#_Toc534161371)

[**Hình 3.14:** Lược đồ hoạt động nghiệp vụ Đăng nhập 39](#_Toc534161372)

[**Hình 3.15:** Lược đồ tuần tự nghiệp vụ Đăng nhập 39](#_Toc534161373)

[**Hình 3.16:** Lược đồ hoạt động nghiệp vụ Tạo bài viết mới 40](#_Toc534161374)

[**Hình 3.17:** Lược đồ tuần tự nghiệp vụ Tạo bài viết mới 40](#_Toc534161375)

[**Hình 3.18:** Lược đồ hoạt động nghiệp vụ Đánh giá bài viết 41](#_Toc534161376)

[**Hình 3.19:** Lược đồ tuần tự nghiệp vụ Đánh giá bài viết 41](#_Toc534161377)

[**Hình 3.20:** Lược đồ hoạt động nghiệp vụ Lưu trữ bài viết 42](#_Toc534161378)

[**Hình 3.21:** Lược đồ tuần tự nghiệp vụ Lưu trữ bài viết 42](#_Toc534161379)

[**Hình 3.22:** Lược đồ hoạt động nghiệp vụ Bình luận 43](#_Toc534161380)

[**Hình 3.23:** Lược đồ tuần tự nghiệp vụ Bình luận 43](#_Toc534161381)

[**Hình 3.24:** Lược đồ hoạt động nghiệp vụ Báo cáo bài viết 44](#_Toc534161382)

[**Hình 3.25:** Lược đồ tuần tự nghiệp vụ Báo cáo bài viết 44](#_Toc534161383)

[**Hình 3.26:** Lược đồ hoạt động nghiệp vụ Theo dõi/ Bỏ theo dõi Tag 45](#_Toc534161384)

[**Hình 3.27:** Lược đồ tuần tự nghiệp vụ Theo dõi/ Bỏ theo dõi Tag 45](#_Toc534161385)

[**Hình 3.28:** Lược đồ hoạt động nghiệp vụ Duyệt bài viết 46](#_Toc534161386)

[**Hình 3.29:** Lược đồ tuần tự nghiệp vụ Duyệt bài viết 47](#_Toc534161387)

[**Hình 3.30:** Lược đồ hoạt động nghiệp vụ Gán vai trò người dùng 47](#_Toc534161388)

[**Hình 3.31:** Lược đồ tuần tự nghiệp vụ Gán vai trò người dùng 48](#_Toc534161389)

[**Hình 4.1:** Lược đồ ERD mô tả quan hệ giữa các bảng trong cơ sở dữ liệu 49](#_Toc534161390)

[**Hình 5.1:** Khởi tạo ứng dụng Spring Boot 61](#_Toc534161391)

[**Hình 5.2:** Chọn các dependency cần thiết cho một ứng dụng Spring Boot 62](#_Toc534161392)

[**Hình 5.3:** Cấu trúc thư mục ứng dụng Spring Boot 63](#_Toc534161393)

[**Hình 5.4:** Thêm các thư viện dependency cho ứng dụng trong pom.xml 64](#_Toc534161394)

[**Hình 5.5:** Cấu hình tập tin application.properties cho ứng dụng Spring Boot 65](#_Toc534161395)

[**Hình 5.6:** Cơ chế hoạt động của CORS 66](#_Toc534161396)

[**Hình 5.7:** Cấu hình AuthTokenFilter của ứng dụng 67](#_Toc534161397)

[**Hình 5.8:** Cài đặt Yarn cho quản lý thư viện Front-end 70](#_Toc534161398)

[**Hình 5.9:** Kiểm tra phiên bản của Yarn 70](#_Toc534161399)

[**Hình 5.10:** Khởi tạo dự án Angular 6 70](#_Toc534161400)

[**Hình 5.11:** Cấu trúc thư mục Front-end của dự án 71](#_Toc534161401)

[**Hình 5.12:** Các thư viện cho ứng dụng cài đặt trong package.json 72](#_Toc534161402)

[**Hình 5.13:** Các thư viện trong mục devDependencies 73](#_Toc534161403)

[**Hình 5.14:** Thêm các thư viện preload cho ứng dụng trong angular.json 73](#_Toc534161404)

[**Hình 6.1:** Tạo dự án trên Github 74](#_Toc534161405)

[**Hình 6.2:** Thêm các thành viên vào dự án 74](#_Toc534161406)

[**Hình 6.3:** Tạo branch develop cho nhóm phát triển 75](#_Toc534161407)

[**Hình 6.4:** Clone ứng dụng về máy làm việc cá nhân 75](#_Toc534161408)

[**Hình 6.5:** Thao tác với dựn án qua phần mềm ToitoseGit 76](#_Toc534161409)

[**Hình 6.6:** Push code lên branch develop sau mỗi thay đổi 77](#_Toc534161410)

[**Hình 7.1:** Mô hình hoạt động của ứng dụng và các Service 78](#_Toc534161411)

[**Hình 7.2:** Cấu trúc REST Controller của ứng dụng 79](#_Toc534161412)

[**Hình 7.3:** Mô tả hoạt động API xác thực người dùng 81](#_Toc534161413)

# GIỚI THIỆU ĐỀ TÀI

## 1. Tính cấp thiết của đề tài (Lý do chọn đề tài)

Trong thời đại ngày nay, lập trình đã trở nên phổ biến với mọi người từ mọi cấp học, từ mọi ngành bậc của xã hội. Không chỉ những kỹ sư công nghệ thông tin mới cần biết lập trình, dù học ở cấp bậc hay ngành nghề nào thì có kiến thức về lập trình cũng là một lợi điểm rất lớn nên nhu cầu tìm hiểu về lập trình chưa bao giờ là đủ. Việc học hỏi kiến thức lập trình giúp rất nhiều vào đời sống, xã hội như: bác sĩ có thể sử dụng các phần mềm giúp cho việc quản lý bệnh nhân thời giờ khám một cách nhanh chóng giảm thiểu tình trạng chờ, các kế toán có thể sử dụng phần mềm thống kê tính toán mà ở đó chỉ cần kế toán nhập liệu thông minh sẽ xuất ra được hóa đơn cho công ty,... Và việc học hỏi như vậy đối với những người vừa đi làm vừa chăm sóc gia đình là một điều khá khó khăn. Và chính vì lẽ đó, nhóm thực hiện báo cáo quyết định chọn đề tài xây dựng một cộng đồng, diễn đàn chia sẻ kiến thức lập trình tổng hợp, từ cơ bản đến nâng cao để phù hợp với mọi tầng lớp, ngành nghề. Mà ở đó mọi người có thể chia sẻ những kiến thức học được ở trường hoặc những kinh nghiệm đúc kết được từ thực tiễn. Từ đó, giúp mọi người tạo nên một cộng đồng cùng nhau chia sẻ và học tập hoàn toàn phi lơi nhuận.

Bên cạnh đó nhóm chọn nền tảng Web đặc biệt là Web Service với tính năng Single Page giúp cho diễn đàn mang hiệu năng tốt hơn, hỗ trợ cho các nền tảng di động với tính tương thích các kích cỡ màn hình giúp cho những người không có điều kiện trên máy tính cũng cõ thể dễ dàng theo dõi diễn đàn. Trong đề tài tiểu luận chuyên ngành nhóm thực hiện báo cáo sẽ sử dụng Framework Spring Boot kết hợp với RESTful Web Service để xây dựng ứng dụng trên nền tảng Web với tên đề tài: “Tìm hiểu và xây dựng ứng dụng Spring RESTful”.

## 2. Mục tiêu đề tài

Mục tiêu nhóm đặt ra trong tiểu luận chuyên ngành trước hết là tìm hiểu toàn diện về công nghệ sử dụng như Spring Boot, Angular và tập trung vào mô hình Web Service RESTful.

Tiếp theo là xây dựng được các chức năng cơ bản của một diễn đàn có phân chia theo quyền người dùng và quyền quản trị. Sử dụng phân quyền theo AccessToken cho tài khoản người dùng và viết các API cho mô hình RESTful. Từ đó nhóm sẽ kết hợp Spring Boot và Angular để gọi đến các API đó và tiến hành chạy API trên giao diện ứng dụng Single Page.

Mục tiêu cuối cùng nhóm đặt ra là có thể trong thời gian tìm hiểu nhóm không thể viết diễn đàn với nhiều chức năng nhưng sẽ cố gắng viết API cho việc phát triển chức năng sau này. Các API có thể hoạt động tốt và hiệu năng tốt khi được gọi tới.

## 3. Đối tượng và phạm vi nghiên cứu

Đối tượng nhóm nghiên cứu hướng đến là các bạn sinh viên học ngành công nghệ thông tin với kiến thức đã học được muốn chia sẻ tới mọi người. Cùng với đó là nhóm các anh chị, thầy cô đã đi làm và có kiến thức chuyên môn, kinh nghiệm muốn truyền đạt lại cho mọi người. Và đối tượng nghiên cứu cuối cùng nhóm hướng đến là nhóm các bạn, anh chị muốn học hỏi kiến thức lập trình mới nhưng không đủ thời gian tham gia các khóa học thì diễn đàn là nơi họ có thể học hỏi từ đây và nuôi đam mê lập trình của mình.

Phạm vi nghiên cứu của nhóm thực hiện chủ yếu nghiên cứu về cơ sở lý thuyết về Web Service và RESTful API cùng với đó là các vận dụng các cơ sở lý thuyết này vào xây dựng ứng dụng trong Spring Framework và ở đây là ứng dụng Spring Boot Restful API.

## 4. Phương pháp nghiên cứu

Từ phạm vi nghiên cứu mà nhóm rút ra được lộ trình cũng như phương pháp nghiên cứu cho đề tài như sau:

- Cơ sở lý thuyết tìm hiểu về Web Service

- Cơ sở lý thuyết tìm hiểu về RESTful API

- Cơ sở lý thuyết tìm hiểu về Sping Boot RESTful API

- Nghiên cứu từ lý thuyết đến thực tiễn vận dụng cơ sở lý thuyết để xây dựng diễn đàn chia sẻ kiến thức lập trình

- Nghiên cứu sử dụng Internet và sách về cơ sở lý thuyết

- Tổng hợp và viết báo cáo về quá trình nghiên cứu

## 5. Cấu trúc và nội dung đề tài

Cấu trúc của tiểu luận bao gồm phần giới thiệu đề tài sẽ đi sâu vào lý do chọn đề tài là lý do chọn đề tài, mục tiêu đề tài đặt ra, đối tượng, phạm vi và các phương pháp nghiên cứu.

Sau phần giới thiệu sẽ là phần nội dung là tám chương bao gồm cơ sở lý thuyết và phần vận dụng xây dựng ứng dụng diễn đàn. Phân tích yêu cầu, nghiệp vụ và xây dựng cơ sở dữ liệu, quản lý phiên bản, cài đặt và xây dựng backend và front-end cho ứng dụng. Phần cuối trong nội dung, nhóm báo cáo sẽ tiến hành kiểm thử một số chức năng của ứng dụng.

Cuối cùng là phần kết luận cùng với tài liệu tham khảo và phụ lục có liên quan trong tiểu luận và nhóm báo cáo đã liệt kê.

# KHẢO SÁT HIỆN TRẠNG

## 1.1. Hiện trạng

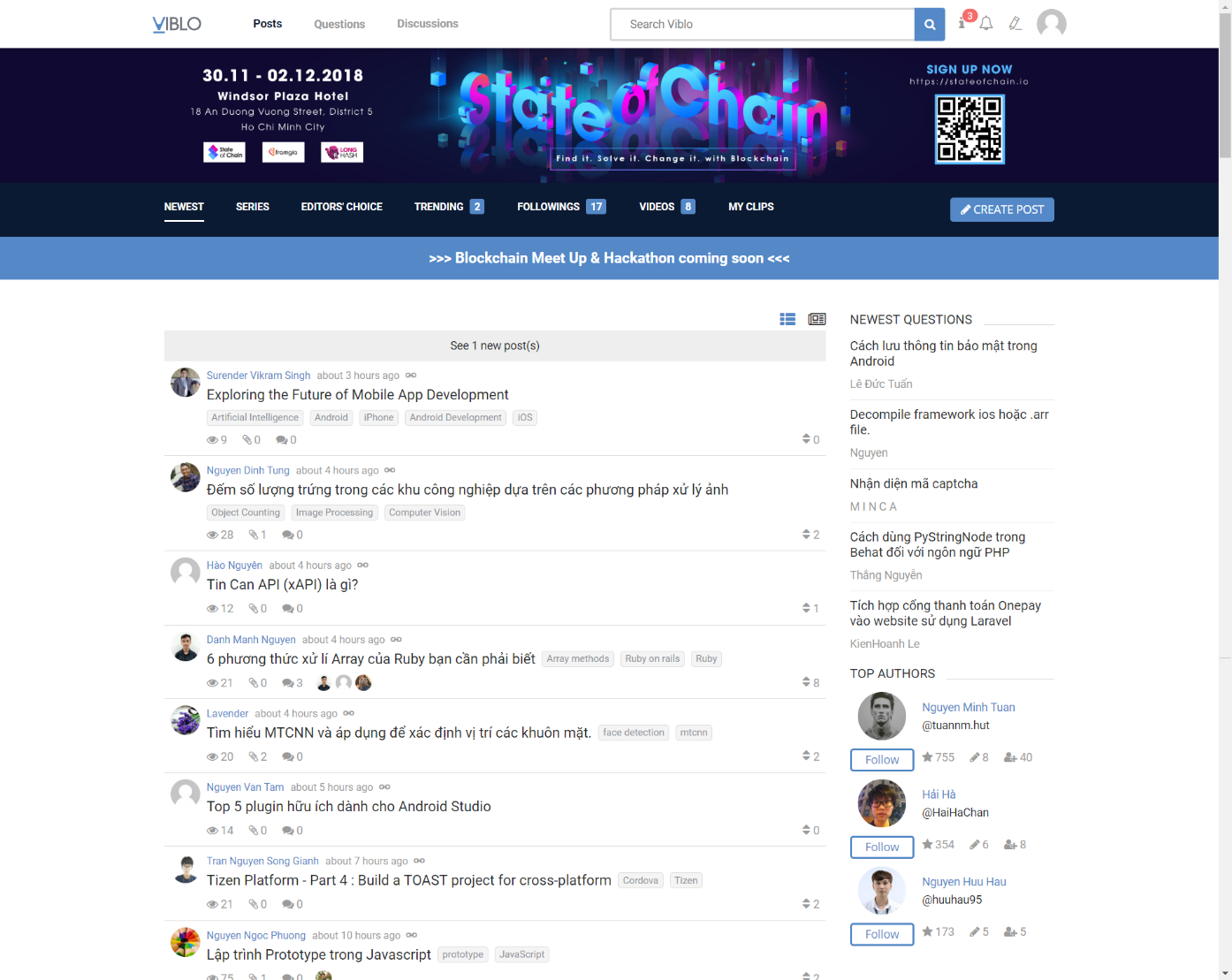
Hiện nay trong thời đại công nghệ 4.0 nở rộ nhu cầu về nguồn nhân lực công nghệ thông tin trong tất cả các lĩnh vực từ giáo dục, y tế, sản xuất, ... Nguồn nhân lực này là tiền đề tạo ra các sản phẩm công nghệ giúp số hóa, tự động hóa các quy trình làm việc. Và để đào tạo ra nguồn nhân lực dồi dào có kĩ năng không chỉ là công việc của các trường, các trung tâm mà còn là công việc của mọi người những người chia sẻ kiến thức ấy với nhau tạo ra một cộng đồng công nghệ nói chung và lập trình nói riêng.

Các kênh chia sẻ lập trình nổi tiếng hiện nay là: Facebook Group, Stack Overflow, Medium và các blog lập trình,... Kênh lập trình thì nhiều nhưng nội dung càng nhiều thì chất lượng càng loãng có thể gây ra sự khó khăn trong tìm kiếm nếu không tối ưu hóa diễn đàn một cách khoa học trong tìm kiếm, gắn thẻ nội dung hoặc các bài spam... Bên cạnh đó các bài viết trên blog cá nhân thường chia sẻ nhưng khó tiếp cận đến lượng người dùng đông đảo nếu không có cách quảng cáo phù hợp.

Số lượng người muốn chia sẻ kiến thức lập trình hoặc các kĩ năng trong quá trình học và làm tích lũy được ngày càng nhiều nhưng lại khó chia sẻ một cách chủ động, bên cạnh đó là số người mới đặt các câu hỏi càng nhiều và không đánh giá được chất lượng câu hỏi trong một diễn đàn chia sẻ.

## 1.2. Khảo sát

### 1.2.1. Diễn đàn Viblo



**Hình 1.1:** Giao diện diễn đàn Viblo

**- Chức năng**

+ Quản lý bài viết theo Tags

+ Cho phép đăng câu hỏi hoặc bài viết

+ Đăng các bài viết theo Series

+ Liệt kê bài viết: Newest, Series, Editor’s choice, Trending, Following, Videos, Clip

+ Có Table of Content trong bài viết

+ Gợi ý bài viết mới và Top Author trong danh sách bài viết

+ Có gợi ý bài viết liên quan và các bài viết hay của tác giả trong bài viết

+ Lưu bài viết với chức năng Clip

+ Nhật ký thảo luận của tất cả các bài viết

+ Vote bài viết, câu hỏi

+ Comment bài viết được đính kèm link và hình ảnh

+ Cho phép xem trước bài viết, comment trước khi post

+ Follow tác giả để nhận thông báo bài viết mới

+ Có hỗ trợ app di động

**- Giao diện**

+ Giao diện thân thiện với người dùng

+ Phần comment bị giới hạn một số chức năng

+ Chưa hỗ trợ đa ngôn ngữ

**- Ưu điểm**

+ Giao diện thân thiện, dễ sử dụng

+ Hỗ trợ đầy đủ các công cụ để tạo bài viết

+ Công cụ tìm kiếm mạnh mẽ

+ Cập nhật các bài viết Realtime

+ Hỗ trợ Table of content giúp người đọc dễ theo dõi nội dung

+ Quản lý bài viết theo tag giúp người dùng dễ tìm kiếm bài viết

+ Hỗ trợ lưu bài viết, câu hỏi, series

+ Thống kê rõ ràng thông tin người dùng

**- Nhược điểm**

+ Chưa hỗ trợ đa ngôn ngữ

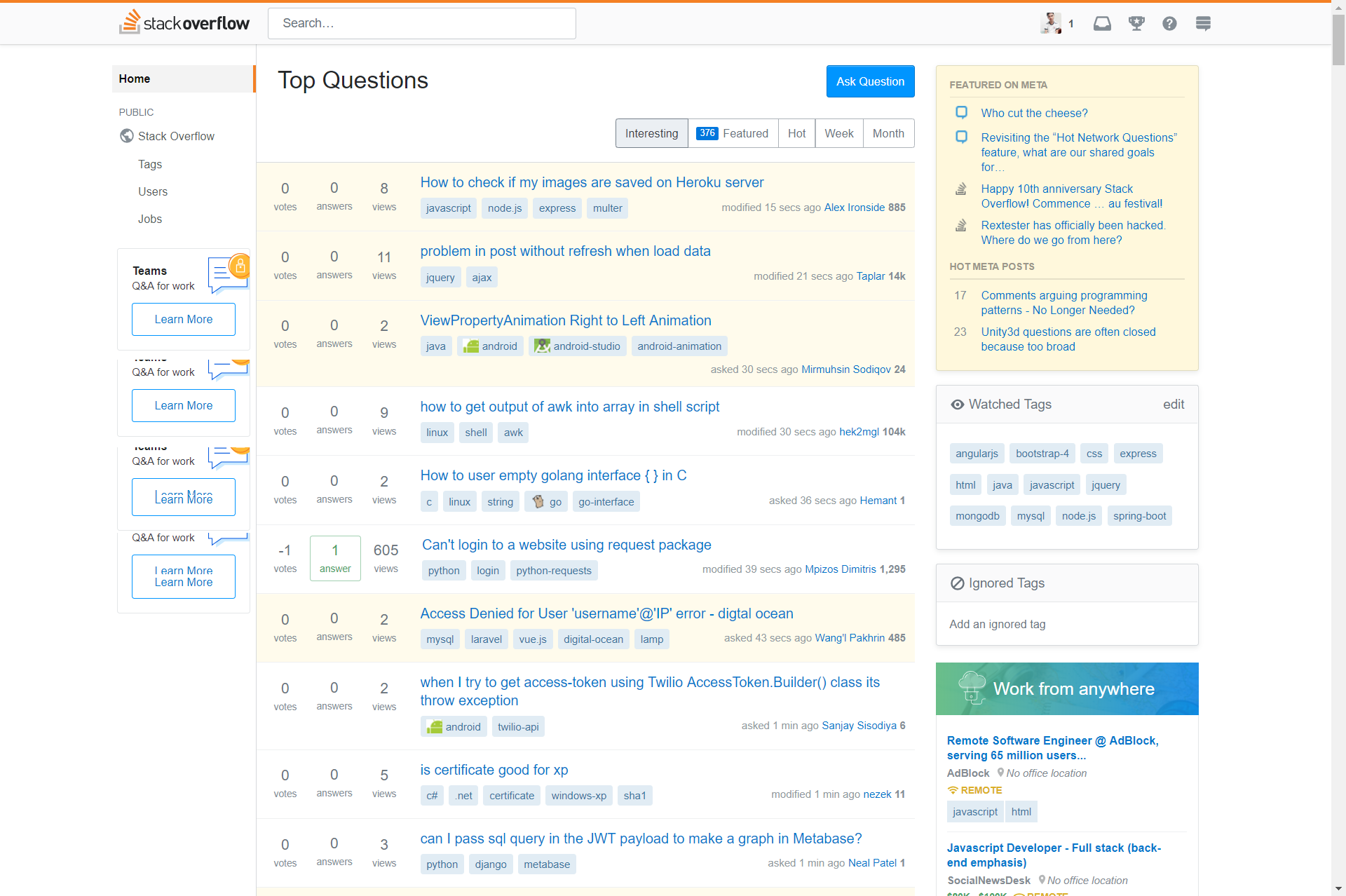
+ Công cụ soạn comment chưa đầy đủ

+ Quản lý thông báo chưa tốt

+ Không hỗ trợ chọn tag yêu thích

+ Không hỗ trợ huy hiệu

### 1.2.2. Diễn đàn Stackoverflow



**Hình 1.2:** Giao diện diễn đàn Stackoverflow

**- Chức năng**

+ Quản lý bài viết theo Tags

+ Cho phép đăng câu hỏi

+ Liệt kê bài viết: Interesting, Featured, Hot, Week, Month

+ Chọn tag và hiển thị bài viết theo tag yêu thích

+ Gợi ý top bài viết trong danh sách bài viết

+ Có gợi ý bài viết liên quan và các bài viết hay của tác giả trong bài viết

+ Vote câu hỏi, trả lời

+ Comment bài viết được đính kèm link, code, hình ảnh

+ Hỗ trợ các công ty đăng việc làm

+ Tìm kiếm việc làm

**- Giao diện**

+ Giao diện rất khó sử dụng

+ Chưa có preview cho soạn thảo bài viết và comment

+ Chưa hỗ trợ đa ngôn ngữ

+ Quá nhiều chữ

+ Bố cục chưa hợp lý

**- Ưu điểm**

+ Hỗ trợ tương đối đầy đủ các công cụ để tạo bài viết

+ Quản lý bài viết theo tag giúp người dùng dễ tìm kiếm bài viết

+ Cho phép chọn những tag yêu thích và duyệt bài viết liên quan

+ Thống kê rõ ràng thông tin, hoạt động của người dùng

+ Hỗ trợ quyền lợi người dùng tốt

+ Hỗ trợ công ty đăng việc làm

+ Xử lý tốt phần chi tiết bài đăng

**- Nhược điểm**

+ Chưa hỗ trợ đa ngôn ngữ

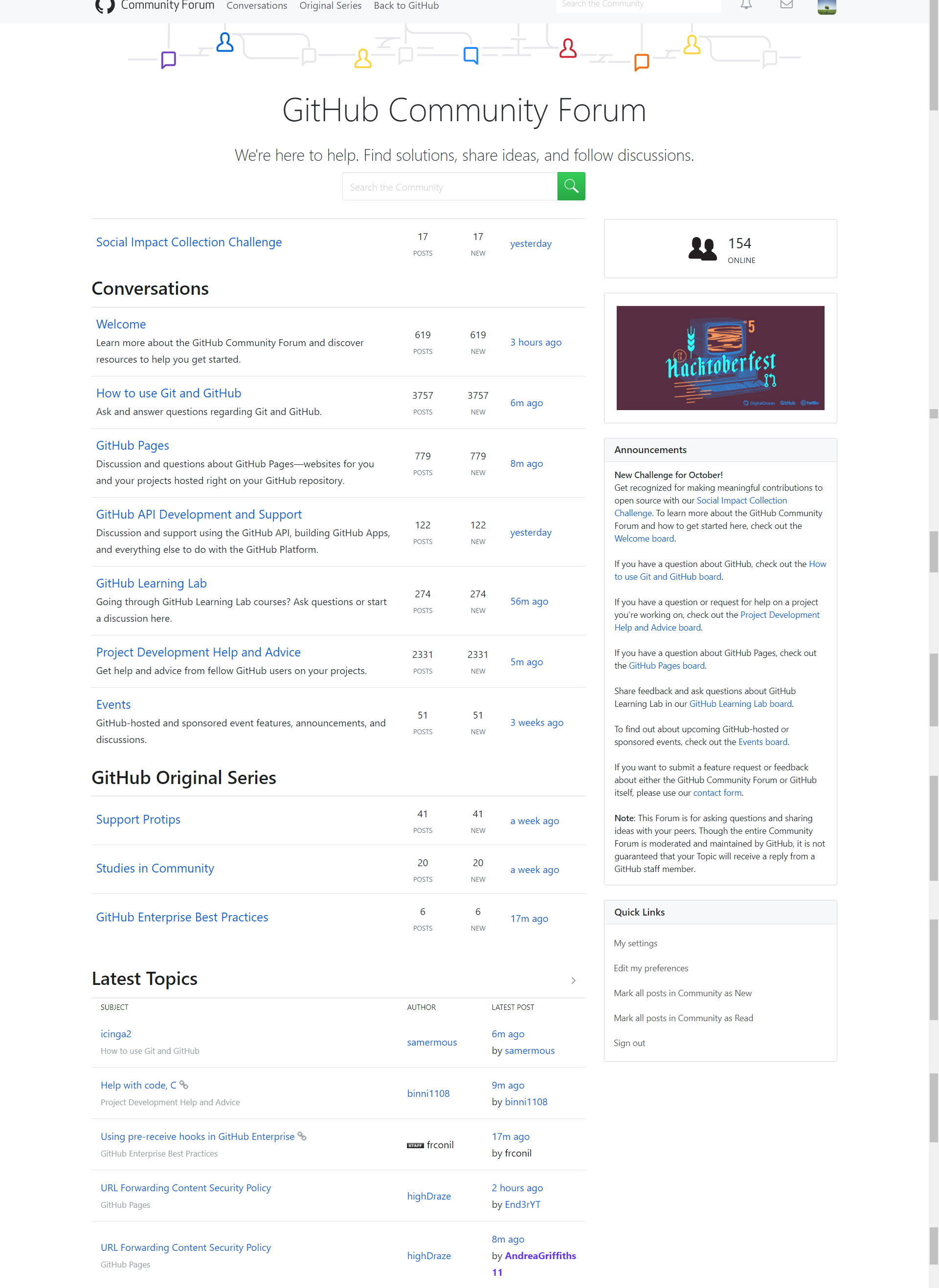
+ Công cụ soạn thảo chưa có preview

+ Giao diện chưa hợp lý, xấu và khó sử dụng

+ Không hỗ trợ đăng bài viết

+ Công cụ tìm kiếm chưa tốt

### 1.2.3. Diễn đàn Github Community



**Hình 1.3:** Giao diễn diễn đàn Github Community

**- Chức năng**

+ Cho phép đăng câu hỏi và bài viết

+ Liệt kê bài viết: Conversation, Series

+ Gợi ý top bài viết trong danh sách bài viết

+ Có gợi ý bài viết liên quan và các bài viết hay của tác giả trong bài viết

+ Vote trả lời

+ Comment bài viết được đính kèm link, code, hình ảnh

+ Hỗ trợ preview khi soạn thảo bài viết và comment

**- Giao diện**

+ Giao diện khó sử dụng

+ Chưa hỗ trợ đa ngôn ngữ

+ Bố cục chưa hợp lý

**- Ưu điểm**

+ Hỗ trợ đầy đủ các công cụ để tạo bài viết và comment

+ Cho phép duyệt bài viết liên quan

+ Thống kê rõ ràng thông tin, hoạt động của người dùng

**- Nhược điểm**

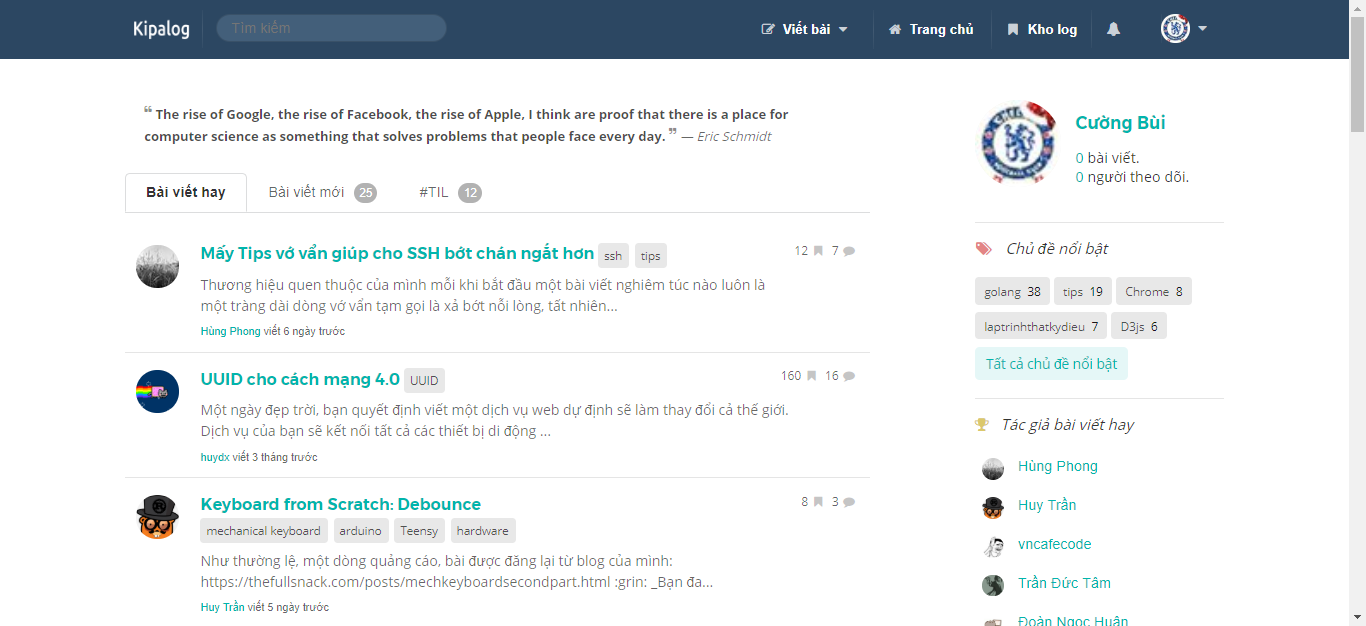
+ Chưa hỗ trợ đa ngôn ngữ

+ Giao diện chưa tối ưu cho người sử dụng

+ Công cụ tìm kiếm chưa tốt

+ Chỉ hỗ trợ like bài viết

### 1.2.4. Diễn đàn Kipalog



**Hình 1.4:** Giao diện diễn đàn Kipalog

**- Chức năng**

+ Có chức năng bài viết hay, bài viết mới

+ Quản lý thẻ theo chủ đề nổi bật

+ Có danh sách tác giả bài viết hay

+ Kho log đánh dấu lại các bài viết của người dùng

+ Có tính năng người theo dõi thẻ và theo dõi tác giả

+ Có liên kết đăng nhập bằng Facebook và Github

**- Giao diện**

+ Giao diện nhìn đơn giản dễ sử dụng cho người mới

+ Các của sổ được chia rõ ràng bố cục hợp lý

+ Màu sắc hài hòa không quá lòe loẹt

**- Ưu điểm**

+ Liên kết mạng xã hội giúp tiết kiệm thời gian đăng ký

+ Có quản lý theo thẻ dễ dàng tìm kiếm bài viết hơn

+ Đánh dấu lại bài viết giúp sau này có thể tìm kiếm lại dễ dàng

**- Nhược điểm**

+ Vì giao diện đơn giản nên dễ gây nhàm chán người sử dụng

+ Chưa có tính năng phân ra thành bài viết và câu hỏi nên khó quản lý đâu là dạng câu hỏi

# CƠ SỞ LÝ THUYẾT

## 2.1. Tìm hiểu về Web Service

### 2.1.1. Khái niệm về Web Service

Dịch vụ web ( hay còn gọi là web service) là một khái niệm rộng hơn so với khái niệm web thông thường. Nó là sự kết hợp các máy tính cá nhân với các thiết bị khác, các cơ sở dữ liệu và các mạng máy tính để tạo thành một cơ cấu tính toán ảo mà người sử dụng có thể làm việc thông qua các trình duyệt mạng. Các Web Service thường cung cấp các dữ liệu thô mà nó khó hiểu đối với đa số người dùng thông thường, chúng thường được trả về dưới dạng XML hoặc JSON. Hiện nay, web service trở thành dịch vụ mạnh mẽ, cung cấp lợi ích cho các doanh nghiệp, khách hàng, cá nhân, trong nhiều lĩnh vực thực tế: thông tin thương mại, dịch vụ du lịch, tỉ giá, chứng khoán...

Một số chuẩn truyền dữ liệu của Web Service : SOAP, XML-RPC, hay mới nhất là REST.

### 2.1.2. Đặc điểm của Web Service

- Cho phép client và server tương tác ngay cả trong môi trường khác nhau. (Ví dụ server chạy linux, client chạy windows).

- Phần lớn được xây dựng dựa trên mã nguồn mở và phát triển các chuẩn đã được công nhận. (Ví dụ XML).

- Nó có thể triển khai bởi 1 phần mềm ứng dụng phía server (Ví dụ : PHP, Oracle Application server, Microsoft .NET)

### 2.1.3. Ưu điểm của Web Service

- Web service cung cấp nền tảng rộng lớn chạy được trên các hệ điều hành khác nhau

- Nâng cao khả năng tái sử dụng nguồn tài nguyên của hệ thống

- Tạo mối quan hệ tương tác lẫn nhau, dễ dàng cho việc phát triển các ứng dụng phân tán.

- Thúc đẩy mạnh mẽ vào hệ thống tích hợp và giảm được sự phức tạp của hệ thống, giảm giá thành phần tương tác tốt với hệ thống doanh nghiệp.

- Sử dụng các giao thức và chuẩn mở, giao thức và định dạng dữ liệu dựa trên văn bản giúp các lập trình viên dễ dàng hiểu được

### 2.1.4. Nhược điểm của Web Service

- Có nhiều chuẩn giao tiếp khiến người dùng khó nắm bắt

- Có thể xảy ra thiệt hại lớn vào khoảng thời gian không hoạt động của web service như: có thể lỗi nếu như máy tính không nâng cấp, thiếu các giao tiếp trong việc vận hành.

- Phải quan tâm nhiều hơn đến vẫn đề bảo mật

### 2.1.5. Kiến trúc của Web Service

Dịch vụ Web gồm có 3 chuẩn chính: SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) và UDDI (Universal Description, Discovery, and Integration). Hình 1 mô tả chồng giao thức của dịch vụ Web, trong đó UDDI được sử dụng để đăng ký và khám phá dịch vụ Web đã được miêu tả cụ thể trong WSDL. Giao tác UDDI sử dụng SOAP để nói chuyện với UDDI server, sau đó các ứng dụng SOAP yêu cầu một dịch vụ Web. Các thông điệp SOAP được gửi đi chính xác bởi HTTP và TCP/IP.

Chồng giao thức dịch vụ Web là tập hợp các giao thức mạng máy tính được sử dụng để định nghĩa, xác định vị trí, thi hành và tạo nên dịch vụ Web tương tác với những ứng dụng hay dịch vụ khác. Chồng giao thức này có 4 thành phần chính:

Dịch vụ vận chuyển (Service Transport): có nhiệm vụ truyền thông điệp giữa các ứng dụng mạng, bao gồm những giao thức như HTTP, SMTP, FTP, JSM và gần đây nhất là giao thức thay đổi khổi mở rộng (Blocks Extensible Exchange Protocol- BEEP).

Thông điệp XML: có nhiệm vụ giải mã các thông điệp theo định dạng XML để có thể hiểu được ở mức ứng dụng tương tác với người dùng. Hiện tại, những giao thức thực hiện nhiệm vụ này là XML-RPC, SOAP và REST.

Mô tả dịch vụ: được sử dụng để miêu tả các giao diện chung cho một dịch vụ Web cụ thể. WSDL thường được sử dụng cho mục đích này, nó là một ngôn ngữ mô tả giao tiếp và thực thi dựa trên XML. Dịch vụ Web sẽ sử dụng ngôn ngữ này để truyền tham số và các loại dữ liệu cho các thao tác và chức năng mà dịch vụ Web cung cấp.

Khám phá dịch vụ: tập trung dịch vụ vào trong một nơi được đăng ký, từ đó giúp một dịch vụ Web có thể dễ dàng khám phá ra những dịch vụ nào đã có trên mạng, tốt hơn trong việc tìm kiếm những dịch vụ khác để tương tác. Một dịch vụ Web cũng phải tiến hành đăng ký để các dịch vụ khác có thể truy cập và giao tiếp. Hiện tại, UDDI API thường được sử dụng để thực hiện công việc này.

Trong đó, tầng giao thức tương tác dịch vụ (Service Communication Protocol) với công nghệ chuẩn là SOAP. SOAP là giao thức nằm giữa tầng vận chuyển và tầng mô tả thông tin về dịch vụ, cho phép người dùng triệu gọi một dịch vụ từ xa thông qua một thông điệp XML. Ngoài ra, để các dịch vụ có tính an toàn, toàn vẹn và bảo mật thông tin, trong kiến trúc dịch vụ Web, chúng ta có thêm các tầng Policy, Security, Transaction, Management.

### 2.1.6. Các thành phần của Web Service

**2.1.6.1. XML – eXtensible Markup Language**

Là một chuẩn mở do W3C đưa ra cho cách thức mô tả dữ liệu, nó được sử dụng để định nghĩa các thành phần dữ liệu trên trang web và cho những tài liệu B2B. Về hình thức, XML hoàn toàn có cấu trúc thẻ giống như ngôn ngữ HTML nhưng HTML định nghĩa thành phần được hiển thị như thế nào thì XML lại định nghĩa những thành phần đó chứa cái gì. Với XML, các thẻ có thể được lập trình viên tự tạo ra trên mỗi trang web và được chọn là định dạng thông điệp chuẩn bởi tính phổ biến và hiệu quả mã nguồn mở.

Do dịch vụ Web là sự kết hợp của nhiều thành phần khác nhau nên nó sử dụng các tính năng và đặc trưng của các thành phần đó để giao tiếp. XML là công cụ chính để giải quyết vấn đề này và là kiến trúc nền tảng cho việc xây dựng một dịch vụ Web, tất cả dữ liệu sẽ được chuyển sang định dạng thẻ XML. Khi đó, các thông tin mã hóa sẽ hoàn toàn phù hợp với các thông tin theo chuẩn của SOAP hoặc XML-RPC và có thể tương tác với nhau trong một thể thống nhất.

**2.1.6.2. WSDL – Web Service Description Language**

WSDL định nghĩa cách mô tả dịch vụ Web theo cú pháp tổng quát của XML, bao gồm các thông tin:

- Tên dịch vụ

- Giao thức và kiểu mã hóa sẽ được sử dụng khi gọi các hàm của dịch vụ Web

- Loại thông tin: thao tác, tham số, những kiểu dữ liệu (có thể là giao diện của dịch vụ Web cộng với tên cho giao diện này).

Một WSDL hợp lệ gồm hai phần: phần giao diện (mô tả giao diện và phương thức kết nối) và phần thi hành mô tả thông tin truy xuất CSDL. Cả hai phần này sẽ được lưu trong 2 tập tin XML tương ứng là tập tin giao diện dịch vụ và tập tin thi hành dịch vụ. Giao diện của một dịch vụ Web được miêu tả trong phần này đưa ra cách thức làm thế nào để giao tiếp qua dịch vụ Web. Tên, giao thức liên kết và định dạng thông điệp yêu cầu để tương tác với dịch vụ Web được đưa vào thư mục của WSDL.

WSDL thường được sử dụng kết hợp với XML schema và SOAP để cung cấp dịch vụ Web qua Internet. Một client khi kết nối tới dịch vụ Web có thể đọc WSDL để xác định những chức năng sẵn có trên server. Sau đó, client có thể sử dụng SOAP để lấy ra chức năng chính xác có trong WSDL.

**2.1.6.3. Universal Description, Discovery, and Integration (UDDI)**

Để có thể sử dụng các dịch vụ, trước tiên client phải tìm dịch vụ, ghi nhận thông tin về cách sử dụng và biết được đối tượng nào cung cấp dịch vụ. UDDI định nghĩa một số thành phần cho biết các thông tin này, cho phép các client truy tìm và nhận những thông tin được yêu cầu khi sử dụng dịch vụ Web.

– Cấu trúc UDDI :

+ Trang trắng – White pages: chứa thông tin liên hệ và các định dạng chính yếu của dịch vụ Web, chẳng hạn tên giao dịch, địa chỉ, thông tin nhận dạng… Những thông tin này cho phép các đối tượng khác xác định được dịch vụ.

+ Trang vàng – Yellow pages: chứa thông tin mô tả dịch vụ Web theo những loại khác nhau. Những thông tin này cho phép các đối tượng thấy được dịch vụ Web theo từng loại với nó.

+ Trang xanh – Green pages: chứa thông tin kỹ thuật mô tả các hành vi và các chức năng của dịch vụ Web.

+ Loại dịch vụ – tModel: chứa các thông tin về loại dịch vụ được sử dụng.

Những thông tin về dịch vụ Web được sử dụng và công bố lên mạng sử dụng giao thức này. Nó sẽ kích hoạt các ứng dụng để tìm kiếm thông tin của dịch vụ Web khác nhằm xác định xem dịch vụ nào sẽ cần đến nó.

**2.1.6.4. SOAP – Simple Object Access Protocol**

Chúng ta đã hiểu cơ bản dịch vụ Web như thế nào nhưng vẫn còn một vấn đề khá quan trọng. Đó là làm thế nào để truy xuất dịch vụ khi đã tìm thấy? Câu trả lời là các dịch vụ Web có thể truy xuất bằng một giao thức là Simple Object Access Protocol – SOAP. Nói cách khác chúng ta có thể truy xuất đến UDDI registry bằng các lệnh gọi hoàn toàn theo định dạng của SOAP.

SOAP là một giao thức giao tiếp có cấu trúc như XML. Nó được xem là cấu trúc xương sống của các ứng dụng phân tán được xây dựng từ nhiều ngôn ngữ và các hệ điều hành khác nhau. SOAP là giao thức thay đổi các thông điệp dựa trên XML qua mạng máy tính, thông thường sử dụng giao thức HTTP.

Một client sẽ gửi thông điệp yêu cầu tới server và ngay lập tức server sẽ gửi những thông điệp trả lời tới client. Cả SMTP và HTTP đều là những giao thức ở lớp ứng dụng của SOAP nhưng HTTP được sử dụng và chấp nhận rộng rãi hơn bởi ngày nay nó có thể làm việc rất tốt với cơ sở hạ tầng Internet.

Cấu trúc một thông điệp theo dạng SOAP: Thông điệp theo định dạng SOAP là một văn bản XML bình thường bao gồm các phần tử sau:

– Phần tử gốc – envelop: phần tử bao trùm nội dung thông điệp, khai báo văn bản XML như là một thông điệp SOAP.

– Phần tử đầu trang – header: chứa các thông tin tiêu đề cho trang, phần tử này không bắt buộc khai báo trong văn bản. Header còn có thể mang những dữ liệu chứng thực, những chứ ký số, thông tin mã hóa hay cài đặt cho các giao dịch khác.

– Phần tử khai báo nội dung chính trong thông điệp – body, chứa các thông tin yêu cầu và thông tin được phản hồi.

– Phần tử đưa ra các thông tin về lỗi -fault, cung cấp thông tin lỗi xảy ra trong qúa trình xử lý thông điệp.

Một SOAP đơn giản trong body sẽ lưu các thông tin về tên thông điệp, tham chiếu tới một thể hiện của dịch vụ, một hoặc nhiều tham số. Có 3 kiểu thông báo sẽ được đưa ra khi truyền thông tin: request message(tham số gọi thực thi một thông điệp), respond message (các tham số trả về, được sử dụng khi yêu cầu được đáp ứng) và cuối cùng là fault message (thông báo tình trạng lỗi).

Kiểu truyền thông: Có 2 kiểu truyền thông

– Remote procedure call (RPC): cho phép gọi hàm hoặc thủ tục qua mạng. Kiểu này được khai thác bởi nhiều dịch vụ Web.

– Document: được biết đến như kiểu hướng thông điệp, nó cung cấp giao tiếp ở mức trừu tượng thấp, khó hiểu và yêu cầu lập trình viên mất công sức hơn.

Hai kiểu truyền thông này cung cấp các định dạng thông điệp, tham số, lời gọi đến các API khác nhau nên việc sử dụng chúng tùy thuộc vào thời gian và sự phù hợp với dịch vụ Web cần xây dựng.

Cấu trúc dữ liệu: Cung cấp những định dạng và khái niệm cơ bản giống như trong các ngôn ngữ lập trình khác như kiểu dữ liệu (int, string, date…) hay những kiều phức tạp hơn như struct, array, vector… Định nghĩa cấu trúc dữ liệu SOAP được đặt trong namespace SOAP-ENC.

Mã hóa: Giả sử service rquester và service provider được phát triển trong Java, khi đó mã hóa SOAP là làm thế nào chuyển đổi từ cấu trúc dữ liệu Java sang SOAP XML và ngược lại, bởi vì định dạng cho Web Service chính là XML. Bất kỳ một môi trường thực thi SOAP nào cũng phải có một bảng chứa thông tin ánh xạ nhằm chuyển đổi từ ngôn ngữ Java sang XML và từ XML sang Java – bảng đó được gọi là SOAPMappingRegistry. Nếu một kiểu dữ liệu được sử dụng dưới một dạng mã hóa thì sẽ có một ánh xạ tồn tại trong bộ đăng ký của môi trường thực thi SOAP đó.

## 2.2. Tìm hiểu về RESTful Web Service và RESTful API

### 2.2.1. Tổng quan về RESTful Web Service

REST đã được sử dụng rộng rãi thay thế cho các Web Service dựa trên SOAP và WSDL.

Những khái niệm đầu tiên về REST ( REpresentational State Transfer) được đưa ra vào năm 2000 trong luận văn tiến sĩ của Roy Thomas Fielding (đồng sáng lập giao thức HTTP). Trong luận văn ông giới thiệu khá chi tiết về các ràng buộc, quy ước cũng như cách thức thực hiện với hệ thống để có được một hệ thống REST.

REST định nghĩa các quy tắc kiến trúc để thiết kế Web Service, chú trọng vào tài nguyên hệ thống, bao gồm các trạng thái tài nguyên được định dạng như thế nào và được truyền tải qua HTTP, và được viết bởi nhiều ngôn ngữ khác nhau. Nếu tính theo số dịch vụ mạng sử dụng, REST đã nổi lên trong vài năm qua như là một mô hình thiết kế dịch vụ chiếm ưu thế. Trong thực tế, REST đã có những ảnh hưởng lớn và gần như thay thế SOAP và WSDL vì nó đơn giản và dễ sử dụng hơn rất nhiều.

REST là một bộ quy tắc để tạo ra một ứng dụng Web Service, mà nó tuân thủ 4 nguyên tắc thiết kế cơ bản sau:

- Sử dụng các phương thức HTTP một cách rõ ràng

- Phi trạng thái

- Hiển thị cấu trúc thư mục như các URls

- Truyền tải JavaScript Object Notation (JSON), XML hoặc cả hai.

### 2.2.2. Tổng quan về API

API là viết tắt của Application Programming Interface, một giao diện mà một hệ thống máy tính hay ứng dụng cung cấp để cho phép các yêu cầu dịch vụ có thể được tạo ra từ các chương trình máy tính khác, và/hoặc cho phép dữ liệu có thể được trao đổi qua lại giữa chúng. Chúng ta có thể viết các API chuẩn và cho phép tất cả các platform gửi request tới API đó mà không phải thay đổi code quá nhiều.

Cách API làm việc: Phía người dùng gửi request, API sẽ gửi lại response là liệu có thể làm được cái người dùng muốn hay ko. Và API được xây dựng trên 2 thành phần chính: Request và Response.

Trong nhiều năm qua, API được mô tả là giao diện kết nối chung với ứng dụng. Gần đây API hiện đại bổ sung một số tính năng mới, làm cho nó trở lên hữu ích:

- API hiện đại tuân thủ các chuẩn (thường là HTTP và REST), dễ sửu dụng, dễ truy cập và dễ hiểu. Tham khảo bài viết HTML5 là gì tại đây

- Các API này được xử lý giống như các sản phẩm nhiều hơn là code. Chúng được thiết kế cho các đối tượng người dùng cụ thể (chẳng hạn như các nhà phát triển thiết bị di động), và có các phiên bản cho người dùng và duy trì vòng đời của nó.

- Vì các API được chuẩn hóa nhiều hơn nên vấn đề bảo mật và quản lý cũng nghiêm ngặt hơn, cũng như các vấn đề theo dõi và quản lý hiệu suất, quy mô.

- Giống như các sản phẩm phần mềm khác, API hiện đại cũng có vòng đời phát triển phần mềm (SDLC) riêng từ thiết kế, thử nghiệm, xây dựng, quản lý và các phiên bản. Ngoài ra các API hiện đại cũng được ghi lại cho người dùng và các phiên bản.

### 2.2.3. Định nghĩa RESTful API

RESTful API là các API được thiết kế theo quy tắc chuẩn hoàn toàn dựa trên kiến trúc truyền dữ liệu REST cho các ứng dụng web để quản lý các tài nguyên.

Trọng tâm của REST quy định cách sử dụng các HTTP method (như GET, POST, PUT, DELETE) và cách định dạng các URL cho ứng dụng web để quản các tài nguyên của ứng dụng đó.

### 2.2.4. Nhu cầu và mục đích sử dụng

Hiện nay môi trường Internet và các ứng dụng hoạt động dựa trên Internet và Web Service rất nhiều, số lượng thông tin cần truyền lên và nhận về rất lớn vì vậy cần sử dụng RESTful API để có thể sử dụng tối đa chức năng của các method của HTTP trong việc truyền tải dữ liệu mà không tốn quá nhiều dung lượng bởi vì RESTful trả về kiểu JSON hoặc XML nên trong quá trình người dùng cần lấy dữ liệu không tốn quá nhiều tài nguyên hệ thống và khi truy cập vào đúng API cần lấy thì trả về tài nguyên đúng với những yêu cầu của người dùng.

RESTful API có thể được áp dụng trong bất kì Framework nào cũng như với nhiều ngôn ngữ lập trình như: Java, NodeJS, Spring Boot, C#, Android, ...

Việc API hóa các hàm trong một ứng dụng Web giúp cho việc quản lý tài nguyên một cách khoa học, hiệu quả đồng thời có thể mở rộng cho các bên thứ 3 sử dụng API.

Mỗi phương thức quy định một chức năng cụ thể của công việc như lấy thông tin dùng GET, thêm mới dùng POST, cập nhật dùng PUT, xóa dùng DELETE giúp cho việc giao tiếp giữa client và server trong việc giao tiếp thông tin trở nên gọn gàng hơn.

- Sử dụng khi ứng dụng không đặt nặng vào vấn đề bảo mật, hoặc khi máy chủ không đủ mạnh để có thể kiểm soát tài nguyên thì REST là một lựa chọn phù hợp.

### 2.2.5. Quy tắc chuẩn trong thiết kế RESTful API

**2.2.5.1. Sử dụng các phương thức HTTP một cách rõ ràng**

REST đặt ra một quy tắc đòi hỏi lập trình viên xác định rõ ý định của mình thông qua các phương thức của HTTP. Thông thường ý định đó bao gồm lấy dữ liệu, trèn dữ liệu, cập nhập dữ liệu hoặc xóa dữ liệu. Khi muốn thực hiện một trong các ý định trên phải tuân thủ các quy tắc sau:

- Để tạo một tài nguyên trên máy chủ, cần sử dụng phương thức POST.

- Để truy xuất một tài nguyên, sử dụng GET.

- Để thay đổi trạng thái một tài nguyên hoặc để cập nhật nó, sử dụng PUT.

- Để huỷ bỏ hoặc xoá một tài nguyên, sử dụng DELETE.

Các nguyên tắc ở trên là không bắt buộc, thực tế khi thiết kế API có thể sử dụng phương thức GET để yêu cầu lấy dữ liệu, trèn, sửa hoặc xóa dữ liệu trên Server. Tuy nhiên REST đưa ra các nguyên tắc ở trên mục đích đưa mọi thứ trở lên rõ ràng và dễ hiểu.

**2.2.5.2. Phi trạng thái (Stateless)**

Một đặc điểm của REST là phi trạng thái (stateless), có nghĩa là nó không lưu giữ thông tin của client. Chẳng hạn vừa gửi yêu cầu để xem trang thứ 2 của một tài liệu, và bây giờ muốn xem trang tiếp theo (sẽ là trang 3). REST không lưu trữ lại thông tin rằng trước đó nó đã phục vụ trang số 2. Điều đó có nghĩa là REST không quản lý phiên làm việc (Session).

Như vậy, các thành phần máy chủ phi trạng thái ít phức tạp hơn để thiết kế, viết và phân bổ thông qua máy chủ được cân bằng tải. Dịch vụ phi trạng thái không chỉ hoạt động tốt hơn, nó còn chuyển hầu hết vai trò duy trì trạng thái sang ứng dụng ở máy khách. Trong một dịch vụ mạng RESTful, máy chủ chịu trách nhiệm đưa ra các phản hồi và cung cấp một cách thức cho phép Client duy trì trạng thái ứng dụng của chính nó.

**2.2.5.3. Đưa ra cấu trúc thư mục giống các URI**

REST đưa ra một cấu trúc để người dùng có thể truy cập vào tài nguyên của nó thông qua các URL, tài nguyên ở đây là tất cả những cái mà có thể gọi tên được (Video, ảnh, báo cáo thời tiết,..)

Tạo ra các REST Serivce để trả về cho người dùng các nguồn tài nguyên tương ứng.

Các địa chỉ REST Service cần phải thật trực quan đến mức người dùng dễ đoán. Hãy nghĩ về một địa chỉ (URI) giống như một gợi ý rõ ràng, dễ đoán rằng nó đang trỏ tới cái gì và cung cấp tài nguyên gì. Tóm lại, cấu trúc của một URI nên được đơn giản, có thể dự đoán, và dễ hiểu.

Một vài nguyên tắc bổ sung để lưu ý trong khi nói về cấu trúc địa chỉ của RESTful Web service là:

- Giấu các đuôi tài liệu mở rộng của bản gốc trong máy chủ (.jsp, .php, .asp), nếu có, vì vậy có thể giấu một số thứ mà không cần thay đổi địa chỉ Urls.

- Để mọi thứ là chữ thường.

- Thay thế các khoảng trống bằng gạch chân hoặc hoặc gạch nối (một trong hai loại).

- Tránh các chuỗi yêu cầu càng nhiều càng tốt.

- Thay vì sử dụng mã (404 Not Found) khi yêu cầu địa chỉ cho một phần đường dẫn, luôn luôn cung cấp một trang mặc định hoặc tài nguyên như một phản hồi.

**2.2.5.4. Truyền tải XML, JSON hoặc cả hai**

Khi Client gửi một yêu cầu tới web service nó thường được truyền tải dưới dạng XML hoặc JSON và thông thường nhận về với hình thức tương tự.

Đôi khi Client cũng có thể chỉ định kiểu dữ liệu nhận về mà nó mong muốn (JSON, hoặc XML,..), các chỉ định này được gọi là các kiểu MINE, nó được gửi kèm trên phần HEADER của request.

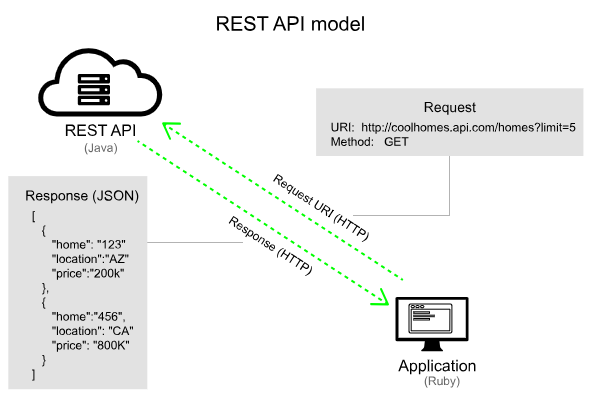
Dưới đây là các kiểu MINE phổ biến thường sử dụng với REST Service:

**Bảng 2.1:** Danh mục MINE phổ biến sử dụng với REST Service

|  |  |
| --- | --- |
| **MIME-Type** | **Content-Type** |
| JSON | application/json |
| XML | application/xml |
| XHTML | application/xhtml+xml |

### 2.2.6. Các thành phần trong RESTful API

Trong mô hình giao tiếp dữ liệu giữa Client và Server sử dụng RESTful API gồm 2 thành phần chính đó là: Resquest và Response



**Hình 2.1:** Mô hình giao tiếp giữa Client và Server sử dụng RESTful API

Request là là thông tin gửi từ client lên server. Khi sử dụng trình duyệt và gõ một địa chỉ nào đó thì ngay lập tức trình duyệt sẽ dựa vào tên domain để gửi yêu cầu truy cập đến địa chỉ IP mà domain này đang trỏ tới lúc này phía server sẽ phân tích yêu cầu và sẽ gửi luồng xử lý tới vị trí vị trí lưu trữ của mã nguồn và nhiệm vụ của các mã nguồn là tiếp nhận yêu cầu, phân tích request đó và trả kết quả lại cho client. Một request đúng chuẩn cần có 4 thành phần:

- URL: là địa chỉ duy nhất cho request gửi lên server, thường là đường dẫn tới một hàm xử lí logic và trong RESTful API chính là URI đến API đã được lập trình sẵn trên server

- Method: là phương thức hành động người dùng muốn tác động lên dữ liệu. Có 4 loại method hay được dùng và rất quen thuộc là: GET, POST, PUT, DELETE.

- Headers: nơi chứa các thông tin cần thiết của 1 request nhưng người dùng không biết có sự tồn tại của nó. Ví dụ: độ dài của request body, thời gian gửi request, loại thiết bị đang sử dụng, ...

- Body: nơi chứa thông tin mà người dùng sẽ điền vào để khi lên server cần xử lý các thông tin liên quan của người dùng sẽ sử dụng dữ liệu trong phần body này.

Response: Sau khi nhận được request từ phía người dùng, API sẽ xử lý request đó và gửi ngược lại cho người dùng 1 response phản hổi lại những yêu cầu hoặc thông tin mà người dùng mong muốn. Cấu trúc của 1 response tương đối giống phần request nhưng Status code sẽ thay thế cho URL và Method. Tóm lại, nó có cầu trúc 3 phần chính:

- Status code: là những con số có 3 chữ số và có duy nhất 1 ý nghĩa, ví dụ như vài lỗi thần thánh quen thuộc “404 Not Found” hoặc “503 Service Unavailable”.

- Headers: giống với headers trong request nhưng bây giờ là phần header của response.

- Body: phần response body có thể tùy vào API mà gửi ngược lại cho người dùng trong đó có thể chứa các code trạng thái do API quy định cùng dữ liệu kèm theo nhưng số trang người dùng yêu cầu, số resource trả về tương ứng, hoặc đơn giản là lỗi từ server trả về nếu API có lỗi.

### 2.2.7. Cách làm việc với RESTful API

Các API được chia thành các phiên bản để tiện cho việc cập nhật, mở rộng sau này vì vây các URI của API thường có dạng: \*\*/api/v1/\*\* , \*\*/api/v2/\*\* ,...

Người dùng gửi request đến Server trong đó có chứa phương thức giao tiếp với dữ liệu và các thông tin như token, authentication, ... đến một API đã được Web Service quy định để truy cập vào các resource. Khi request đến server đúng hàm xử lý của API đó, logic code sẽ được thực thi để làm việc với resource sau khi đã truy cập resource thành công thì server trả về response cho người dùng trong đó có quy định status code và phần dữ liệu. Và sau đó xử lý front-end để phân tích các dữ liệu dưới dạng JSON hoặc XML này về đúng với dữ liệu mà người dùng mong muốn.

Thường dữ liệu body truyền lên Server và truyền về Client có dạng XML hoặc JSON để giảm dung lượng gửi giúp tốc độ truy cập API và resource nhanh chóng hơn

Status code cũng có thể tùy biến theo lỗi của ứng dụng và các Web Service có thể gửi một status code đính kèm với resource để phân biệt với Status Code mặc định của giao thức và status code tùy biến này nằm trong phần body của response.

### 2.2.8. Ưu nhược điểm của RESTful API

**2.2.8.1. Ưu điểm của RESTful API**

Thiết kế web trước đây sử dụng SOAP (Simple Object Access Protocol) và WSDL (Web Service Definition Language), tuy nhiên bây giờ REST tối ưu hơn so với 2 phương pháp này.

Rõ ràng về URL (REST URL đại diện cho resource xác định chứ không phải hành động)

Trả về nhiều định dạng khác nhau như: JSON, XML, ...

Hiệu suất tốt, tin cậy, dễ phát triển, mở rộng sau này và phân chia ra từng phiên bản.

**2.2.8.2. Nhược điểm của RESTful API**

Bảo mật kém hơn so với SOAP vì các API dưới dạng mở cho phép các bên thư ba truy cập nên có thể gây khó khăn trong việc xác thực

Do không lưu trạng thái giữa Client - Server nên khối lượng thông tin truyền tải lớn dẫn tới nếu truy vấn một lượng thông tin lớn sẽ gây tốn tài nguyên hệ thống.

Chỉ hoạt động trên giao thức HTTP chủ yếu sử dụng các phương thưc GET, POST, PUT, DELETE

## 2.3. Tìm hiểu RESTful API trong ứng dụng Spring Boot

### 2.3.1. Khái niệm về ứng dụng Spring Boot

Spring Boot là một dự án phát triển trong hệ sinh thái Spring framework. Nó giúp cho các lập trình viên chúng ta đơn giản hóa quá trình lập trình một ứng dụng với Spring, chỉ tập trung vào việc phát triển nghiệp vụ cho ứng dụng. Một ứng dụng Spring Boot có thể dử dụng các thử viện trong hệ sinh thái Spring Framework để làm việc hiệu quả như JPA Data, Spring Security,..

### 2.3.2. Spring Boot REST

Nếu như ứng dụng Spring Boot bình thường chúng ta sử dụng Annotation **@Controller** thì trong một ứng dụng Spring Boot REST ta chỉ cần thao tác thêm Annotaion @RestController thì ứng dụng sẽ trả về là @ResponBody là String dễ dàng cho việc trả về là dữ liệu JSON.

Điều tiếp theo nói đến việc Spring đã hỗ trợ REST rất nhiều đó là các Annotation về Request và Response chúng ta có thể sử dụng các Annotation **@RequestParam** hay **@PathVariable** để các các giá trị truyền từ client lên server một cách nhanh chóng.

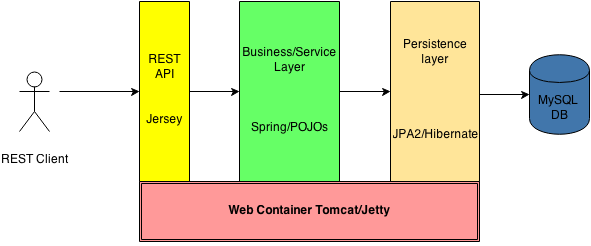
Spring hỗ trợ các phương thức truyền dữ liệu GET, PUT, POST, DELETE nên khi thao tác dữ liệu chỉ cần gọi đến lớp RequestMethod sẽ lấy được kiểu phương thức mong muốn.

Tiếp đến là tính tăng lồng @RequestMapping giúp cho ứng dụng của chúng ta có thể đi sâu vào các API nhỏ nằm trong API cha.



**Hình 2.2:** Ví dụ về các Annotation trong Spring Boot RESTful

### 2.3.3. RESTful API trong mô hình ba lớp Spring Boot

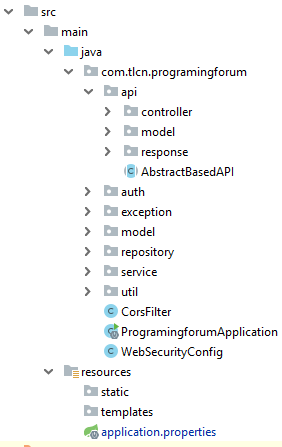
****

**Hình 2.3:** RESTful API trong mô hình ba lớp Spring Boot

Người dùng Client sẽ gửi các request yêu cầu resource qua các giao diện API, tại các REST Endpoint được định nghĩa trong Controller là nơi xử lý logic như xác thực quyền, xử lý resource gọi với tầng Service. Tại tầng Service sẽ gọi đến các tầng Persistence là tầng giao tiếp với Database và trong Spring Boot tằng này chính là Repository làm việc với các Entity là hiện thực hóa của các table tương ứng trong Database. Quá trình này hoàn thành sẽ được trả về cho Controller quyết định nếu có lỗi sẽ trả về Application Exception theo Status Code định nghĩa của ứng dụng tương ứng với resource còn nếu không có lỗi gì báo thành công và trả về resource tương ứng kèm Status Code chứa trong Response.

Endpoint trong Controller được định nghĩa khi người dùng truy cập tới resource, có thể hiểu ở đây Endpoint là các điểm đến của request mà người dùng gửi lên theo REST domain trong Endpoint chứa gói thông tin yêu cầu resource và phương thức giao tiếp với resource.

### 2.3.4. RESTful API phía Backend



**Hình 2.4:** Cấu trúc backend của một ứng dụng Spring Boot theo chuẩn RESTful API

Trong ứng dụng Spring package API sẽ có Controller và Response là các package liên quan đến xử lý phần API trong đó:

- Controller sẽ sử dụng Annotation @RestController để chỉ rõ class Controller đó là một class sử dụng phương thức REST sẽ trả về ResponseBody dạng JSON

- Response sẽ chứa class APIStatus chứa các Status Code do ứng dụng quy định khi giao tiếp với API, ResponseUtil là class giúp trả về cho Client API giao tiếp thành công hoặc xảy ra lỗi hiện Exception với Status Code tương ứng.

- Package Exception sẽ chứa các class ApplicationException xử lý lỗi theo chuẩn API để show ra lỗi cho client.

- Package Model sẽ chứa Entity là package chứa các bảng có trong cơ sở dữ liệu và hiện thực hóa các bảng lên thành các đối tượng Java và để khi thao tác sử dụng JPA Data giúp cho việc giao tiếp với cơ sở dữ liệu nhanh chóng, tiện lợi hơn. Tiếp theo là model Requesst, Response tương ứng là các model giúp cho việc lấy dữ liệu theo API một cách ngắn gọn nhất tránh lấy lên hoặc trả về quá nhiều thông tin không cần thiết.

- Package Repository sẽ là package chứa các class giao tiếp giữa Entity và Database sử dụng các thư viện của Spring như JPA Data hoặc Hibernate.

- Package Service phân chia thành các Service làm các công việc cụ thể để khi Client request tới Controller sẽ gọi đến các hàm trong Service, các hàm trong Service sẽ thao tác với Repository ở trong Service này.

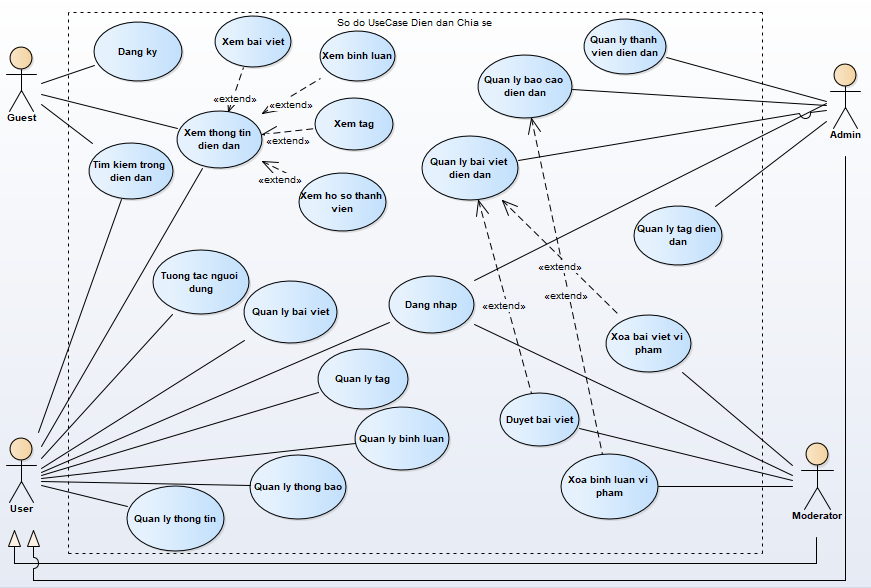
- Package Util sẽ chứa các class cần thiết cho ứng dụng như Constant sẽ chứa các API URL giúp cho quá trình bảo trì và nâng cấp sau này trở nên nhanh chóng và thuận tiện hơn.

- Ngoài ra còn có các class cấu hình ứng dụng Spring Boot để khởi chạy ứng dụng và cấu hình bảo mật Security, cấu hình Annotation trong ứng dụng.

# PHÂN TÍCH YÊU CẦU ỨNG DỤNG

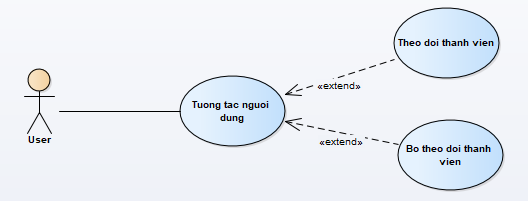
## 3.1. Sơ đồ Use Case

**3.1.1. Sơ đồ Use Case tổng quát**



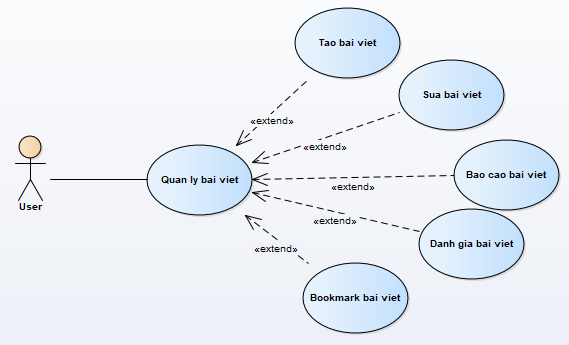
**Hình 3.1:** Sơ đồ Use Case Tổng quát diễn đàn

**3.1.2. Sơ đồ Use Case Tương tác người dùng**



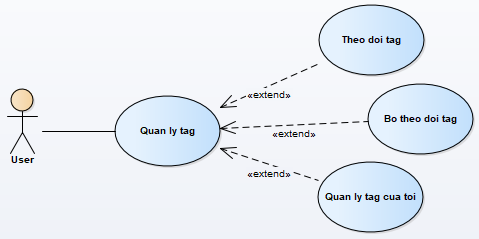
**Hình 3.2:** Sơ đồ Use Case Tương tác người dùng

**3.1.3. Sơ đồ Use Case Quản lý bài viết**



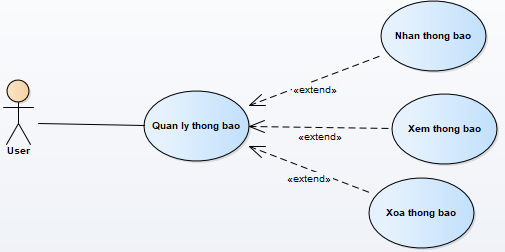
**Hình 3.3:** Sơ đồ Use Case Quản lý bài viết

**3.1.4. Sơ đồ Use Case Tag**



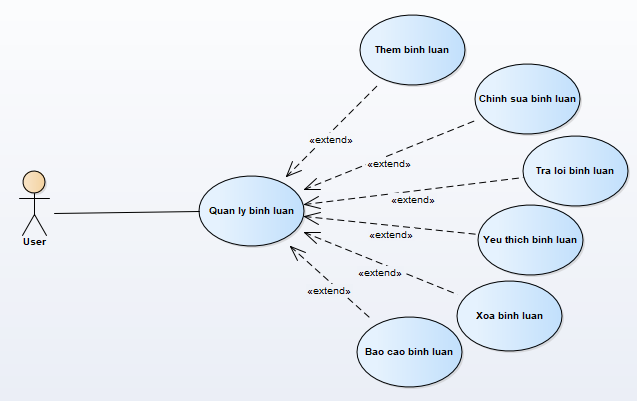
**Hình 3.4:** Sơ đồ Use Case Quản lý Tag

**3.1.5. Sơ đồ Use Case Quản lý Thông báo**



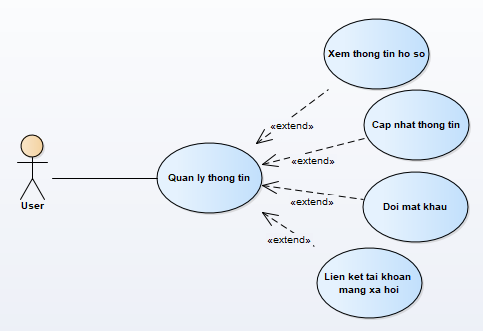
**Hình 3.5:** Sơ đồ Use Case Quản lý thông báo

**3.1.6. Sơ đồ Use Case Quản lý bình luận**



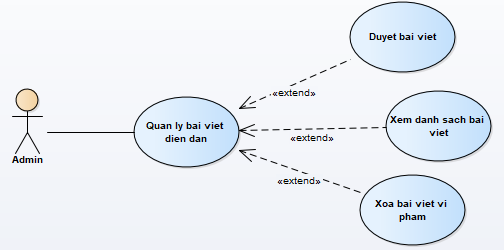
**Hình 3.6:** Sơ đồ Use Case Quản lý bình luận

**3.1.7. Sơ đồ Use Case Quản lý thông tin**



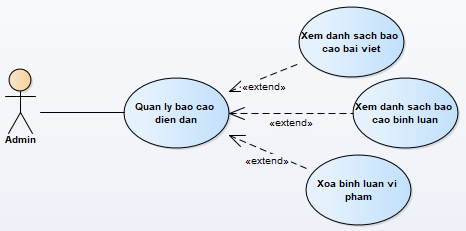
**Hình 3.7:** Sơ đồ Use Case Quản lý thông tin

**3.1.8. Sơ đồ Use Case Quản lý bài viết diễn đàn**



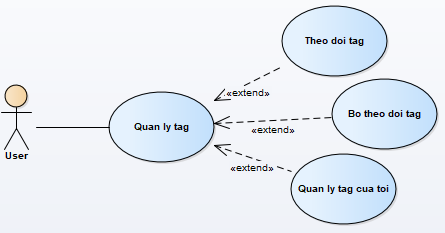
**Hình 3.8:** Sơ đồ Use Case Quản lý bài viết diễn đàn

**3.1.9. Sơ đồ Use Case Quản lý báo cáo diễn đàn**



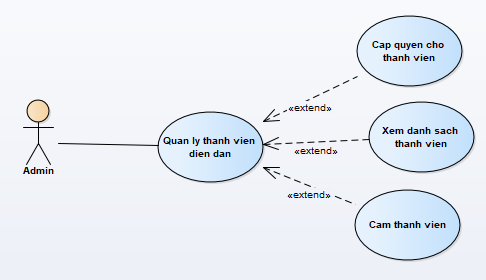
**Hình 3.9:** Sơ đồ Use Case Quản lý báo cáo diễn đàn

**3.1.10. Sơ đồ Use Case Quản lý tag diễn đàn**



**Hình 3.10:** Sơ đồ Use Case Quản lý tag diễn đàn

**3.1.11. Sơ đồ Use Case Quản lý thành viên diễn đàn**



**Hình 3.11:** Sơ đồ Use Case Quản lý thành viên diễn đàn

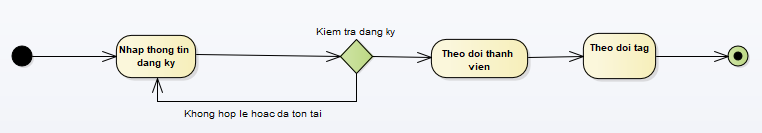
## 3.2. Phân tích nghiệp vụ

### 3.2.1. Nghiệp vụ Đăng ký

**3.2.1.1. Mô tả Use Case**

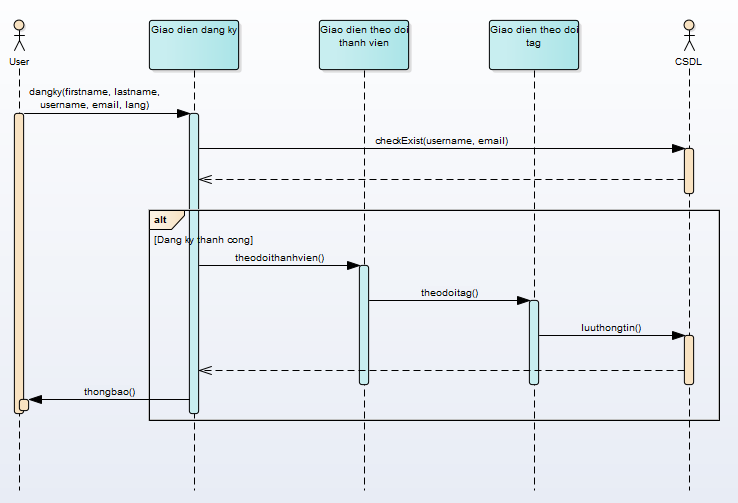
Người dùng có thể đăng ký tài khoản bằng username, email trên trang đăng ký sau đó sẽ được chuyển vào trang theo dõi người dùng và theo dõi tag để người dùng có thể tiện cho việc cập nhật các bài viết mới nhất theo tag hoặc theo người dùng sau này.

**3.2.1.2. Lược đồ hoạt động**



**Hình 3.12:** Lược đồ hoạt động cho nghiệp vụ Đăng ký

**3.2.1.3. Lược đồ tuần tự**



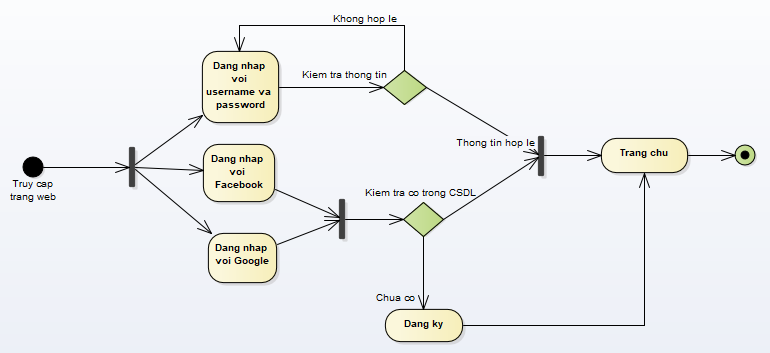
**Hình 3.13:** Lược đồ tuần tự cho nghiệp vụ Đăng ký

### 3.2.2. Nghiệp vụ Đăng nhập

**3.2.2.1. Mô tả nghiệp vụ**

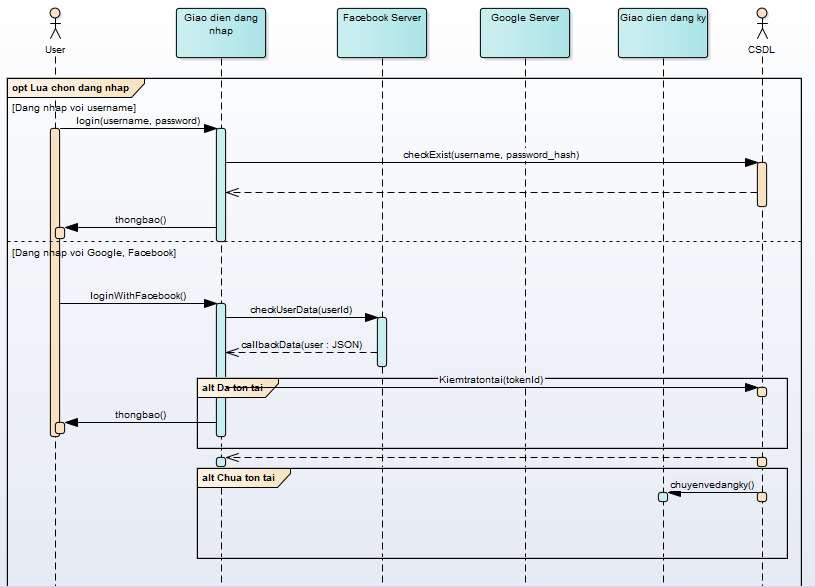
Khách có thể truy cập diễn đàn để theo dõi các bài viết mới, đọc bình luận về các câu hỏi, các tag trên diễn đàn nhưng để đăng bài, bình luận hoặc theo dõi tag thì phải đăng nhập vào diễn đàn. Diễn đàn cũng cấp 3 lựa chọn đăng nhập là đăng nhập với tài khoản mật khẩu khi đăng ký, đăng nhập với Facebook và Google, với việc đăng nhập bằng Facebook và Google người dùng trong lần đăng nhập đầu tiên sẽ kiểm tra coi tài khoản mạng xã hội đã liên kết hay chưa nếu chưa chuyển qua trang hoàn tất hồ sơ cho người dùng trong lần đầu và các lần đăng nhập sau sẽ đăng nhập như đăng nhập với tài khoản mật khẩu bình thường.

**3.2.2.2. Lược đồ hoạt động**



**Hình 3.14:** Lược đồ hoạt động nghiệp vụ Đăng nhập

**3.2.2.3. Lược đồ tuần tự**



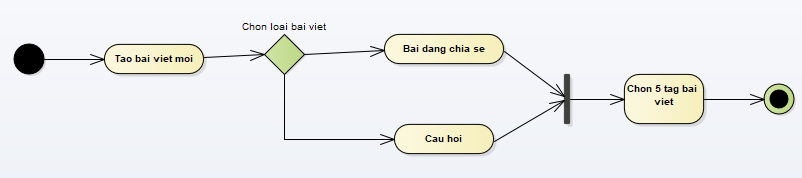
**Hình 3.15:** Lược đồ tuần tự nghiệp vụ Đăng nhập

### 3.2.3. Nghiệp vụ Tạo bài viết mới

**3.2.3.1. Mô tả nghiệp vụ**

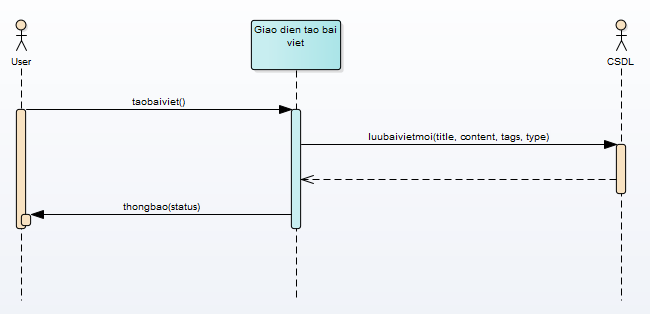
Người dùng sau khi đăng nhập chọn chức năng tạo bài viết mới để đăng bài chia sẻ, trong đăng bài có thể loại bài viết là bài chia sẻ hay câu hỏi cần giải đáp. Người dùng nhập nội dung với định dạng markdown sau đó nhấn đăng bài viết để cập nhật bài viết khi được duyệt bài thì người dùng có thể thấy bài viết của mình trên trang chủ.

**3.2.3.2. Lược đồ hoạt động**



**Hình 3.16:** Lược đồ hoạt động nghiệp vụ Tạo bài viết mới

**3.2.3.3. Lược đồ tuần tự**



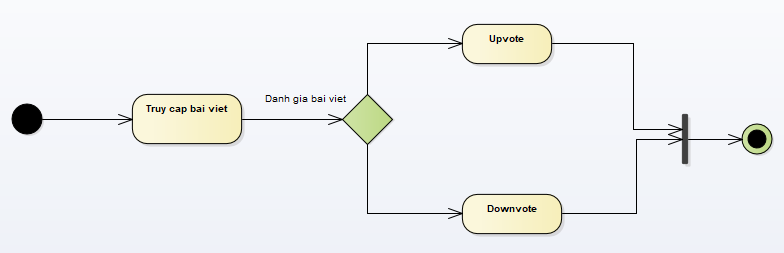
**Hình 3.17:** Lược đồ tuần tự nghiệp vụ Tạo bài viết mới

### 3.2.4. Nghiệp vụ Đánh giá bài viết

**3.2.4.1. Mô tả nghiệp vụ**

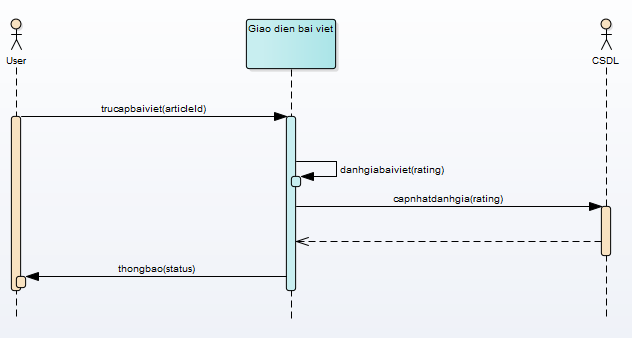
Người dùng có thể đánh giá một bài viết hoặc một câu hỏi là tốt hoặc không tốt thông qua chức năng “upvote” và “downvote”. Từ đó tạo nên các đánh giá khách quan nhất cho một bài viết nếu có thể có nhiều người tham gia đánh giá. Khi người dùng đánh giá “upvote” hoặc “downvote” chỉ số đánh giá bài viết sẽ thay đổi theo sự đánh giá lần lượt là +1 hoặc -1.

**3.2.4.2. Lược đồ hoạt động**



**Hình 3.18:** Lược đồ hoạt động nghiệp vụ Đánh giá bài viết

**3.2.4.3. Lược đồ tuần tự**



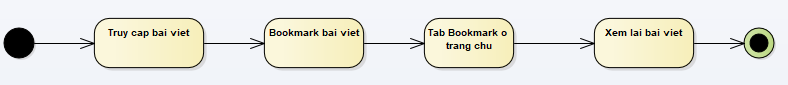
**Hình 3.19:** Lược đồ tuần tự nghiệp vụ Đánh giá bài viết

### 3.2.5. Nghiệp vụ Lưu trữ bài viết (Bookmark)

**3.2.5.1. Mô tả nghiệp vụ**

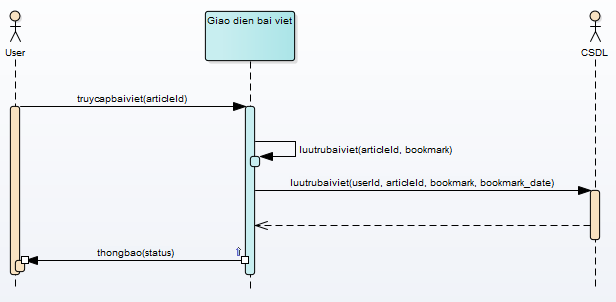
Người dùng có thể lưu trữ lại bài viết hoặc câu hỏi hay sau này có thể truy cập xem lại thì chọn chức năng bookmark bài viết. Chức năng này hỗ trợ một tab Lưu trữ ở trang chủ khi xem lại người dùng có thể lọc theo loại bài viết là bài đăng hay câu hỏi và thời gian đã bookmark.

**3.2.5.2. Lược đồ hoạt động**



**Hình 3.20:** Lược đồ hoạt động nghiệp vụ Lưu trữ bài viết

**3.2.5.3. Lược đồ tuần tự**



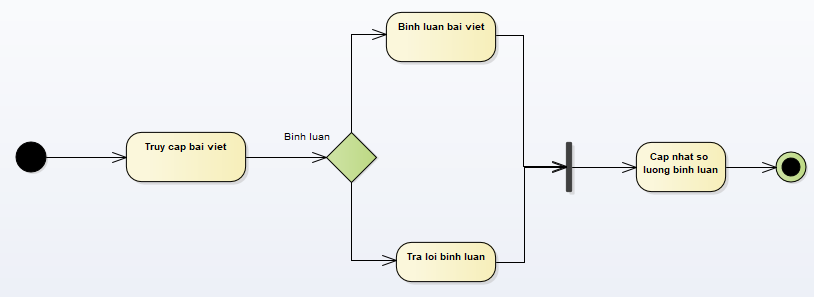
**Hình 3.21:** Lược đồ tuần tự nghiệp vụ Lưu trữ bài viết

### 3.2.6. Nghiệp vụ Bình luận

**3.2.6.1. Mô tả nghiệp vụ**

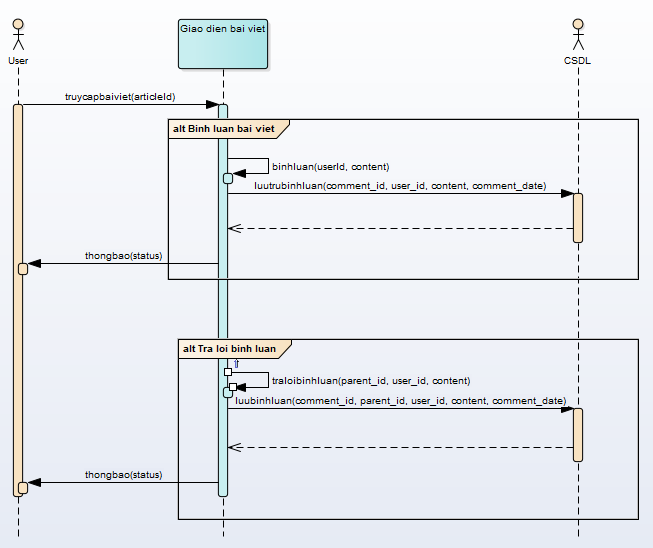
Thành viên tham gia diễn đàn có thể bình luận hoặc trả lời câu hỏi trong phần bình luận, phần bình luận chia theo cấp 1 cha – n con, 1 bình luận cha có thể có nhiều bình luận con tham gia. Nghiệp vụ bình luận này có thể dùng để đánh giá lượng Trending của một bài viết theo mức độ người quan tâm và người dùng có thể tương tác với nhau trong phần bình luận này hoặc đánh dấu bình luận là câu trả lời đúng nhất hoặc có thể tham gia báo cáo các bình luận vi phạm đến quản trị viên hoặc người quản lý diễn đàn

**3.2.6.2. Lược đồ hoạt động**



**Hình 3.22:** Lược đồ hoạt động nghiệp vụ Bình luận

**3.2.6.3. Lược đồ tuần tự**



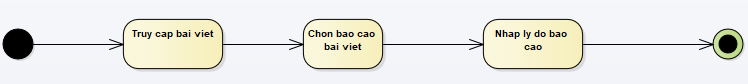
**Hình 3.23:** Lược đồ tuần tự nghiệp vụ Bình luận

### 3.2.7. Nghiệp vụ Báo cáo bài viết

**3.2.7.1. Mô tả nghiệp vụ**

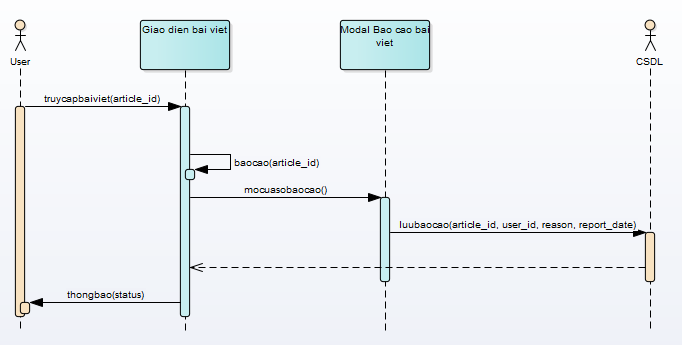
Trong một diễn đàn chắc chắn không thể chỉ bài viết hay mà còn có các bài viết vi phạm vì vậy chức năng báo cáo bài viết hoặc câu hỏi spam ra đời giúp cho việc quản lý diễn đàn tốt hơn. Người dùng có thể chọn chức năng báo cáo bài viết vi phạm từ đó giúp chon quản trị viên diễn đàn kịp thời chặn xóa những bài viết gây ảnh hưởng cộng đồng.

**3.2.7.2. Lược đồ hoạt động**



**Hình 3.24:** Lược đồ hoạt động nghiệp vụ Báo cáo bài viết

**3.2.7.3. Lược đồ tuần tự**



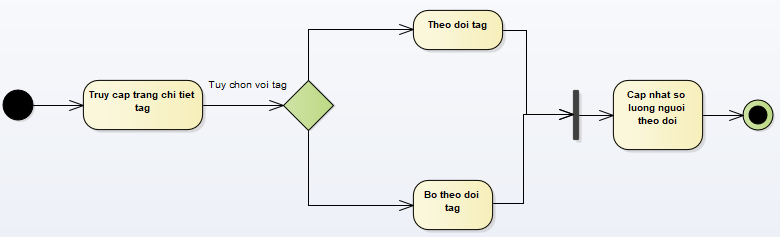
**Hình 3.25:** Lược đồ tuần tự nghiệp vụ Báo cáo bài viết

### 3.2.8. Nghiệp vụ Theo dõi tag/ Bỏ theo dõi tag

**3.2.8.1. Mô tả nghiệp vụ**

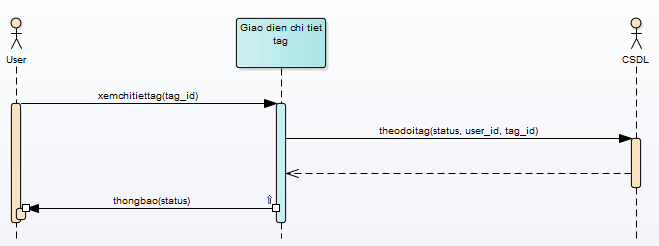
Ngay từ lúc đăng ký người dùng đã có thể chọn theo dõi các Tag để cập nhật bài viết về các tag liên quan. Khi người dùng chọn theo dõi tag thì số lượng người thoe dõi cũng tăng lên và ngược lại bỏ theo dõi thì số lượng này thay đổi trong trang chi tiết của tag. Trong trang chi tiết tag người dùng cũng có thể xem các bài viết có gắn tag và các câu hỏi cùng với các người theo dõi tag đó.

**3.2.8.2. Lược đồ hoạt động**



**Hình 3.26:** Lược đồ hoạt động nghiệp vụ Theo dõi/ Bỏ theo dõi Tag

**3.2.8.3. Lược đồ tuần tự**



**Hình 3.27:** Lược đồ tuần tự nghiệp vụ Theo dõi/ Bỏ theo dõi Tag

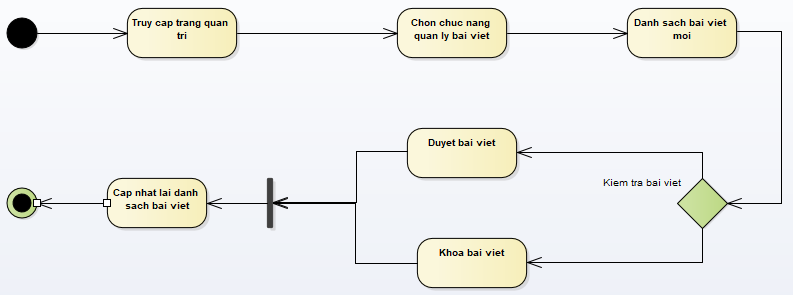
Ở đây nhóm dựa vào status để lưu vào cơ sở dữ liệu nêu không chia làm 2 trường hợp nên khi thực hiện lược đồ tuần tự chỉ dựa vào status mà xử lý logic code.

### 3.2.9. Nghiệp vụ Duyệt bài viết

**3.2.9.1. Mô tả nghiệp vụ**

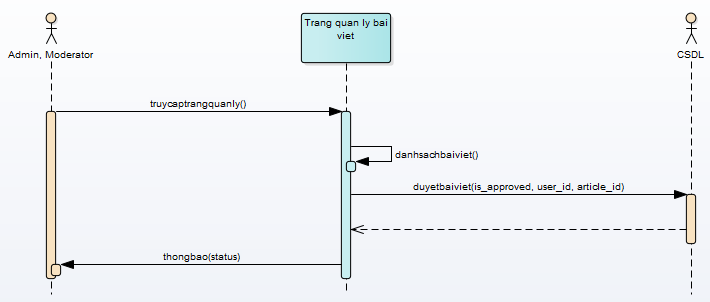
Admin hoặc Moderator có thể duyệt bài viết mới chọn trạng thái là chấp nhận bài viết hoặc không chấp nhận bài viết. Chức năng này giúp cho việc quản lý đầu vào tốt nhất có thể chỉ cho phép các bài viết chất lượng hoặc câu hỏi thì có thể xuất hiện trên trang chủ. Chức năng báo cáo bài viết sẽ hỗ trợ cho chức năng duyệt bởi vì có thể các bài viết đó người kiểm duyệt cảm thấy phù hợp nhưng các thành viên trong diễn đàn cảm thấy không tốt thì có thể báo cáo cho người kiểm xuyệt xem xét.

**3.2.9.2. Lược đồ hoạt động**



**Hình 3.28:** Lược đồ hoạt động nghiệp vụ Duyệt bài viết

**3.2.9.3. Lược đồ tuần tự**



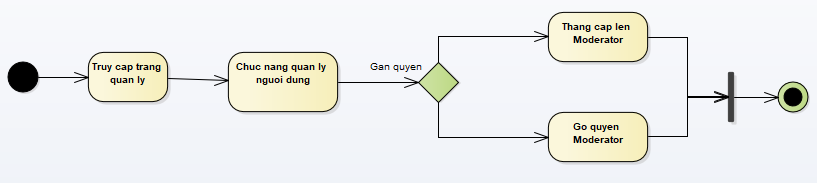
**Hình 3.29:** Lược đồ tuần tự nghiệp vụ Duyệt bài viết

### 3.2.10. Nghiệp vụ Gán vai trò cho người dùng

**3.2.10.1. Mô tả nghiệp vụ**

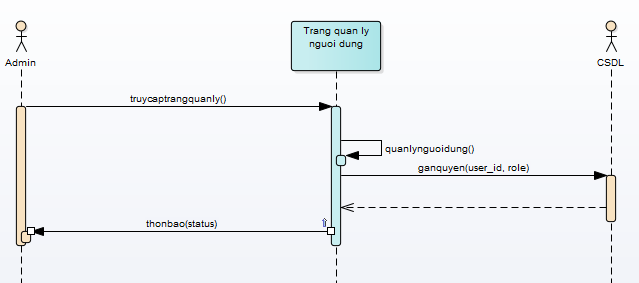
Vai trò Admin có thể thực hiện chức năng gán vai trò cho người dùng thành viên. Admin có thể cho thành viên trở thành Moderator giúp cho mình kiểm duyệt bài viết, kiểm duyệt báo cáo vi phạm, khóa các bài viết và bình luận vi phạm. Và khi Moderator không hoạt động đúng với vai trò của mình thì có thể bị gỡ quyền trở về lại với vai trò thành viên bình thường.

**3.2.10.2. Lược đồ hoạt động**



**Hình 3.30:** Lược đồ hoạt động nghiệp vụ Gán vai trò người dùng

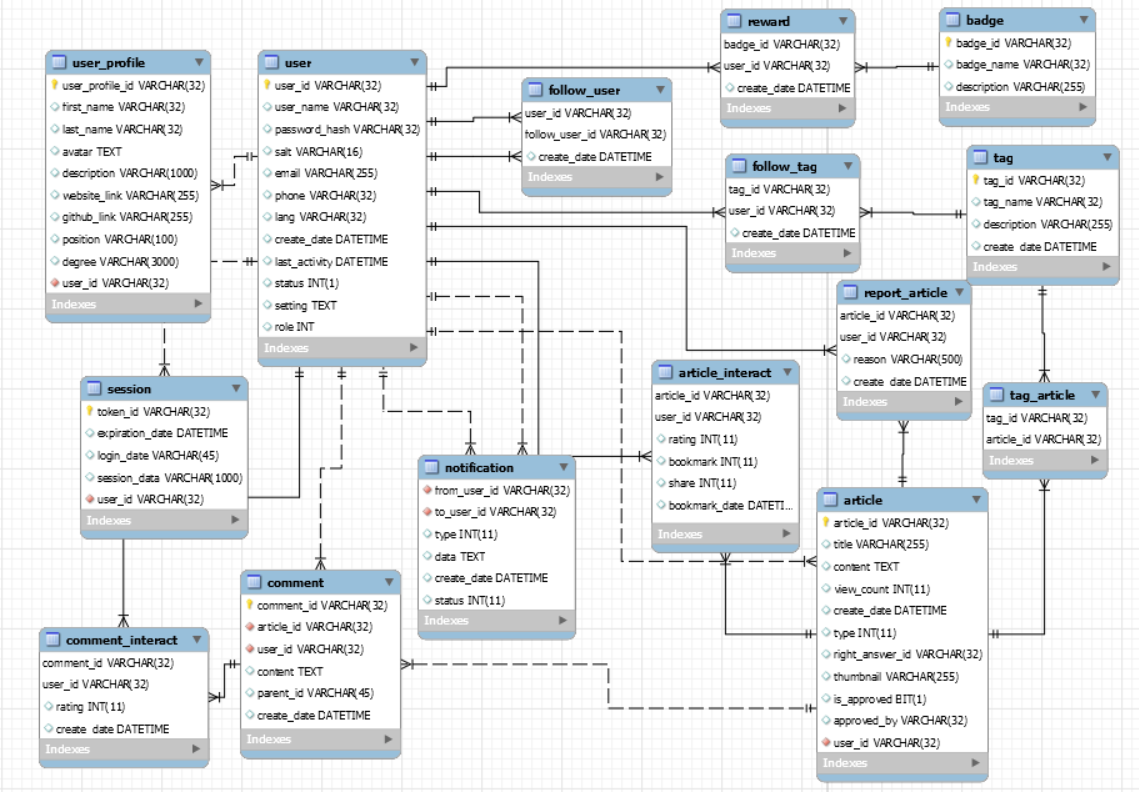
**3.2.10.3. Lược đồ tuần tự**



**Hình 3.31:** Lược đồ tuần tự nghiệp vụ Gán vai trò người dùng

# THIẾT KẾ CƠ SỞ DỮ LIỆU

## 4.1. Lược đồ ERD



**Hình 4.1:** Lược đồ ERD mô tả quan hệ giữa các bảng trong cơ sở dữ liệu

## 4.2. Bảng mô tả thuộc tính

**4.2.1. Bảng user**

**Bảng 4.1:** Bảng mô tả thuộc tính bảng user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** | **Kiểu dữ liệu và kích thước** | **Ghi chú** |
| 1 | user\_id | Mã người dùng | varchar(32) | Khóa chính |
| 2 | user\_name | Tên người dùng | varchar(32) |  |
| 3 | password\_hash | Mật khẩu người dùng đã được hash | varchar(32) |  |
| 4 | salt | Mã salt để đọc mật khẩu hash | varchar(32) |  |
| 5 | email | Mật khẩu người dùng | varchar(255) |  |
| 6 | phone | Số điện thoại người dùng | varchar(32) |  |
| 7 | lang | Ngôn ngữ của người dùng | varchar(32) |  |
| 8 | create\_date | Ngày tạo tài khoản người dùng | datetime |  |
| 9 | last\_activity | Ngày chỉnh sửa cuối cùng | datetime |  |
| 10 | setting | Cài đặt của người dùng | text |  |
| 11 | role | Vai trò của người dùng  1. Thành viên  2. Moderator  3. Admin | int(11) |  |
| 12 | status | Trạng thái của tài khoản người dùng là hoạt động hay đã bị khóa | int(11) |  |

**4.2.2. Bảng user\_profile**

**Bảng 4.2:** Bảng mô tả thuộc tính bảng user\_profile

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** | **Kiểu dữ liệu và kích thước** | **Ghi chú** |
| 1 | user\_profile\_id | Mã bảng user\_profile | varchar(32) | Khóa chính |
| 2 | first\_name | Tên của người dùng | varchar(32) |  |
| 3 | last\_name | Họ của người dùng | varchar(32) |  |
| 4 | avatar | Ảnh đại diện của người dùng | text |  |
| 5 | description | Mô tả của người dùng | varchar(1000) |  |
| 6 | website\_link | Đường dẫn tới trang web của người dùng hoặc blog,... | varchar(255) |  |
| 7 | github\_link | Đường dẫn tới Github cá nhân của người dùng | varchar(255) |  |
| 8 | position | Chức vụ hiện tại đang làm của người dùng | varchar(100) |  |
| 9 | company | Công ty người dùng đang làm việc | varchar(255) |  |
| 10 | user\_id | Mã người dùng để liên kết với bảng user | varchar(32) | Khóa ngoại |

**4.2.3. Bảng article**

**Bảng 4.3:** Bảng mô tả thuộc tính bảng article

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** | **Kiểu dữ liệu và kích thước** | **Ghi chú** |
| 1 | article\_id | Mã bài viết | varchar(32) | Khóa chính |
| 2 | user\_id | Mã người đăng bài | varchar(32) | Khóa ngoại |
| 3 | title | Tiêu đề bài viết | varchar(255) |  |
| 4 | content | Nội dung với định dạng markdown | Text |  |
| 5 | view\_count | Số lượng người xem bài viết | int(11) |  |
| 6 | create\_date | Ngày tạo bài viết | datetime |  |
| 7 | type | Loại bài viết  1. Bài đăng chia sẻ  2. Bài đăng hỏi đáp | int(11) |  |
| 8 | right\_answer\_id | Câu trả lời đúng được đánh dấu | varchar(32) |  |
| 9 | is\_approved | Trạng thái bài đăng đã được kiểm duyệt hay chưa | bit(1) |  |
| 10 | status | Trạng thái bài viết đã bị xóa hya chưa | int(11) |  |
| 11 | approved\_by | Mã người duyệt bài | varchar(32) |  |

**4.2.4. Bảng article\_interact**

**Bảng 4.4:** Bảng mô tả thuộc tính bảng article\_interact

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** | **Kiểu dữ liệu và kích thước** | **Ghi chú** |
| 1 | article\_id | Mã bài viết | varchar(32) | Khóa chính |
| 2 | user\_id | Mã người tương tác bài viết | varchar(32) | Khóa chính |
| 3 | rating | Đánh giá của người đọc với bài viết  0. Chưa đánh giá  -1. Bài viết kém chất lượng  1. Bài viết hay | int(11) |  |
| 4 | bookmark | Trạng thái người dùng lưu trữ bài viết vào danh sách bookmark  1. Đã lưu  0. Chưa lưu | int(11) |  |
| 5 | share | Trại thái người dùng chia sẻ bài viết  1. Đã chia sẻ  0. Chưa chia sẻ | int(11) |  |
| 6 | bookmark\_date | Ngày người dùng lưu trữ bài viết | datetime |  |

**4.2.5. Bảng comment**

**Bảng 4.5:** Bảng mô tả thuộc tính bảng comment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** | **Kiểu dữ liệu và kích thước** | **Ghi chú** |
| 1 | comment\_id | Mã bình luận | varchar(32) | Khóa chính |
| 2 | article\_id | Mã bài viết chứa bình luận | varchar(32) | Khóa ngoại |
| 3 | user\_id | Mã người bình luận | varchar(32) | Khóa ngoại |
| 4 | content | Nội dung bình luận | text |  |
| 5 | parent\_id | Mã bình luận cha nếu đây là bình luận cấp con | varchar(32) |  |
| 6 | create\_date | Ngày người dùng bình luận | datetime |  |
| 7 | status | Trạng thài tồn tại của bình luận | int(11) |  |

**4.2.6. Bảng comment\_interact**

**Bảng 4.6:** Bảng mô tả thuộc tính bảng user\_interact

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** | **Kiểu dữ liệu và kích thước** | **Ghi chú** |
| 1 | comment\_id | Mã bình luận | varchar(32) | Khóa chính |
| 2 | user\_id | Mã người tương tác bình luận | varchar(32) | Khóa chính |
| 3 | rating | Trạng thái tương tác với bình luận | int(11) |  |
| 4 | create\_date | Ngày tương tác | datetime |  |

**4.2.7. Bảng tag**

**Bảng 4.7:** Bảng mô tả thuộc tính bảng tag

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** | **Kiểu dữ liệu và kích thước** | **Ghi chú** |
| 1 | tag\_id | Mã thẻ trong diễn đàn | varchar(32) | Khóa chính |
| 2 | tag\_name | Tên của thẻ trong diễn đàn | varchar(32) |  |
| 3 | description | Mô tả về thẻ | varchar(255) |  |
| 4 | create\_date | Ngày tương tác | datetime |  |

**4.2.8. Bảng tag\_article**

**Bảng 4.8:** Bảng mô tả thuộc tính bảng tag\_article

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** | **Kiểu dữ liệu và kích thước** | **Ghi chú** |
| 1 | tag\_id | Mã thẻ trong diễn đàn | varchar(32) | Khóa chính |
| 2 | article\_id | Mã bài viết có chưa thẻ | varchar(32) | Khóa chính |

**4.2.9. Bảng report\_article**

**Bảng 4.9:** Bảng mô tả thuộc tính bảng report\_article

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** | **Kiểu dữ liệu và kích thước** | **Ghi chú** |
| 1 | article\_id | Mã bài viết bị báo cáo | varchar(32) | Khóa chính |
| 2 | user\_id | Mã người báo cáo | varchar(32) | Khóa chính |
| 3 | reason | Lý do báo cáo vi phạm | varchar(500) |  |
| 4 | create\_date | Ngày tương tác | datetime |  |

**4.2.10. Bảng report\_comment**

**Bảng 4.10:** Bảng mô tả thuộc tính bảng report\_comment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** | **Kiểu dữ liệu và kích thước** | **Ghi chú** |
| 1 | comment\_id | Mã bình luận bị báo cáo | varchar(32) | Khóa chính |
| 2 | user\_id | Mã người báo cáo | varchar(32) | Khóa chính |
| 3 | reason | Lý do báo cáo vi phạm | text |  |
| 4 | create\_date | Ngày tương tác | datetime |  |

**4.2.11. Bảng session**

**Bảng 4.11:** Bảng mô tả thuộc tính bảng session

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** | **Kiểu dữ liệu và kích thước** | **Ghi chú** |
| 1 | token\_id | Mã Access Token khi người dùng đăng nhập | varchar(32) | Khóa chính |
| 2 | expiration\_date | Ngày hết hạn của token | datetime |  |
| 3 | login\_date | Ngày đăng nhập của người dùng | datetime |  |
| 4 | sesion\_data | Dữ liệu người dùng đăng nhập | varchar(1000) |  |
| 5 | user\_id | Mã người dùng đăng nhập | varchar(32) | Khóa ngoại |

**4.2.12. Bảng follow\_tag**

**Bảng 4.12:** Bảng mô tả thuộc tính bảng follow\_tag

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** | **Kiểu dữ liệu và kích thước** | **Ghi chú** |
| 1 | tag\_id | Mã tag được theo dõi | varchar(32) | Khóa chính |
| 2 | user\_id | MÃ người dùng theo dõi tag | varchar(32) | Khóa chính |
| 3 | create\_date | Ngày theo dõi tag | datetime |  |

**4.2.13. Bảng notification**

**Bảng 4.13:** Bảng mô tả thuộc tính bảng notification

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** | **Kiểu dữ liệu và kích thước** | **Ghi chú** |
| 1 | from\_user\_id | Mã người dùng được tương tác thông báo | varchar(32) | Khóa ngoại |
| 2 | to\_user\_id | Mã người dùng nhận thông báo | varchar(32) | Khóa ngoại |
| 3 | type | Loại thông báo | int(11) |  |
| 4 | data | Thông tin của thông báo | text |  |
| 5 | create\_date | Ngày thông tạo | datetime |  |
| 6 | status | Trạng thái tồn tại của thông báo | int(11) |  |

# CÀI ĐẶT ỨNG DỤNG

## 5.1. Ngôn ngữ lập trình

### 5.1.1. Ngôn ngữ lập trình Java

Java là ngôn ngữ lập trình hướng đối tượng, đa mục đích và có khả năng làm việc hầu như trên bất kỳ nền tảng nào mà không cần phải biên dịch lại.

Việc chọn ngôn ngữ Java để lâp trình ứng dụng Android là vì:

- Đơn giản: Java được thiết kế để dễ sử dụng, viết, biên dịch, gỡ lỗi và học hơn các ngôn ngữ lập trình khác.

- Hướng đối tượng: Cho phép tạo các chương trình mô-đun và mã có thể tái sử dụng.

- Không phụ thuộc vào nền tảng: Khả năng di chuyển dễ dàng từ hệ thống máy tính này sang hệ thống khác

- Tools và library: Có lượng API, công cụ và thư viện hỗ trợ phong phú

- Bảo mật: Ngôn ngữ Java, trình biên dịch, trình thông dịch và runtime environtment đều được phát triển với sự chú trọng về bảo mật.

### 5.1.2. Ngôn ngữ lập trình TypeScript

Typescript là một dự án mã nguồn mở được Microsoft phát triển, được xem là một phiên bản nâng cao của Javascript.

TypeScript là một ngôn ngữ giúp cung cấp quy mô lớn hơn so với JavaScript.

Typescript được xem là phiên bản nâng cao của Javascript vì nó được bổ sung những tùy chọn kiểu tĩnh và các lớp hướng đối tượng, nó bao hàm luôn ES6(ECMAScript 6 2015) - phiên bản mới nhất của Javascript.

TypeScript thêm các namespace, class và module tùy chọn vào JavaScript. TypeScript hỗ trợ các công cụ cho các ứng dụng JavaScript quy mô lớn cho bất kỳ trình duyệt nào, cho bất kỳ máy chủ nào, trên bất kỳ hệ điều hành nào.

### 5.1.3. Ngôn ngữ truy vấn HQL trong JPA Data

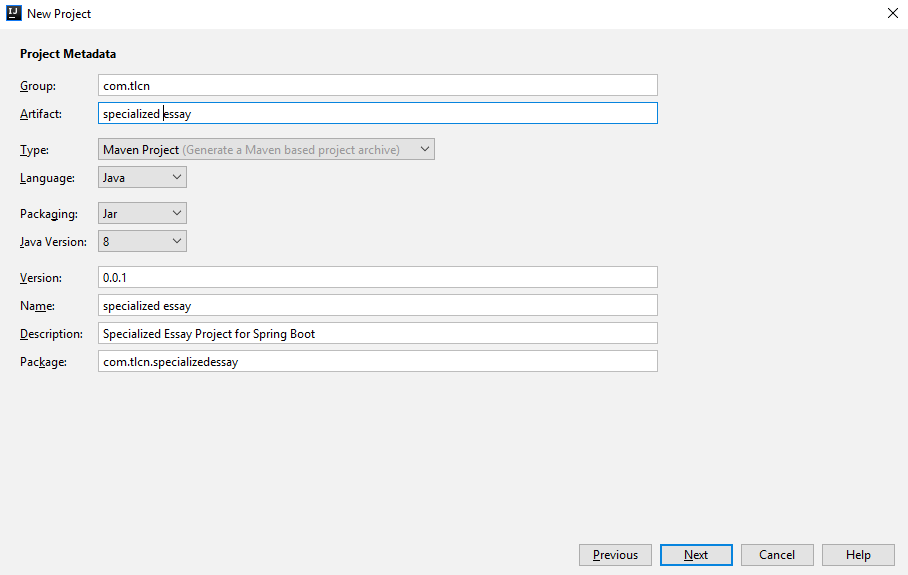
Hibernate Query Language (HQL) là một ngôn ngữ truy vấn hướng đối tượng, tương tự như SQL nhưng thay vì làm việc trên các bảng và cột, HQL làm việc với các đối tượng Entity trong ứng dụng Spring Boot và JPA Data và các thuộc tính của chúng. Các truy vấn HQL được dịch bởi Hibernate thành các truy vấn SQL thông thường, lần lượt thực hiện các công việc trên cơ sở dữ liệu.

HQL đước sử dụng để tránh sự phức tạp về tính linh hoạt của cơ sở dữ liệu và tận dụng chiến lược generation và caching của Hibernate.

Các từ khoá như SELECT, FROM và WHERE,... không phân biệt chữ hoa và chữ thường nhưng các thuộc tính như tên bảng và cột có phân biệt chữ hoa chữ thường trong HQL.

## 5.2. Cài đặt ứng dụng phía backend

### 5.2.1. Khởi tạo ứng dụng Spring Boot

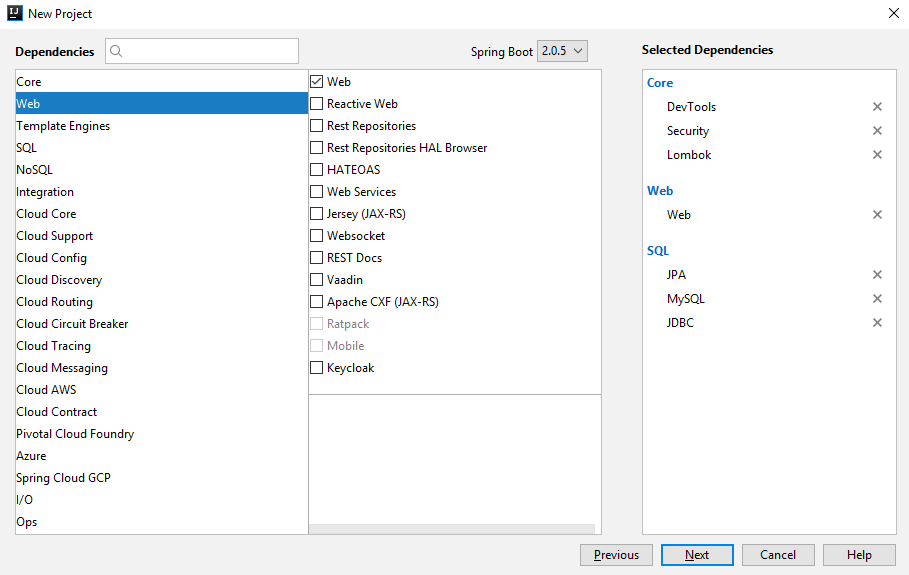


**Hình 5.1:** Khởi tạo ứng dụng Spring Boot

Nhóm báo cáo thực hiện chọn IDE IntelliJ cho việc viết ứng dụng vì nó hỗ trợ tốt cho Spring Boot và các plugin liên quan thuận tiện cho việc xây dựng ứng dụng

Sử dụng phiên bản Java 1.8

Tiến hành nhập các cấu hình mặc định và sử dụng Maven để import thư viện

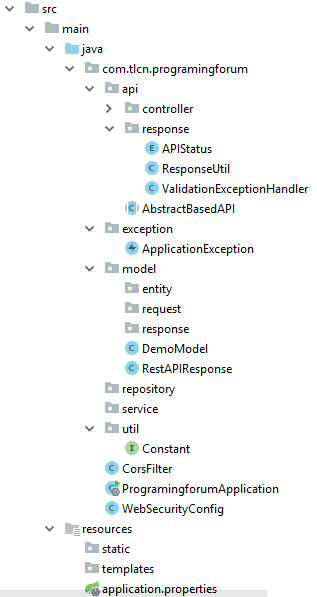


**Hình 5.2:** Chọn các dependency cần thiết cho một ứng dụng Spring Boot

Thêm các dependency cần thiết cho dự án như DevTools, Security, Lombok, Web , JPA, MySQL, JDBC

Server khởi chạy ứng dụng Spring Boot được nhúng trong gói Spring Starter nên không cần cài đặt Server ngoài như Tomcat hoặc GrassFish

Sau khi đã tiến hành chọn các gói thư viện sử dụng sẽ tới bước tạo project và thêm các package như hình:



**Hình 5.3:** Cấu trúc thư mục ứng dụng Spring Boot

### 5.2.2. Thêm các dependency cho ứng dụng

Vì ứng dụng web Spring Boot quản lý các depencency thông qua Maven nên trước hết may phải có cài đặt Maven.



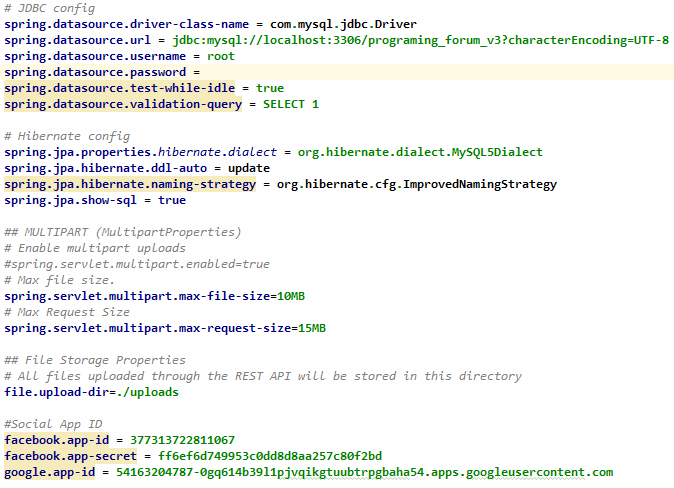
**Hình 5.4:** Thêm các thư viện dependency cho ứng dụng trong pom.xml

### 5.2.3. Cấu hình application.properties cho ứng dụng Spring Boot

Một trong những ưu điểm của Spring Boot đó là cấu hình có thể làm nhanh chóng thông qua các hướng dẫn trên trang chủ và cấu trình trong tập tin .java thay vì khó khăn cấu hình trong các tập tin .xml như trước đây. Cấu hình các thuộc tính cần dùng trong ứng dụng cũng như các khóa làm việc, các giá trị, đường dẫn trong ứng dụng có thể tập hợp và cấu hình một lần trong application.properties.

Bên dưới là các thông số cấu hình cần thiết cho ứng dụng Spring Boot RESTful của nhóm báo cáo như JDBC, đường dẫn tới database, tài khoản đăng nhập server MySQL, các cấu hình khi làm việc với JPA Data.

Tiếp đến là cấu hình các key nếu làm việc với các bên thư ba như Facebook, Google, Box Service vì các bên thư ba này yêu cầu mã khóa mới cho phép ta truy cập và các nguồn tài nguyên API của họ.

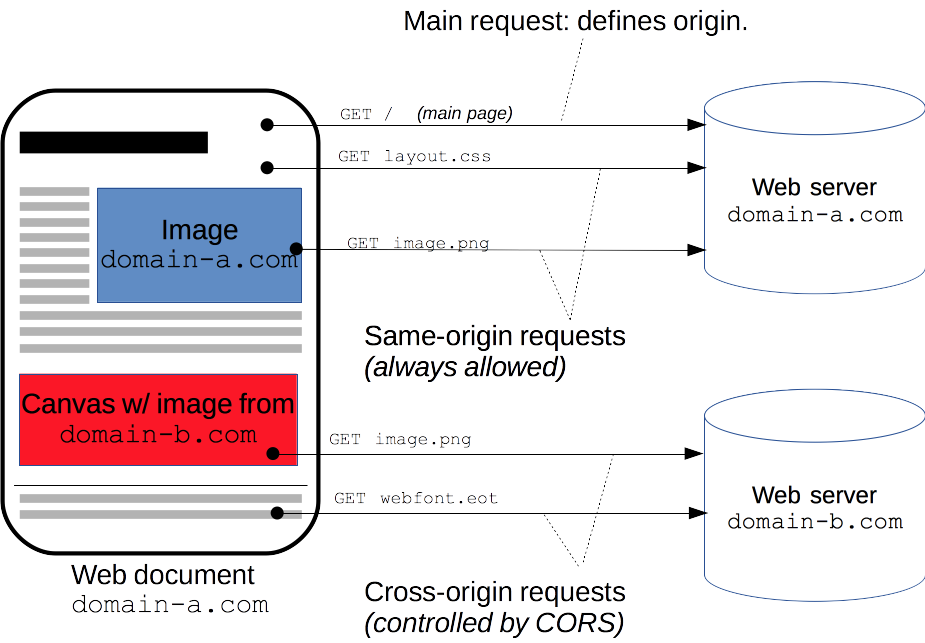


**Hình 5.5:** Cấu hình tập tin application.properties cho ứng dụng Spring Boot

### 5.2.4. Cấu hình CORS cho ứng dụng làm việc với API

Một tài nguyên tạo một yêu cầu cross-origin HTTP khi nó gửi yêu cầu từ một tên miền, hoặc cổng khác mà không phải là chính nó. Ví dụ, một trang HTML được trả về từ http://domain-a.com , gửi một yêu cầu <img> src đến http://domain-b.com/image.jpg. Nhiều trang web ngày nay thường lấy tài nguyên như file css, hình ảnh và các tập lệnh từ các tên miền khác.

Vì lí do bảo mật, các trình duyệt hạn chế các yêu cầu cross-origin HTTP bắt nguồn từ trong các tập lệnh. Ví dụ, XMLHttpRequest và Fetch theo sau same-origin policy. Vì thế, một ứng dụng web sử dụng XMLHttpRequest hoặc Fetch chỉ có thể tạo yêu cầu HTTP đến tên miền của chính nó. Để cải thiện ứng dụng web, các nhà phát triển yêu cầu browser vendor cho phép các yêu cầu cross-domain.



**Hình 5.6:** Cơ chế hoạt động của CORS

Cơ chế Cross-Origin Resource Sharing (CORS) cho phép các máy chủ web kiểm soát truy cập cross-domain, cơ chế này cho phép bảo mật dữ liệu truyền qua cross-domain. Trình duyệt mới sử dụng CORS trong một API container - như là XMLHttpRequest hoặc Fetch - để giảm thiểu rủi ro của yêu cầu cross-origin HTTP.

Cấu hình CORS trong ứng dụng để cho phép client gửi yêu cập với server của ứng dụng như sau:

@Override  
**public void** doFilter(ServletRequest req, ServletResponse res, FilterChain chain) **throws** IOException, ServletException {  
 HttpServletResponse response = (HttpServletResponse) res;  
 response.setHeader(**"Access-Control-Allow-Origin"**, **"\*"**);  
 response.setHeader(**"Access-Control-Allow-Methods"**, **"POST, OPTIONS, PUT, DELETE"**);  
 response.setHeader(**"Access-Control-Max-Age"**, **"3600"**);  
 response.setHeader(**"Access-Control-Allow-Headers"**, **"Content-Type, x-requested-with, X-Custom-Header, X-Access-Token, X-Frame-Options"**);  
 chain.doFilter(req, res);  
}

Đầu tiên phải override lại hàm doFilter của Servlet để tiến hành set các header của ứng dụng. Tiếp theo là set header cho các phương thức chúng ta cho phép giao tiếp với Server là POST, OPTIONS, PUT, DELETE và set hedaer thời gian tồn tại trong trong cơ chế kiểm soát truy cập. Và cuối cùng là chấp nhận các header nào được phép đi vào ứng dụng.

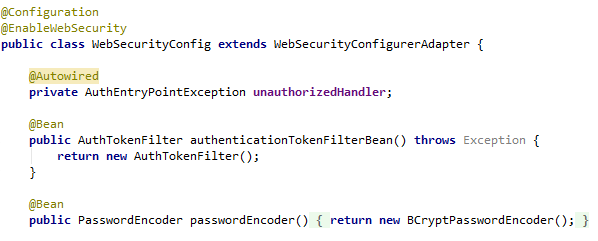
### 5.2.5. Cấu hình Spring Security

Một trong những yếu tố tạo nên thành công của hệ sinh thái Spring Framework đó chính là Spring Security với tính năng bảo mật vượt trội cho phép kiểm soát tối đa request và response của ứng dụng.

Lợi ích lớn nhất của Spring Security là giúp tích hợp tính năng xác thực và phân quyền một cách dễ dàng vào ứng dụng.

Chống lại CSRF attack Bảo vệ Session Fixation Mã hóa mật khẩu. Cache control X-XSS-Protection …

Cấu hình Spring Security của nhóm như sau:



**Hình 5.7:** Cấu hình AuthTokenFilter của ứng dụng

Cấu hình lại hàm AuthTokenFilter theo token của ứng dụng và hàm mã hõa password sẽ sử dụng dựa trên Spring Security.

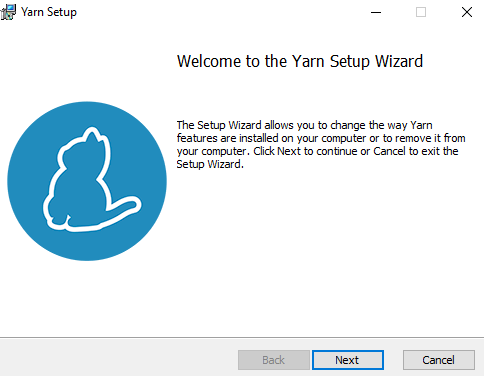
@Override  
 **protected void** configure(HttpSecurity http) **throws** Exception {  
 http  
 *// we don't need CSRF because our token is invulnerable* .csrf().disable()  
 .exceptionHandling().authenticationEntryPoint(**unauthorizedHandler**).and()  
 *// don't create session* .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.***STATELESS***).and()  
 .authorizeRequests()  
 *// Allow access public resource* .antMatchers(  
 HttpMethod.***GET***,  
 **"/"**,  
 **"/upload/\*\*"**,  
 **"/\*.html"**,  
 **"/favicon.ico"**,  
 **"/\*\*/\*.html"**,  
 **"/\*\*/\*.css"**,  
 **"/\*\*/\*.js"**,  
 **"/\*\*/\*.png"**,  
 **"/\*\*/\*.gif"** ).permitAll()  
 *// allow CORS option calls* .antMatchers(HttpMethod.***OPTIONS***, **"/api/v1/\*\*"**).permitAll()  
 .antMatchers(  
 **"/api/v1/auth/login"**,  
 **"/api/v1/auth/admin/login"**,  
 **"/api/v1/user/signup"**,  
 **"/api/v1/auth/\*\*"**).permitAll()  
 .antMatchers(HttpMethod.***GET***,  
 **"/api/v1/article/{id}"**,  
 **"/api/v1/article/view"**,  
 **"/api/v1/article/stats"**,  
 **"/api/v1/interact"**,  
 **"/api/v1/profile"**,  
 **"/api/v1/tag/most"**,  
 **"/api/v1/comment/list"**,  
 **"/api/v1/profile/{id}"**,  
 **"/api/v1/author/top"**).permitAll()  
 .antMatchers(HttpMethod.***POST***,  
 **"/api/v1/article/list"**,  
 **"/api/v1/article/related"**,  
 **"/api/v1/article/same-author"**,  
 **"/api/v1/article/trending"**).permitAll().anyRequest().authenticated();  
  
 *// Custom JWT based security filter* http.addFilterBefore(authenticationTokenFilterBean(), UsernamePasswordAuthenticationFilter.**class**);  
  
 *// disable page caching* http.headers().cacheControl();  
 }

Cuối cùng là hàm configure sẽ tiến hành cấp quyền API cho người dùng có thể dử dụng phương thức nào truy cập vào API nào và xử lý việc truy cập trái phép vào các API thông và việc trả về lỗi không có Token để truy cập.

## 5.3. Cài đặt ứng dụng phía Front-end

### 5.3.1. Cài đặt Yarn quản lý các thư viện node package

Yarn là công cụ quản lý thư viện javascript mã nguồn mở tốc độ cao, tin cậy và bảo mật nhằm thay thế NPM (Node Package Management). Được hình thành bởi các kỹ sư Facebook, Google, Exponent và Tilde, và đã được sử dụng thực tế mức Production tại các công ty trên. Đây là dự án bắt đầu được viết từ tháng 1/2016, sau quãng thời gian thử nghiệm và hoàn thiện đã được mở ra thành mã nguồn mở, và thật không có gì ngạc nhiên khi với những tính năng nổi trội vượt bậc đã có 10.000 stars chỉ trong 1 ngày tại Github. Chứng tỏ sự thành công và là tín hiệu thay thế rõ ràng NPM. Với cơ chề cài đặt các package song song trong một đường ống tạo về tốc độ cài đặt thì Yarn được nhóm chọn sử dụng trong dự án này cho việc cài đặt các thư viện vào Front-end.



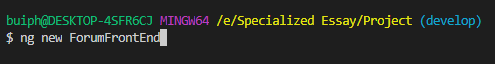
**Hình 5.8:** Cài đặt Yarn cho quản lý thư viện Front-end

Yarn có phiên bản cài đặt nên chỉ cần lên trang chủ tài về và cài đặt. Sau đó kiểm tra yarn –version để kiểm tra cài đặt thành công hay không.



**Hình 5.9:** Kiểm tra phiên bản của Yarn

### 5.3.2. Khởi tạo ứng dụng Angular 6

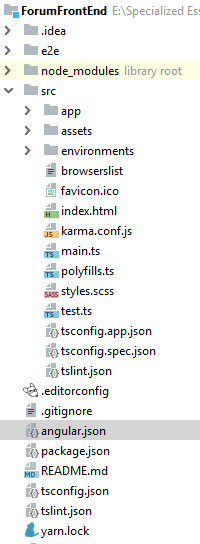


**Hình 5.10:** Khởi tạo dự án Angular 6

Cài đặt Angular/CLI phiên bản 6.1.0

Chạy lệnh “ng new ForumFrontEnd” cho việc khởi tạo ứng dụng

Sau khi chạy ta được một project Front-end như hình:



**Hình 5.11:** Cấu trúc thư mục Front-end của dự án

### 5.3.4. Thêm các thư viện cần thiết cho ứng dụng trong package.json



**Hình 5.12:** Các thư viện cho ứng dụng cài đặt trong package.json

Trong mục scripts sẽ là các lệnh để chạy ứng dụng như start, build hoặc test. Hoặc có thể sử dụng Yarn để start bằng lệnh yarn run start.

Trong dependencies sẽ chứa các package khi dự án được deploy sau này vì vậy các thư viện nào cần cho quá trình chạy production thì phải đặt vào đây.

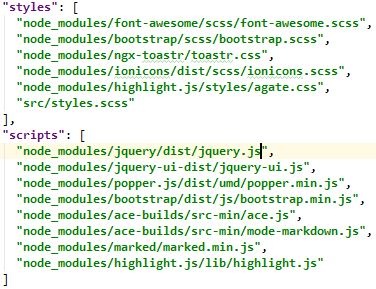
Cuối cùng là devDependencies sẽ là nơi chứa các thư viện liên quan trong qua trình phát triển và kiểm thử vì vậy khi deploy hoặc phát hành không nhất thiết phải cần các thư viện này.



**Hình 5.13:** Các thư viện trong mục devDependencies

### 5.3.5. Cấu hình preload cho angular.json

Angular hỗ trợ việc lazy load nhưng đối với các thư viện module nào cũng cần thiết thì chúng ta cần import nó vào tập tin angular.json để khi khởi chạy ứng dụng nó sẽ tải đầu tiên cho toàn ứng dụng. Ở đây là nhóm import các thư viện css và js cho các hiệu ứng trong ứng dụng.

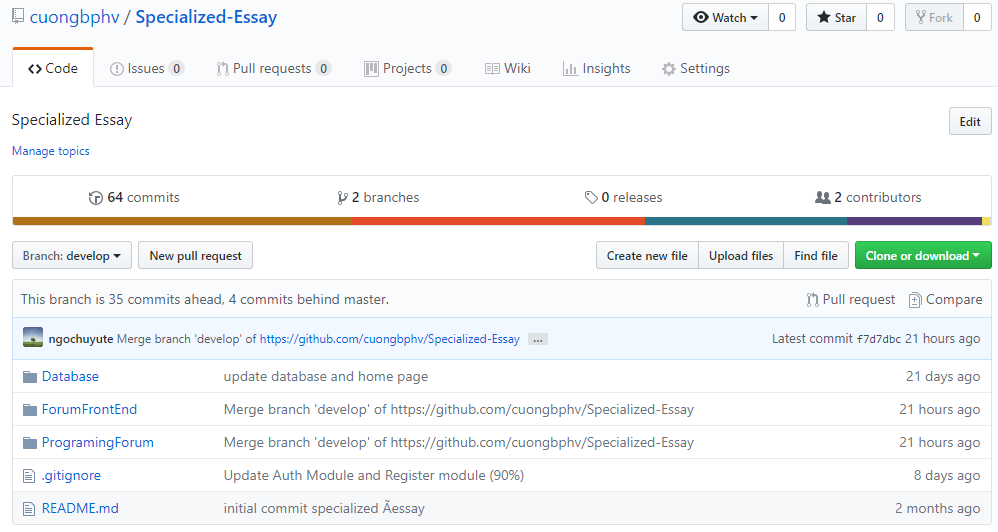


**Hình 5.14:** Thêm các thư viện preload cho ứng dụng trong angular.json

# QUẢN LÝ MÃ NGUỒN VÀ PHIÊN BẢN

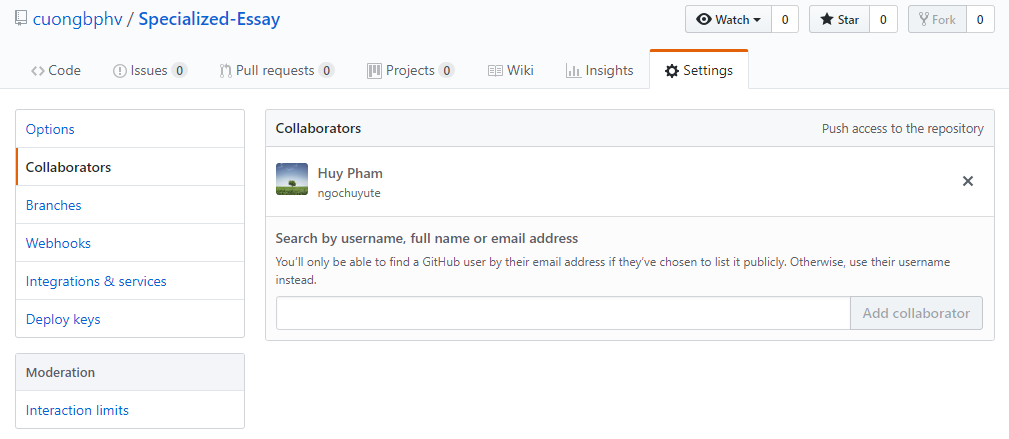
## 6.1. Tạo dự án tại Github

Đặt tên là Specialized Essay và tiến hành cấu hình README và thêm các thành viên trong nhóm vào project.



**Hình 6.1:** Tạo dự án trên Github

Thêm thành viên tham gia vào dự án:

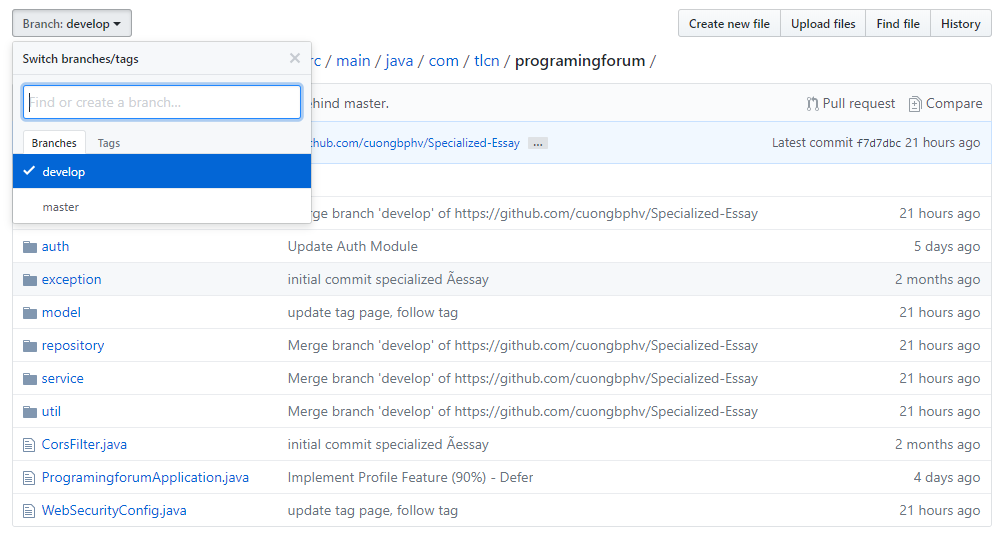


**Hình 6.2:** Thêm các thành viên vào dự án

## 6.2. Khởi tạo dự án

- Clone dự án về và thêm các thư mục cần thiết như database, backend, front-end và file .gitignore để tránh các thư viện và các cấu hình local trên mỗi máy thành viên.

- Tạo một branch develop để làm việc trên nhánh develop

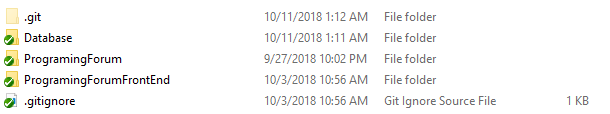


**Hình 6.3:** Tạo branch develop cho nhóm phát triển

## 6.3. Làm việc với dự án

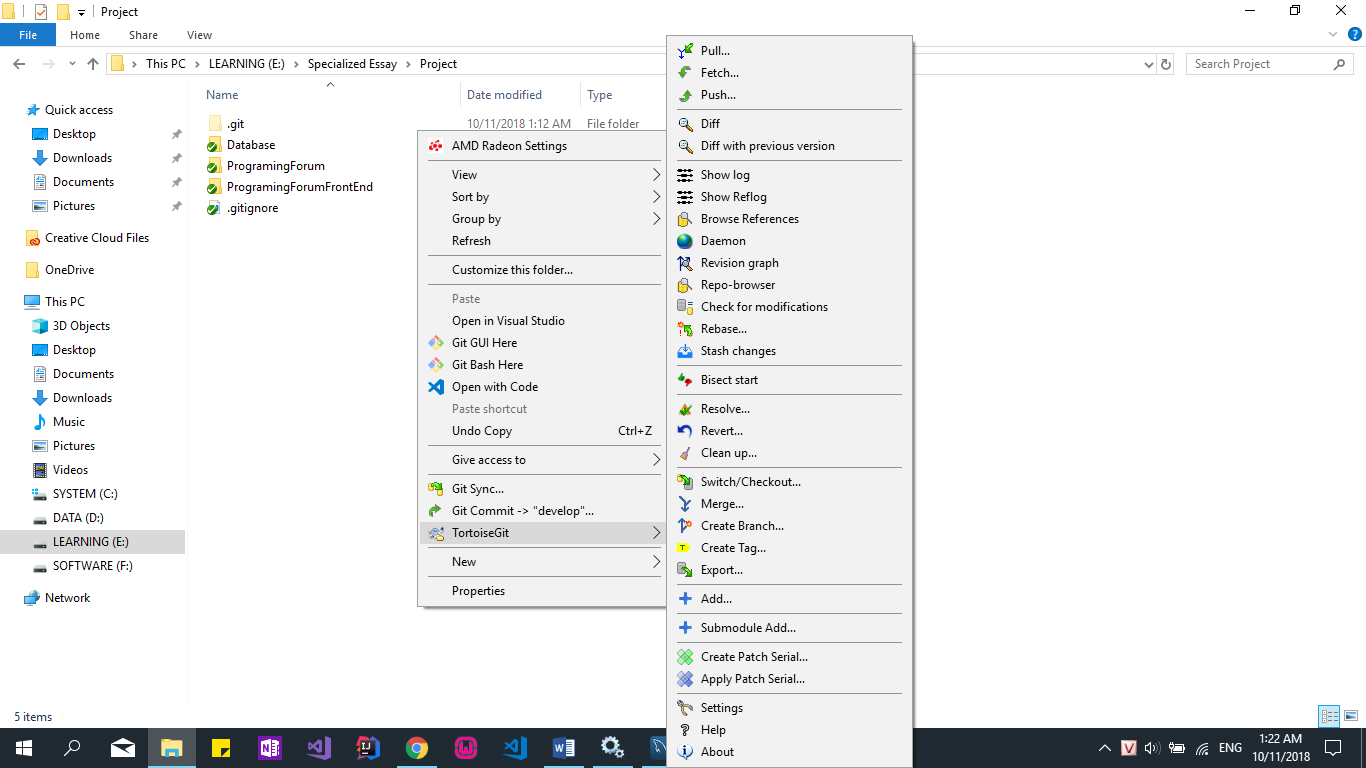
Các thành viên sẽ làm việc trên nhánh develop sau mỗi mỗi chức năng hoàn thành trưởng nhóm sẽ merge lại với branch master để kịp tiến độ dự án

Commit, pull và push các thư mục, tập tin mới lên gitlab.



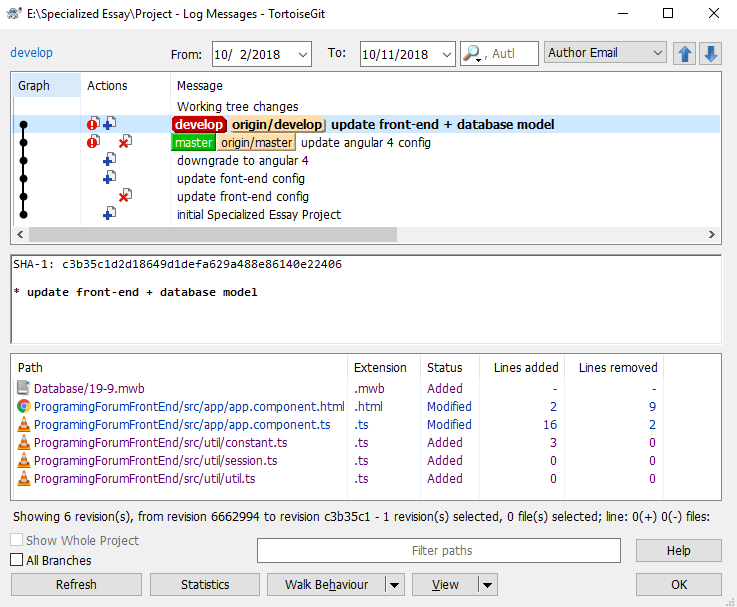
**Hình 6.4:** Clone ứng dụng về máy làm việc cá nhân

Sử dụng phần mềm TorToiseGit để quản lý mã nguồn trong dự án trên giao diện một cách thuận tiện hơn



**Hình 6.5:** Thao tác với dựn án qua phần mềm ToitoseGit

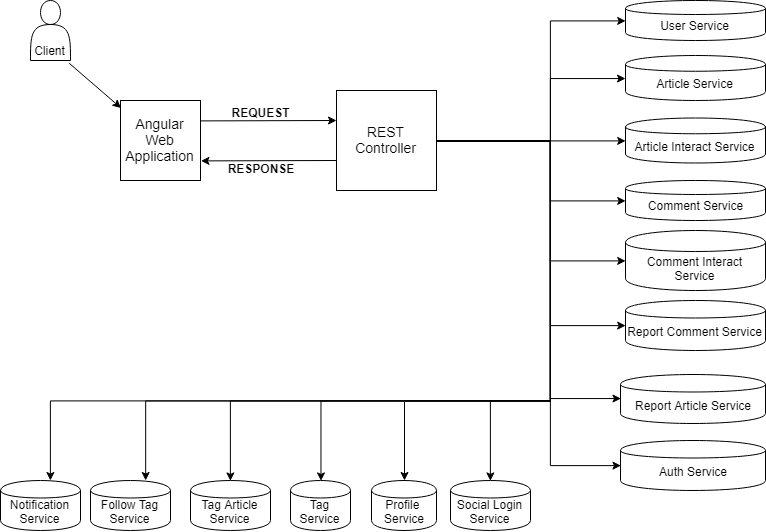
Xem lại lịch sử commit và các lần tác động đến dự án để phòng trường hợp xảy ta lỗi với git log



**Hình 6.6:** Push code lên branch develop sau mỗi thay đổi

# XÂY DỰNG RESTFUL API CHO ỨNG DỤNG

## 7.1. Mô hình Service trong ứng dụng

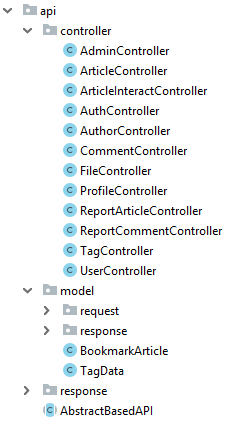


**Hình 7.1:** Mô hình hoạt động của ứng dụng và các Service

Trong mô hình trên khi người dùng làm việc với các chức năng của diễn đàn như đăng nhập, đăng bài viết, bình luận, đánh giá bài viết,... thì các tương tác ấy người dùng đang làm việc trên Front-end là Angular. Angular sẽ có các lệnh gọi đến các API được định nghĩa trong ứng dụng ở đây sẽ là /api/v1/\* , khi có lệnh gọi đến các API với các phương thức cụ thể như GET, POST, PUT, DELETE trong REST Controller và các API này hợp lệ, quyền người dùng đúng thì Controller sẽ tiến hành gọi đến các Service của ứng dụng mà ở đây tùy vào thao tác và API được gọi mà Controller gọi đến Service tương ứng.

Sau khi đã có các Service thì các dữ liệu người dùng yêu cầu lên có thể lưu trữ vào cơ sở dữ liệu hoặc yêu cầu lấy dữ liệu thì các Service sẽ tiến hành query dữ liệu và trả về cho REST Controller và từ Controller trả về Response cho Angular. Và ứng dụng Angular sẽ hiển thị các dữ liệu mà người dùng mong muốn lên của sổ trình duyệt từ đó tạo ra một luồng làm việc đồng nhất trong một ứng dụng RESTful API.

## 7.2. Cấu trúc REST Controller trong ứng dụng



**Hình 7.2:** Cấu trúc REST Controller của ứng dụng

Tổng quan ứng dụng sẽ quản lý một API trong Controller của Spring Boot, khi có yêu cầu từ người dùng gọi tới @RequestMapping trong Controller thì sẽ thực thi logic code bên trong API đó. Bên cạnh đó còn có các đối tượng cho việc Request và Response vì trong nhiều trường hợp API gửi lên là một đối tượng khi đó đòi hỏi API cần một @RequestBody cho các phương thức POST, PUT. Khi đó đối tượng trong Request sẽ được sử dụng đến, tương tự cho đối tượng Response khi API cần trả về các thông tin khác với Entity và các thông tin liên quan khi đó các đối tượng Response được sử dụng thay thế cho các Entity trả về.

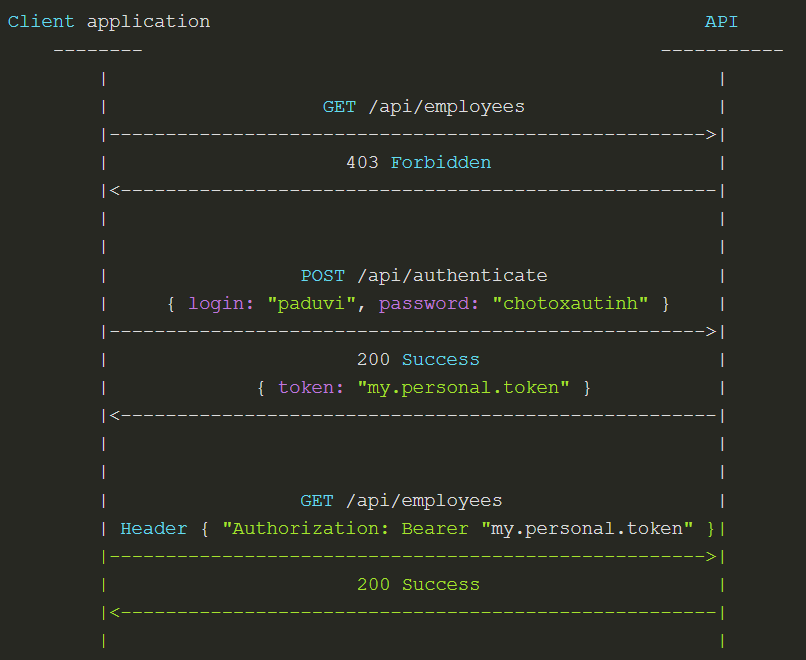
Tiếp theo không thể thiếu là một lớp AbstractBasedAPI cho việc kế thừa của tất cả các API trong lớp này sẽ xây dựng sẵn các hàm xác thực người dùng cùng với vai trò của người gửi yêu cầu đến API đó nhằm việc chặn nếu có truy cập không đúng ngay đầu API.

**public abstract class** AbstractBasedAPI {  
  
 **protected final** Logger **LOGGER** = LoggerFactory.*getLogger*(**this**.getClass());  
  
 @Autowired  
 **protected** ResponseUtil **responseUtil**;  
  
  
 @Autowired  
 **protected** CustomUserAuthService **customUserAuthService**;  
  
  
 **public void** validatePermission(AuthUser user, **int** role) {  
 **if** (user == **null**) {  
 **throw new** ApplicationException(APIStatus.***ERR\_ACCOUNT\_INVALID***);  
 }  
 **if** (user.getRole() > role) {  
 **throw new** ApplicationException(APIStatus.***ERR\_FORBIDDEN***);  
 }  
 }  
  
  
 **public** AuthUser getAuthUserFromSession(HttpServletRequest request) {  
 String authToken = request.getHeader(Constant.***HEADER\_TOKEN***);  
 *// try to load session* AuthUser user = **customUserAuthService**.loadUserByAccessToken(authToken);  
 **return** user;  
 }  
}

Lớp API cha ảo này còn có một đối tượng ResponseUtil hỗ trợ trong việc API trả về trong đó chứ status và data trả về nếu thành công.

@Component  
**public class** ResponseUtil {  
  
 **private** RestAPIResponse \_createResponse(APIStatus apiStatus, Object data) {  
 **return new** RestAPIResponse(apiStatus, data);  
 }  
  
 *// base method* **public** ResponseEntity<RestAPIResponse> buildResponse(APIStatus apiStatus, Object data, HttpStatus httpStatus) {  
 **return new** ResponseEntity(\_createResponse(apiStatus, data), httpStatus);  
 }  
  
 **public** ResponseEntity<RestAPIResponse> successResponse(Object data) {  
 **return** buildResponse(APIStatus.***OK***, data, HttpStatus.***OK***);  
 }  
  
 **public** ResponseEntity<RestAPIResponse> badRequestResponse(List<Constant.ParamError> errors) {  
  
 Map<String, String> errMap = **null**;  
  
 **if** (errors != **null**) {  
  
 errMap = **new** HashMap<>();  
 **for** (Constant.ParamError error : errors) {  
 errMap.put(error.getName(), error.getDesc());  
 }  
 }  
  
 **return** buildResponse(APIStatus.***ERR\_BAD\_REQUEST***, errMap, HttpStatus.***BAD\_REQUEST***);  
 }  
}

## 7.2. Xác thực trong ứng dụng



**Hình 7.3:** Mô tả hoạt động API xác thực người dùng

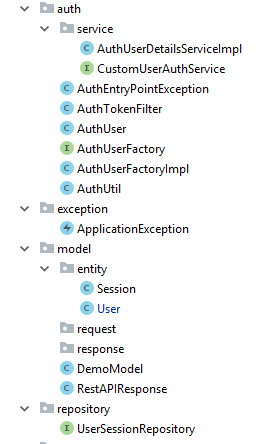
Ứng dụng xác thực người dùng theo mô hình Auth0 sử dụng Access Token. Nguyên tắc hoạt động của mô hình này là khi người dùng muốn truy cập tài nguyên từ hệ thống, người dùng gửi Tên tài khoản và mật khẩu lên Server để xác thực, sau đó Server sẽ xác thực và nếu đúng sẽ trả về một chuỗi ký tự gọi là Access Token cho người dùng. Người dùng sẽ lấy chuỗi Access Token này cho vào Header của Request và có thể truy cập tài nguyên của hệ thống. Nếu người dùng gửi Request không có Access Token hoặc Access Token sai thì hệ thống sẽ trả về lỗi không xác thực. Sau đây là các đặc điểm của mô hình này:

- Không Session: Với mô hình này bạn không cần phải giữ các session data trên server để xác thực người dùng. Các công việc cần làm sẽ chỉ như nguyên tắc hoạt động của nó ở trên.

- Không cần xử lý các session rác: Thông thường, các sessions sẽ có thời hạn hết hạn và cần phải được xử lý kiểu xoá đi các sessions "rác". Token hoàn toàn có thể sở hữu chính expiry date của chính nó kèm với dữ liệu người dùng. Cho nên khi tầng Security xác thực Token, nó có thể check expiry time của token và đơn giản là từ chối truy cập.

- RESTful services đúng nghĩa: Chỉ khi không sử dụng session thì mới có thể tạo nên một service thuần RESTful, bởi vì một service thuần RESTful được định nghĩa là phải stateless.

Các thành phần của module được định nghĩa trong ứng dụng:



**Hình 7.3**: Các thành phần module Xác thực trong ứng dụng

- AuthUser: thực thi giao diện UserDetails của Spring. Lớp này định nghĩa những thông tin của người dùng đang sử dụng Access Token.

- AuthUserDetailsServiceImpl: kế thừa từ lớp UserDetailsService của Spring. Lớp này có các phương thức dùng để truy cập thông tin của người dùng (AuthUser) đang sử dụng Access Token.

- AuthUserFactory và AuthUserFactoryImpl: Giao diện và lớp này gồm nhiều phương thức có chức năng dùng để tạo ra AuthUser khi người dùng thực hiện đăng nhập

- AuthTokenFilter: kế thừa từ lớp OncePerRequestFilter của Spring. Lớp này có chức năng thực hiện lọc (filter) cho mỗi request từ client. Nếu Access Token đúng thì mới cho phép truy cập tài nguyên từ hệ thống.

- AuthEntryPointException: thực thi giao diện AuthenticationEntryPoint của Spring. Nó được sử dụng để kiểm tra username/password đính kèm theo request có hợp lệ hay không. Nếu không thì sẽ quăng Exception và trả về thông tin lỗi cho client.

- AuthUtil: Lớp này bao gồm những phương thức tiện ích hỗ trợ thêm cho việc thực thi các lớp ở trên.

# KIỂM THỬ ỨNG DỤNG

## 8.1 Kế hoạch kiểm thử chung

Dựa vào kiến thức đã học trong môn kiểm thử phần mềm nhóm đã lập kế hoạch kiểm thử chia làm các kỹ thuật kiểm thử sau:

- Kiểm thử các chức năng theo từng module của dự án (Functional Testing)

- Kiểm thử phi chức năng trong dự án (Non-functional Testing)

- Kiểm thử hiệu suất của dự án (Performance Testing)

- Kiểm thử bảo mật và kiểm soát truy cập (Security & Access Control Testing)

- Kiểm thử đơn vị chia nhỏ module kiểm thử (Unit Testing)

Trong mỗi kỹ thuật kiểm thử nhóm sẽ chia theo:

- Mục tiêu tiếp cận

- Các module nào sẽ kiểm thử

- Môi trường kiểm thử

- Tiêu chí thành công

- Tiêu chí thất bại

## 8.2. Chi tiết testcase đã kiểm thử

### 8.2.1. Kiểm thử vai trò người dùng

**Bảng 8.1:** Bảng testcase kiểm thử vai trò người dùng

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Mô tả test case** | **Quy trình** | **Kết quả mong đợi** | **Người test** | **Chrome 71.0.3578 (64bit)** | **Firefox 64.0b9 (64bit)** |
| 1 | Kiểm thử chức năng đăng nhập | 1. Nhấn vào đường dẫn đăng nhập trên thanh điều hướng  2. Nhập tài khoản và mật khẩu  3. Nhấn nút đăng nhập | 3. Ứng dụng chuyển trở về trang chủ với tên người dùng hiển thị trên thanh điều hướng | Cường | Pass | Pass |
| 2 | Kiểm thử chức năng đăng ký | 1. Ở cửa sổ đăng nhập nhấn vào liên kết “Đăng ký”  2. Nhập họ, tên, email, username đã trùng  3. Nhấn nút Đăng ký | 2. Báo lỗi email, username đã tồn tại | Cường | Pass | Pass |
| 3 | Kiểm thử chức năng đăng ký | 1. Ở cửa sổ đăng nhập nhấn vào liên kết “Đăng ký”  2. Nhập họ, tên, email, username đúng và không bị trùng  3. Nhấn nút Đăng ký | 3. Đăng ký thành công chuyển qua trang điền thông tin | Cường | Pass | Pass |
| 4 | Kiểm thử chức năng tạo bài viết | 1. Ở trang chủ nhấn nút “Viết bài”  2. Ở trang viết bài điền tiêu đề và nội dung với định dạng markdown  3. Chọn loại bài viết là bài viết  4. Gắn thẻ  5. Nhấn nút Đăng bài | 5. Tạo bài viết với loại bài đăng thành công | Cường | Pass | Pass |
| 5 | Kiểm thử chức năng bình luận | 1. Trong một bài viết bất kỳ, kéo xuống phần bình luận  2. Gõ nội dung bình luận mới  3. Nhấn nút “Bình luận” | 3. Bình luận mới được thêm thành công | Cường | Pass | Pass |
| 6 | Kiểm thử chức năng sửa bài viết | 1. Vào bài viết là tài khoản đăng nhập đã đăng  2. Chọn nút “Chỉnh sửa” phía sau tiêu đề  3. Chuyển đến trang chỉnh sửa bài viết  4. Chỉnh sửa thông tin  5. Nhấn nút “Cập nhật” | 5. Cập nhật bài viết thành công | Cường | Pass | Pass |
| 7 | Kiểm thử chức năng tương tác | 1. Ở cửa sổ bài viết nhấn nút “Upvote”  2. Nhấn nút “downvote” | 1. Số upvote tăng lên 1 và thông báo đánh giá thành công  2. Số đánh giá giảm xuống 1 và thông báo downvote thành công | Cường | Pass | Pass |
| 8 | Kiểm thử chức năng lưu trữ bài viết | 1. Ở của sổ bài viết nhấn nút bookmark  2. Trở về trang chủ và chọn tab “Bookmark” | 1. Thông báo lưu trữ thành công  2. Trong cửa sổ Bookmark có xuất hiện bài viết đã lưu | Cường | Pass | Pass |
| 9 | Kiểm thử chức năng báo cáo bài viết | 1. Ở cửa sổ bài viết nhấn vào nút báo cáo  2. Ở modal box nhập lý do báo cáo  3. Nhấn nút báo cáo | 1. Cửa sổ báo cáo modal box hiện ra  2. Thông báo gửi báo cáo thành công | Cường | Pass | Pass |

### 8.2.2. Kiểm thử vai trò Admin

**Bảng 8.2:** Bảng testcase kiểm thử vai trò Admin

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Mô tả test case** | **Quy trình** | **Kết quả mong đợi** | **Người test** | **Chrome 71.0.3578 (64bit)** | **Firefox 64.0b9 (64bit)** |
| 1 | Kiểm thử chức năng đăng nhập | 1. Nhấn vào đường dẫn đăng nhập trên thanh điều hướng  2. Nhập tài khoản và mật khẩu  3. Nhấn nút đăng nhập  4. Trên thanh điều hướng chọn truy cập Dashboard | 3. Ứng dụng chuyển trở về trang chủ với tên người dùng hiển thị trên thanh điều hướng  4. Truy cập Dashboard với role admin thành công | Huy | Pass | Pass |
| 2 | Kiểm thử chức năng duyệt bài | 1. Ở cửa sổ quản lý bài viết, chọn 1 bài viết mới  2. Chấp nhận bài viết mới đó | 2. Trạng thái bài viết công khai và xuất hiện trên trang chủ | Huy | Pass | Pass |
| 3 | Kiểm thử chức năng chặn bài viết | 1. Ở cửa sổ quản lý bài viết, chọn 1 bài viết mới  2. Chọn nút Chặn bài đăng | 2. Bài đăng thay đổi trạng thái không chấp nhận và không xuất hiện ở trang chủ | Huy | Pass | Pass |
| 4 | Kiểm thử chức năng chặn người dùng vi phạm | 1. Ở của sổ quản lý người dùng  2. Chọn một người dùng đang hoạt động  3. Chọn chức năng chặn người dùng | 3. Người dùng bị xóa khỏi danh sách thành viên hoạt động | Huy | Pass | Pass |
| 5 | Kiểm thử chức năng gán quyền cho người dùng | 1. Ở của sổ quản lý người dùng  2. Chọn một người dùng đang hoạt động  3. Gán quyền Moderator cho người dùng | 3. Người gán quyền có thể vào Dashboar với role Moderator và làm các chức năng của Moderator | Huy | Pass | Pass |

### 8.2.3. Kiểm thử vai trò Moderator

**Bảng 8.3:** Bảng testcase kiểm thử vai trò Moderator

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Mô tả test case** | **Quy trình** | **Kết quả mong đợi** | **Người test** | **Chrome 71.0.3578 (64bit)** | **Firefox 64.0b9 (64bit)** |
| 1 | Kiểm thử chức năng đăng nhập | 1. Nhấn vào đường dẫn đăng nhập trên thanh điều hướng  2. Nhập tài khoản và mật khẩu  3. Nhấn nút đăng nhập  4. Trên thanh điều hướng chọn truy cập Dashboard | 3. Ứng dụng chuyển trở về trang chủ với tên người dùng hiển thị trên thanh điều hướng  4. Truy cập Dashboard với role moderator thành công | Huy | Pass | Pass |
| 2 | Kiểm thử chức năng duyệt bài | 1. Ở cửa sổ quản lý bài viết, chọn 1 bài viết mới  2. Chấp nhận bài viết mới đó | 2. Trạng thái bài viết công khai và xuất hiện trên trang chủ | Huy | Pass | Pass |
| 3 | Kiểm thử chức năng chặn bài viết | 1. Ở cửa sổ quản lý bài viết, chọn 1 bài viết mới  2. Chọn nút Chặn bài đăng | 2. Bài đăng thay đổi trạng thái không chấp nhận và không xuất hiện ở trang chủ | Huy | Pass | Pass |

# KẾT LUẬN

## 1. Kết quả đạt được

Đầu tiên kết quả đạt được nhóm đã hoàn thành dự án theo thời gian đã định ra của môn tiểu luận chuyên ngành và sau đó là các kiến thức không chỉ lá lý thuyết mà còn kiến thức thực tiễn áp dụng cho sau này. Đó là các kiến thức về Web Service, RESTful API, Spring Boot với RESTful API. Bên cạnh đó là các kiến thức về thiết kế giao diện, lập trình front-end và quản lý dự án với Git qua đó làm việc và giao tiếp với nhóm trở nên tốt hơn.

Thông qua đề tài tiểu luận chuyên ngành nhóm cũng đã đúc rút ra các bài học cũng như những kinh nghiệm để áp dụng cho khóa luận tốt nghiệp vào kỳ sau.

Lời cuối cùng nhóm xin chân thành cảm ơn thầy Minh Đạo đã giúp nhóm trong môn học Tiểu luận chuyên ngành giúp nhóm có nhiều cái nhìn, kiến thức và hướng đi trong việc xây dựng đề tài và làm việc với dự án đó. Nhóm chân thành cảm ơn thầy!

## 2. Kinh nghiệm rút ra từ quá trình thực hiện đề tài

Qua đề tài “Tìm hiểu và xây dựng ứng dụng Spring RESTful”. Nhóm thực hiện báo cáo đã hoàn thành đề tài kết hợp RestFul API với front-end sử dụng Angular 6 và Bootstrap. Sau đề tài này, nhóm thực hiện báo cáo rút ra được một số kinh nghiệm như sau:

- Khi chọn đề tài phải cân nhắc về độ lớn, nội dung, khả năng thực hiện và ước chừng thời gian gặp rủi ro trong quá trình thực hiện.

- Tìm hiểu công nghệ nào phù hợp với dự án của nhóm sau đó tìm kiếm một tài liệu tin cậy và theo sát (thường thì nên sử dụng tài liệu chính thức của công nghệ đó)

- Phân bố hợp lý thời gian trong quá trình tìm hiểu và thực thi dự án, có phối hợp giữa mặt tìm hiểu và thực thi.

- Nên chia phần tìm hiểu công nghệ cho các thành viên trong nhóm, sau đó các thành viên chia sẻ những kiến thức tìm hiểu được với nhau. Điều này sẽ tiết kiệm được thời gian cũng như có khả năng đồng bộ logic của dự án giúp các thành viên tương tác tốt với nhau hơn trong khi thực hiện cũng như bảo trì dự án.

- Trước khi đi vào thực thi dự án, nên thực hiện khảo sát các sản phẩm tương tự, chọn ra các điểm mạnh để phát huy và điểm yếu để phòng trừ. Cũng như từ đó có thể làm ra sự khác biệt của dự án.

- Lập kế hoạch chi tiết cho dự án và luôn bám sát theo kế hoạch. Không được chủ quan cũng như để mọi thứ đi quá xa so với kế hoạch đề ra ban đầu.

- Luôn thực hiện test các chức năng sau khi hoàn thành, tránh việc làm xong cả dự án rồi mới test một lần. Bởi vì sau mỗi chức năng, nếu chúng ta không test kỹ thì có thể gây ra các vấn đề đối với rất nhiều chức năng ở sau đó.

- Các thành viên phải có trách nhiệm với đề tài. Phân chia công việc hợp lý giữa các thành viên để tránh gây tranh cãi, xung đột nội bộ.

**3. Hướng phát triển cho đề tài**

Nhóm sẽ phát triển thêm chức năng Group cho diễn đàn nhưng sẽ cân nhắc vì diễn đàn mang khuynh hướng mở chia sẻ kiến thức mở nhưng thành lập nhóm có thể khó khăn trong việc quản lý thành viên và sự chia sẻ rộng rãi kiến thức.

Chức năng thành viên nhận thông báo về bài viết mới qua email hoặc qua chat bot sẽ giúp cho thành viên cập nhật tin tức hàng ngày nếu không có đủ thời gian trực tuyến.

Chức năng tạo các kênh video cho phép người dùng trả phí để học như một số kênh khác như Udemy, Treehouse,... như vậy diễn đàn vừa là nơi chia sẻ vừa là nơi thành viên có thể tạo nguồn thu nhập thụ động cho bản thân.

# TÀI LIỆU THAM KHẢO

1. Lý thuyết Web Service: <https://viblo.asia/p/tim-hieu-ban-dau-ve-web-service-jdWrvwQ5Mw38>

2. Lý thuyết RESTful API: <https://o7planning.org/vi/10773/restful-web-service-la-gi>

3. Ngôn ngữ HQL: <http://viettuts.vn/hibernate/hibernate-query-language>

4. <https://developer.mozilla.org/vi/docs/Web/HTTP/Access_control_CORS> , Khái niệm CORS

2. <https://text.relipasoft.com/2017/05/co-ban-ve-rest-va-restful-api/> , Tổng quan về REST và RESTful API

4. <https://viblo.asia/p/thiet-ke-restful-api-GrLZD98Vlk0> , Thiết kế RESTful API chuẩn

5. <https://idratherbewriting.com/learnapidoc/docapis_what_is_a_rest_api.html> , Hình ảnh sử dụng trong phần Các thành phần trong RESTful API

6. <http://www.codingpedia.org/ama/tutorial-rest-api-design-and-implementation-in-java-with-jersey-and-spring/> , Hình ảnh sử dụng trong phần RESTful API trong mô hình 3 lớp Spring Boot