

BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCMS



Đồ Án Công Nghệ Thông Tin  
Đề tài: Xây dựng trang web đọc báo điện tử

Trần Quốc Siêu - 20110558

*Thành phố Hồ Chí Minh, Ngày 01 Tháng 10 năm 2023*



# Mục Lục

<b>Mục Lục.....</b>	<b>3</b>
<b>Danh mục hình ảnh .....</b>	<b>5</b>
<b>Danh mục bảng .....</b>	<b>7</b>
<b>Lời mở đầu .....</b>	<b>8</b>
<b>I. Đặc tả đề tài: .....</b>	<b>9</b>
1. Úng dụng của phần mềm .....	9
2. Dữ liệu, thông tin đầu vào .....	9
3. Tình huống sử dụng .....	9
4. Giao diện dự kiến:.....	9
<b>II. Thiết kế .....</b>	<b>11</b>
1. Thiết kế cơ sở dữ liệu.....	11
a. Diagram .....	11
b. Mô tả cơ sở dữ liệu.....	11
2. Mô tả thiết kế.....	15
a. Giới thiệu chung.....	15
b. Apache Tomcat .....	16
c. Java Servlet .....	16
d. Mô hình MVC .....	17
e. Maven.....	17
3. Xây dựng thuật toán .....	17
a. Beans .....	17
b. Controller .....	20
c. Filters .....	37
d. Services .....	38
e. Utils .....	45
f. Views .....	48
4. Giao diện ứng dụng.....	50
a. Trang Home .....	50
b. Trang đăng nhập và đăng ký.....	53

c.	<b>Trang của quản trị viên</b> .....	54
d.	<b>Trang của biên tập viên</b> .....	56
e.	<b>Trang của người kiểm duyệt</b> .....	58
<b>III.</b>	<b>Cài đặt và kiểm thử</b> .....	<b>59</b>
1.	<b>Cài đặt</b> .....	<b>59</b>
1.1.	Tải các tài nguyên cần thiết .....	59
1.2.	Cấu hình để deploy trang web .....	59
2.	<b>Kiểm thử</b> .....	<b>63</b>
<b>IV.</b>	<b>Danh sách các tài khoản sử dụng</b> .....	<b>71</b>
1.	<b>Tài khoản Premium</b> .....	71
2.	<b>Tài khoản admin</b> .....	71
3.	<b>Tài khoản Người kiểm duyệt</b> .....	71
4.	<b>Tài khoản biên tập viên</b> .....	71

## Danh mục hình ảnh

Hình 1 Giao diện dự kiến .....	10
Hình 2 Diagram cơ sở dữ liệu web đọc báo điện tử .....	11
Hình 3 PostgreSQL.....	12
Hình 4 Các beans của trang web .....	18
Hình 5 Các thuộc tính trong bean Article.....	18
Hình 6 Các hàm tạo trong bean Article .....	19
Hình 7 Các phương thức getter và setter trong bean Article bean .....	20
Hình 8 Các Controller có trong web.....	20
Hình 9 AccoutServlet Xử lý yêu cầu GET .....	22
Hình 10 AccountServlet Xử lý yêu cầu POST .....	23
Hình 11 AccounServlet xử lý một số chức năng.....	24
Hình 12 AdminServlet Xử lý yêu cầu GET .....	25
Hình 13 AdminServlet Xử lý yêu cầu POST .....	26
Hình 14 AdminServlet xử lý các chức năng.....	27
Hình 15 Xử lý các chức năng câu DefaultServlet .....	28
Hình 16 EditorServlet xử lý yêu cầu GET .....	29
Hình 17 EditorServlet xử lý yêu cầu POST .....	29
Hình 18 EditorServlet xử lý chức năng cập nhật trạng thái .....	30
Hình 19 HomeServlet Xử lý yêu cầu GET (tóm tắt) .....	31
Hình 20 HomeServlet Xử lý yêu cầu POST .....	32
Hình 21 HomeServlet xử lý một số chức năng .....	32
Hình 22 StaffServlet Xử lý yêu cầu GET.....	33
Hình 23 StaffServlet Xử lý yêu cầu POST .....	34
Hình 24 WriterServlet Xử lý yêu cầu GET (tóm tắt) .....	35
Hình 25 WriterServlet xử lý yêu cầu POST .....	36
Hình 26 WriterServlet xử lý một số chức năng.....	36
Hình 27 Các loại filter cho mỗi yêu cầu.....	37
Hình 28 Chức năng của LayoutFilter .....	37
Hình 29 Chức năng của SessionInitFilter.....	38
Hình 30 Tất cả Service được sử dụng trong trang web .....	38
Hình 31 Một số đoạn code để xử lý va lấy dữ liệu của ArticleService .....	39
Hình 32 Một số đoạn code để xử lý va lấy dữ liệu của CategoryService .....	40
Hình 33 Một số đoạn code để xử lý va lấy dữ liệu của CommentService .....	40
Hình 34 Một số đoạn code để xử lý va lấy dữ liệu của EmailService .....	41
Hình 35 Hàm Findall() trong ParentCategoryService .....	42
Hình 36 Mã nguồn của TagHasArticleService.....	42
Hình 37 Một số phương thức xử lý dữ liệu của TagService .....	43
Hình 38 Một số phương thức xử lý dữ liệu của UserService .....	45
Hình 39 Thư mục Utils .....	45

Hình 40 Kết nối đến cơ sở dữ liệu trong file DbUtils .....	46
Hình 41 ServletUtils dùng để xử lý các chuyển hướng và chuyển tiếp .....	47
Hình 42 Phương thức Verify để xử lý captcha của VerifyRecaptcha .....	48
Hình 43 Các file .jsp để xử lý giao diện của người dùng .....	49
Hình 44 Trang home.....	50
Hình 45 Chi tiết bài báo 1 (chưa đăng nhập).....	50
Hình 46 Chi tiết bài báo 2 (chưa đăng nhập).....	51
Hình 47 Trang home khi tìm kiếm từ khóa .....	51
Hình 48 Trang home với danh mục được chọn .....	52
Hình 49 Chi tiết bài báo 1 (đã đăng nhập).....	52
Hình 50 Chi tiết bài báo 2 (đã đăng nhập).....	53
Hình 51 Giao diện đăng nhập .....	53
Hình 52 Trang khôi phục mật khẩu .....	54
Hình 53 Trang tạo tài khoản .....	54
Hình 54 Trang quản lý người dùng .....	55
Hình 55 Trang quản lý bài viết.....	55
Hình 56 Trang quản lý danh mục .....	56
Hình 57 Trang quản lý tag .....	56
Hình 58 Trang quản lý bài viết cá nhân .....	57
Hình 59 Trang soạn thảo viết báo dành cho biên tập viên (Phần 1).....	57
Hình 60 Trang soạn thảo viết báo dành cho biên tập viên (Phần 2).....	58
Hình 61 Trang quản lý bài viết của người kiểm duyệt.....	58
Hình 62 Chính sửa cách chạy dự án trong IntelliJ .....	60
Hình 63 Chọn Tomcat Server làm host .....	60
Hình 64 Chọn Cách deploy cho Apache Tomcat .....	61
Hình 65 Chọn Artifact .....	61
Hình 66 Chính sửa lại Path trang web .....	62
Hình 67 Chính sửa kết nối cơ sở dữ liệu .....	62
Hình 68 Hiện ra log của Tomcat .....	63
Hình 69 Deploy thành công.....	63

## Danh mục bảng

<b>Bảng 1: Bảng mô tả các Table .....</b>	13
<b>Bảng 2 Bảng Articles .....</b>	13
<b>Bảng 3 Bảng Users .....</b>	14
<b>Bảng 4 Bảng Roles .....</b>	14
<b>Bảng 5 Bảng Comments .....</b>	14
<b>Bảng 6 Bảng Status .....</b>	14
<b>Bảng 7 Bảng Categories .....</b>	15
<b>Bảng 8 Bảng Parent_categories.....</b>	15
<b>Bảng 9 Bảng Tag_has_articles .....</b>	15
<b>Bảng 10 Bảng Tags .....</b>	15
<b>Bảng 11 Bảng Editor_manage_categories.....</b>	15
<b>Bảng 12 Testcase Đăng ký Tài khoản với Email không hợp lệ .....</b>	64
<b>Bảng 13 Testcase Đăng ký Tài khoản với Email đã tồn tại trong hệ thống .....</b>	65
<b>Bảng 14 Testcase đăng ký tài khoản với username đã tồn tại trong hệ thống .....</b>	67
<b>Bảng 15 Testcase nhập 2 trường password và confirmpassword không trùng nhau.....</b>	69

## **Lời mở đầu**

Tôi xin được bắt đầu bài trình bày với sự tôn trọng và sự hồi hộp trước tầm quan trọng của môn đồ án Công nghệ Thông tin, nơi tôi có cơ hội thực hành và áp dụng những kiến thức đã học vào thực tế. Đồ án của tôi hôm nay tập trung vào một đề tài mang tính ứng dụng cao, đó là "Xây dựng trang web đọc báo điện tử".

Trong thời đại ngày nay, công nghệ thông tin đã trở thành trực cốt của mọi hoạt động xã hội, và internet đóng vai trò quan trọng nhất trong việc truyền tải thông tin. Trang web đọc báo điện tử không chỉ là một nguồn thông tin nhanh chóng mà còn là một phương tiện tiện lợi, đa dạng, và linh hoạt để đáp ứng nhu cầu đọc tin tức của người sử dụng. Chính vì vậy, việc xây dựng một trang web đọc báo điện tử hiệu quả không chỉ đòi hỏi sự hiểu biết sâu sắc về công nghệ mà còn yêu cầu sự tinh tế trong thiết kế và trải nghiệm người dùng.

Trong đồ án này, tôi sẽ tìm hiểu và thực hiện quá trình phát triển một trang web đọc báo điện tử từ các khâu cơ bản như lập kế hoạch, thiết kế giao diện, quản lý cơ sở dữ liệu, đến việc triển khai và duy trì hệ thống. Đồng thời, tôi cũng sẽ đặc biệt chú trọng vào việc tối ưu hóa trang web để đảm bảo tính linh hoạt, hiệu suất và trải nghiệm người dùng tốt nhất.

Như vậy, tôi hy vọng rằng đồ án này không chỉ là cơ hội để tôi học hỏi và áp dụng kiến thức, mà còn là một đóng góp nhỏ trong việc nâng cao chất lượng và trải nghiệm người dùng trên không gian mạng.

## **I. Đặc tả đề tài:**

### **1. Ứng dụng của phần mềm**

Đọc báo là một công việc hằng ngày phổ biến của mọi người trên toàn thế giới, ai ai cũng muốn cập nhật những thông tin nóng từ khắp nơi trên thế giới. Nhu cầu đọc báo của người dùng ngày càng tăng vọt và Internet cũng dần phổ biến rộng rãi. Kết hợp cả hai điều này ứng dụng đọc báo điện tử trực tiếp trên trang web có thể phục vụ hầu hết các nhu cầu của người dùng cũng như mang lại sự tiện lợi, tiết kiệm chi phí và thời gian.

### **2. Dữ liệu, thông tin đầu vào**

Các bài báo sẽ được các biên tập viên chuyên soạn mỗi ngày trên trang web và được các quản trị viên đọc để kiểm tra chất lượng sau đó mới phát hành cho người đọc trên trang web. Cả quá trình này đều được thực hiện bởi người dùng tùy thuộc vào loại tài khoản mà họ sử dụng.

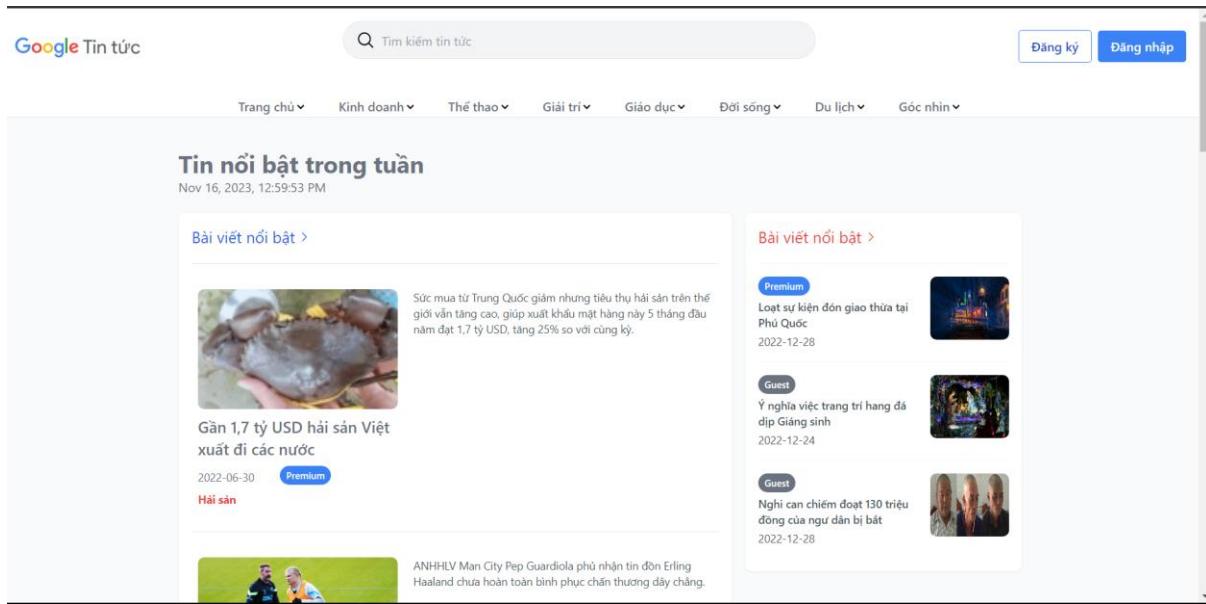
### **3. Tình huống sử dụng**

**Đối với người dùng thông thường:** trang web là nơi các độc giả cập nhật các thông tin mới nhất từ trong nước đến ngoài nước. Ngoài ra còn có các chuyên mục blog chia sẻ giữa các cá nhân với nhau về kinh nghiệm sống cũng như tạo dựng một cộng đồng.

**Đối với các tòa soạn:** Biên tập viên có thể viết các bài báo và tải lên trên trang web để mọi người có thể cập nhật thông tin mới nhất.

### **4. Giao diện dự kiến:**

Về phần giao diện trang web này sẽ tham khảo giao diện của GoogleNews một trang web đọc báo của Google



Hình 1 Giao diện dự kiến

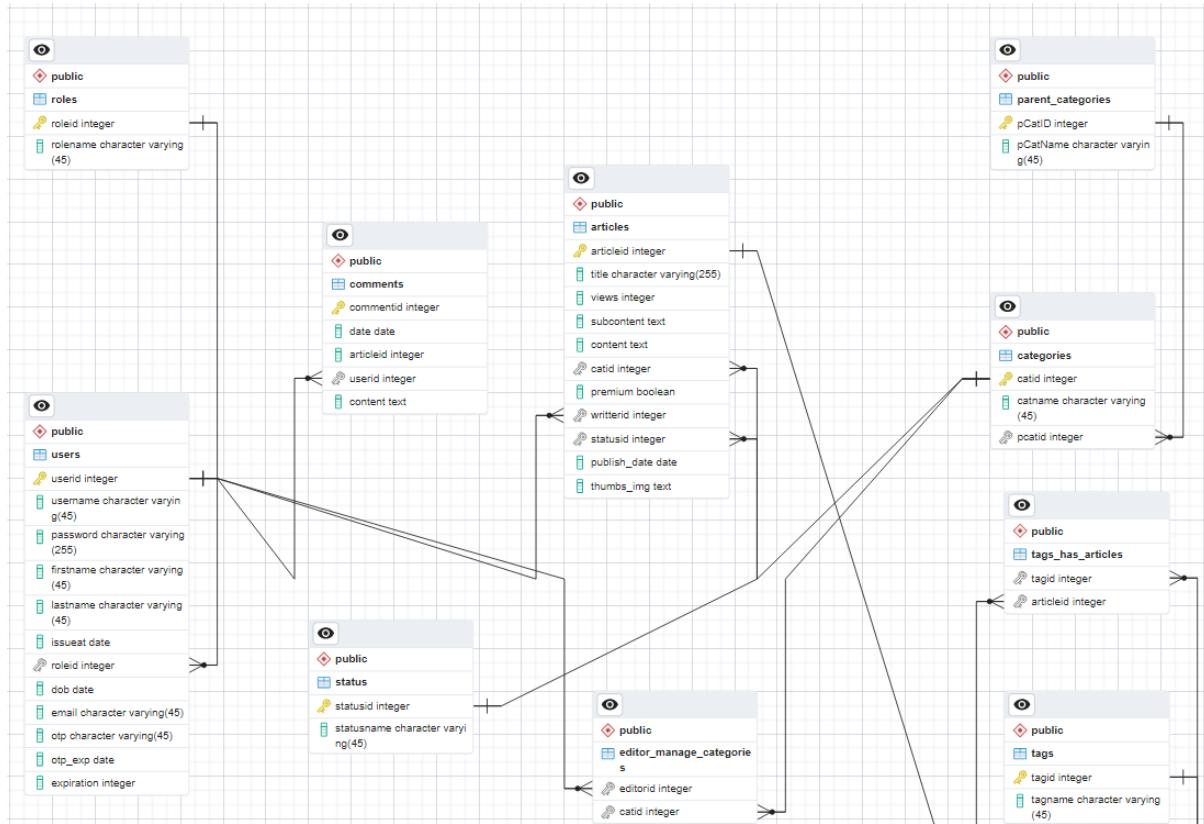
Ở góc bên trái sẽ là logo GoogleNews, ở giữa sẽ là thanh tìm kiếm, bên phải sẽ là 2 nút đăng nhập và đăng ký.

Đến thanh điều hướng sẽ chia thành các mục phổ biến trên báo.

## II. Thiết kế

### 1. Thiết kế cơ sở dữ liệu

#### a. Diagram



Hình 2 Diagram cơ sở dữ liệu web đọc báo điện tử

#### b. Mô tả cơ sở dữ liệu

Trang web đọc báo sử dụng hệ quản trị cơ sở dữ liệu PostgreSQL để quản lý.

PostgreSQL là một hệ quản trị cơ sở dữ liệu (DBMS) mã nguồn mở, mạnh mẽ, và có tính năng đầy đủ. Được phát triển từ hơn 15 năm trước, PostgreSQL đã trở thành một trong những hệ quản trị cơ sở dữ liệu phổ biến nhất trên thế giới. Nó hỗ trợ nhiều tính năng tiên tiến như giao thức ACID (Atomicity, Consistency, Isolation, Durability), đa người dùng, và mô hình đối tượng-quan hệ.



Hình 3 PostgreSQL

#### Đặc điểm nổi bật của PostgreSQL:

- **Độ ổn định và Tin cậy:** PostgreSQL nổi tiếng với độ ổn định và độ tin cậy cao, thậm chí trong các môi trường có khối lượng công việc lớn.
- **Hỗ trợ Đối tượng-Quan hệ:** PostgreSQL không chỉ là một hệ quản trị cơ sở dữ liệu quan hệ thông thường mà còn hỗ trợ các đối tượng phức tạp như hình học và âm thanh.
- **Mở rộng và Mô-đun:** PostgreSQL cho phép người dùng mở rộng và tùy chỉnh bằng cách thêm vào nó các mô-đun mở rộng.
- **Ngôn ngữ Hàm Truy vấn và Triggers:** PostgreSQL hỗ trợ một loạt các ngôn ngữ hàm truy vấn và triggers, tạo điều kiện cho việc xử lý sự kiện và logic phức tạp.

#### Dưới đây là bảng mô tả các Table trong CSDL

Chú thích:

- *Tên trường màu đỏ: Khóa chính*
- *Tên trường màu xanh: Khóa ngoại*

**Bảng 1: Bảng mô tả các Table**

STT	Tên bảng	Mục đích
1	Articles	Chứa thông tin của các bài báo
2	Users	Chứa thông tin của các người dùng
3	Roles	Bảng phân quyền các người dùng
4	Comments	Chứa các câu bình luận của người dùng
5	Status	Tình trạng của các bài báo
6	Categories	Danh mục, thể loại các bài báo
7	Parent_categories	Danh mục cha của các danh mục chính
8	Tags_has_articles	Các tags có trong các bài báo
9	Tags	Định nghĩa các tag
10	Editor_manage_categories	Chứa thông tin người chỉnh sửa các danh mục

**Bảng 2 Bảng Articles**

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	ArticleID	Int	Số ID của bài báo
2	Title	Varchar(45)	Tiêu đề bài báo
3	Views	Integer	Lượt xem của bài báo
4	Subcontent	Text	Tiêu điểm bài báo
5	Content	Text	Nội dung bài báo
6	CatID	Int	Số ID của danh mục
7	Premium	Boolean	Xác nhận bài báo có dành cho độc giả Premium hay không
8	WriterID	Int	Số ID của người viết bài báo
9	statusID	Int	Số ID tình trạng bài báo
10	Publish_date	Datetime	Ngày xuất bản bài báo
11	Thumbs_img	Text	Hình ảnh của bài báo

**Bảng 3** **Bảng Users**

STT	Tên Trường	Kiểu dữ liệu	Mục đích
1	<b>UserID</b>	Int	Số ID của người dùng
2	Username	Varchar(45)	Tên tài khoản
3	Password	Varchar(255)	Mật khẩu
4	Firstname	Varchar(45)	Họ
5	Lastname	Varchar(45)	Tên
6	Issueat	Date	Ngày tạo tài khoản
7	<b>RoleID</b>	Int	Số ID của vai trò người dùng
8	Dob	Date	Ngày sinh của người dùng
9	Email	Varchar(45)	Tài khoản Email người dùng
10	Otp	Varchar(45)	Mã OTP người dùng
11	Otp_exp	Date	Ngày hết hạn OTP
12	Expiration	Int	Thời gian hết hạn Premium

**Bảng 4** **Bảng Roles**

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	<b>RoleID</b>	Int	Số ID của vai trò
2	Rolename	Varchar(45)	Tên vai trò

**Bảng 5** **Bảng Comments**

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	<b>CommentID</b>	Int	Số ID của bình luận
2	Date	Date	Ngày bình luận
3	ArticleID	Int	Số ID bài báo đã bình luận
4	<b>UserID</b>	Int	Số ID của người dùng đã bình luận
5	Content	Text	Nội dung bình luận

**Bảng 6** **Bảng Status**

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	<b>StatusID</b>	Int	Số ID của trạng thái
2	Statusname	Varchar(45)	Tên trạng thái

**Bảng 7** Bảng Categories

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	CatID	Int	Số ID của danh mục
2	Catname	Varchar(45)	Tên danh mục
3	PcatID	Int	Số ID của danh mục cha

**Bảng 8** Bảng Parent\_categories

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	PCatID	Int	Số ID của danh mục cha
2	PCatname	Varchar(45)	Tên danh mục cha

**Bảng 9** Bảng Tag\_has\_articles

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	TagID	Int	Số ID của Tag
2	ArticleID	Int	Số ID của Article có tag

**Bảng 10** Bảng Tags

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	TagID	Int	Số ID của Tag
2	Tagname	Varchar(45)	Tên của tag

**Bảng 11** Bảng Editor\_manage\_categories

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	EditorID	Int	Số ID của người dùng có role Editor
2	CatID	Int	Số ID của bài viết mà Editor chỉnh sửa

## 2. Mô tả thiết kế

### a. Giới thiệu chung

Trong quá trình phát triển ứng dụng web, tôi sử dụng một bộ công nghệ mạnh mẽ và linh hoạt, tận dụng sức mạnh của Java trong môi trường web. Java Servlet, một công nghệ trong Java Enterprise Edition (Java EE), chịu trách nhiệm xử lý yêu cầu

HTTP, thực hiện business logic, và tương tác với cơ sở dữ liệu. Servlet thường đóng vai trò Controller trong mô hình MVC, điều khiển luồng làm việc của ứng dụng.

Để tạo ra giao diện người dùng động và dễ bảo trì, tôi sử dụng JavaServer Pages (JSP). JSP là công nghệ giúp kết hợp mã Java và HTML, tạo ra trang web linh hoạt và thân thiện với người dùng.

Cùng với mô hình MVC (Model-View-Controller), tôi có một kiến trúc tổ chức ứng dụng mạnh mẽ. Model đại diện cho dữ liệu và business logic, Servlet làm Controller để xử lý yêu cầu và tương tác với Model, trong khi JSP đảm nhận vai trò View, hiển thị thông tin cho người dùng.

Để quản lý dự án và phụ thuộc, tôi sử dụng Maven và POM XML. Maven giúp tổ chức mã nguồn, quản lý thư viện, và thực hiện quy trình build một cách hiệu quả.

Tổng cộng, sự kết hợp của Java Servlet, JSP, và mô hình MVC cùng với Maven tạo ra một môi trường phát triển ứng dụng web linh hoạt, mạnh mẽ và dễ bảo trì.

### **b. Apache Tomcat**

Tomcat sẽ được sử dụng để host server để triển khai và chạy các Java Servlet. Tomcat cung cấp một môi trường thích hợp cho phát triển và triển khai ứng dụng web Java.

### **c. Java Servlet**

Java Servlet được gọi là một công nghệ trong Java Enterprise Edition (Java EE) được sử dụng để xây dựng ứng dụng web. Một Servlet là một lớp Java được thiết kế để mở rộng khả năng của máy chủ web hoặc một container servlet để tạo ra các ứng dụng web động.

Servlet chủ yếu được sử dụng để xử lý các yêu cầu HTTP từ máy khách (trình duyệt web) và tạo ra các trang web động dựa trên yêu cầu đó. Servlet có thể thực hiện nhiều chức năng như đọc và ghi dữ liệu từ và vào cơ sở dữ liệu, xử lý logic kinh doanh, và tương tác với các thành phần khác của ứng dụng web.

Cụ thể, một Servlet trong Java thường triển khai các phương thức HTTP như doGet() để xử lý yêu cầu GET và doPost() để xử lý yêu cầu POST. Servlet được quản lý bởi một container servlet như Apache Tomcat, và nó có thể được kết hợp với các công nghệ khác như JavaServer Pages (JSP) để tạo ra ứng dụng web đầy đủ chức năng và linh hoạt.

#### **d. Mô hình MVC**

Mô hình MVC tên đầy đủ là Model-View-Controller, một mẫu kiến trúc phân tách một ứng dụng thành ba thành phần logic chính Model, View và Controller. Do đó viết tắt MVC. Mỗi thành phần kiến trúc được xây dựng để xử lý khía cạnh phát triển cụ thể của một ứng dụng. MVC tách lớp logic nghiệp vụ và lớp hiển thị ra riêng biệt. Ngày nay, kiến trúc MVC đã trở nên phổ biến để thiết kế các ứng dụng web cũng như ứng dụng di động.

- **Model**

- **JavaBeans và Service Layer:** Thông tin từ cơ sở dữ liệu (Model) sẽ được đại diện bằng các JavaBeans, và có thể được truy cập thông qua một service layer. Các service sẽ xử lý các thao tác đọc và ghi dữ liệu.

- **View:**

- **JSP Pages:** Trang JSP sẽ đảm nhận vai trò của View trong mô hình MVC. Chúng sẽ chịu trách nhiệm hiển thị dữ liệu được cung cấp bởi Controller và tương tác với người dùng.

- **Controller:**

- **Servlets:** Servlets sẽ đóng vai trò của Controller trong mô hình MVC. Chúng sẽ xử lý yêu cầu từ người dùng, tương tác với Model để có dữ liệu và chuyển nó đến View để hiển thị.Maven

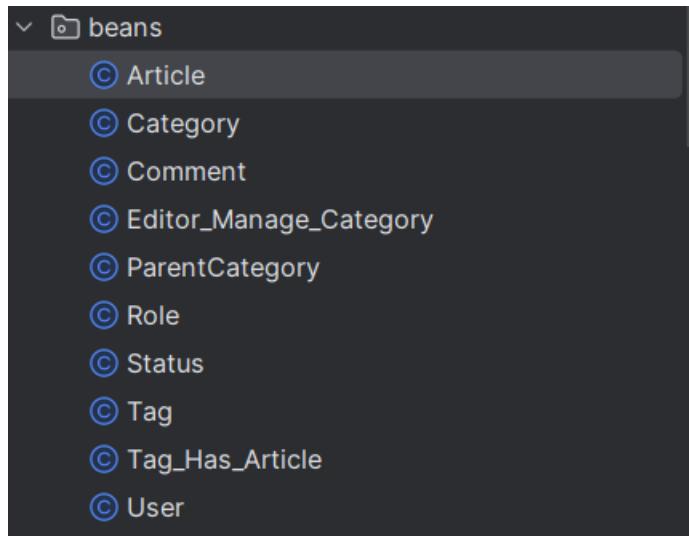
#### **e. Maven**

Maven và POM XML sẽ giúp quản lý dự án và các phụ thuộc một cách dễ dàng. POM XML sẽ đặt ra các thông tin như các thư viện cần thiết, cấu hình build, và các plugin hỗ trợ.

### **3. Xây dựng thuật toán**

#### **a. Beans**

Các beans đại diện cho dữ liệu và business logic của ứng dụng, lưu trữ thông tin như người dùng, bài viết, và các đối tượng liên quan đến ứng dụng. Cung cấp các phương thức để truy cập và cập nhật dữ liệu. Dưới đây là các beans được khởi tạo trong trang web:



Hình 4 Các beans của trang web

Mỗi bean sẽ đơn giản đại diện cho thông tin về các thành phần trong ứng dụng gồm có 4 phần sau:

- **Thuộc tính:**

- Mỗi bean sẽ đều có các thuộc tính đặc trưng như ‘ID’, ‘name’, và các thuộc tính khác phụ thuộc vào tính chất của đối tượng
- Ví dụ: các thuộc tính trong bean Article

```
public class Article {
    7 usages
    private int articleID;
    5 usages
    private String title;
    6 usages
    private Date publish_date;
    4 usages
    private int views;
    5 usages
    private String subContent;
    4 usages
    private String content;
    5 usages
    private int catID;
    5 usages
    private boolean premium;
    4 usages
    private int writerID;
    6 usages
    private int statusID;
    5 usages
    private String thumbs_img;
```

Hình 5 Các thuộc tính trong bean Article

- **Hàm tạo:**

- Cung cấp hàm tạo mặc định không tham số để tạo đối tượng mới mà không cần dữ liệu ban đầu.
- Hỗ trợ hàm tạo với tham số, giúp tạo đối tượng với thông tin cụ thể như tiêu đề, ngày xuất bản, danh mục, v.v.
- Hỗ trợ hàm tạo có tham số để tạo đối tượng mới khi không có ID được chỉ định, thường sử dụng khi tạo mới chưa lưu vào cơ sở dữ liệu.

```

public Article(int articleID, String title, Date publish_date, int views, String subContent, String content,
    this.articleID = articleID;
    this.title = title;
    this.publish_date = publish_date;
    this.views = views;
    this.subContent = subContent;
    this.content = content;
    this.catID = catID;
    this.premium = premium;
    this.writerID = writerID;
    this.statusID = statusID;
    this.thumbs_img = thumbs_img;
}

▲ Jinkky
public Article() {
}
▲ Jinkky
public Article(String title, Date publish_date, int views, String subContent, String content, int catID, boolean
    this.articleID = -1;
    this.title = title;
    this.publish_date = publish_date;
    this.views = views;
    this.subContent = subContent;
    this.content = content;
    this.catID = catID;
    this.premium = premium;
    this.writerID = writerID;
    this.statusID = statusID;
    this.thumbs_img = thumbs_img;
}

```

Hình 6 Các hàm tạo trong bean Article

#### • Phương thức Getter và setter

- Cung cấp các phương thức getter và setter để truy cập và cập nhật giá trị của các thuộc tính.
- Sử dụng tính đóng gói để bảo vệ dữ liệu và kiểm soát cách truy cập đến thuộc tính.

```

  ↳ Jinkky
    public String getThumbs_img() { return thumbs_img; }

  ↳ Jinkky
    public void setThumbs_img(String thumbsImg) { this.thumbs_img = thumbsImg; }
  ↳ Jinkky
    public int getArticleID() { return articleID; }

  ↳ Jinkky
    public void setArticleID(int articleID) { this.articleID = articleID; }

  ↳ Jinkky
    public String getTitle() { return title; }

  ↳ Jinkky
    public void setTitle(String title) { this.title = title; }

  17 usages  ↳ Jinkky
    public Date getPublish_date() { return publish_date; }

  no usages  ↳ Jinkky
    public void setPublish_date(Date publish_date) { this.publish_date = publish_date; }

  ↳ Jinkky
    public int getViews() { return views; }

  ↳ Jinkky
    public void setViews(int views) { this.views = views; }

  5 usages  ↳ Jinkky
    public String getSubContent() { return subContent; }

```

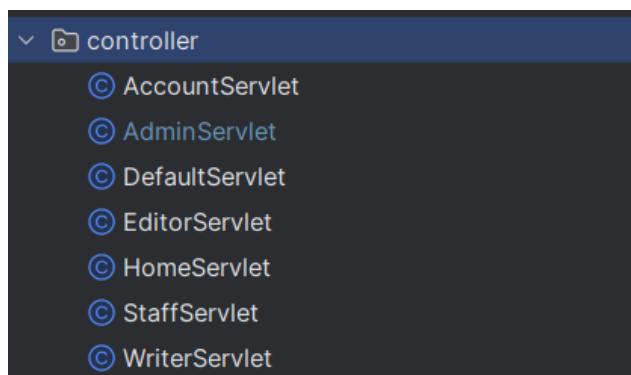
Hình 7 Các phương thức getter và setter trong bean Article bean

- **Hỗ trợ thêm:**

- Hàm tạo có tham số được thiết kế để tạo một đối tượng mới khi không có ID được chỉ định. Thường được sử dụng khi tạo một đối tượng mới chưa được lưu vào cơ sở dữ liệu.

### b. Controller

Controller là thành phần chính của mô hình MVC (Model-View-Controller) trong ứng dụng web. Nhiệm vụ chính của Controller là xử lý các yêu cầu từ người dùng, tương tác với Model để lấy hoặc cập nhật dữ liệu, và sau đó chuyển dữ liệu đó đến View để hiển thị cho người dùng.



Hình 8 Các Controller có trong web

Mỗi Controller sẽ thực hiện các nhiệm vụ sau:

- **Xử Lý Yêu Cầu:**
  - Controller nhận các yêu cầu HTTP từ phía Client (trình duyệt web hoặc ứng dụng di động) thông qua các phương thức như GET, POST, và các phương thức khác.
  - Sử dụng các phương thức xử lý yêu cầu để quyết định cách xử lý dữ liệu và chuyển hướng flow của ứng dụng.
- **Tương Tác với Model:**
  - Gọi các phương thức của Model để lấy dữ liệu từ cơ sở dữ liệu hoặc cập nhật dữ liệu vào cơ sở dữ liệu.
  - Đảm bảo tính logic và hiệu suất trong quá trình tương tác với Model.
- **Chuyển Dữ Liệu đến View:**
  - Sau khi có dữ liệu từ Model, Controller chuyển gói dữ liệu này và quyết định view cần được hiển thị.
  - Gửi dữ liệu đến View để hiển thị thông tin cho người dùng.

Dưới đây là mô tả cho từng Controller:

#### **AccountServlet**

Đối với ‘**doGet**’ (Xử lý yêu cầu GET):

- **Login (/Account/Login):**
  - Nếu người dùng đã đăng nhập, chuyển hướng đến trang chính.
  - Nếu chưa đăng nhập, hiển thị trang đăng nhập.
- **Register (/Account/Register):**
  - Hiển thị trang đăng ký.
- **Forget Password (/Account/ForgetPassword):**
  - Hiển thị trang khôi phục mật khẩu.
- **Verify (/Account/Verify):**
  - Kiểm tra mã OTP, nếu đúng, chuyển hướng đến trang cập nhật mật khẩu.
  - Nếu không đúng, hiển thị thông báo lỗi.
- **Profile (/Account/Profile):**

- Kiểm tra xem người dùng đã đăng nhập chưa.
- Nếu đã đăng nhập, hiển thị trang thông tin cá nhân.
- Nếu chưa đăng nhập, chuyển hướng đến trang đăng nhập.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String path = request.getPathInfo();
    if(path == null || path.equals("/")){
        path = "/Index";
    }
    HttpSession session;
    switch (path){
        case "/Login":
            session = request.getSession();
            if((Boolean) session.getAttribute("s: auth")){
                ServletUtils.redirect(url: "/Home", request, response);
            } else ServletUtils.forward(url: "/views/vwAccount/login.jsp", request, response);
            break;
        case "/Register":
            ServletUtils.forward(url: "/views/vwAccount/register.jsp", request, response);
            break;
        case "/ForgotPassword":
            ServletUtils.forward(url: "/views/vwAccount/recovery.jsp", request, response);
            break;
        case "/Verify":
            String otp = request.getParameter("s: otp");
            User u = UserService.findUserByOTP(otp);
            if(u != null){
                //Sau khi đã verify rồi thì sẽ xóa OTP của user đó
                UserService.deleteOTP(u.getUserId());

                request.setAttribute("s: verifiedUser", u);
                ServletUtils.forward(url: "/views/vwAccount/newpwd.jsp", request, response);
            }
    }
}
```

Hình 9 AccoutServlet Xử lý yêu cầu GET

Đối với ‘**doPost**’ (Xử lý yêu cầu POST):

- Login (/Account/Login):
  - Xử lý đăng nhập người dùng.
  - Chuyển hướng đến trang chính tương ứng với quyền của người dùng.
- Logout (/Account/Logout):
  - Đăng xuất người dùng và chuyển hướng đến trang chính.
- Register (/Account/Register):
  - Xử lý đăng ký người dùng, kiểm tra Captcha.
  - Hiển thị thông báo kết quả đăng ký.
- Update (/Account/Update):
  - Xử lý cập nhật thông tin cá nhân của người dùng.
  - Hiển thị thông báo kết quả cập nhật.
- Forget Password (/Account/ForgetPassword):

- Gửi mã OTP qua email để khôi phục mật khẩu.
- Hiển thị thông báo về việc kiểm tra email.
- Update Password (/Account/UpdatePassword):
  - Cập nhật mật khẩu sau quá trình khôi phục.
  - Hiển thị thông báo kết quả khôi phục mật khẩu.

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String path = request.getPathInfo();
    switch (path) {
        case "/Login":
            loginUser(request, response);
            break;
        case "/Logout":
            logoutUser(request, response);
            break;
        case "/Register":
            registerUser(request, response);
            break;
        case "/Update":
            updateUser(request, response);
            break;
        case "/ForgotPassword":
            forgetPassword(request, response);
            break;
        case "/UpdatePassword":
            updatePassword(request, response);
            break;
        default:
            ServletUtils.forward(url: "/views/404.jsp", request, response);
            break;
    }
}
```

Hình 10 AccountServlet Xử lý yêu cầu POST

Tóm tắt chung:

- Controller quản lý các yêu cầu liên quan đến tài khoản người dùng, bao gồm đăng nhập, đăng ký, cập nhật thông tin cá nhân và khôi phục mật khẩu.
- Sử dụng các trang JSP để hiển thị giao diện và tương tác với người dùng.
- Thực hiện kiểm tra quyền và chuyển hướng đến trang tương ứng sau khi đăng nhập.
- Sử dụng các phương thức của **UserService** để thực hiện các thao tác liên quan đến người dùng.

```

private static void logoutUser(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    HttpSession session = request.getSession();
    session.removeAttribute("auth");
    session.removeAttribute("authUser");
    String url = "/Home";
    ServletUtils.redirect(url, request, response);
}

1 usage  ± Jinkky
private static void updateUser(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    HttpSession session = request.getSession();

    int userid = Integer.parseInt(request.getParameter("userid"));
    String username = ((User) session.getAttribute("authUser")).getUsername();

    String firstname = request.getParameter("firstname");
    String lastname = request.getParameter("lastname");
    Date dob = Date.valueOf(request.getParameter("dob"));
    String email = request.getParameter("email");

    User u = new User(userid, username, firstname, lastname, dob, email);
    UserService.update(u);

    //Cập nhật lại session với thông tin user vừa update
    session.setAttribute("authUser", u);
    request.setAttribute("successfulUpdate", "Cập nhật thông tin thành công!");
    ServletUtils.forward(url: "/views/vwAccount/profile.jsp", request, response);
}

```

Hình 11 AccounServlet xử lý một số chức năng

## AdminServlet

Đối với ‘**doGet**’ (Xử lý yêu cầu GET):

- Quản lý Người Dùng (/Staff/Admin/User):
  - Hiển thị danh sách người dùng có phân trang.
  - Cho phép chuyển đến trang người dùng tiếp theo hoặc trang trước.
  - Hiển thị thông tin chi tiết và quyền của người dùng.
  - Cho phép thay đổi quyền của người dùng.
- Quản lý Bài Viết (/Staff/Admin/Article):
  - Hiển thị danh sách bài viết có phân trang.
  - Cho phép chuyển đến trang bài viết tiếp theo hoặc trang trước.
  - Hiển thị thông tin chi tiết và trạng thái của bài viết.
  - Cho phép thay đổi trạng thái của bài viết hoặc xóa bài viết.
- Quản lý Thẻ (/Staff/Admin/Tag):
  - Hiển thị danh sách thẻ có phân trang.
  - Cho phép chuyển đến trang thẻ tiếp theo hoặc trang trước.
  - Cho phép thêm, sửa đổi và xóa thẻ.
- Quản lý Danh Mục (/Staff/Admin/Category):
  - Hiển thị danh sách danh mục có phân trang.

- Cho phép chuyển đến trang danh mục tiếp theo hoặc trang trước.
- Cho phép thêm, sửa đổi và xóa danh mục.

```

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("authUser");
    int perm = u.getRoleID();
    if(perm == 4){
        String path = request.getPathInfo();
        if(path == null || path.equals("/")){
            path = "/User";
        }
        switch (path){
            case "/User":
                //Lay tat ca user
                int page = Integer.parseInt(request.getParameter("page"));
                List<User> allOriginalUsers = UserService.findAll();
                List<User> allUsers = UserService.findAllWithPaging(page);
                int maxPage = allOriginalUsers.size()/10 + 1;
                request.setAttribute("allUsers", allUsers);
                request.setAttribute("page", page);
                request.setAttribute("maxPage", maxPage);
                ServletUtils.forward(url: "/views/vwAdmin/vwAdminUser/index.jsp", request, response);
                break;
            case "/Article":
                int articlepage = Integer.parseInt(request.getParameter("page"));
                List<Article> allArticles = ArticleService.findAllWithPaging(articlepage);
                List<Article> allOriginalArticles = ArticleService.findAll();
                int maxArticlePage = allOriginalArticles.size()/10 + 1;
                request.setAttribute("maxPage", maxArticlePage);
                request.setAttribute("allArticles", allArticles );
                request.setAttribute("page", articlepage);
                ServletUtils.forward(url: "/views/vwAdmin/vwAdminArticle/index.jsp", request, response);
        }
    }
}

```

Hình 12 AdminServlet Xử lý yêu cầu GET

### Đối với ‘**doPost**’ (Xử lý yêu cầu POST):

- Thay Đổi Quyền Người Dùng (/Staff/Admin/ChangeUserRole):
  - Xử lý yêu cầu thay đổi quyền của người dùng.
  - Chuyển hướng đến trang quản lý người dùng.
- Xóa Người Dùng (/Staff/Admin/DeleteUser):
  - Xử lý yêu cầu xóa người dùng.
  - Chuyển hướng đến trang quản lý người dùng.
- Thay Đổi Trạng Thái Bài Viết (/Staff/Admin/ChangeArticleStatus):
  - Xử lý yêu cầu thay đổi trạng thái của bài viết.
  - Chuyển hướng đến trang quản lý bài viết.
- Xóa Bài Viết (/Staff/Admin/DeleteArticle):
  - Xử lý yêu cầu xóa bài viết.
  - Chuyển hướng đến trang quản lý bài viết.
- Thêm Thẻ (/Staff/Admin/AddTag):

- Xử lý yêu cầu thêm mới thẻ.
  - Chuyển hướng đến trang quản lý thẻ.
- Sửa Đổi Thẻ (/Staff/Admin/UpdateTag):
  - Xử lý yêu cầu sửa đổi thông tin thẻ.
  - Chuyển hướng đến trang quản lý thẻ.
- Xóa Thẻ (/Staff/Admin/DeleteTag):
  - Xử lý yêu cầu xóa thẻ.
  - Chuyển hướng đến trang quản lý thẻ.
- Thêm Danh Mục (/Staff/Admin/AddCategory):
  - Xử lý yêu cầu thêm mới danh mục.
  - Chuyển hướng đến trang quản lý danh mục.
- Sửa Đổi Danh Mục (/Staff/Admin/EditCategory):
  - Xử lý yêu cầu sửa đổi thông tin danh mục.
  - Chuyển hướng đến trang quản lý danh mục.
- Xóa Danh Mục (/Staff/Admin/DeleteCategory):
  - Xử lý yêu cầu xóa danh mục.
  - Chuyển hướng đến trang quản lý danh mục.

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("authUser");
    int perm = u.getRoleID();
    if(perm == 4){
        String path = request.getPathInfo();
        switch (path) {
            case "/ChangeUserRole":
                changeRole(request, response);
                break;
            case "/DeleteUser":
                deleteUser(request, response);
                break;
            case "/ChangeArticleStatus":
                changeArticleStatus(request, response);
                break;
            case "/DeleteArticle":
                deleteArticle(request, response);
                break;
            case "/AddTag":
                addTag(request, response);
                break;
            case "/UpdateTag":
                updateTag(request, response);
                break;
            case "/DeleteTag":
                deleteTag(request, response);
                break;
            case "/AddCategory":
                addCategory(request, response);
                break;
        }
    }
}
```

Hình 13 AdminServlet Xử lý yêu cầu POST

Tóm tắt chung:

- Controller này quản lý các yêu cầu liên quan đến quản trị viên (Admin) như quản lý người dùng, bài viết, thẻ và danh mục.
- Sử dụng các trang JSP để hiển thị giao diện và tương tác với người dùng.
- Thực hiện kiểm tra quyền và chuyển hướng đến trang chính sau khi thực hiện các thao tác.

```
usage: Jinkky
private static void changeRole(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    int userid = Integer.parseInt(request.getParameter("userid"));
    int roleid = Integer.parseInt(request.getParameter("roleid"));

    User u = new User(userid, roleid);
    UserService.updateRole(u);

    ServletUtils.redirect(url: "/Staff/Admin/User?page=1", request, response);
}

usage: Jinkky
private static void deleteUser(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    int userid = Integer.parseInt(request.getParameter("userid"));

    UserService.delete(userid);

    ServletUtils.redirect(url: "/Staff/Admin/User?page=1", request, response);
}

usage: Jinkky
private static void changeArticleStatus(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    int articleid = Integer.parseInt(request.getParameter("articleid"));
    int statusid = Integer.parseInt(request.getParameter("statusid"));

    Article a = new Article(articleid, statusid);
    ArticleService.updateStatus(a);

    ServletUtils.redirect(url: "/Staff/Admin/Article?page=1", request, response);
}
```

Hình 14 AdminServlet xử lý các chức năng

## DefaultServlet

Đối với ‘**doGet**’ (Xử lý yêu cầu GET):

- Nếu không có đường dẫn cụ thể hoặc đường dẫn là /, chuyển hướng đến trang chủ (/Home).
- Nếu có đường dẫn khác, chuyển hướng đến trang 404.

Đối với ‘**doPost**’ (Xử lý yêu cầu POST):

- Chuyển hướng đến trang 404 khi nhận yêu cầu POST.

Tóm tắt chung:

- Controller này chỉ xử lý yêu cầu GET và POST đơn giản để chuyển hướng đến trang chủ hoặc trang 404 tùy thuộc vào đường dẫn được yêu cầu.

- Sử dụng ServletUtils để thực hiện các chuyển hướng và chuyển tiếp yêu cầu.
- Nếu đường dẫn không phải là /, chuyển hướng đến trang 404 để thông báo về đường dẫn không hợp lệ.

```
package com.example.googlenewsclone.controller;

import ...

▲ Jinkky
@WebServlet(name = "DefaultServlet", value = "/")
public class DefaultServlet extends HttpServlet {
    no usages ▲ Jinkky
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String path = request.getPathInfo();
        if (path == null || path.equals("/")) {
            ServletUtils.redirect(url: "/Home", request, response);
        } else
            ServletUtils.forward(url: "views/404.jsp", request, response);
    }

    no usages ▲ Jinkky
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        ServletUtils.forward(url: "views/404.jsp", request, response);
    }
}
```

Hình 15 Xử lý các chức năng câu DefaultServlet

## EditorServlet

Đối với ‘**doGet**’ (Xử lý yêu cầu GET):

- Trang Chủ Editor:
  - Hiển thị danh sách bài viết dựa trên trạng thái và trang được chỉ định.
  - Cho phép người biên tập xem và quản lý bài viết của mình.
  - Sử dụng phân trang để hiển thị một lượng hạn chế các bài viết trên mỗi trang.
  - Thực hiện chuyển hướng đến trang 404 nếu có đường dẫn không hợp lệ.

```

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("authUser");
    int perm = u.getRoleID();
    if (perm == 3) {
        String path = request.getPathInfo();
        if (path == null || path.equals("/")) {
            path = "/Index";
        }
        switch (path) {
            case "/Index":
                List<Article> allOriginalArticles = ArticleService.findAll();
                int statusid = Integer.parseInt(request.getParameter("statusid"));
                int page = Integer.parseInt(request.getParameter("page"));
                int maxPage = allOriginalArticles.size() / 10 + 1;
                List<Article> allArticles = ArticleService.findAllByStatusIdAndPaging(statusid, page);
                request.setAttribute("allArticles", allArticles);
                request.setAttribute("page", page);
                request.setAttribute("statusid", statusid);
                request.setAttribute("maxPage", maxPage);
                ServletUtils.forward(url: "/views/vwEditor/index.jsp", request, response);
                break;
            default:
                ServletUtils.forward(url: "/views/404.jsp", request, response);
                break;
        }
    } else {
        ServletUtils.redirect(url: "/Home", request, response);
    }
}

```

Hình 16 EditorServlet xử lý yêu cầu GET

Đối với ‘**doPost**’ (Xử lý yêu cầu POST):

- Cập Nhật Trạng Thái Bài Viết:
  - Thay đổi trạng thái của bài viết (public, draft, hoặc rejected).
  - Nếu trạng thái là public, cập nhật ngày xuất bản của bài viết.
  - Thực hiện chuyển hướng đến trang danh sách bài viết sau khi cập nhật.

```

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("authUser");
    int perm = u.getRoleID();
    if (perm == 3) {
        String path = request.getPathInfo();
        switch (path) {
            case "/UpdateStatus":
                updateStatus(request, response);
                break;
            default:
                ServletUtils.forward(url: "/views/404.jsp", request, response);
                break;
        }
    } else{
        ServletUtils.redirect(url: "/Home", request, response);
    }
}

```

Hình 17 EditorServlet xử lý yêu cầu POST

Tóm tắt chung:

- Controller này cho phép biên tập viên quản lý các bài viết của mình, xem trạng thái và thực hiện các thao tác như cập nhật trạng thái.
- Sử dụng ServletUtils để thực hiện các chuyển hướng và chuyển tiếp yêu cầu.
- Nếu không phải là biên tập viên, chuyển hướng đến trang chủ.

```
private static void updateStatus(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    int articleid = Integer.parseInt(request.getParameter("articleid"));
    int statusid = Integer.parseInt(request.getParameter("statusid"));

    if(statusid == 4){
        Date publish_date = Date.valueOf(LocalDate.now());
        Article a = new Article(articleid, statusid, publish_date);
        ArticleService.publishArticle(a);
    } else{
        Article a = new Article(articleid, statusid);
        ArticleService.updateStatus(a);
    }

    ServletUtils.redirect(url: "/Staff/Editor/Index?statusid=0&page=1", request, response);
}
```

Hình 18 EditorServlet xử lý chức năng cập nhật trạng thái

## HomeServlet

Đối với ‘**doGet**’ (Xử lý yêu cầu GET):

- Trang Chủ:
  - Hiển thị các thông tin chính như danh sách chuyên mục, bài viết nổi bật, bài viết mới nhất, và bài viết ngẫu nhiên.
  - Sử dụng phân trang cho bài viết nổi bật và mới nhất.
  - Thực hiện chuyển hướng đến trang 404 nếu có đường dẫn không hợp lệ.
- Trang Chi Tiết Bài Viết:
  - Hiển thị thông tin chi tiết của một bài viết cụ thể, bao gồm tác giả, chuyên mục, các tag, bài viết liên quan, và các comment.
  - Kiểm tra quyền truy cập để xem nội dung bài viết (kiểm tra trạng thái và quyền premium).
  - Cập nhật lượt xem của bài viết khi người dùng truy cập.
- Trang Theo Dõi Chuyên Mục:
  - Hiển thị danh sách bài viết theo chuyên mục được chọn.
  - Xử lý trường hợp chuyên mục trống hoặc không tồn tại.

- Trang Theo Dõi Tag:
  - Hiển thị danh sách bài viết theo tag được chọn.
  - Xử lý trường hợp tag trống hoặc không tồn tại.
- Trang Tìm Kiếm:
  - Hiển thị danh sách bài viết dựa trên từ khóa tìm kiếm.
  - Kiểm tra quyền truy cập để xem nội dung bài viết (kiểm tra trạng thái và quyền premium).

```

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String path = request.getPathInfo();
    if (path == null || path.equals("/")) {
        path = "/Index";
    }
    HttpSession session;
    switch (path) {
        case "/Index":
            //Tim tất cả chuyên mục
            List<Category> categories = CategoryService.findAll();
            request.setAttribute("categories", categories);

            //Tim top 9 bài viết có nhiều lượt views cao nhất
            List<Article> topViews = ArticleService.sortByView();
            request.setAttribute("topArticles", topViews);

            //Tim 9 bài viết xuất bản mới nhất
            List<Article> newestDate = ArticleService.sortByDate();
            request.setAttribute("newestArticles", newestDate);

            //Tim 15 bài viết ngẫu nhiên toàn mục
            List<Article> random = ArticleService.findArticlesbyRandom();
            request.setAttribute("randomArticles", random);

            ServletUtils.forward(url: "/views/vwHome/index.jsp", request, response);
            break;
        case "/Article":
            int id = 0;
            try{
                id = Integer.parseInt(request.getParameter("id"));
            }catch (NumberFormatException e){
    
```

Hình 19 HomeServlet Xử lý yêu cầu GET (tóm tắt)

Đối với ‘**doPost**’ (Xử lý yêu cầu POST):

- Thêm Bình Luận:
  - Xử lý việc thêm bình luận mới vào một bài viết.
  - Chuyển hướng người dùng sau khi thêm bình luận.

```

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String path = request.getPathInfo();
    switch (path) {
        case "/AddComment":
            addComment(request, response);
            break;
        default:
            ServletUtils.forward(url: "/views/404.jsp", request, response);
            break;
    }
}

```

Hình 20 HomeServlet Xử lý yêu cầu POST

Tóm tắt chung:

- Controller này quản lý nhiều chức năng liên quan đến trang chủ và hiển thị chi tiết bài viết.
- Kiểm tra quyền truy cập của người dùng để xem nội dung bài viết (trạng thái, quyền premium).
- Sử dụng ServletUtils để thực hiện các chuyển hướng và chuyển tiếp yêu cầu.
- Thực hiện xử lý các trường hợp lỗi hoặc trường hợp không có dữ liệu.

```

private static void addComment(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");

    String content = request.getParameter(s: "content");
    Date date = Date.valueOf(LocalDate.now());
    int userid = Integer.parseInt(request.getParameter(s: "userid"));
    int articleid = Integer.parseInt(request.getParameter(s: "articleid"));
    Comment c = new Comment(content, date, userid, articleid);
    CommentService.add(c);

    ServletUtils.redirect(url: "/Home/Article?id=" + request.getParameter(s: "articleid"), request, response);
}

1 usage ▲ Jinkky
private static void ftxSearch(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    String keywords = request.getParameter(s: "ftxsearch");
    List<Article> articleslist = ArticleService.ftxSearch(keywords);
    request.setAttribute(s: "keywords", keywords);
    request.setAttribute(s: "ftxSearchArticles", articleslist);

    List<Category> catList = CategoryService.findAll();
    request.setAttribute(s: "categories", catList);

    ServletUtils.forward(url: "/views/vwHome/bySearch.jsp", request, response);
}

1 usage ▲ Jinkky
private static void ftxSearchPremiumFirst(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    String keywords = request.getParameter(s: "ftxsearch");
    List<Article> articleslist = ArticleService.ftxSearchPremiumFirst(keywords);
    request.setAttribute(s: "keywords", keywords);
    request.setAttribute(s: "ftxSearchArticles", articleslist);
}

```

Hình 21 HomeServlet xử lý một số chức năng

## StaffServlet

Đối với ‘**doGet**’ (Xử lý yêu cầu GET):

- Trang Editor:
  - Hiển thị trang danh sách bài viết cho vai trò "Editor".
  - Sử dụng chuyển hướng để đưa người dùng đến trang danh sách bài viết của Editor.
- Trang Writer:
  - Hiển thị trang danh sách bài viết cho vai trò "Writer".
  - Sử dụng chuyển hướng để đưa người dùng đến trang danh sách bài viết của Writer.
- Trang Admin:
  - Hiển thị trang quản lý người dùng cho vai trò "Admin".
  - Sử dụng chuyển hướng để đưa người dùng đến trang quản lý người dùng của Admin.
- Trang 404:
  - Trang lỗi được hiển thị khi có đường dẫn không hợp lệ hoặc không khớp.

```
public class StaffServlet extends HttpServlet {  
    no usages  ↳ Jinkky  
    @Override  
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
        String path = request.getPathInfo();  
        switch (path){  
            case "/Editor":  
                ServletUtils.forward( url: "/views/vwEditor/index.jsp", request, response);  
                break;  
            case "/Writer":  
                ServletUtils.forward( url: "/views/vwWriter/index.jsp", request, response);  
                break;  
            case "/Admin":  
                ServletUtils.forward( url: "/views/vwAdmin/vwAdminUser/index.jsp", request, response);  
                break;  
            default:  
                ServletUtils.forward( url: "/views/404.jsp", request, response);  
                break;  
        }  
    }  
}
```

Hình 22 StaffServlet Xử lý yêu cầu GET

Đối với ‘**doPost**’ (Xử lý yêu cầu POST):

- Sử dụng chuyển hướng để đưa người dùng đến trang lỗi 404 nếu có yêu cầu POST không hợp lệ.

```

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String path = request.getPathInfo();
    switch (path) {
        default:
            ServletUtils.forward(url: "/views/404.jsp", request, response);
            break;
    }
}

```

Hình 23 StaffServlet Xử lý yêu cầu POST

Tóm tắt chung:

- Controller này chủ yếu quản lý việc chuyển hướng người dùng đến các trang tương ứng với vai trò (Editor, Writer, Admin).
- Mọi thay đổi hoặc mở rộng sau này có thể được thêm vào trong các phương thức doGet và doPost để xử lý yêu cầu cụ thể của từng vai trò.

### WriterServlet

Đối với ‘**doGet**’ (Xử lý yêu cầu GET):

- Trang Index cho Writer:
  - Hiển thị danh sách bài viết cho vai trò "Writer".
  - Chia bài viết thành các trang với quản lý phân trang.
  - Cho phép chọn xem bài viết theo trạng thái và trang cụ thể.
- Trang Thêm Bài Viết (Add):
  - Hiển thị trang thêm bài viết với danh sách chuyên mục và tag để chọn.
  - Người dùng cần đăng nhập để thêm bài viết.
  - Làm chuyển hướng đến trang đăng nhập nếu người dùng chưa đăng nhập.
- Trang Chính Sửa Bài Viết (Edit):
  - Hiển thị trang chỉnh sửa bài viết với thông tin chi tiết, danh sách chuyên mục và tag để chọn.
- Trang 404:
  - Trang lỗi được hiển thị khi có đường dẫn không hợp lệ hoặc không khớp.

```

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("authUser");
    int perm = u.getRoleID();
    if (perm == 2) {
        String path = request.getPathInfo();
        if(path == null || path.equals("/")) {
            path = "/Index";
        }
        switch (path){
            case "/Index":
                List<Article> originalArticles = ArticleService.findListArticleByWriterID(u.getUserID());
                int maxPage = originalArticles.size() / 10 + 1;
                int statusid = Integer.parseInt(request.getParameter("statusid"));
                int page = Integer.parseInt(request.getParameter("page"));
                List<Article> writerArticle;
                writerArticle = ArticleService.findArticlesByWriterIDAndStatusAndPage(u.getUserID(), statusid);
                request.setAttribute("statusid", statusid);
                request.setAttribute("page", page);
                request.setAttribute("articles", writerArticle);
                request.setAttribute("maxPage", maxPage);
                ServletUtils.forward(url: "/views/vwWriter/index.jsp", request, response);

                break;
            case "/Add":
                session = request.getSession();
                if(!((boolean) session.getAttribute("auth"))){
                    ServletUtils.redirect(url: "/Account/Login", request, response);
                } else {
                    List<Category> listcategories = CategoryService.findAll();
                    List <Tag> listtags = TagService.findAll();
                    request.setAttribute("categories", listcategories);
                }
        }
    }
}

```

Hình 24 WriterServlet Xử lý yêu cầu GET (tóm tắt)

Đối với ‘**doPost**’ (Xử lý yêu cầu POST):

- Thêm Bài Viết (Add):
  - Xử lý yêu cầu thêm bài viết.
  - Lưu thông tin bài viết mới và các tag liên quan.
  - Chuyển hướng đến trang danh sách bài viết của Writer.
- Chính Sửa Bài Viết (Edit):
  - Xử lý yêu cầu chỉnh sửa bài viết.
  - Lưu thông tin chỉnh sửa và cập nhật các tag liên quan.
  - Chuyển hướng đến trang danh sách bài viết của Writer.
- Xóa Bài Viết (Delete):
  - Xử lý yêu cầu xóa bài viết.
  - Chuyển hướng đến trang danh sách bài viết của Writer sau khi xóa.
- Xử Lý Mặc Định (Default):
  - Chuyển hướng đến trang lỗi 404 nếu có yêu cầu không hợp lệ.

```

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String path = request.getPathInfo();
    switch (path) {
        case "/Add":
            addArticle(request, response);
            break;
        case "/Edit":
            editArticle(request, response);
            break;
        case "/Delete":
            deleteArticle(request, response);
            break;
        default:
            ServletUtils.forward(url: "/views/404.jsp", request, response);
            break;
    }
}

```

Hình 25 WriterServlet xử lý yêu cầu POST

Tóm tắt chung:

- Controller này quản lý chức năng của vai trò "Writer" trong hệ thống.
- Đảm bảo người dùng được chuyển hướng đến trang đăng nhập khi cần thiết.
- Chức năng quản lý tag của bài viết được thực hiện thông qua TagHasArticleService.

```

private static void addArticle(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    String title = request.getParameter(s: "title");
    String[] tags = request.getParameterValues(s: "tags");

    String subcontent = request.getParameter(s: "subcontent");
    String thumbs_img = request.getParameter(s: "thumbs_img");
    String content = request.getParameter(s: "content");
    int catid = Integer.parseInt(request.getParameter(s: "catid"));
    int writerid = Integer.parseInt(request.getParameter(s: "writerid"));
    Boolean premium = Boolean.valueOf(request.getParameter(s: "premium"));
    ArticleService.add(title, subcontent, content, thumbs_img, catid, writerid, premium);
    Article excutedArticle = ArticleService.findLast();
    for(String id:tags){
        System.out.println("tag id" + Integer.parseInt(id));
        System.out.println("article id"+ excutedArticle.getArticleID());
        TagHasArticleService.add(Integer.parseInt(id), excutedArticle.getArticleID());
    }
    ServletUtils.redirect(url: "/Staff/Writer/Index?statusid=0&page=1", request, response);
}

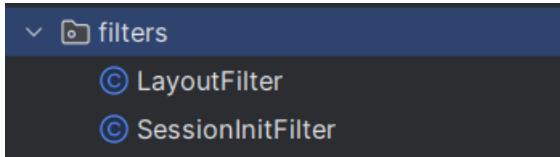
usage ▲ Jinkky
private static void editArticle(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    String id = request.getParameter(s: "id");
    String title = request.getParameter(s: "title");
    String thumbs_img = request.getParameter(s: "thumbs_img");
    String content = request.getParameter(s: "content");
    String subcontent = request.getParameter(s: "subcontent");
    Boolean premium = Boolean.valueOf(request.getParameter(s: "premium"));
    ArticleService.update(Integer.parseInt(id), title, subcontent, content, thumbs_img, premium);
    ServletUtils.redirect(url: "/Staff/Writer/Index?statusid=0&page=1", request, response);
}

```

Hình 26 WriterServlet xử lý một số chức năng

### c. Filters

Mỗi khi có yêu cầu đến trang web, nó sẽ đi kèm với filter, layout filter dùng để tải các danh mục trên thanh menu. Có 2 Filter được áp dụng vào trang web:



Hình 27 Các loại filter cho mỗi yêu cầu

#### **LayoutFilter**

- Filter này được áp dụng cho mọi đường dẫn ("/\*") trong ứng dụng.
- Dùng để chia sẻ thông tin chung như danh mục cha và danh mục trong toàn bộ ứng dụng web.
- Thuộc tính "parentCategories" và "categories" có thể được sử dụng bởi các trang JSP hoặc servlet để hiển thị thông tin liên quan đến danh mục.

Hàm doFilter:

- Thực hiện trước khi yêu cầu được chuyển đến servlet hoặc JSP và sau khi được xử lý bởi servlet hoặc JSP.
- Tìm tất cả các danh mục cha (Parent Category) và đặt chúng vào thuộc tính yêu cầu "parentCategories".
- Tìm tất cả các danh mục (Category) và đặt chúng vào thuộc tính yêu cầu "categories".

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws ServletException, IOException {
    //Tim tất cả Parent Cat
    List<ParentCategory> parentList = ParentCategoryService.findAll();
    request.setAttribute("parentCategories", parentList);

    //Tim tất cả Category
    List<Category> catList = CategoryService.findAll();
    request.setAttribute("categories", catList);

    chain.doFilter(request, response);
}
```

Hình 28 Chức năng của LayoutFilter

#### **SessionInitFilter:**

- Filter này được áp dụng cho mọi đường dẫn ("/\*") trong ứng dụng.
- Dùng để khởi tạo phiên đăng nhập và đối tượng người dùng nếu chúng không tồn tại.
- Giúp đảm bảo rằng có một phiên đăng nhập và đối tượng người dùng khả dụng trước khi xử lý các yêu cầu khác trong ứng dụng web.

Hàm doFilter:

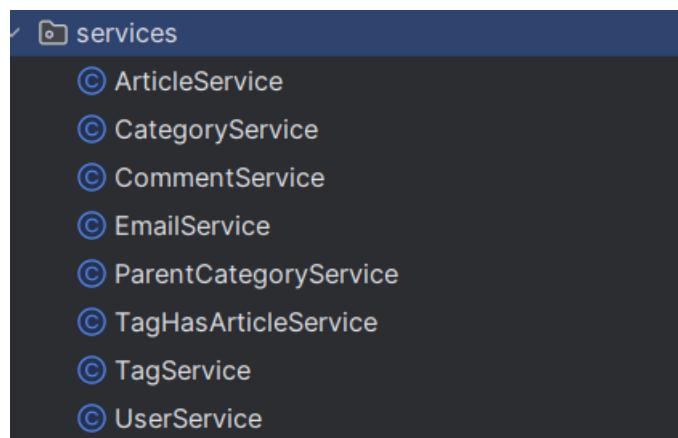
- Thực hiện trước khi yêu cầu được chuyển đến servlet hoặc JSP và sau khi được xử lý bởi servlet hoặc JSP.
- Ép kiểu ServletRequest thành HttpServletRequest để sử dụng các tính năng của HTTP.
- Kiểm tra xem có một phiên đăng nhập (auth) đã được thiết lập trong phiên hay không.
- Nếu không, thiết lập giá trị mặc định là false cho auth và một đối tượng User trống rỗng cho authUser.
- Tiếp tục chuỗi lọc với chain.doFilter(req, res) để chuyển yêu cầu đến các lọc hoặc servlet tiếp theo.

```
public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws ServletException, IOException {
    HttpServletRequest request = (HttpServletRequest) req;
    HttpSession session = request.getSession();
    if (session.getAttribute("auth") == null){
        session.setAttribute("auth", false);
        session.setAttribute("authUser", new User());
    }
    chain.doFilter(req, res);
}
```

Hình 29 Chức năng của SessionInitFilter

#### d. Services

Services đóng vai trò quan trọng trong xử lý business logic, tương tác với cơ sở dữ liệu và cung cấp dữ liệu cho Controllers và Views.



Hình 30 Tất cả Service được sử dụng trong trang web

Dưới đây là mô tả kĩ hơn cho từng Services

## ArticleService

ArticleService cung cấp các phương thức để thao tác với dữ liệu bài viết trong cơ sở dữ liệu. Các chức năng chính bao gồm:

- Lấy danh sách bài viết theo nhiều tiêu chí khác nhau (ID, trạng thái, người viết, danh mục, tag).
- Tìm kiếm bài viết theo từ khóa.
- Xử lý việc thêm mới, cập nhật, xoá bài viết.
- Sắp xếp và trả về danh sách bài viết theo các tiêu chí như lượt xem, ngày xuất bản, và ngẫu nhiên.
- Cập nhật số lượt xem của bài viết.
- Xử lý các thao tác liên quan đến người viết và trạng thái của bài viết.

```
public class ArticleService {  
    // Jinkky  
    public static List<Article> findAll(){  
        try (Connection con = DbUtils.getConnection()) {  
            final String query = "select * from articles ORDER BY articleid";  
            List<Article> list = con.createQuery(query)  
                .executeAndFetch(Article.class);  
            return list;  
        }  
    }  
    // 2 usages Jinkky  
    public static List<Article> findAllWithPaging(int page){  
        int currentOffset = (page-1)*10;  
        try (Connection con = DbUtils.getConnection()) {  
            final String query = "select * from articles ORDER BY articleid offset :offset limit 10";  
            List<Article> list = con.createQuery(query)  
                .addParameter("offset", currentOffset)  
                .executeAndFetch(Article.class);  
            return list;  
        }  
    }  
}
```

Hình 31 Một số đoạn code để xử lý và lấy dữ liệu của ArticleService

## CategoryService

CategoryService cung cấp các phương thức để thao tác với dữ liệu danh mục (category) trong cơ sở dữ liệu. Các chức năng chính bao gồm:

- Lấy danh sách tất cả các danh mục.
- Lấy danh sách các danh mục phân trang.
- Lấy thông tin chi tiết của một danh mục dựa trên ID.
- Xóa một danh mục dựa trên ID.
- Thêm mới một danh mục.

- Cập nhật thông tin của một danh mục dựa trên ID.

```

public class CategoryService {
    # Jinkky
    public static List<Category> findAll(){
        try (Connection con = DbUtils.getConnection()) {
            final String query = "select * from categories order by catid";
            return con.createQuery(query)
                .executeAndFetch(Category.class);
        }
    }

    1 usage  # Jinkky
    public static List<Category> findAllByPaging(int page){
        try (Connection con = DbUtils.getConnection()) {
            int currentOffset = (page-1) * 10;
            final String query = "select * from categories order by catid offset :offset limit 10";
            return con.createQuery(query)
                .addParameter( name: "offset", currentOffset)
                .executeAndFetch(Category.class);
        }
    }
}

```

Hình 32 Một số đoạn code để xử lý và lấy dữ liệu của CategoryService

## CommentService

CommentService cung cấp các phương thức để thao tác với dữ liệu bình luận (comment) trong cơ sở dữ liệu. Các chức năng chính bao gồm:

- Lấy danh sách tất cả các bình luận của một bài viết dựa trên ID của bài viết.
- Thêm mới một bình luận vào cơ sở dữ liệu.

```

public class CommentService{
    4 usages  # Jinkky
    public static List<Comment> findAllCommentInArticle(int articleId){
        try (Connection con = DbUtils.getConnection()) {
            final String query = "select * from comments where articleid = :articleid";
            List<Comment> list = con.createQuery(query)
                .addParameter( name: "articleid", articleId)
                .throwOnMappingFailure(false)
                .executeAndFetch(Comment.class);
            return list;
        }
    }
    # Jinkky
    public static void add(Comment c){
        try (Connection con = DbUtils.getConnection()) {
            final String query = "INSERT INTO comments (date, articleid, userid, content) VALUES (:date,:articleid,:userid,:content)";
            con.createQuery(query)
                .addParameter( name: "date", c.getDate())
                .addParameter( name: "articleid", c.getArticleID())
                .addParameter( name: "userid", c.getUserID())
                .addParameter( name: "content", c.getContent())
                .executeUpdate();
        }
    }
}

```

Hình 33 Một số đoạn code để xử lý và lấy dữ liệu của CommentService

## EmailService

EmailService cung cấp phương thức để gửi email xác nhận và khôi phục mật khẩu đến người dùng. Các chức năng chính bao gồm:

- `sendEmail`: Phương thức này nhận các thông số như **host**, **port**, **userName**, **password**, **toAddress**, **subject**, **message**, và **user**. Nó sử dụng JavaMail API để gửi một email thông qua một tài khoản email được cung cấp. Nội dung của email chứa một mã OTP (One Time Password) được sinh ra ngẫu nhiên bằng thư viện RandomString. Sau khi gửi email, mã OTP cũng được cập nhật trong cơ sở dữ liệu thông qua lớp UserService.

Cấu trúc email được xây dựng dưới dạng HTML, bao gồm các thành phần như tiêu đề, hình ảnh, nội dung chính, và một liên kết để xác nhận OTP.

```
public static void sendEmail(String host, String port,
                             final String userName, final String password, String toAddress,
                             String subject, String message, User user) throws MessagingException {
    // sets SMTP server properties
    Properties properties = new Properties();
    properties.put("mail.smtp.host", host);
    properties.put("mail.smtp.port", port);
    properties.put("mail.smtp.auth", "true");
    properties.put("mail.smtp.starttls.enable", "true");
    properties.put("mail.smtp.ssl.trust", "*");
    properties.put("mail.smtp.ssl.protocols", "TLSv1.2");
    properties.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");

    // creates a new session with an authenticator
    Authenticator auth = getPasswordAuthentication() -> {
        return new PasswordAuthentication(userName, password);
    };

    Session session = Session.getInstance(properties, auth);

    // tạo một email
    MimeMessage msg = new MimeMessage(session);
    msg.setFrom(new InternetAddress(userName));
    InternetAddress[] toAddresses = { new InternetAddress(toAddress) };
    msg.setRecipients(Message.RecipientType.TO, toAddresses);
    msg.setSubject(subject, charset: "UTF-8");
    msg.setSentDate(new Date());
    String OTP = RandomString.make( length: 30 );

    message = message.replace( target: "OTP", OTP );
    msg.setContent(message, type: "text/html; charset=UTF-8");
    // gửi mail
    Transport.send(msg);
```

Hình 34 Một số đoạn code để xử lý và lấy dữ liệu của EmailService

## ParentCategoryService

ParentCategoryService cung cấp phương thức để truy xuất danh sách các danh mục cha. Lớp này chủ yếu thực hiện các hoạt động đơn giản với cơ sở dữ liệu. Chức năng chính bao gồm:

- **findAll():** Truy xuất toàn bộ danh sách các danh mục cha từ bảng parent\_categories.

```
public class ParentCategoryService {  
    ▲ Jinkky  
    public static List<ParentCategory> findAll(){  
        try (Connection con = DbUtils.getConnection()) {  
            final String query = "select * from parent_categories";  
  
            return con.createQuery(query)  
                .executeAndFetch(ParentCategory.class);  
        }  
    }  
}
```

Hình 35 Hàm Findall() trong ParentCategoryService

## TagHasArticleService

TagHasArticleService có nhiệm vụ quản lý liên kết giữa các bài viết và các tag. Lớp này cung cấp các phương thức sau:

- **findAll(int id):** Truy xuất toàn bộ các liên kết giữa tag và bài viết dựa trên articleid.
- **add(int tagid, int articleid):** Thêm một liên kết mới giữa tag và bài viết vào cơ sở dữ liệu.

```
public class TagHasArticleService {  
    ▲ Jinkky  
    public static List<Tag_Has_Article> findAll(int id){  
        try (Connection con = DbUtils.getConnection()) {  
            final String query = "select * from tags_has_articles where articleid = :articleid order by tagid";  
  
            return con.createQuery(query)  
                .addParameter( name: "articleid", id)  
                .executeAndFetch(Tag_Has_Article.class);  
        }  
    }  
  
    ▲ Jinkky  
    public static void add(int tagid, int articleid){  
        try (Connection con = DbUtils.getConnection()) {  
            final String query = "INSERT INTO tags_has_articles(tagid, articleid) VALUES (:tagid, :articleid)";  
  
            con.createQuery(query)  
                .addParameter( name: "tagid", tagid)  
                .addParameter( name: "articleid", articleid)  
                .executeUpdate();  
        }  
    }  
}
```

Hình 36 Mã nguồn của TagHasArticleService

## TagService

TagService có nhiệm vụ quản lý các tag của bài viết. Lớp này cung cấp các phương thức sau:

- **findAll():** Truy xuất toàn bộ các tag từ cơ sở dữ liệu và trả về danh sách.
- **findAllWithPaging(int page):** Truy xuất các tag với phân trang, dựa trên trang hiện tại.
- **findByArticle(int articleid):** Truy xuất danh sách các tag liên kết với một bài viết cụ thể dựa trên articleid.
- **findById(int tagid):** Truy xuất một tag cụ thể dựa trên tagid.
- **add(String tagname):** Thêm một tag mới vào cơ sở dữ liệu.
- **update(String tagname, int tagid):** Cập nhật thông tin của một tag.
- **delete(int tagid):** Xóa một tag dựa trên tagid.

```
public class TagService {  
    ▲ Jinkky  
    public static List<Tag> findAll(){  
        try (Connection con = DbUtils.getConnection()) {  
            final String query = "select * from tags order by tagid";  
  
            return con.createQuery(query)  
                .executeAndFetch(Tag.class);  
        }  
    }  
  
    1 usage ▲ Jinkky  
    public static List<Tag> findAllWithPaging(int page){  
        try (Connection con = DbUtils.getConnection()) {  
            int currentOffset = (page-1) * 10;  
            final String query = "select * from tags order by tagid offset :offset limit 10";  
  
            return con.createQuery(query)  
                .addParameter(name: "offset", currentOffset)  
                .executeAndFetch(Tag.class);  
        }  
    }  
  
    4 usages ▲ Jinkky  
    public static List<Tag> findByArticle(int articleid){  
        try (Connection con = DbUtils.getConnection()) {  
            final String query = "select tha.tagid, tagname from tags inner join tags_has_articles tha on tags.tagid = tha.tagid where tha.articleid = :articleid";  
  
            return con.createQuery(query)  
                .addParameter(name: "articleid", articleid)  
                .executeAndFetch(Tag.class);  
        }  
    }  
}
```

Hình 37 Một số phương thức xử lý dữ liệu của TagService

## UserService

UserService có nhiệm vụ quản lý thông tin người dùng. Lớp này cung cấp các phương thức sau:

- **findAll():** Truy xuất toàn bộ người dùng từ cơ sở dữ liệu và trả về danh sách.
- **findAllWithPaging(int page):** Truy xuất các người dùng với phân trang, dựa trên trang hiện tại.
- **findByUsername(String username):** Truy xuất một người dùng dựa trên tên người dùng (username).
- **findById(int id):** Truy xuất một người dùng dựa trên userid.
- **findAllUsernameCommentinArticle(int articleid):** Truy xuất danh sách các người dùng có comment trong một bài viết dựa trên articleid.
- **findUserByEmail(String email):** Truy xuất một người dùng dựa trên địa chỉ email (email).
- **findUserByOTP(String otp):** Truy xuất một người dùng dựa trên mã OTP (otp).
- **add(User u):** Thêm một người dùng mới vào cơ sở dữ liệu.
- **update(User u):** Cập nhật thông tin của một người dùng.
- **updateRole(User u):** Cập nhật vai trò của một người dùng.
- **updateOTP(int id, String otp):** Cập nhật OTP của một người dùng dựa trên userid.
- **updatePassword(int id, String password):** Cập nhật mật khẩu của một người dùng dựa trên userid.
- **deleteOTP(int id):** Xóa OTP của một người dùng dựa trên userid.
- **delete(int id):** Xóa một người dùng dựa trên userid.

```

public class UserService {
    ▲ Jinkky
    public static List<User> findAll(){
        try (Connection con = DbUtils.getConnection()) {
            final String query = "select * from users ORDER BY username";
            return con.createQuery(query)
                .executeAndFetch(User.class);
        }
    }

    1 usage ▲ Jinkky
    public static List<User> findAllWithPaging(int page){
        int currentOffset = (page-1)*10;
        try (Connection con = DbUtils.getConnection()) {
            final String query = "select * from users ORDER BY username offset :offset limit 10";
            return con.createQuery(query)
                .addParameter( name: "offset", currentOffset)
                .executeAndFetch(User.class);
        }
    }

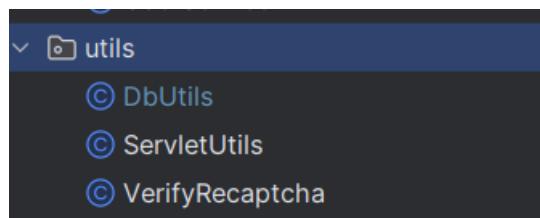
    2 usages ▲ Jinkky
    public static User findByUsername (String username){
        try (Connection con = DbUtils.getConnection()) {
            final String query = "select * from users where username = :username";
            List<User> list = con.createQuery(query)
                .addParameter( name: "username", username)
        }
    }
}

```

Hình 38 Một số phương thức xử lý dữ liệu của UserService

### e. Utils

Là nơi chứa các đoạn code kết nối tới cơ sở dữ liệu, ServletUtils và xử lý mã captcha từ người dùng



Hình 39 Thư mục Utils

Dưới đây là mô tả chức năng của 3 file trên:

#### **DbUtils**

Đoạn mã trong class DbUtils triển khai một utility class để quản lý kết nối đến cơ sở dữ liệu PostgreSQL bằng thư viện SQL2O trong ứng dụng web. Cung cấp phương thức getConnection để lấy đối tượng Connection, giúp tương tác với cơ sở dữ liệu, và sử dụng thông tin kết nối như URL, tên người dùng và mật khẩu được cung cấp tường minh.

```

package com.example.googlenewsclone.utils;
import ...

63 usages  ↳ Jinkky*
public class DbUtils {

    1 usage
    static Sql2o sql2o = new Sql2o(url: "jdbc:postgresql://localhost:5432/GoogleNews", user: "postgres", pass: ██████████);

    56 usages  ↳ Jinkky
    public static Connection getConnection() { return sql2o.open(); }
}

```

Hình 40 Kết nối đến cơ sở dữ liệu trong file DbUtils

## ServletUtils

ServletUtils là một utility class trong ứng dụng web, cung cấp các phương thức tiện ích để thực hiện chuyển hướng và chuyển tiếp trong các Servlet. Dưới đây là tóm tắt ngắn gọn:

- forward Method:
  - Mục Đích: Thực hiện chuyển hướng (forward) request và response đến một địa chỉ URL khác trong ứng dụng.
  - Đối Số:
    - url: Địa chỉ URL cần chuyển hướng đến.
    - request: HttpServletRequest của servlet hiện tại.
    - response: HttpServletResponse của servlet hiện tại.
  - Hoạt Động: Tạo một đối tượng RequestDispatcher với địa chỉ URL cần chuyển hướng và sử dụng phương thức forward để thực hiện chuyển hướng.
- redirect Method:
  - Mục Đích: Thực hiện chuyển tiếp (redirect) request và response đến một địa chỉ URL khác trong ứng dụng.
  - Đối Số:
    - url: Địa chỉ URL cần chuyển tiếp đến.
    - request: HttpServletRequest của servlet hiện tại.
    - response: HttpServletResponse của servlet hiện tại.
  - Hoạt Động: Sử dụng phương thức sendRedirect của HttpServletResponse để thực hiện chuyển tiếp, bổ sung đường dẫn ngữa cảnh của request.

```

84 usages  ± Jinkky
public class ServletUtils {
    47 usages  ± Jinkky
    public static void forward(String url, HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
        RequestDispatcher rd = request.getRequestDispatcher(url);
        rd.forward(request, response);
    }
    30 usages  ± Jinkky
    public static void redirect(String url, HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
        response.sendRedirect( s: request.getContextPath() + url);
    }
}

```

Hình 41 ServletUtils dùng để xử lý các chuyển hướng và chuyển tiếp

## VerifyRecaptcha

VerifyReCaptcha chịu trách nhiệm xác minh reCAPTCHA thông qua Google reCAPTCHA API. Dưới đây là mô tả ngắn gọn của các thành phần chính:

- url: Địa chỉ URL của Google reCAPTCHA API.
- secret: Khóa bí mật sử dụng để xác minh reCAPTCHA, được cung cấp bởi Google.
- USER\_AGENT: Chuỗi xác định User Agent cho request, trong trường hợp này là "Mozilla/5.0".

Phương thức:

- verify:
  - Mục Đích: Xác minh reCAPTCHA thông qua Google reCAPTCHA API.
  - Đối Số:
    - gRecaptchaResponse: Chuỗi token reCAPTCHA nhận được từ giao diện người dùng.
  - Kết Quả: Trả về true nếu xác minh thành công, ngược lại trả về false.
  - Hoạt Động:
    - Tạo URL và kết nối HttpsURLConnection với Google reCAPTCHA API.
    - Thiết lập các thuộc tính của request.
    - Gửi request POST chứa thông tin reCAPTCHA token và khóa bí mật.
    - Nhận và đọc response từ Google reCAPTCHA API.
    - Phân tích và trả về kết quả xác minh từ JSON response.

```

public static boolean verify(String gRecaptchaResponse) throws IOException {
    if (gRecaptchaResponse == null || "".equals(gRecaptchaResponse)) {
        return false;
    }

    try{
        URL obj = new URL(url);
        HttpsURLConnection con = (HttpsURLConnection) obj.openConnection();

        // add request header
        con.setRequestMethod("POST");
        con.setRequestProperty("User-Agent", USER_AGENT);
        con.setRequestProperty("Accept-Language", "en-US,en;q=0.5");

        String postParams = "secret=" + secret + "&response="
            + gRecaptchaResponse;

        // Send post request
        con.setDoOutput(true);
        DataOutputStream wr = new DataOutputStream(con.getOutputStream());
        wr.writeBytes(postParams);
        wr.flush();
        wr.close();

        int responseCode = con.getResponseCode();
        System.out.println("\nSending 'POST' request to URL : " + url);
        System.out.println("Post parameters : " + postParams);
        System.out.println("Response Code : " + responseCode);

        BufferedReader in = new BufferedReader(new InputStreamReader(
            con.getInputStream()));
        String inputLine;
    }
}

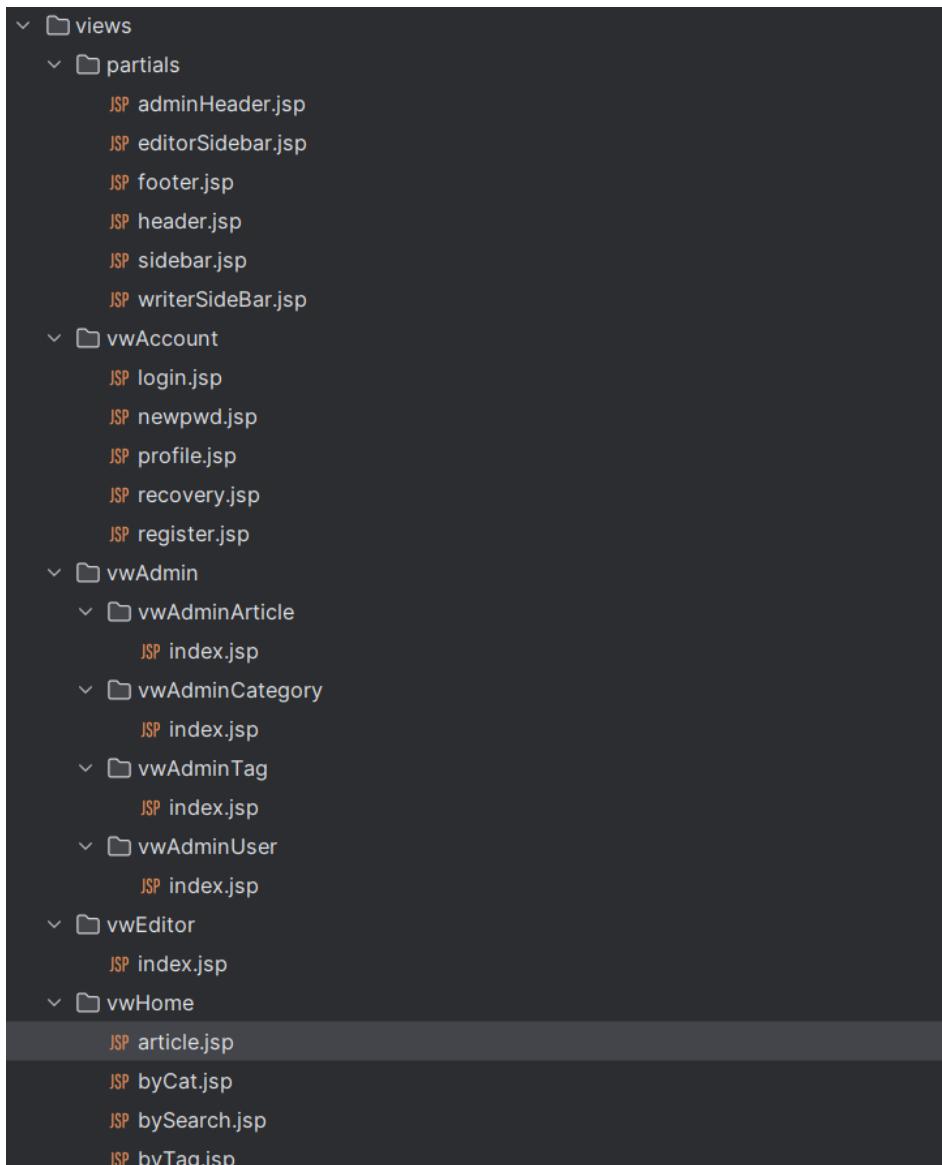
```

Hình 42 Phương thức Verify để xử lý captcha của VerifyRecaptcha

## f. Views

- Chức Năng:
  - Hiển Thị Dữ Liệu từ JavaBeans: View nhận dữ liệu từ JavaBeans thông qua Controller và hiển thị nó dưới dạng giao diện người dùng. Điều này bao gồm hiển thị danh sách, chi tiết, form nhập liệu và các thành phần khác.
  - Tương Tác Người Dùng: Xử lý sự kiện người dùng như nút bấm, form submit, và gửi yêu cầu đến Controller để xử lý.
  - Hiển Thị Lỗi và Thông Báo: Hiển thị thông báo lỗi hoặc thông tin thành công khi thực hiện các thao tác.
- Liên Kết với Controller:
  - Dispatcher: View không trực tiếp truy cập JavaBeans; thay vào đó, nó tương tác với Controller để yêu cầu dữ liệu và cập nhật giao diện.

- Forward và Redirect: View có thể sử dụng các chức năng forward và redirect để điều hướng người dùng đến các trang khác.
- Các Thẻ JSP:
  - Scripting Elements: Sử dụng các thẻ JSP để nhúng mã Java trong trang JSP, giúp xử lý logic phức tạp trực tiếp trong trang.
- Dữ Liệu Động từ JavaBeans:
  - EL (Expression Language): Sử dụng EL để hiển thị dữ liệu động từ JavaBeans trực tiếp trong trang JSP mà không cần sử dụng mã Java phức tạp. EL giúp đơn giản hóa cú pháp và làm cho trang JSP trở nên dễ đọc hơn.

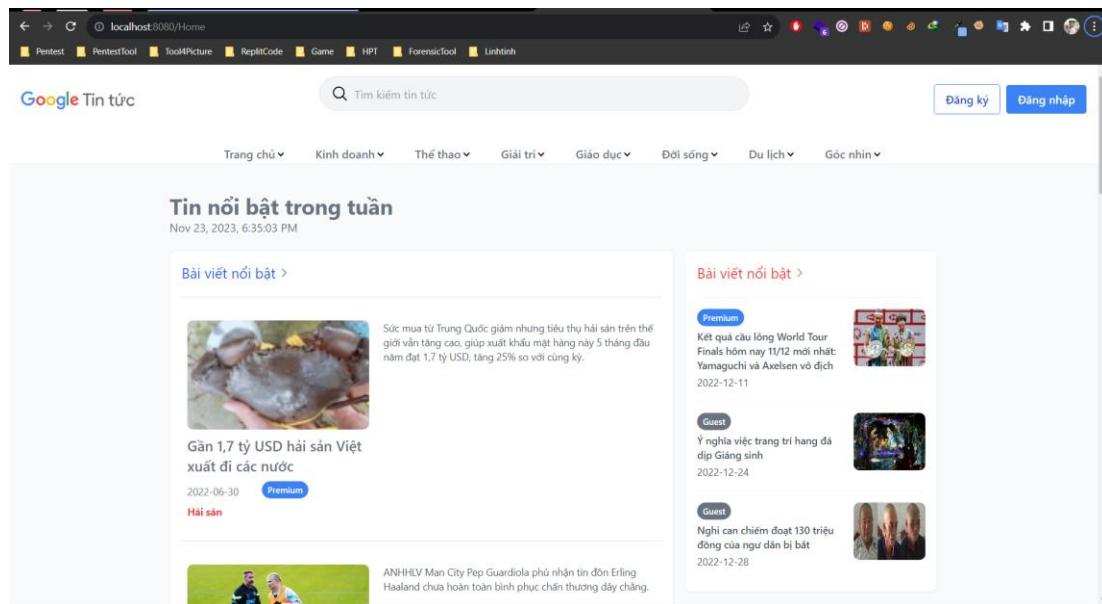


Hình 43 Các file .jsp để xử lý giao diện của người dùng

## 4. Giao diện ứng dụng

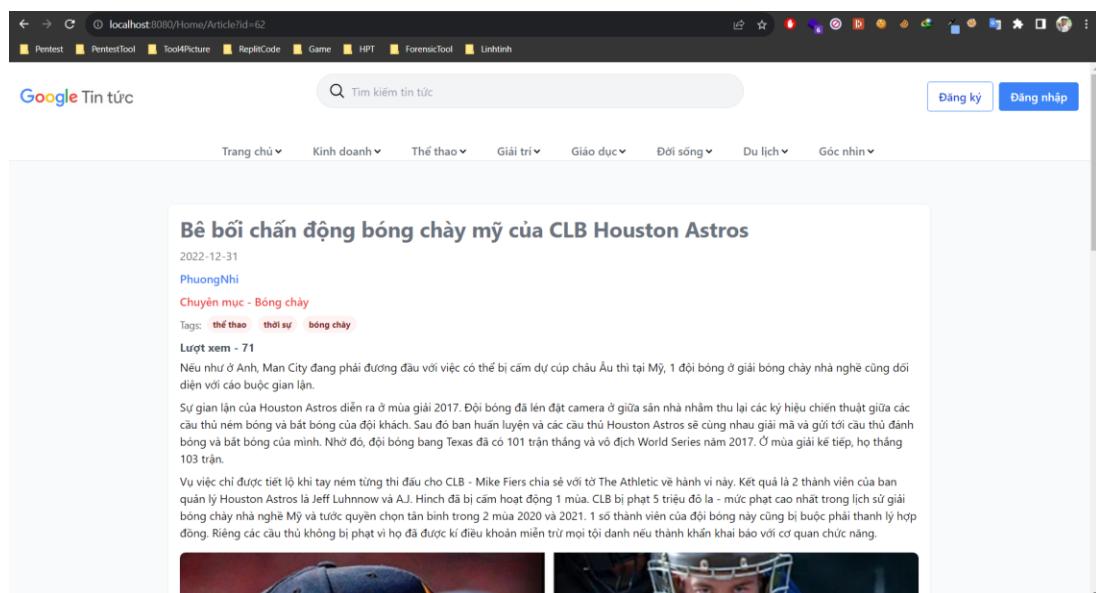
### a. Trang Home

Giao diện khi người dùng truy cập đến trang web:



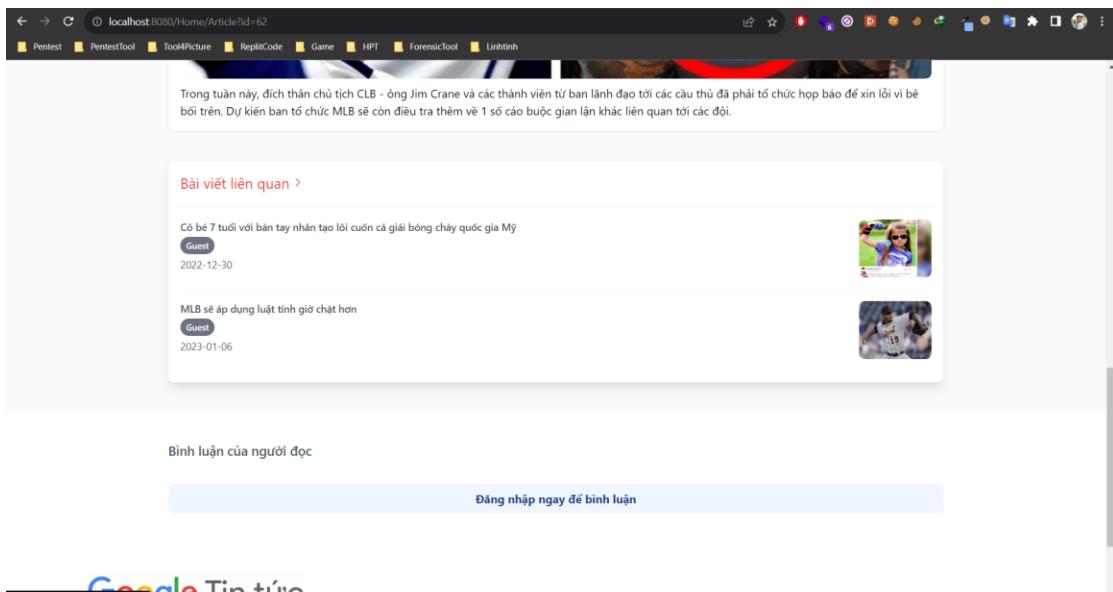
Hình 44 Trang home

Khi người dùng bấm vào một bài báo (không phải premium):



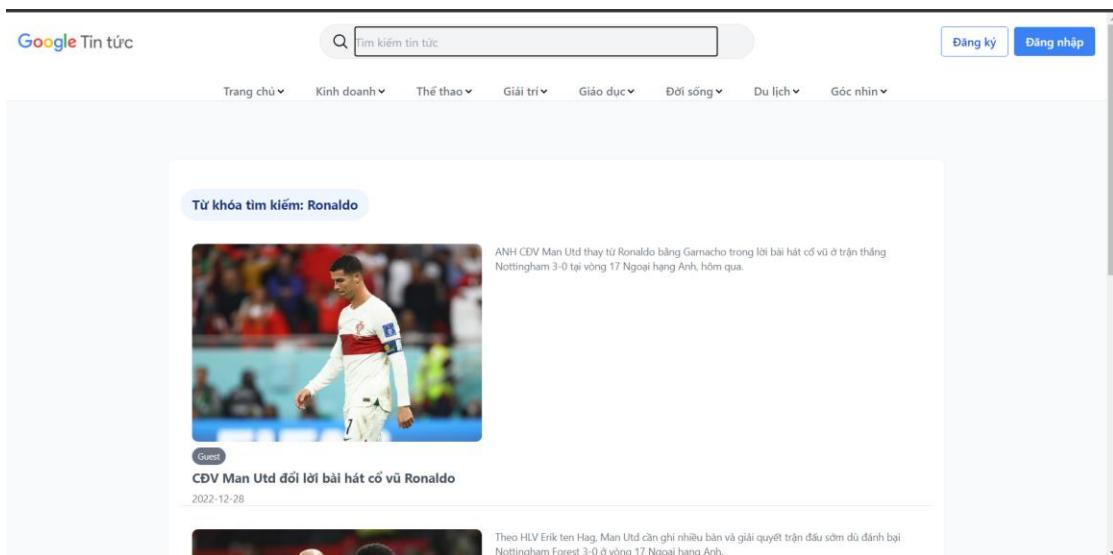
Hình 45 Chi tiết bài báo 1 (chưa đăng nhập)

Cuối trang một bài báo:



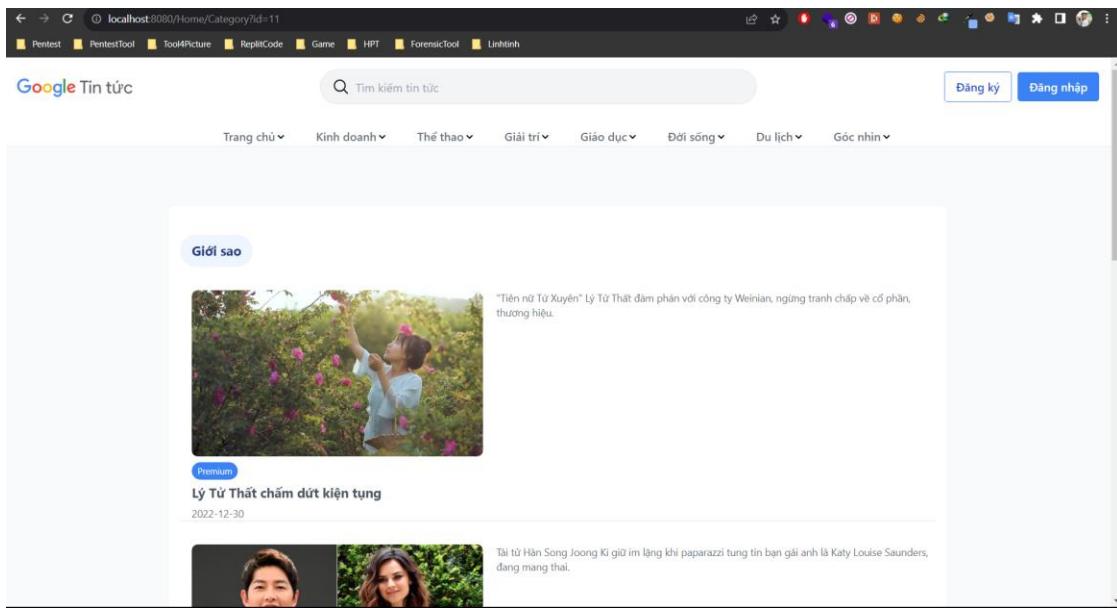
Hình 46 Chi tiết bài báo 2 (chưa đăng nhập)

Khi người dùng tìm kiếm một bài báo:



Hình 47 Trang home khi tìm kiếm từ khóa

Khi người dùng bấm vào một danh mục chính:

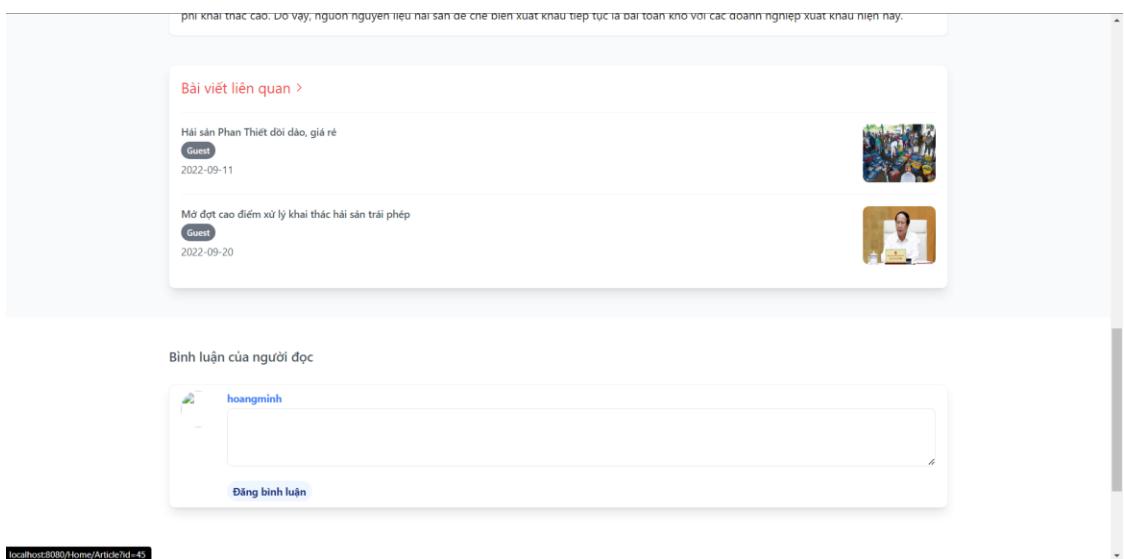


Hình 48 Trang home với danh mục được chọn

Người dùng đăng nhập và xem bài báo Premium:



Hình 49 Chi tiết bài báo 1 (đã đăng nhập)



Hình 50 Chi tiết bài báo 2 (đã đăng nhập)

## b. Trang đăng nhập và đăng ký

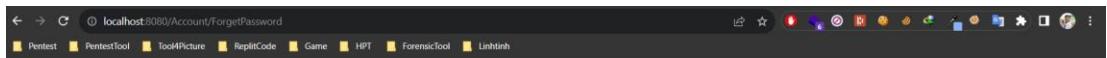
Trang đăng nhập:

The screenshot shows a login page with the following fields and buttons:

- Tài khoản: Input field labeled "Tài khoản của bạn".
- Mật khẩu: Input field labeled "Nhập mật khẩu của bạn".
- Forgot password? (Quên mật khẩu?)
- Login button: "Đăng nhập" (Login).
- Create account button: "Tạo tài khoản" (Create account).

Hình 51 Giao diện đăng nhập

Trang khôi phục mật khẩu:



## Google Tin tức

Khôi phục mật khẩu

Email

Email của bạn

Gửi OTP khôi phục

Chuyển sang đăng nhập

Hình 52 Trang khôi phục mật khẩu

## Trang tạo tài khoản:

Trang đăng ký

Tên  
Tên của bạn

Tài khoản đăng nhập  
Tên tài khoản

Email  
Email của bạn

Mật khẩu  
Nhập mật khẩu của bạn

Xác nhận mật khẩu  
Xác nhận mật khẩu

Bạn là ?  
Người đọc bài  
**Người đọc bài**  
Người viết bài  
 I'm not a robot

Hình 53 Trang tạo tài khoản

## c. Trang của quản trị viên

Đăng nhập vào user admin

localhost:8080/Staff/Admin/User?page=1

TÊN NGƯỜI DÙNG	EMAIL	TÊN ĐĂNG NHẬP	VAI TRÒ	HÀNH ĐỘNG
Nhat	admin	admin	Subscriber	Xóa người dùng
BaoUyen	baouyen	baouyen	Writer	Xóa người dùng
HoangMinh	hoangminh@gmail.com	hoangminh	Editor	Xóa người dùng
Hoang Nam	hoangnam	hoangnam	Subscriber	Xóa người dùng
LamUyen	lamuyen@gmail.com	lamuyen	Editor	Xóa người dùng
MinhKy	minhky	minhky	Writer	Xóa người dùng
NguyetAnh	nguyetanh	nguyetanh	Writer	Xóa người dùng

Hình 54 Trang quản lý người dùng

localhost:8080/Staff/Admin/Article?page=1

ID	TÊN BÀI VIẾT	LUỢT XEM	TRANG THÁI	HÀNH ĐỘNG	XEM
2	HLV Park phát biểu sau khi nhận huân chương ...	15	Đã xuất bản	Xóa bài viết	🔗
5	Hà Nội điều chỉnh linh vực phụ trách của các p...	24	Đã xuất bản	Xóa bài viết	🔗
6	Chủ tịch tỉnh Thanh Hóa bị khiển trách	0	Đã xuất bản	Xóa bài viết	🔗
7	'Alice in Borderland 2': Hội kết của trò chơi sin...	39	Đã xuất bản	Xóa bài viết	🔗
8	'Emily in Paris 3' - tình yêu mù quáng	0	Đã xuất bản	Xóa bài viết	🔗
9	Hà Anh Tuấn: Ông Kitaro yêu thiên nhiên, con ...	30	Đã xuất bản	Xóa bài viết	🔗
11	Ocean M.O.B đưa quan điểm tài chính vào âm ...	0	Đã xuất bản	Xóa bài viết	🔗

Hình 55 Trang quản lý bài viết

ID	TÊN DANH MỤC	DANH MỤC CHA	HÀNH ĐỘNG
1	Tin mới	Trang chủ	<input checked="" type="radio"/> <input type="button" value="Xoá"/>
2	Tin hot	Trang chủ	<input checked="" type="radio"/> <input type="button" value="Xoá"/>
3	Tin vẫn trong ngày	Trang chủ	<input checked="" type="radio"/> <input type="button" value="Xoá"/>
4	Nông sản	Kinh doanh	<input checked="" type="radio"/> <input type="button" value="Xoá"/>
5	Hải sản	Kinh doanh	<input checked="" type="radio"/> <input type="button" value="Xoá"/>
6	Lâm sản	Kinh doanh	<input checked="" type="radio"/> <input type="button" value="Xoá"/>

Hình 56 Trang quản lý danh mục

ID	TÊM TAG	HÀNH ĐỘNG
1	IT	<input checked="" type="radio"/> <input type="button" value="Xoá"/>
2	công	<input checked="" type="radio"/> <input type="button" value="Xoá"/>
3	tiền	<input checked="" type="radio"/> <input type="button" value="Xoá"/>
4	nông	<input checked="" type="radio"/> <input type="button" value="Xoá"/>
5	hang	<input checked="" type="radio"/> <input type="button" value="Xoá"/>
6	thực	<input checked="" type="radio"/> <input type="button" value="Xoá"/>
7	tiêu	<input checked="" type="radio"/> <input type="button" value="Xoá"/>

Hình 57 Trang quản lý tag

#### d. Trang của biên tập viên

Đăng nhập vào tài khoản của một biên tập viên:

ID	Tiêu đề	TRẠNG THÁI	HÀNH ĐỘNG
73	Đại học Bách khoa Hà Nội rút ngắn bài thi đánh giá tư duy	Đã xuất bản	
9	Hà Anh Tuấn: Ông Kitaro yêu thiên nhiên, con người Việt Nam'	Đã xuất bản	
88	Ngủ bao nhiêu là đủ?	Đã xuất bản	
93	Thế giới chưa lập tức ở át đòn du khách Trung Quốc	Đã xuất bản	
113	'Chuyện rồi rầm' không ai lo	Đã xuất bản	
19	Lý Tử Thất chấm dứt kiện tụng	Đã xuất bản	
25	Sống sót hai ngày trên biển nhờ bám vào phao tín hiệu	Đã xuất bản	

Hình 58 Trang quản lý bài viết cá nhân

Đăng bài viết

Loại bài viết  
Premium

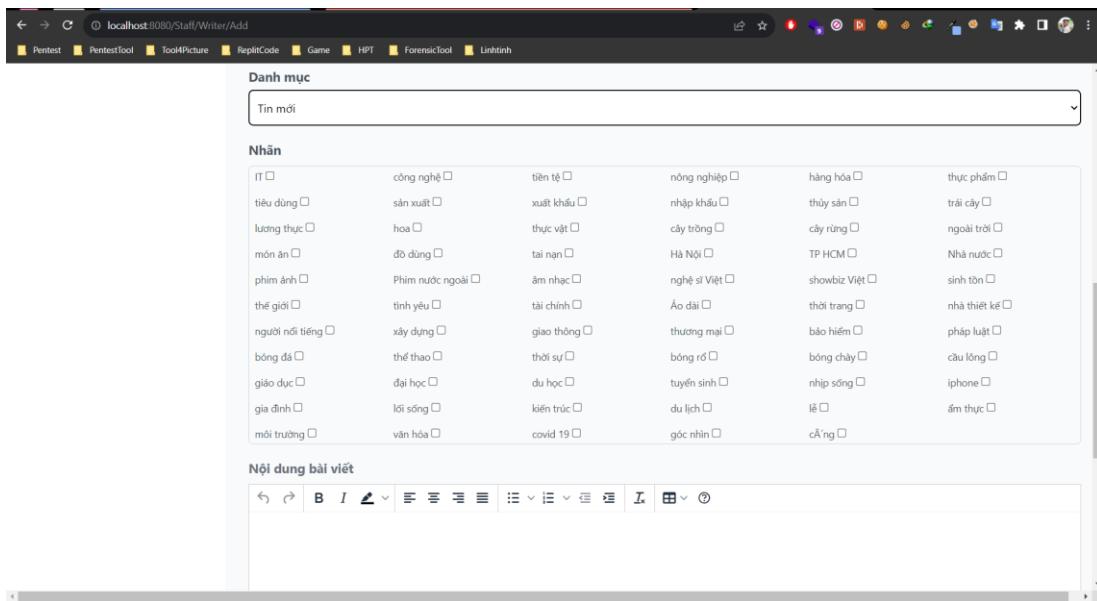
Tiêu đề bài viết

Đường dẫn Thumbnail  
/

Tóm tắt

|

Hình 59 Trang soạn thảo viết báo dành cho biên tập viên (Phần 1)



Hình 60 Trang soạn thảo viết báo dành cho biên tập viên (Phần 2)

### e. Trang của người kiểm duyệt

Đăng nhập vào tài khoản của người kiểm duyệt:

Người kiểm duyệt có thể từ chối hoặc cho phép xuất bản bài báo

ID	TIẾU ĐỀ	TRẠNG THÁI	HÀNH ĐỘNG
121	HLV Shin Tae-yong: 'Indonesia thua vì mặt sân không tốt'	Đang chờ	
122	Cựu hậu vệ Ngoại hạng Anh xin lỗi khi Indonesia thua Việt Nam	Đang chờ	
127	Thủ tướng Phạm Minh Chính sắp thăm Lào	Đang chờ	
128	Hai tân Phó thủ tướng nhận quyết định bổ nhiệm	Đang chờ	

Hình 61 Trang quản lý bài viết của người kiểm duyệt

### **III. Cài đặt và kiểm thử**

#### **1. Cài đặt**

##### **1.1. Tải các tài nguyên cần thiết**

**B1.** Download IDE IntelliJ IDEA Ultimate Edition (trên Windows) theo đường dẫn: <https://www.jetbrains.com/idea/download/thanks.html?platform=windows>

**B2.** Tải JDK 19 theo đường dẫn sau:

[https://download.oracle.com/java/19/archive/jdk-19.0.2\\_windows-x64\\_bin.zip \(sha256\)](https://download.oracle.com/java/19/archive/jdk-19.0.2_windows-x64_bin.zip (sha256)) và giải nén bỏ vào C:\Program Files\

**B3.** Tải Apache Tomcat theo đường dẫn sau:

<https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.83/bin/apache-tomcat-9.0.83-windows-x64.zip> và giải nén trong thư mục chứa trang web

**B4.** Tải PostgreSQL theo đường link sau:

<https://sbp.enterprisedb.com/getfile.jsp?fileid=1258771>

**B5.** Tải Pgadmin4 theo đường link:

<https://www.postgresql.org/ftp/pgadmin/pgadmin4/v8.0/windows/>

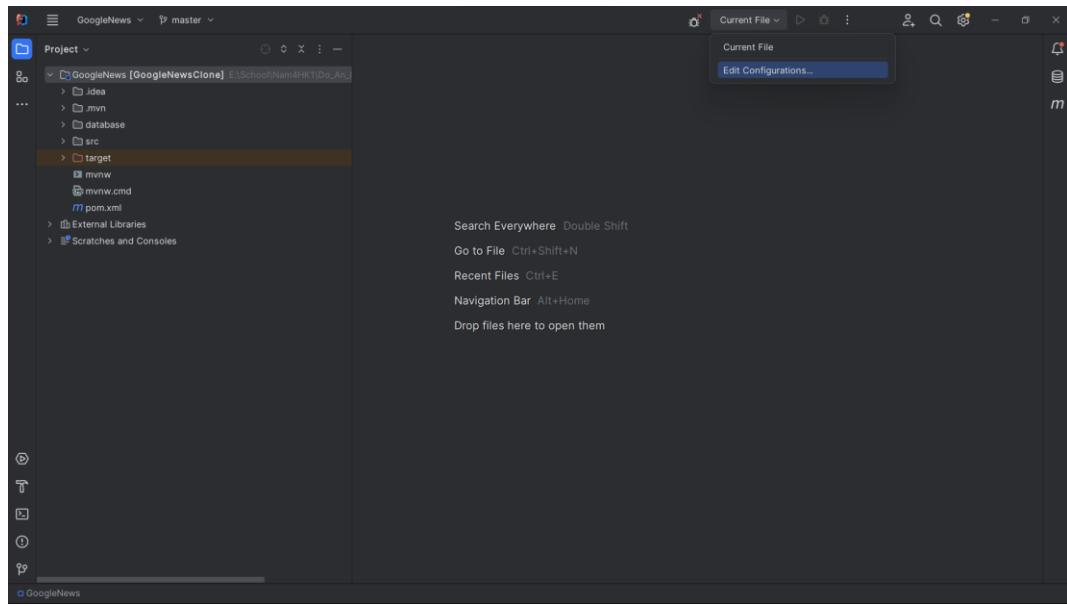
**B6.** Tạo thư mục chứa trang web sau đó mở Terminal ngay tại đó và dùng lệnh: **git clone <https://github.com/Jinkky/GoogleNews.git>**

##### **1.2. Cấu hình để deploy trang web**

**B7.** Mở PgAdmin4 để chạy script tạo cơ sở dữ liệu cho trang web, Tạo một database mới, sau đó chuột phải vào database đã tạo chọn ‘Query Tool’, Import file GoogleNewsClone.sql vào và chạy script để tạo cơ sở dữ liệu

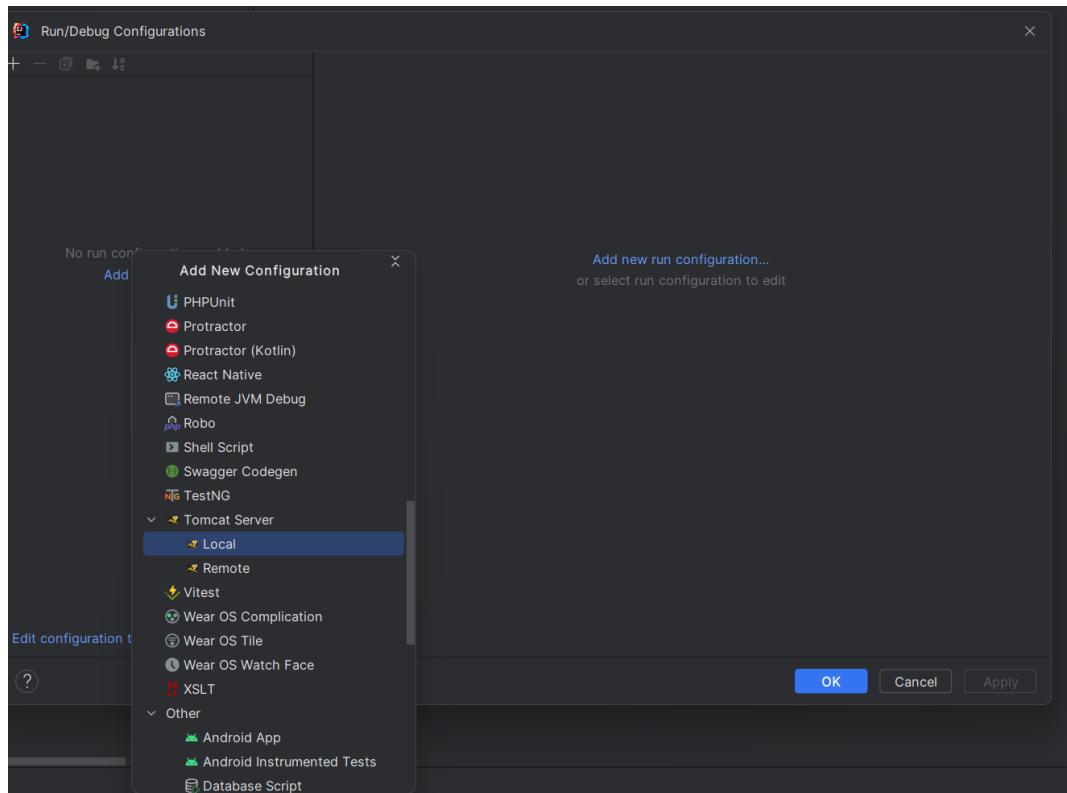
**B8.** Mở IntelliJ và mở thư mục của trang web Làm theo các bước dưới hình ảnh sau:

a. Chọn Edit Configuration



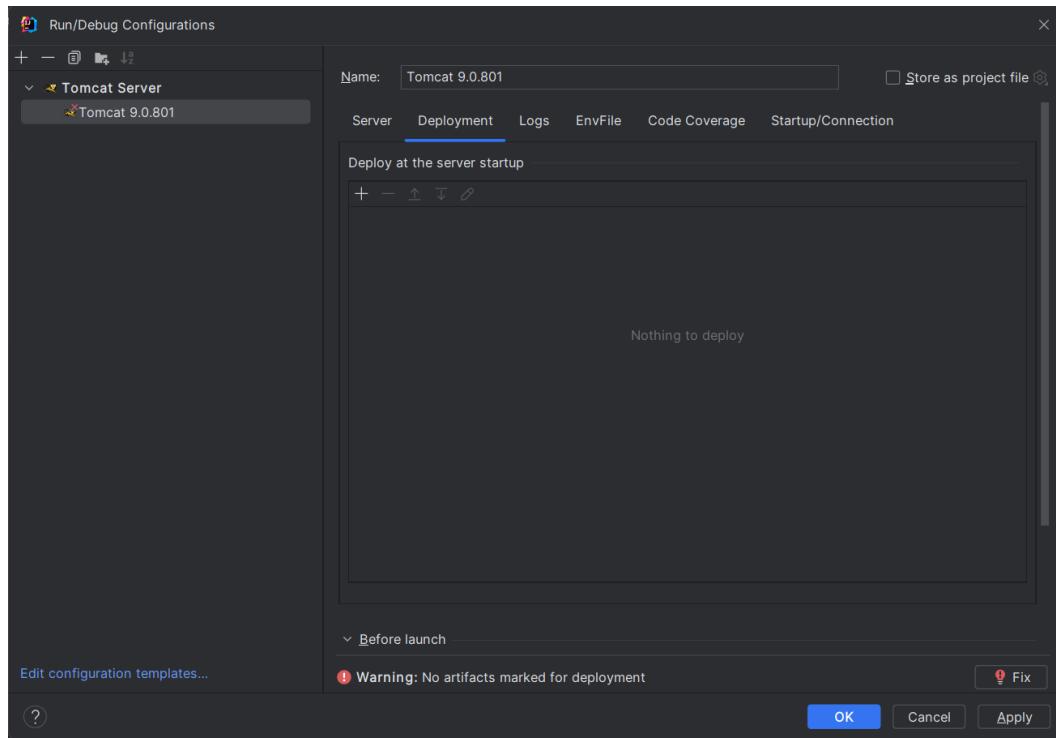
Hình 62 Chỉnh sửa cách chạy dự án trong IntelliJ

b. Ở góc trái chọn Add new, chọn Tomcat Server, Chọn Local



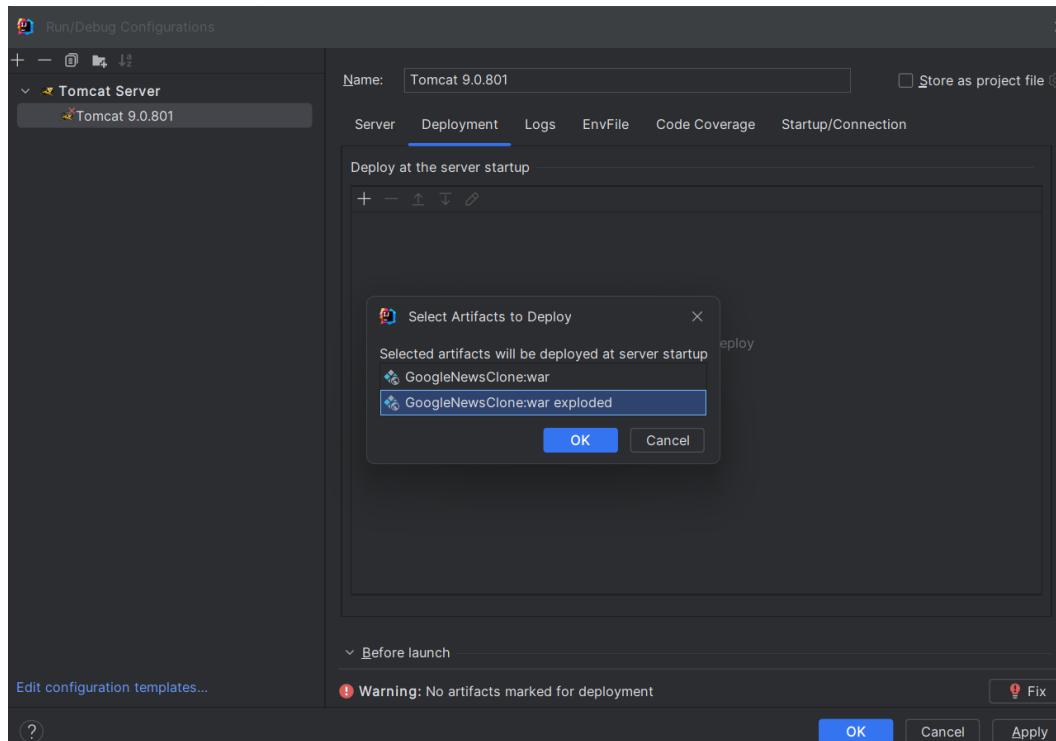
Hình 63 Chọn Tomcat Server làm host

c. Sau đó bấm vào phần Deploy



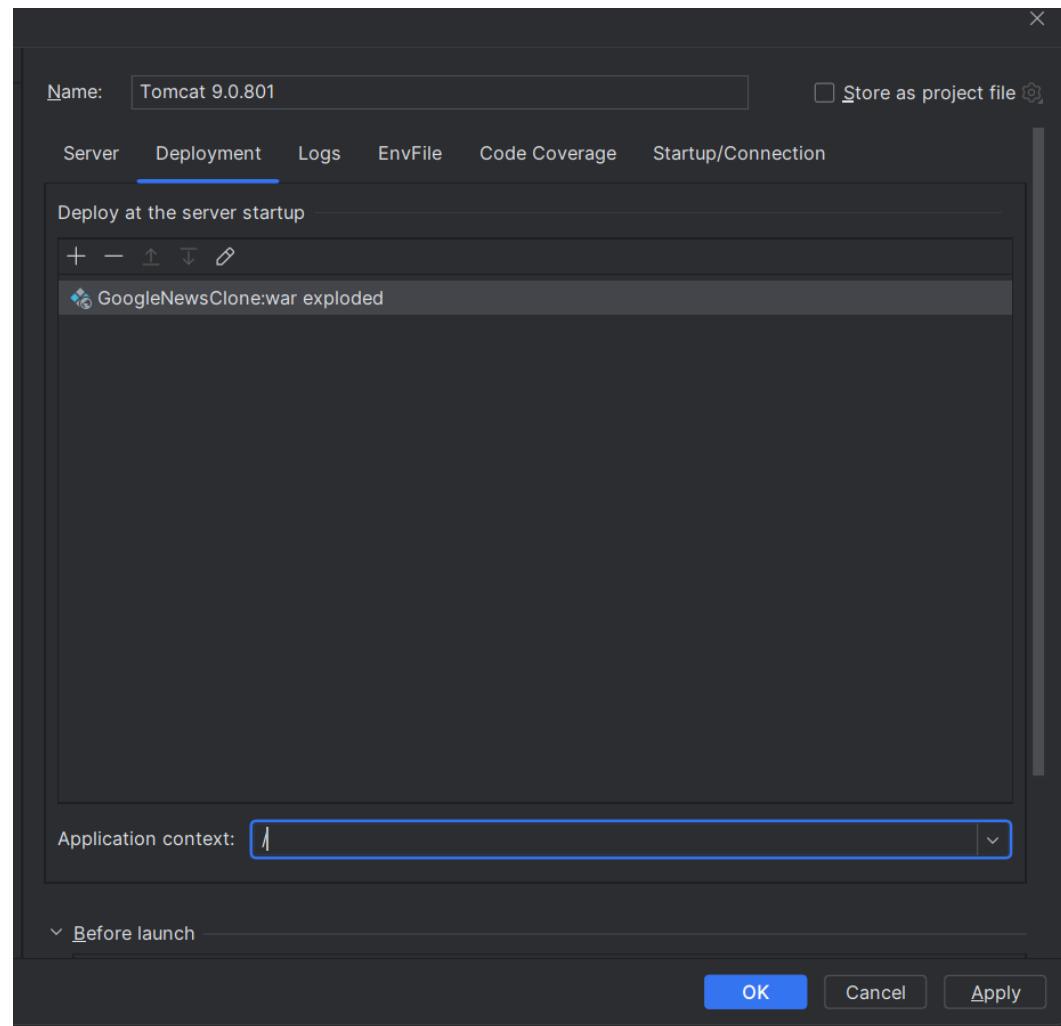
Hình 64 Chọn Cách deploy cho Apache Tomcat

d. Bấm vào dấu +, chọn Artifact, Chọn như hình phía dưới



Hình 65 Chọn Artifact

e. Xóa phần path phía dưới và bấm ok



Hình 66 Chính sửa lại Path trang web

**B9.** Chính sửa thông tin kết nối trong file DbUtils theo cơ sở dữ liệu đã tạo theo user và password theo format sau:

**jdbc:postgresql://localhost:5432/[tên dbs]", "[user tạo dbs]", "[mật khẩu postgresql]"**

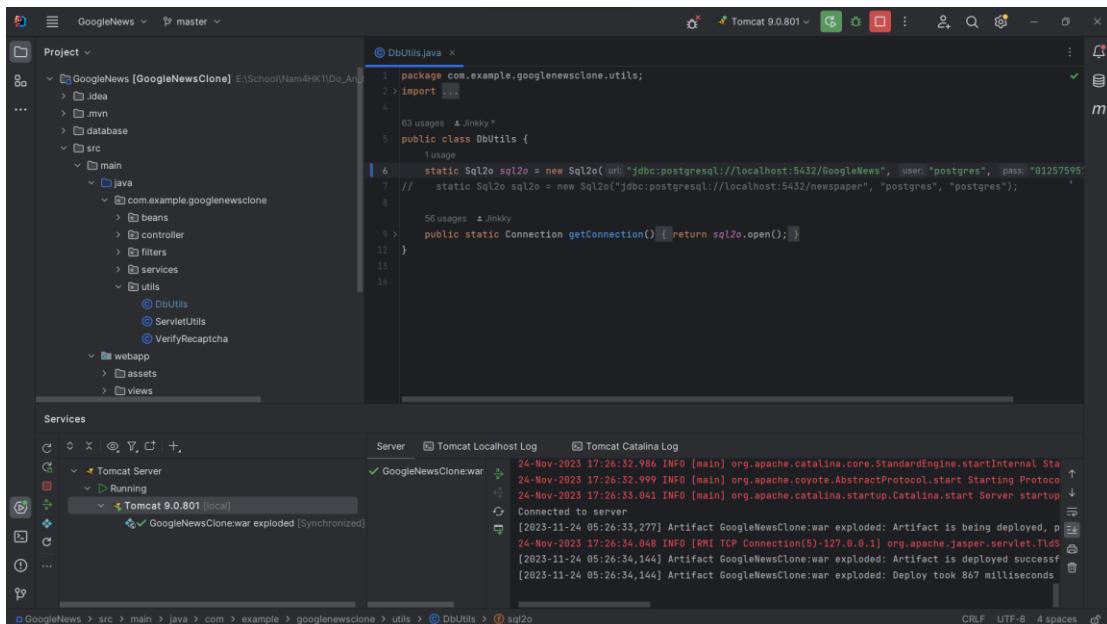
```

package com.example.googlenewclone.utils;
import ...
public class DbUtils {
    static Sql2o sql2o = new Sql2o("jdbc:postgresql://ec2-3-229-161-70.compute-1.amazonaws.com:5432/d86g9mu73tide7");
    public static Connection getConnection() { return sql2o.open(); }
}

```

Hình 67 Chính sửa kết nối cơ sở dữ liệu

**B10.** Bấm nút run phía trên cùng và ta đã deploy thành công trang web:



```

package com.example.googlenewsclone.utils;
import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;

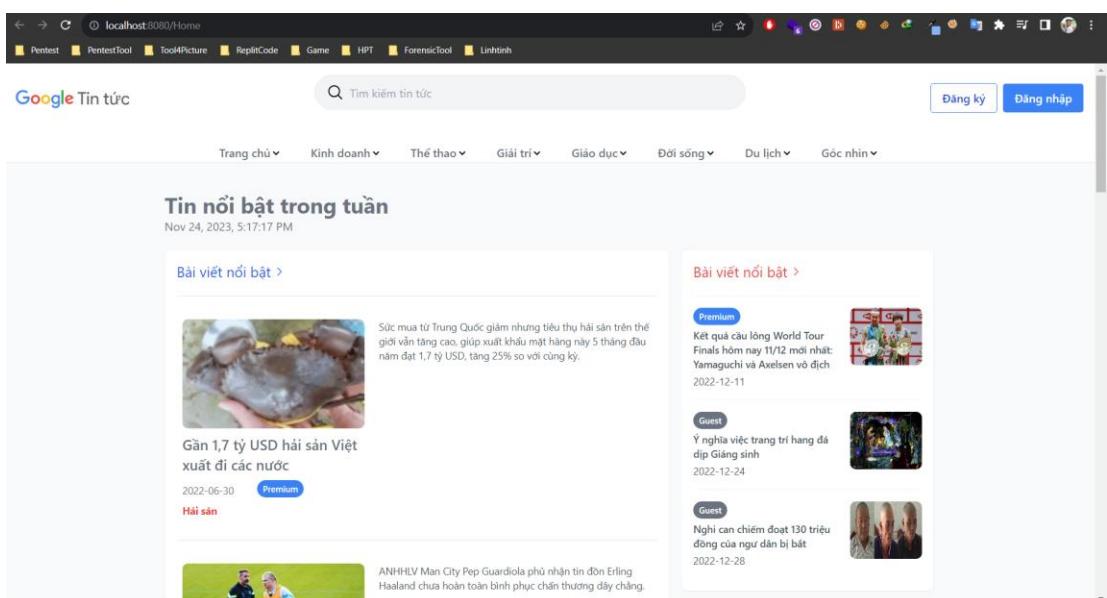
public class DbUtils {
    static HikariConfig config = new HikariConfig();
    static HikariDataSource dataSource;

    static {
        config.setJdbcUrl("jdbc:postgresql://localhost:5432/GoogleNews");
        config.setUsername("postgres");
        config.setPassword("0123456789");
        config.setMaximumPoolSize(10);
        config.setConnectionTestQuery("SELECT 1");
        config.addDataSourceProperty("cachePrepStmts", "true");
        config.addDataSourceProperty("prepStmtCacheSize", "200");
        config.addDataSourceProperty("prepStmtCacheSqlLimit", "2048");
        dataSource = new HikariDataSource(config);
    }

    public static Connection getConnection() {
        return dataSource.getConnection();
    }
}

```

Hình 68 Hiện ra log của Tomcat



Hình 69 Deploy thành công

## 2. Kiểm thử

**Bảng 12 Testcase Đăng ký Tài khoản với Email không hợp lệ**

<b>Test Case Đăng ký tài khoản</b>	
<b>Test Case ID:</b> TCDK_001	<b>Test Designed by:</b> Trần Quốc Siêu
<b>Test Priority (Low/Medium/High):</b> high	<b>Test Designed date:</b> 24/11/2023
<b>Module Name:</b> Đăng ký tài khoản	<b>Test Executed by:</b> Trần Quốc Siêu
<b>Test title:</b> Kiểm tra chức năng đăng ký với email không hợp lệ	<b>Test Executed date:</b> 24/11/2023
<b>Description:</b> Test trang đăng ký	

**Pre-conditions:** người dùng sử dụng email chưa có trong hệ thống

**Dependence:**

Step	Test Steps	Test Data	Excepted Results	Actual Results	Status
1	Nhập Tên	name= “test”			
2	Nhập Tên đăng nhập	Username= “test123”			
3	Nhập Email	Email= “Test123qgmail.com”	Thông tin đã được nhập	Thông tin đã được nhập	Pass
4	Nhập mật khẩu	Password= “123123”			
5	Nhập xác nhận mật khẩu	ConfirmPassword= “123123”			
6	Chọn bạn là ?	Role = “người đọc bài”			
7	Nhấn vào xác nhận captcha	Verfiy Captcha	Verified	Verified	Pass
8	Nhấn nút đăng ký		Hiển thị lỗi: “Please enter an email address”	Hiển thị lỗi: “Please enter an email address”	Pass

**Post-conditions:**

Tài khoản không được tạo thành công do người dùng cung cấp email không hợp lệ

**Bảng 13 Testcase Đăng ký Tài khoản với Email đã tồn tại trong hệ thống**

<b>Test Case Đăng ký tài khoản</b>	
<b>Test Case ID:</b> TCDK_002	<b>Test Designed by:</b> Trần Quốc Siêu
<b>Test Priority (Low/Medium/High):</b> high	<b>Test Designed date:</b> 24/11/2023
<b>Module Name:</b> Đăng ký tài khoản	<b>Test Executed by:</b> Trần Quốc Siêu
<b>Test title:</b> Kiểm tra chức năng đăng ký với email đã tồn tại trong hệ thống	<b>Test Executed date:</b> 24/11/2023
<b>Description:</b> Test trang đăng ký	

**Pre-conditions:** người dùng sử dụng email đã tồn tại trong hệ thống

**Dependence:**

Step	Test Steps	Test Data	Excepted Results	Actual Results	Status
1	Nhập Tên	name= “test”			
2	Nhập Tên đăng nhập	Username= “test123”			
3	Nhập Email	Email= “hoangminh@gmail.com”	Thông tin đã được nhập	Thông tin đã được nhập	Pass
4	Nhập mật khẩu	Password= “123123”			
5	Nhập xác nhận mật khẩu	ConfirmPassword= “123123”			
6	Chọn bạn là ?	Role = “người đọc bài”			
7	Nhấn vào xác nhận captcha	Verfiy Captcha	Verified	Verified	Pass
8	Nhấn nút đăng ký		Hiển thị lỗi: “Email đã tồn tại trong hệ thống”	Hiển thị thông tin: “đăng ký thành công”	Fail

**Post-conditions:**

Tài khoản vẫn được tạo thành công khi người dùng cung cấp email đã tồn tại trong hệ thống

**Bảng 14 Testcase đăng ký tài khoản với username đã tồn tại trong hệ thống**

<b>Test Case Đăng ký tài khoản</b>	
<b>Test Case ID:</b> TCDK_003	<b>Test Designed by:</b> Trần Quốc Siêu
<b>Test Priority (Low/Medium/High):</b> high	<b>Test Designed date:</b> 24/11/2023
<b>Module Name:</b> Đăng ký tài khoản	<b>Test Executed by:</b> Trần Quốc Siêu
<b>Test title:</b> Kiểm tra chức năng đăng ký với username đã tồn tại trong hệ thống	<b>Test Executed date:</b> 24/11/2023
<b>Description:</b> Test trang đăng ký	

**Pre-conditions:** người dùng sử dụng username đã tồn tại trong hệ thống

**Dependence:**

Step	Test Steps	Test Data	Excepted Results	Actual Results	Status
1	Nhập Tên	name= “test”			
2	Nhập Tên đăng nhập	Username= “hoangminh”			
3	Nhập Email	Email= “test@gmail.com”			
4	Nhập mật khẩu	Password= “123123”	Thông tin đã được nhập	Thông tin đã được nhập	Pass
5	Nhập xác nhận mật khẩu	ConfirmPassword= “123123”			
6	Chọn bạn là ?	Role = “người đọc bài”			
7	Nhấn vào xác nhận captcha	Verfiy Captcha	Verified	Verified	Pass
8	Nhấn nút đăng ký		Hiển thị lỗi: “tên người dùng đã tồn tại, vui long chọn tên khác”	Hiển thị lỗi: “tên người dùng đã tồn tại, vui long chọn tên khác”	Pass

**Post-conditions:**

Tài khoản không được tạo khi người dùng cung cấp username đã tồn tại trong hệ thống

**Bảng 15 Testcase nhập 2 trường password và confirmPassword không trùng nhau**

<b>Test Case Đăng ký tài khoản</b>	
<b>Test Case ID:</b> TCDK_004	<b>Test Designed by:</b> Trần Quốc Siêu
<b>Test Priority (Low/Medium/High):</b> high	<b>Test Designed date:</b> 24/11/2023
<b>Module Name:</b> Đăng ký tài khoản	<b>Test Executed by:</b> Trần Quốc Siêu
<b>Test title:</b> Kiểm tra trường xác nhận mật khẩu khi người dùng cung cấp mật khẩu không trùng với trường mật khẩu	<b>Test Executed date:</b> 24/11/2023
<b>Description:</b> Test trang đăng ký	

<b>Pre-conditions:</b> Không <b>Dependence:</b>
--

Step	Test Steps	Test Data	Excepted Results	Actual Results	Status
1	Nhập Tên	name= “test”			
2	Nhập Tên đăng nhập	Username= “test1231”			
3	Nhập Email	Email= “test@gmail.com”			
4	Nhập mật khẩu	Password= “123123”	Thông tin đã được nhập	Thông tin đã được nhập	Pass
5	Nhập xác nhận mật khẩu	ConfirmPassword= “123123123”			
6	Chọn bạn là ?	Role = “người đọc bài”			
7	Nhấn vào xác nhận captcha	Verfiy Captcha	Verified	Verified	Pass
8	Nhấn nút đăng ký		Hiển thị lỗi: “mật khẩu và xác nhận mật khẩu không trùng nhau”	Hiển thị thông tin: “Đăng ký Thành Công!”	Fail

**Post-conditions:**

Tài khoản vẫn được tạo khi người dùng cung cấp password và confirm password khác nhau

## **IV. Danh sách các tài khoản sử dụng**

### **1. Tài khoản Premium**

Username: hoangnam

Password: hoangnam

### **2. Tài khoản admin**

Username: admin

Password: admin

### **3. Tài khoản Người kiểm duyệt**

Username: hoangminh

Password: hoangminh

### **4. Tài khoản biên tập viên**

Username: minhky

Password: minhky