

TRƯỜNG ĐẠI HỌC HỌC VĂN LANG
KHOA CÔNG NGHỆ THÔNG TIN



TÀI LIỆU THỰC HÀNH
SPRING BOOT

**CHƯƠNG 08: XÂY DỰNG CÁC CHỨC NĂNG PHÂN
QUYỀN**

Giảng viên biên soạn: Ths. Nguyễn Minh Tân

2024

BUỔI 8: XÂY DỰNG CÁC CHỨC NĂNG PHÂN QUYỀN

I. MỤC TIÊU

Sau khi học xong bài này, sinh viên có thể nắm được:

- Nắm vững các khái niệm cơ bản về xác thực người dùng (authentication) và phân quyền (authorization).
- Hiểu rõ sự khác biệt giữa xác thực người dùng và phân quyền người dùng, cũng như tầm quan trọng của việc bảo mật thông tin người dùng.
- Biết cách thiết kế và triển khai hệ thống xác thực người dùng, từ việc đăng nhập, đăng ký đến quên mật khẩu.
- Có kỹ năng lập trình để xác thực thông tin người dùng thông qua form và cơ sở dữ liệu.
- Hiểu cách sử dụng sessions và cookies để duy trì trạng thái đăng nhập của người dùng.
- Nắm vững các phương pháp bảo mật khi làm việc với sessions và cookies,...

II. NỘI DUNG THỰC HÀNH:

8.1 Thêm Dependency vào `pom.xml`

- Mở thư mục `pom.xml` trong dự án và tiến hành thêm dependency của **Spring Boot Security** và **Thymeleaf Extras Spring Security** cho Thymeleaf.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
  <groupId>org.thymeleaf.extras</groupId>
  <artifactId>thymeleaf-extras-springsecurity6</artifactId>
</dependency>
```

Lưu ý: Sau khi thêm dependency đảm bảo rằng phải Reload Project

8.2 Bổ sung lớp `User` và `Role`

Tạo các entity `User` và `Role` để đại diện cho người dùng và quyền hạn của họ trong cơ sở dữ liệu.

Tại package model trong đường dẫn **src/main/java/com/vanlang/webbanhang/model** tạo thêm các class **Role**, **User**

8.2.1 Entity class User.java:

```
package com.vanlang.webbanhang.model;

import jakarta.persistence.*;
import jakarta.validation.constraints.*;
import lombok.*;
import org.hibernate.Hibernate;
import org.hibernate.validator.constraints.Length;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.util.Collection;
import java.util.HashSet;
import java.util.Objects;
import java.util.Set;

@Getter
@Setter
@ToString
@RequiredArgsConstructor
@AllArgsConstructor
@Builder
@Entity
@Table(name = "user")
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name = "username", length = 50, unique = true)
    @NotBlank(message = "Username is required")
    @Size(min = 1, max = 50, message = "Username must be between 1 and 50 characters")
    private String username;
    @Column(name = "password", length = 250)
    @NotBlank(message = "Password is required")
    private String password;
    @Column(name = "email", length = 50, unique = true)
    @NotBlank(message = "Email is required")
    @Size(min = 1, max = 50, message = "Email must be between 1 and 50 characters")
    @Email
    private String email;
    @Column(name = "phone", length = 10, unique = true)
    @Length(min = 10, max = 10, message = "Phone must be 10 characters")
    @Pattern(regexp = "[0-9]*$", message = "Phone must be number")
    private String phone;
    @Column(name = "provider", length = 50)
    private String provider;
    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(name = "user_role", joinColumns = @JoinColumn(name = "user_id"), inverseJoinColumns = @JoinColumn(name = "role_id"))
    private Set<Role> roles = new HashSet<>();

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        Set<Role> userRoles = this.getRoles();
        return userRoles.stream().map(role -> new
```

```
SimpleGrantedAuthority(role.getName()))).toList();
    }

    @Override
    public String getPassword() {
        return password;
    }

    @Override
    public String getUsername() {
        return username;
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || Hibernate.getClass(this) != Hibernate.getClass(o))
return false;
        User user = (User) o;
        return getId() != null && Objects.equals(getId(), user.getId());
    }

    @Override
    public int hashCode() {
        return getClass().hashCode();
    }
}
```

8.2.2 Entity class Role.java

```
package com.vanlang.webbanhang.model;

import jakarta.persistence.*;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.Size;
import lombok.*;
import org.hibernate.Hibernate;
```

```
import org.springframework.security.core.GrantedAuthority;

import java.util.HashSet;
import java.util.Objects;
import java.util.Set;

@Getter
@Setter
@ToString
@RequiredArgsConstructor
@AllArgsConstructor
@Builder
@Entity
@Table(name = "role")
public class Role implements GrantedAuthority {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @NotBlank(message = "Name is required")
    @Column(name = "name", length = 50, nullable = false)
    @Size(max = 50, message = "Name must be less than 50 characters")
    private String name;
    @Size(max = 250, message = "Description must be less than 250
characters")
    @Column(name = "description", length = 250)
    private String description;
    @ManyToMany(mappedBy = "roles", cascade = CascadeType.ALL)
    @ToString.Exclude
    private Set<User> users = new HashSet<>();

    @Override
    public String getAuthority() {
        return name;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || Hibernate.getClass(this) != Hibernate.getClass(o))
return false;
        Role role = (Role) o;
        return getId() != null && Objects.equals(getId(), role.getId());
    }

    @Override
    public int hashCode() {
        return getClass().hashCode();
    }
}
```

8.3 Xác định vai trò người dùng sử dụng enum trong java

Trong môi trường phát triển phần mềm, việc quản lý các vai trò người dùng một cách hiệu quả là rất quan trọng để đảm bảo các quyền truy cập phù hợp đến các tài nguyên và chức

năng của ứng dụng. Trường hợp này sử dụng một enum để quản lý các vai trò người dùng trong ứng dụng Java:

Trong đường dẫn `src/main/java/com.vanlang.webbanhang` tạo file **Role.enum**

```
package com.vanlang.webbanhang;

import lombok.AllArgsConstructor;

@AllArgsConstructor
public enum Role {
    ADMIN(1), // Vai trò quản trị viên, có quyền cao nhất trong hệ thống.
    USER(2); // Vai trò người dùng bình thường, có quyền hạn giới hạn.
    public final long value; // Biến này lưu giá trị số tương ứng với mỗi vai trò.
}
```

Ở đây, `'Role.enum'` định nghĩa hai vai trò: **ADMIN** và **USER**, mỗi vai trò được gán một giá trị số duy nhất. **ADMIN** được gán giá trị **1** và **USER** được gán giá trị **2**. Mỗi vai trò này sau đó có thể được sử dụng để kiểm soát quyền truy cập trong các tình huống khác nhau trong ứng dụng.

8.4 Cấu hình bảo mật sử dụng Spring Security

Lớp cấu hình này bao gồm các điều chỉnh cho phép kiểm soát truy cập, xác thực và quản lý phiên người dùng trong ứng dụng web. Bao gồm cấu hình cho trang đăng nhập, đăng xuất, quản lý phiên, và trang xử lý các lỗi truy cập bị từ chối.

Tạo một class cấu hình `'SecurityConfig.java'` tại đường dẫn `src/main/java/com.vanlang.webbanhang`:

```
package com.vanlang.webbanhang;

import com.vanlang.webbanhang.service.UserService;
import jakarta.validation.constraints.NotNull;
import lombok.RequiredArgsConstructor;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
```

```
import org.springframework.security.web.SecurityFilterChain;

@Configuration // Đánh dấu lớp này là một lớp cấu hình cho Spring Context.
@EnableWebSecurity // Kích hoạt tính năng bảo mật web của Spring Security.
@RequiredArgsConstructor // Lombok tự động tạo constructor có tham số cho tất cả các trường final.
public class SecurityConfig {
    private final UserService userService; // Tiêm UserService vào lớp cấu hình này.

    @Bean // Đánh dấu phương thức trả về một bean được quản lý bởi Spring Context.
    public UserDetailsService userDetailsService() {
        return new UserService(); // Cung cấp dịch vụ xử lý chi tiết người dùng.
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder(); // Bean mã hóa mật khẩu sử dụng BCrypt.
    }

    @Bean
    public DaoAuthenticationProvider authenticationProvider() {
        var auth = new DaoAuthenticationProvider(); // Tạo nhà cung cấp xác thực.
        auth.setUserDetailsService(userDetailsService()); // Thiết lập dịch vụ chi tiết người dùng.
        auth.setPasswordEncoder(passwordEncoder()); // Thiết lập cơ chế mã hóa mật khẩu.
        return auth; // Trả về nhà cung cấp xác thực.
    }

    @Bean
    public SecurityFilterChain securityFilterChain(@NotNull HttpSecurity http) throws Exception {
        return http
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/css/**", "/js/**", "/",
                    "/oauth/**", "/register", "/error", "/products", "/cart", "/cart/**")
                .permitAll() // Cho phép truy cập không cần xác thực.

                .requestMatchers("/products/edit/**",
                    "/products/add", "/products/delete")
                .hasAnyAuthority("ADMIN") // Chỉ cho phép ADMIN truy cập.

                .requestMatchers("/api/**")
                .permitAll() // API mở cho mọi người dùng.
                .anyRequest().authenticated() // Bất kỳ yêu cầu nào khác cần xác thực.
            )
            .logout(logout -> logout
                .logoutUrl("/logout")
                .logoutSuccessUrl("/login") // Trang chuyển hướng sau khi đăng xuất.

                .deleteCookies("JSESSIONID") // Xóa cookie.
            )
    }
}
```

```
        .invalidateHttpSession(true) // Hủy phiên làm việc.
        .clearAuthentication(true) // Xóa xác thực.
        .permitAll()
    )
    .formLogin(formLogin -> formLogin
        .loginPage("/login") // Trang đăng nhập.
        .loginProcessingUrl("/login") // URL xử lý đăng
nhập.
        .defaultSuccessUrl("/") // Trang sau đăng nhập thành
công.
        .failureUrl("/login?error") // Trang đăng nhập thất
bại.
        .permitAll()
    )
    .rememberMe(rememberMe -> rememberMe
        .key("vanlang")
        .rememberMeCookieName("vanlang")
        .tokenValiditySeconds(24 * 60 * 60) // Thời gian nhớ
đăng nhập.
        .userDetailsService(userDetailsService())
    )
    .exceptionHandling(exceptionHandling -> exceptionHandling
        .accessDeniedPage("/403") // Trang báo lỗi khi truy
cập không được phép.
    )
    .sessionManagement(sessionManagement -> sessionManagement
        .maximumSessions(1) // Giới hạn số phiên đăng nhập.
        .expiredUrl("/login") // Trang khi phiên hết hạn.
    )
    .httpBasic(httpBasic -> httpBasic
        .realmName("vanlang") // Tên miền cho xác thực cơ
bản.
    )
    .build(); // Xây dựng và trả về chuỗi lọc bảo mật.
}
```

Phân phân quyền trong code trên được xử lý chủ yếu thông qua cấu hình **HttpSecurity** trong phương thức **securityFilterChain**. Đây là một cách tiếp cận tập trung cho phép bạn định nghĩa các quy tắc truy cập đến các tài nguyên khác nhau trong ứng dụng của bạn dựa trên vai trò của người dùng. Dưới đây là giải thích chi tiết về từng phần trong cấu hình này:

Xác thực người dùng và phân quyền:

```
.authorizeHttpRequests(auth -> auth...
```

Phương thức này bắt đầu một chuỗi các cấu hình xác thực và phân quyền cho các yêu cầu HTTP.

```
.requestMatchers("/css/**", "/js/**", "/", "/oauth/**", "/register", "/error")
.permitAll()
```


Khoa Công nghệ Thông tin

Cấu hình này cho phép tất cả người dùng truy cập vào các tài nguyên cơ bản như CSS, JS, trang chủ, và các trang đăng ký hay báo lỗi mà không cần xác thực.

```
.requestMatchers("/product/edit/**", "/product/add", "/product/delete")  
.hasAnyAuthority("ADMIN")
```

Chỉ người dùng có vai trò "ADMIN" mới có quyền truy cập vào các đường dẫn liên quan đến chỉnh sửa, thêm, hoặc xóa sản phẩm.

```
.requestMatchers("/product", "/cart", "/cart/**")  
.hasAnyAuthority("ADMIN", "USER")
```

Người dùng có vai trò "ADMIN" hoặc "USER" đều có thể truy cập vào các đường dẫn liên quan đến sản phẩm và giỏ hàng.

```
.requestMatchers("/api/**")  
.permitAll()
```

Tất cả người dùng đều có thể truy cập vào các đường dẫn API mà không cần xác thực.

Cấu hình đăng nhập và đăng xuất:

```
.formLogin(...)
```

Định nghĩa cách thức đăng nhập bằng form, bao gồm trang đăng nhập, URL xử lý đăng nhập, và trang chuyển hướng sau khi đăng nhập thành công hoặc thất bại.

```
.logout(...)
```

Định nghĩa cách thức đăng xuất, bao gồm URL xử lý đăng xuất, trang chuyển hướng sau khi đăng xuất, và các cài đặt về cookie và phiên làm việc.

Quản lý phiên và nhớ đăng nhập:

```
.sessionManagement(...)
```

Cấu hình quản lý phiên, chẳng hạn như số lượng phiên tối đa cho phép từ một tài khoản.

```
.rememberMe(...)
```

Cài đặt cho tính năng nhớ đăng nhập, giúp người dùng không cần đăng nhập lại trong một khoảng thời gian nhất định.

Xử lý ngoại lệ:

```
.exceptionHandling(...)
```

Cấu hình trang xử lý khi truy cập bị từ chối (khi người dùng không có quyền truy cập vào tài nguyên).

Mỗi phần của cấu hình này đóng vai trò quan trọng trong việc bảo vệ ứng dụng và đảm bảo rằng mỗi người dùng chỉ có thể truy cập vào các tài nguyên phù hợp với vai trò của họ. Cấu hình phân quyền như vậy giúp tăng cường an toàn và bảo mật cho ứng dụng của bạn.

8.5 Xây dựng Repository

Repository trong Spring Boot là các interface giúp tương tác với cơ sở dữ liệu một cách dễ dàng, sử dụng Spring Data JPA để tự động hóa các thao tác CRUD và truy vấn. Dưới đây là hướng dẫn chi tiết về cách tạo các repository cần thiết:

Trong thư mục theo đường dẫn `src/main/java/com.vanlang.webbanhang/repository` tạo các file sau: `UserRepository.java`, `IUserRepository.java`, `IRoleRepository.java`

8.5.1 UserRepository.java

File này chứa interface **UserRepository** để quản lý các thao tác liên quan đến đối tượng **User**. **UserRepository** mở rộng **JpaRepository**, cho phép thực hiện các thao tác cơ bản và cả một số truy vấn tùy chỉnh:

```
package com.vanlang.webbanhang.repository;

import com.vanlang.webbanhang.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
}
```

Phương thức `findByUsername` sẽ truy vấn cơ sở dữ liệu để tìm một người dùng dựa trên tên đăng nhập của họ.

8.5.2 IUserRepository.java

Để minh họa việc quản lý các phiên bản khác nhau của repository cho cùng một mô hình, chúng ta có **IUserRepository**. Đây là một phiên bản khác của repository cho **User**, có thể được sử dụng trong các trường hợp khác nhau:

```
package com.vanlang.webbanhang.repository;

import com.vanlang.webbanhang.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
```

```
import java.util.Optional;

@Repository
public interface IUserRepository extends JpaRepository<User, String> {
    Optional<User> findByUsername(String username);
}
```

Phương thức `findByUsername` ở đây trả về `Optional<User>`, giúp xử lý trường hợp không tìm thấy người dùng một cách an toàn hơn.

8.5.3 IRoleRepository.java

Interface **IRoleRepository** quản lý các thao tác với đối tượng Role. Nó cũng mở rộng từ `JpaRepository` và bao gồm các phương thức cụ thể cho việc quản lý vai trò:

```
package com.vanlang.webbanhang.repository;

import com.vanlang.webbanhang.model.Role;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface IRoleRepository extends JpaRepository<Role, Long> {
    Role findRoleById(Long id);
}
```

Phương thức `findRoleById` trả về một **Role** dựa trên **ID**, hỗ trợ việc truy xuất thông tin chi tiết của vai trò.

8.6 Xây dựng class UserService

Class **UserService** đóng vai trò quan trọng trong việc tổ chức và quản lý các hoạt động liên quan đến người dùng và vai trò của họ trong hệ thống. Điều này đảm bảo tính nhất quán và hiệu quả trong việc quản lý người dùng và vai trò của họ.

Tiếp theo, tạo class **UserService.java** được đặt tại đường dẫn sau
src/main/java/com.vanlang.webbanhang/service.

Mã nguồn mẫu `UserService.java`:

```
package com.vanlang.webbanhang.service;

import com.vanlang.webbanhang.Role;
import com.vanlang.webbanhang.model.User;
import com.vanlang.webbanhang.repository.IRoleRepository;
import com.vanlang.webbanhang.repository.IUserRepository;
import jakarta.validation.constraints.NotNull;
import lombok.extern.slf4j.Slf4j;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.Optional;

@Service
@Slf4j
@Transactional
public class UserService implements UserDetailsService {
    @Autowired
    private IUserRepository userRepository;
    @Autowired
    private IRoleRepository roleRepository; // Lưu người dùng mới vào cơ sở
    dữ liệu sau khi mã hóa mật khẩu.

    public void save(@NotNull User user) {
        user.setPassword(new
BCryptPasswordEncoder().encode(user.getPassword()));
        userRepository.save(user);
    }

    // Gán vai trò mặc định cho người dùng dựa trên tên người dùng.
    public void setDefaultRole(String username) {
        userRepository.findByUsername(username).ifPresentOrElse(user -> {
user.getRoles().add(roleRepository.findRoleById(Role.USER.value));
        userRepository.save(user);
    }, () -> {
        throw new UsernameNotFoundException("User not found");
    });
    } // Tải thông tin chi tiết người dùng để xác thực.

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        var user = userRepository.findByUsername(username).orElseThrow(() ->
new UsernameNotFoundException("User not found"));
        return
org.springframework.security.core.userdetails.User.withUsername(user.getUser
name()).password(user.getPassword()).authorities(user.getAuthorities()).acco
untExpired(!user.isAccountNonExpired()).accountLocked(!user.isAccountNonLock
ed()).credentialsExpired(!user.isCredentialsNonExpired()).disabled(!user.isE
nabled()).build();
    }

    // Tìm kiếm người dùng dựa trên tên đăng nhập.
    public Optional<User> findByUsername(String username) throws
UsernameNotFoundException {
        return userRepository.findByUsername(username);
    }
}
```

Giải thích mã nguồn:

- **Autowired:** Tự động tiêm **IUserRepository** và **IRoleRepository** để tương tác với cơ sở dữ liệu.
- **Mã hóa mật khẩu:** Sử dụng **BCryptPasswordEncoder** để mã hóa mật khẩu trước khi lưu vào cơ sở dữ liệu.
- **Quản lý vai trò:** Phương thức **setDefaultRole** gán vai trò mặc định cho người dùng khi đăng ký.
- **Xác thực người dùng:** Phương thức **loadUserByUsername** cung cấp cơ chế xác thực người dùng thông qua Spring Security.

8.7 Xây dựng UserController

UserController xử lý các yêu cầu từ người dùng cuối liên quan đến hoạt động của người dùng, chẳng hạn như đăng nhập và đăng ký. **Controller** này sử dụng **UserService** để thực hiện các nghiệp vụ liên quan đến người dùng và vai trò của họ trong hệ thống.

Trong đường dẫn **src/main/java/com.vanlang.webbanhang/controller** tạo file **`UserController.java`**:

```
package com.vanlang.webbanhang.controller;

import com.vanlang.webbanhang.model.User;
import com.vanlang.webbanhang.service.UserService;
import jakarta.validation.Valid;
import jakarta.validation.constraints.NotNull;
import lombok.RequiredArgsConstructor;
import org.springframework.context.support.DefaultMessageSourceResolvable;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;

@Controller // Đánh dấu lớp này là một Controller trong Spring MVC.
@RequestMapping("/")
@RequiredArgsConstructor
public class UserController {
    private final UserService userService;

    @GetMapping("/login")
    public String login() {
        return "users/login";
    }

    @GetMapping("/register")
    public String register(@NotNull Model model) {
        model.addAttribute("user", new User()); // Thêm một đối tượng User
        // mới vào model
    }
}
```

```
        return "users/register";
    }

    @PostMapping("/register")
    public String register(@Valid @ModelAttribute("user") User user, //
        Validate đối tượng User
        @NotNull BindingResult bindingResult, // Kết quả
        của quá trình validate
        Model model) {
        if (bindingResult.hasErrors()) { // Kiểm tra nếu có lỗi validate
            var errors =
                bindingResult.getAllErrors().stream().map(DefaultMessageSourceResolvable::ge
                    tDefaultMessage).toArray(String[]::new);
            model.addAttribute("errors", errors);
            return "users/register"; // Trả về lại view "register" nếu có
            lỗi
        }
        userService.save(user); // Lưu người dùng vào cơ sở dữ liệu
        userService.setDefaultRole(user.getUsername()); // Gán vai trò mặc
        định cho người dùng
        return "redirect:/login"; // Chuyển hướng người dùng tới trang
        "login"
    }
}
```

5.8 Phát triển các trang đăng nhập và đăng ký

Xây dựng các trang HTML/Thymeleaf cho việc đăng nhập và đăng ký.

Đầu tiên tạo thư mục **users** tại đường dẫn **src/main/resources/templates**.

Tạo 2 view **`login.html`** và **`register.html`** đặt tại thư mục **src/main/resources/templates/users**

8.8.1 File **`register.html`**

Trang đăng ký cho phép người dùng mới nhập thông tin cá nhân và tạo một tài khoản trong hệ thống. Trang này sử dụng Bootstrap để đảm bảo tính thẩm mỹ và phù hợp với các thiết bị khác nhau, đồng thời tích hợp Thymeleaf để xử lý các tương tác phía server như xác thực dữ liệu và hiển thị thông báo lỗi.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
    xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
        layout:decorate="~{layout}">
<head><title>Register</title>
    <link rel="stylesheet"
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.cs
        s">
</head>
<body>
<section layout:fragment="content" class="container mt-5"><h1 class="mb-
4">Login</h1>
    <form th:action="@{/register}" method="post">
```

```
<div th:if="{errors}" class="alert alert-danger justify-content-center" role="alert">
    <ul>
        <li th:each="error : {errors}" th:text="{error}"
class="text-danger text-start"></li>
    </ul>
</div>
<div class="form-group mb-4">
    <label for="email"></label>
    <input type="email" class="form-control" id="email" name="email"
placeholder="Enter your email">
</div>
<div class="form-group mb-4">
    <label for="username"></label>
    <input type="text" class="form-control" id="username"
name="username"
placeholder="Enter your username">
</div>
<div class="form-group mb-4">
    <label for="password"></label>
    <input type="password" class="form-control" id="password"
name="password"
placeholder="Enter your password">
</div>
<div class="form-group mb-4">
    <label for="phone"></label>
    <input type="tel" class="form-control" id="phone" name="phone"
placeholder="Enter your phone">
</div>
<div class="d-grid gap-2 form-action">
    <button type="submit" class="btn btn-primary btn-lg btn-block">Sign up</button>
    <p class="mt-3 mb-0">Already have an account? <a class="text-info text-center" th:href="{ '/login' }">Login
in?</a>
    </p>
</div>
</form>
</section>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

8.8.2 File `login.html`

Trang đăng nhập cho phép người dùng truy cập bằng tên người dùng và mật khẩu đã đăng ký. Trang này cũng hỗ trợ hiển thị các thông báo liên quan đến quá trình đăng nhập, như thông báo lỗi khi nhập sai thông tin hoặc xác nhận đã đăng xuất thành công.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
```

```
        layout:decorate=~{layout}">
<head><title>Login</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.cs
s">
</head>
<body>
<section layout:fragment="content" class="container mt-5"><h1 class="mb-
4">Login</h1>
    <form th:action="@{/login}" method="post">
        <div th:if="{param.error}" class="alert alert-danger"> Invalid
username and password.</div>
        <div th:if="{param.logout}" class="alert alert-success"> You have
been logged out.</div>
        <div class="form-group mb-4">
            <label for="username"></label>
            <input type="text" class="form-control" required id="username"
name="username"
                placeholder="Username"></div>
        <div class="form-group mb-4">
            <label for="password"></label>
            <input type="password" class="form-control" required
id="password" name="password"
                placeholder="Password"></div>
        <div class="form-check d-flex justify-content-start mb-4">
            <input type="checkbox" class="form-check-input" name="remember-
me" id="remember-me">
            <label
                class="form-check-label" for="remember-me"> Remember me
            </label>
        </div>
        <div class="d-grid gap-2 form-action">
            <button type="submit" class="btn btn-primary btn-lg btn-
block">Login</button>
            <p class="mt-3 mb-0">Don't have an account? <a class="text-info
text-center" th:href="{'/register'}">Sign
                up?</a>
            </p>
        </div>
    </form>
</section>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.m
in.js"></script>
</body>
</html>
```

8.8.3 Cập nhật `layout.html`

Để hỗ trợ việc đăng nhập và đăng xuất, **layout.html** được cập nhật với các nút **Login** và **Logout** trên thanh điều hướng, cùng với các thông báo khi đăng nhập thành công. Sử dụng **Spring Security**, trang này sẽ thích ứng hiển thị dựa trên trạng thái xác thực của người dùng, tăng cường trải nghiệm người dùng và bảo mật cho ứng dụng.


```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
xmlns:sec="http://www.thymeleaf.org/extras/spring-security5"
xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head>
    <meta charset="UTF-8">
    <title>Layout</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.cs
s">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-primary">
    <div class="container-fluid"><a class="navbar-brand text-white"
href="/products">Web Bán Hàng</a>
    <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarSupportedContent"
    aria-controls="navbarSupportedContent" aria-expanded="false"
aria-label="Toggle navigation"><span
    class="navbar-toggler-icon"></span></button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
            <li class="nav-item"><a class="nav-link text-white active"
aria-current="page"
                                href="/products">Product</a></li>
            <li class="nav-item"
sec:authorize="hasAnyAuthority('ADMIN') "><a th:href="@{/products/add}"
class="nav-link text-white">Add New
                Product</a></li>
            <li class="nav-item"
sec:authorize="hasAnyAuthority('ADMIN') "><a th:href="@{/categories/add}"
class="nav-link text-white">Add New
                Category</a></li>
        </ul>
        <ul class="navbar-nav ms-auto mb-2 mb-lg-0 align-items-center">
            <li class="nav-item d-flex align-items-center"
sec:authorize="isAuthenticated() "><span
                class="navbar-text text-white"> Xin chào, <span
sec:authentication="name"
style="margin-right: 20px;"></span> </span></li>
            <li class="nav-item" sec:authorize="isAuthenticated() ">
                <form th:action="@{/logout}" method="post">
                    <button class="btn btn-outline-light"
type="submit">Logout</button>
                </form>
            </li>
            <li class="nav-item" sec:authorize="!isAuthenticated() "><a
class="btn btn-outline-light" href="/login">Login</a>
            </li>
        </ul>
    </div>
</div>
</nav>
<div class="container mt-5">
```

```
<section layout:fragment="content"> <!-- Nội dung cụ thể của từng trang  
sẽ được đặt tại đây --> </section>  
</div>  
<script  
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.m  
in.js"></script>  
</body>  
</html>
```

Đoạn mã trên sử dụng Thymeleaf và Spring Security, tích hợp một hệ thống phân quyền để quản lý hiển thị các phần của giao diện người dùng dựa trên các quyền của người dùng. Đây là một cách tiếp cận hiệu quả để bảo vệ và cá nhân hóa nội dung của ứng dụng web.

Giải thích mã nguồn:

a) Phân Quyền Quản Trị (Admin)

```
</li>  
<li class="nav-item" sec:authorize="hasAnyAuthority('ADMIN')">  
  <a th:href="@{/products/add}" class="nav-link text-white">Add New Product</a>  
</li>  
<li class="nav-item" sec:authorize="hasAnyAuthority('ADMIN')">  
  <a th:href="@{/categories/add}" class="nav-link text-white">Add New  
  Category</a>  
</li>
```

Sử dụng thuộc tính `sec:authorize="hasAnyAuthority('ADMIN')"` để kiểm tra nếu người dùng hiện tại có quyền 'ADMIN'. Chỉ những người dùng có quyền này mới có nhìn thấy và truy cập vào các liên kết để thêm sản phẩm mới và thêm danh mục mới trên thanh điều hướng. Điều này đảm bảo rằng chỉ những người có thẩm quyền quản lý mới có thể thực hiện các thao tác này, giúp bảo vệ tính toàn vẹn của ứng dụng.

b) Xác Thực Người Dùng

```
<li class="nav-item d-flex align-items-center"  
sec:authorize="isAuthenticated()">  
  <span class="navbar-text text-white">  
    Xin chào, <span sec:authentication="name" style="margin-right:  
20px;"></span>  
  </span>  
</li>  
<li class="nav-item" sec:authorize="isAuthenticated()">  
  <form th:action="@{/logout}" method="post">  
    <button class="btn btn-outline-light" type="submit">Logout</button>  
  </form>  
</li>
```

`sec:authorize="isAuthenticated()"` được sử dụng để kiểm tra nếu người dùng đã đăng nhập. Khi người dùng đã xác thực, họ sẽ thấy tên của mình trên thanh điều hướng cùng với một nút đăng xuất. Điều này cung cấp trải nghiệm người dùng liền mạch và cá nhân hóa, tăng cường tính năng bảo mật và sự thuận tiện.

```
<li class="nav-item" sec:authorize="!isAuthenticated()">
  <a class="btn btn-outline-light" href="/login">Login</a>
</li>
```

`sec:authorize="!isAuthenticated()"` cho phép hiển thị nút đăng nhập cho những người dùng chưa đăng nhập. Điều này khuyến khích người dùng tham gia và tương tác với các tính năng của trang web, đồng thời duy trì tính bảo mật bằng cách không hiển thị các tùy chọn nhạy cảm.

Ý Nghĩa và Lợi Ích

Sử dụng Thymeleaf và Spring Security để quản lý quyền và xác thực người dùng không chỉ giúp tăng cường bảo mật cho ứng dụng web bằng cách hạn chế truy cập vào các chức năng nhạy cảm mà còn cải thiện trải nghiệm người dùng bằng cách cung cấp giao diện đáp ứng tốt với trạng thái xác thực của họ. Điều này làm cho trang web trở nên thân thiện và dễ sử dụng hơn, đồng thời giữ cho thông tin người dùng an toàn và bảo mật.

8.8.4 Cập nhật file `products-list.html`

Cập nhật file **products-list.html** để thêm chức năng quản lý sản phẩm cho người dùng có quyền quản trị (**ADMIN**) và tính năng thêm sản phẩm vào giỏ hàng cho tất cả người dùng đã xác thực. Đây là bước cần thiết để đảm bảo tính bảo mật và hiệu quả của ứng dụng thương mại điện tử.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:sec="http://www.thymeleaf.org/extras/spring-security5"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{layout}">
<head>
  <title th:text="${title} ?: 'Products List'">Products List</title>
  <link rel="stylesheet"
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body>
<section layout:fragment="content"><h1>Products List</h1>
```

```
<table class="table table-bordered table-hover">
  <thead class="table-dark">
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Price</th>
      <th>Description</th>
      <th>Category Name</th>
      <th>Actions</th>
      <th>Add To Cart</th>
    </tr>
  </thead>
  <tbody>
    <tr th:each="product : ${products}">
      <td th:text="${product.id}"></td>
      <td th:text="${product.name}"></td>
      <td th:text="${product.price}"></td>
      <td th:text="${product.description}"></td>
      <td th:text="${product.category.name}"></td>
      <td> <!-- Hiển thị nút sửa và xóa chỉ dành cho người dùng ADMIN -->
        <div sec:authorize="hasAuthority('ADMIN')"><a
th:href="@{/products/edit/{id} (id=${product.id})}"
                                class="btn
btn-success btn-sm">Sửa</a>
          <a th:href="@{/products/delete/{id} (id=${product.id})}"
class="btn btn-danger btn-sm"
          onclick="return confirm('Bạn có chắc
không?') ">Xóa</a>
        </div>
      </td>
      <td> <!-- Nút thêm vào giỏ hàng, hiển thị cho tất cả người dùng
đã xác thực -->
        <form th:action="@{/cart/add}" method="post"
sec:authorize="isAuthenticated()">
          <input type="number" name="quantity"
min="1" value="1"
          class="form-control d-inline-block"
          style="width: 70px;">
          <input type="hidden" th:value="${product.id}"
name="productId"/>
          <button type="submit" class="btn btn-warning btn-
sm">Thêm Vào Giỏ</button>
        </form>
      </td>
    </tr>
  </tbody>
</table>
</section>
</body>
</html>
```

Giải thích Các Thay Đổi:

Quyền Quản Trị (ADMIN): Các nút "Edit" và "Delete" chỉ được hiển thị cho người dùng có quyền ADMIN. Điều này giúp đảm bảo rằng chỉ người dùng được phép mới có thể chỉnh

sửa hoặc xóa sản phẩm, bảo vệ thông tin sản phẩm khỏi các thay đổi không mong muốn hoặc truy cập trái phép.

Thêm vào Giỏ Hàng: Chức năng này được hiển thị cho mọi người dùng đã xác thực, cho phép họ thêm sản phẩm vào giỏ hàng của mình, hỗ trợ các hoạt động mua sắm trên trang web.

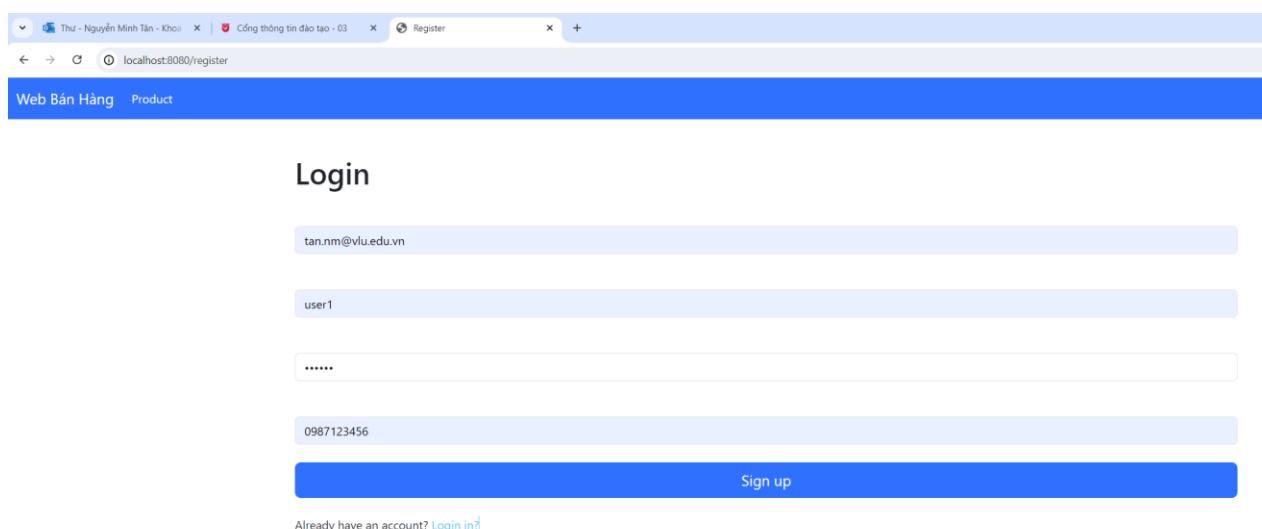
Qua cách tiếp cận này, bạn không chỉ cải thiện tính năng bảo mật của trang web mà còn tăng cường khả năng tương tác người dùng, tạo trải nghiệm mua sắm dễ dàng và an toàn hơn.

8.9 Tiến hành build và kiểm tra kết quả

Tiến hành xây dựng, kiểm tra và đánh giá các chức năng quan trọng của ứng dụng web, bao gồm việc đăng ký, đăng nhập, và phân quyền cho người dùng. Dưới đây là một hướng dẫn chi tiết về cách thực hiện từng bước:

8.9.1 Trang đăng ký

Người dùng cần truy cập vào URL `http://localhost:8080/register` để mở trang đăng ký. Tại đây, người dùng có thể điền thông tin cần thiết để tạo một tài khoản mới. Sau khi điền đầy đủ thông tin và gửi biểu mẫu, tài khoản người dùng mới sẽ được tạo trong hệ thống.



tan.nm@vlu.edu.vn

user1

.....

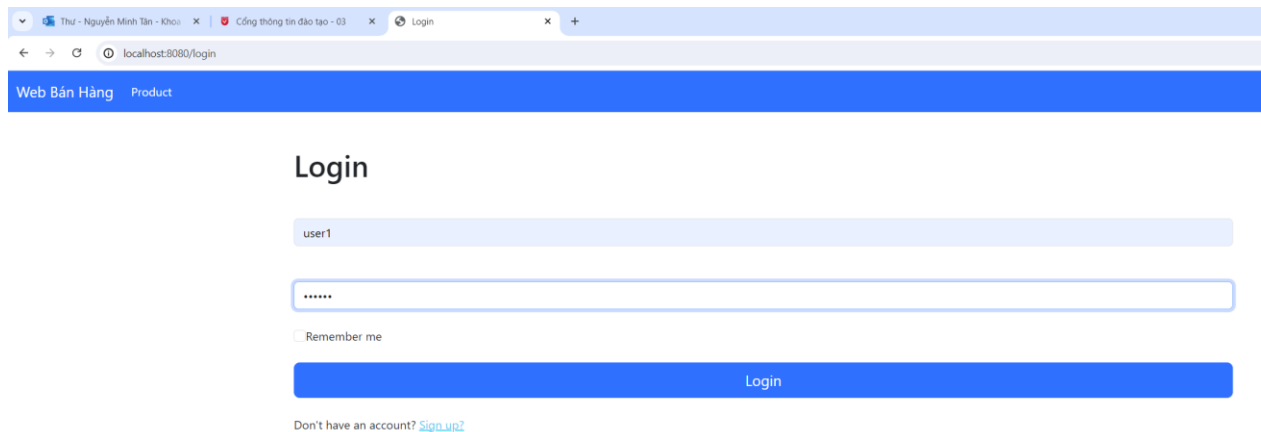
0987123456

Sign up

Already have an account? [Login in?](#)

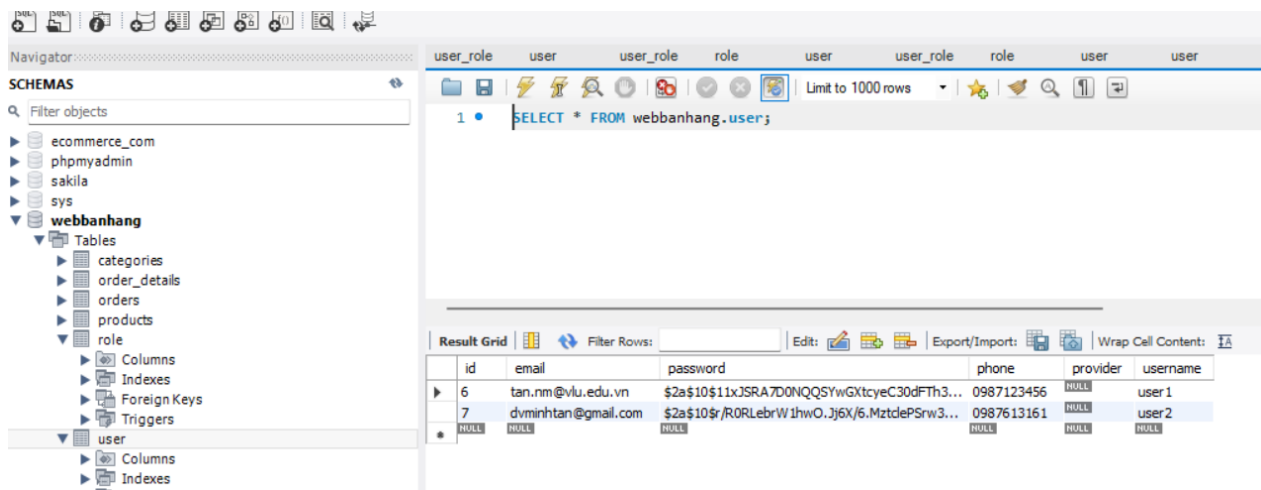
8.9.2 Trang đăng nhập

Sau khi tài khoản đã được tạo, người dùng có thể truy cập vào trang đăng nhập **http://localhost:8080/login** để kiểm tra khả năng đăng nhập sử dụng tài khoản vừa đăng ký.



8.9.3 Kiểm tra Database

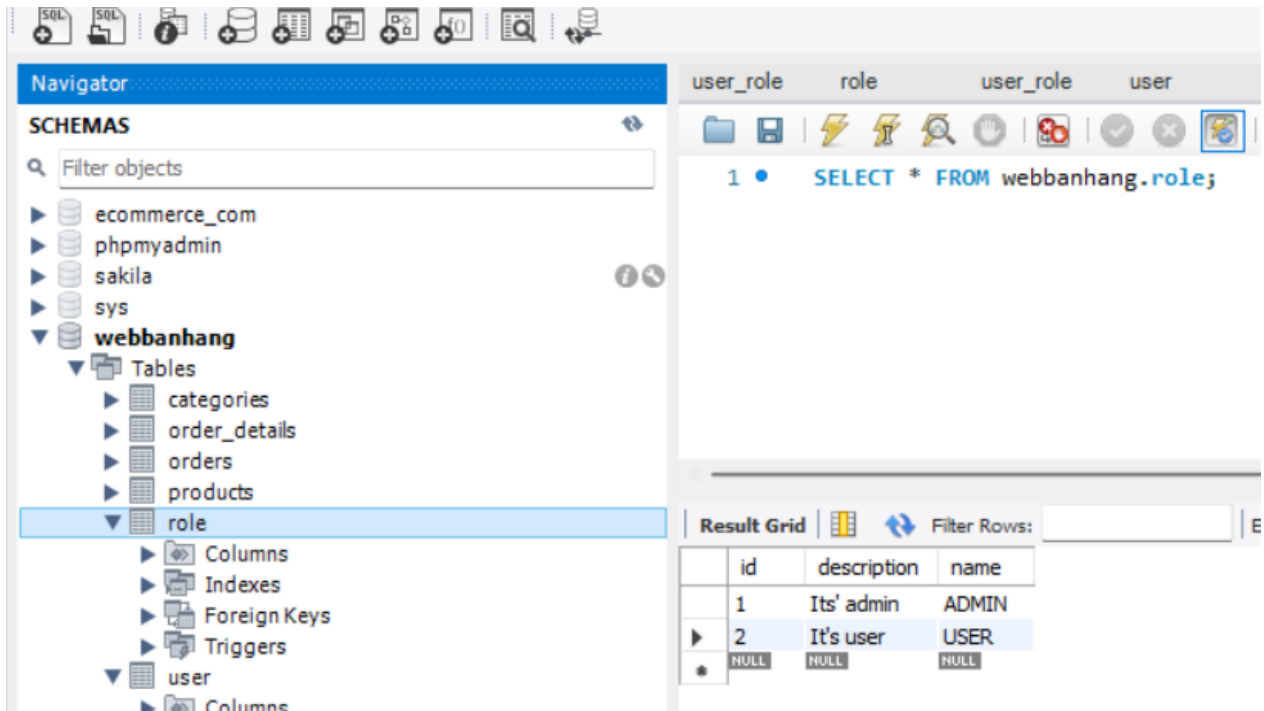
Sau khi người dùng đăng ký, kiểm tra trong cơ sở dữ liệu, đặc biệt là bảng “user”, để đảm bảo rằng thông tin người dùng đã được lưu trữ chính xác. Điều này quan trọng để xác nhận rằng hệ thống đang hoạt động chính xác và không có lỗi xảy ra trong quá trình đăng ký.



| id | email | password | phone | provider | username |
|----|---------------------|--|------------|----------|----------|
| 6 | tan.nm@vlu.edu.vn | \$2a\$10\$11xJSRA7D0NQSYwGXtceC30dFTh3... | 0987123456 | NULL | user 1 |
| 7 | dvminhtan@gmail.com | \$2a\$10\$/R0RLebrW1hwO.Jj6X/6.MztdePSrw3... | 0987613161 | NULL | user 2 |

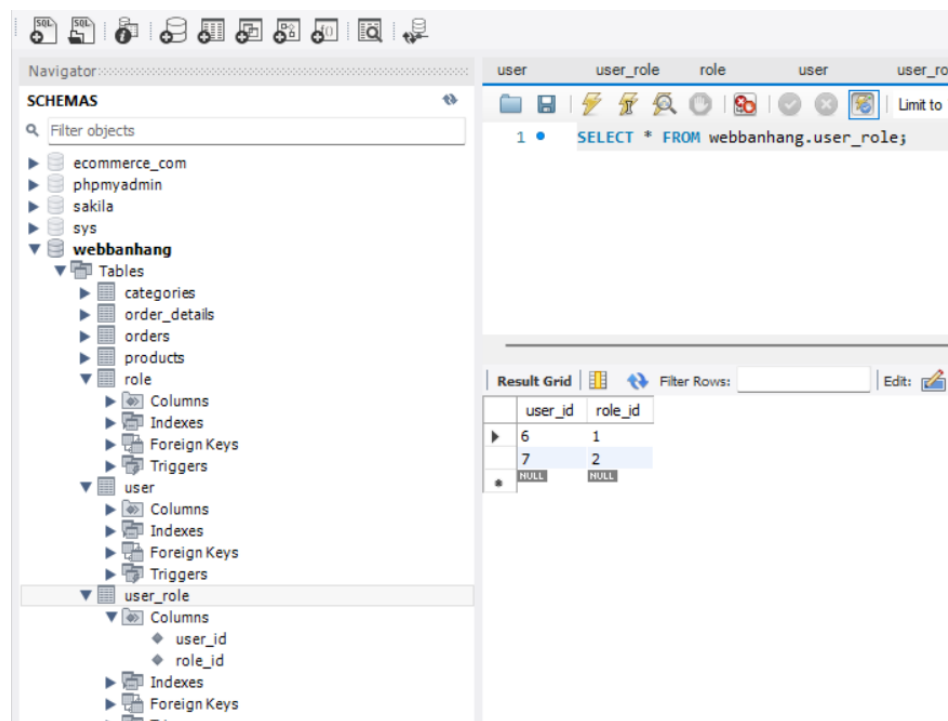
8.9.4 Tiến hành tạo role để phân quyền cho các tài khoản:

Truy cập vào database và đi tới bảng `role`. Tại đây, thêm 2 role là ADMIN và USER để xác thực phân quyền cho người dùng.



8.9.5 Tiến hành phân quyền cho các tài khoản:

Truy cập vào database và đi tới bảng `user_role`. Tại đây, thêm hoặc sửa đổi các mục để phân quyền cho người dùng. Chẳng hạn, bạn có thể phân quyền 'ADMIN' cho 'user1':



User_id = 6 : Tương ứng với `user1`

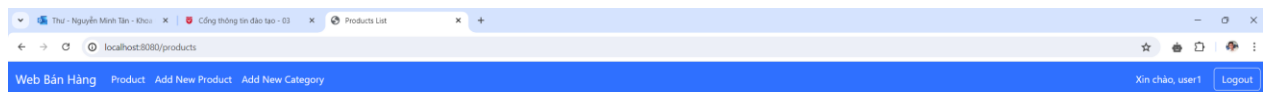
Role_id = 1: Tương ứng với quyền ADMIN

Các tài khoản người dùng khác mặc định quyền 'USER'

8.9.6 Đăng nhập và kiểm tra phân quyền

Sau khi phân quyền, đăng nhập vào ứng dụng bằng tài khoản có quyền 'ADMIN' và kiểm tra các chức năng đặc quyền như quản lý sản phẩm hoặc người dùng. Đồng thời, đăng nhập bằng tài khoản người dùng bình thường để đảm bảo rằng họ không truy cập được các chức năng đặc quyền.

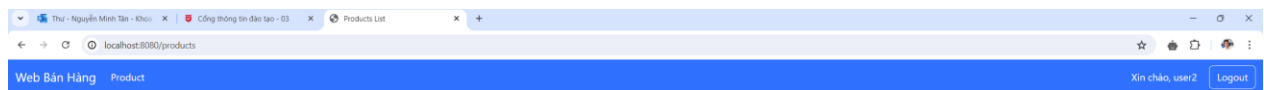
- Trang đăng nhập tài khoản quyền 'ADMIN':



Products List

| ID | Name | Price | Description | Category Name | Actions | Add To Cart |
|----|-------------------|--------|-------------------|---------------|---|--------------------------------|
| 1 | Laptop Phong Vũ 1 | 2000.0 | Laptop Phong Vũ 1 | Computer | Sửa Xóa | 1 Thêm Vào Giỏ |
| 2 | Mobile | 2222.0 | 2222 | Computer | Sửa Xóa | 1 Thêm Vào Giỏ |
| 3 | Laptop | 222.0 | 222 | Computer | Sửa Xóa | 1 Thêm Vào Giỏ |
| 4 | Laptop | 22.0 | 2222 | Laptop | Sửa Xóa | 1 Thêm Vào Giỏ |
| 5 | Laptop | 222.0 | 222 | Computer | Sửa Xóa | 1 Thêm Vào Giỏ |
| 6 | Laptop | 222.0 | 222 | Computer | Sửa Xóa | 1 Thêm Vào Giỏ |
| 7 | Laptop | 222.0 | 222 | Computer | Sửa Xóa | 1 Thêm Vào Giỏ |
| 8 | Laptop Minh Tân | 222.0 | Laptop Minh Tân | Computer | Sửa Xóa | 1 Thêm Vào Giỏ |

- Trang đăng nhập của người dùng bình thường:



Products List

| ID | Name | Price | Description | Category Name | Actions | Add To Cart |
|----|-------------------|--------|-------------------|---------------|---------|--------------------------------|
| 1 | Laptop Phong Vũ 1 | 2000.0 | Laptop Phong Vũ 1 | Computer | | 1 Thêm Vào Giỏ |
| 2 | Mobile | 2222.0 | 2222 | Computer | | 1 Thêm Vào Giỏ |
| 3 | Laptop | 222.0 | 222 | Computer | | 1 Thêm Vào Giỏ |
| 4 | Laptop | 22.0 | 2222 | Laptop | | 1 Thêm Vào Giỏ |
| 5 | Laptop | 222.0 | 222 | Computer | | 1 Thêm Vào Giỏ |
| 6 | Laptop | 222.0 | 222 | Computer | | 1 Thêm Vào Giỏ |
| 7 | Laptop | 222.0 | 222 | Computer | | 1 Thêm Vào Giỏ |
| 8 | Laptop Minh Tân | 222.0 | Laptop Minh Tân | Computer | | 1 Thêm Vào Giỏ |

Thực hiện các bước này sẽ giúp đảm bảo rằng ứng dụng của bạn hoạt động chính xác theo yêu cầu và quyền của người dùng được xử lý một cách an toàn và hiệu quả.