MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Bike Sharing System

-Proiect Final Ingineria Sistemelor-

Studentă: Strujan Florentina

Grupa: 302310

Anul 3, semestrul 1

Profesor coordonator: Eneia Nicolae Todoran

Echipă formată din: Strujan Florentina, Stroe Mădălina-Ionela

Cuprins

1.Introducere

Pentru proiectul Bike Sharing System am folosit Spring Framework, care este o platformă utilizată pentru simplificarea scrierii aplicațiilor în limbajul Java. Pe lângă Spring, am folosit și MySQL pentru gestionarea bazelor de date.

Aplicația este folosită pentru închirierea bicicletelor unor clienți. Pentru ca un client să poată închiria o bicicletă, este necesară înregistrarea acestuia, logarea ulterioară și editarea disponibilității bicicletei în unavailable. La pornirea aplicației , aceasta are posibilitatea alegerii funcționalităților atât pentru admin, cât și pentru client. Adminul are posibilitatea de a se loga pentru efectuarea operațiilor de modificare sau ștergere bicicletă ,dar și de ștergere a clienților actuali.

Am organizat proiectul în arhitectura MVC = model – view – controller. Modelul încapsulează modelele utilizate, datele aplicației. View-ul este folosit pentru aspect, în special fișierele HTML(+ css). Controller-ul este folosit pentru procesarea operațiilor. Spring Web MVC folosește un DispatcherServlet pentru a trata cererile și răspunsurile de tip HTTP.

2.Miniproiect

Pentru miniproiect am urmărit tutorialul din primul link din bibliografie , atât eu, cât și colega de proiect. Ea s-a ocupat de partea de view (fișierele .css și .html).

Controller:

```
import java.util.List;
import javax.servlet.http.HttpServletRequest;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.MailException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.MailException;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@Controller
public class AppController {

	private Logger logger=LoggerFactory.getLogger(AppController.class);

	@Autowired
	private NotificationService notificationService;

	@Autowired
	private UserProfile repo;

	@Autowired
	private BikeProfile repoBike;

	@GetMapping("/index")
	public String viewHomePage()
	{
		return "index";
	}



	@GetMapping("/register")
```

```
	public String showSignUpForm(Model model)
	{
		model.addAttribute("user",new User());
		return "signup_form";
	}

	@PostMapping("/process_register")
	public String processRegistration(User user)
	{
		BCryptPasswordEncoder encoder=new BCryptPasswordEncoder();
		String encodedPass=encoder.encode(user.getPassword());
		user.setPassword(encodedPass);
		repo.save(user);

		//User user= new User();
		user.setEmail(user.getEmail());
		user.setFirstName("te");
		user.setLastName("iubi");

		try
		{
			notificationService.sendNotification(user);
		}catch(MailException e)
		{
			logger.info("Error sending message: "+e.getMessage());
		}

		return "register_succes";
	}

	@GetMapping("/list_users")
	public String viewUsersList(Model model)
	{
		List<User> listUsers=repo.findAll();
		model.addAttribute("listUsers", listUsers);
		return "users";
	}

	@GetMapping("/list_bikes")
	public String viewBikesList(Model model)
	{
		List<Bike> listBikes=repoBike.findAll();
		model.addAttribute("listBikes", listBikes);
		return "bikes";
	}

}
```

Model:

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="bikesTable")
public class Bike {

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private Long id;
	@Column(nullable=false,length=20)
	private String brand;
	@Column(nullable=false,length=20)
	private String color;
	@Column(nullable=false,length=10)
	private String fullspeed;
	@Column(nullable=false,length=10)
	private String size;
	@Column(nullable=false,length=10)
	private String location;

	public Bike() {

	}

	public Long getId() {
		return id;
	}

	public void setId(Long id) {
		this.id = id;
	}

	public String getBrand() {
		return brand;
	}

	public void setBrand(String brand) {
		this.brand = brand;
	}

	public String getColor() {
```

```
		return color;
	}

	public void setColor(String color) {
		this.color = color;
	}

	public String getFullspeed() {
		return fullspeed;
	}

	public void setFullspeed(String fullspeed) {
		this.fullspeed = fullspeed;
	}

	public String getSize() {
		return size;
	}

	public void setSize(String size) {
		this.size = size;
	}

	public String getLocation() {
		return location;
	}

	public void setLocation(String location) {
		this.location = location;
	}
}
```

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

public interface BikeProfile extends JpaRepository<Bike, Long> {

		@Query("Select b from Bike b where b.brand=?1") //?1 pentru primul parametru

		Bike findbyBrand(String brand);
	}
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration;

@SpringBootApplication(exclude = { SecurityAutoConfiguration.class })

public class BikesApplication {

	public static void main(String[] args) {
		SpringApplication.run(BikesApplication.class, args);
	}

}


import java.util.Collection;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

public class CustomUserDetails implements UserDetails {

	private User user;

	public CustomUserDetails(User user) {
		super();
		this.user = user;
	}

	public User getUser() {
		return user;
	}

	public void setUser(User user) {
		this.user = user;
	}

	@Override
	public Collection<? extends GrantedAuthority> getAuthorities() {
		return null;
	}

	@Override
	public String getPassword() {
		return user.getPassword();
	}

	@Override
```

```
	public String getUsername() {
		return user.getEmail();
	}

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

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

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

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

	public String getFullName()
	{
		return user.getFirstName()+" "+user.getLastName();
	}

}



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;

public class CustomUserDetailsService implements UserDetailsService {

	@Autowired
	private UserProfile repo;

	@Override
	public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {
		User user=repo.findbyEmail(email);

		if(user==null)
```

```
		{
			throw new UsernameNotFoundException("User not found!");
		}
		return new CustomUserDetails(user);
	}
}


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.MailException;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.stereotype.Service;

@Service
public class NotificationService {

private JavaMailSender javaMailSender;

	@Autowired
	public NotificationService(JavaMailSender javaMailSender)
	{
		this.javaMailSender=javaMailSender;
	}

	public void sendNotification(User user) throws MailException
	{
		SimpleMailMessage mail=new SimpleMailMessage();

		mail.setFrom("bike.bss503@gmail.com");
		mail.setTo(user.getEmail());
		mail.setSubject("Notification");
		mail.setText("Thank you for registering with us!");

		javaMailSender.send(mail);
	}

}



import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

public class PasswordEncoder {

	public static void main(String[] args) {

		BCryptPasswordEncoder encoder=new BCryptPasswordEncoder();
```

```
		String rawPassword="passw";
		String encodedPass=encoder.encode(rawPassword);

		System.out.println(encodedPass);
	}

}



import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="usersTable")
public class User {

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private Long id;

	@Column(nullable=false,unique=true,length=20)
	private String firstName;
	@Column(nullable=false,unique=true,length=20)
	private String lastName;
	@Column(nullable=false,unique=true,length=45)
	private String email;
	@Column(nullable=false,unique=true,length=45)
	private String phoneNumber;
	@Column(nullable=false,unique=true,length=64)
	private String password;

	@Column(nullable=false,unique=false,length=64)
	private String payment;



	public User() {

		this.setPayment("Credit card");
	}


	public Long getId() {
```

```
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getPhoneNumber() {
		return phoneNumber;
	}
	public void setPhoneNumber(String phoneNumber) {
		this.phoneNumber = phoneNumber;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}

	public String getPayment() {
		return payment;
	}

	public void setPayment(String payment) {
		this.payment = payment;
	}


}
```

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

public interface UserProfile extends JpaRepository<User, Long> {

	@Query("Select u from User u where u.email=?1") //?1 pentru primul parametru

	User findbyEmail(String email);
}



import javax.sql.DataSource;
import org.springframework.beans.factory.annotation.Autowired;
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.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

	@Autowired
	private DataSource dataSource;

	@Bean
	public UserDetailsService userDetailsService()
	{
		return new CustomUserDetailsService();
	}

	@Bean
	public BCryptPasswordEncoder passwordEncoder()
	{
		return new BCryptPasswordEncoder();
	}

	@Bean
	public DaoAuthenticationProvider authenticationProvider()
	{
		DaoAuthenticationProvider authProvider= new DaoAuthenticationProvider();
```

```
		authProvider.setUserDetailsService(userDetailsService());
		authProvider.setPasswordEncoder(passwordEncoder());

		return authProvider;
	}

	@Override
	protected void configure(AuthenticationManagerBuilder auth) throws Exception {
		auth.authenticationProvider(authenticationProvider());
	}

	@Override
	protected void configure(HttpSecurity http) throws Exception {
		http.authorizeRequests()
		.antMatchers("/list_users")
		.authenticated()
		.anyRequest().permitAll()
		.and()
		.formLogin()
			.usernameParameter("email")
			.defaultSuccessUrl("/list_users")
			.permitAll()
		.and()
		.logout().logoutSuccessUrl("/").permitAll();
	}
}
```

View:

Css file:

```
@charset "ISO-8859-1";

body
{
	margin:0;
	background: #222;
	font-family: 'Work Sans', sans-serif;
	font-weight: 400;
}
.container
{
	width: 80%;
	margin: 0 auto;
}
```

```
header
{
	background: #55d6aa;
}

header::after
{
	content: '';
	display:table;
	clear:both;
}
h1
{
	color: #444;
}
h2
{
	color: #444;
}
h3
{
	color: #444;
}
h4
{
	color: #444;
}
nav
{
	float:right;
}
nav ul
{
	margin:0;
	padding:0;
	list-style: none;
}
nav li
{
	display: inline-block;
	margin-left:70px;
	padding-top:23px;
	position:relative;

}
nav a
{
	color: #444;
```

```
		text-decoration: none;
		text-transform: uppercase;
		font-size:14px;
}
nav a:hover
{
		color:#000;
}
nav a::before
{
		content:'';
		diplay: block;
		height: 5px;
		width:100%;
		background-color:#444;

		position: absolute;
		top:0;
		width: 0%;

		transition: all ease-in-out 250ms;
}
nav a:hover::before
{
		width:100%;
}
table
{
		width: 100%;
		border-collapse: collapse;
}
tr:nth-of-type(odd) {
  background: #eee;
}
th {
  background: #333;
  color: white;
  font-weight: bold;
}
td, th {
  padding: 6px;
  border: 1px solid #ccc;
  text-align: left;
}
p
{
		color: #55d6aa;
}
```

```
input[type=submit]
{
	background-color: #55d6aa;
	border: none;
	color: white;
	padding: 8px;
	text-align: center;
	text-decoration: none;
	display: inline-block;
	font-size: 15px;
	margin: 4px 2px;
	cursor: pointer;
	border-radius: 50%;
}

input[type=text] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  box-sizing: border-box;
  border: 3px solid #ccc;
  -webkit-transition: 0.5s;
  transition: 0.5s;
  outline: none;
}

input[type=text]:focus {
  border: 3px solid #555;
}

input[type=password] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  box-sizing: border-box;
  border: 3px solid #ccc;
  -webkit-transition: 0.5s;
  transition: 0.5s;
  outline: none;
}

input[type=password]:focus {
  border: 3px solid #555;
}
```

HTML files:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeLeaf.org">
<head>
<meta charset="ISO-8859-1">
<title>All Users</title>
<link rel="stylesheet" href="main.css" type="text/css" />
</head>
<body>
	<div class="container text-center">

	<div align="left">
		<form th:action="@{/index}">
			<input type="submit" class="btn btn-primary" value="Welcome Page"/>
		</form>
		</div>
	<div align="center">
			<p>
				Welcome <b>[[${#request.userPrincipal.principal.fullName}]]</b>
			</p>
		</div>
		<div>
			<h1 align="center">List of All Users</h1>
		</div>
		<div>
			<table class="table table-striped table-bordered">
				<thead class="thead-dark">
					<tr>
						<th>User ID</th>
						<th>E-mail</th>
						<th>First Name</th>
						<th>Last Name</th>
						<th>Phone</th>
						<th>Payment method</th>
					</tr>
				</thead:
				<tbody>
					<tr th:each="user:${listUsers}">
						<td th:text="${user.id}">User ID</td>
						<td th:text="${user.email}">E-mail</td>
						<td th:text="${user.firstName}">First Name</td>
						<td th:text="${user.lastName}">Last Name</td>
						<td th:text="${user.phoneNumber}">Phone</td>
						<td th:text="${user.payment}">Phone</td>
					</tr>
				</tbody>
```

```
                </table>
                <br>
        </div>
        <div align="center">
        <form th:action="@{/logout}" method="post">
                <input type="submit" class="btn btn-primary" value="Sign Out"/>
        </form>
        </div>
    </div>

</body>
</html>
```

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeLeaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Registration Page</title>
<link rel="stylesheet" href="main.css" type="text/css" />
</head>
<body>
    <div class="container text-center">

        <div>
            <h1 align="center">User Registration - Sign Up</h1>
            <br>
        </div>

        <div>
            <form th:action="@{/process_register}" method="post"
                style="max-width: 600px;margin:0 auto;"
                th:object="${user}">
                <div>
                    <div>
                        <label>First Name </label>
                        <div>
                            <input type="text" th:field="*{firstName}" required >
                        </div>
                    </div>
                    <div>
                        <label>Last Name </label>
                        <div>
                            <input type="text" th:field="*{lastName}" required >
                        </div>
```

```
                                    </div>
                                    <div>
                                        <label>E-mail </label>
                                        <div>
                                            <input type="text" th:field="*{email}" required >
                                        </div>

                                    </div>
                                    <div>
                                        <label>Phone</label>
                                        <div>
                                            <input type="text" th:field="*{phoneNumber}"
required >
                                        </div>
                                    </div>
                                    <div>
                                        <label>Password</label>
                                        <div>
                                            <input type="password" th:field="*{password}"
required >
                                        </div>
                                        <br>

                                    </div>
                                    <div>
                                        <label>Payment method</label>
                                        <div>
                                            <input type="radio" id="card" name="pay"
value="card">
                                            <label for="card">Credit Card</label><br>
                                        </div>
                                    </div>
                                    <br><br>
                                    <div align="center">
                                        <input type="submit" class="btn btn-primary" value="Sign
Up"/>
                                    </div>
                                </div>
                            </form>
                        </div>
                    </div>
                </body>
                </html>
```

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeLeaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Registration Succeeded</title>
<link rel="stylesheet" href="main.css" type="text/css" />
</head>
<body>
	<div class="container text-center">
		<div>
			<h3>You have signed up succesfully!</h3>
			<h4><a th:href="@{/login}">Click here to login</a></h4>
		</div>
	</div>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Registration Error</title>
</head>
<body>
	<div class="container text-center">
		<div>
			<h3>An error occured while registering. Please try again!</h3>
			<h4><a th:href="@{/signup_form}">Click here to register</a></h4>
		</div>
	</div>
</body>
</html>
```

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeLeaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Welcome Page</title>

<link rel="stylesheet" href="main.css" type="text/css" />
</head>
```

```
<body>
	<header>
		<div class="container text-center">

			<img	src="http://clipart-library.com/images_k/bike-clipart-transparent/bike-clipart-transparent-23.png" alt="bike" width="80" height="60">
			<nav>
				<ul>
					<li><a th:href="@{/register}">Register</a></li>
					<li><a th:href="@{/list_users}">List All Users</a></li>
					<li><a th:href="@{/list_bikes}">List All Bikes</a></li>
					<li><a th:href="@{/login}">Login</a></li>
				</ul>
			</nav>
		</div>
	</header>
</body>
</html>
```

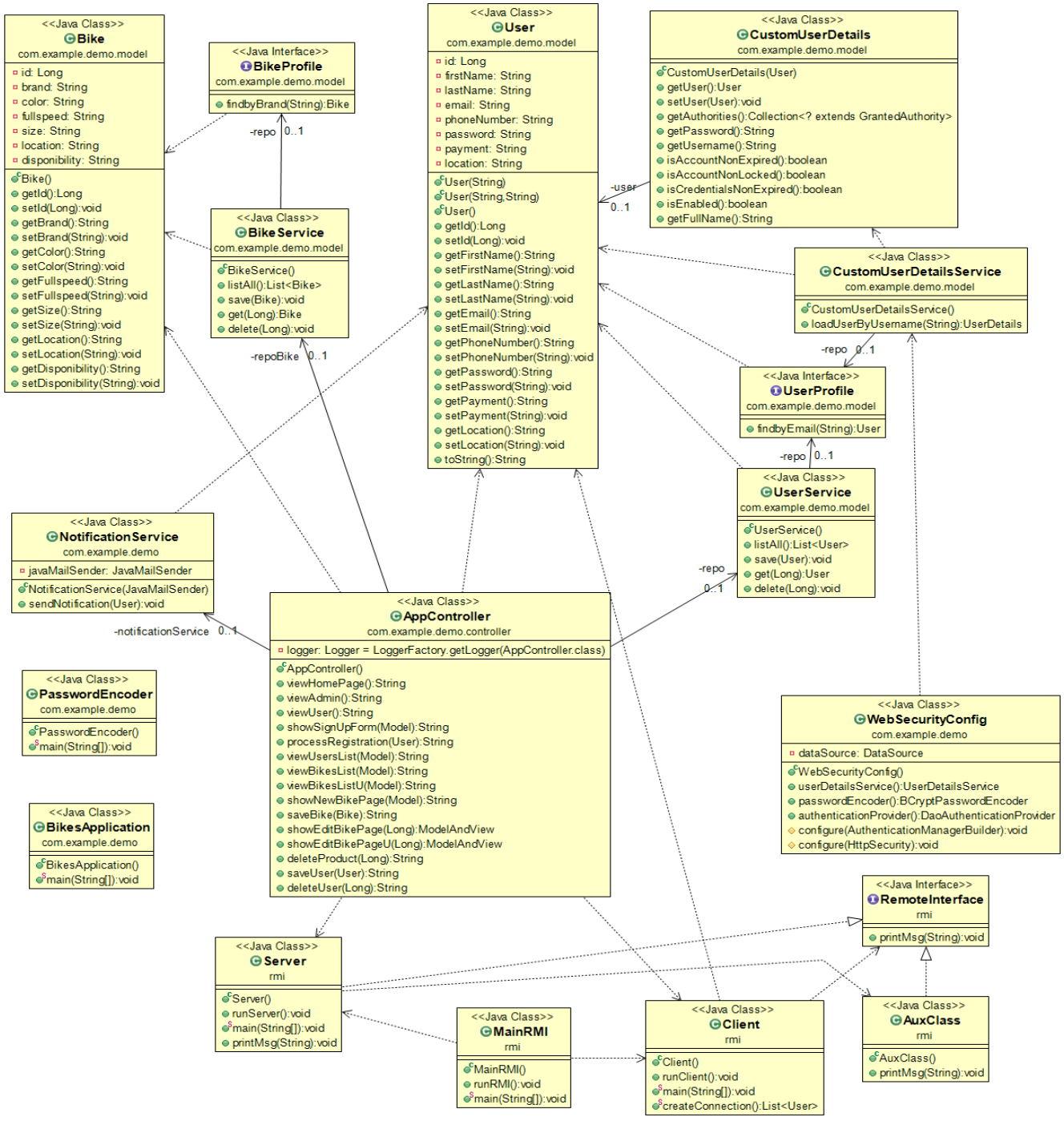
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeLeaf.org">
<head>
<meta charset="ISO-8859-1">
<title>All Bikes</title>
<link rel="stylesheet" href="main.css" type="text/css" />
</head>
<body>
	<div class="container text-center">
	<div align="left">
		<form th:action="@{/index}">
			<input type="submit" class="btn btn-primary" value="Welcome Page"/>
		</form>
		</div>
		<div align="center">
			<p>
				Welcome <b>[[${#request.userPrincipal.principal.fullName}]]</b>
			</p>
		</div>
				<div>
			<h1 align="center">List of All Bikes</h1>
		</div>
				<div>
			<table class="table table-striped table-bordered">
				<thead class="thead-dark">
					<tr>
```

```
                                        <th>Bike ID</th>
                                        <th>Brand</th>
                                        <th>Color</th>
                                        <th>Full Speed</th>
                                        <th>Size</th>
                                        <th>Location</th>
                                    </tr>

                                </thead>
                                <tbody>
                                    <tr th:each="bike:${listBikes}">
                                        <td th:text="${bike.id}">Bike ID</td>
                                        <td th:text="${bike.brand}">Brand</td>
                                        <td th:text="${bike.color}">Color</td>
                                        <td th:text="${bike.fullspeed}">Full Speed</td>
                                        <td th:text="${bike.size}">Size</td>
                                        <td th:text="${bike.location}">Size</td>
                                    </tr>
                                </tbody>
                            </table>
                            <br>
                        </div>
                        <div align="center">
                        <form th:action="@{/logout}" method="post">
                            <input type="submit" class="btn btn-primary" value="Sign Out"/>
                        </form>
                        </div>
                </div>
</body>
</html>
```

3.Proiect

3.1 Diagrame

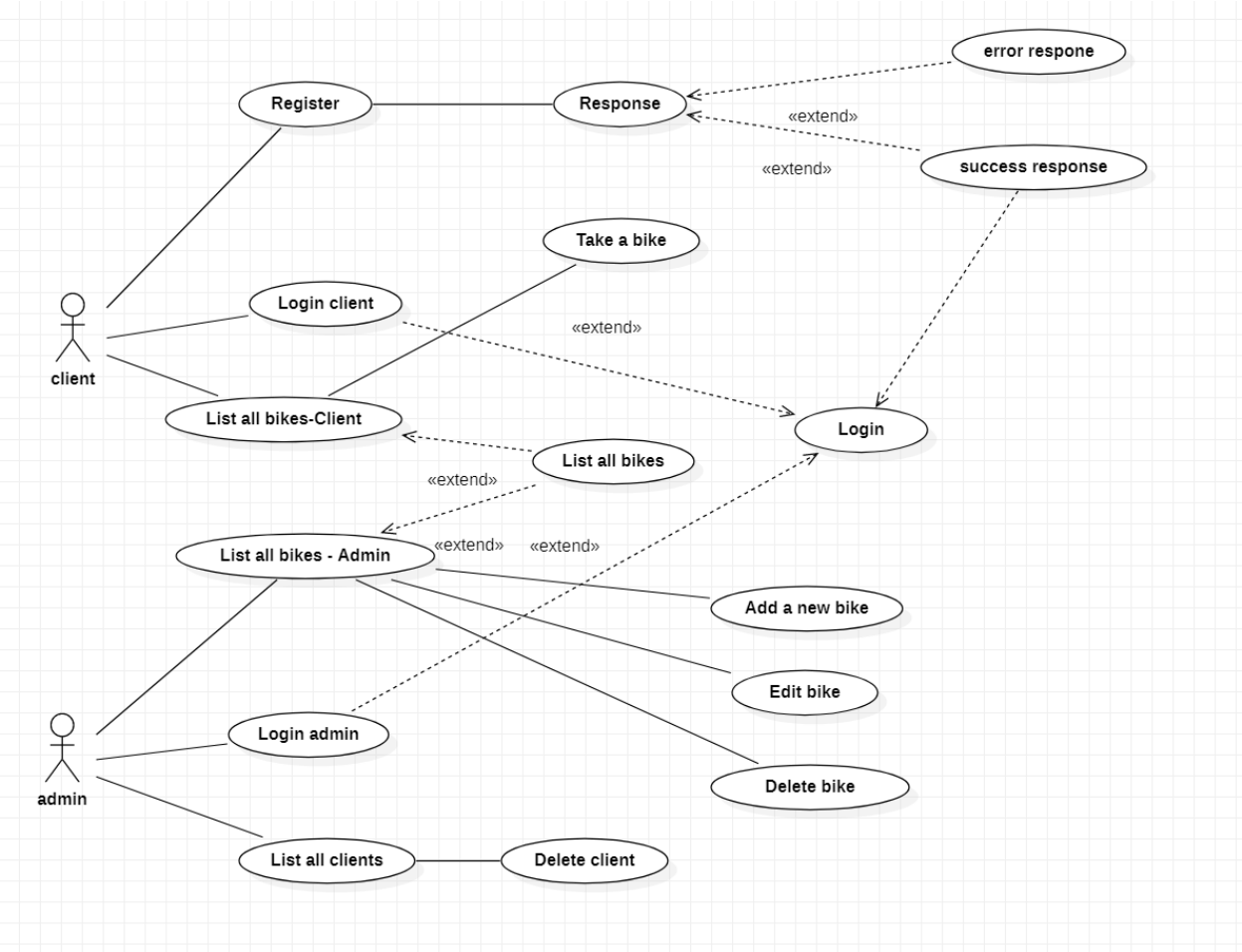
3.1.1 Diagrama de clase



În plus față de mini-proiect eu am adăugat clasele de UserService și BikeService pentru realizarea operațiilor de ștergere, adăugare și editare a datelor din baza de date corespunzătoare userilor și bicicletelor, dar și mult mai multe metode clasei AppController.

Colega mea a completat proiectul cu clasele corespunzătoare Java RMI.

3.2.2. Diagrama Use Case



Pe lângă diagrama inițială, am adăugat funcționalitățile de ștergere client, ștergere, adăugare și editare bicicletă, împărțite explicit pentru cele două tipuri de utilizatori.

3.2 Explicații și implementare

În primul rând, am adăugat două clase Service, una pentru User, una pentru Bike, pentru implementarea explicită a operațiilor de selectare, editare, ștergere și creare.

```
import java.util.List;
import javax.transaction.Transactional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
@Transactional
public class UserService {
        @Autowired
   private UserProfile repo;

   public List<User> listAll() {
      return repo.findAll();
   }

   public void save(User user) {
      repo.save(user);
   }

   public User get(Long id) {
      return repo.findById(id).get();
   }

   public void delete(Long id) {
      repo.deleteById(id);
   }

}

import java.util.List;
import javax.transaction.Transactional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
@Transactional
public class BikeService {
```

```
        @Autowired
    private BikeProfile repo;

    public List<Bike> listAll() {
        return repo.findAll();
    }

    public void save(Bike bike) {
        repo.save(bike);
    }

    public Bike get(Long id) {
        return repo.findById(id).get();
    }

    public void delete(Long id) {
        repo.deleteById(id);
    }
}
```

Ulterior, am modificat clasa AppController pentru adăugarea noilor funcționalități și legarea acestora pentru ordinea și logica site-ului.

```
import java.util.List;
import com.example.demo.*;
import javax.servlet.http.HttpServletRequest;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.MailException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.MailException;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.ModelAndView;
import com.example.demo.NotificationService;
import com.example.demo.model.*;
```

```
import rmi.Client;
import rmi.MainRMI;
import rmi.Server;

@Controller
public class AppController {

	private Logger logger=LoggerFactory.getLogger(AppController.class);

	@Autowired
	private NotificationService notificationService;

	@Autowired
	private UserService repo;

	@Autowired
	private BikeService repoBike;

	@GetMapping("/index")
	public String viewHomePage()
	{
		Server s=new Server();
		s.runServer();
		return "index";
	}

	@GetMapping("/admin")
	public String viewAdmin()
	{
		return "admin";
	}

	@GetMapping("/user")
	public String viewUser()
	{
		return "user";
	}

	@GetMapping("/register")
	public String showSignUpForm(Model model)
	{
		model.addAttribute("user",new User());
		return "signup_form";
	}

	@PostMapping("/process_register")
	public String processRegistration(User user)
	{
```

```
		BCryptPasswordEncoder encoder=new BCryptPasswordEncoder();
		String encodedPass=encoder.encode(user.getPassword());
		user.setPassword(encodedPass);
		repo.save(user);

		//MainRMI main= new MainRMI();
		//main.runRMI();
		Client c= new Client();
		c.runClient();

		try
		{
			notificationService.sendNotification(user);

		}catch(MailException e)
		{
			logger.info("Error sending message: "+e.getMessage());
		}

		return "register_succes";
}

@GetMapping("/list_users")
public String viewUsersList(Model model)
{
		List<User> listUsers=repo.listAll();
		model.addAttribute("listUsers", listUsers);
		return "users";
}

@GetMapping("/list_bikes")
public String viewBikesList(Model model)
{
		List<Bike> listBikes=repoBike.listAll();
		model.addAttribute("listBikes", listBikes);
		return "bikes";
}

@GetMapping("/list_bikes_user")
public String viewBikesListU(Model model)
{
		List<Bike> listBikes=repoBike.listAll();
		model.addAttribute("listBikes_user", listBikes);
		return "bikes_user";
}

@RequestMapping("/new")
public String showNewBikePage(Model model) {
```

```
    Bike bike = new Bike();
    model.addAttribute("bike", bike);

    return "new_bike";
}

@RequestMapping(value = "/saveB", method = RequestMethod.POST)
public String saveBike(@ModelAttribute("bike") Bike bike) {
    repoBike.save(bike);

    return "index";
}

@RequestMapping("/editB/{id}")
public ModelAndView showEditBikePage(@PathVariable(name = "id") Long id) {
    ModelAndView mav = new ModelAndView("edit_bike");
    Bike bike = repoBike.get(id);
    mav.addObject("bike", bike);

    return mav;
}

@RequestMapping("/editBU/{id}")
public ModelAndView showEditBikePageU(@PathVariable(name = "id") Long id) {
    ModelAndView mav = new ModelAndView("edit_bike_user");
    Bike bike = repoBike.get(id);
    mav.addObject("bike", bike);

    return mav;
}

@RequestMapping("/deleteB/{id}")
public String deleteProduct(@PathVariable(name = "id") Long id) {
    repoBike.delete(id);

    return "admin";
}


@RequestMapping(value = "/saveU", method = RequestMethod.POST)
public String saveUser(@ModelAttribute("user") User user) {
    repo.save(user);

    return "admin";
}


@RequestMapping("/deleteU/{id}")
```

```
	public String deleteUser(@PathVariable(name = "id") Long id) {
	   repo.delete(id);

	   return "admin";
	}
}
```

Pentru partea de design, am implementat noi fișiere .html pentru fiecare tip de user și pentru diferitele functionalități.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeLeaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Admin Page</title>

<link rel="stylesheet" href="main.css" type="text/css" />
</head>
<body>
	<header>
		<div class="container text-center">
			<img
src="https://cdn.dribbble.com/users/27903/screenshots/4207049/bike.png?compress=1&resize=400x300"
alt="bike" width="80" height="60">
			<nav>
				<ul>
					<li><a th:href="@{/list_users}">List All Users</a></li>
					<li><a th:href="@{/list_bikes}">List All Bikes</a></li>
					<li><a th:href="@{/login}">Login</a></li>
				</ul>
			</nav>
		</div>
	</header>

	<div align="left">
		<form th:action="@{/index}">
			<input type="submit" class="btn btn-primary" value="Welcome Page"/>
		</form>
	</div>
</body>
</html>
```

```
<!DOCTYPE html>
```

```
<html xmlns:th="http://www.thymeLeaf.org">
<head>
<meta charset="ISO-8859-1">
<title>User Page</title>

<link rel="stylesheet" href="main.css" type="text/css" />
</head>
<body>
	<header>
		<div class="container text-center">
			<img
src="https://cdn.dribbble.com/users/27903/screenshots/4207049/bike.png?compress=1&resize=400x300"
alt="bike" width="80" height="60">

			<nav>
				<ul>
					<li><a th:href="@{/list_bikes_user}">List All Bikes</a></li>
					<li><a th:href="@{/register}">Register</a></li>
					<li><a th:href="@{/login}">Login</a></li>
				</ul>
			</nav>
		</div>
	</header>

	<div align="left">
		<form th:action="@{/index}">
			<input type="submit" class="btn btn-primary" value="Welcome Page"/>
		</form>
	</div>
</body>
</html>
```

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeLeaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Create New Bike</title>
<link rel="stylesheet" href="main.css" type="text/css" />
</head>
<body>
	<div align="left">
		<form th:action="@{/admin}">
			<input type="submit" class="btn btn-primary" value="Admin Page"/>
		</form>
	</div>
```

```
        <div align="center">
    <h1>Create New Bike</h1>
    <br />
    <form action="#" th:action="@{/saveB}" th:object="${bike}"
      method="post">

      <table border="0" cellpadding="10">
        <tr>
                <td>Brand:</td>
                <td><input type="text" th:field="*{brand}" /></td>
        </tr>
        <tr>
                <td>Color:</td>
                <td><input type="text" th:field="*{color}" /></td>
        </tr>
        <tr>
                <td>Full speed:</td>
                <td><input type="text" th:field="*{fullspeed}" /></td>
        </tr>
        <tr>
                <td>Size:</td>
                <td><input type="text" th:field="*{size}" /></td>
        </tr>
        <tr>
                 <td>Location:</td>
                <td><input type="text" th:field="*{location}" /></td>
        </tr>
        <tr>
                <td>Disponibility:</td>
                <td><input type="text" th:field="*{disponibility}" /></td>
        </tr>
        <tr>
           <td colspan="2"><button type="submit">Save</button> </td>
        </tr>
      </table>
    </form>
  </div>

</body>
</html>



<!DOCTYPE html>
<html xmlns:th="http://www.thymeLeaf.org">
<head>
<meta charset="ISO-8859-1">
```

```
<title>Edit User</title>
<link rel="stylesheet" href="main.css" type="text/css" />
</head>
<body>
		<div align="left">
			<form th:action="@{/admin}">
				<input type="submit" class="btn btn-primary" value="Admin Page"/>
			</form>
		</div>
  <div align="center">
    <h1>Edit User</h1>
    <br />
    <form action="#" th:action="@{/saveU}" th:object="${user}"
      method="post">

      <table border="0" cellpadding="10">
        <tr>
          <td>E-mail:</td>
          <td><input type="text" th:field="*{email}" readonly="readonly" /></td>
        </tr>
        <tr>
          <td>First Name:</td>
          <td><input type="text" th:field="*{firstName}" /></td>
        </tr>
        <tr>
          <td>Last Name:</td>
          <td><input type="text" th:field="*{lastName}" /></td>
        </tr>
        <tr>
          <td>Location:</td>
          <td><input type="text" th:field="*{location}" /></td>
        </tr>
        <tr>
          <td>Phone:</td>
          <td><input type="text" th:field="*{phoneNumber}" /></td>
        </tr>
        <tr>
          <td>Payment:</td>
          <td><input type="text" th:field="*{payment}" /></td>
        </tr>
        <tr>
          <td colspan="2"><button type="submit">Save</button> </td>
        </tr>
      </table>
    </form>
  </div>
</body>
</html>
```

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeLeaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Edit Bike</title>
<link rel="stylesheet" href="main.css" type="text/css" />
</head>
<body>
		<div align="left">
			<form th:action="@{/admin}">
				<input type="submit" class="btn btn-primary" value="Admin Page"/>
			</form>
		</div>
  <div align="center">
    <h1>Edit Bike</h1>
    <br />
    <form action="#" th:action="@{/saveB}" th:object="${bike}"
      method="post">

      <table border="0" cellpadding="10">
        <tr>
          <td>Brand:</td>
          <td><input type="text" th:field="*{brand}" /></td>
        </tr>
        <tr>
          <td>Color:</td>
          <td><input type="text" th:field="*{color}" /></td>
        </tr>
        <tr>
          <td>Full speed:</td>
          <td><input type="text" th:field="*{fullspeed}" /></td>
        </tr>
        <tr>
          <td>Size:</td>
          <td><input type="text" th:field="*{size}" /></td>
        </tr>
        <tr>
          <td>Location:</td>
          <td><input type="text" th:field="*{location}" /></td>
        </tr>
        <tr>
          <td>Disponibility:</td>
```

```
            <td><input type="text" th:field="*{disponibility}" /></td>
          </tr>
          <tr>
            <td colspan="2"><button type="submit">Save</button> </td>
          </tr>
        </table>
      </form>
    </div>
</body>
</html>
```

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeLeaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Edit Bike</title>
<link rel="stylesheet" href="main.css" type="text/css" />
</head>
<body>
          <div align="left">
                    <form th:action="@{/user}">
                              <input type="submit" class="btn btn-primary" value="User Page"/>
                    </form>
          </div>
   <div align="center">
      <h1>Edit Bike</h1>
      <br />
      <form action="#" th:action="@{/saveB}" th:object="${bike}"
         method="post">

        <table border="0" cellpadding="10">
          <tr>
            <td>Brand:</td>
            <td><input type="text" th:field="*{brand}" readonly="readonly"/></td>
          </tr>
          <tr>
            <td>Color:</td>
            <td><input type="text" th:field="*{color}" readonly="readonly"/></td>
          </tr>
          <tr>
            <td>Full speed:</td>
            <td><input type="text" th:field="*{fullspeed}" readonly="readonly"/></td>
          </tr>
          <tr>
            <td>Size:</td>
```

```
            <td><input type="text" th:field="*{size}" readonly="readonly"/></td>
          </tr>
          <tr>
            <td>Location:</td>
            <td><input type="text" th:field="*{location}" readonly="readonly"/></td>
          </tr>
          <tr>
            <td>Disponibility:</td>
            <td><input type="text" th:field="*{disponibility}" /></td>
          </tr>
          <tr>
            <td colspan="2"><button type="submit">Save</button> </td>
          </tr>
        </table>
      </form>
    </div>
</body>
</html>
```

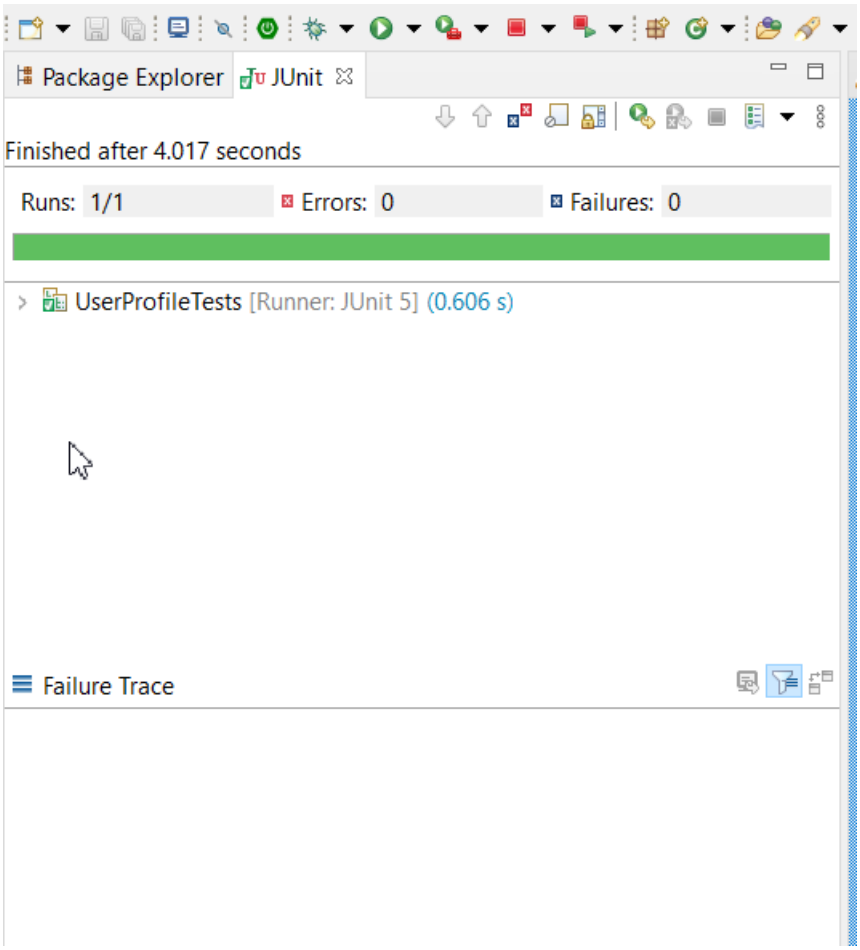
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeLeaf.org">
<head>
<meta charset="ISO-8859-1">
<title>All Bikes</title>
<link rel="stylesheet" href="main.css" type="text/css" />
</head>
<body>
        <div class="container text-center">

        <div align="left">
                <form th:action="@{/user}">
                        <input type="submit" class="btn btn-primary" value="User Page"/>
                </form>
                </div>

                <div align="center">
                        <p>
                                Welcome <b>[[${#request.userPrincipal.principal.fullName}]]</b>
                        </p>
                </div>

                <div>
                        <h1 align="center">List of All Bikes</h1>
                </div>
```

```
<div>
	<table border="0" cellpadding="10">
		<tr>
			<th>Bike ID</th>
			<th>Brand</th>
			<th>Color</th>
			<th>Full Speed</th>
			<th>Size</th>
			<th>Location</th>
			<th>Disponibility</th>
			<th>Actions</th>
		</tr>


		<tbody>
			<tr th:each="bike:${listBikes_user}">
				<td th:text="${bike.id}">Bike ID</td>
				<td th:text="${bike.brand}">Brand</td>
				<td th:text="${bike.color}">Color</td>
				<td th:text="${bike.fullspeed}">Full Speed</td>
				<td th:text="${bike.size}">Size</td>
				<td th:text="${bike.location}">Location</td>
				<td th:text="${bike.disponibility}">Disponibility</td>

				<td>
	<a th:href="@{'editBU/'+${bike.id}}">Edit</a>
	</td>
			</tr>
		</tbody>
	</table>
	<br>
</div>


<div align="center">
<form th:action="@{/logout}" method="post">
	<input type="submit" class="btn btn-primary" value="Sign Out"/>
</form>
</div>

</div>

</body>
</html>
```

Pentru a testa buna funcționare a aplicației, am adăugat 3 clase de testare pentru funcționarea aplicației, pentru adăugarea și găsirea unui utilizator și a unei biciclete.

```
import static org.assertj.core.api.Assertions.assertThat;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;
import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase.Replace;
import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
import org.springframework.boot.test.autoconfigure.orm.jpa.TestEntityManager;
import org.springframework.test.annotation.Rollback;
import com.example.demo.model.User;
import com.example.demo.model.UserProfile;

@DataJpaTest
@AutoConfigureTestDatabase(replace=Replace.NONE)
@Rollback(false)
public class UserProfileTests {

	@Autowired
	private UserProfile repo;

	@Autowired
	private TestEntityManager entityManager;

	@Test
	public void testCreateUser() {
		User user=new User();
		user.setEmail("violeta@yahoo.com");
		user.setFirstName("Violeta");
		user.setLastName("Mihai");
		user.setPhoneNumber("0757123823");
		user.setPassword("pass");
		user.setPayment("credit card");
		user.setLocation("Cluj");

		User savedUser = repo.save(user);

		User existUser = entityManager.find(User.class, savedUser.getId());

		assertThat(existUser.getEmail()).isEqualTo(user.getEmail());
	}

	@Test
	public void testFindUserByEmail()
```

```
	{
		String email = "strujan.florentina@yahoo.com";
		User user=repo.findbyEmail(email);

		assertThat(user).isNotNull();
	}
}
```




```
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class BikesApplicationTests {

	@Test
```

```
	void contextLoads() {
	}

}
```

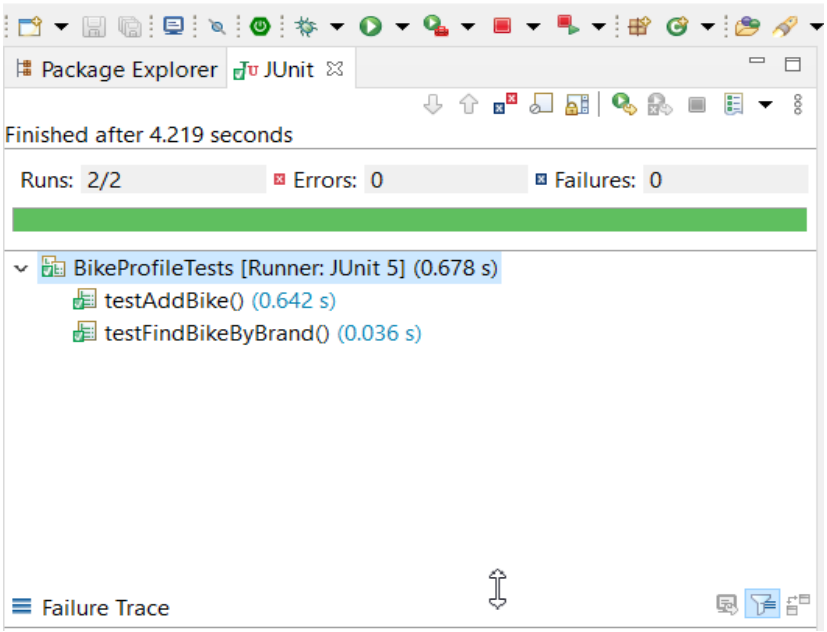
Am folosit testul de la adăugarea bicicletelor și pentru o populare rapidă și ușoara a tabelei Bike.

```
import static org.assertj.core.api.Assertions.assertThat;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;
import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase.Replace;
import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
import org.springframework.boot.test.autoconfigure.orm.jpa.TestEntityManager;
import org.springframework.test.annotation.Rollback;

import com.example.demo.model.Bike;
import com.example.demo.model.BikeProfile;


@DataJpaTest
@AutoConfigureTestDatabase(replace=Replace.NONE)
@Rollback(false)
public class BikeProfileTests {

	@Autowired
	private BikeProfile rep;

	@Autowired
	private TestEntityManager entityManager;

	@Test
	public void testFindBikeByBrand()
	{
		String brand = "Altruiste Bikes";
		Bike bike=rep.findbyBrand(brand);

		assertThat(bike).isNotNull();
	}

	@Test
	public void testAddBike() {
		Bike bike=new Bike();
		bike.setBrand("Steppenwolf");
```

```
bike.setColor("Black");
bike.setFullspeed("100km/h");
bike.setSize("M");
bike.setLocation("Bucharest");
bike.setDisponibility("Available");

Bike savedBike= rep.save(bike);

Bike bike1=new Bike();
bike1.setBrand("Velovie");
bike1.setColor("Green");
bike1.setFullspeed("95km/h");
bike1.setSize("L");
bike1.setLocation("Cluj-Napoca");
bike1.setDisponibility("Available");

Bike savedBike1= rep.save(bike1);

Bike bike2=new Bike();
bike2.setBrand("Altruiste Bikes");
bike2.setColor("Red");
bike2.setFullspeed("120km/h");
bike2.setSize("S");
bike2.setLocation("Bucharest");
bike2.setDisponibility("Available");

Bike savedBike2= rep.save(bike2);

Bike bike3=new Bike();
bike3.setBrand("Thesis");
bike3.setColor("Black");
bike3.setFullspeed("60km/h");
bike3.setSize("XL");
bike3.setLocation("Cluj-Napoca");
bike3.setDisponibility("Available");

Bike savedBike3= rep.save(bike3);

Bike bike4=new Bike();
bike4.setBrand("Stinner Frameworks");
bike4.setColor("Purple");
bike4.setFullspeed("45km/h");
bike4.setSize("M");
bike4.setLocation("Cluj-Napoca");
bike4.setDisponibility("Available");

Bike savedBike4= rep.save(bike4);
```

```
Bike bike5=new Bike();
bike5.setBrand("Rotwild");
bike5.setColor("Black");
bike5.setFullspeed("55km/h");
bike5.setSize("L");
bike5.setLocation("Bucharest");
bike5.setDisponibility("Available");

Bike savedBike5= rep.save(bike5);

Bike bike6=new Bike();
bike6.setBrand("Racycles");
bike6.setColor("Yellow");
bike6.setFullspeed("100km/h");
bike6.setSize("XS");
bike6.setLocation("Bucharest");
bike6.setDisponibility("Available");

Bike savedBike6= rep.save(bike6);

Bike bike7=new Bike();
bike7.setBrand("Pashley");
bike7.setColor("Black");
bike7.setFullspeed("70km/h");
bike7.setSize("L");
bike7.setLocation("Cluj-Napoca");
bike7.setDisponibility("Available");

Bike savedBike7= rep.save(bike7);

Bike bike8=new Bike();
bike8.setBrand("Nukeproof");
bike8.setColor("Brown");
bike8.setFullspeed("60km/h");
bike8.setSize("Medium");
bike8.setLocation("Bucharest");
bike8.setDisponibility("Available");

Bike savedBike8= rep.save(bike8);

Bike bike9=new Bike();
bike9.setBrand("Mondraker");
bike9.setColor("Black");
bike9.setFullspeed("100km/h");
bike9.setSize("XL");
bike9.setLocation("Cluj-Napoca");
bike9.setDisponibility("Available");
```

```
                Bike savedBike9= rep.save(bike9);

                Bike bike10=new Bike();
                bike10.setBrand("Fondriest");
                bike10.setColor("Green");
                bike10.setFullspeed("80km/h");
                bike10.setSize("M");
                bike10.setLocation("Cluj-Napoca");
                bike10.setDisponibility("Available");

                Bike savedBike10= rep.save(bike10);

                Bike bike11=new Bike();
                bike11.setBrand("Davidson");
                bike11.setColor("Blue");
                bike11.setFullspeed("120km/h");
                bike11.setSize("L");
                bike11.setLocation("Bucharest");
                bike11.setDisponibility("Available");

                Bike savedBike11= rep.save(bike11);

                Bike existBike = entityManager.find(Bike.class, savedBike.getId());
                assertThat(existBike.getBrand()).isEqualTo(bike.getBrand());
        }
}
```



4.Bibliografie

https://www.youtube.com/watch?v=aRLoSDOlU3w&ab_channel=CodeJava

https://www.youtube.com/watch?v=QloyS2dt9T4

https://www.baeldung.com/spring-boot-crud-thymeleaf

https://docs.spring.io/spring-framework/docs/current/reference/html/testing.html

https://spring.io/guides/gs/accessing-data-mysql/

https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html

https://spring.io/projects/spring-boot