ГУАП КАФЕДРА № 43

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ ПРЕПОДАВАТЕЛЬ

|  |  |  |
| --- | --- | --- |
| ассистент |  | П.А.Степанов |
| должность, уч. степень, звание | подпись, дата | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4 |
| «Разработка формы логина» |
| по дисциплине: ТЕХНОЛОГИЯ РАЗРАБОТКИ СЕРВЕРНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ |

РАБОТУ ВЫПОЛНИЛА

|  |  |  |
| --- | --- | --- |
| СТУДЕНТКА ГР. | 4232 | Е.А.Уткина |
|  | подпись, дата | инициалы, фамилия |

Санкт-Петербург

2024

1. Цель работы:

Ознакомиться с основами авторизации и аутентификации. Разработать форму логина.

1. Задание:

- Переведите веб-сервер на использование SSL

- Реализуйте form-based authentification

- Защитите ресурсы, требующие запись, и введите аудит

Вариант 14: Сдача недвижимости в аренду.

1. Листинг программы:

ApartmentRestController.Java:

/\*  
 \* this code is available under GNU GPL v3  
 \* https://www.gnu.org/licenses/gpl-3.0.en.html  
 \*/  
package info.stepanoff.trsis.samples.rest;  
  
import info.stepanoff.trsis.samples.rest.model.ApartmentDTO;  
import info.stepanoff.trsis.samples.rest.model.ApartmentDataWithoutID;  
import info.stepanoff.trsis.samples.service.ApartmentService;  
import io.swagger.v3.oas.annotations.Operation;  
import io.swagger.v3.oas.annotations.Parameter;  
import lombok.RequiredArgsConstructor;  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.security.Principal;  
import java.util.List;  
  
@Slf4j  
@RestController  
@RequestMapping(value = "/public/rest/apartments", produces = "application/json")  
@RequiredArgsConstructor  
public class ApartmentRestController {  
  
 private final ApartmentService apartmentService;  
  
 @RequestMapping(value = "", method = RequestMethod.GET)  
 @Operation(summary = "Получить список квартир для сдачи",  
 description = "Получить список квартир для сдачи - расширенное описание",  
 responses = {  
 @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "200",  
 description = "Успешное выполнение"),  
 @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "404",  
 description = "Ресурс не найден")  
 })  
 public ResponseEntity<List<ApartmentDTO>> browse() {  
 return ResponseEntity.ok(apartmentService.listAll());  
 }  
  
 @Operation(summary = "Удаление записи о квартире",  
 description = "Удаление записи о квартире - расширенное описание",  
 responses = {  
 @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "200",  
 description = "Успешное выполнение"),  
 @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "401",  
 description = "Требуется аутентификация"),  
 @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "403",  
 description = "Аутентификация предоставлена, но у пользователя нет доступа"),  
 @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "404",  
 description = "Ресурс не найден")  
 })  
 @RequestMapping(value = "/{id}", method = RequestMethod.DELETE)  
 public void delete(@PathVariable("id")  
 @Parameter(description = "Идентификатор квартиры") Integer id, Principal principal) {  
 if (principal == null) {  
 throw new ForbiddenException();  
 }  
 apartmentService.delete(id);  
 }  
  
 @Operation(summary = "Демонстрация работы delete при помощи метода get",  
 description = "Демонстрационный метод")  
 @RequestMapping(value = "/mockdelete/{id}", method = RequestMethod.GET)  
 public void mockdelete(@PathVariable("id") Integer id, Principal principal) {  
 if (principal == null) {  
 throw new ForbiddenException();  
 }  
  
 apartmentService.delete(id);  
 }  
  
 @Operation(summary = "Создать новую запись о квартире",  
 description = "Создать новую запись о квартире - расширенное описание",  
 responses = {  
 @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "200",  
 description = "Успешное выполнение"),  
 @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "401",  
 description = "Требуется аутентификация"),  
 @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "403",  
 description = "Аутентификация предоставлена, но у пользователя нет доступа"),  
 @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "404",  
 description = "Ресурс не найден")  
 })  
  
 @RequestMapping(value = "", method = RequestMethod.POST)  
 public ResponseEntity<ApartmentDTO> handleApartmentRequest(  
 @Parameter(description = "JSON с данными о квартире", required = true)  
 @RequestBody ApartmentDataWithoutID requestBody, Principal principal) {  
 if (principal == null) {  
 throw new ForbiddenException();  
 }  
  
 String city = requestBody.getCity();  
 String address = requestBody.getAddress();  
 Integer price = requestBody.getPrice();  
  
 return ResponseEntity.ok(apartmentService.add(city, address, price));  
 }  
  
 @Operation(summary = "Поиск квартиры по городу",  
 description = "Поиск квартиры по городу - расширенное описание",  
 responses = {  
 @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "200",  
 description = "Успешное выполнение"),  
 @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "401",  
 description = "Требуется аутентификация"),  
 @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "403",  
 description = "Аутентификация предоставлена, но у пользователя нет доступа"),  
 @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "404",  
 description = "Ресурс не найден")  
 })  
 @RequestMapping(value = "/city/{city}", method = RequestMethod.GET)  
 public ResponseEntity<ApartmentDTO> findByCity(@PathVariable("city")  
 @Parameter(description = "Город квартиры") String city) {  
 return ResponseEntity.ok(apartmentService.findByCity(city));  
 }  
}

Apartments.html:

<!DOCTYPE html>  
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org"  
 xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity6">  
<head>  
 <meta charset="utf-8"/>  
 <meta http-equiv="X-UA-Compatible" content="IE=edge"/>  
 <meta name="viewport" content="width=device-width, initial-scale=1"/>  
  
 <meta name="\_csrf" content="${\_csrf.token}" th:content="${\_csrf.token}"/>  
 <meta name="\_csrf\_header" content="${\_csrf.parameterName}" th:content="${\_csrf.parameterName}"/>  
  
 <title>Сдача квартир в аренду</title>  
  
 <link href="../static/css/bootstrap.min.css" th:href="@{/css/bootstrap.min.css}" rel="stylesheet">  
 <script src="../static/js/angular.js" th:src="@{js/angular.js}"></script>  
 <script src="../static/js/jquery-3.7.1.slim.min.js" th:src="@{js/jquery-3.7.1.slim.min.js}"></script>  
 <script src="../static/js/bootstrap.js" th:src="@{js/bootstrap.js}"></script>  
 <script src="../static/js/apartments.js" th:src="@{js/apartments.js}"></script>  
  
 <style>  
 body {  
 background-color: #f0f2f5; /\* Более светлый и современный фон \*/  
 font-family: 'Roboto', sans-serif; /\* Изменен шрифт на Roboto \*/  
 padding: 40px;  
 }  
 h1 {  
 font-size: 2.5rem;  
 color: #2c3e50;  
 font-weight: 600;  
 }  
 .form-heading {  
 font-size: 1.8rem;  
 color: #34495e;  
 margin-top: 30px;  
 margin-bottom: 20px;  
 }  
 .table-container {  
 margin-top: 40px;  
 }  
 .form-control {  
 border-radius: 8px;  
 font-size: 1rem;  
 padding: 10px;  
 border: 1px solid #ced4da;  
 }  
 .btn {  
 border-radius: 8px;  
 padding: 10px 20px;  
 font-size: 1rem;  
 transition: background-color 0.2s ease;  
 }  
 .btn-primary {  
 background-color: #007bff;  
 border-color: #007bff;  
 }  
 .btn-primary:hover {  
 background-color: #0056b3;  
 border-color: #0056b3;  
 }  
 .btn-danger {  
 background-color: #dc3545;  
 border-color: #dc3545;  
 }  
 .btn-danger:hover {  
 background-color: #c82333;  
 border-color: #bd2130;  
 }  
 .alert {  
 margin-top: 20px;  
 font-size: 1.1rem;  
 }  
 .table th {  
 background-color: #007bff;  
 color: #fff;  
 font-size: 1rem;  
 text-align: center;  
 }  
 .table td {  
 text-align: center;  
 vertical-align: middle;  
 font-size: 1rem;  
 }  
 .thead-dark th {  
 background-color: #007bff;  
 border-color: #007bff;  
 }  
 .table-hover tbody tr:hover {  
 background-color: #f2f2f2;  
 }  
 label {  
 font-size: 1rem;  
 color: #495057;  
 margin-bottom: 5px;  
 }  
 input::placeholder {  
 color: #adb5bd;  
 font-style: italic;  
 }  
 .form-group {  
 margin-bottom: 20px;  
 }  
 </style>  
</head>  
<body ng-app="apartments" ng-controller="ApartmentsController">  
<div class="container">  
 <div class="d-flex justify-content-between align-items-center">  
 <h1>Список квартир для сдачи в аренду</h1>  
 <div>  
 <div sec:authorize="isAnonymous()">  
 <a href="/login" class="btn btn-primary">Login</a>  
 </div>  
 <div sec:authorize="isAuthenticated()">  
 <form name="logoutForm" th:action="@{/logout}" method="post" style="display: inline;">  
 <input type="submit" class="btn btn-danger" value="Выйти"/>  
 </form>  
 </div>  
 </div>  
 </div>  
  
 <div class="table-container" ng-controller="getApartments" ng-show="apartmentsList.length > 0">  
 <table id="apartmentsTable" class="table table-hover table-bordered table-striped">  
 <thead class="thead-dark">  
 <tr>  
 <th>ID</th>  
 <th>Город</th>  
 <th>Адрес</th>  
 <th>Цена, руб./мес</th>  
 <th>Действия</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr ng-repeat="apartment in apartmentsList" id="{{apartment.id}}">  
 <td>{{apartment.id}}</td>  
 <td>{{apartment.city}}</td>  
 <td>{{apartment.address}}</td>  
 <td>{{apartment.price}}</td>  
 <td>  
 <div sec:authorize="isAuthenticated()">  
 <button class="btn btn-danger btn-sm" ng-click="deleteApartment(apartment.id)">Удалить</button>  
 </div>  
 </td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
  
 <div class="alert alert-danger" role="alert" ng-show="errormessage">{{errormessage}}</div>  
  
 <div class="mt-4" sec:authorize="isAuthenticated()">  
 <form role="form" ng-submit="addApartment()">  
 <h3 class="form-heading">Добавить квартиру для сдачи</h3>  
 <div class="form-group">  
 <label for="inputCity">Город</label>  
 <input type="text" class="form-control" id="inputCity" name="inputCity" placeholder="Город" ng-model="inputCity" required>  
 </div>  
 <div class="form-group">  
 <label for="inputAddress">Адрес</label>  
 <input type="text" class="form-control" id="inputAddress" name="inputAddress" ng-model="inputAddress" placeholder="Адрес" required>  
 </div>  
 <div class="form-group">  
 <label for="inputPrice">Цена, руб./мес</label>  
 <input type="number" class="form-control" id="inputPrice" name="inputPrice" ng-model="inputPrice" placeholder="Цена, руб./мес" required>  
 </div>  
 <button type="submit" class="btn btn-primary">Добавить</button>  
 </form>  
 </div>  
</div>  
</body>  
</html>

Application.properties:

server.port: 8443  
server.ssl.key-store: target/keystore.p12  
server.ssl.key-store-password: mypassword  
server.ssl.keyStoreType: PKCS12  
server.ssl.keyAlias: tomcat

SecurityConfiguration.java:

/\*  
 \* this code is available under GNU GPL v3  
 \* https://www.gnu.org/licenses/gpl-3.0.en.html  
 \*/  
package info.stepanoff.trsis.samples.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
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.crypto.password.PasswordEncoder;  
import org.springframework.security.web.SecurityFilterChain;  
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;  
  
@Configuration  
@EnableWebSecurity  
public class SecurityConfiguration {  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http.authorizeHttpRequests(authorizationManagerRequestMatcherRegistry  
 -> authorizationManagerRequestMatcherRegistry.requestMatchers("/js/\*\*", "/css/\*\*", "/apartments", "/public/rest/\*").permitAll()  
 .anyRequest().authenticated())  
 .formLogin(httpSecurityFormLoginConfigurer  
 -> httpSecurityFormLoginConfigurer  
 .loginPage("/login")  
 .usernameParameter("login")  
 .passwordParameter("pass")  
 .permitAll()  
 .defaultSuccessUrl("/apartments"))  
 .logout(httpSecurityLogoutConfigurer -> httpSecurityLogoutConfigurer.permitAll()  
 .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))  
 .logoutSuccessUrl("/login"));  
 return http.build();  
 }  
  
 //@Autowired  
 public void configureGlobal(AuthenticationManagerBuilder auth, PasswordEncoder passwordEncoder) throws Exception {  
 auth.inMemoryAuthentication()  
 .withUser("user")  
 .password(passwordEncoder.encode("password"))  
 .roles("ALL")  
 .and()  
 .withUser("admin")  
 .password(passwordEncoder.encode("admin"))  
 .roles("ALL");  
 }  
  
}

LoginController.java:

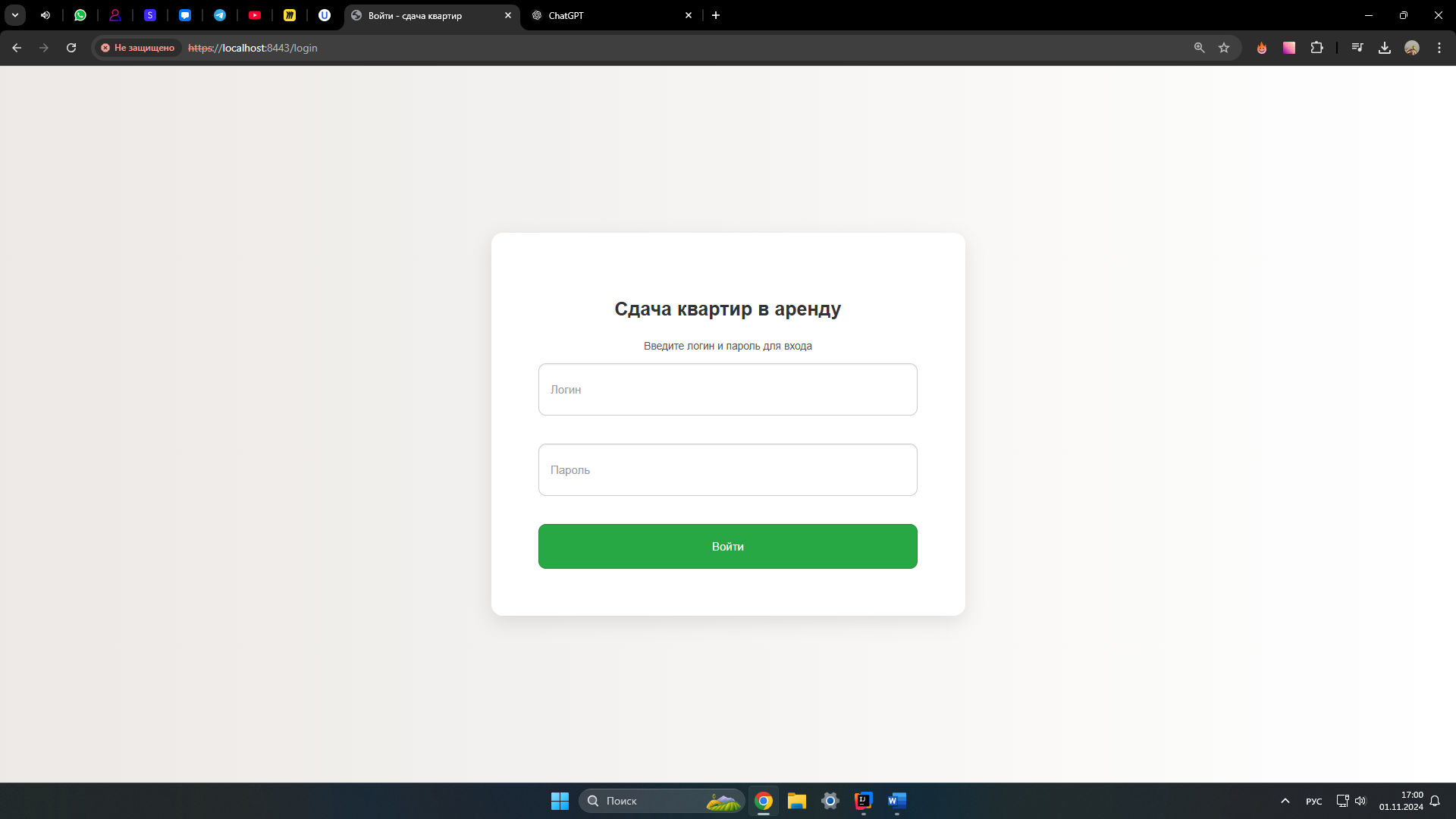
package info.stepanoff.trsis.samples.controller;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.RequestMapping;  
  
@Controller  
public class LoginController {  
  
 @RequestMapping("/login")  
 public String login() {  
 return "login";  
 }  
}

Login.html:

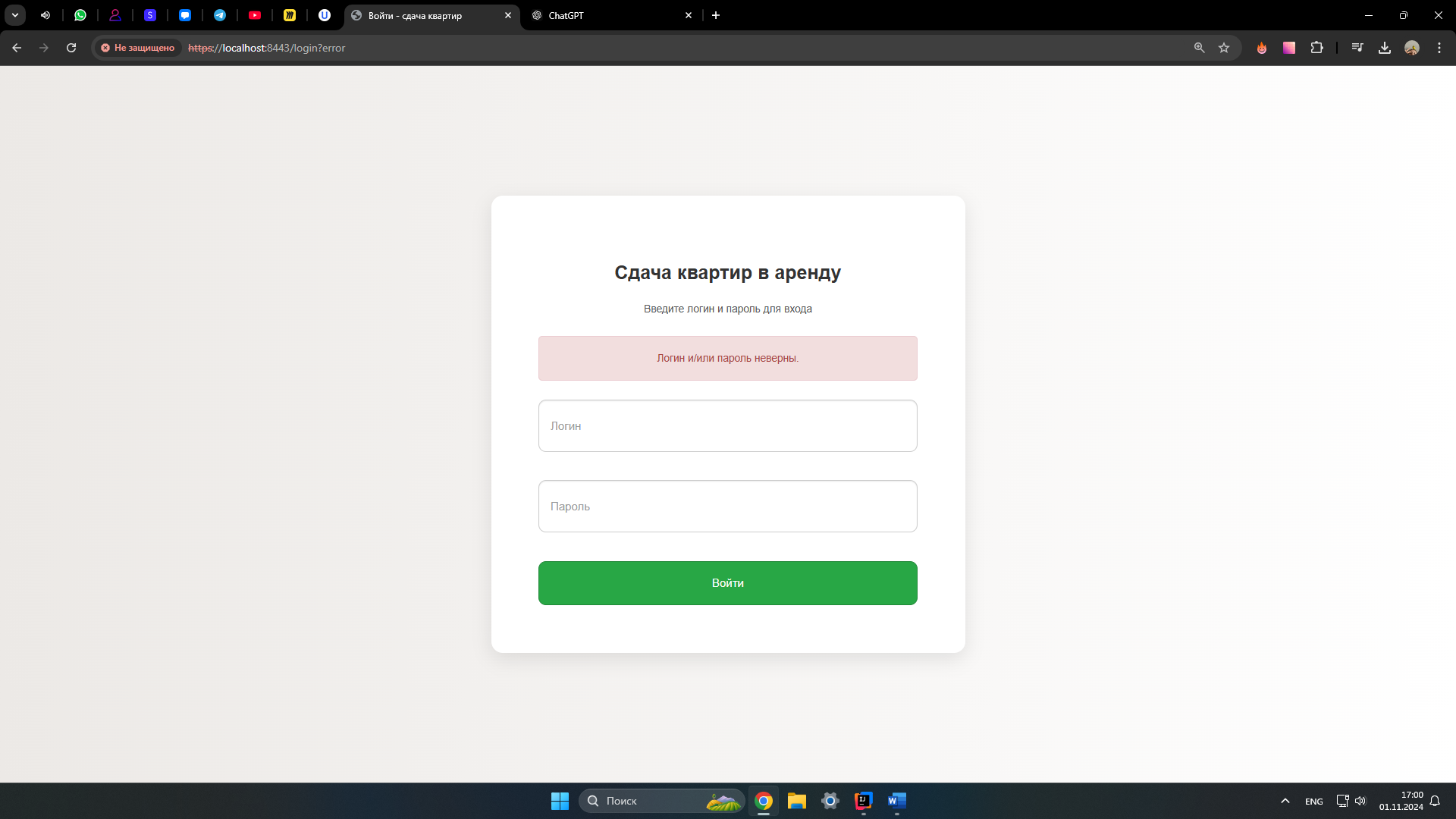
<!DOCTYPE html>  
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org"  
 xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity6">  
<head>  
 <meta charset="utf-8"/>  
 <meta http-equiv="X-UA-Compatible" content="IE=edge"/>  
 <meta name="viewport" content="width=device-width, initial-scale=1"/>  
  
 <title>Войти - сдача квартир</title>  
  
 <link href="../static/css/bootstrap.min.css" th:href="@{/css/bootstrap.min.css}" rel="stylesheet">  
 <script src="../static/js/jquery-3.7.1.slim.min.js" th:src="@{js/jquery-3.7.1.slim.min.js}"></script>  
 <script src="../static/js/bootstrap.js" th:src="@{js/bootstrap.js}"></script>  
  
 <style>  
 body {  
 background: linear-gradient(to right, #ece9e6, #ffffff); /\* Мягкий фон \*/  
 height: 100vh;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 font-family: 'Roboto', sans-serif;  
 margin: 0;  
 }  
 .login-form {  
 background: #ffffff;  
 padding: 50px; /\* Увеличенные отступы \*/  
 width: 100%;  
 max-width: 500px; /\* Значительно увеличенная ширина формы \*/  
 border-radius: 12px;  
 box-shadow: 0px 6px 20px rgba(0, 0, 0, 0.1);  
 text-align: center;  
 }  
 .form-heading {  
 font-size: 2rem; /\* Увеличенный заголовок \*/  
 margin-bottom: 20px;  
 color: #333;  
 font-weight: 600;  
 }  
 p {  
 font-size: 1.1rem;  
 color: #555;  
 }  
 .form-group {  
 margin-bottom: 30px; /\* Большие отступы между полями \*/  
 }  
 .form-control {  
 height: 55px; /\* Увеличенные поля ввода \*/  
 font-size: 1.2rem;  
 border-radius: 8px;  
 border: 1px solid #cccccc;  
 }  
 .btn-primary {  
 background-color: #28a745; /\* Новый цвет кнопки \*/  
 border-color: #218838;  
 width: 100%;  
 padding: 14px; /\* Увеличенная кнопка \*/  
 font-size: 1.2rem;  
 border-radius: 8px;  
 transition: background-color 0.3s ease;  
 }  
 .btn-primary:hover {  
 background-color: #218838;  
 }  
 .alert {  
 font-size: 1.1rem;  
 margin-top: 20px;  
 text-align: center;  
 }  
 </style>  
</head>  
<body>  
<div class="login-form">  
 <h2 class="form-heading">Сдача квартир в аренду</h2>  
 <p>Введите логин и пароль для входа</p>  
  
 <div th:if="${param.error}" class="alert alert-danger">Логин и/или пароль неверны.</div>  
 <div th:if="${param.logout}" class="alert alert-success">Вы успешно вышли.</div>  
  
 <form th:action="@{/login}" method="post">  
 <div class="form-group">  
 <input type="text" class="form-control" id="login" name="login" placeholder="Логин" required>  
 </div>  
 <div class="form-group">  
 <input type="password" class="form-control" id="pass" name="pass" placeholder="Пароль" required>  
 </div>  
 <button type="submit" class="btn btn-primary">Войти</button>  
 </form>  
</div>  
</body>  
</html>

1. Пример выполнения программы:

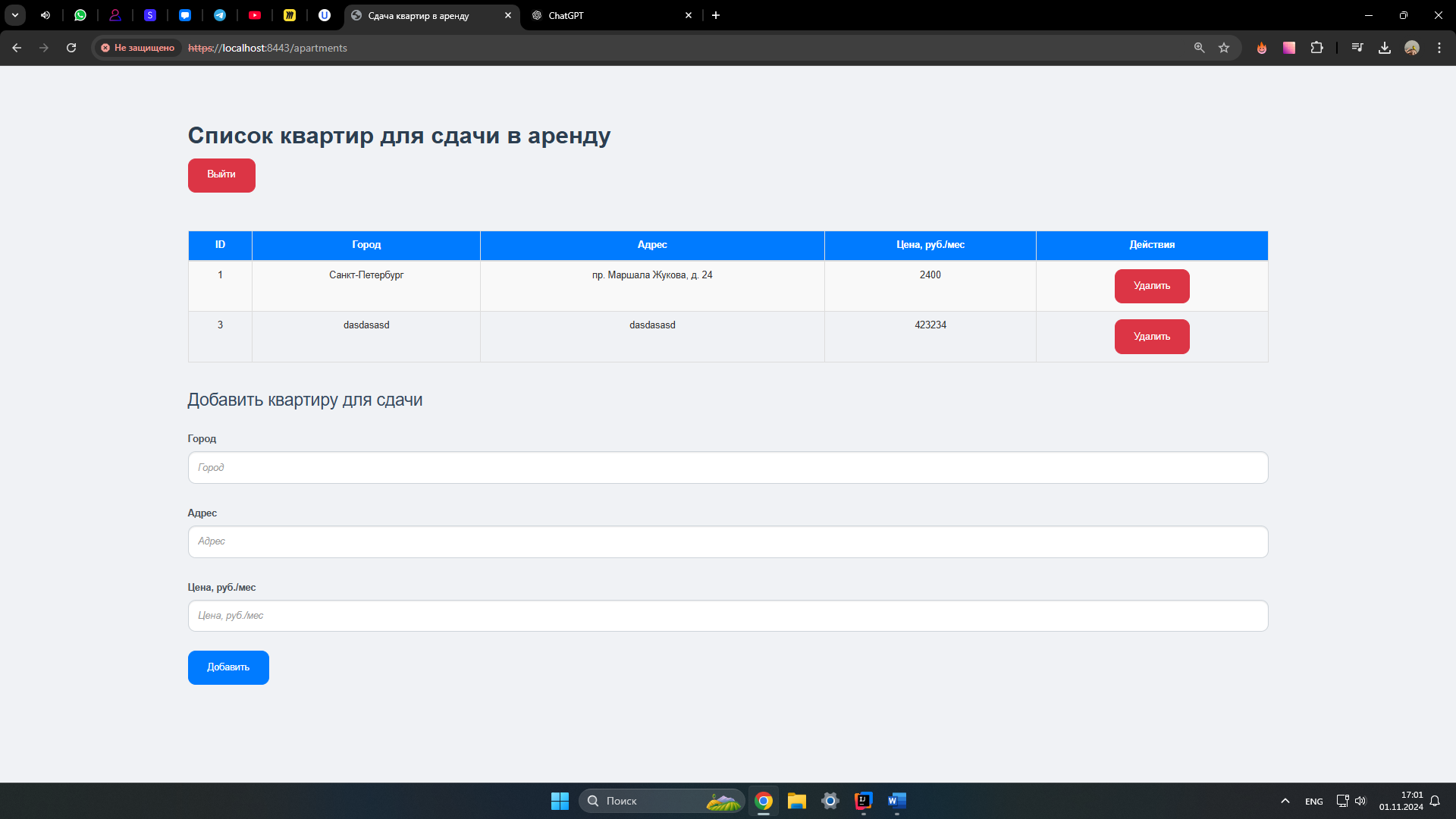
Форма логина:



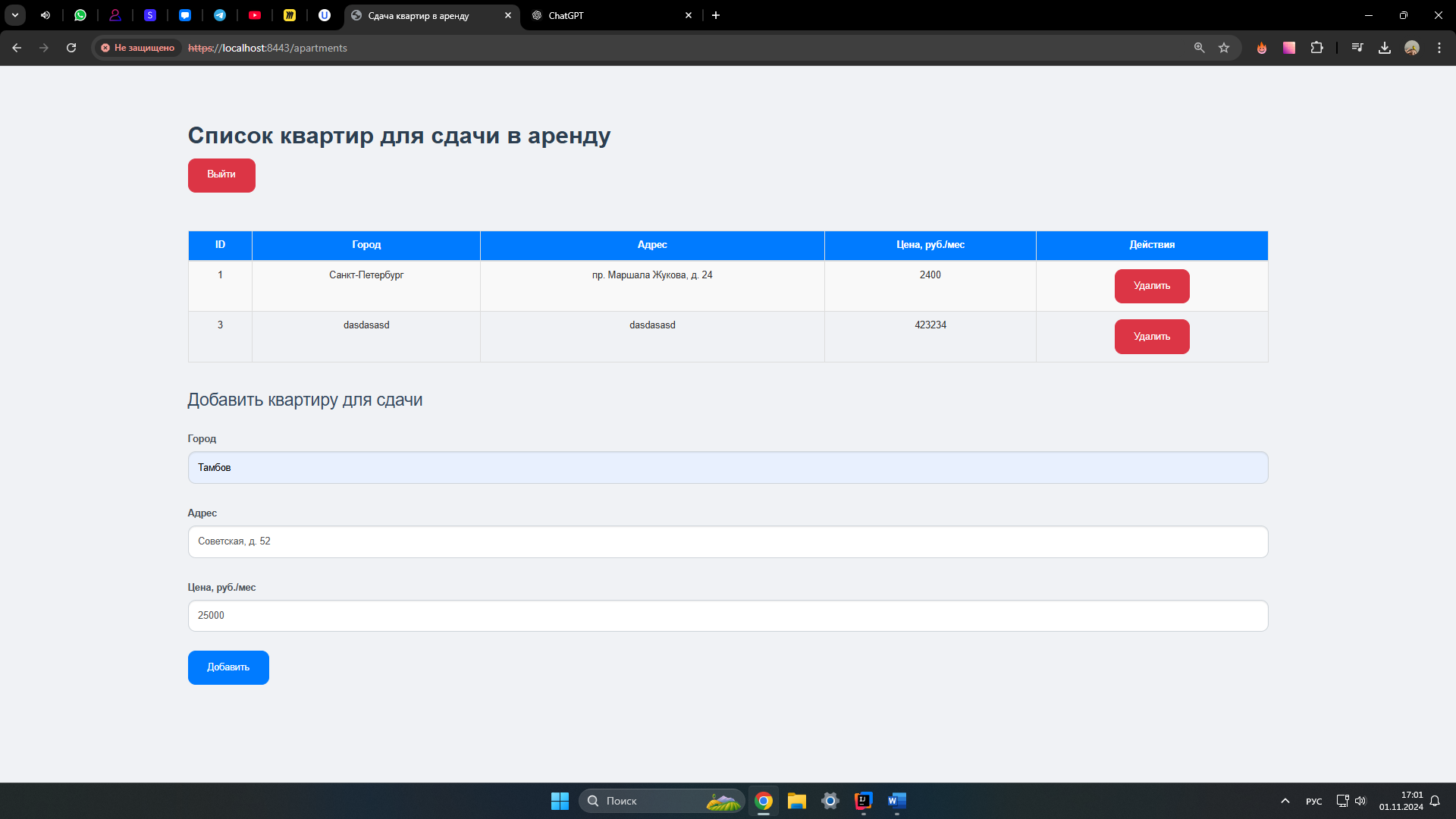
Ошибка входа:

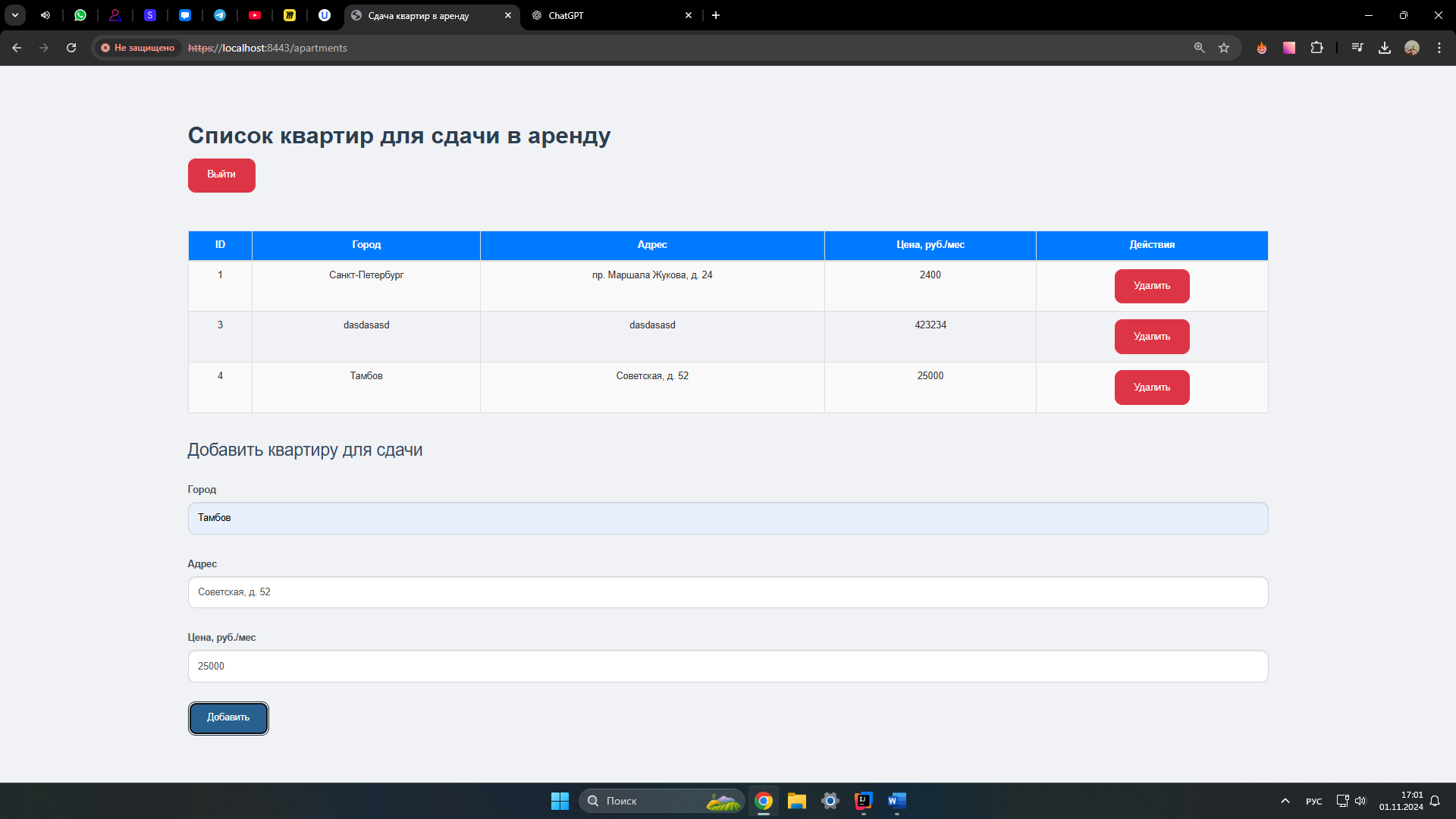


Успешный вход:



Добавление квартиры:





1. Вывод:

Изучены основы авторизации и аутентификации. Разработана форма логина.