Міністерство освіти і науки України

|  |
| --- |
| Український державний університет науки і технологій |
| Навчально-науковий інститут  «Український державний хіміко-технологічний університет» |

Факультет комп’ютерних наук та інженерії

Кафедра інформаційних систем

Лабораторна робота №3

З дисципліни Веб дизайн

Тема: «Обробка помилок доступу до даних»

Виконала:

студент гр. 3-КН-22

Дар’я ДОСОВА

Дніпро, 2024

Мета роботи: навчитись обробляти помилки отримані від сервера, та відображати інформацію для користувача.

Додайте на веб сторінку можливість видаляти дані з сервера

ресурсів;

3.2.3 На сервері додайте механізм перехоплення помилки від одного

мікросервіса і корректну передачу даних на інші вузли;

3.2.4 При отриманні помилки від серверу на сторінку зробіть

корректне відображення суті помилки (401 та 403);

Змінений код main.js:  
  
var JWT = "";  
  
// Компонент для авторизації  
Vue.component('auth-bar', {  
 template: `  
 <div>  
 <div class="field">  
 <label class="label">Username</label>  
 <div class="control">  
 <input type="login" class="input" v-model="login" />  
 </div>  
 </div>  
 <div>  
 <label class="label">Password</label>  
 <div class="field has-addons">  
 <div class="control is-expanded">  
 <input type="password" class="input" v-model="password" />  
 </div>  
 </div>  
 </div>  
 <input type="button" class="button" value="Login" @click="auth" />  
 <div v-if="token" class="token-display">  
 <strong>JWT:</strong> {{ token }}  
 </div>  
 </div>`,  
 methods: {  
 auth: function () {  
 this.$http  
 .get('auth/authorisation?login=' + this.login + '&password=' + this.password)  
 .then((response) => {  
 JWT = response.bodyText;  
 this.token = JWT;  
 });  
 },  
 },  
 data() {  
 return {  
 login: '',  
 password: '',  
 token: '',  
 };  
 },  
});  
  
// Компонент для завантаження даних  
Vue.component('persons-info-bar', {  
 props: ['persons'],  
 template: `  
 <div>  
 <input type="button" value="Load Data" @click="load" />  
 <div v-for="person in persons" :key="person.id">  
 <div>  
 <i>({{ person.id }})</i>  
 {{ person.lastName }} {{ person.firstName }}  
 <input type="button" value="X" @click="del"/>  
 </div>  
 </div>  
 </div>`,  
 methods: {  
 load: function () {  
 if (!JWT) {  
 alert("Please log in to access data.");  
 return;  
 }  
  
 const headers = {  
 'Content-Type': 'application/json',  
 'Authorization': 'Bearer ' + JWT,  
 };Vue.component('persons-info-bar', {  
 props: ['persons'],  
 template: `  
 <div>  
 <input type="button" value="Load Data" @click="load" />  
 <div v-for="person in persons" :key="person.id">  
 <div>  
 <i>({{ person.id }})</i>  
 {{ person.lastName }} {{ person.firstName }}  
 <input type="button" value="X" @click="del(person)" />  
 </div>  
 </div>  
 </div>`,  
 methods: {  
 load: function () {  
 if (!JWT) {  
 alert("Please log in to access data.");  
 return;  
 }  
  
 const headers = {  
 'Content-Type': 'application/json',  
 'Authorization': 'Bearer ' + JWT,  
 };  
  
 this.$http  
 .get('http://localhost:45000/classes/persons', { headers })  
 .then((response) => {  
 response.json().then((data) => {  
 this.persons.splice(0);  
 data.forEach((person) => this.persons.push(person));  
 });  
 })  
 .catch((error) => {  
 // Обробка помилок  
 if (error.status === 401) {  
 alert('Помилка завантаження даних,\nспочатку треба авторизуватись');  
 } else if (error.status === 500) {  
 alert('Помилка на сервері. Спробуйте пізніше.');  
 } else {  
 alert('Невідома помилка,\nповторіть пізніше');  
 }  
 });  
 },  
  
 del: function(person) {  
 if (!JWT) {  
 alert("Please log in to perform this action.");  
 return;  
 }  
  
 const headers = {  
 'Content-Type': 'application/json',  
 'Authorization': 'Bearer ' + JWT,  
 };  
  
 this.$http  
 .delete('http://localhost:45000/classes/persons/' + person.id, { headers })  
 .then(response => {  
 this.persons.splice(this.persons.indexOf(person), 1);  
 alert('Запис успішно видалено');  
 })  
 .catch(error => {  
 if (error.status === 401) {  
 alert('Помилка видалення даних,\nспочатку треба авторизуватись');  
 } else if (error.status === 403) {  
 alert('Для видалення недостатньо прав');  
 } else if (error.status === 401 && error.bodyText.includes("expired")) {  
 alert('Термін дії вашого токену сплив. Потрібно повторно авторизуватись');  
 } else {  
 alert('Невідома помилка,\nповторіть пізніше');  
 }  
 });  
 },  
 },  
 });  
  
  
// Створення додатку Vue  
new Vue({  
 el: '#app',  
 template: `  
 <div>  
 <auth-bar />  
 <persons-info-bar :persons="persons" />  
 </div>`,  
 data: {  
 persons: [],  
 },  
});

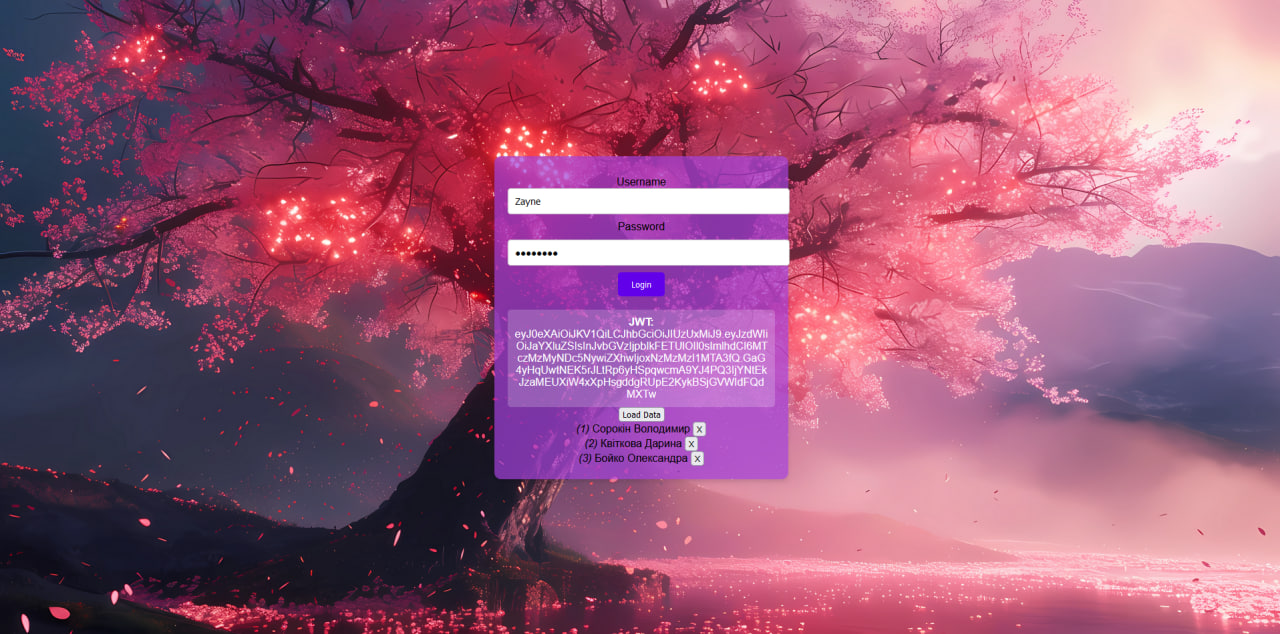
Змінений код index.html:

<!DOCTYPE html>  
<html lang="en">  
  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>DB Editor</title>  
 <!-- Підключення бібліотек Vue.js -->  
 <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>  
 <script src="https://cdn.jsdelivr.net/npm/vue-resource@1.5.1"></script>  
 <style>  
 body {  
 background-image: url('https://images.alphacoders.com/135/1350953.png');  
 background-size: cover;  
 background-position: center;  
 background-repeat: no-repeat;  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
 display: flex;  
 justify-content: center;  
 align-items: center;  
 height: 100vh;  
 overflow: hidden;  
 }  
  
 .container {  
 text-align: center;  
 height: auto;  
 display: flex;  
 justify-content: center;  
 align-items: center;  
 }  
  
 div.text {  
 display: inline-block;  
 width: 400px;  
 background-color: rgba(164, 66, 220, 0.7);  
 text-align: center;  
 border-radius: 10px;  
 overflow-wrap: break-word;  
 padding: 20px;  
 box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);  
 }  
  
 #app {  
 font-size: 14px;  
 color: white;  
 }  
  
 .field {  
 margin: 10px 0;  
 }  
  
 .input {  
 width: 100%;  
 padding: 10px;  
 font-size: 14px;  
 border-radius: 5px;  
 border: 1px solid #ccc;  
 }  
  
 .button {  
 background-color: #6200ea;  
 color: white;  
 padding: 10px 20px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 }  
  
 .button:hover {  
 background-color: #3700b3;  
 }  
  
 .token-display {  
 margin-top: 20px;  
 padding: 10px;  
 background-color: rgba(255, 255, 255, 0.2);  
 border-radius: 5px;  
 overflow-wrap: break-word;  
 word-wrap: break-word;  
 color: white;  
 }  
 </style>  
</head>  
  
<body>  
<div class="container">  
 <div class="text">  
 <!-- Головний контейнер Vue -->  
 <div id="app"></div>  
 </div>  
</div>  
  
<!-- Підключення файлу main.js -->  
<script>  
 var JWT = "";  
  
 // Компонент для авторизації  
 Vue.component('auth-bar', {  
 template: `  
 <div>  
 <div class="field">  
 <label class="label">Username</label>  
 <div class="control">  
 <input type="login" class="input" v-model="login" />  
 </div>  
 </div>  
 <div>  
 <label class="label">Password</label>  
 <div class="field has-addons">  
 <div class="control is-expanded">  
 <input type="password" class="input" v-model="password" />  
 </div>  
 </div>  
 </div>  
 <input type="button" class="button" value="Login" @click="auth" />  
 <div v-if="token" class="token-display">  
 <strong>JWT:</strong> {{ token }}  
 </div>  
 </div>`,  
 methods: {  
 auth: function () {  
 this.$http  
 .get('auth/authorisation?login=' + this.login + '&password=' + this.password)  
 .then((response) => {  
 JWT = response.bodyText;  
 this.token = JWT;  
 });  
 },  
 },  
 data() {  
 return {  
 login: '',  
 password: '',  
 token: '',  
 };  
 },  
 });  
  
 // Компонент для завантаження даних  
 Vue.component('persons-info-bar', {  
 props: ['persons'],  
 template: `  
 <div>  
 <input type="button" value="Load Data" @click="load" />  
 <div v-for="person in persons" :key="person.id">  
 <div>  
 <i>({{ person.id }})</i>  
 {{ person.lastName }} {{ person.firstName }}  
 <input type="button" value="X" @click="del(person)" />  
 </div>  
 </div>  
 </div>`,  
 methods: {  
 load: function () {  
 if (!JWT) {  
 alert("Please log in to access data.");  
 return;  
 }  
  
 const headers = {  
 'Content-Type': 'application/json',  
 'Authorization': 'Bearer ' + JWT,  
 };  
  
 this.$http  
 .get('http://localhost:45000/classes/persons', { headers })  
 .then((response) => {  
 response.json().then((data) => {  
 this.persons.splice(0);  
 data.forEach((person) => this.persons.push(person));  
 });  
 })  
 .catch((error) => {  
 // Обробка помилок  
 if (error.status === 401) {  
 alert('Помилка завантаження даних,\nспочатку треба авторизуватись');  
 } else if (error.status === 500) {  
 alert('Помилка на сервері. Спробуйте пізніше.');  
 } else {  
 alert('Невідома помилка,\nповторіть пізніше');  
 }  
 });  
 },  
  
 del: function(person) {  
 if (!JWT) {  
 alert("Please log in to perform this action.");  
 return;  
 }  
  
 const headers = {  
 'Content-Type': 'application/json',  
 'Authorization': 'Bearer ' + JWT,  
 };  
  
 this.$http  
 .delete('http://localhost:45000/classes/persons/' + person.id, { headers })  
 .then(response => {  
 this.persons.splice(this.persons.indexOf(person), 1);  
 alert('Запис успішно видалено');  
 })  
 .catch(error => {  
 if (error.status === 401) {  
 alert('Помилка видалення даних,\nспочатку треба авторизуватись');  
 } else if (error.status === 403) {  
 alert('Для видалення недостатньо прав');  
 } else if (error.status === 401 && error.bodyText.includes("expired")) {  
 alert('Термін дії вашого токену сплив. Потрібно повторно авторизуватись');  
 } else {  
 alert('Невідома помилка,\nповторіть пізніше');  
 }  
 });  
 },  
 },  
});  
  
  
 // Головний Vue екземпляр  
 new Vue({  
 el: '#app',  
 template: `  
 <div>  
 <auth-bar />  
 <persons-info-bar :persons="persons" />  
 </div>`,  
 data: {  
 persons: [],  
 },  
 });  
</script>  
</body>  
  
</html>

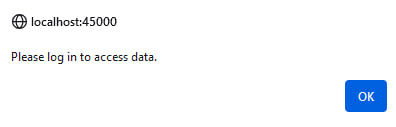
Механізм перехоплення і передач ExceptionsHandler :

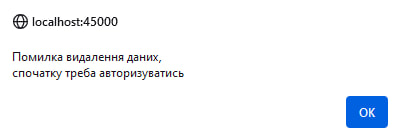
package org.example.rest;  
  
  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.ControllerAdvice;  
import org.springframework.web.bind.annotation.ExceptionHandler;  
import org.springframework.web.client.HttpClientErrorException;  
  
@ControllerAdvice  
public class ExceptionsHandler {  
 @ExceptionHandler(HttpClientErrorException.class)  
 protected ResponseEntity<Object> handleException(  
 HttpClientErrorException ex) {  
 return new ResponseEntity<>(ex, ex.getStatusCode());  
 }  
}

Результат виконання:  
1) При авторизації є тепер можливість видалити дані

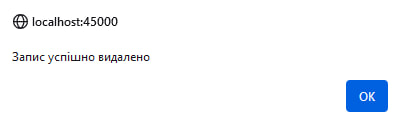


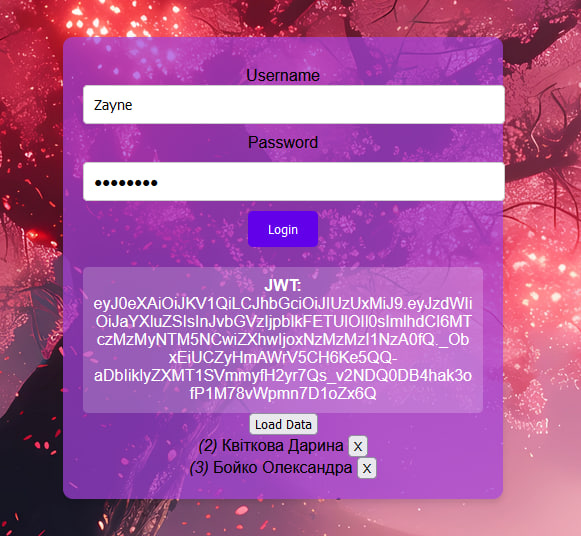
2) Якщо спробуємо отримати дані без авторизації





3) Видалення даних





Висновок : У результаті виконання цієї роботи було набуте розуміння основ обробки помилок, отриманих від сервера, а також засобів ефективного відображення відповідної інформації для користувача. Опанувавши механізми обробки помилок, вдалося забезпечити стабільність і зручність роботи додатка, адже користувач отримує чітке повідомлення про причину неполадки.