Міністерство освіти і науки України

Національний університет «Одеська політехніка»

Інститут комп'ютерних систем

Кафедра інформаційних систем

Курсова робота

З дисципліни: «Об’єктно-орієнтоване програмування»

Тема: «Система для організації виставок»

Варіант №24

Виконав:

студент групи АІ-231

Пейдж Антон Ендрю

Перевірив:

Годовиченко Микола Анатолійович

Одеса 2025

**1 ВСТУП**

У сучасному світі організація виставок мистецтва вимагає ефективного обліку експонатів, митців, локацій та виставок, які відбуваються у різний час і в різних містах. Враховуючи потребу у збереженні, систематизації та зручному доступі до інформації про культурні події, постає необхідність створення автоматизованої системи для адміністрування таких заходів.

Метою цієї курсової роботи є розробка програмного забезпечення, яке дозволяє створювати та керувати виставками, обліковувати експонати та їх авторів, призначати локації проведення заходів, а також відображати статистику. У системі реалізовано підтримку ролей користувачів, автентифікацію за логіном і паролем, авторизацію за допомогою JWT, а також вхід через обліковий запис Google (OAuth2).

Для реалізації програмного продукту було використано мову програмування Java з використанням фреймворку Spring Boot, базу даних PostgreSQL та архітектуру REST API. Додаток забезпечує зручний інтерфейс для взаємодії через HTTP-запити, а також можливість віддаленого доступу до розгорнутої системи на хмарній платформі Render.

Результатом виконання курсової роботи є готовий веб-застосунок, що забезпечує повний цикл керування виставками з урахуванням вимог до безпеки та масштабованості.

1. **АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ**

**2.1 Постановка задачі**

Сфера організації виставок охоплює діяльність музеїв, галерей, культурних центрів та приватних ініціатив, які проводять заходи з експонування творів мистецтва. Для ефективного управління такими подіями необхідна система, яка дозволяє обліковувати митців, їхні роботи, місця проведення виставок та взаємозв'язки між ними.

Основними проблемами у сфері організації виставок є:

* Відсутність централізованого обліку експонатів та їх авторів
* Складність відстеження переміщення експонатів між локаціями
* Неможливість швидко отримати статистичну інформацію про виставки
* Відсутність зручного інструменту для планування майбутніх виставок
* Складність координації між різними учасниками процесу (кураторами, митцями, адміністрацією)

Розроблювана система має вирішити ці проблеми шляхом створення єдиної платформи для управління всіма аспектами виставкової діяльності.

**2.2 Аналіз бізнес-процесів**

Основні сценарії використання системи:

Сценарій 1: Організація нової виставки

1. Адміністратор створює нову виставку, вказуючи назву, дати проведення та локацію
2. Куратор додає експонати до виставки, вибираючи з наявних у системі
3. Система автоматично формує список учасників (митців) виставки
4. Генерується звіт про виставку з усіма деталями

Сценарій 2: Управління колекцією

1. Адміністратор додає нового митця до системи
2. Додаються експонати цього митця з описом та роком створення
3. Експонати можуть бути призначені на різні виставки
4. Система відслідковує історію участі експонатів у виставках

Сценарій 3: Аналіз та звітність

1. Користувач запитує статистику по виставках
2. Система надає інформацію про найпопулярніші локації
3. Можна отримати список експонатів без виставок
4. Формуються звіти про активність митців за країнами

**2.3 Модель користувачів системи**

Система передбачає роботу наступних категорій користувачів:

Адміністратор системи:

* Повний доступ до всіх функцій
* Управління користувачами та їх правами
* Створення та редагування всіх сутностей
* Доступ до статистичних звітів

Куратор виставок:

* Створення та редагування виставок
* Призначення експонатів на виставки
* Перегляд інформації про митців та експонати
* Формування звітів по виставках

Оглядач (гість):

* Перегляд інформації про поточні виставки
* Перегляд каталогу експонатів та митців
* Доступ до публічної статистики

**2.4 Опис основних сутностей системи**

Artist (Митець) — містить інформацію про ім'я та країну митця. Один митець може бути автором багатьох експонатів та брати участь у різних виставках.

Exhibit (Експонат) — включає назву, опис, рік створення та посилання на автора. Експонат може бути частиною кількох виставок протягом свого існування.

Location (Локація) — описує місце проведення виставки з назвою та повною адресою. В одній локації може проходити кілька виставок у різний час.

Exhibition (Виставка) — визначає культурну подію з конкретною назвою, датами початку та завершення, а також вказує локацію проведення.

ExhibitionEntry (Участь експоната у виставці) — проміжна сутність, що відображає зв'язок між конкретним експонатом та виставкою, дозволяючи реалізувати відношення many-to-many.

**2.5 Функціональні вимоги до системи**

Система повинна забезпечувати:

* Створення, редагування, перегляд та видалення всіх основних сутностей
* Встановлення зв'язків між експонатами та виставками
* Пошук та фільтрацію даних за різними критеріями
* Генерацію статистичних звітів
* Аутентифікацію та авторизацію користувачів
* RESTful API для взаємодії з фронтенд-додатками
* Безпечне зберігання даних у реляційній базі даних

Таким чином, система дозволяє зберігати, редагувати, переглядати та пов'язувати всі необхідні компоненти виставкової діяльності, що забезпечує цілісне управління інформацією у цій предметній області.

**3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

У цьому розділі подано загальну архітектуру розробленої інформаційної системи для організації виставок. Основну увагу приділено класам доменної моделі, їх взаємозв’язкам та логіці взаємодії між сутностями. Також розглянуто особливості реалізації багатокористувацького доступу з урахуванням автентифікації через JWT і OAuth2.

**3.1 Структура класів та їх зв’язки**

На основі функціональних вимог і логіки предметної області були визначені такі основні сутності:

* Artist (Митець) — зберігає інформацію про автора творів мистецтва, включаючи ім’я та країну походження. Один митець може мати багато експонатів.
* Exhibit (Експонат) — містить дані про твір мистецтва (назву, опис, рік створення) і має посилання на автора.
* Location (Локація) — описує фізичне місце проведення виставки з назвою та адресою.
* Exhibition (Виставка) — визначає саму подію, має назву, дати початку й завершення, а також пов’язується з локацією.
* ExhibitionEntry (Участь у виставці) — проміжна таблиця для реалізації зв’язку «багато-до-багатьох» між виставками та експонатами.

На діаграмі класів (Рисунок 1) показано, що:

Кожен Exhibit належить одному Artist.

Кожен Exhibit може брати участь у багатьох Exhibition, а одна Exhibition включає багато Exhibit через ExhibitionEntry.

Кожна Exhibition проводиться в певній Location.

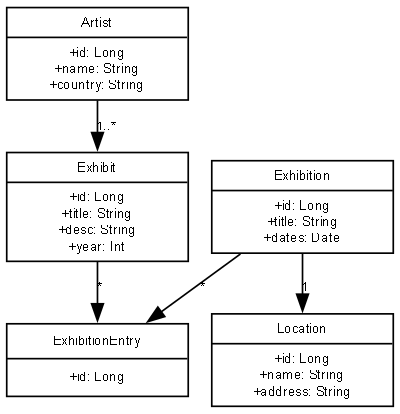


Рисунок 1 – Діаграма класів системи для організації виставок

**3.2 Модель користувачів та безпека доступу**

У системі реалізовано багаторівневу авторизацію:

* Гості можуть переглядати публічні дані (поточні виставки, митців, експонати).
* Зареєстровані користувачі отримують доступ до створення й редагування даних відповідно до ролі.

Для автентифікації реалізовано два механізми:

1. Вхід за логіном і паролем із видачею JWT-токену.
2. OAuth2-автентифікація через Google.

Це дозволяє інтегрувати систему з сучасними вебфреймворками та гарантує безпечний захист даних і ролей.

**3.3 Взаємодія компонентів**

* Система має тришарову архітектуру:
* Контролери приймають HTTP-запити.
* Сервіси обробляють логіку.
* Репозиторії взаємодіють із базою даних через Spring Data JPA.

Взаємодія REST-контролерів з клієнтом реалізована через JSON. Дані передаються у вигляді DTO-об’єктів, що полегшує розробку фронтенду та забезпечує захист від надлишкового доступу до сутностей.

**3.4 REST API**

У системі реалізовано 25 REST-запитів згідно з технічним завданням. Кожен запит відповідає за певну операцію над сутністю або за обчислення звітної інформації.

У таблиці нижче наведено всі реалізовані REST-запити, їхній опис, метод доступу, приклад URL та наявність авторизації через JWT:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Метод | Endpoint | Призначення |
| 1 | POST | /api/artists | Додати митця |
| 2 | GET | /api/artists | Отримати всіх митців |
| 3 | PUT | /api/artists/{id} | Оновити дані митця |
| 4 | DELETE | /api/artists/{id} | Видалити митця |
| 5 | POST | /api/exhibits | Додати експонат |
| 6 | GET | /api/exhibits | Отримати всі експонати |
| 7 | PUT | /api/exhibits/{id} | Оновити експонат |
| 8 | DELETE | /api/exhibits/{id} | Видалити експонат |
| 9 | POST | /api/exhibitions | Додати виставку |
| 10 | GET | /api/exhibitions | Отримати всі виставки |
| 11 | PUT | /api/exhibitions/{id} | Оновити виставку |
| 12 | DELETE | /api/exhibitions/{id} | Видалити виставку |
| 13 | POST | /api/locations | Додати локацію |
| 14 | GET | /api/locations | Отримати всі локації |
| 15 | PUT | /api/locations/{id} | Оновити локацію |
| 16 | POST | /api/exhibition-entries | Призначити експонат на виставку |
| 17 | DELETE | /api/exhibition-entries/{id} | Видалити експонат з виставки |
| 18 | GET | /api/exhibition-entries/by-exhibition/{id} | Отримати експонати певної виставки |
| 19 | GET | /api/exhibitions/artist/{artistId} | Отримати виставки митця |
| 20 | GET | /api/exhibits/byYear/{year} | Отримати експонати певного року |
| 21 | GET | /api/exhibitions/statistics | Статистика по виставках |
| 22 | GET | /api/artists/by-country?country=UA | Митці за країною |
| 23 | GET | /api/locations/top | Локації з найбільше виставок |
| 24 | GET | /api/exhibits/without-exhibition | Отримати експонати без виставки |
| 25 | GET | /api/exhibitions/current | Поточні виставки |

Після деплою запити можна тестувати через Postman або curl за адресою:https://kursova-6pe9.onrender.com

**3.5 Діаграма компонентів**

Для демонстрації структури додатку створено UML-діаграму компонентів, яка відображає залежності між головними шарами застосунку:

* Controller layer – приймає HTTP-запити.
* Service layer – містить бізнес-логіку.
* Repository layer – взаємодіє з базою даних через JPA.
* Security layer – забезпечує автентифікацію та авторизацію (JWT, OAuth2).
* PostgreSQL – зберігання даних.
* Frontend (умовний) – взаємодія з REST API.

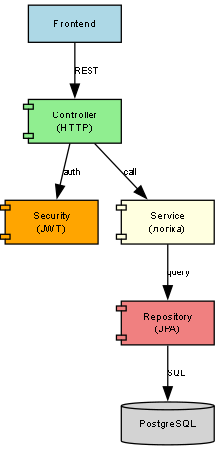


Рисунок 2 – Діаграма компонентів системи

**3.6 Діаграма послідовності (автентифікація)**

На діаграмі нижче зображено процес логіну користувача через логін/пароль із видачею JWT-токену.

1. Користувач надсилає POST-запит /auth/login.
2. Контролер передає дані в AuthenticationService.
3. Якщо дані вірні — генерується JWT.
4. Токен повертається клієнту.

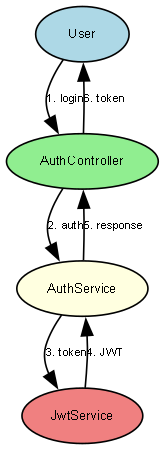


Рисунок 3 – Діаграма послідовності авторизації через логін/пароль

**4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ**

У цьому розділі представлено технічну реалізацію інформаційної системи для організації виставок. Система побудована з використанням Spring Boot та PostgreSQL, а також реалізує сучасні механізми безпеки: автентифікацію на основі JWT-токенів та авторизацію через Google OAuth2. Проєкт має багаторівневу архітектуру: модель → репозиторій → сервіс → контролер.

**4.1 Середовище розробки**

* Мова програмування: Java 17
* Фреймворк: Spring Boot 3.4.4
* База даних: PostgreSQL (хостинг на Render)
* Система збірки: Maven
* Інструменти: IntelliJ IDEA, Postman, GitHub, Render
* Безпека: Spring Security, JWT, OAuth2 (Google)
* Frontend: не реалізовувався, доступ через REST API

**4.2 Моделі даних (Entity-класи та DTO)**

Кожна сутність у системі зберігається як клас з анотаціями JPA. Для зручності взаємодії з фронтендом також використовуються DTO-класи.

Приклад entity-класу Artist:

@Entity

public class Artist {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String country;

}

DTO для автентифікації:

@Data

public class AuthRequest {

private String username;

private String password;

}

@Data

@AllArgsConstructor

public class AuthResponse {

private String token;

}

**4.3 Репозиторії**

Для доступу до бази даних використовуються інтерфейси JpaRepository.java

@Repository

public interface ArtistRepository extends JpaRepository<Artist, Long> {

List<Artist> findByCountry(String country);

}

@Repository

public interface AppUserRepository extends JpaRepository<AppUser, Long> {

Optional<AppUser> findByUsername(String username);

}

**4.4 Сервіси**

Сервіси реалізують основну бізнес-логіку застосунку.

@Service

public class ArtistService {

private final ArtistRepository artistRepository;

public ArtistService(ArtistRepository artistRepository) {

this.artistRepository = artistRepository;

}

public List<Artist> getAll() {

return artistRepository.findAll();

}

public Artist create(Artist artist) {

return artistRepository.save(artist);

}

}

**4.5 Контролери**

Контролери приймають HTTP-запити та передають їх у відповідний сервіс.

@RestController

@RequestMapping("/api/artists")

public class ArtistController {

private final ArtistService artistService;

public ArtistController(ArtistService artistService) {

this.artistService = artistService;

}

@GetMapping

public List<Artist> getAll() {

return artistService.getAll();

}

@PostMapping

public Artist create(@RequestBody Artist artist) {

return artistService.create(artist);

}

}

Контролер автентифікації:

@RestController

@RequestMapping("/auth")

public class AuthController {

private final AuthenticationService authService;

public AuthController(AuthenticationService authService) {

this.authService = authService;

}

@PostMapping("/register")

public AuthResponse register(@RequestBody AuthRequest request) {

return authService.register(request);

}

@PostMapping("/login")

public AuthResponse login(@RequestBody AuthRequest request) {

return authService.authenticate(request);

}

}

**4.6 Конфігурація безпеки**

Система підтримує автентифікацію з JWT і OAuth2. Основні налаштування описані в SecurityConfig.

@Configuration

public class SecurityConfig {

private final JwtAuthFilter jwtAuthFilter;

private final CustomUserDetailsService userDetailsService;

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

return http

.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth

.requestMatchers("/auth/\*\*", "/oauth2/\*\*", "/success").permitAll()

.anyRequest().authenticated()

)

.oauth2Login(oauth -> oauth.defaultSuccessUrl("/success", true))

.sessionManagement(session -> session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))

.addFilterBefore(jwtAuthFilter, UsernamePasswordAuthenticationFilter.class)

.build();

}

}

**4.7 Тестування API**

API доступне на: https://kursova-6pe9.onrender.com

Всі запити до /api/\*\* потребують JWT-токена:

Authorization: Bearer <eyJhbGciOiJIUzM4NCJ9.eyJzdWIiOiJ0ZXN0dXNlcjEiLCJpYXQiOjE3NDkzMzA4NDgsImV4cCI6MTc0OTQxNzI0OH0.w-2O4DjlhACzNNatEdwENjANzZHcpzfqSKQNuAq\_fWkPdvTMq59BU04JONm0VOlA>

**5 ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ**

**5.1 Мета та методика тестування**

Метою тестування є перевірка працездатності реалізованої інформаційної системи для організації виставок, зокрема перевірка:

* усіх основних функціональних можливостей (CRUD-операції);
* безпечної автентифікації через JWT і Google OAuth2;
* роботи API в продакшн-середовищі (https://kursova-6pe9.onrender.com);
* правильності статус-кодів та обробки помилок.

Методи тестування:

* ручне тестування через Postman;
* перевірка HTTP-відповідей (200, 201, 401, 400 тощо);
* аналіз логів виконання в середовищі Render;
* перевірка поведінки при некоректних даних (валідація).

**5.2 Інструменти тестування**

Для тестування застосовувались такі інструменти:

* Postman — основний інструмент для перевірки REST API;
* браузер — для перевірки OAuth2-авторизації;
* Render — продакшн-середовище для розгортання та тестування;
* IntelliJ IDEA — логічне налагодження та дебаг.

**5.3 Тестування авторизації та автентифікації**

5.3.1 Реєстрація нового користувача

POST /auth/register

Використовується для створення нового облікового запису. У відповідь очікується JWT-токен.



Рисунок 4 – Реєстрація нового користувача через Postman

5.3.2 Авторизація користувача

POST /auth/login

Повертає JWT-токен після успішної авторизації.



Рисунок 5 – Авторизація користувача з отриманням токена

**5.4 Тестування захищених запитів**

5.4.1 Доступ з токеном

GET /api/artists з заголовком Authorization: Bearer <jwt>



Рисунок 6 – Отримання митців із JWT-токеном

5.4.2 Доступ без токена

Спроба виконати GET /api/artists без токена призводить до 401 Unauthorized.

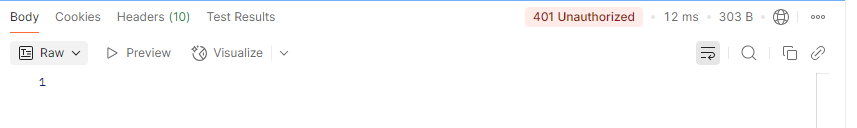


Рисунок 7 – Відмова в доступі без авторизації

**5.5 Тестування CRUD-операцій**

5.5.1 Створення нового митця

POST /api/artists

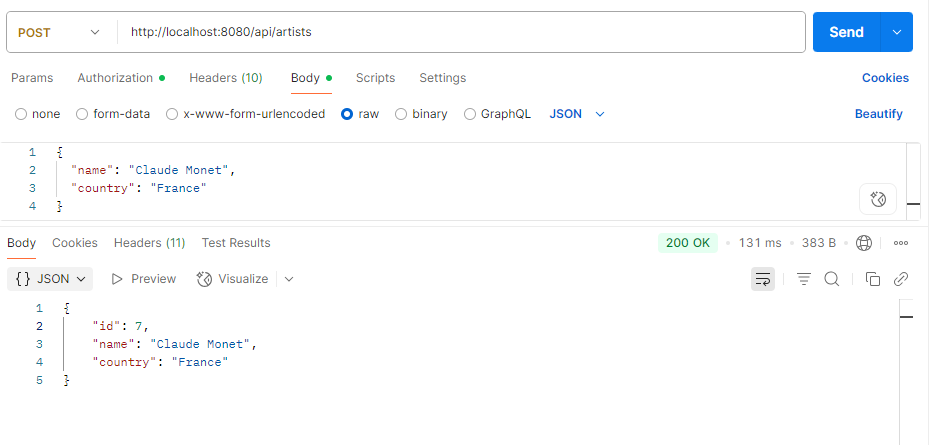


Рисунок 8 – Створення нового митця через Postman

5.5.2 Оновлення митця

PUT /api/artists/{id}



Рисунок 9 – Оновлення даних про митця

5.5.3 Видалення митця

DELETE /api/artists/{id}

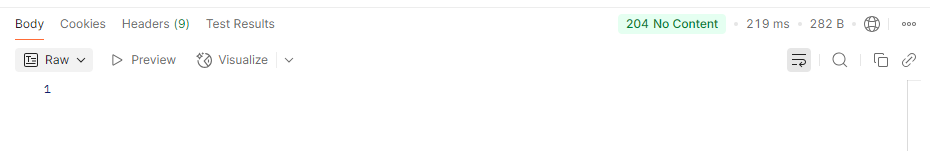


Рисунок 10 – Видалення митця з бази

**5.6 Тестування фільтрів та звітів**

5.6.1 Отримання експонатів за роком

GET /api/exhibits/byYear/2020

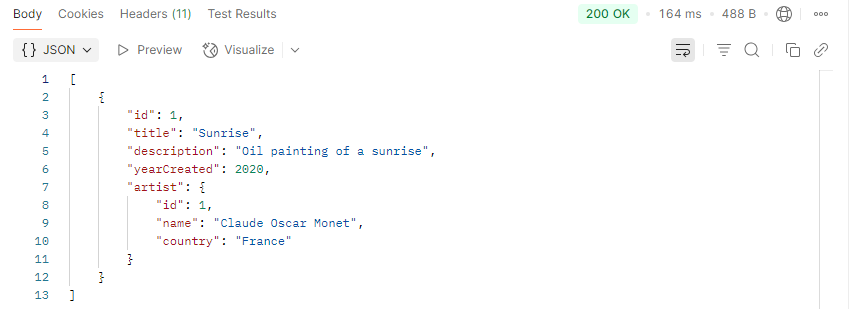


Рисунок 11 – Вивід експонатів за роком створення

5.6.2 Отримання виставок митця

GET /api/exhibitions/byArtist/{id}

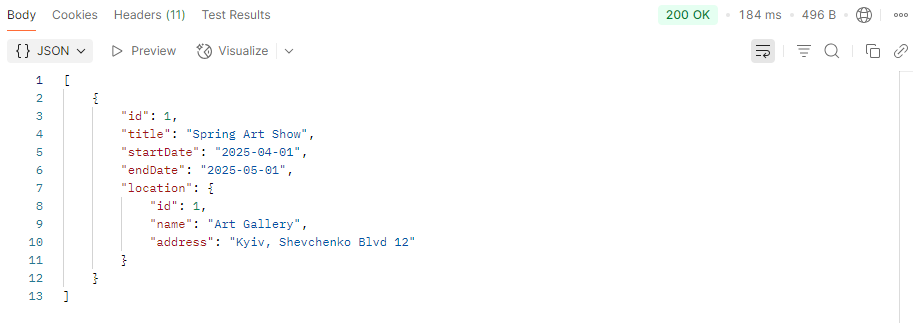


Рисунок 12 – Виставки, у яких бере участь обраний митець

5.6.3 Поточні виставки

GET /api/exhibitions/current



Рисунок 13 – Поточні активні виставки

**5.7 Тестування авторизації через OAuth2**

5.7.1 Ініціація входу через Google

GET /oauth2/authorization/google

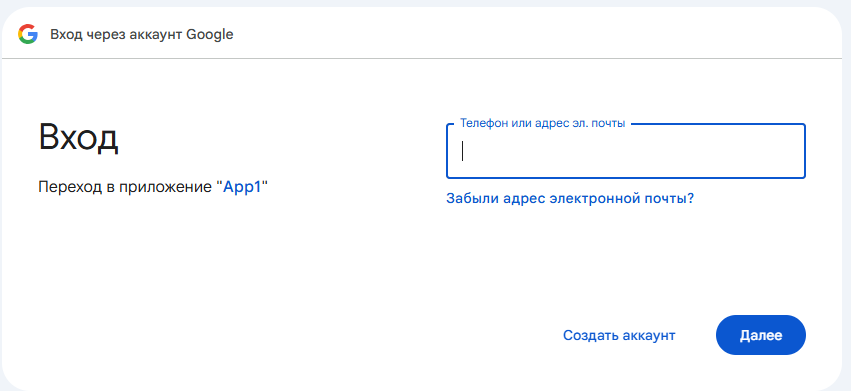


Рисунок 14 – Перенаправлення на авторизацію Google

5.7.2 Успішний вхід

Після авторизації користувач потрапляє на сторінку /success.

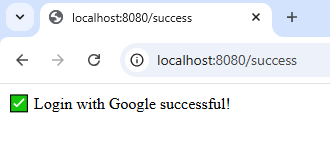


Рисунок 15 – Повідомлення про успішний вхід через Google

**5.8 Виявлені помилки і способи їх усунення**

|  |  |  |
| --- | --- | --- |
| № | Помилка | Спосіб усунення |
| 1 | 401 Unauthorized при запитах до /api/\*\* без токена | Додано JWT в заголовок запиту |
| 2 | no main manifest attribute, in app.jar під час деплою на Render | Вказано mainClass у pom.xml і перегенеровано .jar через Spring Boot Plugin |
| 3 | method not found під час запуску проєкту | Встановлено плагін Lombok, перезібрано проект |
| 4 | Dependency 'lombok' not found у pom.xml | Додано правильну версію lombok та оновлено Maven залежності |
| 5 | Експонати не зʼявлялися у виставках | Неправильно заповнені ExhibitionEntry, додано в data.sql коректні ID |

**5.9 Тестування у production-середовищі**

Розгорнута система доступна за посиланням:

https://kursova-6pe9.onrender.com

Було протестовано всі запити. Всі функції працюють коректно.

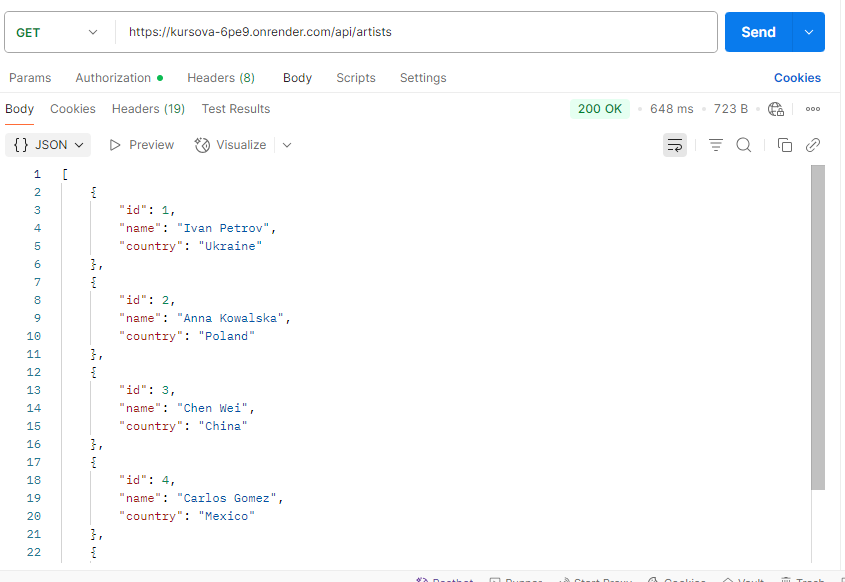


Рисунок 16 – Створення митця на production-сервері

**5.10 Результати перевірки основних функцій**

|  |  |  |
| --- | --- | --- |
| **Найменування функціоналу** | **Результат** | **Коментар** |
| Реєстрація та авторизація (JWT) | Працює | Токен генерується, доступ надається |
| Авторизація через Google (OAuth2) | Працює | Після входу перенаправлення на /success |
| CRUD-операції над митцями | Працюють | Додавання, редагування, видалення, перегляд |
| Робота з виставками та експонатами | Працює | Зв’язки між сутностями реалізовано |
| Захист ендпоінтів | Працює | Доступ без токена неможливий (401), з токеном – успішно |
| Робота в продакшн-середовищі | Працює | Успішне розгортання та доступність з будь-якого пристрою |

**ВИСНОВКИ**

У результаті виконання курсової роботи на тему «Система для організації виставок» було реалізовано повноцінний веб-застосунок з підтримкою REST-архітектури, автентифікації користувачів, фільтрації та звітності. Розроблена система дозволяє ефективно обліковувати митців, їхні експонати, виставки та локації проведення, а також формувати статистику щодо участі експонатів у подіях.

Основні технічні досягнення проєкту включають створення п'яти пов'язаних сутностей з правильно налаштованими реляційними зв'язками типу один-до-багатьох та багато-до-багатьох. Було реалізовано двадцять п'ять REST-запитів для повноцінної взаємодії з системою, що забезпечує виконання всіх необхідних CRUD-операцій над даними.

Особливу увагу приділено безпеці системи через впровадження JWT-автентифікації для захищеного доступу до API та підтримки OAuth2-авторизації через Google, що забезпечує гнучкість та зручність входу користувачів. Усі компоненти системи було протестовано з використанням Postman, що підтвердило коректність роботи всіх реалізованих функцій.

Проєкт було успішно задеплоєно на хмарну платформу Render за адресою https://kursova-6pe9.onrender.com, що демонструє готовність системи до практичного використання.

Практичне значення розробленої системи полягає в можливості її застосування як основи для реальних виставкових платформ, внутрішніх адміністративних панелей музеїв та галерей, або як навчального прикладу для вивчення сучасних технологій розробки на Spring Boot.

Перспективи подальшого розвитку системи включають додавання фронтенд-інтерфейсу на базі React або Vue.js, можливість завантаження зображень експонатів, міжнародну локалізацію інтерфейсу та розмежування прав доступу за ролями користувачів.

Загалом, розроблений проєкт повністю відповідає технічному завданню та охоплює всі вимоги курсової роботи з об'єктно-орієнтованого програмування. Завдяки обраній архітектурі система є масштабованою та може бути легко адаптована під потреби реальних культурних установ.

**ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Spring Boot Documentation – Spring Framework Reference Documentation. URL: https://docs.spring.io/spring-boot/docs/current/reference/html/ (дата звернення: 15.11.2024).
2. PostgreSQL Documentation – The World's Most Advanced Open Source Relational Database. URL: https://www.postgresql.org/docs/ (дата звернення: 20.11.2024).
3. JWT – JSON Web Token Introduction and Specification. URL: https://jwt.io/introduction (дата звернення: 25.11.2024).
4. Spring Security Reference Documentation – Authentication and Authorization for Java Applications. URL: https://docs.spring.io/spring-security/reference/index.html (дата звернення: 28.11.2024).
5. Google Identity Platform – OAuth 2.0 Login. URL: https://developers.google.com/identity (дата звернення: 02.12.2024).
6. Render.com – Cloud Hosting for Web Services. URL: https://render.com/ (дата звернення: 05.12.2024).
7. Postman – API Platform for Building and Testing APIs. URL: https://www.postman.com/ (дата звернення: 10.12.2024).
8. Годовиченко М. А. Методичні вказівки до виконання курсової роботи з дисципліни «Об'єктно-орієнтоване програмування». Одеса : Національний університет «Одеська Політехніка», 2024. 59 с.
9. ДСТУ 8302:2015. Бібліографічне посилання. Загальні положення та правила складання. [Чинний від 2015-07-01]. Вид. офіц. Київ, 2015. 17 с.