



Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної
підготовки

Випускна кваліфікаційна робота бакалавра

**Інформаційна система управління ліцензованим аудіо контентом
(на прикладі Spotify)**

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проєкту

№	П.І.Б.	Група
1	Дверницька О. В.	КН-П-211
2	Михайлюк М. О.	КН-Д-211
3	Небесна О. В.	КН-П-211
4	Рибак Д. О.	КН-П-211
5	Хіміч А. І.	КН-Д-211
6	Щербаков Д. Е.	КН-Д-211

Автор звіту

_____ Рибак Дмитро Олексійович

Науковий керівник	Доцент кафедри ІТ та ФП, к.т.н., доцент Данилець Є. В.
-------------------	---

АНОТАЦІЯ

УДК 004.9

P-49

Рибак Д. О. Інформаційна система управління ліцензованим аудіо контентом (на прикладі Spotify): випускна кваліфікаційна робота на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп'ютерні науки». - Одеса: ОТУШ, 2025 - 14 с.

Робота присвячена дослідженню, розробці та впровадженню клієнт-серверної інформаційної системи для потокового прослуховування аудіоконтенту. Основним функціональним призначенням системи є забезпечення доступу користувачів до цифрової музики у режимі реального часу через вебінтерфейс або мобільний застосунок.

Актуальність проєкту зумовлена глобальними тенденціями цифровізації медіаконтенту, стрімким зростанням аудиторії стримінгових сервісів, а також необхідністю створення адаптивних, безпечних і масштабованих мультимедійних платформ. Сучасні вимоги до зручності користування, швидкодії та персоналізації контенту обумовлюють потребу у впровадженні інноваційних технологій в архітектуру таких систем.

Об'єктом дослідження виступає багаторівнева інформаційна система для аудіострімінгу, що об'єднує різноманітні клієнтські середовища (веб та мобільні пристрої) через єдиний програмний інтерфейс. Предмет дослідження — методи організації взаємодії між клієнтською та серверною частинами системи для забезпечення стабільної, захищеної та ефективною передачі медіаконтенту.

У якості задач дослідження було встановлене наступне:

- визначити оптимальні технології для реалізації клієнтської та серверної частин системи;
- розробити інтерфейси користувача з урахуванням принципів UI/UX-дизайну;
- реалізувати систему реєстрації, автентифікації, завантаження та прослуховування треків, управління плейлистами;
- впровадити механізми захисту персональних даних і авторизації через JWT-токени;
- провести тестування працездатності системи в різних клієнтських середовищах.

Методи дослідження включають системний аналіз, побудову компонентної архітектури клієнтської частини на основі парадигми об'єктно-орієнтованого програмування, клієнт-серверне моделювання, а також аналіз користувацького досвіду (UI/UX).

Під час підготовки звіту використано 14 джерел – як друкованих, так і електронних.

ЗМІСТ

ВСТУП_____	3
Технічне завдання до проекту_____	4
РОЗДІЛ 1. Загальна характеристика системи_____	6
РОЗДІЛ 2. Реалізація серверної частини_____	8
РОЗДІЛ 3. Результати випробувань_____	11
ВИСНОВКИ_____	13
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ_____	14

ВСТУП

У сучасному цифровому середовищі інформаційні технології стають основою більшості процесів передачі, збереження та обробки даних. Зростання популярності потокових сервісів відображає зміну підходів до споживання медіа контенту — користувачі очікують миттєвого доступу до аудіо та відео будь-де й будь-коли. За оцінками аналітичних агентств, частка аудіострімінгових сервісів у загальному медіаспоживанні продовжує стабільно зростати, стимулюючи розвиток відповідних програмних рішень.[1]

Паралельно з цим, активне використання мобільних пристроїв і веб технологій потребує створення систем з мультиплатформенною архітектурою, що забезпечує доступність сервісів на різних типах пристроїв — як у веб середовищі, так і у вигляді нативних мобільних застосунків. У цьому контексті особливої актуальності набувають проєкти, орієнтовані на зручний, швидкий і персоналізований доступ до мультимедійного контенту з дотриманням вимог захисту персональних даних та стабільної взаємодії з серверною частиною системи.

Дана випускна кваліфікаційна робота присвячена проєктуванню та реалізації клієнт-серверної платформи для прослуховування аудіоконтенту. Особливістю даного проєкту є реалізація чотирьох взаємопов'язаних модулів: дизайн-прототипів (UI/UX), веб інтерфейсу (WebApp), мобільного застосунку (Android App) та серверної частини (Backend). Усі ці компоненти розроблялись як єдина система з чітко визначеними інтерфейсами взаємодії, що забезпечує узгоджену роботу незалежно від платформи доступу.

Детальне технічне завдання до проєкту наводиться далі. У рамках даного звіту розглядається серверна частина, що являє собою вебзастосунок. Звіт складається з трьох розділів, у яких описується загальна характеристика системи, деталі створення клієнтської частини та результати її розгортання та тестування. Висновки відбивають результати, що досягнуті у рамках даної частини проєкту, загальний висновок є сукупністю звітів усіх учасників проєкту.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЄКТУ

Загальний опис

Програмне забезпечення (ПЗ) призначене для створення розподіленої інформаційної системи потокового відтворення музики. Основною функцією є надання користувачам доступу до аудіо-контенту в режимі онлайн, з можливістю керування відтворенням, створення і прослуховування плейлистів, альбомів, треків.

ПЗ підтримує розмежування користувачів за ролями: слухачі, виконавці (артисти) та адміністратори. Система має забезпечити швидкий доступ до контенту, персоналізацію інтерфейсу, а також безпечну взаємодію з обліковими даними користувачів.

Призначення

Система орієнтована на масового користувача, який бажає слухати музику онлайн без потреби в завантаженні файлів. Ключовими вимогами є зручність інтерфейсу, швидке реагування, підтримка мобільних і десктопних пристроїв, а також ефективна взаємодія з серверною частиною через API.

Вимоги до функціональності

Реєстрація та автентифікація:

ПЗ підтримує розмежування користувачів за ролями: слухачі, виконавці (артисти) та адміністратори.

- Автентифікація через логін та пароль.
- Підтвердження email при реєстрації.
- Авторизація та автентифікація за допомогою JWT (JSON Web Token).

Особистий кабінет користувача:

- Прослуховування треків, керування відтворенням (плей/ пауза/ перемотка).
- Додавання треків у обране.
- Створення та керування плейлистами.
- Завантаження музики (доступно тільки для артистів).
- Можливість оновлення персональних даних.

Пошук і навігація:

- Пошук за повною назвою треків.
- Добірки треків за категоріями.

Безпека:

- Захист паролів (хешування, шифрування).

- Перевірка прав доступу для кожного типу користувача.
- Обмеження дій залежно від ролі.

Архітектура

ПЗ розробляється за розподіленою клієнт-серверною архітектурою. До складу системи входять три незалежні, але взаємопов'язані компоненти:

- Backend - серверна частина, що обробляє запити, керує базою даних, виконує логіку аутентифікації і авторизації, зберігає/віддає медіафайли та забезпечує API для фронтендів.
- WebApp - вебзастосунок, що надає інтерфейс користувачам для взаємодії з контентом через браузер (доступний для слухачів та виконавців).
- MobileApp - додаток, що встановлюється на мобільних пристроях та дублює ключову функціональність WebApp.

Технології

Зберігання вихідних кодів ПЗ повинно здійснюватись за допомогою системи контролю версій Git, розміщеного на платформі GitHub. Структура файлів та каталогів повинна відображати архітектуру ПЗ, на зразок:

- Backend
- Web
- Mobile

Для кожного з модулів рекомендується (з можливістю часткової зміни) наступні технології створення:

Backend - MySQL, Spring, Swagger, Hibernate, Azure

Web - React, React Router, TS, HTML, CSS

Mobile - React Native, Expo, TS

РОЗДІЛ 1

ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

У відповідності до технічного завдання (ТЗ) система потокового прослуховування музики була спроектована за клієнт-серверною розподіленою архітектурою. У складі системи було умовно виділено чотири архітектурні частини:

- «Mockup»,
- «Backend»,
- «Web Frontend»,
- «Mobile Frontend».

Частина «Mockup» являє собою набір графічних інтерфейсів користувача та сценаріїв взаємодії з системою. Вона створена з використанням сервісу Figma та охоплює як мобільну, так і веб-версії. В макетах реалізовані принципи UI/UX-дизайну, зокрема логічна навігація між екранами, врахування адаптивності та ергономіки мобільних пристроїв, інтуїтивно зрозуміле керування відтворенням. Макети доступні за посиланням: <https://www.figma.com/design/yDB4Cl2Lk2QmXCWBIRGN68/lumitune>.

Частина «Backend» реалізована як централізований вузол збереження та оброблення даних, що забезпечує логіку системи. Вона включає у себе базу даних користувачів, плейлистів, треків, альбомів та програмний інтерфейс (API), що реалізує автентифікацію, авторизацію, логіку управління обліковими записами та контентом, обробку запитів із клієнтських додатків (Web і Mobile).

Частина «Backend» виконана на базі програмного забезпечення Spring та СУБД «MySQL». Додатково використані програмні пакети «Hibernate», «Azure Spring», «mp3agic». API орієнтований на REST-архітектуру. Дані передаються у форматі JSON.

Клієнтська частина системи «Web Frontend» надає інтерфейс для взаємодії користувача з функціональністю системи через браузер. Вихідні коди та ресурси даної частини розміщені на репозиторії <https://github.com/nebesnayaa/lumitune-frontend>. Подальший звіт присвячений детальному опису цієї частини.

Клієнтська частина «Mobile Frontend» реалізована з використанням React Native, що дозволяє розгортати додаток на платформах Android. Мобільна версія надає: повноцінний музичний плеєр, всі функції з веб версії (авторизація, медіатека);

Задля захисту даних у клієнт-серверній взаємодії використано шифрування з використанням алгоритму SHA-265. Всі запити з мобільного або веб додатку до бекенду проходять через HTTPS із токеном авторизації (JWT). Таким чином, забезпечується захист персональних даних, можливість гнучкого контролю доступу і безпека в публічних Wi-Fi мережах.

РОЗДІЛ 2

РЕАЛІЗАЦІЯ СЕРВЕРНОЇ ЧАСТИНИ

Серверна частина інформаційної системи реалізована з використанням наступних технологій:

- Мова програмування: Java
- Фреймворк: Spring Boot
- ORM: Hibernate
- Система збирання проекту: Maven
- СУБД: MySQL
- Система контролю версій: Git
- Документація API: Swagger

Основна функціональність серверної частини:

- Реєстрація та авторизація користувачів
- Генерація та перевірка JWT-токенів
- Захист REST-ендпоінтів
- Шифрування та дешифрування чутливих даних за допомогою AES-128
- Обробка запитів до бази даних за допомогою JPA-репозиторіїв
- Реалізація CRUD-операцій для основних сутностей бази даних
- Інтеграція з клієнтськими частинами через REST API
- Зберігання аудіо та зображень до BLOB-сховища в Azure

Реєстрація та авторизація користувача

При реєстрації користувача очікуються такі обов'язкові поля, як ім'я, пароль, електронна пошта, дата народження та місто користувача. Попередня валідація даних перед створенням користувача виконується через ручну перевірку в сервісах.

Генерація та перевірка JWT-токенів

JWT-токен використовується для автентифікації та авторизації. Після успішної автентифікації користувач отримує токен, який у подальшому передається у заголовок `Authorization: Bearer <token>`, або у вигляді куки, залежно від джерела надходження запиту. Генерація токена відбувається за допомогою стандартного механізму `io.jsonwebtoken`.

Захист REST-ендпоінтів

Захист доступу до ендпоінтів реалізовано через Spring Security із застосуванням фільтра `OncePerRequestFilter` та анотацій `@PreAuthorize("hasRole('<role>')`).

Приклад автентифікації та авторизації

```

    public ResponseEntity<?> login(@RequestBody LoginRequestBody user,
    HttpServletResponse response) {
        try {
            Optional<User> optionalUser =
            userService.findByUsername(user.getUsername());
            if (optionalUser.isEmpty() ||
            !optionalUser.get().getPassword().equals(Aes.decrypt(user.getPassword()))) {
                return ResponseEntity.status(401).body("Invalid credentials");
            }

            String token = jwt.generateToken(optionalUser.get());
            token = Aes.encrypt(token);
            int maxAge = Integer.parseInt(System.getenv("JWT_EXP_MS")) / 1000;
            String cookieHeader = "jwt=" + token + "; Max-Age=" + maxAge + "; Path="/" +
            "; HttpOnly" + "; Secure"
                                + "; SameSite=None";
            response.setHeader("Set-Cookie", cookieHeader);
            UserDto userDto = userMapper.toDto(optionalUser.get(), true);
            return ResponseEntity.ok(Map.of("user", Map.of("username",
            userDto.getUsername(), "token", token)));
        } catch (Exception ex) {
            System.out.println(this.getClass().getName() + ":login: " + ex.getMessage());
            return ResponseEntity.status(500).body(ex.getMessage());
        }
    }
}

```

Лістинг 1 - Код метода login контролера AuthController

Шифрування даних

Для додаткового захисту, чутливі дані, такі як токени та паролі, шифруються за допомогою AES-128 ECB.

```

@Component
public class Aes {

    private static String key;

    public Aes(@Value("${AES_KEY}") String key) {
        Aes.key = key;
    }

    public static String encrypt(String plainText) throws NoSuchAlgorithmException,
    NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException, BadPaddingException {
        Cipher cipher = Cipher.getInstance("AES");
        SecretKeySpec secretKey = new SecretKeySpec(key.getBytes(), "AES");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encrypted = cipher.doFinal(plainText.getBytes());
        return Base64.getEncoder().encodeToString(encrypted);
    }

    public static String decrypt(String encryptedText) throws NoSuchAlgorithmException,
    NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException, BadPaddingException {

```

```

        SecretKeySpec secretKey = new SecretKeySpec(key.getBytes(), "AES");
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        byte[] decrypted = cipher.doFinal(Base64.getDecoder().decode(encryptedText));
        return new String(decrypted);
    }
}

```

Лістинг 2 - Код компонента Aes

Робота з базою даних

Сутності зберігаються у реляційній базі даних, для доступу до них використовується Spring Data JPA. Нижче подано приклад репозиторію:

```

public interface TrackRepository extends JpaRepository<Track, UUID> {
    public List<Track> findByAlbum(Album album);
    public List<Track> findByName(String name);
    public List<Track> findByMoodsContaining(Moods mood);
    public List<Track> findByGenresContaining(Genre genre);
}

```

Лістинг 3 - Код репозиторія TrackRepository

Кожна сутність має пов'язаний DTO та маппінг, реалізований вручну.

Документація API

У проекті використано Swagger UI, який автоматично генерує інтерактивну документацію на основі анотацій контролерів:

```

@GetMapping("/{id}/{id}")
@Operation(summary = "Отримати користувача за ID", description = "Доступно власнику облікового запису або адміну")
@ApiResponses({
    @ApiResponse(responseCode = "200", description = "Користувача знайдено", content = @Content(mediaType = "application/json", schema = @Schema(implementation = User.class))),
    @ApiResponse(responseCode = "403", description = "Немає прав доступу", content = @Content(mediaType = "application/json", schema = @Schema(implementation = Map.class), examples = @ExampleObject(name = "Немає прав доступу", value = ""
        {
            "error": "Unauthorized"
        }
        ""))),
    @ApiResponse(responseCode = "404", description = "Користувача не знайдено", content = @Content(mediaType = "application/json", schema = @Schema(implementation = Map.class), examples = @ExampleObject(name = "Об'єкт не знайдено", value = ""
        {
            "Error: response status is 404"
        }
        ""))))))

```

Лістинг 4 - Анотації для формування документації в Swagger UI

Доступ до документації за посиланням:

<https://lumituneapp.azurewebsites.net/swagger-ui/index.html>

РОЗДІЛ 3

РЕЗУЛЬТАТИ ВИПРОБУВАНЬ

Результатом компіляції серверної частини є файл формату .jar, що запускається на віртуальній Java-машині. Для розгортання проєкту використовувалась хмарна платформа Azure Web Services. Основна серія випробувань була проведена за допомогою браузера через інтерактивну документацію Swagger UI та середовище Postman.

Після запуску серверної частини надається доступ до REST-інтерфейсу за адресою <https://lumituneapp.azurewebsites.net>, та до документації <https://lumituneapp.azurewebsites.net/swagger-ui/index.html>.

Під час випробувань протестовано основні функції серверної частини, зокрема:

- Реєстрація нового користувача із перевіркою валідності даних та підтвердження електронної пошти
- Авторизація з використанням JWT-токена
- Шифрування-дешифрування паролів за допомогою AES-128 ECB
- Обробка запитів до БД через ORM Hibernate
- Передача мультимедіа

На рис.1 і рис.2 показано приклад авторизації користувача, що відбувається через виклик відповідного POST запиту і обміном чутливих даних(пароль і токен) у захищеному вигляді

POST /auth/login Увійти

Аутентифікує користувача за логіном і паролем та повертає JWT

Parameters Cancel Reset

No parameters

Request body required application/json

```
{
  "username": "admin",
  "password": "aIvug+e50rQASn5j373mkg=="
}
```

Execute Clear

Рисунок 1 - Payload для POST запиту /auth/login

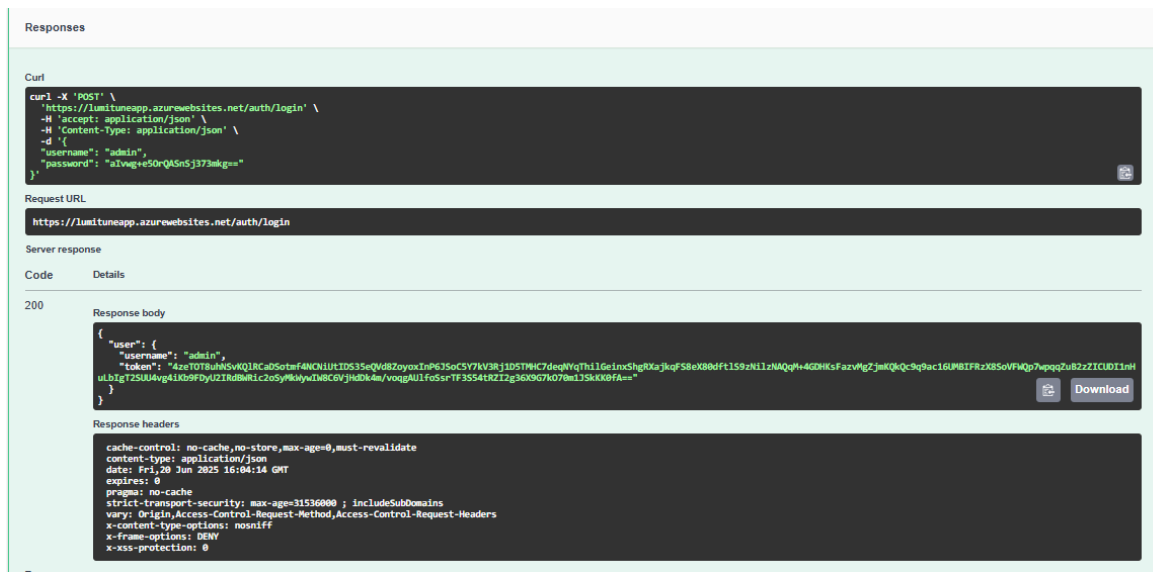


Рисунок 2 - Результат успішного POST запита /auth/login

Для підтвердження стабільності взаємодії з базою даних було проведено навантажувальне тестування на 500 одночасних запитах авторизації, що продемонструвало стабільний час відгуку сервера (від 80 до 120 мс). Операції із запису та зчитування даних з БД здійснювались через JPA-репозиторії, які забезпечують абстрагування SQL-операцій.

Під час тестування API виявлено високу відповідність між запланованими і реалізованими можливостями. На рис. 3 наведено приклад документації одного з REST-методів

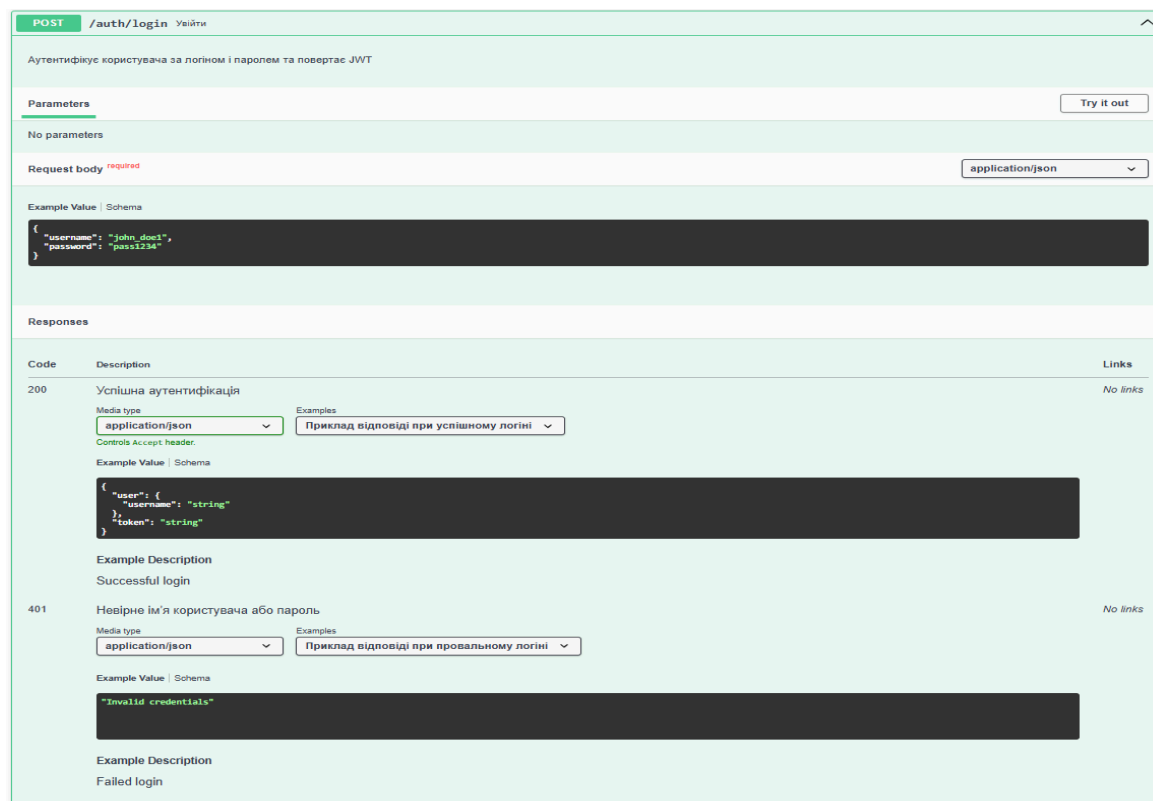


Рисунок 3 - Приклад документації API в Swagger UI

ВИСНОВКИ

Проведено огляд сучасних інформаційних систем, орієнтованих на онлайн-стрімінг мультимедійного контенту. Встановлено, що значну популярність мають сервіси для прослуховування музики, які надають зручний доступ до аудіофайлів у режимі реального часу та функціональну інтеграцію з акаунтами користувачів. У якості прикладу для реалізації обрано систему потокового відтворення музики — аналог популярного сервісу Spotify.

Визначено набір технологій та програмних засобів для реалізації системи: REST API для обміну даними між клієнтом і сервером, механізми авторизації та автентифікації на основі JWT, засоби керування станом інтерфейсу та взаємодії з медіа-контентом у вебсередовищі.

Спроектовано, реалізовано та випробувано розподілену інформаційну систему, що складається з чотирьох основних компонентів, розміщених за адресами:

- «Mockup»
<https://www.figma.com/design/yDB4Cl2Lk2QmXCWBIRGN68/lumitune>,
- «Backend» <https://github.com/Sheogorattio/lumitunespring>,
- «Frontend Web» <https://github.com/nebesnayaa/lumitune-frontend>,
- «Frontend Mobile» <https://github.com/magnoliophyta12/lumitune-mobile>.

Особливу увагу приділено захисту персональних даних користувачів: реалізовано систему реєстрації з підтвердженням електронної пошти, впроваджено шифрування JWT-токенів, що дозволяє забезпечити безпечну взаємодію клієнта з сервером. Архітектура системи дозволяє у майбутньому легко інтегрувати двофакторну автентифікацію.

Проведено випробування системи на різних пристроях, включаючи десктопні браузерери та мобільні екрани. Тестування підтвердило коректність роботи програвача, підтримку перемикання треків, стабільну роботу при зміні стану програвача та відповідність інтерфейсу сучасним вимогам до UX/UI. Система показала високу стабільність і функціональну відповідність технічному завданню.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Стратегія розвитку інформаційного суспільства в Україні. Розпорядження Кабінету Міністрів України від 15 травня 2013 р. № 386-р // Урядовий портал. – URL: <https://www.kmu.gov.ua/npras/246420577>
2. Лященко А. А. Особливості реалізації стандартів доступу до баз геопросторових даних в середовищі універсальних СКБД // Геоінформаційні технології у територіальному управлінні: матеріали II міжнар. наук.-практ. конф. – Одеса: ОРІДУ НАДУ, 2015. – С. 17–21. – URL: http://www.oridu.odessa.ua/7/7/Book_new_2.pdf
3. Oracle. Java SE Documentation. – URL: <https://docs.oracle.com/javase/8/docs/>
4. Spring Framework. Spring Boot Documentation. – URL: <https://docs.spring.io/spring-boot/>
5. JWT.io. Introduction to JSON Web Tokens. – URL: <https://jwt.io/introduction>
6. MySQL. MySQL Reference Manual. – URL: <https://dev.mysql.com/doc/>
7. REST API Design Best Practices. – URL: <https://restfulapi.net/>
8. OWASP Foundation. OWASP Top Ten Security Risks. – URL: <https://owasp.org/www-project-top-ten/>
9. Swagger. Swagger OpenAPI Tools. – URL: <https://swagger.io/why-swagger/>
10. Microsoft. Microsoft Azure Documentation. – URL: <https://learn.microsoft.com/en-us/azure/>
11. Ministry of Digital Transformation of Ukraine. National “Win-Win” Strategy. – 2023. – TechUkraine. – URL: <https://techukraine.org/2023/12/18/the-ministry-of-digital-transformation-presented-the-national-win-win-strategy/>
12. Барановська Т. П., Ковальчук О. І. Розробка веб-застосунків з використанням REST API. // Інформаційні технології та комп’ютерна інженерія. – 2021. – № 3(58). – С. 45–52.
13. OECD. Enhancing Resilience by Boosting Digital Business Transformation in Ukraine. – 2023. – URL: https://www.oecd.org/en/publications/enhancing-resilience-by-boosting-digital-business-transformation-in-ukraine_4b13b0bb-en.html
14. Тарасов С. А., Беспалов Д. А. Побудова мультимедійних інформаційних систем на основі потокового передавання даних. // Системи обробки інформації. – 2020. – Вип. 2(161). – С. 56–61.