Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут» імені Ігоря Сікорського
Факультет інформатики та обчислювальної техніки
Кафедра Інформаційних Систем та Технологій

Лабораторна робота № 3

з дисципліни: «Технології розроблення програмного забезпечення»

Виконав:
Тимчук Владислав
ІА-34
Перевірив:
Мягкий М. Ю.

Київ-2025

Тема: Основи проектування.

Мета: Навчитися проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.

Тема Лабораторного Практикуму:

Музичний програвач (iterator, command, memento, facade, visitor, clientserver)
Музичний програвач становить собою програму для програвання музичних файлів або відтворення потокової музики з можливістю створення, запам’ятовування і редагування списків програвання, перемішування/повторення (shuffle/repeat), розпізнавання різних аудіоформатів, еквалайзер.

Вступ

Ця лабораторна робота охоплює проєктування фізичної та логічної структури системи через діаграми компонентів і розгортання, а також детальну ілюстрацію ключових бізнес-процесів за допомогою діаграм послідовностей. Завершимо представленням ключових фрагментів програмного коду, які реалізують повний клієнт-серверний цикл обробки даних.

Зміст

Хід роботи

1. ДІАГРАМА КОМПОНЕНТІВ

Діаграма компонентів ілюструє логічну архітектуру системи, показуючи, з яких програмних компонентів вона складається та як вони взаємодіють між собою.

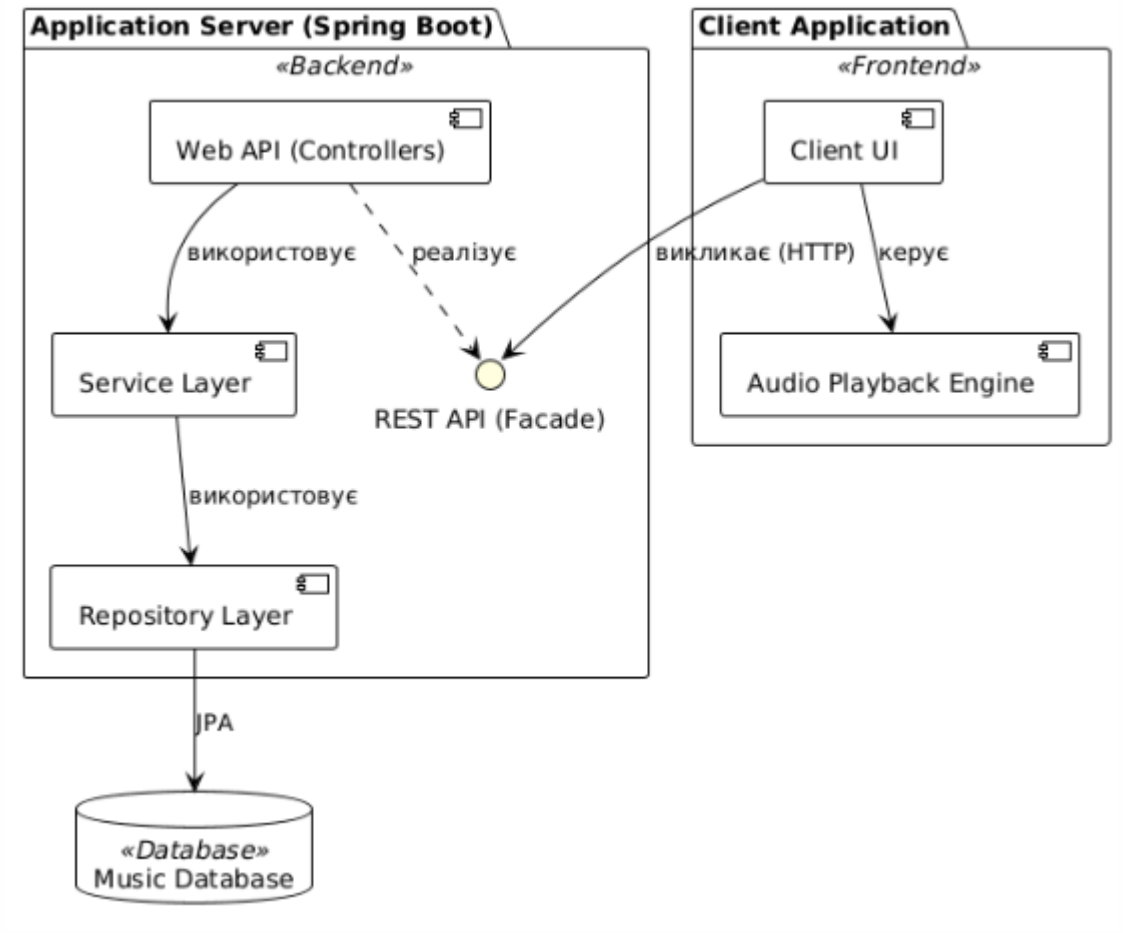

Рис.1 – Діаграма компонентів

Діаграма показує клієнт-серверну архітектуру системи, що складається з трьох основних блоків:

1. **Client Application**: Це клієнт, який містить дві основні частини:
    1. **Client UI**: Графічний інтерфейс користувача з яким взаємодіє користувач.
    2. **Audio Playback Engine**: Компонент, шо відповідає за декодування та відтворення аудіофайлів на пристрої
2. **Application Server**: Це бекенд-частина системи, реалізована на Spring Boot. Вона має чітку багатошарову архітектуру:
    1. **Web API (Controllers)**: Шар, що реалізує шаблон Facade. **Web API Facade** — це шаблон архітектури який використовується для

створення єдиного, спрощеного інтерфейсу до складної системи з кількох внутрішніх API чи мікросервісів. Він надає клієнту простий та зрозумілий REST API для взаємодії з системою, приховуючи складну внутрішню логіку.

2. **Service Layer**: Шар бізнес-логіки, де відбуваються основні операції
3. **Repository Layer**: Шар доступу до даних, що реалізує шаблон Repository для взаємодії з базою даних.

3. **Music Database**: Система управління базами даних де фізично зберігаються всі дані про треки, плейлісти, альбоми та виконавців.

2. ДІАГРАМА РОЗГОРТАННЯ

Діаграма розгортання показує фізичне розміщення компонентів системи на апаратному забезпеченні.

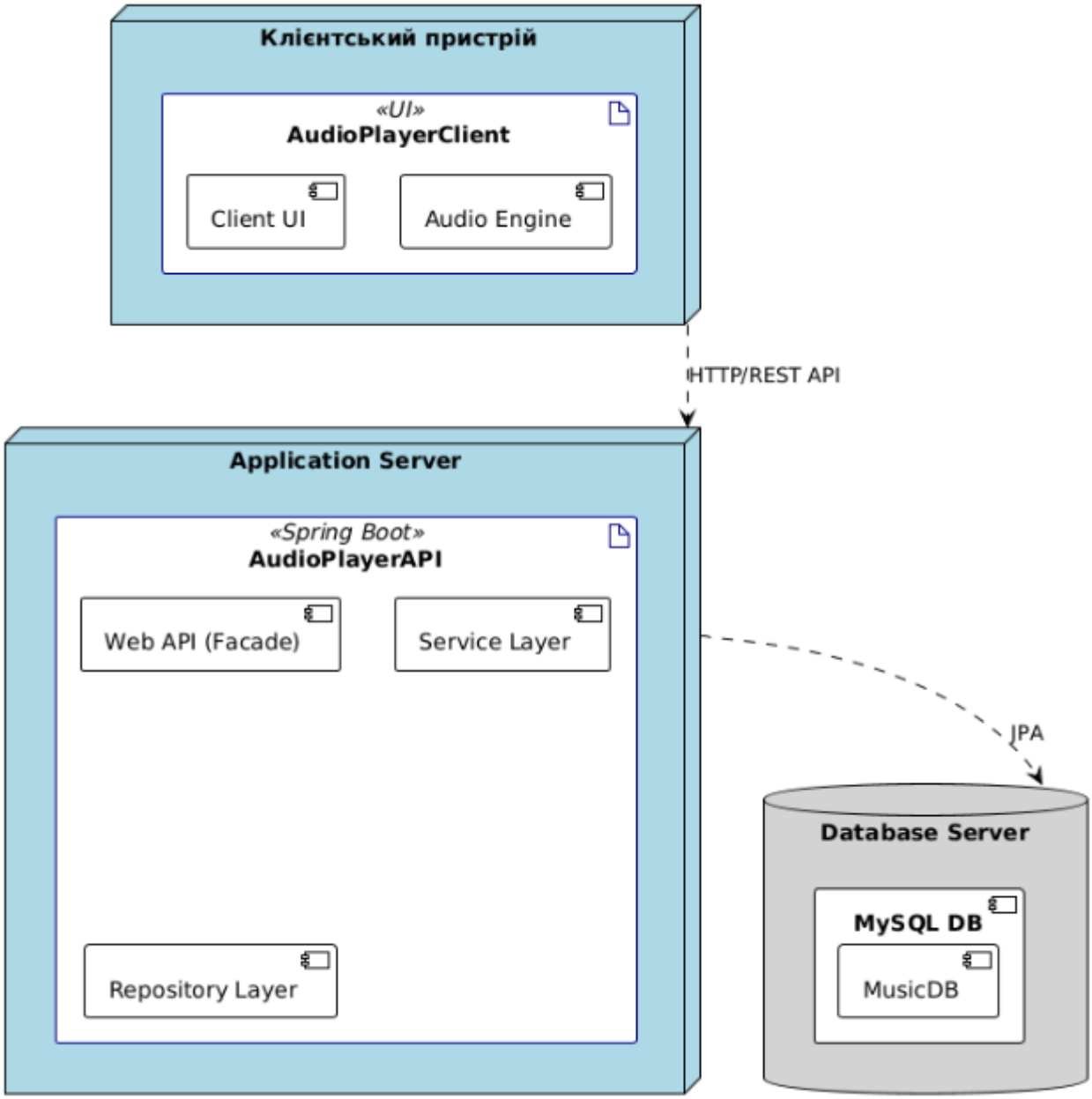

Рис. 2 – Діаграма розгортання

Діаграма відображає фізичну архітектуру системи, яка складається з трьох вузлів:

1. **Клієнтський пристрій (ПК)**: Це фізична машина користувача на якій запускається артефакт AudioPlayerClient, тобто наш додаток
2. **Application Server**: Це сервер на якому розгорнуто AudioPlayerAPI — наш Spring Boot додаток. Він обробляє всі бізнес-запити
3. **Database Server**: Окремий сервер, виділений для роботи СУБД де зберігається база даних MusicDB.

Комунікація між клієнтом та сервером додатку відбувається по мережевому протоколу HTTP. Сервер додатку спілкується з сервером бази даних через JPA.

3. ДІАГРАМА ПОСЛІДОВНОСТЕЙ

Діаграми послідовностей показують взаємодію об'єктів у часі для конкретних сценаріїв.

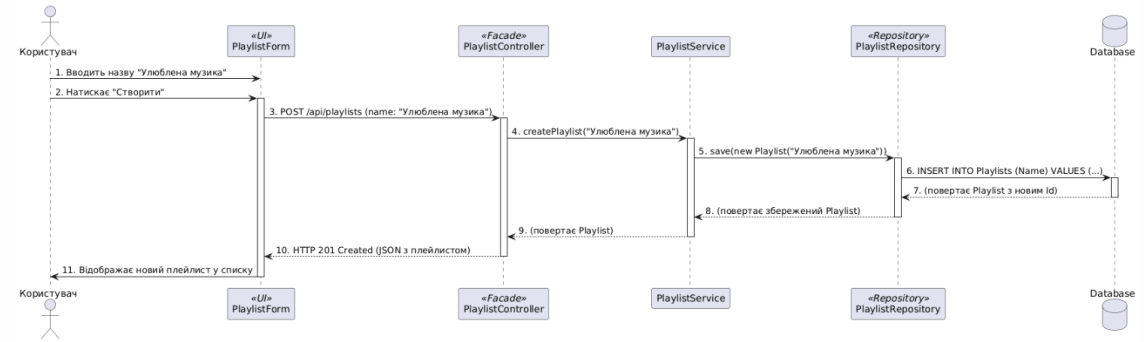

Рис. 3 – Діаграма послідовностей сценарію №1 (Створення нового плейліста)

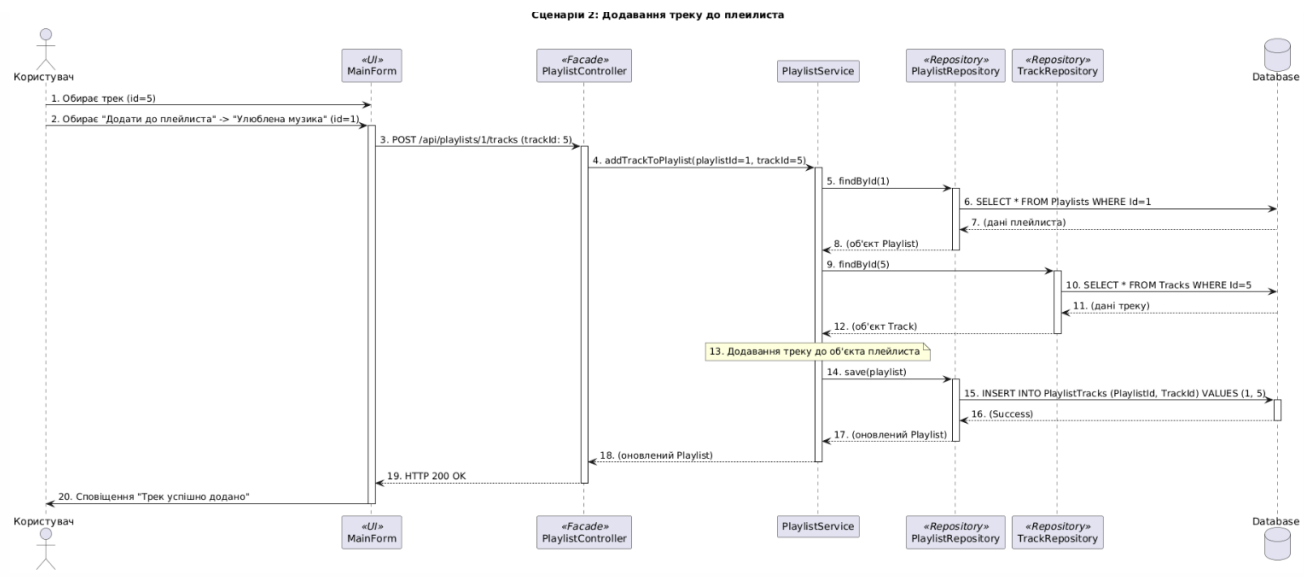

Рис. 4 - Діаграма послідовностей сценарію №2 (Додавання треку до плейліста)

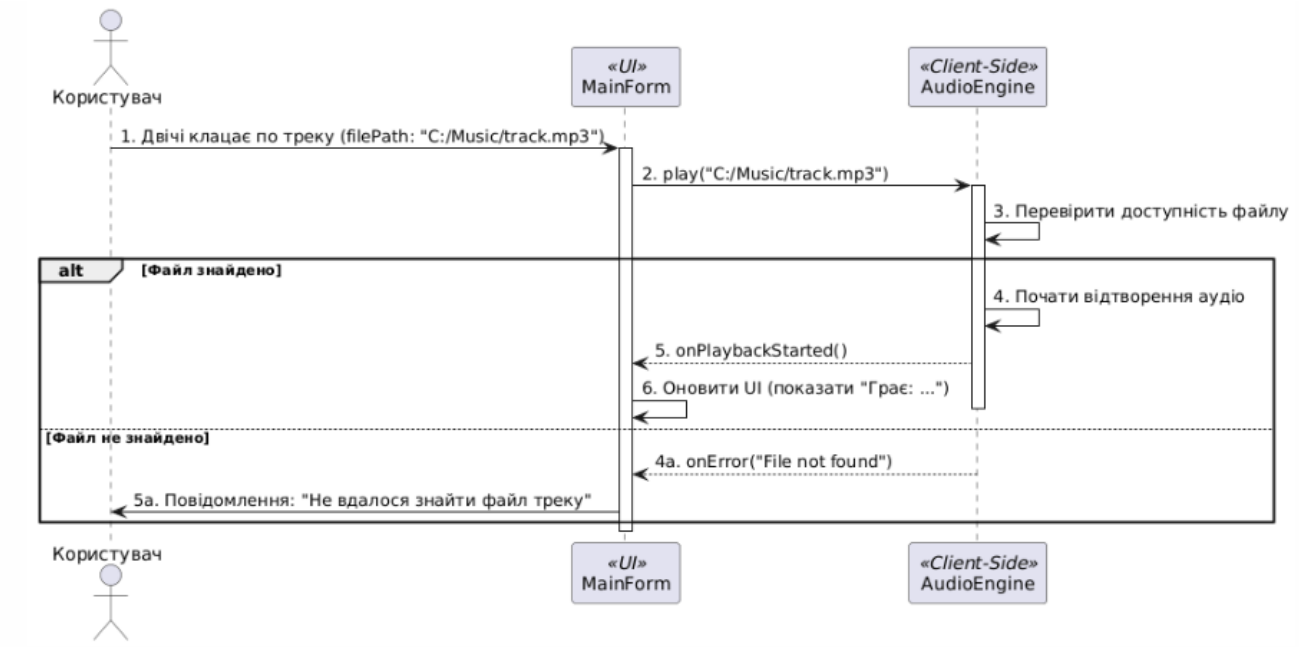

Рис. 5 - Діаграма послідовностей сценарію №3 (Відтворення треку)

4. ВИХІДНИЙ КОД СИСТЕМИ

4.1. Бекенд

Усі класи надані для перегляду у GitHub репозиторії. Посилання: https://github.com/fromz67/TRPZ/tree/main/lab3

- **Сутності;**
- **Репозиторії;**
- **Шари бізнес-логіки;**
- **Контролери (Facade);**

4.2. Клієнт

- **Сервіс для роботи з HTTP HttpClientService.java;**
- **Контролер UI MainFormController.java;**

ВИСНОВОК

У цій лабораторній роботі було завершено проєктування архітектури шляхом візуалізації його логічних та фізичних компонентів. Розроблені програмні компоненти (контролери, сервіси, клієнтський HTTP-сервіс) заклали основу для реалізації додатку, здатного обробляти запити користувача від візуальної форми до бази даних і назад.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. **Що собою становить діаграма розгортання?** Діаграма розгортання — це структурна діаграма UML, яка візуалізує фізичну архітектуру системи. Вона показує, як програмні артефакти розподілені по фізичних або віртуальних вузлах (серверах, пристроях) і як ці вузли з'єднані між собою.
2. **Які бувають види вузлів на діаграмі розгортання?** На діаграмі розгортання бувають два основні види вузлів: вузол пристрою (Device Node), який представляє собою фізичне залізо, та вузол середовища виконання, який представляє програмне середовище всередині пристрою
3. **Які бувають зв’язки на діаграмі розгортання?** На діаграмі розгортання використовуються два основні типи зв'язків: асоціація (Association), що зображується суцільною лінією і показує комунікаційний шлях між вузлами, та залежність (Dependency), що зображується пунктирною стрілкою і показує, що один артефакт розгортається на певному вузлі.
4. **Які елементи присутні на діаграмі компонентів?** Основними елементами діаграми компонентів є самі компоненти (логічні блоки системи, наприклад, [UI], [API**]),** інтерфейси, та зв'язки між ними (залежності, реалізації), що показують, як компоненти взаємодіють для виконання завдань системи.
5. **Що становлять собою зв’язки на діаграмі компонентів?** Зв'язки на діаграмі компонентів показують залежності між ними. Найчастіше це залежність (Dependency), зображена пунктирною стрілкою --->, яка показує, що один компонент використовує сервіси іншого, або зв'язок через інтерфейси, де один компонент реалізує інтерфейс, а інший — використовує його.
6. **Які бувають види діаграм взаємодії?** Діаграми взаємодії — це категорія UML-діаграм, що моделюють динамічну поведінку системи. До них належать: діаграма послідовностей (Sequence Diagram), діаграма комунікації (Communication Diagram), діаграма огляду взаємодії (Interaction Overview Diagram) та діаграма синхронізації (Timing Diagram).
7. **Для чого призначена діаграма послідовностей?** Діаграма послідовностей призначена для візуалізації взаємодії об'єктів у часі. Вона чітко показує порядок відправлення повідомлень між різними учасниками (об'єктами, класами) для виконання конкретного сценарію або варіанту використання.

8. **Які ключові елементи можуть бути на діаграмі послідовностей?** Ключовими елементами є учасники, зображені як прямокутники з пунктирною лінією вниз, повідомлення, зображені стрілками між лініями життя, та смуги активації, які показують, коли об'єкт є активним і виконує операцію.
9. **Як діаграми послідовностей пов’язані з діаграмами варіантів використання?** Діаграми послідовностей деталізують діаграми варіантів використання. Кожен варіант використання може бути розкладений на один або декілька сценаріїв і кожна діаграма послідовностей візуалізує один такий конкретний сценарій, показуючи, як об'єкти взаємодіють для його реалізації.
10. **Як діаграми послідовностей пов’язані з діаграмами класів?** Діаграми послідовностей динамічно демонструють, як об'єкти класів, визначених на діаграмі класів, взаємодіють між собою. Якщо діаграма класів — це статичне креслення, то діаграма послідовностей — це анімація, що показує, як екземпляри цих класів викликають методи один одного для виконання роботи.