Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського” Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №3
Технологія розробки програмного забезпечення
**«Основи проектування розгортання.»**

Виконала студентка
групи ІА–32:
Ткачук М. С.

Перевірив:
Мягкий М. Ю.

Київ 2025

**Зміст**

**Тема:** Основи проектування розгортання

**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

**Завдання:**

Ознайомитись з короткими теоретичними відомостями.

- Проаналізувати діаграми створені в попередній лабораторній роботі а також тему системи та спроєктувати діаграму розгортання використання відповідно до обраної теми лабораторного циклу.
- Розробити діаграму компонентів для проєктованої системи.
- Розробити діаграму розгортання для проєктованої системи.
- Розробити як мінімум дві діаграми послідовностей для сценаріїв прописаних в попередній лабораторній роботі.
- На основі спроєктованих діаграм розгортання та компонентів доопрацювати програмну частину системи. Реалізація системи, додатково до попередньої реалізації, повинна містити як мінімум дві візуальні форми. В системі вже повинен бути повністю реалізована архітектура (повний цикл роботи з даними від вводу на формі до збереження їх в БД і подальшій виборці з БД та відображенням на UI).
- Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму розгортання з описом, діаграму компонентів системи з описом, діаграми послідовностей, а також вихідний код системи, який було додано в цій лабораторній роботі

**Аудіо редактор** (singleton, adapter, observer, mediator, composite, client-server)
Аудіо редактор повинен володіти наступним функціоналом: представлення аудіо даних будь-якого формату в WAVE-формі, вибір і подальші операції

копіювання / вставки / вирізання / деформації по сегменту аудіозапису, можливість роботи з декількома звуковими доріжками, кодування в найбільш поширених форматах (ogg, flac, mp3).

## Теоретичні відомості

Діаграма розгортання (Deployment Diagram) – важливий інструмент у моделюванні архітектури програмних систем. Вона показує, як програмні компоненти (якщо вони є частинами системи) розгортаються на фізичних пристроях, таких як сервери, комп’ютери або інші мережеві вузли. Діаграма розгортання відображає фізичну структуру системи, вказуючи на те, як різні частини програмного забезпечення взаємодіють між собою через апаратні компоненти. Вона допомагає зрозуміти конфігурацію мережі, з’єднання між вузлами та розподіл ресурсів на різних пристроях.

Діаграма компонентів (Component Diagram) – показує структуру системи з точки зору її компонентів та їх взаємодії. Вона використовується для візуалізації фізичних компонентів програмної системи, таких як модулі, бібліотеки, сервіси або інші програмні одиниці, та їх зв’язків. Діаграма компонентів корисна для визначення того, як різні частини системи будуть взаємодіяти між собою на етапі реалізації та для аналізу масштабованості і модульності системи.

Діаграма послідовностей ((Sequence Diagram) – ілюструє, як об’єкти або компоненти взаємодіють один з одним протягом певного сценарію або процесу. Вона зображує обмін повідомленнями між об’єктами в часі, де час йде вертикально, а повідомлення між об’єктами — горизонтально. Діаграма послідовностей дозволяє візуалізувати порядок виконання дій і процесів, що допомагає у розумінні логіки роботи системи на етапі розробки та тестування.

## Хід Роботи

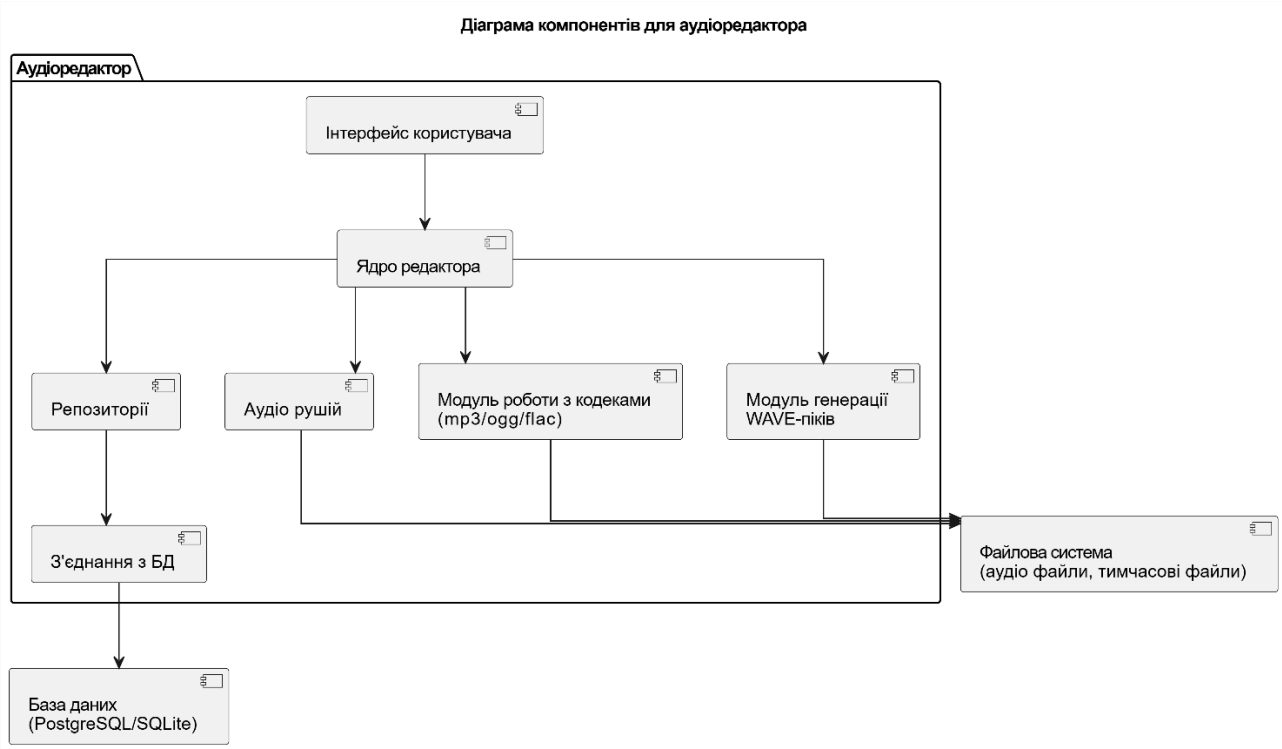


Рис 1. Діаграма компонентів

## Діаграма компонентів

### 1. Основні компоненти

- **Інтерфейс користувача**

  Надає взаємодію з користувачем. Ініціює операції імпорту, редагування, перегляду WAVE-форми, експорту. Відображає стан проєкту, треки, кліпи та прогрес експорту.

- **Ядро редактора**

  Центральний компонент координації. Приймає команди з інтерфейсу користувача, керує моделями домену проєкту, делегує роботу іншим модулям і репозиторіям. Відповідає за транзакційну логіку редагування та узгодженість даних.

- **Аудіо рушій**

Виконує відтворення та міксдаун у PCM. Забезпечує попередній прослух і підготовку суміші перед експортом. Працює з файловою системою для читання буферів і тимчасових даних.

- **Модуль роботи з кодеками (mp3/ogg/flac)**
  Інкапсулює кодування та декодування популярних форматів. Надає уніфікований інтерфейс для імпорту аудіо та експорту результату в обраний формат.
- **Модуль генерації WAVE-піків**
  Обчислює піки/даунсемпл для побудови WAVE-представлення на таймлайні. Читає аудіо з файлової системи та повертає компактні дані для швидкого рендеру.
- **Репозиторії**
  Реалізують доступ до даних за шаблоном Repository. Оперують сутностями Project, Track, Clip, AudioAsset, Selection, ExportSettings, ExportJob. Інкапсулюють SQL і роботу з підключенням.
- **З'єднання з БД**
  Відповідає за встановлення та закриття підключення до бази даних. Використовується репозиторіями. Дає можливість легко змінити драйвер або пул з'єднань.
- **База даних (PostgreSQL/SQLite)**
  Сховище структурованих даних проєкту. Забезпечує цілісність через зовнішні ключі, індекси та обмеження.
- **Файлова система (аудіо файли, тимчасові файли)**
  Зберігає вихідні аудіоресурси, проміжні PCM-буфери, результати експорту й тимчасові файли, що створюються під час обробки.

**2. Взаємодії між компонентами**

- **Інтерфейс користувача → Ядро редактора**

Передає команди користувача. Отримує зворотній стан для відображення на таймлайні та в діалогах.

- **Ядро редактора ↔ Репозиторії**
  Зчитує та зберігає проєкти, треки, кліпи, налаштування експорту і статуси задач. Працює з репозиторіями як з абстракцією над БД.
- **Репозиторії ↔ З'єднання з БД ↔ База даних**
  Репозиторії виконують запити через компонент з'єднання. У БД зберігаються всі структуровані дані аудіоредактора.
- **Ядро редактора → Модуль роботи з кодеками**
  Під час імпорту запитує декодування файлів для створення кліпів. Під час експорту передає параметри та отримує потік закодованих даних.
- **Ядро редактора → Аудіо рушій**
  Формує завдання на відтворення та міксдаун. Рушій повертає статус відтворення і прогрес міксдауну.
- **Ядро редактора → Модуль генерації WAVE-піків**
  Ініціює побудову піків для відображення форми сигналу. Отримані дані передаються назад в інтерфейс користувача.
- **Аудіо рушій ↔ Файлова система**
  Читає й пише PCM та тимчасові файли під час прослуху і міксдауну.
- **Модуль роботи з кодеками ↔ Файлова система**
  Читає вхідні файли під час імпорту. Створює вихідні файли у форматах mp3, ogg, flac під час експорту.
- **Модуль генерації WAVE-піків ↔ Файлова система**
  Зчитує аудіо для аналізу і зберігає проміжні дані піків за необхідності.

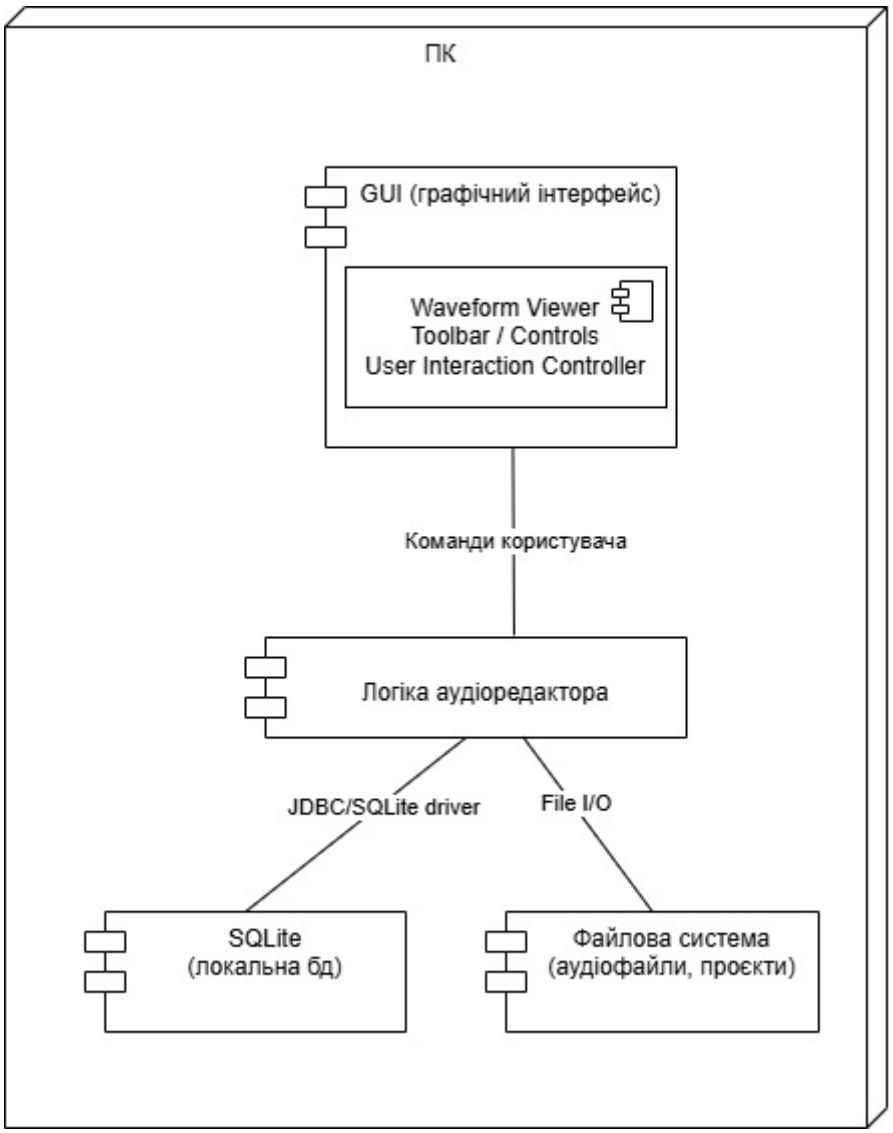


Рис 2. Діаграма розгортання

1. Фізичні вузли (ноди)

Персональний комп'ютер (ПК)
Основний вузол, на якому запускається аудіоредактор. Містить такі компоненти:

- GUI (графічний інтерфейс) - забезпечує взаємодію з користувачем. До його складу входять Waveform Viewer, Toolbar / Controls та User Interaction Controller, які відображають хвильову форму та обробляють події користувача.
- Логіка аудіоредактора - центральний модуль, що отримує команди від GUI та виконує редагування, обробку, збереження і завантаження аудіо.

- JDBC/SQLite driver - компонент для роботи з локальною базою даних, який виконує SQL-запити та зберігає метадані про проєкти і кліпи.
- File I/O - механізм читання та запису файлів у файлову систему.

SQLite (локальна база даних)
Окремий вузол, у якому зберігаються структуровані дані про проєкти, налаштування, треки, кліпи та інші метадані.

Файлова система (аудіофайли, проєкти)
Містить аудіофайли, тимчасові файли, дані проєктів, результати експорту та інші ресурси, необхідні для роботи редактора.

2. Актор

Користувач
Взаємодіє з системою через GUI. Виконує редагування аудіо, переглядає хвильову форму, запускає імпорт та експорт, працює з файлами та проєктами, використовуючи мишу та клавіатуру.

3. Зв'язки між компонентами

- Команди користувача передаються з GUI до логіки аудіоредактора.
- JDBC/SQLite driver забезпечує зв'язок логіки редактора з локальною БД.
- File I/O використовується для читання і запису аудіофайлів та проєктів до файлової системи.
- GUI отримує оновлені дані (наприклад, оновлену хвильову форму) від логіки редактора та відображає їх користувачу.

4. Взаємодія компонентів

- Користувач здійснює дію у GUI (наприклад, виділення аудіосегмента або натискання кнопки).
- GUI фіксує подію і надсилає відповідну команду до логіки аудіоредактора.
- Логіка аудіоредактора виконує операцію, яка може включати редагування аудіо, оновлення хвильової форми, збереження в БД або роботу з файлами.
- Якщо операція пов'язана з файлами, використовується File I/O.
- Якщо змінюється стан проєкту, логіка звертається до SQLite через JDBC/SQLite driver.
- GUI оновлює відображення хвильової форми та інтерфейс відповідно до змін у логіці

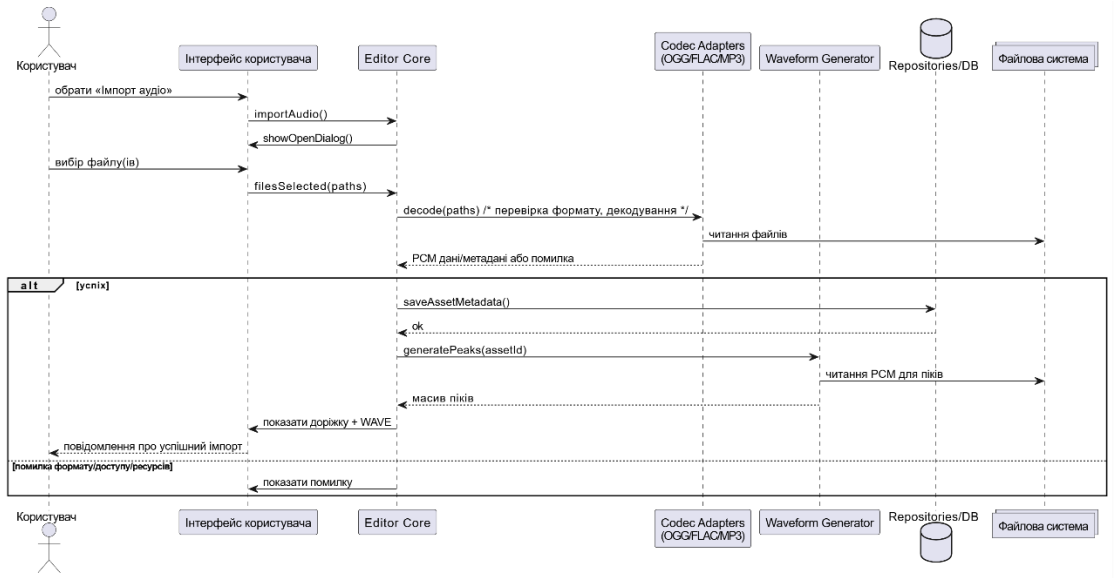


Рис 3.1 Діаграма послідовностей - Імпорт аудіо

## 1) Імпорт аудіо

1. Користувач у інтерфейсі обирає команду «Імпорт аудіо».
2. Інтерфейс відкриває системний діалог вибору файлів і повертає шляхи до вибраних аудіо.
3. Editor Core передає список шляхів у Codec Adapters для перевірки формату і декодування.
4. Codec Adapters читають файли з файлової системи та повертають у Core PCM-дані і метадані (або помилку).
5. За успіху Core зберігає метадані нового ресурсу в Repositories/БД.
6. Core ініціює у Waveform Generator побудову WAVE-піків для щойно імпортованого ресурсу.
7. Waveform Generator читає PCM з файлової системи, обчислює масив піків і повертає його в Core.
8. Core оновлює інтерфейс: додає доріжку/кліп у проєкт, показує хвильову форму.

9. Користувач бачить імпортований матеріал на таймлайні і може продовжувати роботу.

**Винятки та обробка помилок:**

- Непідтримуваний формат або пошкоджений файл – інтерфейс показує повідомлення про помилку та пропонує вибрати інший файл.
- Немає доступу до файлу – повідомлення з порадою перевірити права або шлях.
- Нестача пам’яті під час генерації піків – Core скасовує операцію, показує повідомлення, ресурс у БД не фіксується.

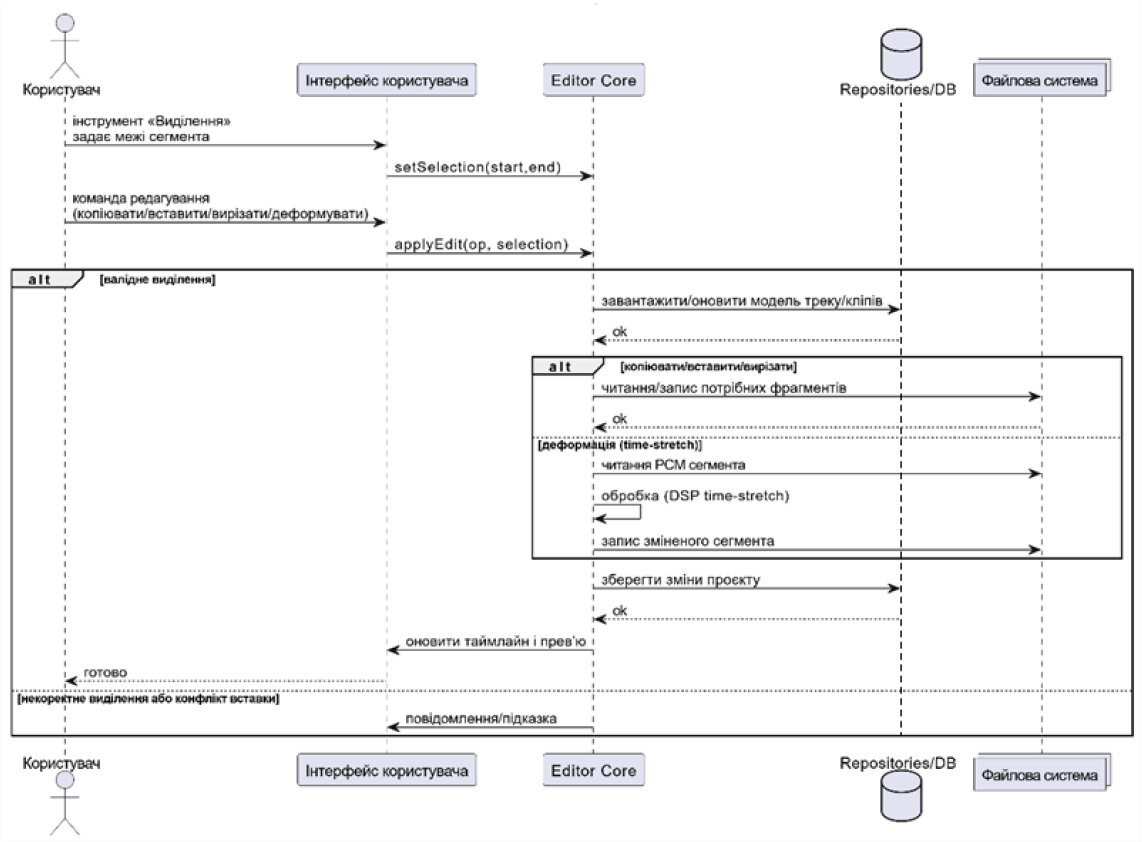


Рис 3.2 Діаграма послідовностей – Редагування сегмента

**2) Редагування сегмента**

1. Користувач виділяє сегмент на таймлайні інструментом «Виділення» (startMs, endMs).
2. Інтерфейс передає Editor Core межі активного виділення.
3. Користувач обирає дію редагування: копіювати або вставити або вирізати або деформувати (time-stretch).
4. Інтерфейс передає в Core команду applyEdit(op, selection).
5. Core завантажує з БД потрібні моделі треку/кліпів і готує план змін.
6. Якщо операція копіювання/вставки/вирізання:
    - Core виконує читання/запис відповідних фрагментів у файловій системі (оновлення меж кліпів, створення або видалення файлів фрагментів, якщо використовується офлайн-рендер).

7. Якщо операція деформації (time-stretch):
    - Core читає PCM сегмента з файлової системи, застосовує DSPалгоритм зміни тривалості/тональності. ○ Core записує змінений сегмент назад у файлову систему і оновлює посилання в моделі.
8. Core фіксує зміни проєкту в БД (оновлені кліпи, позиції, довжини, посилання).
9. Core оновлює інтерфейс: перемальовує таймлайн і прев'ю хвильової форми.
10. Користувач бачить застосований результат.

**Винятки та обробка помилок:**

- Некоректне виділення (негативна довжина, вихід за межі кліпу) – інтерфейс показує підказку і не виконує операцію.
- Конфлікт вставки (накладання на заборонену ділянку, блокування іншою операцією) – повідомлення і пропозиція змінити позицію.
- Помилки вводу-виводу – повідомлення про неможливість зчитати/записати файл із порадою перевірити доступ або диск.

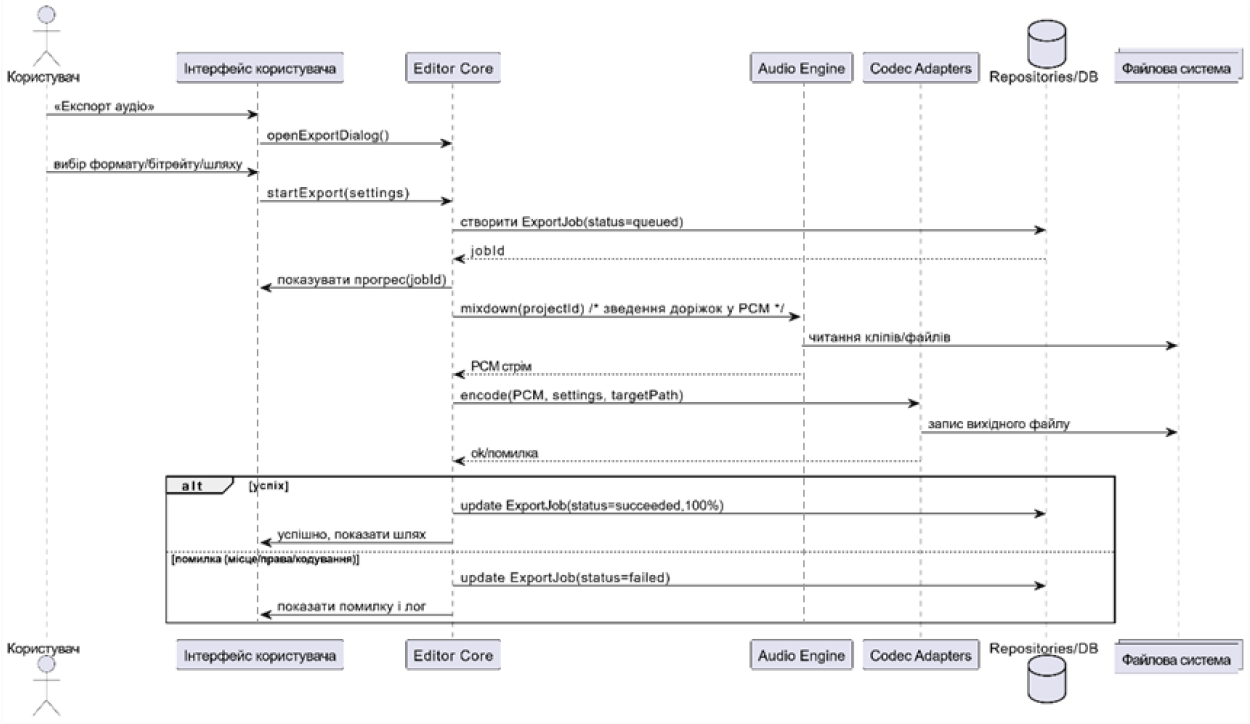


Рис 3.3 Діаграма послідовностей - Експорт аудіо

**3): Експорт аудіо**

1. Користувач у інтерфейсі обирає «Експорт аудіо».
2. Інтерфейс відкриває діалог експорту – користувач задає формат, бітрейт/якість, цільовий шлях.
3. Інтерфейс передає налаштування в Editor Core командою startExport(settings).
4. Core створює запис ExportJob у БД зі статусом queued і показує у UI прогрес-панель.
5. Core викликає Audio Engine для міксдауну – зведення всіх доріжок у єдиний PCM-потік.
6. Audio Engine зчитує кліпи/файли з файлової системи, застосовує гейн/панораму/ефекти, повертає PCM-дані в Core.
7. Core передає PCM у Codec Adapters для кодування у вибраний формат (OGG/FLAC/MP3) та цільовий шлях у файловій системі.

8. Codec Adapters записують вихідний файл, повідомляють Core про результат.
9. За успіху Core оновлює ExportJob у БД на succeeded, прогрес 100%, інтерфейс показує шлях до файлу.
10. Користувач завершує роботу з діалогом експорту або відкриває теку з результатом.

**Винятки та обробка помилок:**

- Нестача місця або відсутні права на запис – ExportJob помічається як failed, інтерфейс показує причину і пропонує обрати іншу теку.
- Помилка кодування (невалідні параметри/кодек) – статус failed, повідомлення з деталями.
- Перервано користувачем – статус failed або canceled, файл видаляється, інтерфейс повідомляє про скасування.

**Вихідний код:**

**https://github.com/mandarinchik21/AudioEditor/tree/main/audio-editor**

**Висновок**: У цій лабораторній роботі було спроєктовано діаграму компонентів, діаграму розгортання та діаграми послідовностей, після чого доопрацьовано програмну частину системи: реалізовано дві візуальні форми, забезпечено повний цикл роботи з даними

## Контрольні запитання

1. Що собою становить діаграма розгортання?

це UML-діаграма, яка показує, як програмні компоненти розміщені на фізичних або віртуальних вузлах (напр. сервери, ПК, мобільні пристрої) та як вони між собою взаємодіють

2. Які бувають види вузлів на діаграмі розгортання?

Є апаратні (напр. сервер, клієнтський комп'ютер, телефон) та програмні (операційні системи, сервери додатків, бази даних)

3. Які бувають зв'язки на діаграмі розгортання?

На діаграмі розгортання бувають асоціативні та комунікаційні зв'язки, що показують обмін даними між вузлами

4. Які елементи присутні на діаграмі компонентів?

На діаграмі компонентів є самі компоненти, інтерфейси, порти, пакети та залежності між ними

5. Що становлять собою зв'язки на діаграмі компонентів?

Зв'язки на діаграмі компонентів відображають залежність одного компонента від іншого

6. Які бувають види діаграм взаємодії?

Діаграма послідовностей, діаграма комунікацій, діаграма таймінгу та діаграма огляду взаємодії

7. Для чого призначена діаграма послідовностей?

Діаграма послідовностей потрібна, щоб показати порядок викликів методів і обмін повідомленнями між об'єктами у часі

8. Які ключові елементи можуть бути на діаграмі послідовностей?

Об'єкти(тробіж учасники), їх життєві лінії, повідомлення (синхронні чи асинхронні виклики), активації та блоки управління

9. Як діаграми послідовностей пов'язані з діаграмами варіантів використання?

Кожен сценарій з діаграми use case можна деталізувати у вигляді діаграми послідовностей, щоб показати кроки виконання

10. Як діаграми послідовностей пов’язані з діаграмами класів

Класи, що описані у діаграмі класів, стають об’єктами у діаграмі послідовностей, і на ній видно, як вони взаємодіють між собою