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 №2

із дисципліни «*Технології розроблення програмного забезпечення*»

Діаграма варіантів використання. Сценарії варіантів використання. Діаграми uml. Діаграми класів. Концептуальна модель системи. Аудіоредактор

Виконала
студентка групи ІА–34
Кузьменко В.С.

Перевірив
викладач
Мягкий М.Ю.

Київ 2025

ЗМІСТ

Тема: Основи проектування.

Мета: Обрати зручну систему побудови UML-діаграм та навчитися будувати діаграми варіантів використання для системи що проєктується, розробляти сценарії варіантів використання та будувати діаграми класів предметної області.

Завдання

• Ознайомитись з короткими теоретичними відомостями.

• Проаналізувати тему та спроєктувати діаграму варіантів використання відповідно до обраної теми лабораторного циклу.

• Спроєктувати діаграму класів предметної області.

• Вибрати 3 варіанти використання та написати за ними сценарії використання.

• На основі спроєктованої діаграми класів предметної області розробити основні класи та структуру бази даних системи. Класи даних повинні реалізувати шаблон Repository для взаємодії з базою даних.

• Нарисувати діаграму класів для реалізованої частини системи.

• Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму варіантів використання відповідно, діаграму класів системи, вихідні коди класів системи, а також зображення структури бази даних.

Тема роботи:

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

UML та його діаграми

UML (Unified Modeling Language) – уніфікована мова візуального моделювання, що використовується для аналізу, проєктування та документування програмних систем. Вона дозволяє описувати систему на різних рівнях: від концептуального до фізичного.

Основні діаграми UML

- Діаграма варіантів використання (Use Case Diagram) – показує вимоги до системи та взаємодію користувачів із нею.
- Діаграма класів (Class Diagram) – описує статичну структуру системи: класи, їх атрибути, методи та зв’язки.

Діаграма варіантів використання

Діаграма *use case* відображає функціональність системи з точки зору користувача.

Основні елементи:

- Актори (Actor) – користувачі або зовнішні системи.
- Варіанти використання (Use Case) – дії або послуги, які система надає актору (наприклад: вхід, перегляд даних, створення транзакції).

Типи відносин:

- Асоціація – прямий зв’язок актора з варіантом використання.
- Include – один сценарій завжди включає інший (обов’язковий).
- Extend – сценарій може бути розширений додатковим (необов’язковим).
- Узагальнення – спадкування ролей або функціоналу.

Для уточнення роботи системи складають сценарії використання (use case scenarios), які описують:

- передумови та постумови;
- учасників;
- короткий опис;
- основний перебіг подій;
- винятки.

Діаграма класів

Діаграма класів показує структуру системи: класи, їх атрибути, методи та зв’язки між ними.

Клас містить:

- назву;
- атрибути (дані);
- методи (операції).

Види зв’язків:

- Асоціація – загальний зв’язок між класами.
- Узагальнення (успадкування) – зв’язок між батьківським і дочірнім класом.
- Агрегація – відношення «ціле–частина», де частини можуть існувати окремо.
- Композиція – сильне відношення «ціле–частина», де частини не існують без цілого.

Логічна структура бази даних

Проєктування бази даних часто виконується на основі діаграми класів.

Різновиди моделей:

- Фізична модель – організація файлів і способів зберігання.
- Логічна модель – таблиці, атрибути, ключі, зв’язки.

Щоб уникнути надмірності даних, застосовують нормалізацію:

- 1НФ – кожен атрибут має лише одне атомарне значення.
- 2НФ – усі неключові атрибути залежать від усього первинного ключа.
- 3НФ – немає транзитивних залежностей (атрибутів, що залежать від інших неключових атрибутів).
- НФ Бойса–Кодда (BCNF) – посилена форма 3НФ, у якій кожна залежність визначається ключем.

Хід роботи

1. Ознайомитися з короткими теоретичними відомостями.
2. Проаналізуйте тему та намалюйте схему прецеденту, що відповідає обраній темі лабораторії.
3. Намалюйте діаграму класів для реалізованої частини системи.
4. Виберіть 3 прецеденти і напишіть на їх основі прецеденти.
5. Розробити основні класи і структуру системи баз даних.
6. Класи даних повинні реалізувати шаблон Репозиторію для взаємодії з базою даних.

Діаграма прецедентів

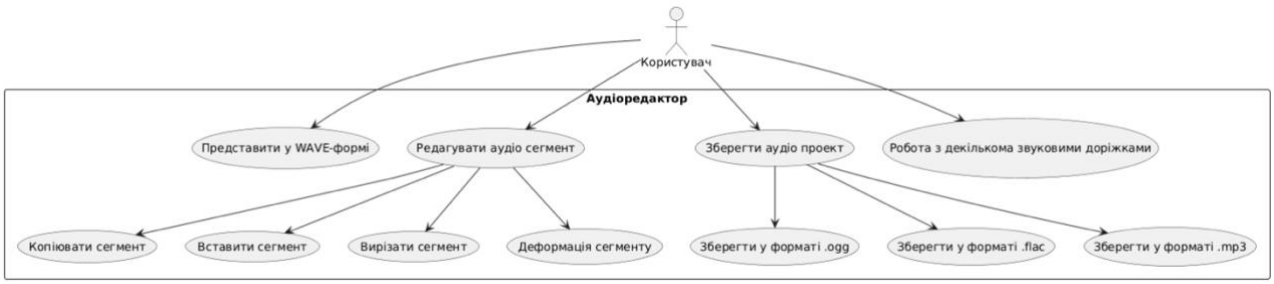

Рис. 1 Діаграма прецедентів

Основна функціональність:

- Відображення аудіосигналу у вигляді хвильової (WAVE-form).
- Операції з сегментами аудіозапису (копіювання, вставка, вирізання, деформація).
- Кодування та збереження в поширених форматах (.ogg, .flac, .mp3).

- Робота з декількома звуковими доріжками.

Головні прецеденти:

Система надає чотири основні сценарії використання: Представити у WAVE-формі**,** Додати аудіо сегмент на звукову доріжку**,** Зберегти аудіо проект та Редагувати аудіо сегмент**.**

Сценарій **Зберегти аудіо проект** розширюють три спеціалізовані сценарії:

- Зберегти аудіо проєкт у форматі .ogg
- Зберегти аудіо проект у форматі .flac
- Зберегти аудіо проєкт у форматі .mp3

Сценарій **Редагувати аудіо сегмент** включає в себе чотири окремих сценарії:

- Копіювати аудіо сегмент
- Вставити аудіо сегмент
- Вирізати аудіо сегмент
- Деформувати аудіо сегмент

Опис ключових 3 прецеденти

1. Представити у WAVE-формі

Опис: Користувач отримує графічне представлення аудіофайлу у вигляді хвилі (waveform) для візуального аналізу та подальших операцій редагування.

Передумови:

- Аудіофайл завантажено в редактор.
- Формат файлу підтримується системою (наприклад, .wav, .mp3, .flac).

Актори:

- **Основний:** Користувач.

- **Вторинний:** Аудіообробник (системний модуль, що відповідає за аналіз аудіоданих).

Основний сценарій:

1. Користувач завантажує аудіофайл через інтерфейс програми.
2. Система перевіряє формат файлу на підтримку.
3. Система декодує аудіодані та аналізує амплітуду сигналу для побудови графіка.
4. Графічний модуль системи відображає WAVE-форму на екрані.
5. Користувач отримує можливість масштабувати, прокручувати та виділяти ділянки графіка для редагування.

Результат:

- WAVE-форма аудіофайлу відображена в інтерфейсі.
- Користувач може продовжити роботу та виконувати операції редагування.

Виключні ситуації:

- Файл має непідтримуваний формат (система виводить повідомлення про помилку).
- Виникла помилка під час завантаження файлу (система пропонує спробувати ще раз).

2. Редагувати аудіо сегмент

Опис: Користувач виконує одну з базових операцій редагування над виділеною ділянкою аудіозапису: копіювання, вставка, вирізання або деформація.

Передумови:

- Аудіофайл завантажено та його WAVE-форма відображена.
- Користувач виділив конкретний сегмент аудіо для маніпуляцій.

Актори:

- **Основний:** Користувач.
- **Вторинний:** Модуль редагування аудіо (внутрішній компонент системи).

Основний сценарій:

1. Користувач виділяє сегмент аудіо на WAVE-формі.
2. Користувач обирає потрібну дію:
    - **Копіювання:** Система поміщає копію сегмента в буфер обміну.
    - **Вставка:** Система вставляє вміст буфера обміну в зазначене місце на доріжці.
    - **Вирізання:** Система видаляє виділений сегмент із доріжки та поміщає його в буфер обміну.
    - **Деформація:** Система надає інтерфейс для налаштування параметрів (швидкість, тональність, реверс).
3. Система застосовує обрану операцію до аудіоданих та оновлює графічне представлення WAVE-форми.

Результат:

- Виділений сегмент аудіо змінено відповідно до обраної операції.
- WAVE-форма оновлена для відображення змін.

Виключні ситуації:

- Межі виділеного сегменту виходять за межі наявного аудіо.
- Введено некоректні параметри для деформації (система повідомляє про помилку).

3. Зберегти аудіо проект

Опис: Користувач зберігає результат роботи у вигляді готового аудіофайлу у обраному форматі, а також (опційно) зберігає проект із метаданими для подальшого редагування.

Передумови:

- В системі є активний аудіопроект, готовий до збереження.
- Користувач вибрав цільовий формат та розташування для файлу.

Актори:

- **Основний:** Користувач.
- **Вторинний:** Модуль кодування/експорту файлів.

Основний сценарій:

1. Користувач ініціює команду "Зберегти проект".
2. Система надає вікно вибору, де користувач вказує:
    - Формат фінального файлу (.ogg, .flac, .mp3).
    - Папку призначення.
3. Система кодує аудіодані у вибраний формат.
4. Система створює файл проекту (наприклад, у власному форматі), який зберігає всі метадані, доріжки та історію редагування.
5. Збережений проект стає доступним для повторного відкриття та редагування.

Результат:

- Аудіопроект успішно збережено як готовий аудіофайл у обраному форматі.
- Створено файл проекту із метаданими для майбутніх змін.

Виключні ситуації:

- Відсутній доступ до вказаної папки для збереження (система запитує вибрати інше місце).
- Обраний формат експорту не підтримується.
- Недостатньо місця на диску для завершення операції.

Діаграма класів

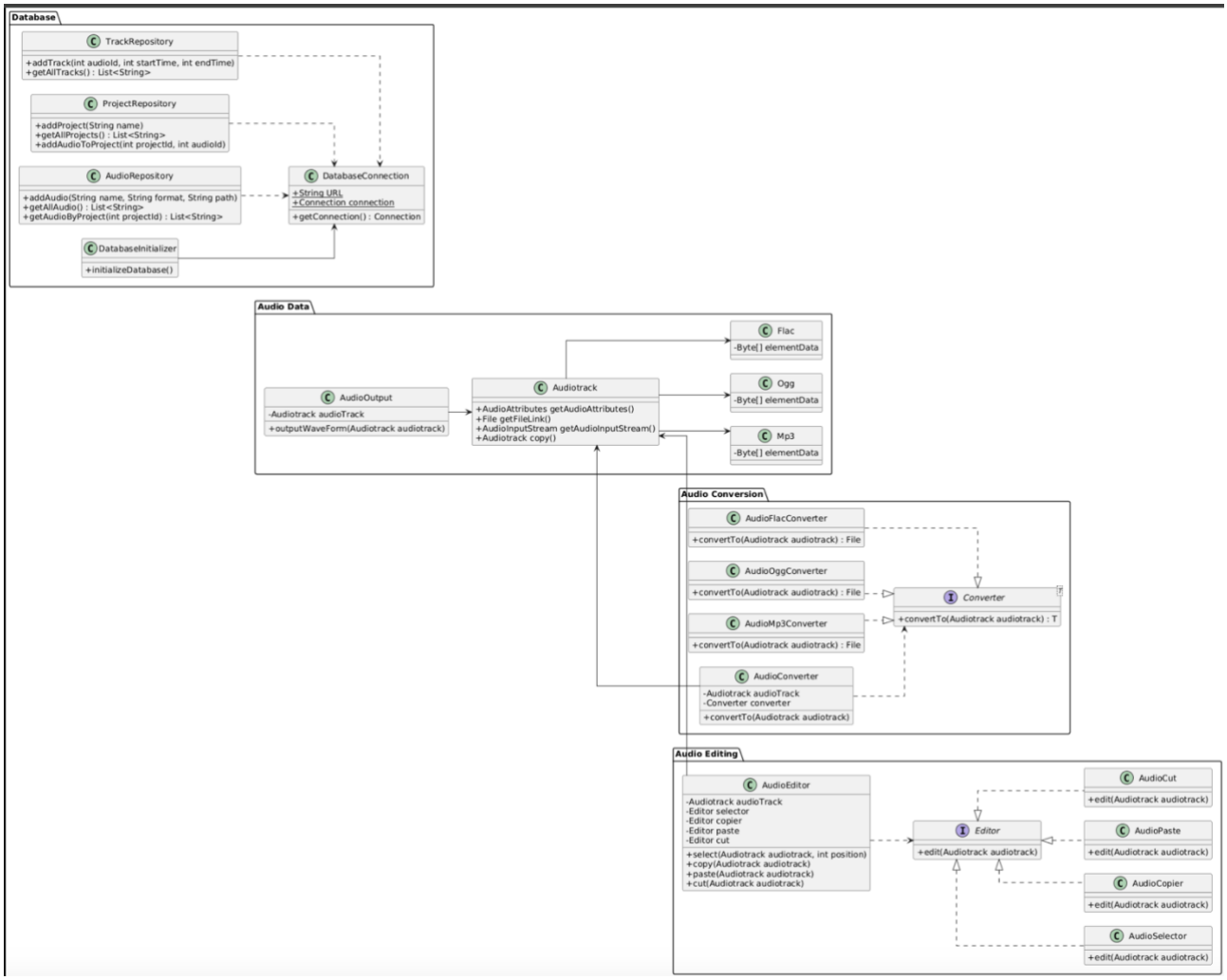

Рис. 2 Діаграма класів

1. Робота з базою даних

Цей блок є фундаментом усієї системи, оскільки саме він відповідає за зберігання та отримання даних. Без правильної роботи бази неможливо було би редагувати аудіо, конвертувати його, відслідковувати стан проєктів.

1.1. DatabaseInitializer

Призначення: клас відповідає за початкове налаштування бази даних. Його головна функція - створити необхідні таблиці та структури у випадку першого запуску системи або оновити їх, якщо структура змінилася.

Метод:

- initializeDatabase() - створює чи оновлює схему бази. Наприклад, таблиці для:
    - аудіофайлів (назва, формат, шлях до файлу),
    - проєктів (назва, дата створення),
    - треків (часові межі, прив’язка до аудіо).

Це дозволяє уникнути ручного створення таблиць і робить програму незалежною від конкретного стану бази.

1.2. DatabaseConnection

Призначення: надає глобальне з’єднання з базою.

Поля:

- static final String URL - константа з адресою, де розташована база (наприклад, jdbc:sqlite:audioeditor.db).
- static Connection connection - активне підключення, яке використовують усі репозиторії.

Метод:

- getConnection() - повертає з’єднання. Це дозволяє уникати повторного створення підключень, що економить ресурси.

2. Репозиторії

Репозиторії формують рівень доступу до даних (DAO). Вони інкапсулюють SQL-запити й дозволяють працювати з даними через об’єктні методи, не думаючи про деталі зберігання.

2.1. AudioRepository

Призначення: управління аудіофайлами у базі.

Методи:

- addAudio(String name, String format, String path) - створює новий запис про аудіофайл.
- getAllAudio(): List<String> - повертає список усіх аудіофайлів для відображення.
- getAudioByProject(int projectId): List<String> - повертає лише ті файли, які належать до певного проєкту.

Таким чином, клас дозволяє швидко отримати потрібні аудіо для подальшої роботи.

2.2. ProjectRepository

Призначення: управління проєктами.

Методи:

- addProject(String name) - створює новий проєкт.
- getAllProjects(): List<String> - отримує список усіх проєктів.
- addAudioToProject(int projectId, int audioId) - створює зв’язок між проєктом і аудіо.

Цей клас потрібен для того, щоб згрупувати роботу з кількома аудіофайлами в одному проєкті.

2.3. TrackRepository

Призначення: управління треками (фрагментами аудіо).

Методи:

- addTrack(int audioId, int startTime, int endTime) - додає новий трек із заданими часовими межами.
- getAllTracks(): List<String> - повертає всі наявні треки.

Цей репозиторій дає змогу відслідковувати й редагувати окремі фрагменти аудіо.

3. Робота з аудіотреками

Ця група класів описує сутність аудіо та базові операції з ним.

3.1. Audiotrack

Призначення: є базовим представленням аудіофайлу. Він не лише зберігає посилання на фізичний файл, але й надає методи для роботи з ним.

Методи:

- getAudioAttributes() - повертає атрибути аудіо (бітрейт, частоту, канали).
- getFileLink() - надає доступ до фізичного файлу.
- getAudioInputStream() - відкриває потік для відтворення або аналізу.
- copy() - створює копію треку (корисно для редактора).

3.2. AudioOutput

Призначення: відповідає за виведення та візуалізацію аудіо.

Поле:

- Audiotrack audioTrack - аудіотрек, який буде відображатися.

Метод:

- outputWaveForm(Audiotrack) - будує графічне зображення хвильової форми сигналу. Це дозволяє користувачу бачити аудіо візуально.

4. Редагування аудіо

Цей блок описує операції над аудіо - виділення, копіювання, вставка, вирізання.

4.1. AudioEditor

Призначення: головний клас для редагування.

Методи:

- select(Audiotrack, int position) - виділяє фрагмент.
- copy(Audiotrack) - копіює його.

- paste(Audiotrack) - вставляє скопійоване.
- cut(Audiotrack) - вирізає частину.

4.2. Editor (інтерфейс)

Задає контракт редагування:

- edit(Audiotrack) - метод, який мають реалізувати всі редактори.

4.3. Реалізації Editor

- AudioSelector - виділення частини треку.
- AudioCopier - копіювання.
- AudioPaste - вставка.
- AudioCut - вирізання.

Таким чином, будь-яку операцію редагування можна розширити або змінити незалежно від інших.

5. Конвертація аудіо

Блок відповідає за зміну формату аудіо.

5.1. AudioConverter

Поля:

- Audiotrack audioTrack - трек для конвертації.
- Converter converter - об’єкт, що виконує конкретну конвертацію.

Метод:

- convertTo(Audiotrack) - виконує перетворення.

5.2. Converter (інтерфейс)

Визначає загальний метод:

- convertTo(Audiotrack) - реалізація залежить від формату.

5.3. Реалізації Converter

- AudioMp3Converter - конвертація в MP3.
- AudioOggConverter - конвертація в OGG.
- AudioFlacConverter - конвертація в FLAC.

Такий підхід дозволяє легко додати новий формат, просто створивши ще одну реалізацію.

6. Формати аудіо

Класи Mp3, Ogg, Flac описують конкретні формати.

Поле:

- Byte[] elementData - містить дані файлу у вигляді байтового масиву.

Загальна структура системи

1. **Робота з даними:** репозиторії забезпечують зберігання і доступ до бази.
2. **Робота з аудіо:** аудіотрек можна відтворювати, копіювати, редагувати.
3. **Конвертація:** система підтримує різні формати і легко розширюється.

Завдяки такій архітектурі програма є модульною, масштабованою та зрозумілою для підтримки**.**

Основні класи та структура репозиторію

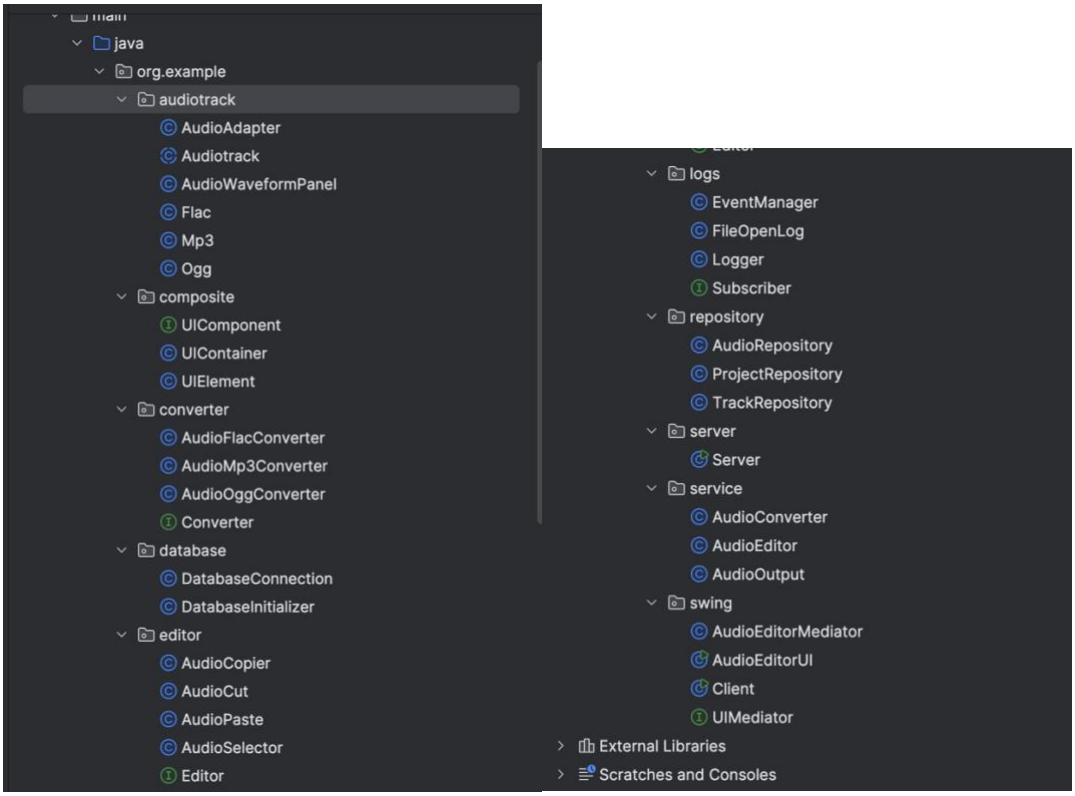

Рис. 3 Основні класи та структура репозиторію

Проєкт аудіоредактора має чітку модульну структуру - пакети розділені за функціональністю, що забезпечує легке масштабування, підтримку та розширення функціоналу.

Основні пакети:

- **audiotrack** - реалізує базову роботу з аудіотреками, підтримку форматів MP3, OGG, FLAC, обробку та візуалізацію хвильової форми.
- **converter** - відповідає за конвертацію між форматами; через інтерфейс Converter реалізовані класи для MP3, OGG, FLAC, що дозволяє легко додавати нові формати.
- **database** - забезпечує роботу з базою даних, включаючи створення, ініціалізацію та управління з’єднанням.

- **editor** - реалізує редагування аудіо; через інтерфейс Editor доступні дії копіювання, вирізання, вставки та виділення частин треків.
- **repository** - містить репозиторії для управління даними про аудіофайли, проєкти та треки, забезпечує збереження та отримання інформації з бази.
- **service** - відповідає за взаємодію з користувачем, містить класи для GUI та головний клас запуску застосунку.

Структура проєкту забезпечує чіткий розподіл обов'язків - зміни та розширення функціоналу можна робити без порушення існуючого коду - зручна обробка аудіо, підтримка різних форматів та інтуїтивні інструменти для редагування й управління файлами.

Структура бази даних

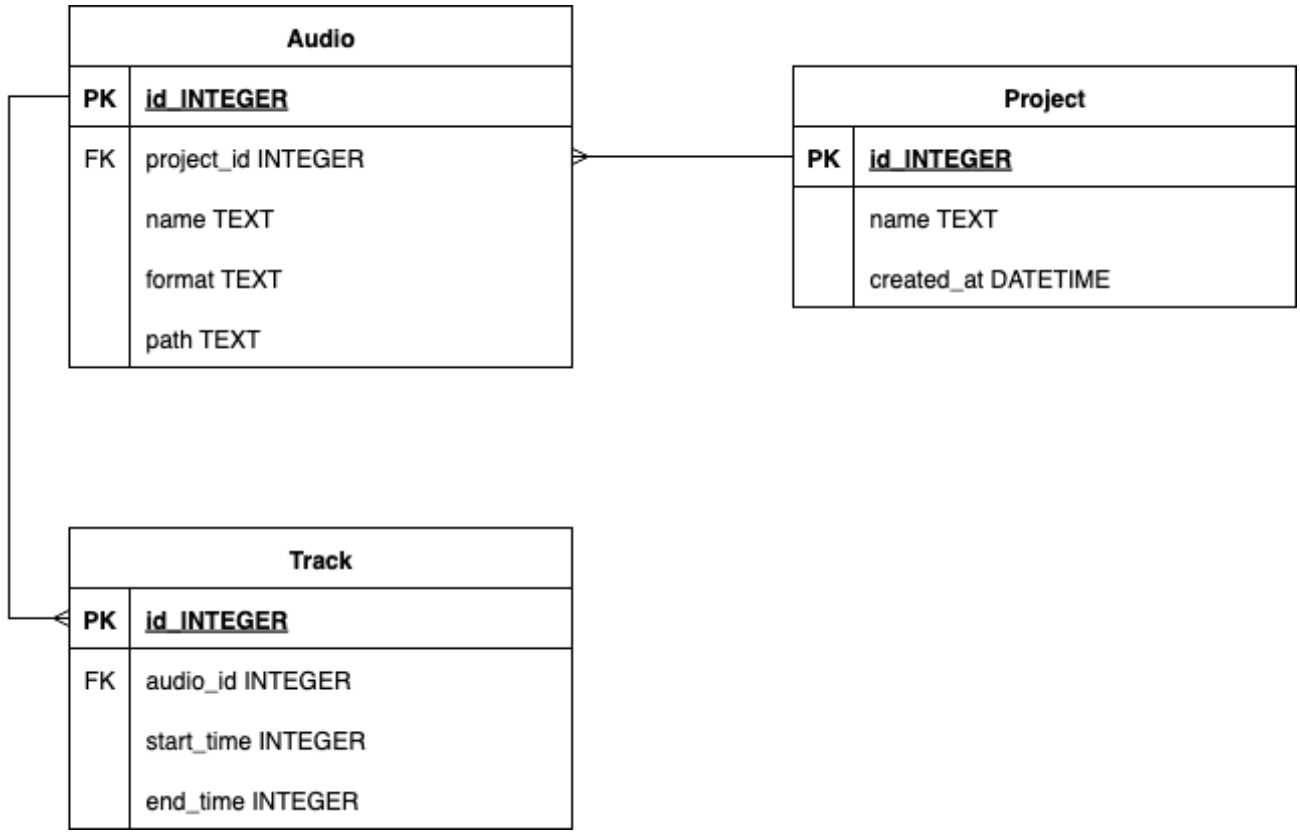

Рис. 4 Структура бази даних

Структура бази даних розроблена для збереження інформації, необхідної для роботи аудіоредактора. Вона включає чотири основні таблиці: **Audio**, **Track**, **Project**, кожна з яких має своє призначення та логічно пов’язана з іншими.

1. Audio

Зберігає дані про аудіофайли.

- **Колонки:**
  - id (INTEGER, PK) - унікальний ідентифікатор аудіофайлу
  - name (TEXT) - назва файлу
  - format (TEXT) - формат файлу (MP3, OGG, FLAC)
  - path (TEXT) - шлях до файлу в системі
    Ця таблиця містить базову інформацію про всі аудіофайли, доступні в проєкті.

2. Таблиця Track

Зберігає інформацію про треки, які є частинами аудіофайлів.

- **Колонки:**
  - id (INTEGER, PK) - унікальний ідентифікатор треку
  - audio_id (INTEGER, FK) - зовнішній ключ на аудіофайл у таблиці Audio
  - start_time (INTEGER) - час початку треку
  - end_time (INTEGER) - час завершення треку
    Таблиця дозволяє працювати з окремими сегментами аудіофайлів.

3. Таблиця Project

Зберігає дані про проєкти користувачів.

- **Колонки:**
  - id (INTEGER, PK) - унікальний ідентифікатор проєкту
  - name (TEXT) - назва проєкту

- created_at (DATETIME) - дата та час створення проєкту Таблиця представляє проєкти, в яких користувач може об’єднувати аудіофайли та працювати з ними.

Зв’язки між таблицями:

- **Audio -> Track:** один аудіофайл може мати кілька треків, кожен трек належить лише одному аудіофайлу.
- **Project -> Audio:** один проєкт може мати кілька аудіофайлів, кожен аудіофайл належить лише одному проєкту.

Поточна структура бази даних забезпечує гнучке управління інформацією - зберігає дані про аудіофайли, дозволяє розбивати їх на треки, організовувати у проєкти та підтримувати зв’язки між ними. Чіткі зв’язки та використання зовнішніх ключів роблять структуру легкою для масштабування та інтеграції з іншими модулями проєкту.

```sql
CREATE TABLE IF NOT EXISTS Audio (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT,
    format TEXT,
    path TEXT
);

CREATE TABLE IF NOT EXISTS Track (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    audio_id INTEGER,
    start_time INTEGER,
    end_time INTEGER,
    FOREIGN KEY (audio_id) REFERENCES Audio(id)
);

CREATE TABLE IF NOT EXISTS Project (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT,
    created_at DATETIME DEFAULT  CURRENT_TIMESTAMP
);
```

Рис. 5 Фіз. модель бази даних

Висновок

У ході виконання даної лабораторної роботи з теми створення аудіоредактора було опрацьовано основи проектування програмних систем із використанням схем прецедентів, діаграм класів та структури бази даних. Було розроблено схему прецедентів, що відображає ключові функції редактора - завантаження аудіо, редагування та експорт файлів.

Побудова діаграми класів дозволила визначити основні компоненти системи, їх атрибути, методи та зв’язки, а також розподіл відповідальності між класами. Обрані прецеденти - завантаження файлу, редагування сегменту та збереження результату - забезпечили чітке розуміння логіки взаємодії користувача з системою.

Створена структура бази даних і основні класи заклали основу для подальшого використання шаблону Репозиторію, що забезпечує ефективну роботу з даними та ізоляцію бізнес-логіки від рівня збереження.

Таким чином, лабораторна робота дозволила сформувати початкову архітектуру аудіоредактора та забезпечила практичне розуміння ключових етапів проектування програмної системи.

Відповіді на теоретичні питання

1. Що таке UML?

UML (Unified Modeling Language) – це уніфікована мова моделювання, яка використовується для візуалізації, специфікації, конструювання та документування об’єктно-орієнтованих систем. UML дозволяє описати структуру та поведінку системи за допомогою різних типів діаграм.

2. Що таке діаграма класів UML?

Діаграма класів UML – це тип діаграми, що показує класи системи, їхні атрибути, методи та зв’язки між класами. Вона використовується для моделювання статичної структури системи.

3. Які діаграми UML називають канонічними?

Канонічні (основні) діаграми UML – це ті, що найчастіше використовуються для опису системи. До них відносять: Діаграма класів, Діаграма об’єктів, Діаграма варіантів використання (Use Case), Діаграма послідовності, Діаграма станів, Діаграма компонентів, Діаграма розгортання (Deployment).

4. Що таке діаграма варіантів використання?

Діаграма варіантів використання (Use Case Diagram) показує функціональні вимоги системи з точки зору користувача. Вона відображає акторів (користувачів або зовнішні системи) і взаємодії між ними та системою.

5. Що таке варіант використання?

Варіант використання (Use Case) – це конкретний сценарій, що описує послідовність дій користувача та системи для досягнення певної мети.

6. Які відношення можуть бути відображені на діаграмі використання?

На діаграмі варіантів використання можуть бути відображені такі відношення:
Асоціація – зв’язок між актором і варіантом використання;

Include (включення) – один варіант використання обов’язково включає інший;
Extend (розширення) – один варіант використання може розширювати поведінку іншого;
Generalization (узагальнення) – один варіант використання або актор є узагальненням іншого.

7. Що таке сценарій?

Сценарій – це конкретний випадок виконання варіанту використання, який описує послідовність дій користувача і системи для досягнення певної мети. Сценарії можуть бути основними або альтернативними.

8. Що таке діаграма класів?

Діаграма класів – графічне представлення класів системи, їх атрибутів, методів та зв’язків між ними. Використовується для моделювання статичної структури програмної системи.

9. Які зв’язки між класами ви знаєте?

Основні зв’язки між класами:
Асоціація – загальний зв’язок між класами;
Узагальнення (Generalization) – спадкування;
Агрегація – слабка «частина-ціле» (клас частково залежить від іншого);
Композиція – сильна «частина-ціле» (життєвий цикл частини залежить від цілого);
Залежність (Dependency) – один клас залежить від іншого лише тимчасово.

10. Чим відрізняється композиція від агрегації?

Агрегація – «слабке» включення: частина може існувати без цілого.
Композиція – «сильне» включення: частина не може існувати без цілого; її життєвий цикл залежить від цілого.

11. Чим відрізняються зв’язки типу агрегації від зв’язків композиції на діаграмах класів?

Агрегація позначається порожнім ромбом на кінці цілого класу.

Композиція позначається заповненим (чорним) ромбом на кінці цілого класу.

12. Що являють собою нормальні форми баз даних?

Нормальні форми – це правила структурування таблиць баз даних, які забезпечують: відсутність надлишкових даних, уникнення аномалій при вставці, оновленні або видаленні даних, правильне поділ даних між таблицями. Основні нормальні форми: 1NF, 2NF, 3NF та BCNF.

13. Що таке фізична модель бази даних? Логічна?

Логічна модель описує структуру даних без прив’язки до конкретної СУБД: таблиці, поля, зв’язки, обмеження.

Фізична модель описує реалізацію бази даних у конкретній СУБД: типи даних, індекси, фізичне розташування таблиць.

14. Який взаємозв’язок між таблицями БД та програмними класами?

Кожен клас у програмі часто відповідає таблиці в базі даних. Атрибути класу – це стовпці таблиці, об’єкти класу – записи (рядки) таблиці, а зв’язки між класами – зовнішні ключі між таблицями.