Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 2

з дисципліни «Технології розроблення програмного забезпечення»

Тема лабораторної роботи: «Діаграма варіантів використання. Сценарії варіантів використання. Діаграми UML. Діаграми класів. Концептуальна модель системи»

Тема проєкта: «5. Аудіо редактор»

Виконала:
студентка групи ІА-33
Котик Іванна

Перевірив:
асистент кафедри ІСТ
Мягкий Михайло Юрійович

Київ 2025

Тема: Діаграма варіантів використання. Сценарії варіантів використання. Діаграми UML. Діаграми класів. Концептуальна модель системи.

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.

Тема проєкта: 5. 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).

1.1. Короткі теоретичні відомості:

Вступ до мови UML

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

У нотації мови UML визначено такі види діаграм:

- варіантів використання (use case diagram);
- класів (class diagram);
- кооперації (collaboration diagram);
- послідовності (sequence diagram);
- станів (statechart diagram);
- діяльності (activity diagram);
- компонентів (component diagram);
- розгортання (deployment diagram).

Діаграма варіантів використання (Use-Cases Diagram)

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.

Сценарії використання

Діаграма варіантів використання надає знання про необхідну функціональність системи в інтуїтивно-зрозумілому вигляді, проте не несе відомостей про фактичний спосіб її реалізації. Конкретні варіанти використання можуть звучати надто загально та не бути придатними для реалізації програмістами.

Для документації варіантів використання у вигляді певної специфікації та усунення неточностей і непорозуміння діаграм варіантів використання, як частину процесу збору та аналізу вимог складаються звані сценарії використання.

Сценарії використання – це текстові уявлення тих процесів, які відбуваються при взаємодії користувачів системи та самої системи. Вони є чітко формалізованими, покроковими інструкціями, що описують той чи інший процес у термінах кроків досягнення мети. Сценарії використання однозначно визначають кінцевий результат.

Сценарії використання описують варіанти використання природною мовою. Вони мають загального, шаблонного вигляду написання, проте рекомендується такий перелік для опису:

- Передумови – умови, які повинні бути виконані для виконання даного варіанту використання;
- Постумови – що виходить в результаті виконання даного варіанту використання;
- Взаємодіючі сторони;

- Короткий опис;
- Основний перебіг подій;
- Винятки; Примітки.

Діаграми 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

Розрізняють дві моделі бази даних – логічну та фізичну. Фізична модель бази даних представляє собор набір бінарних даних у вигляді файлів, структурованих та згрупованих згідно з призначенням (сегменти, екстенти та ін.), що використовується для швидкого та ефективного отримання інформації з жорсткого диска, а також для компактного зберігання та розміщення даних на жорсткому диску.

Логічна модель бази даних є структурою таблиць, уявлень, індексів та інших логічних елементів бази даних, що дозволяють власне програмування та використання бази даних.

Процес створення логічної моделі бази даних зветься проєктування бази даних (database design). Проєктування відбувається у зв'язку з опрацюванням архітектури програмної системи, оскільки база даних створюється зберігання даних, одержуваних з програмних класів.

Відповідно можна розрізнити кілька підходів до зв'язування програмних класів та таблиць: одна таблиця – один клас, одна таблиця – кілька класів, один клас – кілька таблиць. Залежно від обраного підходу, буде ускладнюватись (і уповільнюватися) робота з базою даних при додаванні даних, або отримання даних з БД та парсинг їх у відповідні класи.

З іншого боку, якщо програмні класи є сутності проєктованої системи (елементи предметної області), то таблиці відображають їх технічну реалізацію та спосіб зберігання та зв'язку.

Основним керівництвом під час проєктування таблиць є т. зв. нормальні форми баз даних.

Нормальні форми :

- 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, кожна залежність визначається ключем.

Проєктування БД :

Для проєктування бази даних можна використовувати Schema View у відповідних засобах роботи з СУБД (Microsoft Sql Server Management Studio, PL/SQL Developer та інші) або вбудований засіб Microsoft Office Access. Для створення таблиць необхідно натиснути кнопку «Створити» (CREATE). Це представлено на рисунку 1.15.

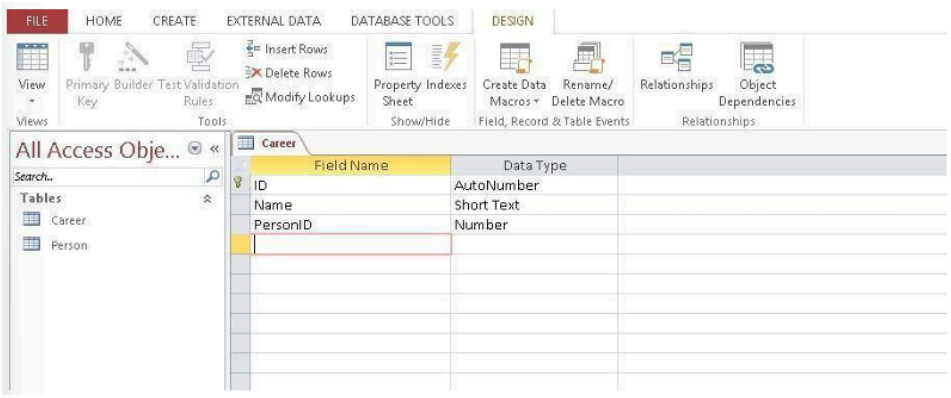

Рисунок 1.1 – Створення таблиці в MS Access

Після заповнення полів та їх типів необхідно призначити первинний ключ. Після цього можна виставити зв'язок між таблицями у вікні «Зв'язки»(Relationships). Зв’язко представлено на рисунку 1.16.

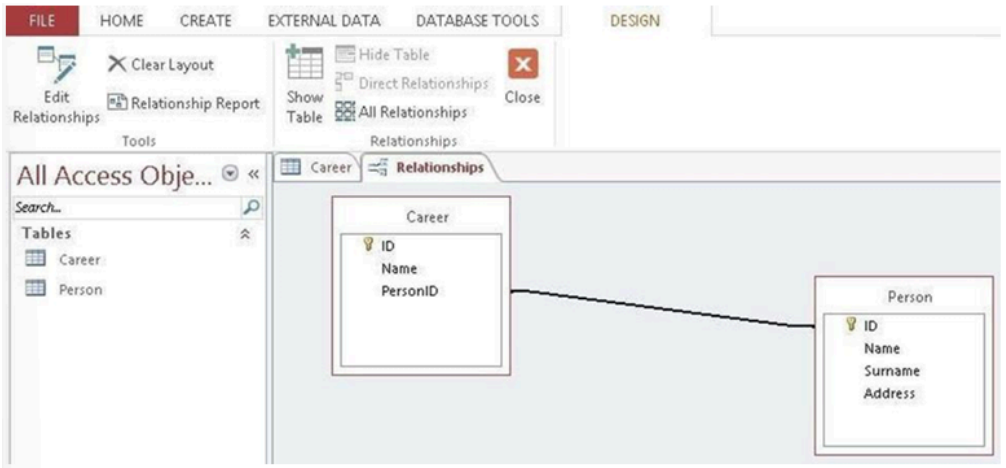

Рисунок 1.2 – Зв’язок між двома таблицями на діаграмі в MS Access

Хід роботи:

2.1. Побудова Use-Cases Diagram.

Опис діаграми прецедентів

Актор: Користувач – основний користувач аудіо редактора, який взаємодіє із системою для створення та редагування аудіо проектів.

Прецеденти:

1. Вхід – користувач входить у систему.
2. Створення проекту – створення нового проекту для роботи з аудіо.
3. Відкриття проекту – завантаження існуючого проекту з бази даних або файлу.
4. Імпорт аудіо – додавання аудіофайлів у проект; включає конвертацію у WAVE через include “Конвертація у WAVE”.
5. Додавання доріжки – створення нової аудіодоріжки в проекті.
6. Видалення доріжки – видалення непотрібної доріжки.

7. Копіювати сегмент – копіювання виділеної частини аудіо; включає Вибрати сегмент.
8. Вирізати сегмент – вирізання виділеного сегмента; включає Вибрати сегмент.
9. Деформація – застосування ефектів до виділеного сегмента; включає Вибрати сегмент.
10. Вставити сегмент – вставка скопійованого або вирізаного сегмента в інше місце.
11. Відтворити аудіо – відтворення поточного проекту.
12. Зупинити аудіо – пауза або зупинка відтворення.
13. Експорт – збереження проекту у потрібному форматі; включає Кодування у формат.

Особливості діаграми:

Використання відношень <<include>> для “Конвертації у WAVE”, “Вибір сегмента” та “Кодування у формат” демонструє обов’язкові кроки, що виконуються під час відповідних операцій.

Тобто умовно можна сказати, що кожне використання варіанта використання “Імпорт аудіо” завжди включатиме виконання варіанта використання “Конвертації у WAVE”, а “Експорт” завжди включатиме – “Кодування у формат” і т. д.

Таким чином, побудована діаграма прецедентів, що зображена на рис.2.1.1, відображає взаємодію користувача із системою, послідовність операцій та залежності між основними функціями аудіо редактора.

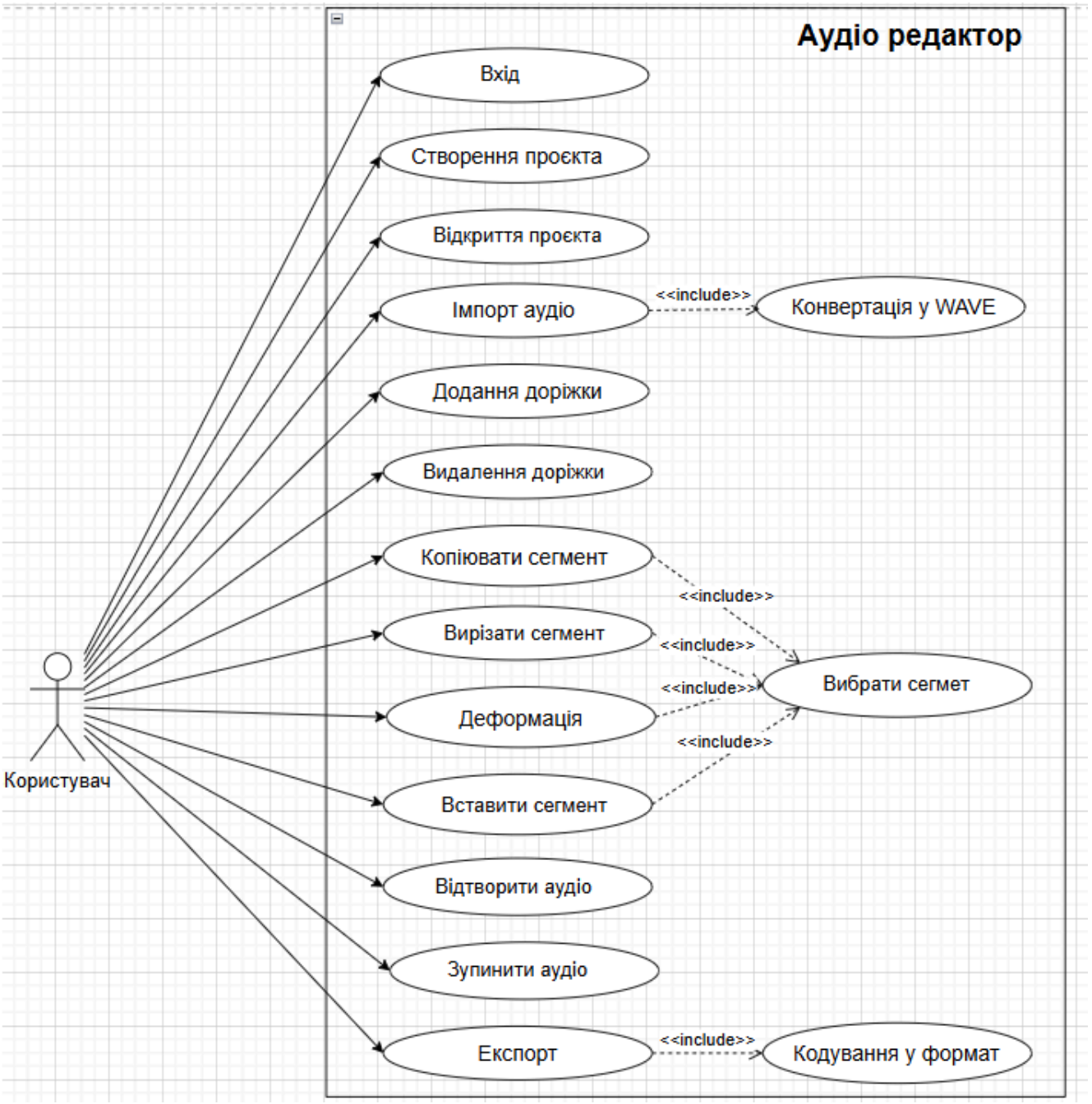

Рисунок 2.1.1 – Діаграма варіантів використання для аудіо редактора

2.2. Спроєктуємо діаграму класів для предметної області.

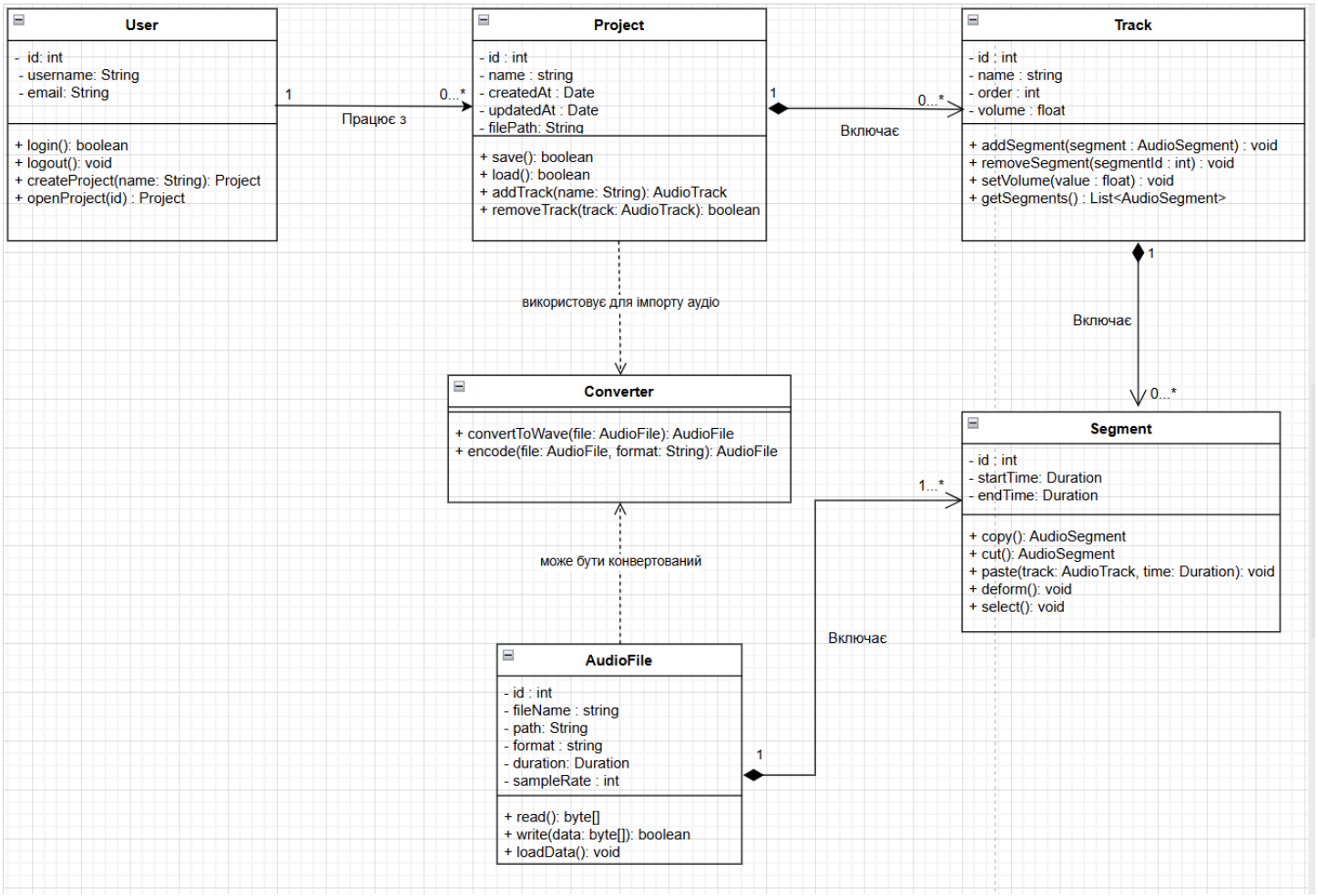

Рисунок 2.2.1 – Діаграма класів для предметної області

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

2.3.1. Сценарій: «Відтворення аудіо»:

Передумови: Відкритий проєкт, що містить щонайменше один аудіосегмент.

Постумови: Система відтворює аудіо, починаючи з поточної позиції, і оновлює курсор відтворення.

Взаємодіючі сторони: Користувач, Система.

Короткий опис: Запускає відтворення аудіо з усіх доріжок проєкту.

Основний перебіг подій:

1. Користувач натискає кнопку «Відтворити».

2. Система починає читати дані з усіх AudioSegment на AudioTrack і направляти їх на аудіовихід.
3. Курсор відтворення на шкалі часу починає рухатися.
4. Користувач натискає кнопку «Зупинити».
5. Система припиняє читання даних, і відтворення зупиняється.

Винятки:

Виняток №1: Не знайдено аудіопристрій. Система виводить повідомлення про помилку, відтворення не починається.

Виняток №2: Пошкоджений аудіофайл. Система може зупинити відтворення або пропустити пошкоджену частину, виводячи повідомлення про помилку.

Примітки: На відтворення впливають налаштування гучності та статус окремих доріжок.

2.3.2. Сценарій: «Збереження проєкту»

Передумови: Відкритий проєкт із внесеними змінами, які ще не були збережені.

Постумови: Усі зміни проєкту (доріжки, сегменти, налаштування) успішно збережені у файл на диску.

Взаємодіючі сторони: Користувач, Система.

Короткий опис: Зберігає поточний стан проєкту у файлову систему, дозволяючи продовжити роботу пізніше.

Основний перебіг подій:

1. Користувач ініціює дію збереження (наприклад, натискає «Ctrl+S» або обирає «Зберегти проєкт»).
2. Система перевіряє, чи має проєкт існуючий шлях збереження.

3. Якщо проєкт зберігається вперше, система запитує у користувача місце для збереження.
4. Система збирає дані про всі AudioTrack та AudioSegment і записує їх у файл проєкту.
5. Система повідомляє про успішне збереження.

Винятки:

Виняток №1: Недостатньо місця на диску. Система виводить повідомлення про помилку, виконання завершується.

Виняток №2: Некоректна назва файлу. Це виникає, якщо користувач вводить заборонені символи (/, \, ?, *, " та інші) в назві файлу. Операційна система не дозволить створити такий файл. Система повинна видати повідомлення з поясненням, що назва файлу є недійсною.

Примітки: При повторному збереженні старий файл проєкту буде перезаписано.

2.3.3. Сценарій: «Видалення аудіосегмента»

Передумови: Відкритий проєкт. На доріжці є щонайменше один сегмент.

Постумови: Обраний сегмент видаляється з доріжки.

Взаємодіючі сторони: Користувач, Система.

Короткий опис: Цей варіант використання дозволяє користувачеві видалити небажаний сегмент з доріжки.

Основний перебіг подій:

1. Користувач вибирає сегмент на дорішці, натиснувши на нього.
2. Користувач ініціює дію «Видалити» (наприклад, натискає клавішу «Delete» або кнопку на панелі інструментів).

3. Система перевіряє, чи було вибрано сегмент.
4. Система видаляє сегмет із відповідної доріжки та з пам'яті.
5. Система оновлює візуальне відображення доріжки на інтерфейсі, заповнюючи простір, де був сегмент.
6. Система повідомляє про успішне видалення.

Винятки:

Виняток №1: Не вибрано сегмент. Якщо користувач намагається видалити сегмент, не вибравши його, система виводить повідомлення про помилку («Будь ласка, виберіть сегмент для видалення») та припиняє виконання.

Примітки: Видалення сегмента з проєкту не впливає на вихідний аудіофайл, що зберігається на диску. Цей сегмент можна відновити.

2.4. На основі спроєктованої діаграми класів предметної області розробимо основні класи та структуру бази даних системи.

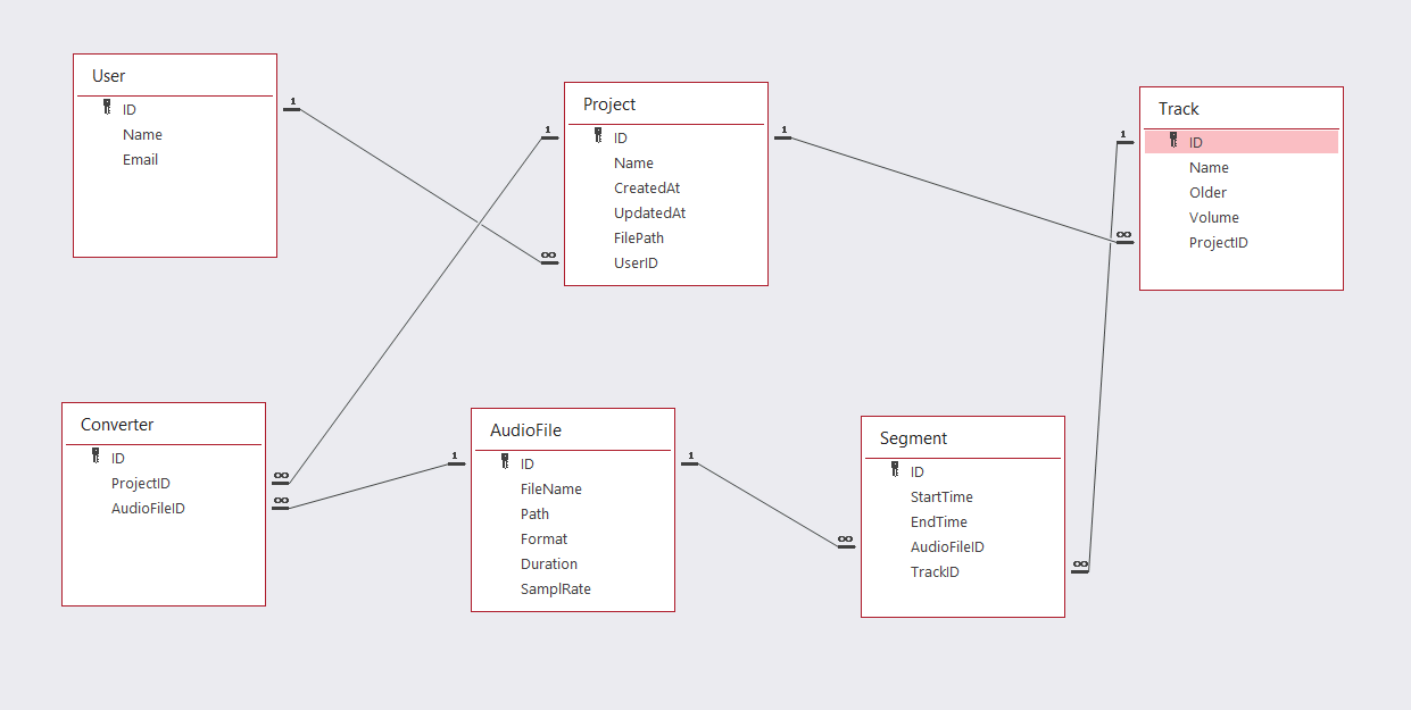

Рисунок 3.4.1 – База даних системи в MS Access

Схема данных | User

Имя поля	Тип данных
ID	Счетчик
Name	Короткий текст
Email	Короткий текст

Схема данных | Project

Имя поля	Тип данных
ID	Счетчик
Name	Короткий текст
CreatedAt	Дата и время
UpdatedAt	Дата и время
FilePath	Гиперссылка
UserID	Числовой

Схема данных | AudioFile

Имя поля	Тип данных
ID	Счетчик
FileName	Короткий текст
Path	Гиперссылка
Format	Короткий текст
Duration	Дата и время
SamplRate	Числовой

Схема данных | Track

Имя поля	Тип данных
ID	Счетчик
Name	Короткий текст
Older	Числовой
Volume	Числовой
ProjectID	Числовой

Схема данных | Segment

Имя поля	Тип данных
ID	Счетчик
StartTime	Дата и время
EndTime	Дата и время
AudioFileID	Числовой
TrackID	Числовой

Схема данных | Converter

Имя поля	Тип данных
ID	Счетчик
ProjectID	Числовой
AudioFileID	Числовой

Рисунок 3.4.2 – Класи бази даних системи в MS Access

2.5. Нарисуємо діаграму класів для реалізованої частини системи.

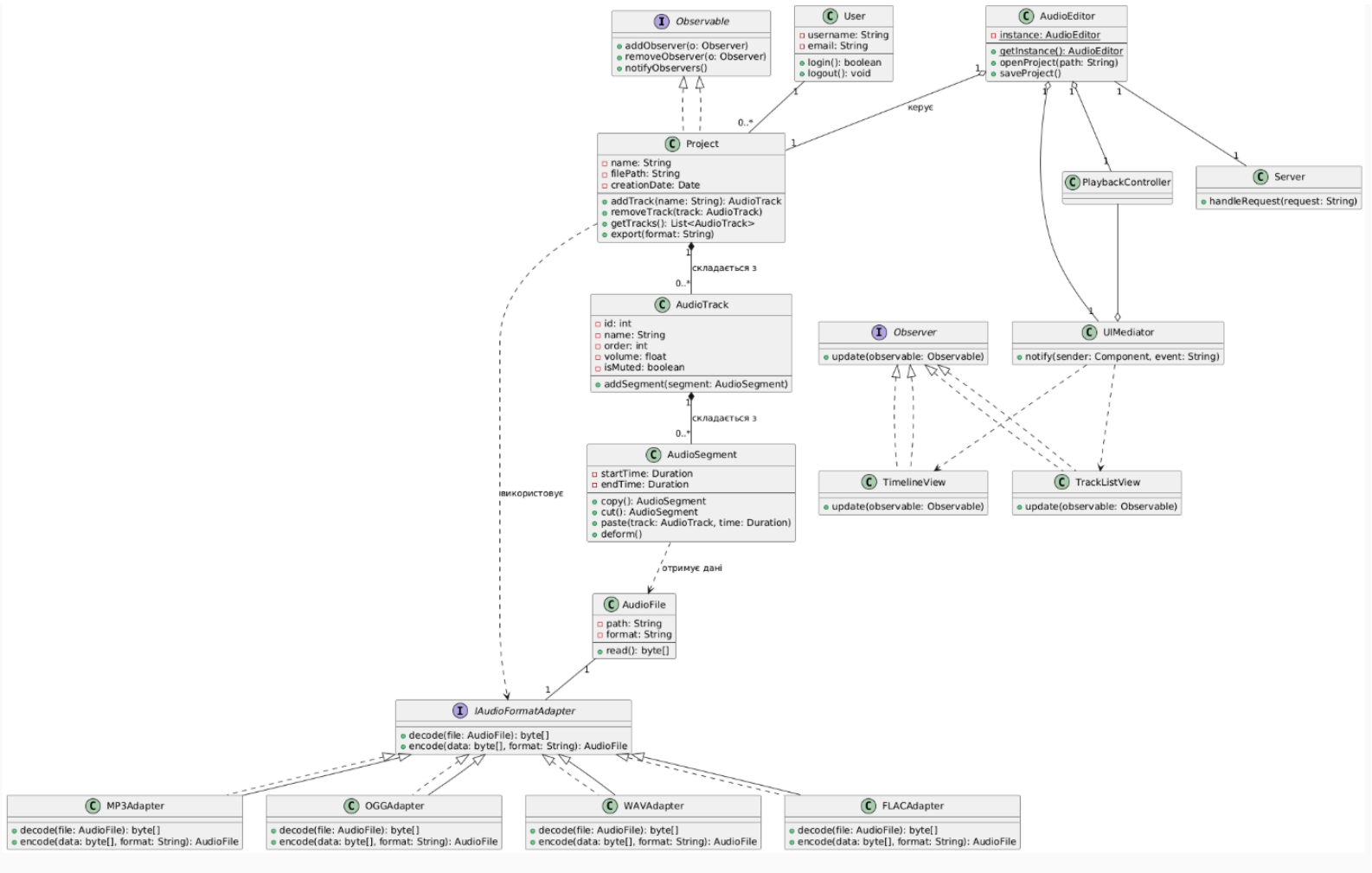

Рисунок 2.5.1 – Діаграма класів для реалізованої частини системи

Примітка: діаграму було побудовано в середовищі PlantUML.

Код для діаграми:

```
@startuml
interface Observable {
  +addObserver(o: Observer)
  +removeObserver(o: Observer)
  +notifyObservers()
}

interface Observer {
  +update(observable: Observable)
}

class User {
  -username: String
  -email: String
```

```
  --
  +login(): boolean
  +logout(): void
}

class AudioEditor {
  -instance: AudioEditor {static}
  --
  +getInstance(): AudioEditor {static}
  +openProject(path: String)
  +saveProject()
}

class Project implements Observable {
  -name: String
  -filePath: String
  -creationDate: Date
  --
  +addTrack(name: String): AudioTrack
  +removeTrack(track: AudioTrack)
  +getTracks(): List<AudioTrack>
  +export(format: String)
}

class AudioTrack {
  -id: int
  -name: String
  -order: int
  -volume: float
  -isMuted: boolean
  --
  +addSegment(segment: AudioSegment)
}

class AudioSegment {
  -startTime: Duration
  -endTime: Duration
  --
  +copy(): AudioSegment
  +cut(): AudioSegment
  +paste(track: AudioTrack, time: Duration)
  +deform()
}

class AudioFile {
  -path: String
  -format: String
  --
  +read(): byte[]
}

interface IAudioFormatAdapter {
  +decode(file: AudioFile): byte[]
  +encode(data: byte[], format: String): AudioFile
}

class MP3Adapter implements IAudioFormatAdapter {
```

```
  +decode(file: AudioFile): byte[]
  +encode(data: byte[], format: String): AudioFile
}

class OGGAdapter implements IAudioFormatAdapter {
  +decode(file: AudioFile): byte[]
  +encode(data: byte[], format: String): AudioFile
}

class WAVAdapter implements IAudioFormatAdapter {
  +decode(file: AudioFile): byte[]
  +encode(data: byte[], format: String): AudioFile
}

class FLACAdapter implements IAudioFormatAdapter {
  +decode(file: AudioFile): byte[]
  +encode(data: byte[], format: String): AudioFile
}

class UIMediator {
  +notify(sender: Component, event: String)
}

class PlaybackController {
}

class TimelineView implements Observer {
  +update(observable: Observable)
}

class TrackListView implements Observer {
  +update(observable: Observable)
}

class Server {
  +handleRequest(request: String)
}

' Зв'язки
User "1" -- "0..*" Project
Project "1" *-- "0..*" AudioTrack : "складається з"
AudioTrack "1" *-- "0..*" AudioSegment : "складається з"
AudioSegment ..> AudioFile : "отримує дані"

AudioFile "1" -- "1" IAudioFormatAdapter
Project ..> IAudioFormatAdapter : "використовує"

AudioEditor "1" o-- "1" Project : "керує"
AudioEditor "1" o-- "1" UIMediator
AudioEditor "1" o-- "1" PlaybackController
AudioEditor "1" -- "1" Server

' Патерни
IAudioFormatAdapter <|-- MP3Adapter
IAudioFormatAdapter <|-- OGGAdapter
IAudioFormatAdapter <|-- WAVAdapter
IAudioFormatAdapter <|-- FLACAdapter
```

```
Observable <|.. Project
Observer <|.. TimelineView
Observer <|.. TrackListView

UIMediator ..> TimelineView
UIMediator ..> TrackListView
PlaybackController --o UIMediator
@enduml
```

Висновки:

У процесі виконання цієї лабораторної роботи ми набули практичних навичок з проєктування програмних систем. Було розроблено ключові UML-діаграми для системи «Аудіо редактор», включаючи діаграму варіантів використання, діаграму класів предметної області та діаграму реалізованої частини системи. Застосовуючи принципи об'єктно-орієнтованого проєктування та шаблони, що ми розглянули, ми змогли створити модель. Отримані результати демонструють вміння використовувати UML як інструмент для розробки сучасної та масштабованої архітектури.