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: «Аудіо редактор»

Виконала:
студентка групи ІА-33
Котик Іванна

Перевірив:
асистент кафедри ІСТ
Мягкий Михайло Юрійович

Київ 2025

Зміст

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. 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.

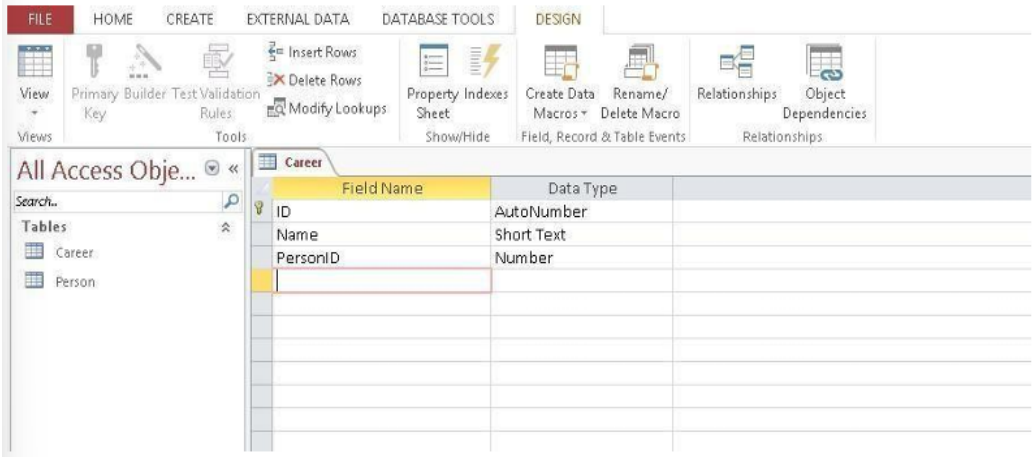


Рисунок 1.1 – Створення таблиці в MS Access

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.

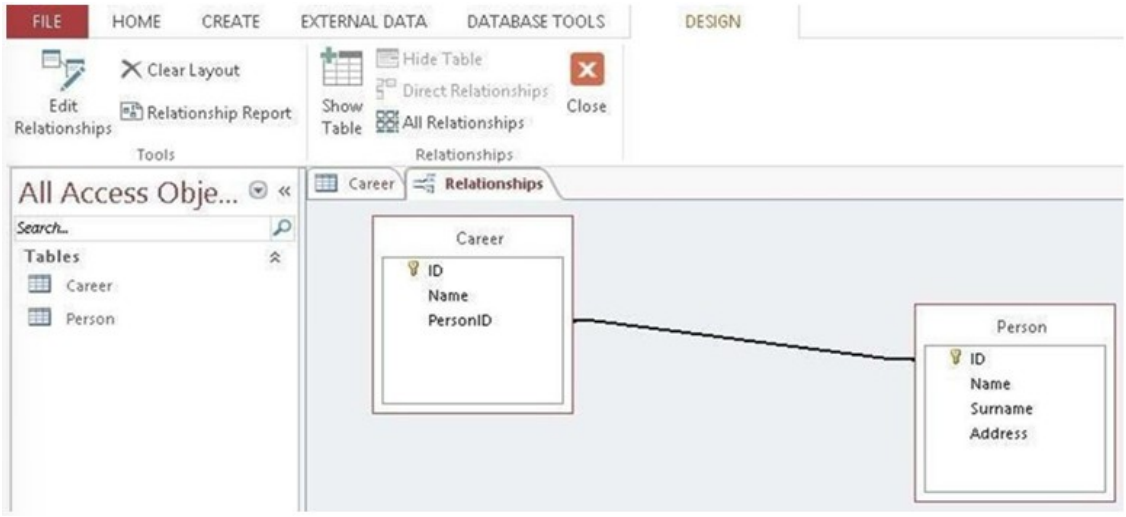


Рисунок 1.2 – Зв’язок між двома таблицями на діаграмі в MS Access

Хід роботи:

2.1. Побудова Use-Cases Diagram.

Опис варіантів використання «Аудіо редактор»

Актор: Користувач – взаємодіє з системою для створення, редагування та відтворення аудіопроєктів.

Основні варіанти використання:

Вхід – користувач входить у систему.

Створення проєкта – дозволяє почати новий проєкт, включає:

- Імпорт аудіо – завантаження існуючих файлів.
- Конвертація у WAVE – автоматична конвертація у внутрішній формат для роботи.

Редагування аудіо – узагальнений сценарій, що включає:

- Додавання доріжки.
- Видалення доріжки.
- Копіювати сегмент.
- Вставити сегмент (вимагає Вибрати сегмент).
- Вирізати сегмент (вимагає Вибрати сегмент).
- Деформація (вимагає Вибрати сегмент).
- Відтворити сегмент аудіо (вимагає Вибрати сегмент).

Відтворити аудіо – програвання всього проєкту.
Зупинити аудіо – зупинка відтворення.
Експорт – збереження результату у вибраному форматі, включає:

- Кодування у формат (mp3, ogg, flac тощо).

Ключові особливості діаграми:

- Є один основний актор (Користувач).

- Є базові дії (створення, редагування, відтворення, експорт).
- Використані відношення <<include>> для позначення підпроцесів (наприклад, «Редагування аудіо» включає «Вибрати сегмент»).
- Є як робота з цілим проєктом, так і з окремими його частинами (сегменти, доріжки).

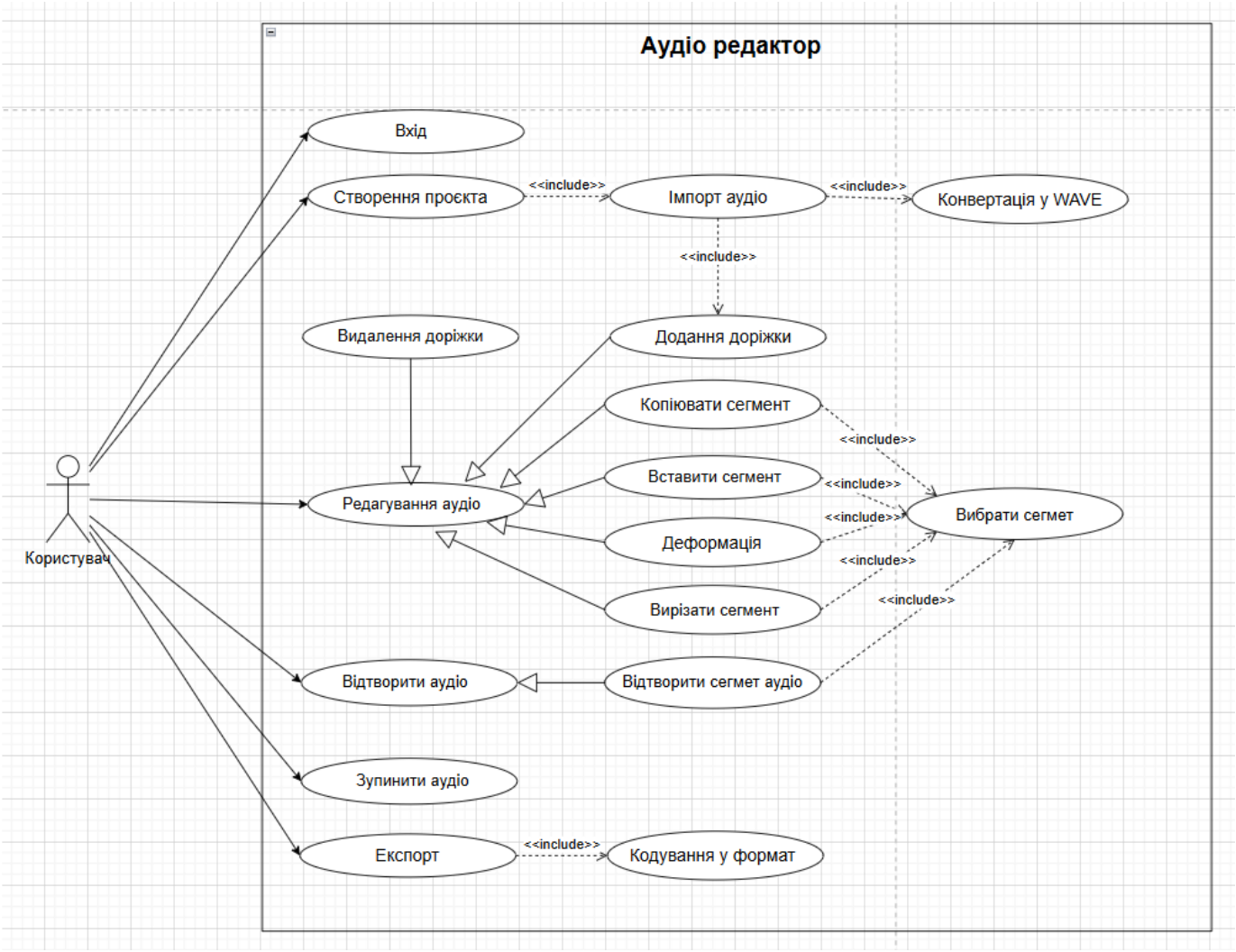


Рисунок 2.1.1 – Діаграма варіантів використання для аудіо редактора

2.2. Спроєктуємо діаграму класів для предметної області.

Опис діаграми класів для предметної області:

- Користувач (User) створює новий або відкриває існуючий проєкт (Project).

- Кожен проєкт містить одну або кілька аудіодоріжок (Track), які організовують аудіоматеріал у логічні шари.
- Кожна доріжка складається з нульової або багатьох частин (Segment), що посилаються на фізичні аудіофайли (AudioFile).
- Операції редагування (копіювання, вирізання, вставка, деформація) застосовуються до сегментів, не змінюючи оригінальні аудіофайли.
- Converter забезпечує сумісність та конвертацію аудіофайлів для роботи в системі.

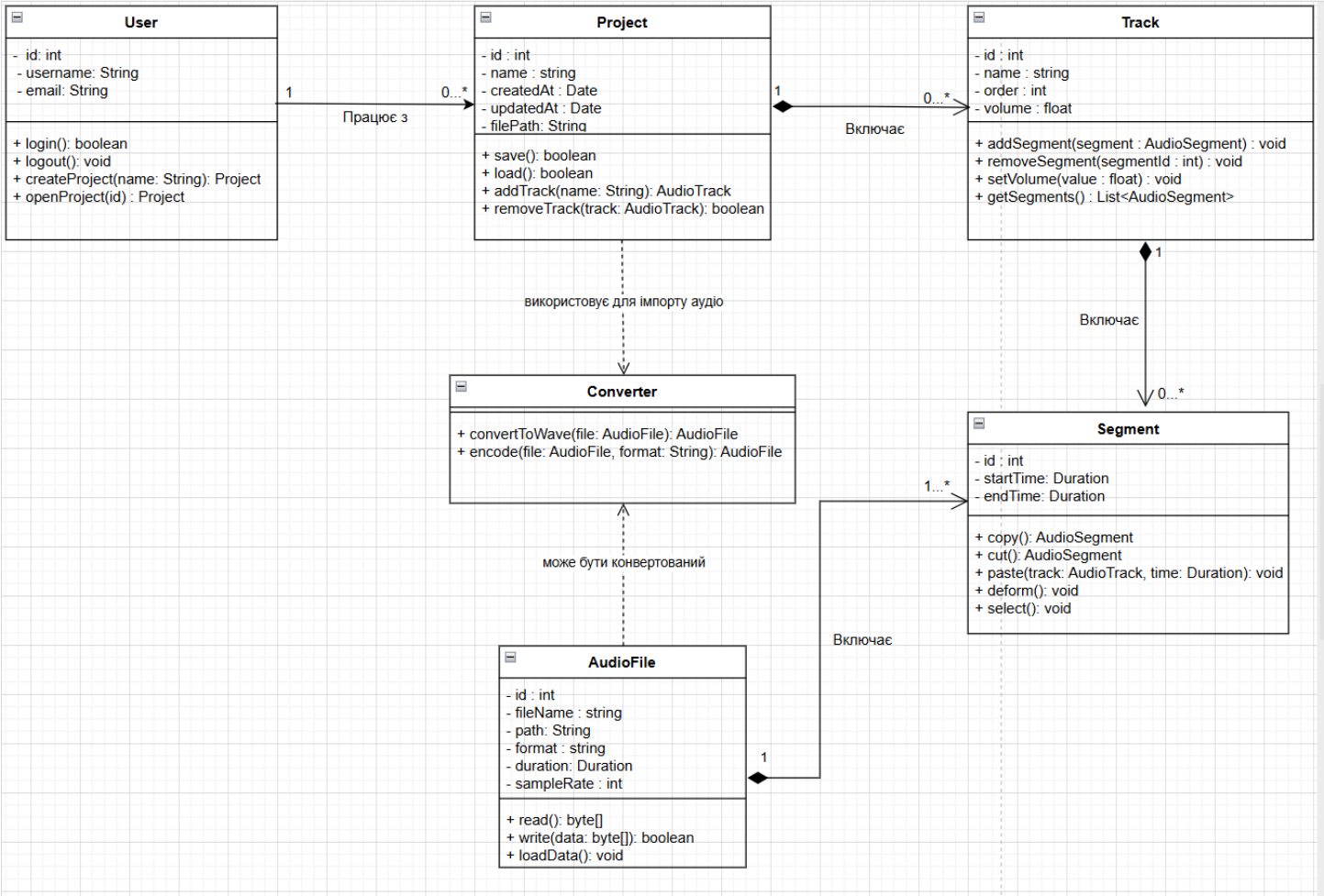


Рисунок 2.2.1 – Діаграма класів для предметної області

2.3. Оберемо 3 варіанти використання та напишемо 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.

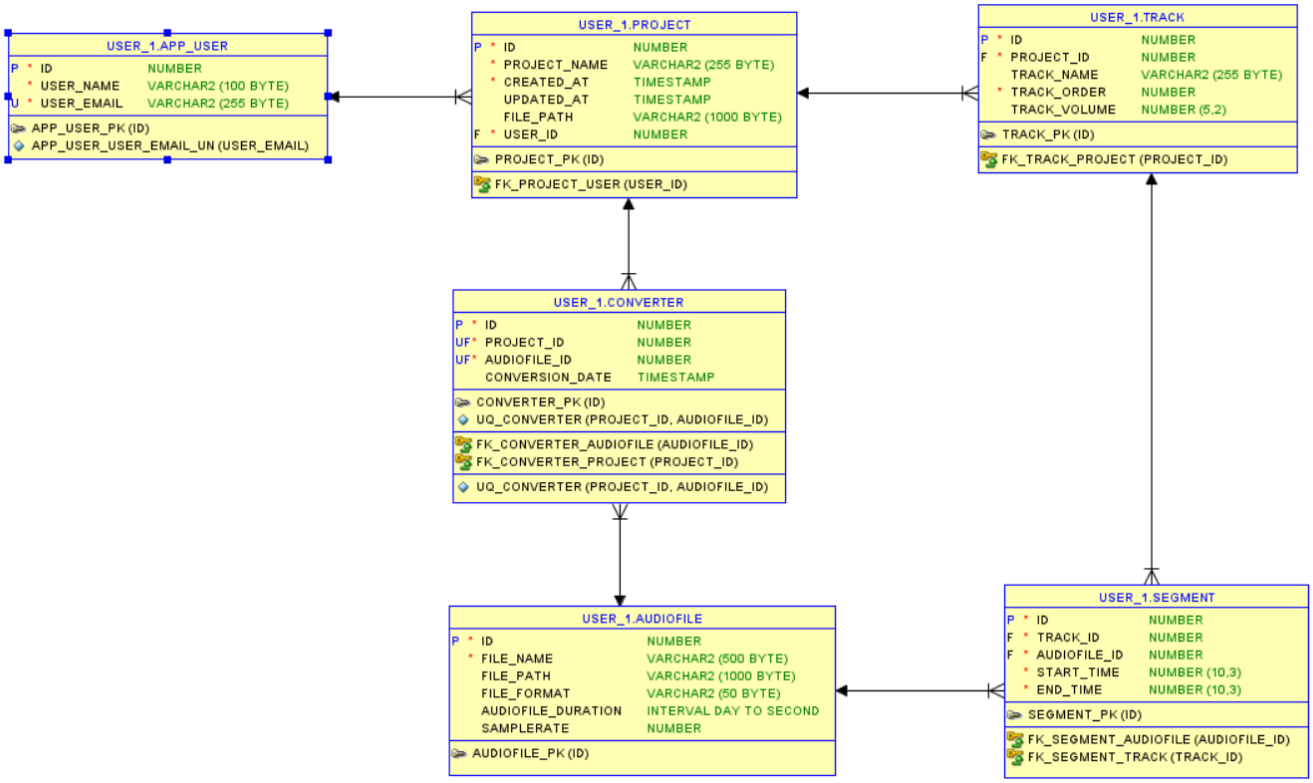


Рисунок 3.4.1 – База даних системи в SQL Developer

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	NUMBER	No	"USER_1"."ISEQ$$_75711".nextval	1	(null)
2	USER_NAME	VARCHAR2(100 BYTE)	No	(null)	2	(null)
3	USER_EMAIL	VARCHAR2(255 BYTE)	No	(null)	3	(null)

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	NUMBER	No	"USER_1"."ISEQ$$_75718".nextval	1	(null)
2	PROJECT_NAME	VARCHAR2(255 BYTE)	No	(null)	2	(null)
3	CREATED_AT	TIMESTAMP(6)	No	SYSTIMESTAMP	3	(null)
4	UPDATED_AT	TIMESTAMP(6)	Yes	(null)	4	(null)
5	FILE_PATH	VARCHAR2(1000 BYTE)	Yes	(null)	5	(null)
6	USER_ID	NUMBER	No	(null)	6	(null)

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	NUMBER	No	"USER_1"."ISEQ$$_75715".nextval	1	(null)
2	FILE_NAME	VARCHAR2(500 BYTE)	No	(null)	2	(null)
3	FILE_PATH	VARCHAR2(1000 B...	Yes	(null)	3	(null)
4	FILE_FORMAT	VARCHAR2(50 BYTE)	Yes	(null)	4	(null)
5	AUDIOFILE_DURATION	INTERVAL DAY(2)...	Yes	(null)	5	(null)
6	SAMPLERATE	NUMBER	Yes	(null)	6	(null)

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	NUMBER	No	"USER_1"."ISEQ$$_75721".nextval	1	(null)
2	PROJECT_ID	NUMBER	No	(null)	2	(null)
3	TRACK_NAME	VARCHAR2(255 BYTE)	Yes	(null)	3	(null)
4	TRACK_ORDER	NUMBER	No	(null)	4	(null)
5	TRACK_VOLUME	NUMBER(5,2)	Yes	1.0	5	(null)

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	NUMBER	No	"USER_1"."ISEQ$$_75724".nextval	1	(null)
2	TRACK_ID	NUMBER	No	(null)	2	(null)
3	AUDIOFILE_ID	NUMBER	No	(null)	3	(null)
4	START_TIME	NUMBER(10,3)	No	(null)	4	(null)
5	END_TIME	NUMBER(10,3)	No	(null)	5	(null)

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	NUMBER	No	"USER_1"."ISEQ$$_75759".nextval	1	(null)
2	PROJECT_ID	NUMBER	No	(null)	2	(null)
3	AUDIOFILE_ID	NUMBER	No	(null)	3	(null)
4	CONVERSION_DATE	TIMESTAMP(6)	Yes	SYSTIMESTAMP	4	(null)

Рисунок 3.4.2 – Класи бази даних системи в SQL Developer

2.5. Нарисуємо діаграму класів для реалізованої частини системи.

Опис діаграми класів для реалізованої частини системи:

У системі «Аудіо редактор» реалізовано кілька ключових сутностей та використано сучасні шаблони проєктування.

- User – представляє користувача, який створює та зберігає проєкти.
- Project – основна сутність, що містить аудіодоріжки та реалізує інтерфейс Observable для сповіщення представлень про зміни.
- AudioTrack – об’єднує кілька сегментів аудіо та дозволяє керувати їхнім порядком і параметрами.

- AudioSegment – описує окремий фрагмент аудіо з можливістю копіювання, вирізання та вставки.
- AudioFile – відповідає за збереження та читання файлів.

Для роботи з різними форматами використано інтерфейс IAudioFormatAdapter і його реалізації (MP3Adapter, OGGAdapter, WAVAdapter, FLACAdapter), що реалізують шаблон Adapter.

Клас AudioEditor реалізує шаблон Singleton і виступає єдиною точкою керування системою.

UIMediator координує взаємодію між компонентами інтерфейсу (шаблон Mediator).

TimelineView та TrackListView реалізують інтерфейс Observer і оновлюються при змінах у проєкті (шаблон Observer).

Ієрархія Project → AudioTrack → AudioSegment побудована за принципом Composite, що дозволяє працювати з аудіо як зі складеною структурою.

Доступ до бази даних реалізовано через шар Repository: інтерфейс IProjectRepository та клас OracleProjectRepository, що інкапсулює роботу з Oracle DB. Бізнес-логіка винесена у ProjectService, який залежить від репозиторію.

Також у системі передбачено взаємодію з Server, що забезпечує принцип Client–Server для віддаленого зберігання чи обробки даних.

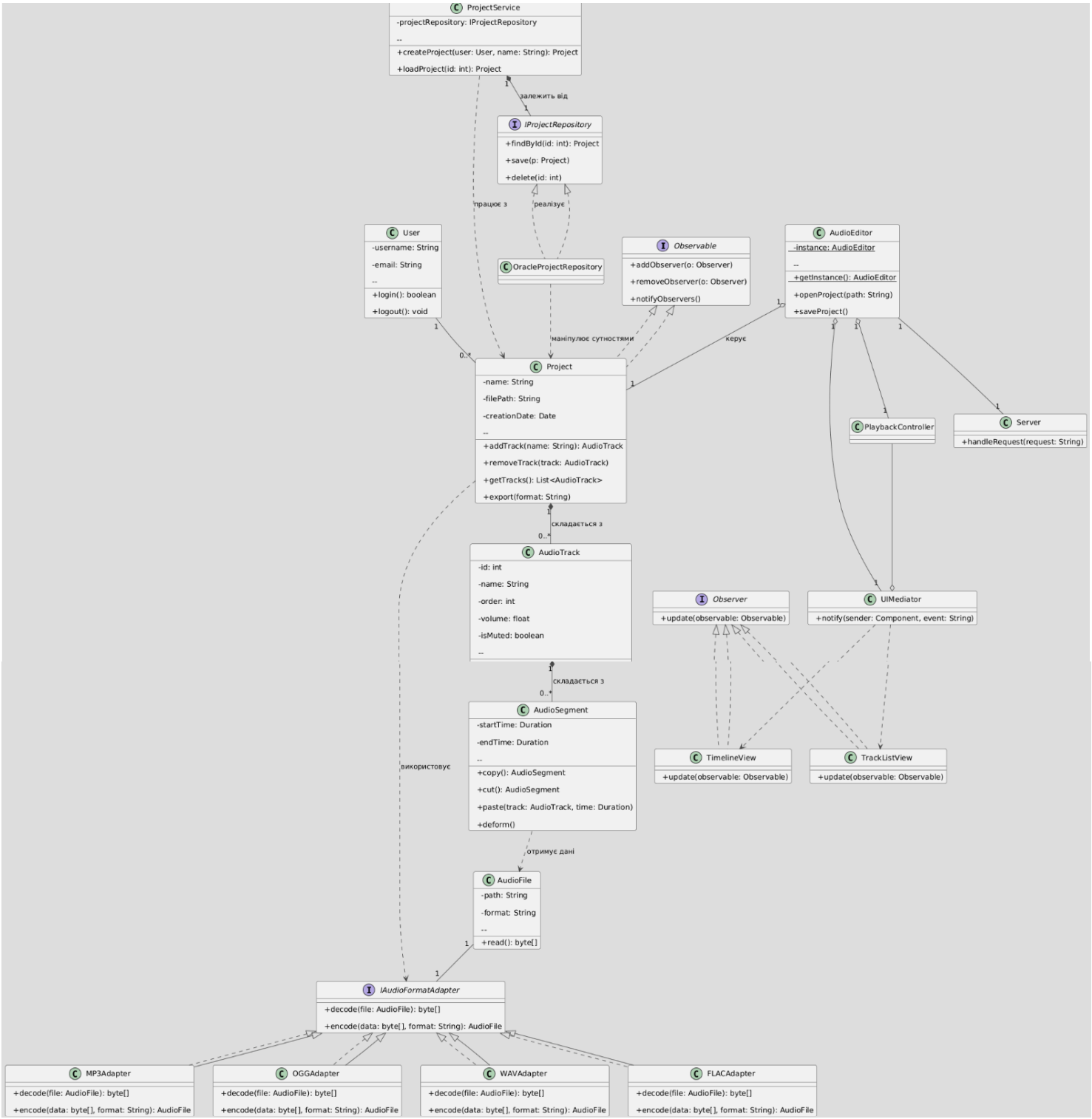


Рисунок 2.5.1 – Діаграма класів для реалізованої частини системи

Примітка: діаграму було побудовано в середовищі PlantUML.

3. Відповіді на контрольні запитання:

1. Що таке UML?

UML - це загальноцільова мова візуального моделювання, яка розроблена для специфікації, візуалізації, проєктування та документування компонентів програмного забезпечення, бізнес-процесів та інших систем.

2. Що таке діаграма класів UML?

Діаграма класів UML - логічна модель, що відбиває статичні аспекти структурної побудови складної системи.

3. Які діаграми UML називають канонічними?

Канонічні діаграми UML - базові діаграми, які входять у стандарт і є найчастіше вживаними для моделювання. Нотація канонічних діаграм є основним засобом розробки моделей мовою UML.

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).*

4. Що таке діаграма варіантів використання?

Діаграма варіантів використання (Use-Cases Diagram) – це UML діаграма за допомогою якої у графічному вигляді можна зобразити вимоги до системи, що розробляється. Діаграма варіантів використання – це вихідна концептуальна модель проєктованої системи, вона не описує внутрішню побудову системи.

5. Що таке варіант використання?

Варіант використання - це опис дій, що виконуються системою та акторами для досягнення певної мети. Він описує, що система робить, не вказуючи, як саме.

6. Які відношення можуть бути відображені на діаграмі використання?

Відношення можуть бути відображені на діаграмі використання:

- *Асоціація (association) – узагальнене, невідоме ставлення між актором та варіантом використання. Розрізняють ненаправлену (двонаправлену) асоціацію - показує взаємодію без акцента на напрямок, та однонаправлену асоціацію - показує, що актор асоціюється з варіантом використання але показує, що варіант використання ініціалізується актором.*

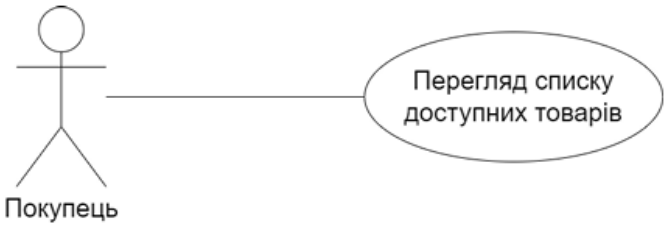


Рисунок 3.1 - Ненаправлена асоціація

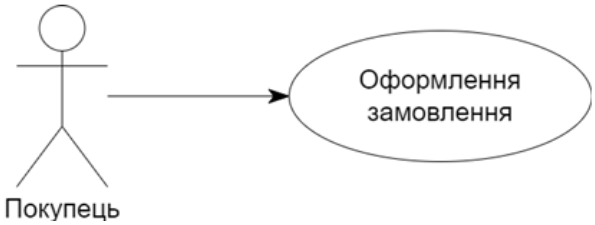


Рисунок 3.2 - Спрямована асоціація

- *Відношення узагальнення (generalization) – показує, що нащадок успадковує атрибути у свого прямого батьківського елементу. Тобто, один елемент моделі є спеціальним або окремим випадком іншого елемента моделі.*

Рисунок 3.3 - Відношення узагальнення

- *Відношення залежності (dependency) визначається як форма взаємозв'язку між двома елементами моделі, призначена для специфікації тієї обставини, що зміна одного елемента моделі призводить до зміни деякого іншого елемента.*
- *Відношення включення (include) – окремий випадок загального відношення залежності між двома варіантами використання, при якому деякий варіант використання містить поведінку, визначену в іншому варіанті використання.*

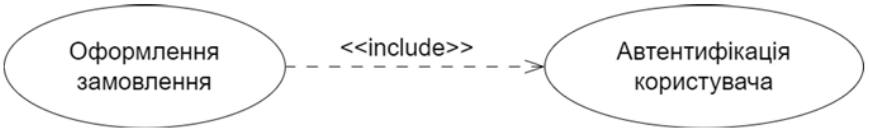


Рисунок 3.4 - Відношення включення

- *Відношення розширення (extend) – показує, що варіант використання розширює базову послідовність дій та вставляє власну послідовність. У цьому на 25 відміну типу відносин «включення» розширена послідовність може здійснюватися залежно від певних умов.*

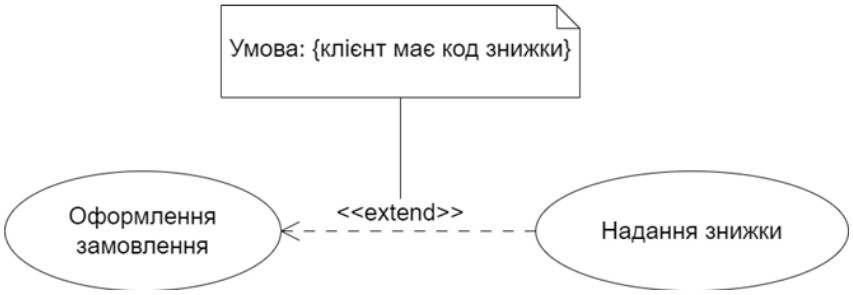


Рисунок 3.5 - Приклад відношення розширення

7. Що таке сценарій?

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.

8. Що таке діаграма класів?

Діаграми класів - це структурна діаграма UML, яка використовуються при моделюванні програмних систем найчастіше. Вони є однією із форм статичного опису системи з погляду її проєктування, показуючи її структуру. Діаграма класів не відображає динамічної поведінки об'єктів зображених на ній класів. На діаграмах класів показуються класи, інтерфейси та відносини між ними.

9. Які зв’язки між класами ви знаєте?

Відносини між класами:

- *Асоціація - це загальний зв'язок, який показує, що один клас знає про існування іншого і може з ним взаємодіяти.*

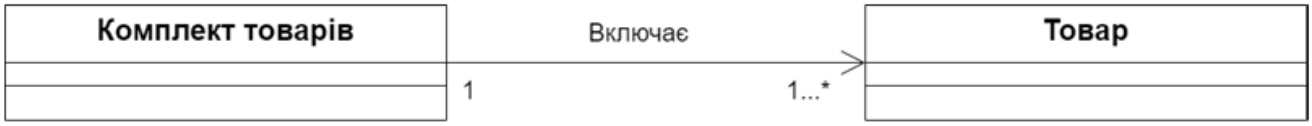


Рисунок 3.6 - Асоціація між класами

- *Узагальнення - на діаграмах класів використовується, щоб показати зв'язок між класом-батьком та класом-нащадком.*

Рисунок 3.7 - Приклад наслідування

- *Агрегацією позначається відношення частина-ціле, коли об'єкти одного класу входять до об'єкта іншого класу. Типовим прикладом таких відносин є списки об'єктів. У цьому випадку список буде виступати агрегатом, а об'єкти, що входять до списку, елементами, що агрегуються.*

Риснок 3.8 - Приклад використання агрегації

- *Композиція - це форма агрегації, де частина не може існувати без об'єкта цілого.*

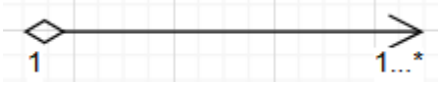
Рисунок 3.9 - Приклад композиції

- *Реалізація — це зв'язок між класом і інтерфейсом. Клас, що реалізує інтерфейс, зобов'язується надати реалізацію (тіло) для всіх методів, визначених в інтерфейсі.*

10. Чим відрізняється композиція від агрегації?

Композиція – це сильний зв’язок «частина–ціле», коли частина не може існувати без цілого, а агрегація – слабкий зв’язок «частина–ціле», коли частина може існувати окремо .

11. Чим відрізняється зв'язки типу агрегації від зв'язків композиції на діаграмах класів?

Агрегація - порожній ромб: 

Композиція - заповнений ромб: 

12. Що являють собою нормальні форми баз даних?

Нормальні форми БД - властивість відношення в реляційній моделі даних, що характеризує його з погляду надмірності, що потенційно призводить до логічно помилкових результатів вибірки або зміни даних.

- *1НФ: усі дані атомарні (немає списків чи повторюваних груп у клітинках).*
- *2НФ: 1НФ + усі неключові атрибути залежать від усього первинного ключа, а не від його частини.*
- *3НФ: 2НФ + неключові атрибути залежать тільки від ключа, а не один від одного.*
- *БКНФ (нормальна форма Бойса–Кодда): посилює 3НФ, усі функціональні залежності ґрунтуються тільки на ключах.*

13. Що таке фізична модель бази даних? Логічна?

Фізична модель бази даних - це набір бінарних даних у вигляді файлів, структурованих та згрупованих згідно з призначенням, 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 - це структура таблиць, уявлень, індексів та інших логічних елементів бази даних, що дозволяють власне програмування та використання бази даних.

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

Зв’язок між класами таблиці БД та програмними класами:

- *Кожен клас у програмі часто відповідає таблиці БД (одна таблиця – один клас, одна таблиця – кілька класів, один клас – кілька таблиць).*
- *Атрибути класу відповідають полям таблиці.*
- *Об'єкт (екземпляру класу) відповідає запису в рядку таблиці.*
- *Якщо програмні класи є сутностями предметної області, то таблиці відображають їх технічну реалізацію та спосіб зберігання та зв'язку.*

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.