Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

Лабораторна робота № 2

із дисципліни «Технології розроблення програмного забезпечення»

Тема: «Основи проектування»

Варіант - 21

Виконав
Студент групи ІА-31:
Козир Я. О.

Перевірив:
Мягкий М. Ю.

Київ 2025

Зміст

1. **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.

2. **Теоретичні відомості:**

Мова UML є загальноцільовою мовою візуального моделювання, яка розроблена для специфікації, візуалізації, проєктування та документування компонентів програмного забезпечення, бізнес-процесів та інших систем. Мова UML є досить строгим та потужним засобом моделювання, який може бути ефективно використаний для побудови концептуальних, логічних та графічних 18 моделей складних систем різного цільового призначення. Ця мова увібрала в себе найкращі якості та досвід методів програмної інженерії, які з успіхом використовувалися протягом останніх років при моделюванні великих та складних систем.

Діаграма – це графічне уявлення сукупності елементів моделі у формі зв'язкового графа, вершинам і ребрам (дугам) якого приписується певна семантика. 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).

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. Діаграма використання складається з:

- Акторів - будь-які об'єкти, суб'єкти чи системи, що взаємодіють з модельованою бізнес-системою ззовні для досягнення своїх цілей або вирішення певних завдань.
- Варіантів використання - служать для опису служб, які система надає актору.
- Відношень: асоціація, узагальнення, залежність, включення, розширення.

Діаграми класів 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.

Діаграма класів містить у собі класи, їхні методи та атрибути, зв’язки. Методи та атрибути мають 4 модифікатори доступу: public, package, protected, private. Зв’язки налічують у собі асоціацію, агрегацію, композицію, успадкування тощо.

3. 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

4. Хід роботи:

Тема : Online radio station

Діаграма варіантів використання

1	Користувач	Реєстрація	Можливість створити новий обліковий запис (Слухач, Діджей, Адміністратор) із захищеним збереженням пароля.
2	Користувач	Авторизація	Вхід у систему з використанням логіна та пароля, розподіл прав доступу згідно з роллю.
3	Користувач	Перегляд станцій	Перегляд списку доступних радіостанцій на головній сторінці та в особистому кабінеті.
4	Слухач	Прослуховування радіо	Відтворення потокового аудіо (HLS) обраної станції. Ефір синхронізований для всіх слухачів (імітація живого радіо).
5	Слухач	Оцінка треків (Лайк/Дизлайк)	Можливість оцінити поточний трек. Рейтинг впливає на статистику треку.
6	Слухач	Збереження станцій (Обране)	Додавання станцій до особистого списку "Улюблених" для швидкого доступу.
7	Діджей	Завантаження та обробка треків	Завантаження аудіофайлів (MP3), які автоматично конвертуються системою у

			формат HLS із різними бітрейтами (64, 92, 128, 196, 224 kb/s).
8	Діджей	Керування плейлистом	Формування черги відтворення зі своєї бібліотеки. Можливість змінювати порядок треків (сортування) та видаляти їх з ефіру.
9	Діджей	Керування ефіром (Пульт)	Запуск та зупинка прямого ефіру (сесії). Перегляд статусу "Онлайн/Офлайн" та таймера тривалості сесії.
10	Діджей	Модерація ефіру в реальному часі	Можливість "м'якого пропуску" (Skip) треку без його видалення та увімкнення режиму "Випадкове відтворення" (Shuffle) при старті.
11	Адміністратор	Керування користувачами	Блокування користувачів (Бан), зміна ролей, призначення Діджеїв на конкретні радіостанції.
12	Адміністратор	Керування станціями	Створення, редагування та видалення радіостанцій у системі.
13	Адміністратор	Глобальна бібліотека та Статистика	Перегляд усіх завантажених треків у системі, фізичне очищення файлів ("сміття"), перегляд статистики прослуховувань (Visitor).

Рис. 1 - Діаграма використання

Сценарії використання:

Прослуховування радіо	
Передумови	Слухач авторизований у системі, радіостанція активна
Постумови	Слухач слухає обрану радіостанцію.
Взаємодіючі сторони	Слухач, Система онлайн-радіостанції.
Короткий опис	Цей варіант використання визначає процес прослуховування радіостанції.
Основний потік подій	Цей варіант використання запускається, коли слухач хоче прослухати радіо.
	1. Слухач переглядає доступні станції.
	2. Слухач обирає станцію.
	3. Система відтворює ефір станції.
Винятки	Станція недоступна. Система повідомляє про помилку, і слухач повертається до перегляду станцій.
Примітки	-

Керувати чергою відтворення	
Передумови	Діджей авторизований у системі, бібліотека треків заповнена
Постумови	Черга відтворення оновлена для стріму.
Взаємодіючі сторони	Діджей, Система онлайн-радіостанції.
Короткий опис	Цей варіант використання визначає процес управління чергою відтворення треків.
Основний потік подій	Цей варіант використання запускається, коли діджей готує стрім.
	1. Діджей переглядає бібліотеку треків.
	2. Діджей переміщує обрані треки до черги відтворення.
	3. Діджей підтверджує чергу.
Винятки	Бібліотека порожня. Система повідомляє про помилку, і діджей повертається до перегляду.
Примітки	-

Керування радіостанціями	
Передумови	Адміністратор авторизований у системі та має відповідні права.
Постумови	Станцію створено, оновлено або видалено з системи.
Взаємодіючі сторони	Адміністратор, Система.
Короткий опис	Визначає процес керування радіостанціями в системі.
Основний потік подій	1. Адміністратор відкриває панель "Керування користувачами".
	2. Адміністратор натискає "Створити нову станцію".
	3. Система відкриває форму для введення даних (назва станції, опис).
	4. Адміністратор заповнює поля та зберігає зміни.
	5. Система створює новий запис про станцію в базі даних.
	6. Система повертає адміністратора до оновленого списку станцій.
Винятки	Станція з такою назвою вже існує. Система повідомляє про помилку.

	Спроба видалити станцію, на якій зараз йде активний стрім. Система може заблокувати дію та видати попередження.
Примітки	Аналогічні потоки існують для редагування та видалення станцій.

Структура БД:

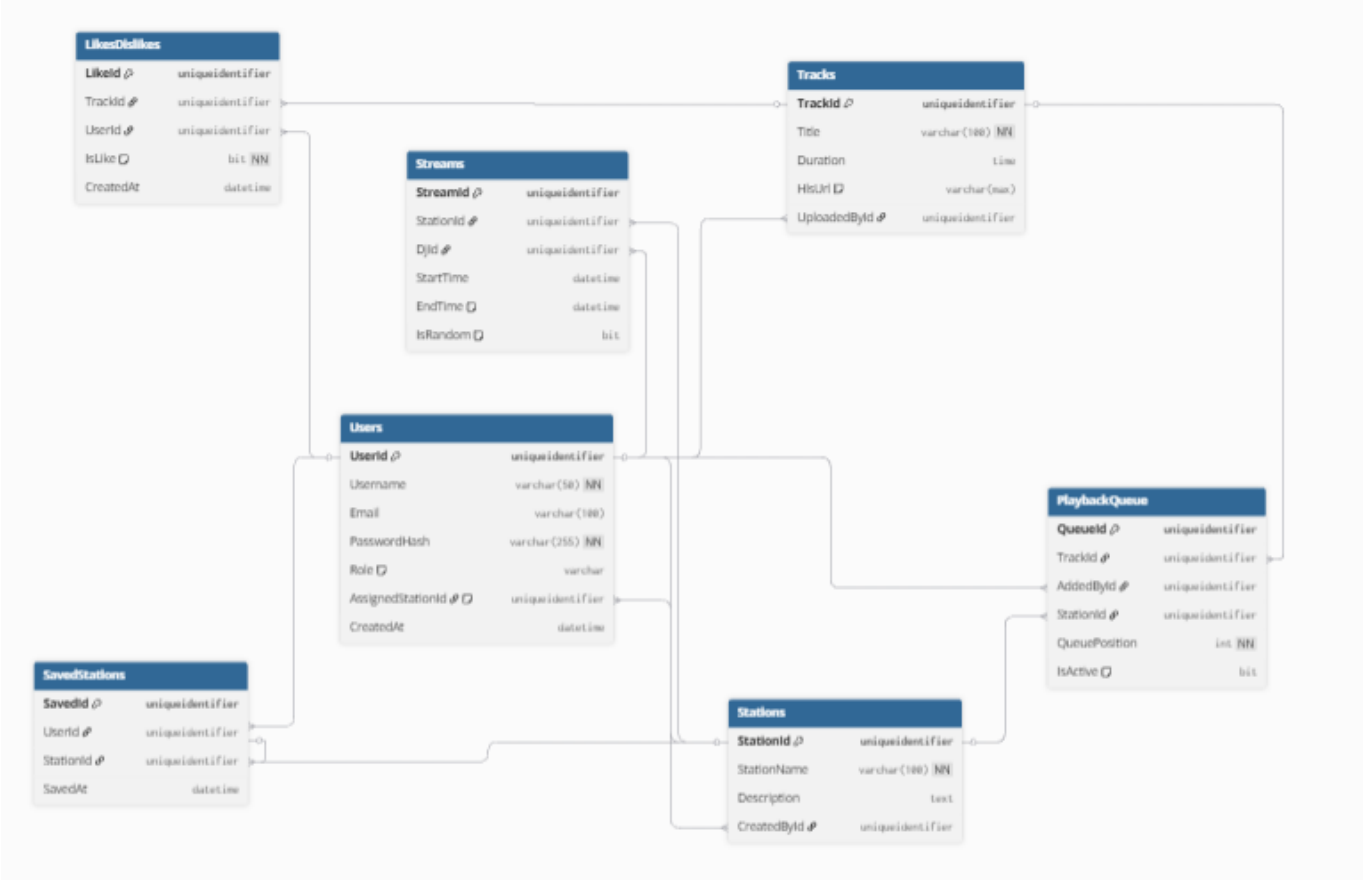

Рис. 2 - Структура БД

1. User (Користувачі) Зберігає облікові дані всіх учасників системи.
   - UserId (PK, Guid): Унікальний ідентифікатор.
   - Username (string): Логін користувача.
   - Email (string): Електронна пошта.
   - PasswordHash (string): Хеш пароля (забезпечує безпеку).
   - Role (string): Роль користувача (Admin, Dj, User, Banned).
   - AssignedStationId (FK, Guid, Nullable): Ідентифікатор станції, до якої прив'язаний Діджей (якщо роль DJ).
   - CreatedAt (DateTime): Дата реєстрації.
2. RadioStation (Радіостанції) Інформація про станції, на яких ведеться мовлення.
   - StationId (PK, Guid): Унікальний ідентифікатор.

- StationName (string): Назва станції.
- Description (string): Опис формату або жанру.
- CreatedById (FK, Guid): Посилання на творця (Адміна).

3. Track (Музичні треки) Глобальна бібліотека аудіофайлів.

- TrackId (PK, Guid): Унікальний ідентифікатор.
- Title (string): Назва композиції.
- Duration (TimeSpan): Тривалість треку.
- UploadedById (FK, Guid): Діджей або Адмін, який завантажив файл.
- HlsUrl (string): Веб-посилання на master.m3u8 файл для потокового відтворення (результат конвертації).

4. PlaybackQueue (Черга відтворення) Визначає, які треки і в якому порядку грають на конкретній станції.

- QueueId (PK, Guid): Унікальний ідентифікатор запису в черзі.
- TrackId (FK, Guid): Посилання на трек.
- StationId (FK, Guid): Посилання на станцію.
- QueuePosition (int): Порядковий номер треку в плейлисті.
- IsActive (bool): Статус треку (true — грає в ефірі, false — тимчасово пропущений/скіпнутий).
- AddedById (FK, Guid): Хто додав трек у чергу.

5. Stream (Сесії мовлення) Журнал активності діджеїв (історія ефірів).

- StreamId (PK, Guid): Унікальний ідентифікатор сесії.
- StationId (FK, Guid): На якій станції йде ефір.
- DjId (FK, Guid): Який діджей веде ефір.
- StartTime (DateTime): Час початку.
- EndTime (DateTime, Nullable): Час завершення (якщо NULL — ефір триває).
- IsRandom (bool): Чи був увімкнений режим випадкового відтворення (Shuffle) під час цієї сесії.

6. SavedStation (Збережені станції) Реалізує функціонал "Улюблене" для слухачів.

- SavedId (PK, Guid): Ідентифікатор.
- UserId (FK, Guid): Слухач.

- StationId (FK, Guid): Обрана станція.
- SavedAt (DateTime): Дата додавання.

7. LikeDislike (Оцінки) Реалізує рейтинг треків.

- LikeId (PK, Guid): Ідентифікатор.
- UserId (FK, Guid): Хто оцінив.
- TrackId (FK, Guid): Який трек оцінено.
- IsLike (bool): Тип оцінки (true — Лайк, false — Дизлайк).
- CreatedAt (DateTime): Час оцінки.

Діаграми класів:

Репозиторії:

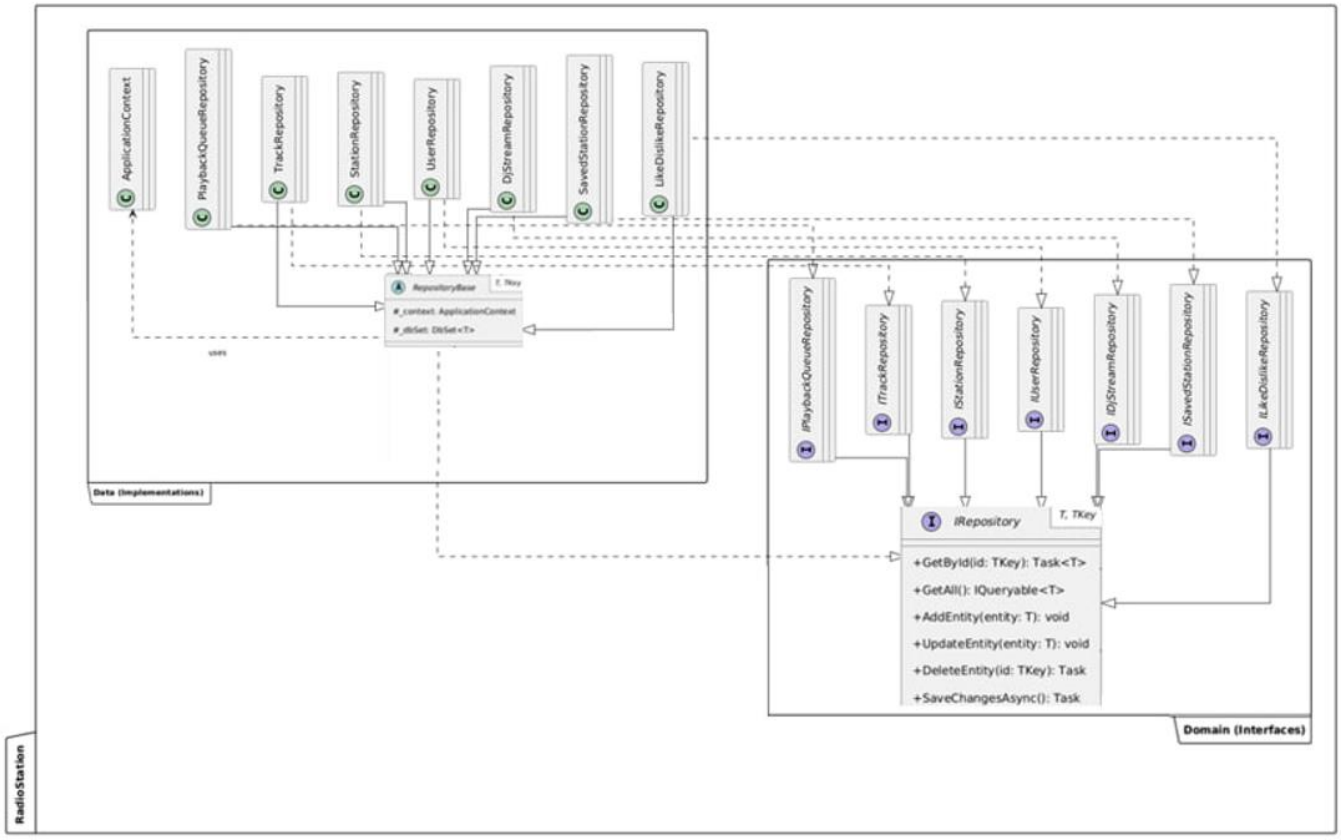

Рис. 3 - Діаграма класів репозиторіїв

На діаграмі класів (Рис. 3) представлена реалізація патерну "Репозиторій" (Repository), який використовується у проєкті для ізоляції логіки доступу до даних від решти бізнес-логіки. Архітектура складається з двох основних частин: узагальнених (generic) компонентів та їхніх специфічних реалізацій.

1. Узагальнені (Generic) компоненти Ці компоненти створюють єдиний "контракт" та спільну реалізацію для всіх репозиторіїв, що дозволяє уникнути дублювання коду.

Інтерфейс IRepository<T, TKey> Призначення: Універсальний "контракт", який описує базовий набір CRUD-операцій (Create, Read, Update, Delete), що має виконувати будь-який репозиторій. Параметри:

- T: Тип сутності (наприклад, User або RadioStationEntity).
- TKey: Тип первинного ключа сутності (наприклад, Guid або int).

Методи:

- GetById(TKey): Повертає задачу (Task), результатом якої є об'єкт типу T за його ключем.
- GetAll(): Повертає запит IQueryable<T> для отримання всіх записів.
- AddEntity(T): Додає новий об'єкт T до контексту (void).
- UpdateEntity(T): Позначає існуючий об'єкт T як змінений (void).
- DeleteEntity(TKey): Асинхронно видаляє об'єкт за його ключем (Task).
- SaveChangesAsync(): Асинхронно зберігає всі зміни в базі даних (Task).

Абстрактний клас RepositoryBase<T, TKey> Призначення: Спільна реалізація інтерфейсу IRepository. Усі конкретні репозиторії успадковують цей клас, щоб автоматично отримати всю базову логіку. Атрибути:

- _context: ApplicationContext: Захищене поле, що зберігає посилання на контекст бази даних (DbContext з Entity Framework).
- _dbSet: DbSet<T>: Захищене поле, що зберігає посилання на конкретну таблицю (DbSet) у базі даних, яка відповідає сутності T. Зв'язки: Реалізує інтерфейс IRepository<T, TKey>.

2. Специфічні реалізації для сутностей Для кожної сутності (наприклад, User, Track, DjStream) створюється власний інтерфейс та клас-реалізація. Це дозволяє за потреби додавати унікальні методи (наприклад, GetUserByUsernameAsync), водночас успадковуючи всю базову логіку.

Сутність	**Специфічний Інтерфейс**	**Клас-Реалізація**
User	IUserRepository	UserRepository
Station	IStationRepository	StationRepository

LikeDislike	ILikeDislikeRepository	LikeDislikeRepository
DjStream	IDjStreamRepository	DjStreamRepository
SavedStation	ISavedStationRepository	SavedStationRepository
Track	ITrackRepository	TrackRepository
PlaybackQueue	IPlaybackQueueRepository	PlaybackQueueRepository

Класи даних:

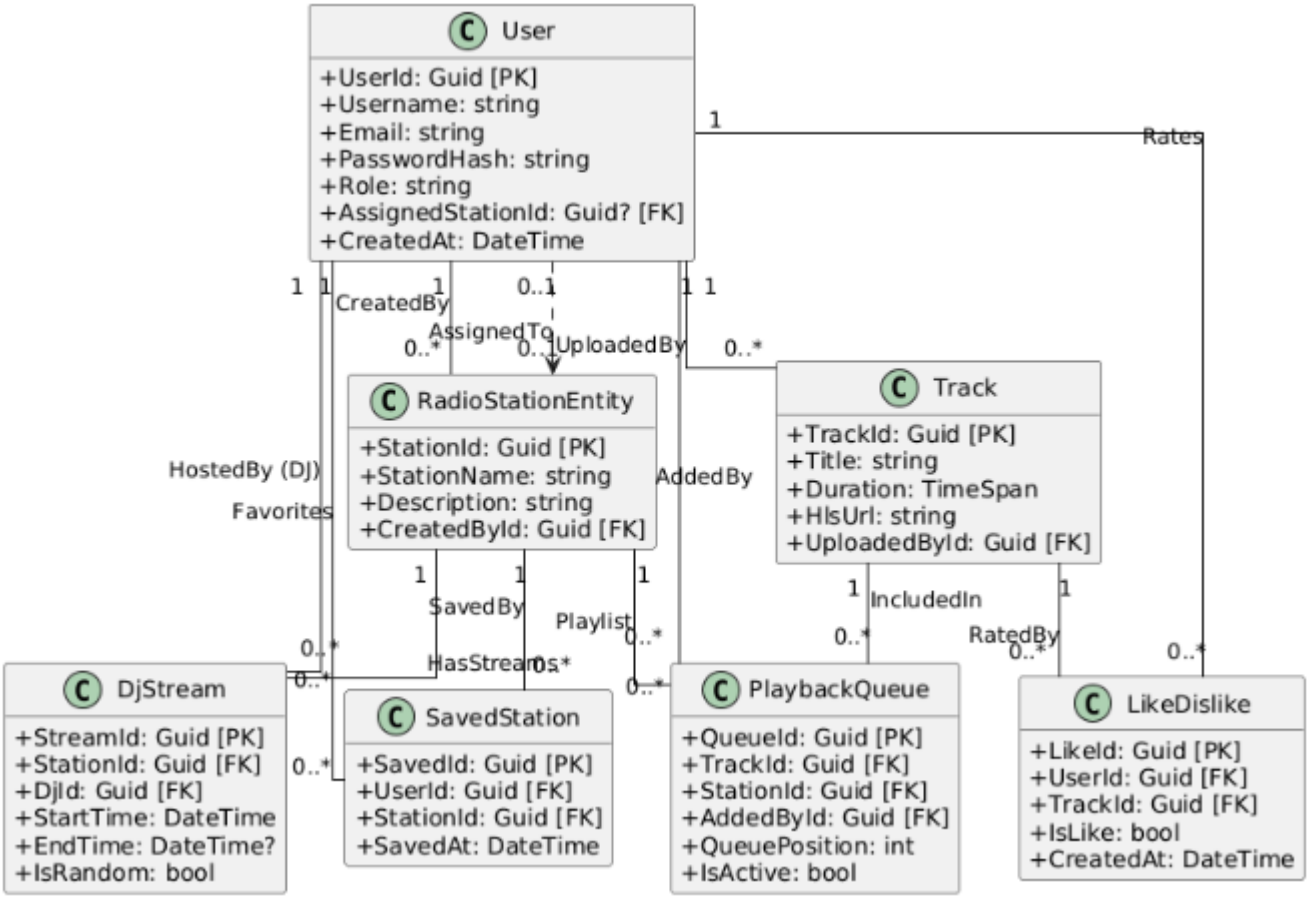

Рис. 4 - Діаграма класів даних

Концептуальна модель системи представлена на діаграмі класів (Рис. 4), яка описує ключові сутності (класи), їхні атрибути та зв’язки між ними.

Клас (Сутність)	**Атрибути (Властивості)**	**Короткий опис**
User	userId, username, passwordHash, role, createdAt, assignedStationId	Зберігає облікові дані, роль та прив'язку Діджея до конкретної станції.

RadioStation	stationId, stationName, description, createdBy	Описує радіостанцію в системі.
Track	trackId, title, duration, hlsUrl, uploadedBy	Медіа-файл. Містить HlsUrl — шлях до потокового маніфесту .m3u8.
DjStream	streamId, stationId, djId, startTime, endTime, isRandom	Сесія ефіру. IsRandom визначає, чи увімкнено режим Shuffle.
PlaybackQueue	queueId, trackId, stationId, isActive, queuePosition	Елемент черги. IsActive дозволяє тимчасово пропускати трек без видалення.
SavedStation	savedId, userId, stationId, savedAt	Зв'язок для списку "Улюблені станції".
LikeDislike	likeId, userId, trackId, isLike, createdAt	Рейтинг треків (Лайк/Дизлайк).

2. Зв’язки між класами (Multiplicity)

Зв'язки (асоціації) на діаграмі показують, як сутності взаємодіють між собою:

- User (1) до RadioStation (0..*): Один користувач (DJ/Admin) може створити багато радіостанцій.
- User (1) до Track (0..*): Один користувач (DJ) може завантажити багато треків.
- User (1) до Stream (0..*): Один користувач (DJ) може провести багато стрімів (сесій ефіру).
- User (1) до PlaybackQueue (0..*): Один користувач (DJ) може додати багато треків до черги.
- User (1) до SavedStation (0..*): Один користувач (Слухач) може зберегти багато станцій ("Обране").
- User (1) до LikeDislike (0..*): Один користувач (Слухач) може поставити багато лайків/дизлайків.
- RadioStation (1) до Stream (0..*): Одна радіостанція може мати багато стрімів (історію трансляцій).
- RadioStation (1) до PlaybackQueue (0..*): Одна радіостанція може мати багато треків у своїй черзі відтворення.
- RadioStation (1) до SavedStation (0..*): Одну радіостанцію можуть зберегти багато користувачів.

- Track (1) до PlaybackQueue (0..*): Один і той самий трек може бути доданий до черги відтворення багато разів (або на різних станціях).
- Track (1) до LikeDislike (0..*): Один трек може мати багато лайків/дизлайків.

Вихідний код класів:

```
namespace RadioStation.Data
{
    public interface IRepository<T, TKey> where T : class
    {
        Task<T> GetById(TKey id);
        IQueryable<T> GetAll();
        void AddEntity(T entity);
        void UpdateEntity(T entity);
        void DeleteEntity(TKey id);
        Task SaveChangesAsync();
    }
}
```

Рис. 5 - Код інтерфейсу IRepository<T, TKey>

```
namespace RadioStation.Data
{
    public abstract class RepositoryBase<T, TKey> : IRepository<T, TKey> where T : class
    {
        protected ApplicationContext _context;
        protected DbSet<T> _dbSet;
        public RepositoryBase(ApplicationContext context)
        {
            _context = context;
            _dbSet = context.Set<T>();
        }
        public Task<T> GetById(TKey id)
        {
            return _dbSet.FindAsync(id).AsTask();
        }
        public IQueryable<T> GetAll()
        {
            return _dbSet.AsQueryable();
        }
        public void AddEntity(T entity)
        {
            _dbSet.Add(entity);
        }
        public void UpdateEntity(T entity)
        {
            _dbSet.Update(entity);
        }
        public void DeleteEntity(TKey id)
        {
            var entity = _dbSet.Find(id);
            if (entity != null)
            {
                _dbSet.Remove(entity);
            }
        }
        public async Task SaveChangesAsync()
        {
```

Рис. 6 - Код класу RepositoryBase<T, TKey>

```
namespace RadioStation.Data
{
    public class User
    {
        [Key]
        public Guid UserId { get; set; }
        public string Username { get; set; }
        public string PasswordHash { get; set; }
        public string Role { get; set; }
        public DateTime CreatedAt { get; set; }

        public ICollection<RadioStation> CreatedStations { get; set; }
        public ICollection<Track> UploadedTracks { get; set; }
        public ICollection<PlaybackQueue> AddedQueues { get; set; }
        public ICollection<SavedStation> SavedStations { get; set; }
        public ICollection<Review> Reviews { get; set; }
        public ICollection<LikeDislike> LikesDislikes { get; set; }
        public ICollection<Stream> DjStreams { get; set; }
    }
}
```

Рис. 7 - Код класу User

```
namespace RadioStation.Data
{
    public class RadioStation
    {
        [Key]
        public Guid StationId { get; set; }
        public string StationName { get; set; }
        public string Description { get; set; }
        public Guid CreatedById { get; set; }

        public User CreatedBy { get; set; }
        public ICollection<PlaybackQueue> Playbacks { get; set; }
        public ICollection<SavedStation> SavedByUsers { get; set; }
        public ICollection<Stream> Streams { get; set; }
    }
}
```

Рис. 8 - Код класу RadioStation

```
namespace RadioStation.Data
{
    public class Track
    {
        [Key]
        public Guid TrackId { get; set; }
        public string Title { get; set; }
        public TimeSpan Duration { get; set; }
        public Guid UploadedById { get; set; }

        public User UploadedBy { get; set; }
        public ICollection<PlaybackQueue> Queues { get; set; }
        public ICollection<Review> Reviews { get; set; }
        public ICollection<LikeDislike> LikesDislikes { get; set; }
    }
}
```

Рис. 9 - Код класу Track

```
namespace RadioStation.Data
{
    public class PlaybackQueue
    {
        [Key]
        public Guid QueueId { get; set; }
        public Guid TrackId { get; set; }
        public Guid StationId { get; set; }
        public Guid AddedById { get; set; }
        public int QueuePosition { get; set; }

        public Track Track { get; set; }
        public RadioStation Station { get; set; }
        public User AddedBy { get; set; }
    }
}
```

Рис. 10 - Код класу PlaybackQueue

```
namespace RadioStation.Data
{
    public class SavedStation
    {
        [Key]
        public Guid SavedId { get; set; }
        public Guid UserId { get; set; }
        public Guid StationId { get; set; }
        public DateTime SavedAt { get; set; }

        public User User { get; set; }
        public RadioStation Station { get; set; }
    }
}
```

Рис. 11 - Код класу SavedStation

```
namespace RadioStation.Data
{
    public class LikeDislike
    {
        [Key]
        public Guid LikeId { get; set; }
        public Guid UserId { get; set; }
        public Guid TrackId { get; set; }
        public bool IsLike { get; set; }
        public DateTime CreatedAt { get; set; }

        public User User { get; set; }
        public Track Track { get; set; }
    }
}
```

Рис. 12 - Код класу LikeDislike

```
namespace RadioStation.Data
{
    public class Stream
    {
        [Key]
        public Guid StreamId { get; set; }
        public Guid StationId { get; set; }
        public Guid DjId { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime? EndTime { get; set; }

        public RadioStation Station { get; set; }
        public User Dj { get; set; }
    }
}
```

Рис. 13 - Код класу Stream

5. Питання до лабораторної роботи:

1. Що таке UML? – універсальна мова для опису та візуалізації систем.
2. Що таке діаграма класів UML? – схема з класами, їхніми полями, методами та зв’язками.
3. Які діаграми UML називають канонічними? – стандартні: класів, об’єктів, варіантів використання, послідовностей тощо.
4. Що таке діаграма варіантів використання? – показує акторів і функції, які вони виконують у системі.
5. Що таке варіант використання? – дія або функція, яку виконує користувач.
6. Які відношення можуть бути відображені на діаграмі використання? – асоціація, include, extend, узагальнення.
7. Що таке сценарій? – опис кроків взаємодії користувача з системою.
8. Що таке діаграма класів? – схема структури системи через класи та їх зв’язки.
9. Які зв’язки між класами ви знаєте? – асоціація, агрегація, композиція, наслідування, залежність.
10. Чим відрізняється композиція від агрегації? – у композиції частини не існують без цілого, в агрегації можуть.
11. Чим відрізняється зв'язки типу агрегації від зв'язків композиції на діаграмах класів? – агрегація має порожній ромб, композиція – зафарбований.
12. Що являють собою нормальні форми баз даних? – правила організації таблиць для уникнення дублювання.
13. Що таке фізична модель бази даних? Логічна? – фізична: як реально зберігається; логічна: концептуальна схема.

14. Який взаємозв'язок між таблицями БД та програмними класами? – таблиця ≈ клас, рядок ≈ об’єкт, стовпчик ≈ поле.

6. Висновок

У процесі виконання лабораторної роботи з теми "Online radio station" було здійснено комплексний аналіз вимог, розроблено моделі системи за допомогою UML, включаючи діаграми варіантів використання, класів та структури даних, а також створено базову архітектуру коду. Використано сучасні підходи до проектування, такі як патерни Repository та асинхронне програмування, для забезпечення гнучкості та масштабовності. Результатом є теоретична основа для реалізації функціональної системи, що відповідає поставленим завданням і може бути розширена в майбутньому.