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 №3

із дисципліни «*ТРПЗ*»

Тема: «ДІАГРАМА РОЗГОРТАННЯ. ДІАГРАМА КОМПОНЕНТІВ. ДІАГРАМА ВЗАЄМОДІЙ ТА ПОСЛІДОВНОСТЕЙ**»**

Виконав:
Студент групи ІА-24
Бакалець А.І.

Перевірив:
Мягкий М.Ю.

Київ-2024

Мета: Навчитися розробляти діаграму розгортання, діаграму компонентів, діаграму взаємодій та послідовностей

Теоретичні відомості

Діаграма розгортання (Deployment Diagram)

Діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонентів.

Діаграма компонетів

Діаграма компонентів (англ. Component diagram) — в UML, діаграма, на якій відображаються компоненти, залежності та зв'язки між ними. Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись. Модуль програмного забезпечення може бути представлено як компоненту. Деякі компоненти існують під час компіляції, деякі — під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Діаграма послідовностей

Діаграма послідовності (англ. sequence diagram) — різновид діаграми в UML. Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність надісланих повідомлень.

Завдання.

1. Ознайомитися з короткими теоретичними відомостями.
2. Розробити діаграму розгортання для проектованої системи.
3. Розробити діаграму компонентів для проектованої системи.
4. Розробити діаграму послідовностей для проектованої системи.
5. Скласти звіт про виконану роботу.

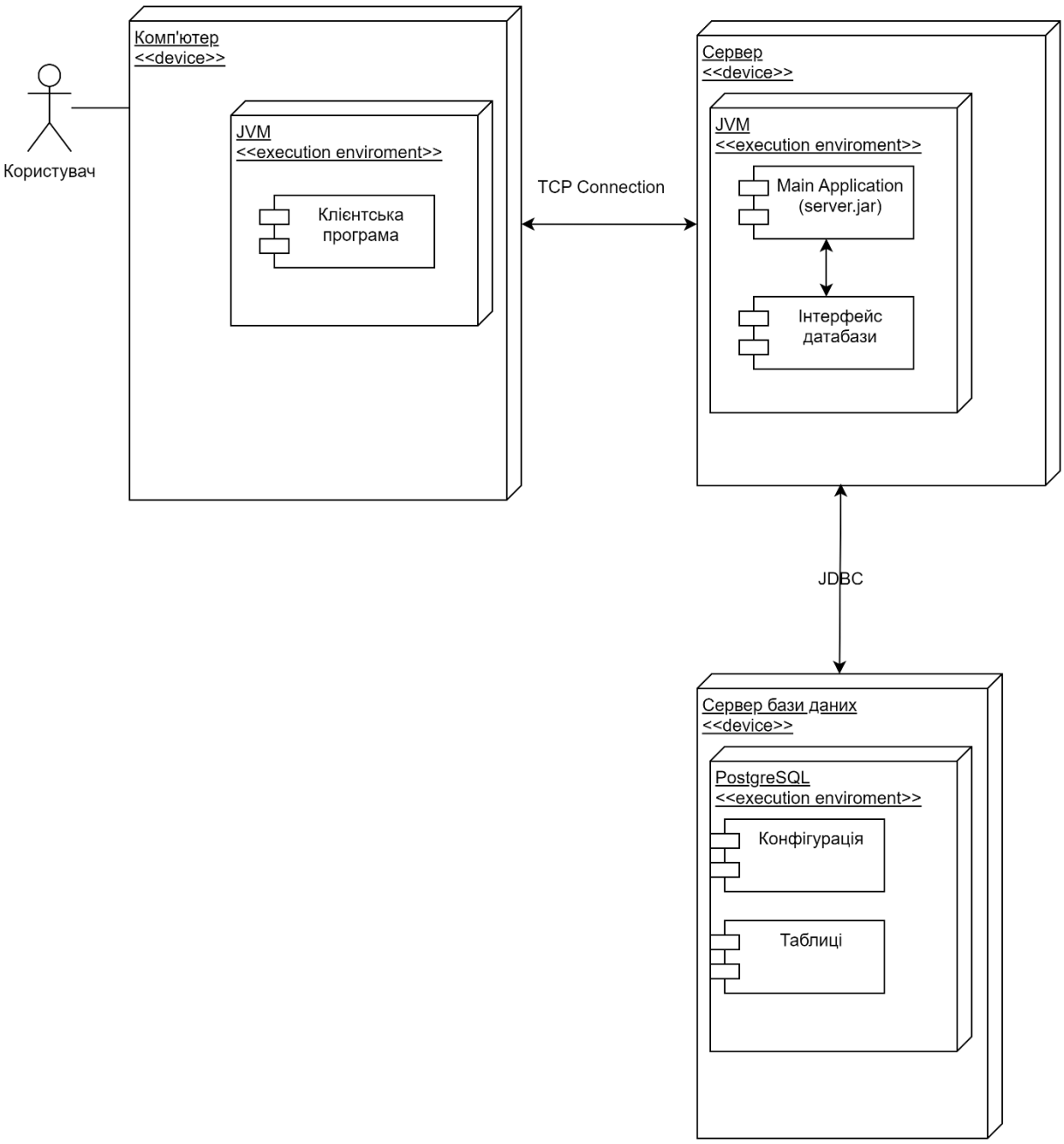
Тема:

..1 Музичний програвач (iterator, command, memento, facade, visitor, client-server)

Музичний програвач становить собою програму для програвання музичних файлів або відтворення потокової музики з можливістю створення, запам'ятовування і редагування списків програвання, перемішування/повторення (shuffle/repeat), розпізнавання різних аудіо-форматів, еквалайзер.

Github: https://github.com/Clasher3000/TRPZ_labs

Діаграма розгортання:



Структура:

1. **Клієнт(Комп’ютер користувача):**

   Середовище виконання:

   JVM – Середовище, яке виконує програму написану на мові програмування Java, використовуючи JDK.

   Компоненти:

   Клієнтська програма – програма, що запускається на клієнті, яка ініціює запити до сервера через сокет

2. **Сервер:**

   Середовище виконання:

   JVM – Середовище, яке виконує програму написану на мові програмування Java, використовуючи JDK.

   Компоненти:

   - Main Application – головна програма на сервері, яка відповідає за бізнес-логіку. Вона приймає запити від клієнта, обробляє їх і відправляє відповіді.
   - Інтерфейс датабази – шар між додатком і базою даних, який забезпечує просту взаємодію між ними, використовуючи Hibernate ORM

3. **Сервер бази даних:**

Середовище виконання:

Postgresql – система керування базами даних (СУБД), в якій зберігаються таблиці та дані.

Компоненти:

- Конфігурація – налаштування для коректної роботи СУБД.
- Таблиці – структури даних, які містять самі дані, що використовуються додатком.

Взаємодії:

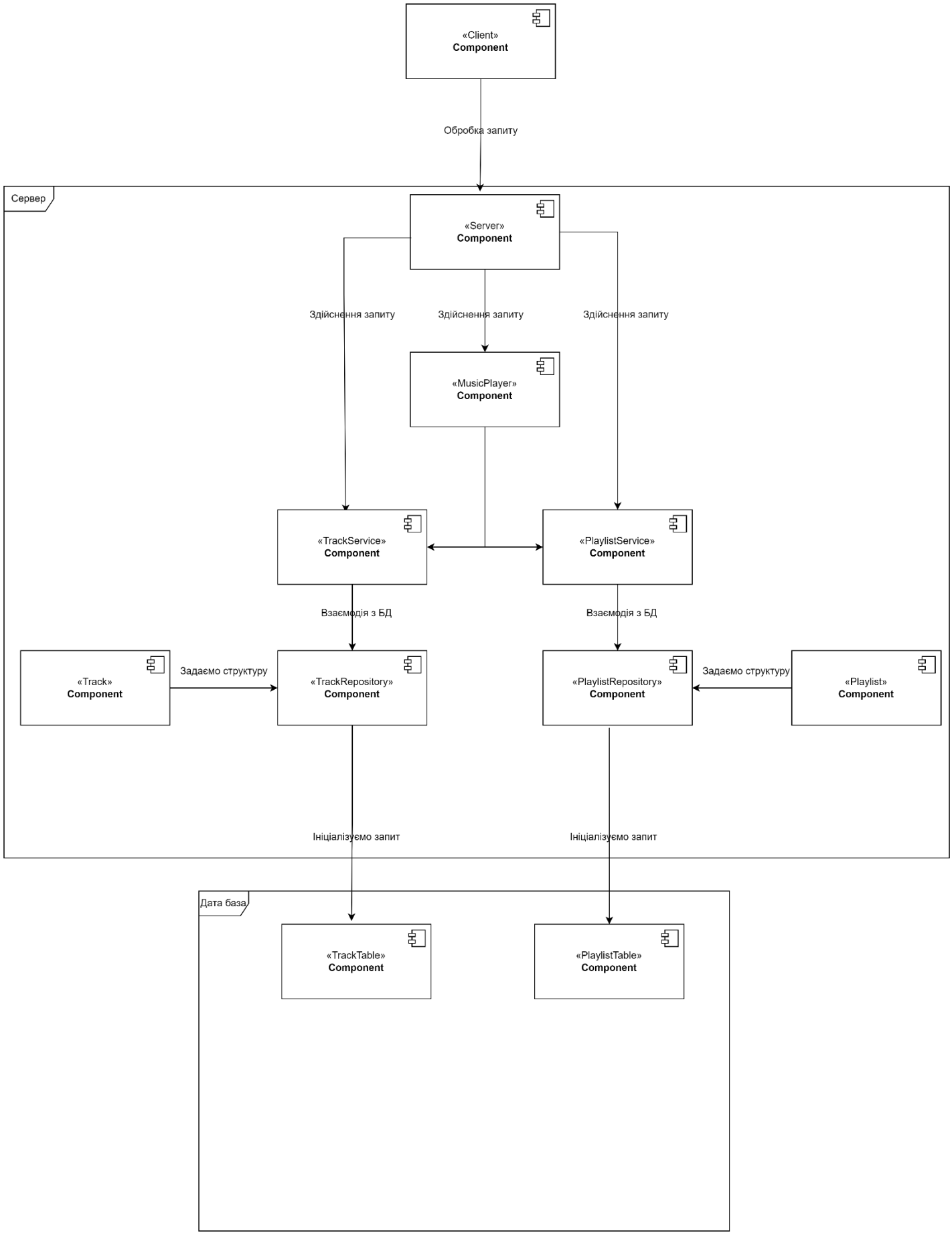
1. **Клієнт (Комп'ютер користувача) – Сервер:**

- **Протокол взаємодії:** TCP (Transmission Control Protocol) – використовується для встановлення надійного з'єднання між клієнтом і сервером через сокети.
- **Механізм передачі даних:** клієнтська програма ініціює сокет-з'єднання із сервером. Після встановлення TCP-з'єднання, клієнт відправляє команду або запит через сокет.
- **Ініціалізація з'єднання:** клієнт створює TCP-сокет і з'єднується із сервером, вказавши IP-адресу та порт сервера.
- **Передача даних:** клієнт надсилає запити до сервера через сокет. Запити є різними командами, необхідними користувачу.
- **Отримання відповіді:** сервер обробляє запит, виконує необхідні дії(наприклад, звертається до бази даних) і відправляє результат клієнту через сокет.
- **Закриття з'єднання:** після завершення обміну даними, клієнт закриває сокет, закінчуючи взаємодію.

2. **Сервер – Сервер бази даних:**

- **Протокол взаємодії:** JDBC (Java Database Connectivity)
- **Механізм взаємодії:** основна програма на сервері взаємодіє з базою даних через інтерфейс бази даних. Коли клієнт надсилає запит, який потребує даних із бази, сервер звертається до бази даних через JDBC, отримує необхідні дані й обробляє їх. При необхідності ми вносимо зміни в таблиці.

Діаграма компонентів



Структура:

- Client - компонент, що запускається на клієнті, яка ініціює запити до сервера через сокет. Також цей компонент відповідає за отримання відповідей від сервера та відображення їх користувачу.

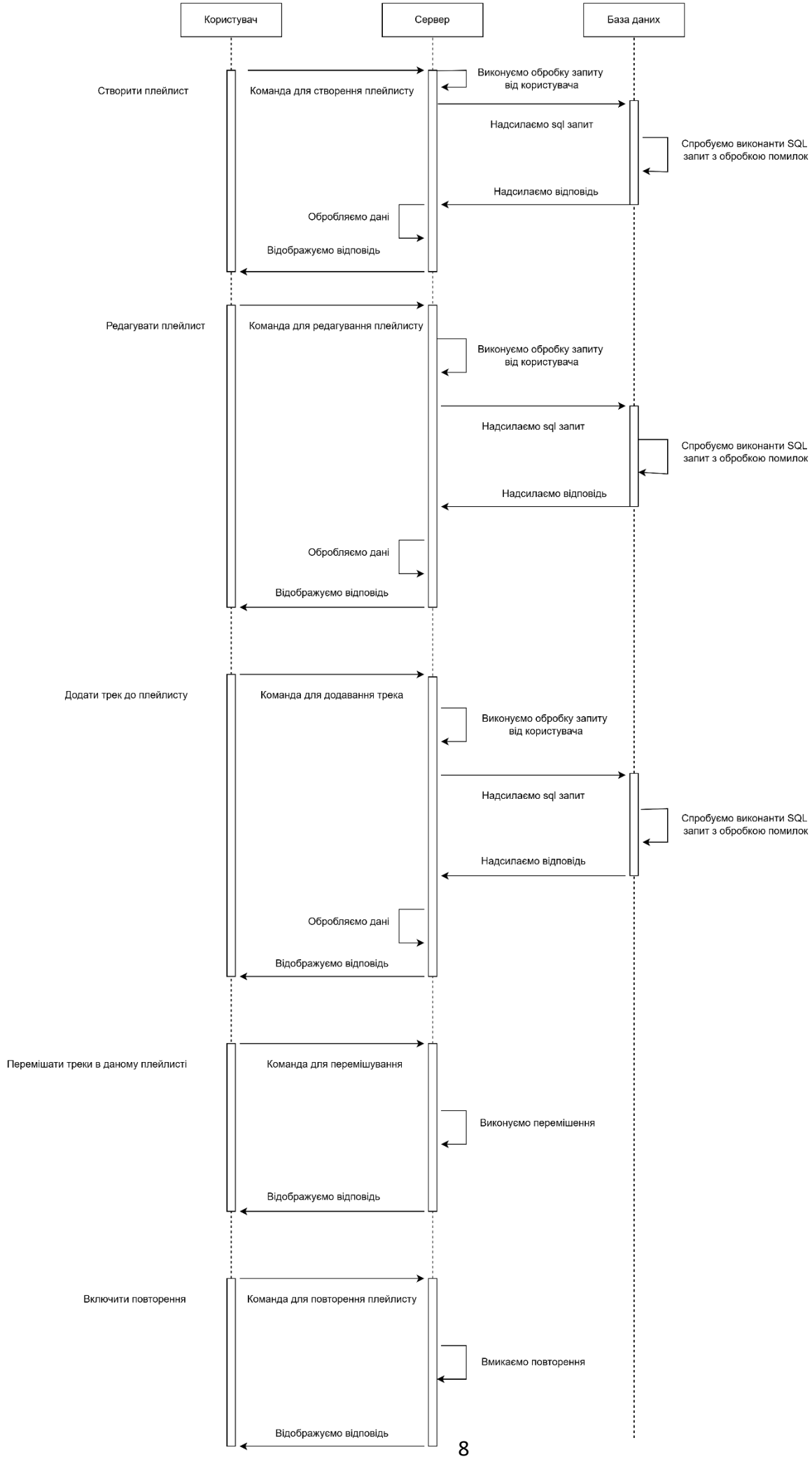
Сервер:

- Server Component – Основний компонент на стороні сервера, який приймає запити від клієнта, обробляє їх і передає відповідні команди іншим компонентам (наприклад, MusicPlayer, PlaylistService і TrackService), також він керує сокетами.
- MusicPlayer Component – Цей компонент реалізує логіку відтворення музики. Він отримує команди від інших компонентів сервера, коли потрібно відтворити музичний трек або плейлист.
- PlaylistService Component– Відповідає за бізнес-логіку, пов'язану з плейлистами. Обробляє запити, які надходять до сервера на взаємодію зі списками відтворення.
- TrackService Component - Відповідає за бізнес-логіку, пов'язану з окремими треками. Виконує запити щодо додавання, видалення або отримання інформації про треки.
- TrackRepository Component – Інтерфейс для взаємодії з базою даних щодо треків. Забезпечує операції з базою даних для компонентів, що працюють із треками. Використовує Hibernate ORM для взаємодії БД.
- PlaylistRepository Component – Інтерфейс для взаємодії з базою даних щодо плейлистів. Використовується для збереження або отримання інформації про списки відтворення. Використовує Hibernate ORM для взаємодії з БД.
- Track Component, Playlist Component – Класи, які представляють треки і плейлисти як об'єкти. Вони використовуються в TrackRepository і PlaylistRepository для збереження або отримання даних у структурованій формі.

Дата база:

- TrackTable, PlaylistTable – Таблиці в базі даних для зберігання треків і списків відтворення відповідно. Взаємодія з ними відбувається через відповідні репозиторії (TrackRepository і PlaylistRepository).

Діаграма послідовностей:


Користувач
Сервер
База даних
Створити плейлист
Команда для створення плейлисту
Виконуємо обробку запиту від користувача
Надсилаємо sql запит
Спробуємо виконанти SQL запит з обробкою помилок
Надсилаємо відповідь
Обробляємо дані
Відображуємо відповідь
Редагувати плейлист
Команда для редагування плейлисту
Виконуємо обробку запиту від користувача
Надсилаємо sql запит
Спробуємо виконанти SQL запит з обробкою помилок
Надсилаємо відповідь
Обробляємо дані
Відображуємо відповідь
Додати трек до плейлисту
Команда для додавання трека
Виконуємо обробку запиту від користувача
Надсилаємо sql запит
Спробуємо виконанти SQL запит з обробкою помилок
Надсилаємо відповідь
Обробляємо дані
Відображуємо відповідь
Перемішати треки в даному плейлисті
Команда для перемішування
Виконуємо перемішення
Відображуємо відповідь
Включити повторення
Команда для повторення плейлисту
Вмикаємо повторення
Відображуємо відповідь

Можливі дії:

1. **Створити плейлист:**

1) Користувач вводить команду для створення плейлиста на клієнті.
2) Сервер отримує запит і формує SQL-запит для створення нового плейлиста.
3) Сервер надсилає SQL-запит до бази даних.
4) База даних додає новий запис у таблицю *PlaylistTable*, або при виникненні помилки ми отримуємо її код та читання/редагування бази даних не відбувається.
5) База даних надсилає відповідь серверу (успішно/помилка).
6) Сервер обробляє відповідь.
7) Клієнт отримує і відображає результат (створений плейлист або помилку).

2. **Редагувати плейлист:**

1) Користувач вводить команду для редагування треку плейлиста на клієнті.
2) Сервер отримує запит і формує SQL-запит для редагування плейлиста.
3) Сервер надсилає SQL-запит до бази даних.
8) База даних редагує запис у таблицю плейлистів (*PlaylistTable*) , або при виникненні помилки ми отримуємо її код та читання/редагування бази даних не відбувається.
4) База даних надсилає відповідь серверу (успішно/помилка).
5) Сервер обробляє відповідь.
6) Клієнт відображає результат (плейлист оновлено або помилка).

3. **Додати трек до плейлиста**:

1) Користувач вводить команду для додавання треку до певного плейлиста на клієнті.
2) Сервер отримує запит і формує SQL-запит для додавання треку до плейлиста.
3) Сервер надсилає SQL-запит до бази даних.
9) База даних додає новий запис у таблицю треків (*TrackTable*) і пов'язує його з плейлистом, або при виникненні помилки ми отримуємо її код та читання/редагування бази даних не відбувається.
4) База даних надсилає відповідь серверу (успішно/помилка).
5) Сервер обробляє відповідь і надсилає її клієнту.
6) Клієнт відображає результат (трек додано або помилка).

4. **Перемішати треки в даному плейлисті:**

1) Користувач вводить команду для перемішування треків на клієнті.
2) Сервер виконує перемішування треків в заданому плейлисті за допомогою алгоритму та отримує результат(успішно/помилка).
3) Клієнт відображає результат (плейлист оновлено або помилка).

5. Включити повторення:

1) Користувач вводить команду для повторення треків на клієнті.
2) Сервер виконує алгоритм повторення треків в заданому плейлисті та отримує результат(успішно/помилка).
3) Клієнт відображає результат (плейлист оновлено або помилка).

P.S. Виправлено орфографічні помилки, додано більш зрозумілі написи до таблиці послідовностей та додано сценарій не коректного sql запиту. Також підписи до повідомлень та самі повідомлення трішки переставлені для кращої читабельності.

Висновок: виконуючи дану лабораторну роботу, я опрацював діаграму розгортання, діаграму компонентів, діаграму послідовностей. Також мною було додано сервер та клієнт, за допомогою яких, мною була розроблена клієнт-серверна архітектура.