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

ЛАБОРАТОРНА РОБОТА №3

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Основи проектування розгортання»

Виконав:

студент групи ІА-33

Грицай Андрій

Перевірив:

асистент кафедри ІСТ

Мягкий Михайло Юрійович

Київ - 2025

Тема: Основи проектування розгортання.
Мета: Навчитися проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.
Тема лабораторного циклу:

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.

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 компонентів (Component diagram)

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

Діаграма послідовностей (Sequence diagram)

Діаграма послідовностей Діаграма послідовності – різновид діаграми в UML. Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність надісланих повідомлень.

Хід роботи
Діаграма розгортання

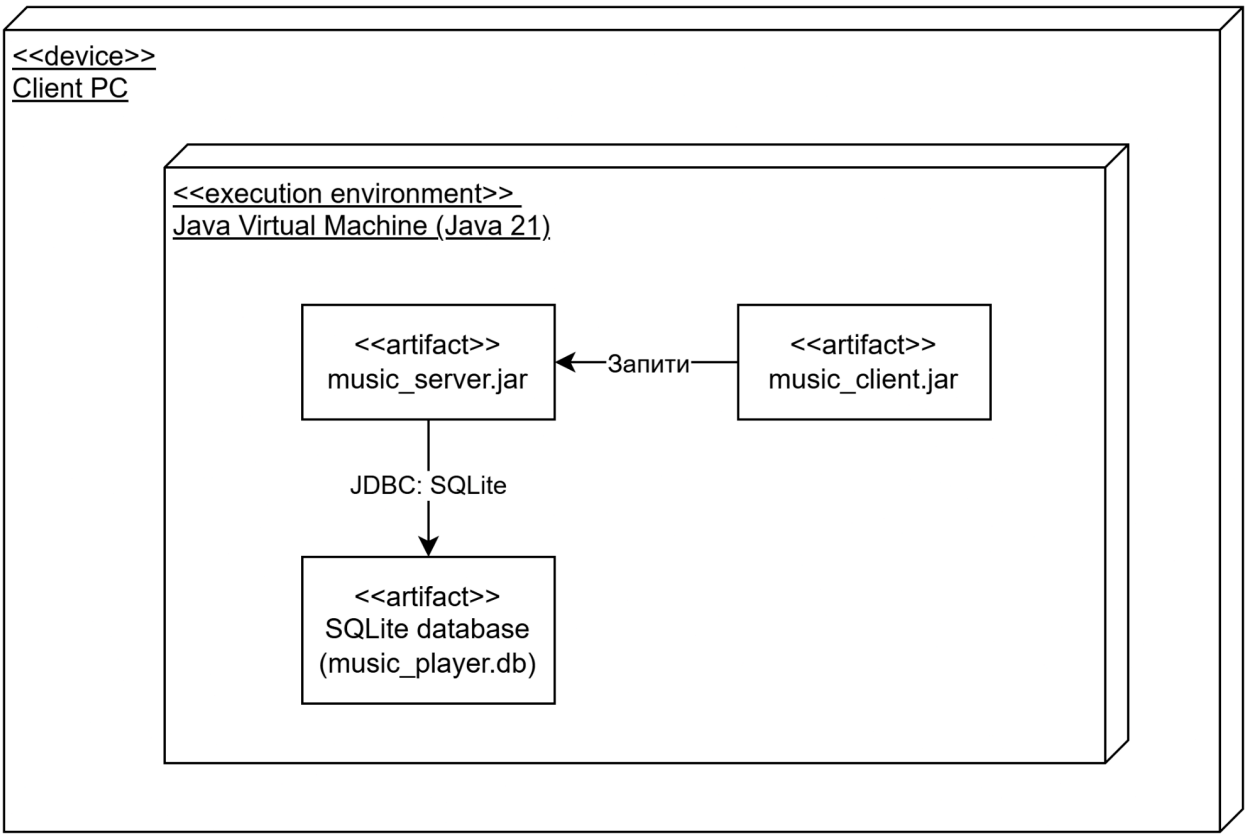

Рисунок 1. - Діаграма розгортання застосунку

Ця діаграма розгортання описує фізичне розміщення компонентів системи музичного програвача в середовищі виконання на комп'ютері користувача.

Client PC (ПК Користувача): Вузол <<device>> Client PC представляє фізичний пристрій, на якому розгорнуто систему. Всередині вузла функціонує середовище виконання <<execution environment>> Java Virtual Machine (Java 21), яке забезпечує платформу для запуску Java-компонентів.

Артефакти системи:

- music_client.jar: Артефакт, що представляє клієнтську частину програми. Він відповідає за надання графічного інтерфейсу користувача та надсилання запитів до серверного компонента.
- music_server.jar: Серверний артефакт, який містить бізнес-логіку програми. Він приймає та обробляє запити від клієнта, а також керує операціями з даними.
- SQLite database (music_player.db): Артефакт бази даних, що являє собою локальний файл SQLite. Він відповідає за фізичне зберігання даних системи (інформація про треки, плейлисти, користувачів).

Протоколи та зв'язки:

- Запити: Позначає логічний канал зв'язку для передачі команд та даних між music_client.jar та music_server.jar в межах віртуальної машини.
- JDBC: SQLite: Протокол Java Database Connectivity, який використовується компонентом music_server.jar для підключення до файлу бази даних music_player.db та виконання SQL-запитів.

Діаграма компонентів

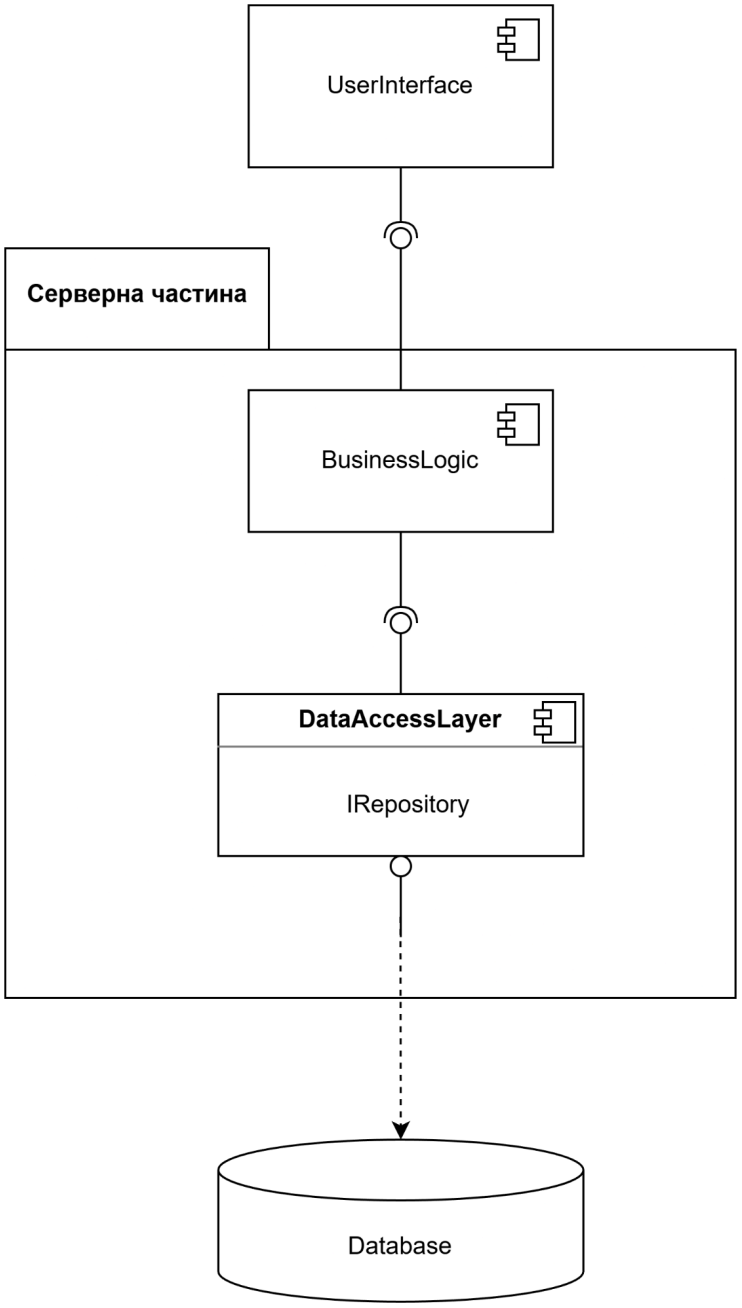

Рисунок 2. - Діаграма компонентів застосунку

На поданій діаграмі компонентів зображено архітектуру системи «Музичний програвач», побудовану за клієнт-серверною моделлю.

Перший рівень представлений компонентом UserInterface. Це клієнтська частина, через яку користувач взаємодіє із системою: керує відтворенням, переглядає бібліотеку та редагує плейлисти. Цей компонент надсилає запити до серверної частини для обробки даних.

Другий рівень — це «Серверна частина», яка містить два ключові компоненти:

- BusinessLogic: Компонент, що містить основну бізнес-логіку системи (керування сесіями, логіка відтворення, обробка сутностей). Він взаємодіє з рівнем даних через визначений інтерфейс.
- DataAccessLayer: Відповідає за безпосередній доступ до даних. Він надає інтерфейс IRepository, який абстрагує бізнес-логіку від деталей реалізації запитів до сховища даних.

Третій рівень — це Database. Цей компонент забезпечує фізичне зберігання всієї інформації системи: даних користувачів, метаданих треків, списків відтворення та налаштувань еквалайзера. Компонент DataAccessLayer звертається до бази даних для виконання операцій читання та запису.

Діаграма демонструє чіткий поділ відповідальності між рівнями презентації, бізнес-логіки та даних. Використання інтерфейсу IRepository забезпечує низьку зв'язність системи, що спрощує її підтримку та тестування.

Діаграма послідовностей

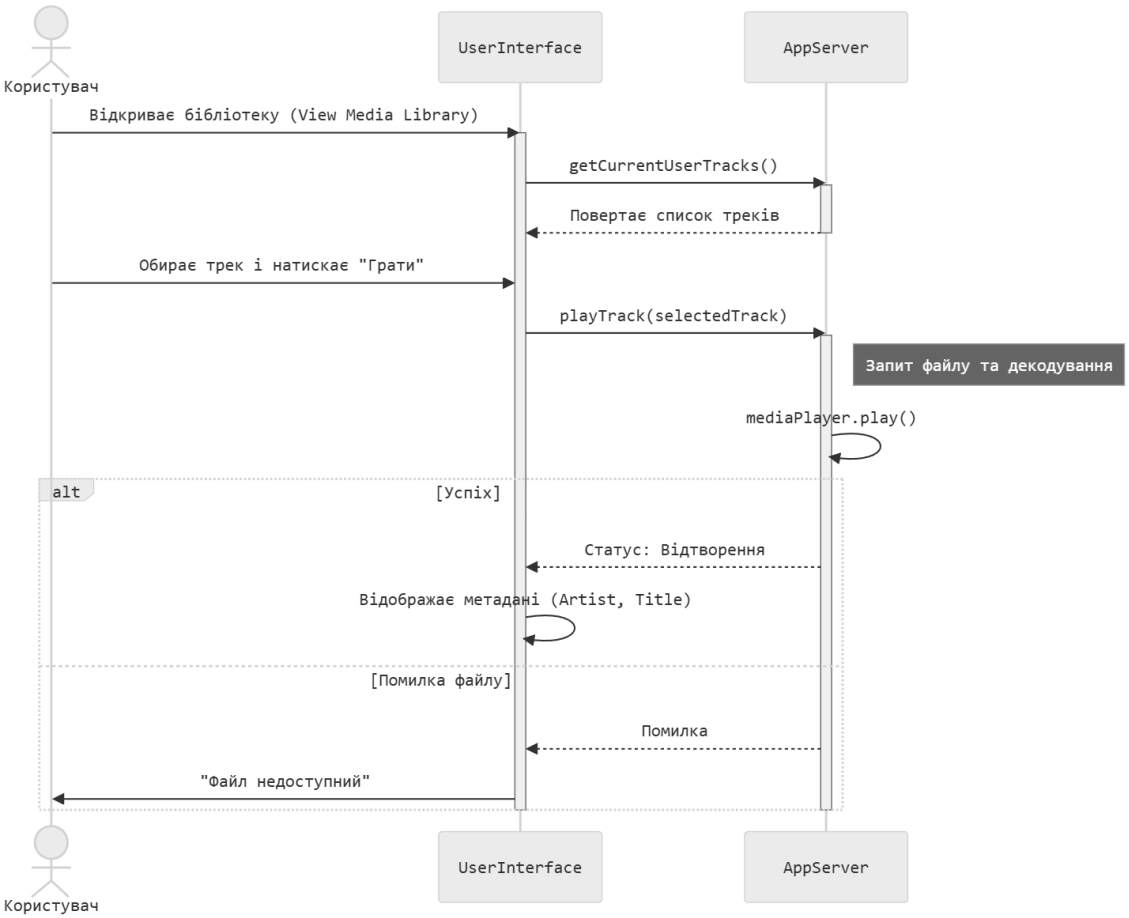

Рисунок 3. - Діаграма послідовностей для сценарію «Відтворення музичного треку»

Діаграма послідовностей для «Сценарію №1: Відтворення музичного треку» ілюструє взаємодію між трьома учасниками: Користувачем, UserInterface (клієнтським додатком) та AppServer (сервером застосунків).

Процес починається, коли Користувач відкриває бібліотеку треків. UserInterface надсилає запит на отримання списку треків поточного користувача до AppServer. AppServer обробляє запит і повертає перелік доступних композицій, який UserInterface відображає Користувачу.

Далі Користувач обирає конкретний трек та натискає кнопку «Грати». UserInterface передає команду playTrack на AppServer. AppServer виконує внутрішню логіку: отримує коректний URI аудіофайлу та ініціює відтворення медіа.

Діаграма використовує блок alt для обробки двох можливих результатів. У випадку «Успіху», AppServer оновлює статус відтворення, а UserInterface змінює іконку на паузу та відображає метадані треку (виконавця та назву). Якщо ж виникає «Помилка файлу» (наприклад, файл пошкоджений або

недоступний), AppServer повертає повідомлення про помилку, і UserInterface інформує про це Користувача.

Таким чином, діаграма демонструє чіткий розподіл обов'язків: UserInterface відповідає за візуалізацію та прийом команд, а AppServer виступає в ролі контролера, що керує доступом до даних та логікою медіаплеєра.

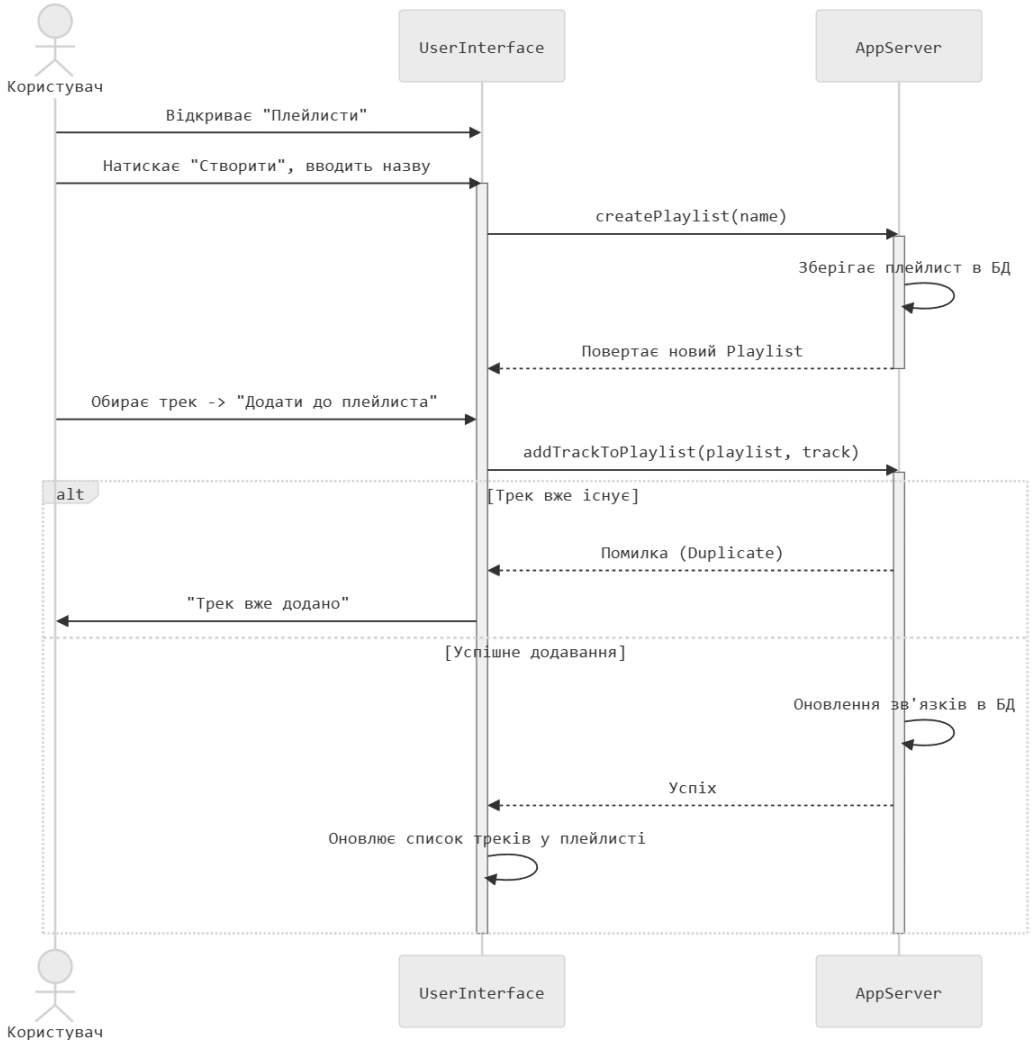

Рисунок 4. - Діаграма послідовностей для сценарію «Керування плейлистом (Створення та редагування)»

Діаграма послідовностей для «Сценарію №2: Керування плейлистом» ілюструє взаємодію між трьома учасниками: Користувачем, UserInterface (клієнтським додатком) та AppServer (сервером застосунків).

Процес починається, коли Користувач відкриває вкладку плейлистів. UserInterface відображає елементи управління, і після того, як користувач натискає «Створити» та вводить назву, інтерфейс надсилає запит createPlaylist

до AppServer. AppServer не зберігає дані локально, а виконує запит до бази даних для створення нового запису та повертає створений об'єкт плейлиста.

Далі діаграма використовує блок alt для обробки додавання треку. Коли користувач додає трек до списку, AppServer перевіряє його наявність. Якщо «Трек вже додано», AppServer повертає помилку дублювання, і UserInterface інформує про це Користувача відповідним повідомленням. Якщо ж додавання успішне, AppServer оновлює зв'язки в базі даних та повертає підтвердження, після чого UserInterface оновлює відображення списку.

Окремо показано потік «Видалення». Користувач ініціює видалення плейлиста, підтверджує дію у діалоговому вікні, і UserInterface надсилає запит на AppServer. AppServer виконує видалення даних, а UserInterface оновлює перелік доступних плейлистів.

Таким чином, діаграма демонструє чіткий розподіл обов'язків: UserInterface відповідає за взаємодію з користувачем та валідацію вводу, а AppServer виступає в ролі координатора бізнес-логіки та збереження даних.

Фото роботи програми

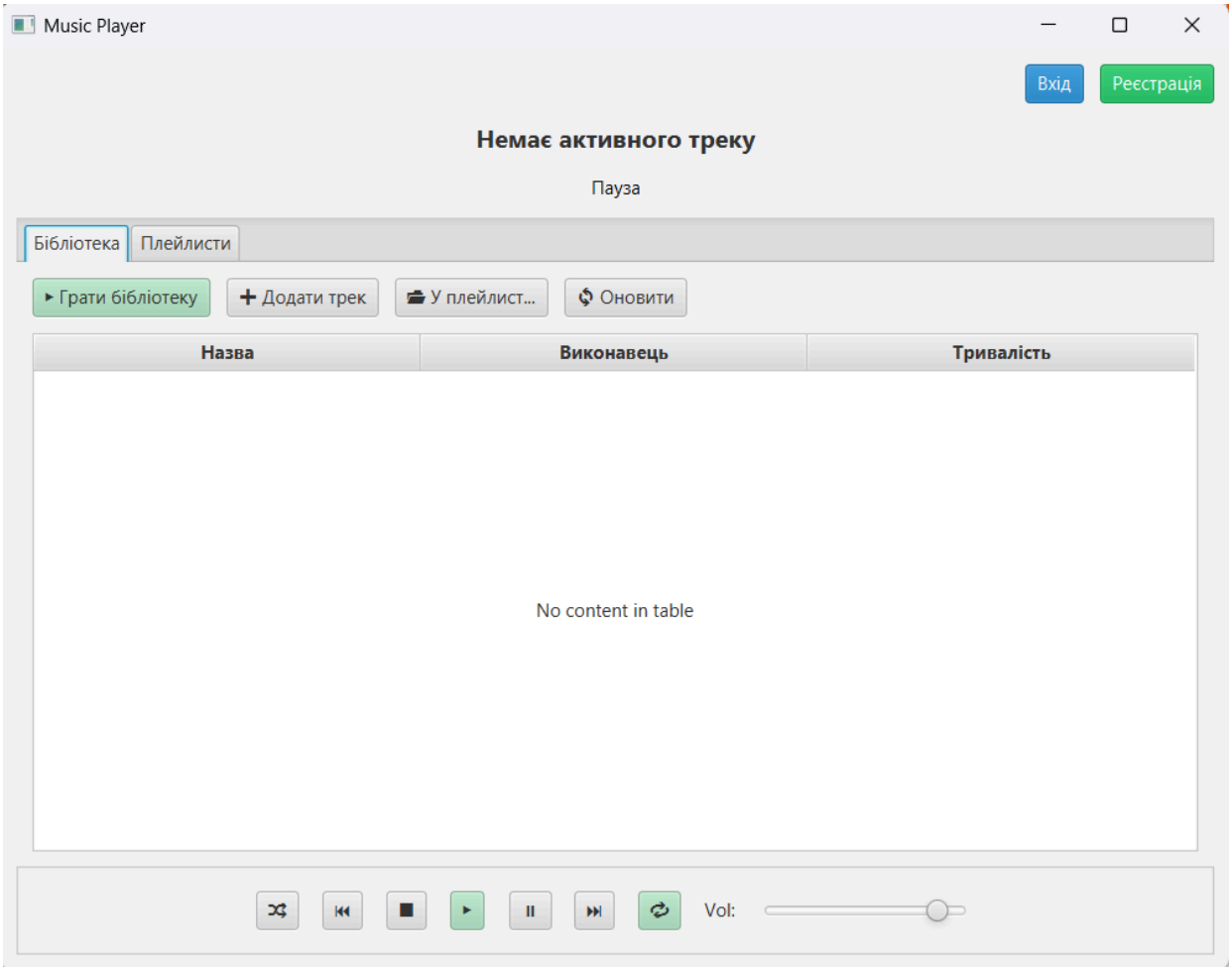

Рисунок 5.

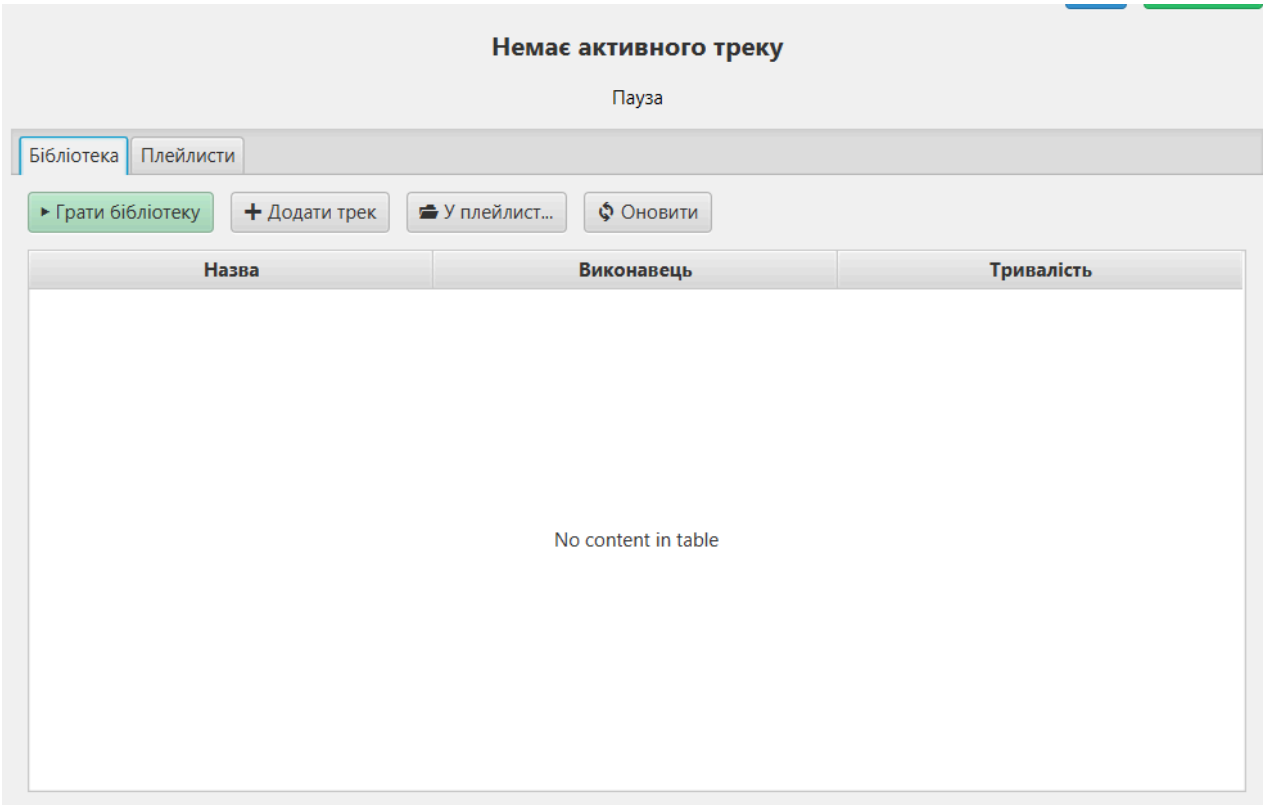
Немає активного треку
Пауза
Бібліотека
Плейлисти
Грати бібліотеку
Додати трек
У плейлист...
Оновити
Назва
Виконавець
Тривалість
No content in table

Рисунок 6.

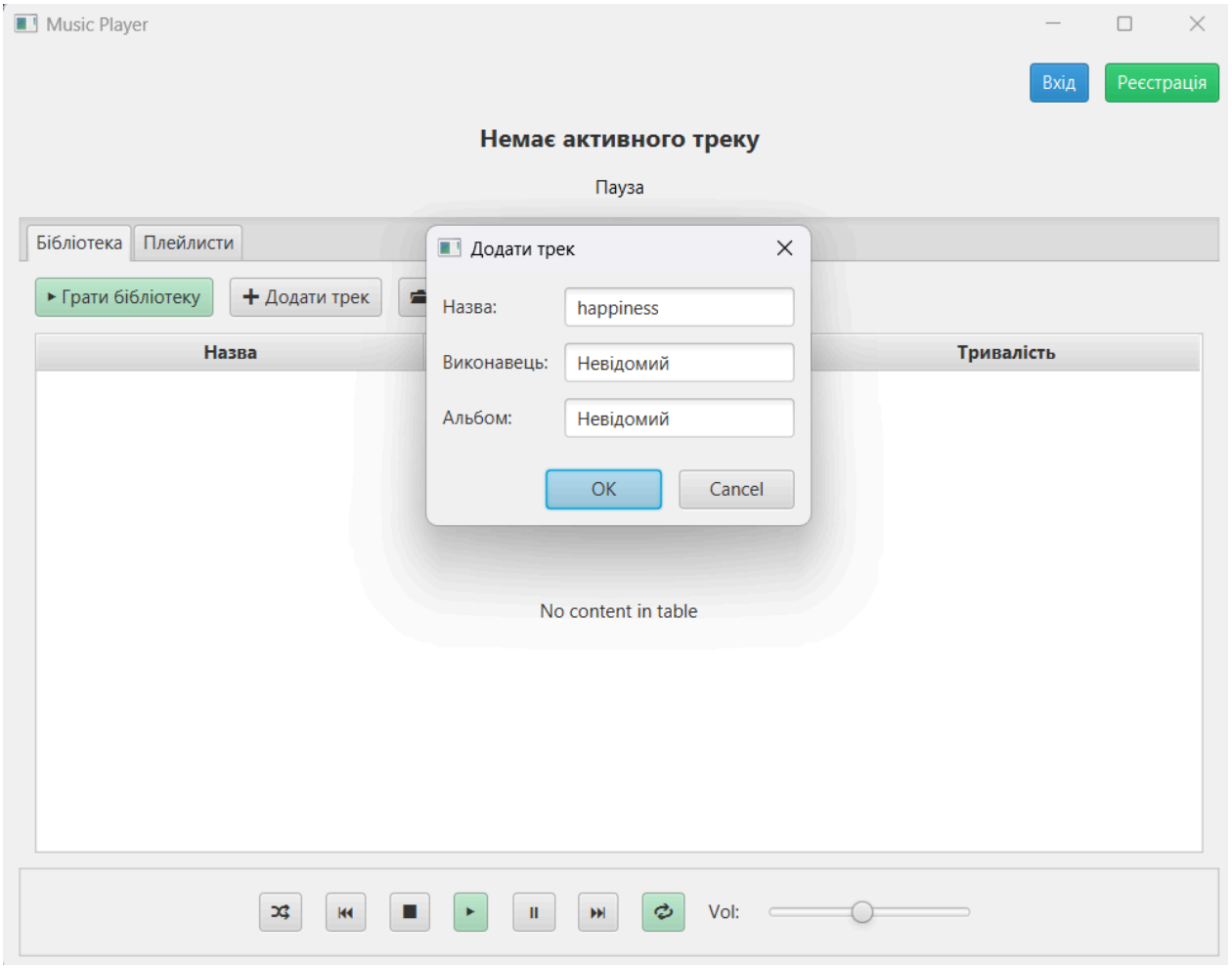
Music Player
Вхід
Реєстрація
Немає активного треку
Пауза
Бібліотека
Плейлисти
Грати бібліотеку
Додати трек
Назва
Тривалість
Додати трек
Назва:
happiness
Виконавець:
Невідомий
Альбом:
Невідомий
OK
Cancel
No content in table
Vol:

Рисунок 7.

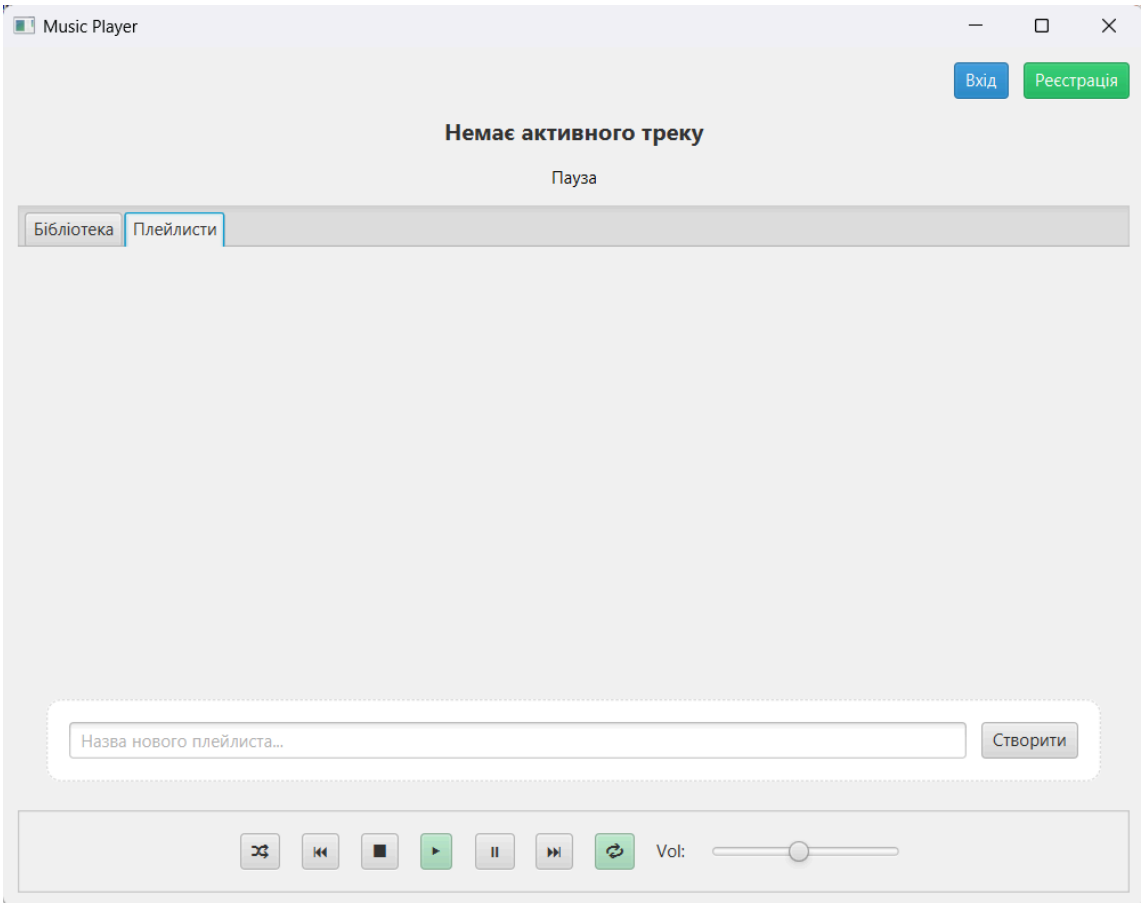

Рисунок 8.

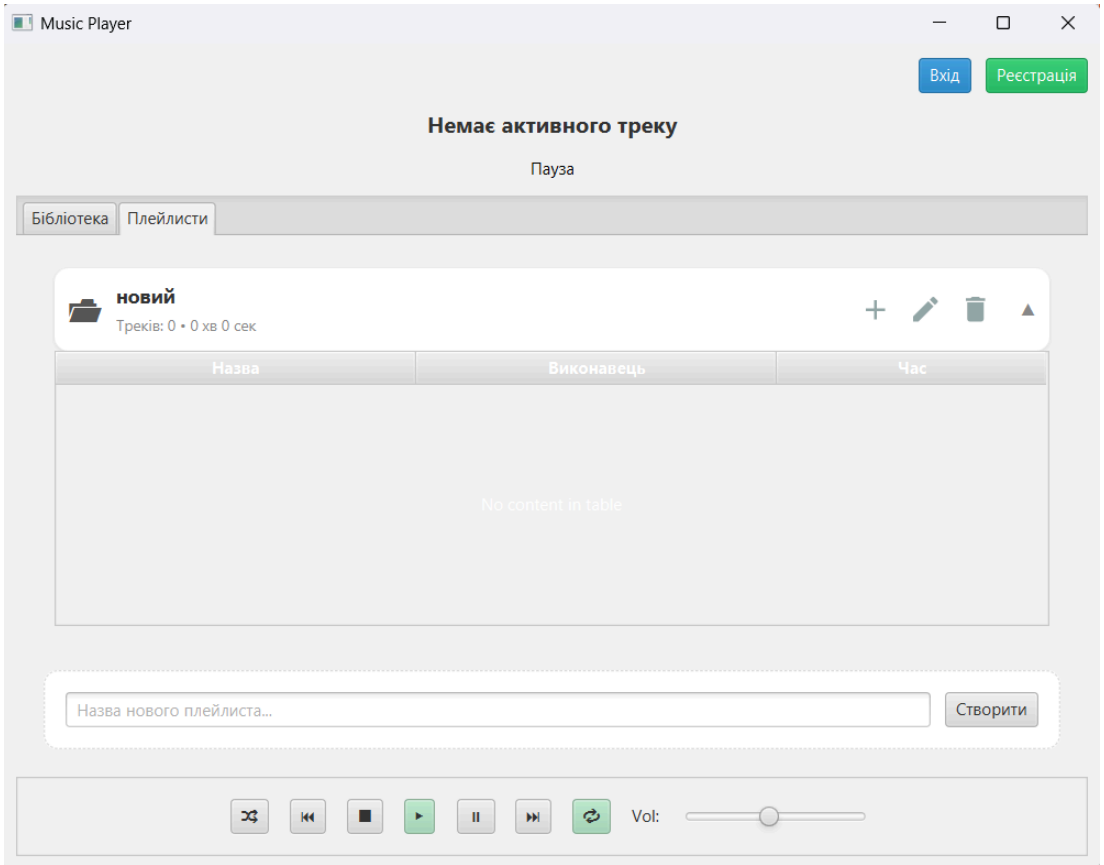

Рисунок 9.

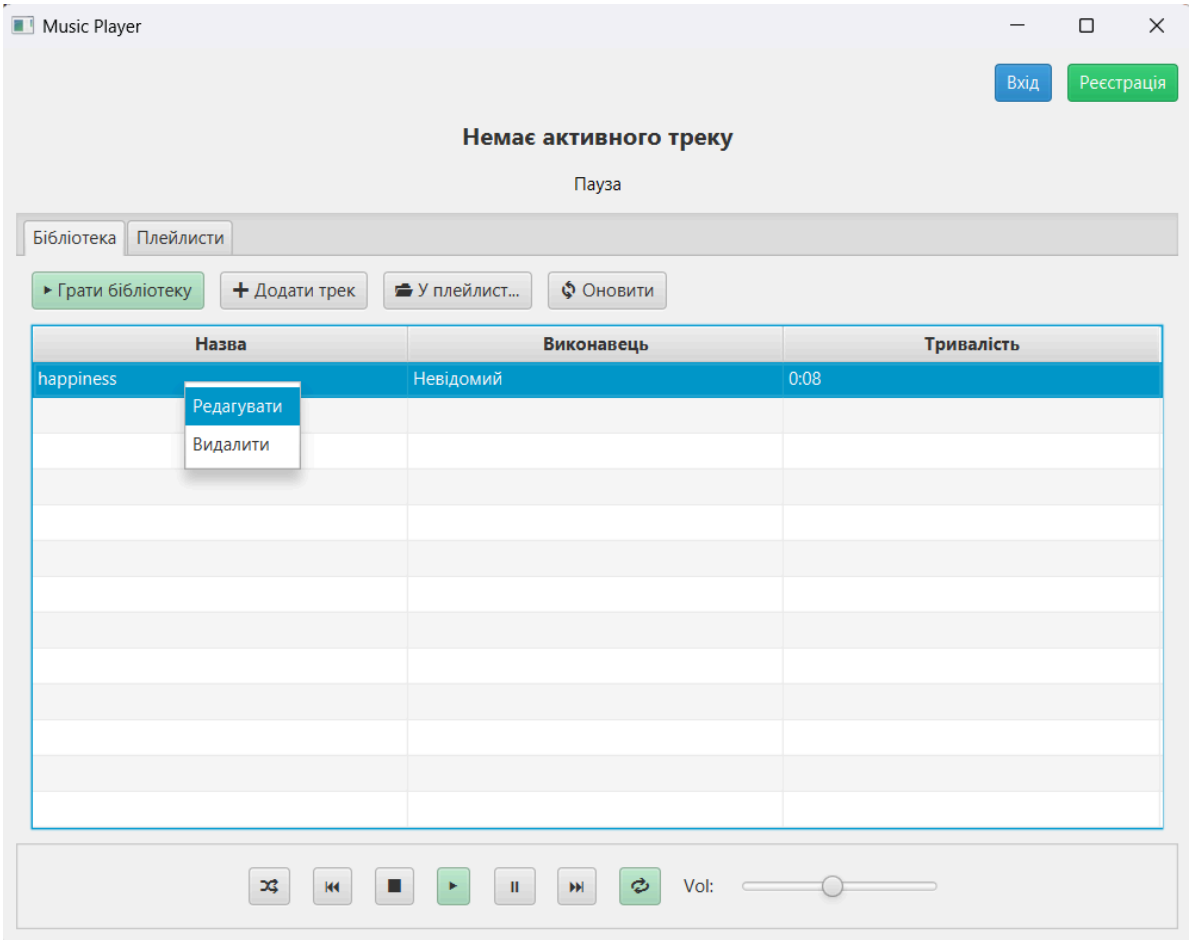

Рисунок 10.

Висновок: У ході лабораторної роботи було успішно застосовано принципи об'єктно-орієнтованого аналізу та проєктування для створення повної архітектури «Музичного програвача». Було проведено аналіз вимог, що включав побудову діаграми варіантів використання та розробку детальних сценаріїв. На основі цього була спроєктована логічна модель у вигляді діаграми класів , а також розроблена схема бази даних з використанням патерну Repository для відокремлення логіки доступу до даних. Архітектура системи була візуалізована за допомогою діаграми компонентів, а її динамічна поведінка — за допомогою діаграм послідовностей. На завершення, діаграма розгортання визначила фізичну клієнт-серверну архітектуру. Результатом роботи є повний, структурований та готовий до подальшої реалізації архітектурний проєкт музичного програвача.

Контрольні запитання

1. Що собою становить діаграма розгортання?

Діаграма розгортання — це тип діаграми UML, який показує фізичне розміщення програмних компонентів системи на апаратних пристроях. Вона відображає, на яких серверах або комп’ютерах розгорнуті частини програми та

як між ними здійснюється зв’язок. Така діаграма використовується для опису архітектури системи й показує, де саме працюють окремі компоненти.

2. Які бувають види вузлів на діаграмі розгортання?

Пристрій — це фізичний елемент системи, наприклад комп’ютер, сервер, смартфон або маршрутизатор. Він представляє апаратне забезпечення, на якому може виконуватись програмне забезпечення. Виконавче середовище — це програмна платформа, яка працює всередині пристрою й забезпечує виконання програм. Прикладом може бути операційна система, віртуальна машина Java або контейнер.

3. Які бувають зв’язки на діаграмі розгортання?

Зв’язок комунікації показує мережеву взаємодію між вузлами — наприклад, обмін даними між сервером і клієнтом. Такий зв’язок відображає, як пристрої або середовища з’єднані між собою. Зв’язок розміщення показує, який програмний компонент або артефакт розгорнуто на певному вузлі. Він відображає фізичне розташування програмних частин системи на пристроях.

4. Які елементи присутні на діаграмі компонентів?

Компонент — це основний елемент діаграми, який представляє окрему частину системи, наприклад модуль, бібліотеку або службу. Інтерфейс показує, які функції або сервіси надає чи використовує компонент Залежність відображає відношення між компонентами, коли один із них потребує функцій іншого. З’єднання показують взаємодію між компонентами через їхні інтерфейси.

5. Що становлять собою зв’язки на діаграмі компонентів?

Зв’язки на діаграмі компонентів показують взаємозалежність і взаємодію між різними компонентами системи. Вони відображають, як один компонент використовує або надає функціональність іншому. Найпоширенішим типом зв’язку є залежність, яка показує, що один компонент потребує інтерфейс або сервіс іншого для своєї роботи. Також можуть бути з’єднання через інтерфейси, які відображають реальну взаємодію між компонентами під час виконання програми.

6. Які бувають види діаграм взаємодії?

Діаграма послідовності показує обмін повідомленнями між об’єктами у часі — тобто хто, коли і в якій послідовності викликає певні операції. Діаграма кооперації (комунікації) відображає ті самі взаємодії, але зосереджується не на часовій послідовності, а на структурних зв’язках між об’єктами, які беруть участь у взаємодії.

7. Для чого призначена діаграма послідовностей?

Діаграма послідовностей призначена для відображення процесу взаємодії між об’єктами у часі. Вона показує, які об’єкти беруть участь у певному сценарії системи, які повідомлення вони надсилають одне одному та в якій послідовності це відбувається. За допомогою такої діаграми можна наочно

простежити логіку виконання операцій, порядок викликів методів і реакцію системи на певні події. Діаграма послідовностей використовується під час аналізу або проєктування програм, щоб описати поведінку системи у динаміці.

8. Які ключові елементи можуть бути на діаграмі послідовностей?

На діаграмі послідовностей основними елементами є об’єкти, лінії життя, повідомлення та активації. Об’єкти представляють учасників взаємодії, які надсилають або отримують повідомлення. Лінія життя показує існування об’єкта протягом часу — вона зображується вертикально під об’єктом. Повідомлення відображають виклики методів або обмін даними між об’єктами та показуються стрілками. Активація позначає період, коли об’єкт виконує певну дію у відповідь на отримане повідомлення.

9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання?

Діаграми послідовностей тісно пов’язані з діаграмами варіантів використання, оскільки вони деталізують поведінку системи, описану у варіантах використання. Діаграма варіантів використання показує, які функції виконує система та які актори взаємодіють із нею. Діаграма послідовностей конкретизує ці сценарії, показуючи покрокову взаємодію об’єктів і обмін повідомленнями, необхідними для реалізації певного варіанту використання.

10. Як діаграми послідовностей пов’язані з діаграмами класів?

Діаграми послідовностей пов’язані з діаграмами класів тим, що вони описують взаємодію об’єктів конкретних класів у часі. Діаграма класів визначає структуру системи, показуючи класи, їхні атрибути та методи, а також зв’язки між класами. Діаграма послідовностей показує, як об’єкти цих класів взаємодіють один з одним під час виконання певного сценарію, які методи викликаються і в якому порядку.