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

Лабораторна робота №3
Технології розроблення програмного забезпечення
«Основи проектування розгортання.»

Виконав:

Студент групи ІА-32

Костінський Макарій

Перевірив:

Мягкий Михайло Юрійович

Зміст

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.

Теоретичні відомості

1. Діаграма розгортання (Deployment Diagram)

Діаграми розгортання представляють фізичне розташування системи, показуючи, на якому фізичному обладнанні запускається та чи інша складова програмного забезпечення .

Головними елементами діаграми є вузли, пов'язані інформаційними шляхами. Вузол (node) – це те, що може містити програмне забезпечення. Вузли бувають двох типів. Пристрій (device) – це фізичне обладнання: комп'ютер або пристрій, пов'язаний із системою. Середовище виконання (execution environment) – це програмне забезпечення, яке саме може включати інше програмне забезпечення, наприклад операційну систему або процес-контейнер (наприклад, вебсервер).

2. Діаграма компонентів

Діаграма компонентів UML є представленням проєктованої системи,

розбитої на окремі модулі. Залежно від способу поділу на модулі розрізняють

три види діаграм компонентів:

• логічні;

• фізичні;

• виконувані.

Коли використовують логічне розбиття на компоненти, то у такому разі проєктовану систему віртуально уявляють як набір самостійних, автономних модулів (компонентів), що взаємодіють між собою.

Коли на діаграмі представляють фізичне розбиття, то в такому разі на діаграмі компонентів показують компоненти та залежності між ними. Залежності показують, що класи в з одного компонента використовують класи з іншого компонента. Фізична модель використовується для розуміння які компоненти повинні бути зібрані в інсталяційний пакет. Також така діаграма показує зміни в якому компоненті будуть впливати на інші компоненти.

На діаграмах компонентів з виконуваним поділом компонентів кожен компонент являє собою деякий файл – виконувані файли (.exe), файли вихідних кодів, сторінки html, бази даних і таблиці тощо. У цьому разі діаграма схожа на діаграму класів, але на більш верхньому рівні – рівні виконуваних файлів або процесів. Такий підхід дає інший розріз представлення системи.

3. Діаграми послідовностей

Діаграма послідовностей (Sequence Diagram) – це один із типів діаграм у моделюванні UML (Unified Modeling Language), який використовується для моделювання взаємодії між об'єктами системи у певній послідовності часу. Вона відображає, як об'єкти обмінюються повідомленнями, показуючи порядок і логіку виконання операцій.

Діаграма складається з таких основних елементів:

Актори (Actors): Зазвичай позначаються піктограмами або назвами. Це користувачі чи інші системи, які взаємодіють із системою. Актори можуть бути зовнішніми стосовно моделювання системи.

Об'єкти або класи: Розміщуються горизонтально на діаграмі. Вони позначаються прямокутниками з іменем об'єкта або класу під прямокутником. Кожен об'єкт має «життєвий цикл», який представлений вертикальною пунктирною лінією (лінія життя).

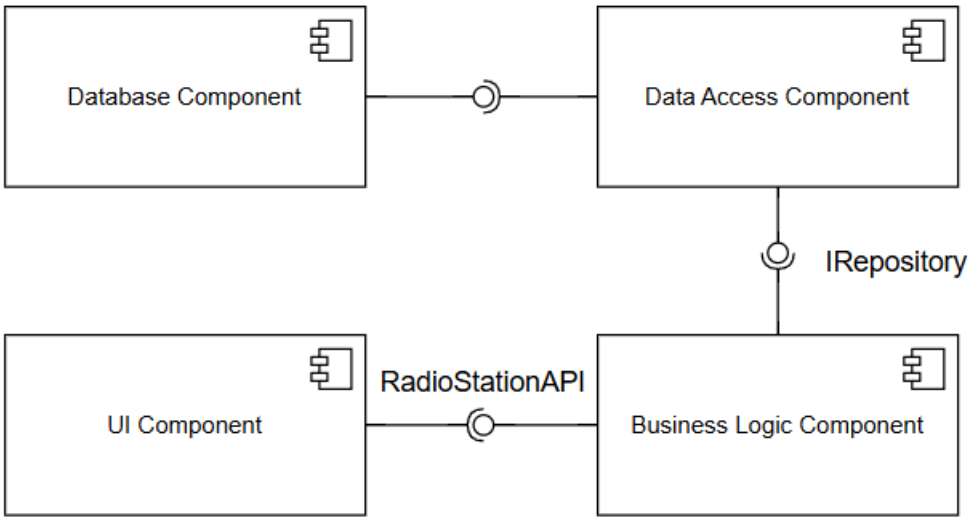
Повідомлення: Це лінії зі стрілками, які з'єднують об'єкти. Вони показують передачу повідомлень чи виклик методів. Стрілка може бути синхронною (звичайна стрілка) або асинхронною (лінія з відкритим трикутником) та з пунктирною лінією, що показує повернення результату.

Активності: Вказують періоди, протягом яких об'єкт виконує певну дію. На діаграмі це позначається прямокутником, накладеним на лінію життя.

Контрольні структури: Використовуються для відображення умов, циклів або альтернативних сценаріїв. Наприклад, блоки "alt" (альтернатива) або "loop" (цикл).

Хід роботи

Діаграма компонентів



На діаграмі компонентів представлено логічну структуру програмної системи, яка розділена на чотири основні модулі.

Система складається з наступних компонентів:

1. UI Component (Компонент інтерфейсу користувача)

Призначення: Відповідає за взаємодію з кінцевим користувачем (слухачем або адміністратором). Відображає дані та приймає команди.

Реалізація в коді: Відповідає пакету ua.kpi.iasa.onlineradio.ui. Включає класи форм (LoginForm, PlayerForm, AdminForm), розроблені з використанням бібліотеки Java Swing.

Залежності: Компонент залежить від бізнес-логіки через інтерфейс RadioStationAPI.

2. Business Logic Component (Компонент бізнес-логіки)

Призначення: Реалізує основний функціонал системи: управління трансляцією, формування плейлистів, обробку правил додавання треків.

Реалізація в коді: Відповідає пакету ua.kpi.iasa.onlineradio.models. Включає класи RadioStation, Streamer, MusicLibrary.

Інтерфейс: Надає інтерфейс RadioStationAPI для використання компонентом UI.

Залежності: Для своєї роботи потребує доступу до даних, тому залежить від компонента Data Access через інтерфейс IRepository.

3. Data Access Component (Компонент доступу до даних)

Призначення: Інкапсулює механізми збереження та отримання даних, приховуючи деталі реалізації сховища від бізнес-логіки.

Реалізація в коді: Відповідає пакетам ua.kpi.iasa.onlineradio.repositories та ua.kpi.iasa.onlineradio.data. Включає класи репозиторіїв (TrackRepository, UserRepository).

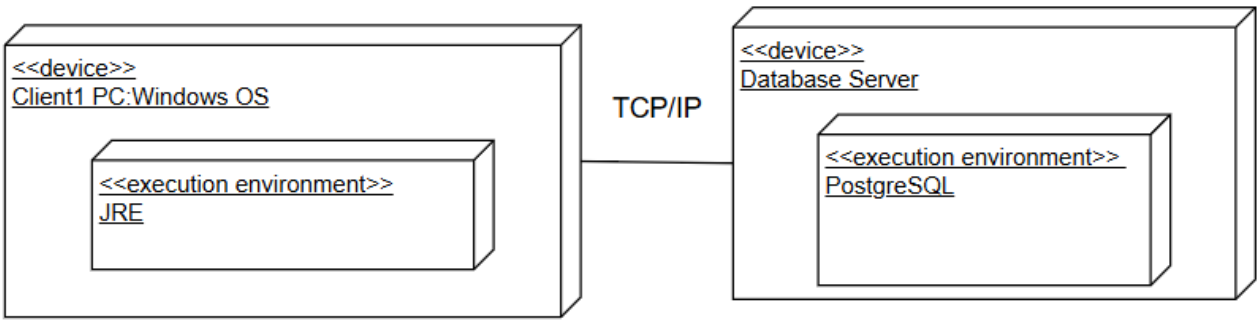
Інтерфейс: Реалізує інтерфейс IRepository, надаючи методи CRUD (Create, Read, Update, Delete).

4. Database Component (База даних)

Призначення: Фізичне збереження інформації про користувачів, треки та статистику прослуховувань.

Реалізація: У поточній версії може бути реалізована як in-memory сховище (колекції Map або List) або як зовнішня реляційна база даних, з якою взаємодіє Data Access Component.

Діаграма розгортання (Deployment Diagram)



На діаграмі представлено фізичну архітектуру системи "Online Radio Station". Система побудована за дворівневою архітектурою "Клієнт-Сервер".

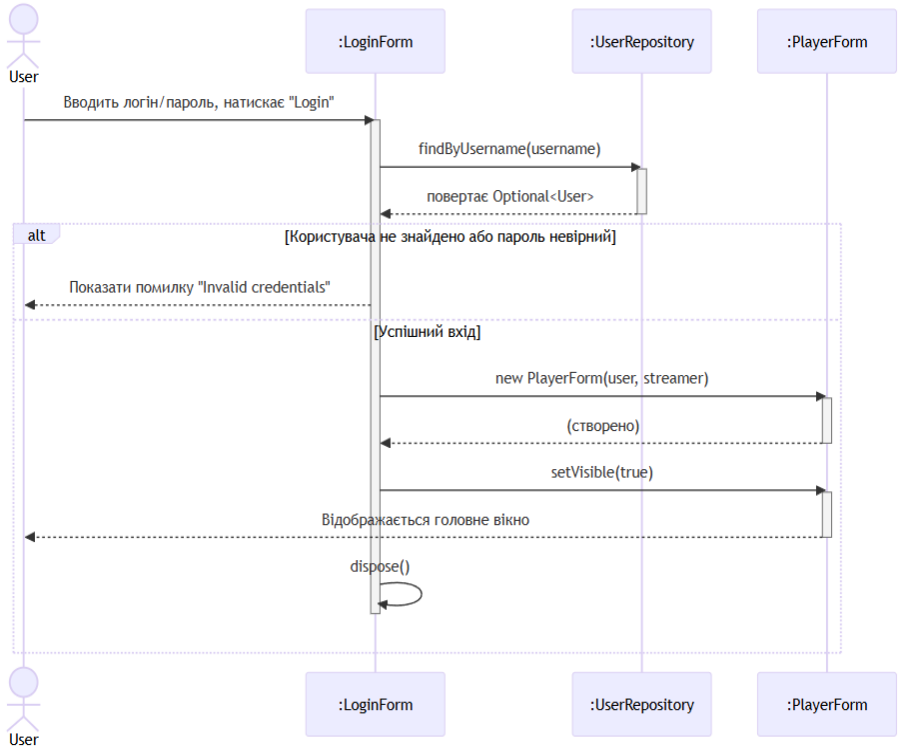
Вузли системи:
Client Workstation (Робоча станція клієнта): Фізичний пристрій (ПК або ноутбук) кінцевого користувача. На цьому вузлі розгорнуто середовище виконання Java Runtime Environment (JRE), яке необхідне для запуску головного виконуваного артефакту системи — файлу OnlineRadioApp.jar. Цей компонент відповідає за візуалізацію інтерфейсу та виконання бізнес-логіки додатку.

Database Server (Сервер бази даних): Виділений сервер, на якому функціонує СУБД (наприклад, PostgreSQL). Тут розміщено артефакт бази даних (Radio_DB), що зберігає інформацію про користувачів, треки, плейлисти та історію прослуховувань.

Комунікація: Взаємодія між клієнтським додатком та сервером бази даних здійснюється через мережевий протокол TCP/IP з використанням технології JDBC для передачі SQL-запитів та отримання даних.

Діаграма послідовностей (Sequence Diagram)

1. Опис діаграми послідовності "Вхід в систему" (User Login)



Ця діаграма моделює взаємодію об'єктів під час спроби користувача увійти до системи. Вона демонструє перевірку облікових даних та перехід до основного функціоналу додатку.

Учасники взаємодії:
User (Користувач): Активна дійова особа, що ініціює процес.

:LoginForm (UI): Графічна форма входу, яка приймає введення даних.

:UserRepository (Data Access): Компонент доступу до даних, що відповідає за пошук інформації про користувачів.

:PlayerForm (UI): Головне вікно плеєра, яке створюється після успішної авторизації.

Алгоритм взаємодії:

Користувач вводить логін та пароль у форму LoginForm і натискає кнопку входу.

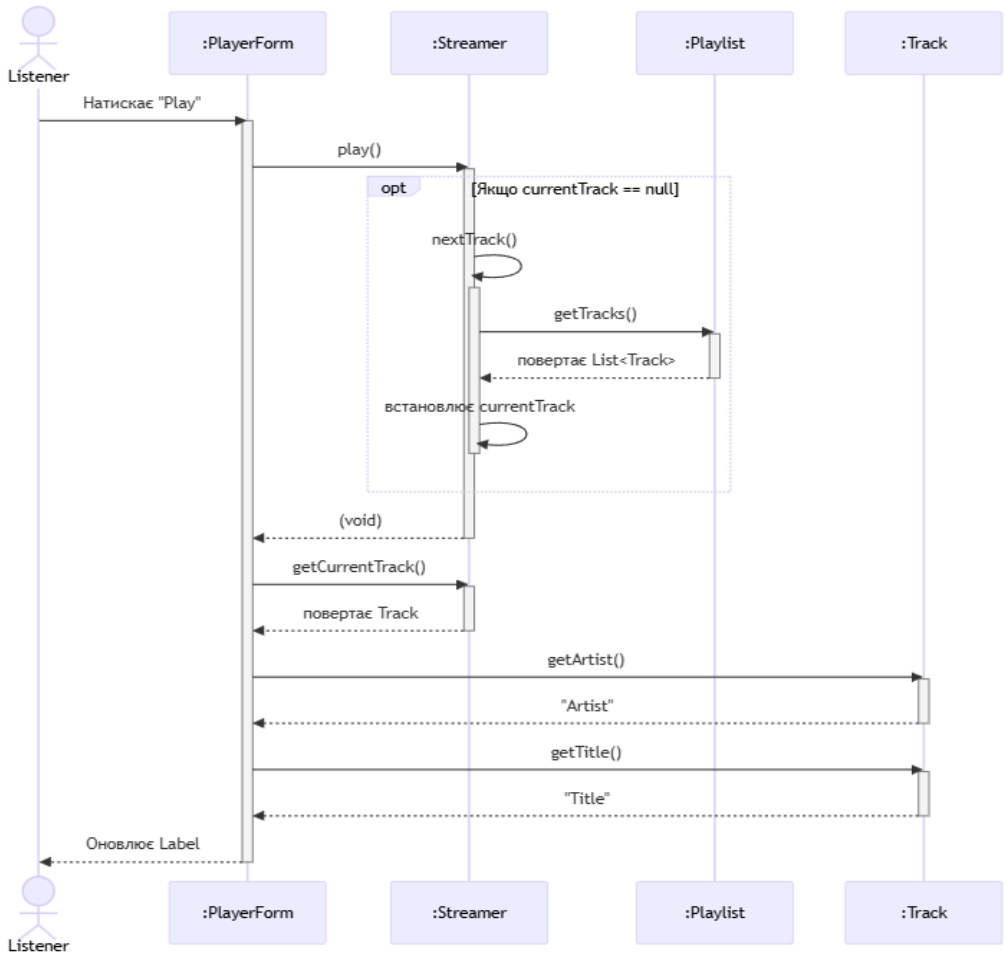
LoginForm надсилає запит findByUsername() до об'єкта UserRepository для пошуку користувача в базі даних.

UserRepository повертає знайдений об'єкт користувача (або null/Optional.empty, якщо такого не існує).

Далі діаграма розгалужується (блок alt):

- Успішний сценарій: Якщо пароль вірний, LoginForm створює новий екземпляр PlayerForm (головне вікно), робить його видимим (setVisible(true)), а сама завершує свою роботу (dispose()). Користувач отримує доступ до плеєра.
- Альтернативний сценарій: Якщо користувача не знайдено або пароль невірний, LoginForm відображає повідомлення про помилку, і перехід до головного вікна не відбувається.

2. Опис діаграми послідовності "Прослуховування музики" (Listening to Music)



Ця діаграма описує динаміку взаємодії між інтерфейсом користувача та бізнес-логікою під час початку відтворення треку. Вона демонструє, як запит користувача обробляється контролером (Streamer) і як оновлюється інтерфейс.

Учасники взаємодії:

Listener (Слухач): Користувач, що взаємодіє з плеєром.

:PlayerForm (UI): Головне вікно програми.

:Streamer (Business Logic): Сервіс, що керує станом відтворення та поточним треком.

:Playlist (Model): Сутність, що зберігає список треків.

:Track (Model): Сутність окремої музичної композиції.

Алгоритм взаємодії:

Слухач натискає кнопку "Play" на формі PlayerForm.

PlayerForm делегує команду бізнес-логіці, викликаючи метод play() у об'єкта Streamer.

- Опціональний блок (opt): Всередині методу play() Streamer перевіряє, чи вибрано поточний трек. Якщо трек не встановлено (наприклад, це перший запуск), Streamer звертається до об'єкта :Playlist, отримує список треків і встановлює перший доступний трек як поточний.

Після запуску відтворення Streamer повертає керування формі.

PlayerForm запитує актуальну інформацію про трек, викликаючи getCurrentTrack().

Отримавши об'єкт :Track, форма послідовно запитує дані про виконавця (getArtist()) та назву (getTitle()).

На основі отриманих даних PlayerForm оновлює візуальні елементи (Label), відображаючи користувачеві інформацію про те, що зараз грає.

Вихідний код

https://github.com/TRPZ-Labs/Lab3

Висновок:

У ході виконання лабораторної роботи набуто практичних навичок проєктування архітектури програмних систем та реалізації графічних інтерфейсів користувача для предметної області «Online Radio Station».

Розроблено діаграму компонентів для логічного розділення системи на шари представлення, бізнес-логіки та доступу до даних, а також діаграму розгортання, що описує фізичну архітектуру клієнт-серверної взаємодії. Крім того, змодельовано динаміку роботи системи через діаграми послідовностей для ключових процесів авторизації та прослуховування музики, що дозволило чітко визначити обмін повідомленнями між об'єктами. На основі спроєктованих моделей модернізовано програмний код, додано візуальні форми з використанням бібліотеки Java Swing та реалізовано їхню взаємодію з логічним шаром та репозиторіями даних.