Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут» імені Ігоря Сікорського
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота № 3

із дисципліни: «Технології розроблення програмного забезпечення»

на тему: «ДІАГРАМА РОЗГОРТАННЯ. ДІАГРАМА КОМПОНЕНТІВ. ДІАГРАМА ВЗАЄМОДІЙ ТА ПОСЛІДОВНОСТЕЙ**»**

Виконав:
студент групи ІА-34
Вінницький Г.Р.
Перевірил:
Мягкий Михайло Юрійович

Київ-2025

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.

18. **Shell (total commander)** (state, prototype, factory method, template method, interpreter, client-server)

Оболонка повинна вміти виконувати основні дії в системі – перегляд файлів папок в файлової системі, перемикання між дисками, копіювання, видалення, переміщення об'єктів, пошук.

2. Розробка діаграми розгортання для проектованої системи:

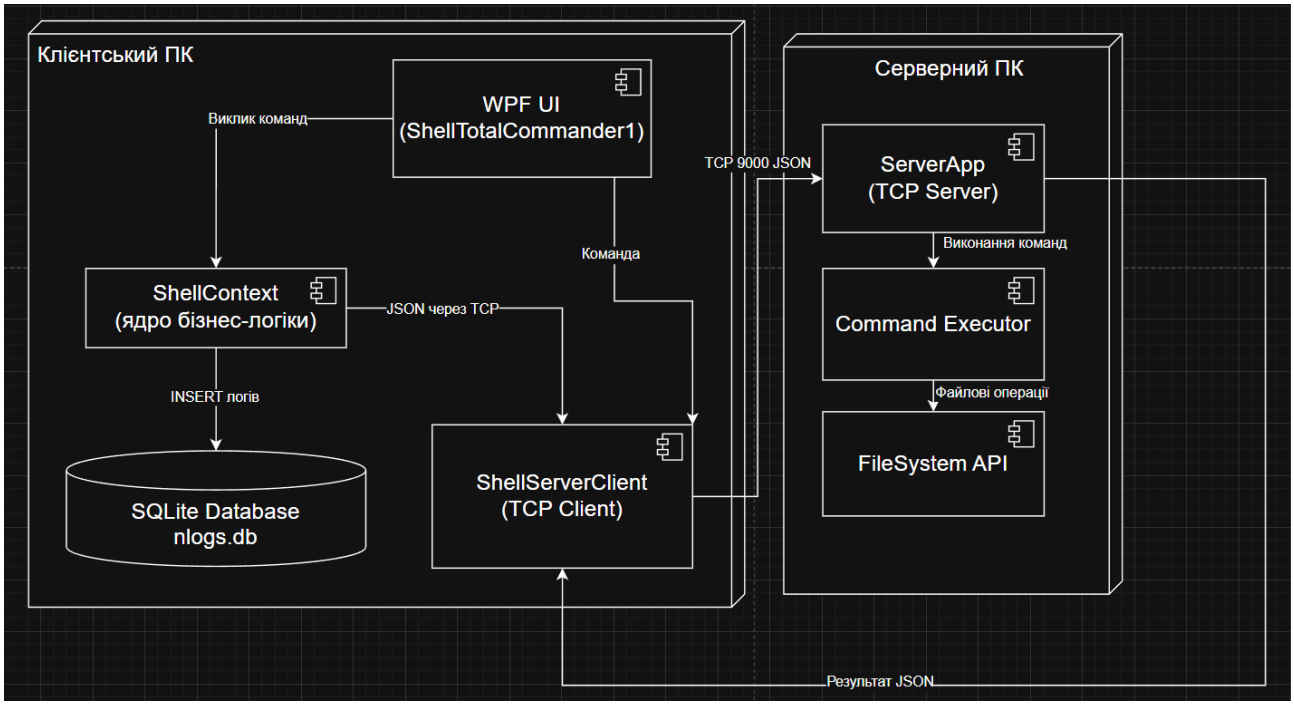


Рис. 1.1 – Діаграма розгортання системи

Опис діаграми розгортання

1. Клієнтський ПК

На клієнтській машині розгорнуто три основні компоненти:

1. WPF UI (ShellTotalCommander1)

Графічний інтерфейс користувача, що забезпечує:

1) перегляд каталогів,

2) навігацію,

3) відображення результатів команд,
4) введення текстових команд.

2. ShellContext (ядро бізнес-логіки)

Компонент, який відповідає за:

1) обробку команд,
2) запуск локального виконання,
3) взаємодію з інтерпретатором та логером,
4) визначення, чи використовувати сервер.

3. ShellServerClient

TCP-клієнт, який:

1) формує JSON-запити на сервер,
2) надсилає команди на порт 9000,
3) приймає JSON-відповіді.

4. SQLite Database (logs.db)

Використовується для:

1) логування виконаних команд,
2) збереження повідомлень результатів, аргументів та часу.

2. Серверний ПК

Сервер може працювати як на окремому вузлі, так і на тому ж комп’ютері, що і клієнт.

1. ServerApp (TCP Server)

Слухає порт 9000, приймає JSON-команди, виконує їх на сервері.

2. Command Executor

Виконує такі операції:

1) перегляд каталогів,
2) копіювання,
3) переміщення,
4) видалення,
5) пошук файлів.

3. FileSystem API

Взаємодіє з файловою системою операційної системи через бібліотеку System.IO.

3. Комунікація між вузлами

TCP-з'єднання (порт 9000)

Обмін JSON-об’єктами:

1) CommandRequest
2) CommandResponse

проходить між:

ShellServerClient (клієнт)

↕ TCP JSON

ServerApp (сервер)

3. Розробка діаграму компонентів для проектованої системи.

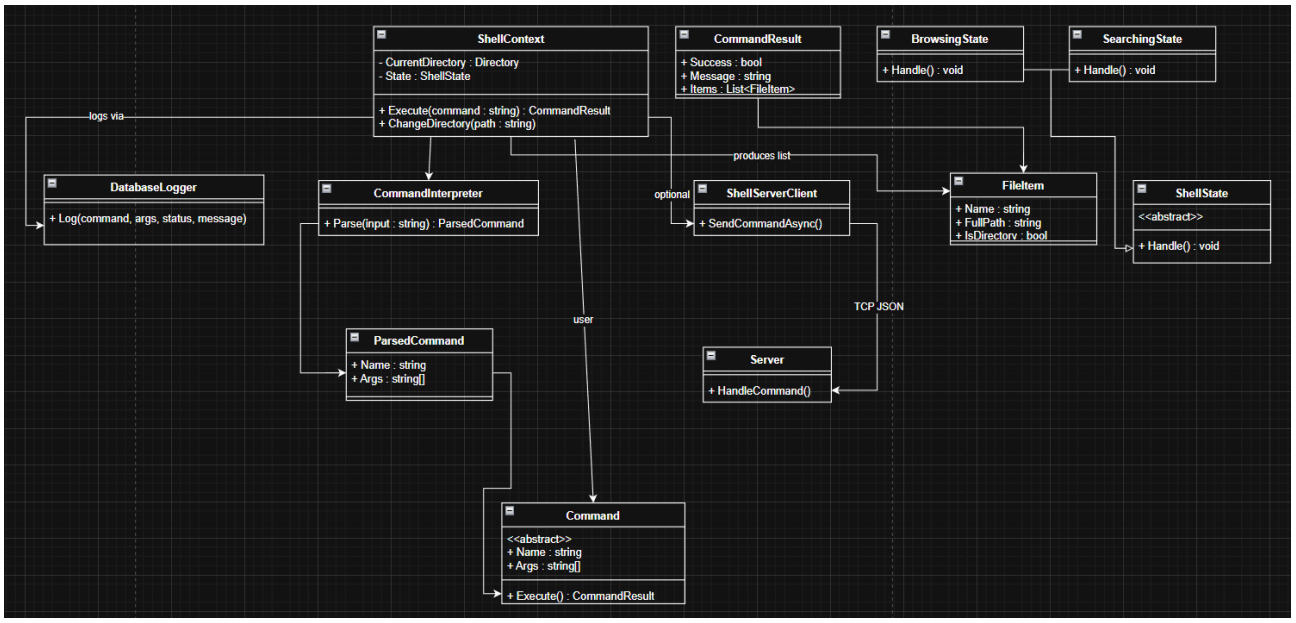


Рис. 1.2 – Діаграма компонентів системи

Опис діаграми класів:

Концептуальна UML-діаграма представляє структуру системи Shell Total Commander на логічному рівні.

ShellContext

Головний клас, що координує роботу системи:

1) зберігає поточну директорію;
2) виконує команди;
3) змінює стан за патерном State;
4) взаємодіє з інтерпретатором, логером та клієнтом сервера.

Command / ParsedCommand

1) ParsedCommand описує структуру розібраної команди (ім’я + аргументи).
2) Command - абстракція виконуваної операції (copy, move, delete, ls, search).

3) Реальний клас Command працює через Template Method / Factory Method.

CommandInterpreter

Відповідає за синтаксичний розбір введеної користувачем строки.

Він перетворює текст у ParsedCommand.

ShellState (BrowsingState, SearchingState)

Патерн State:

1) визначає режим роботи оболонки;
2) стан впливає на доступні команди та логіку.

FileItem

Абстракція файла або директорії:

1) використовується для відображення у UI;
2) передається в CommandResult.

ShellServerClient ↔ Server

Патерн Client-Server:

1) клієнт надсилає JSON-запит;
2) сервер виконує команду і повертає JSON-відповідь.

DatabaseLogger

Відповідає за логування:

1) записує в SQLite факт виконання команди;
2) зберігає аргументи, результат, статус.

CommandResult

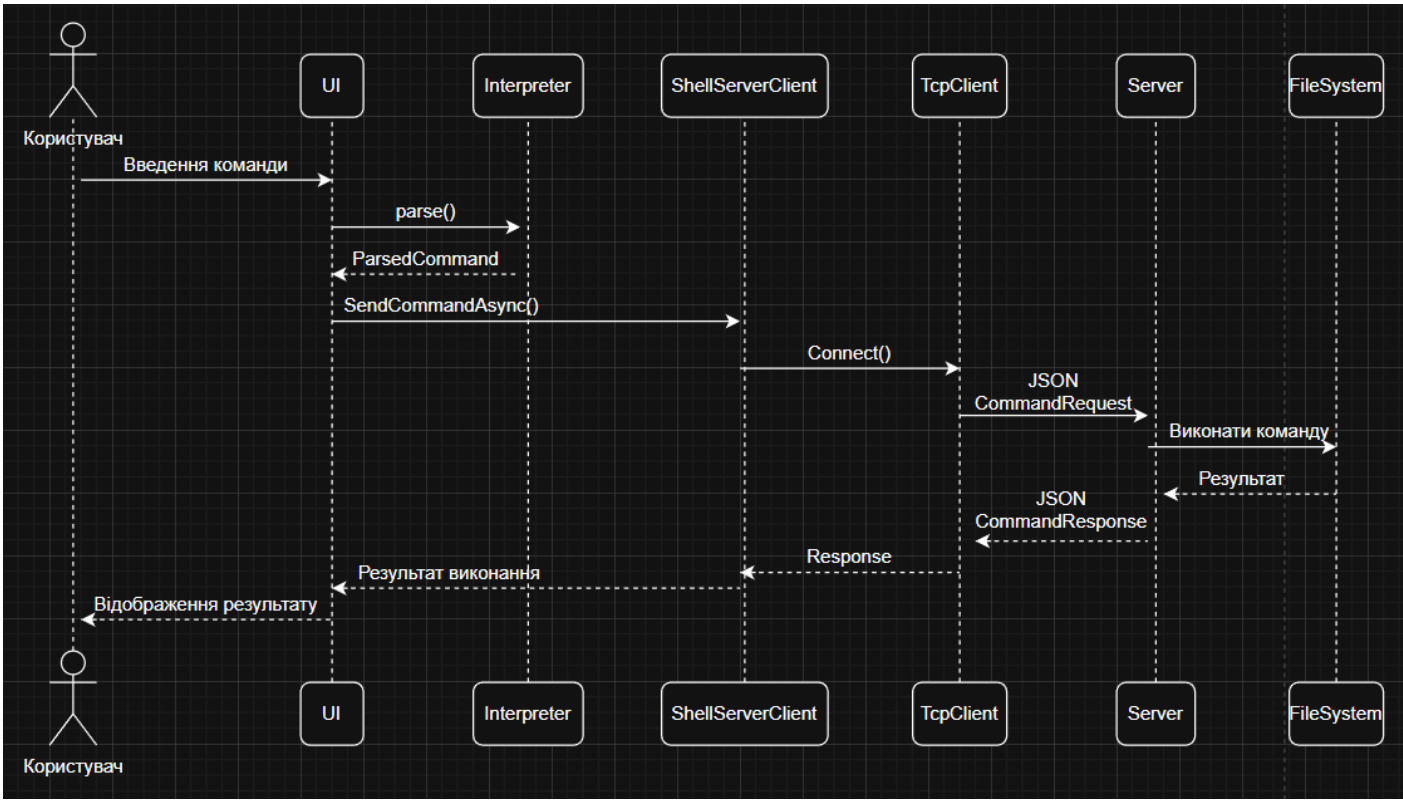
Описує результат виконання:

1) повідомлення;
2) успішність;
3) список елементів (файли/каталоги

4. Розробка діаграми послідовностей для проектованої системи.

Сценарій взаємодії з сервером (UC9)

Діаграма послідовності:

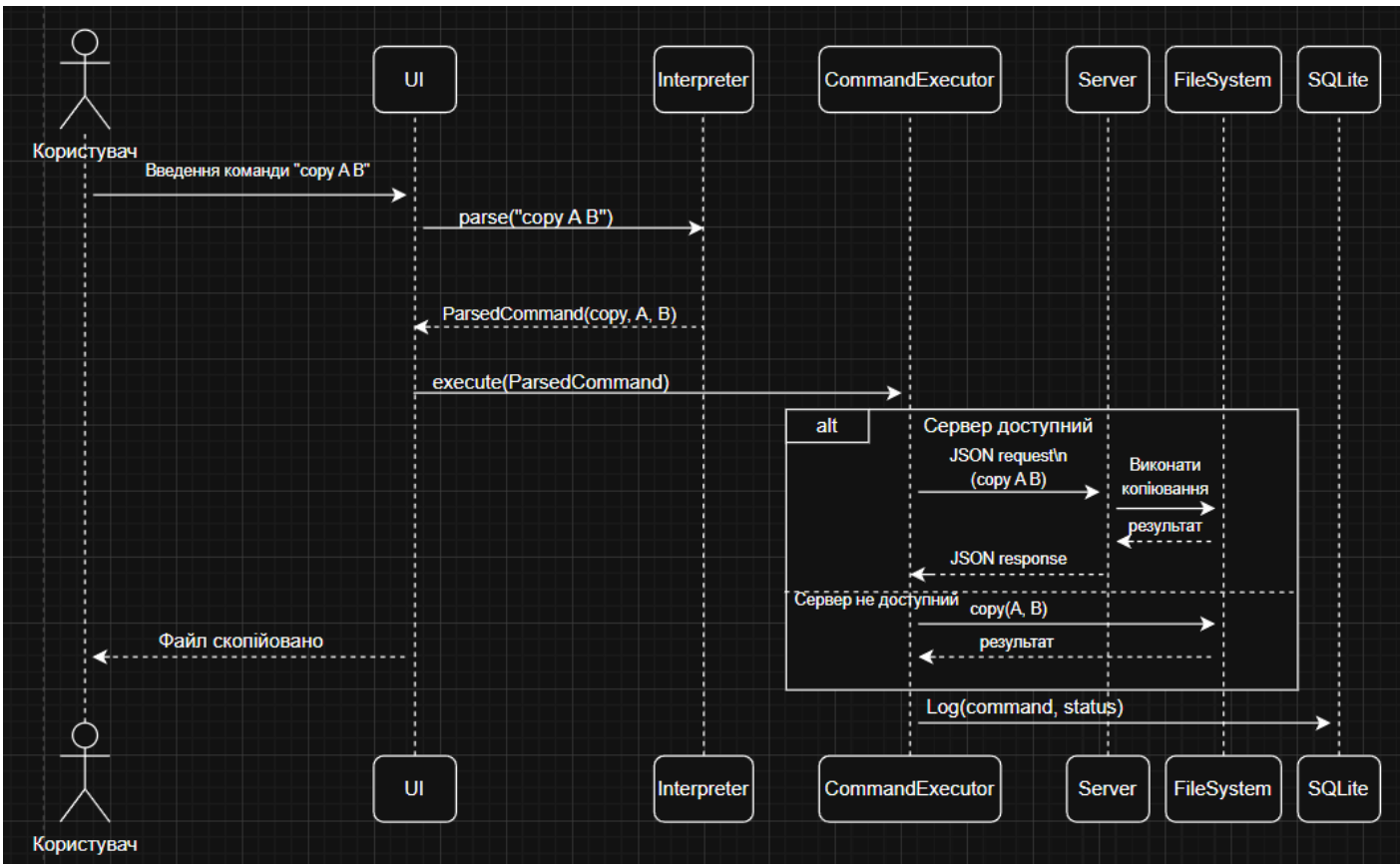


Текстовий опис:

1. Користувач вводить команду (наприклад, ls).
2. UI передає команду до інтерпретатора.
3. ParsedCommand передається у ShellServerClient.
4. Клієнт відкриває TCP-з’єднання на порт 9000.
5. Відправляє JSON-модель CommandRequest.
6. Сервер виконує команду (через локальні файлові операції).
7. Результати повертаються у JSON CommandResponse.
8. UI оновлюється і показує результат.

Виконання команди копіювання через інтерпретатор (UC2 + UC7 + UC8/UC9 + UC10)

Діаграма послідовності:



Текстовий опис сценарію

1. Користувач вводить у текстовому полі команду copy A B.
2. UI передає команду інтерпретатору (Interpreter).
3. Інтерпретатор аналізує текст і формує об’єкт ParsedCommand.
4. UI передає ParsedCommand виконавцю команд (CommandExecutor у ShellContext).
5. Виконавець перевіряє:

    чи доступний сервер:

        якщо так - команда серіалізується в JSON і відправляється на TCP сервер;

        якщо ні - копіювання виконується локально на машині.
6. Після виконання результат заноситься у SQLite через DatabaseLogger.
7. UI відображає користувачеві повідомлення про успіх/помилку.
8. Вікно оновлює список файлів у поточному каталозі.

Висновок: У ході лабораторної роботи я реалізував частину програми описану у попередній роботі, а також побудував діаграми розгортання, компонентів та послідовностей системи.

Питання до лабораторної роботи:

1. Що собою становить діаграма розгортання?

Діаграма розгортання (Deployment diagram) — це UML-діаграма, яка показує фізичне розміщення програмних компонентів на апаратних вузлах (сервери, комп’ютери, пристрої) та зв’язки між ними.

2. Які бувають види вузлів на діаграмі розгортання?

Вузли поділяються на:

1. Апаратні вузли (device nodes) — фізичні пристрої: сервер, ПК, смартфон, маршрутизатор.
2. Віртуальні / програмні вузли (execution environment) — середовища виконання: JVM, контейнер Docker, ОС, сервер додатків.

3. Які бувають зв’язки на діаграмі розгортання?

Основні зв’язки:

1) Communication path — канал зв’язку між вузлами.
2) Deployment — відношення, що показує, який компонент розгорнуто на якому вузлі.

4. Які елементи присутні на діаграмі компонентів?

На діаграмі компонентів використовують:

1) Компоненти (component)
2) Інтерфейси (provided/required interfaces)
3) Порти (ports)
4) Артефакти
5) Залежності між компонентами

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

Зв’язки вказують:

1) Який компонент використовує інший (dependency)
2) Який інтерфейс реалізується або надається компонентом
3) Як компоненти взаємодіють через порти та інтерфейси

6. Які бувають види діаграм взаємодії?

UML визначає два основних типи діаграм взаємодії:

1. Діаграма послідовностей (Sequence diagrams)
2. Діаграма комунікації (Communication diagrams)

Інколи до взаємодії також відносять діаграму таймінгів і взаємодіючих оглядів (interaction overview).

7. Для чого призначена діаграма послідовностей?

Діаграма послідовностей описує часовий порядок обміну повідомленнями між об’єктами при виконанні конкретного сценарію або функціональності системи.

8. Які ключові елементи можуть бути на діаграмі послідовностей?

1) Актори
2) Об’єкти / екземпляри класів
3) Лінії життя (lifelines)
4) Повідомлення (messages)
5) Активності (activation bars)
6) Фрагменти управління (alt, loop, opt, par тощо)
7) Створення і знищення об’єктів

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

1) Кожен варіант використання зазвичай має одну або кілька діаграм послідовностей, які детально описують, *як саме* реалізується цей use case.
2) Use case — це загальний опис функції.
3) Sequence diagram — деталізація кроків виконання цієї функції.

10. Як діаграми послідовностей пов’язані з діаграмами класів?
    1) На діаграмі послідовностей використовуються об’єкти, які є екземплярами класів, описаних у діаграмі класів.
    2) Повідомлення між об’єктами відповідають викликам методів класів.
    3) Таким чином sequence-діаграма показує динамічну поведінку, а діаграма класів — статичну структуру.