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 «*Технології розроблення програмного забезпечення*»
Тема: «Діаграма розгортання. Діаграма компонентів.
Діаграма взаємодій та послідовностей**»**

Виконав:
Студент групи ІА-24
Котлярчук М. С.

Перевірив:
Мягкий М. Ю.

Київ-2024

ЗМІСТ

Тема: Powershell terminal (strategy, command, abstract factory, bridge, interpreter, client-server).
Термінал для powershell повинен нагадувати типовий термінал з можливістю налаштування кольорів синтаксичних конструкцій, розміру вікна, фону вікна, а також виконання команд powershell і виконуваних файлів, а також працювати в декількох вікнах терміналу (у вкладках або одночасно шляхом розділення вікна).

Короткі теоретичні відомості

Діаграма розгортання

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).

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. Найчастіше використовують логічне розбиття на компоненти - у такому разі проєктовану систему віртуально уявляють як набір самостійних, автономних модулів (компонентів), що взаємодіють між собою. Компоненти можуть поділятися за фізичними одиницями - окремі вузли розподіленої системи - набір комп'ютерів і серверів; на кожному з вузлів можуть бути встановлені різні виконувані 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.

Діаграма послідовностей

Для моделювання процесу виконання операцій у мові UML використовуються діаграми послідовностей. Застосовувана в них графічна нотація багато в чому схожа на нотацію діаграми станів, оскільки на цих діаграмах також присутні позначення станів і переходів. Кожен стан на діаграмі послідовностей

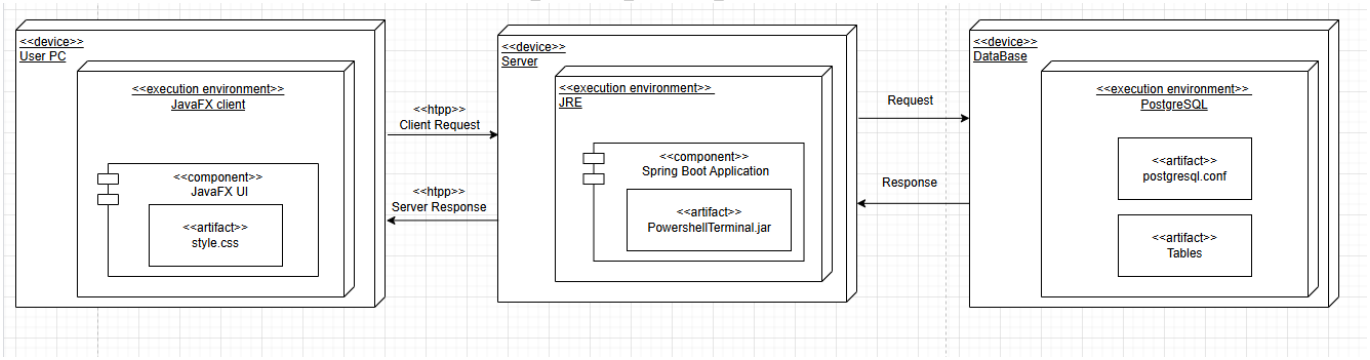
відповідає виконанню деякої елементарної операції, а перехід у наступний стан виконується тільки при завершенні цієї операції.

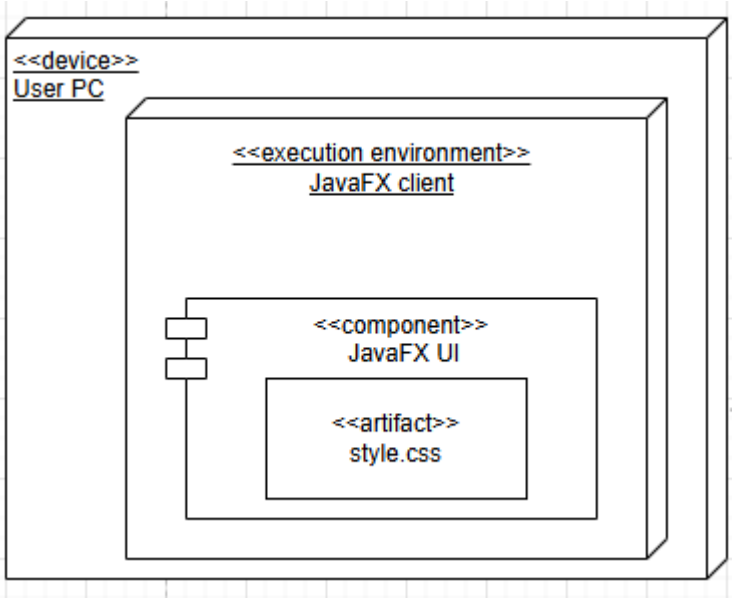
Завдання

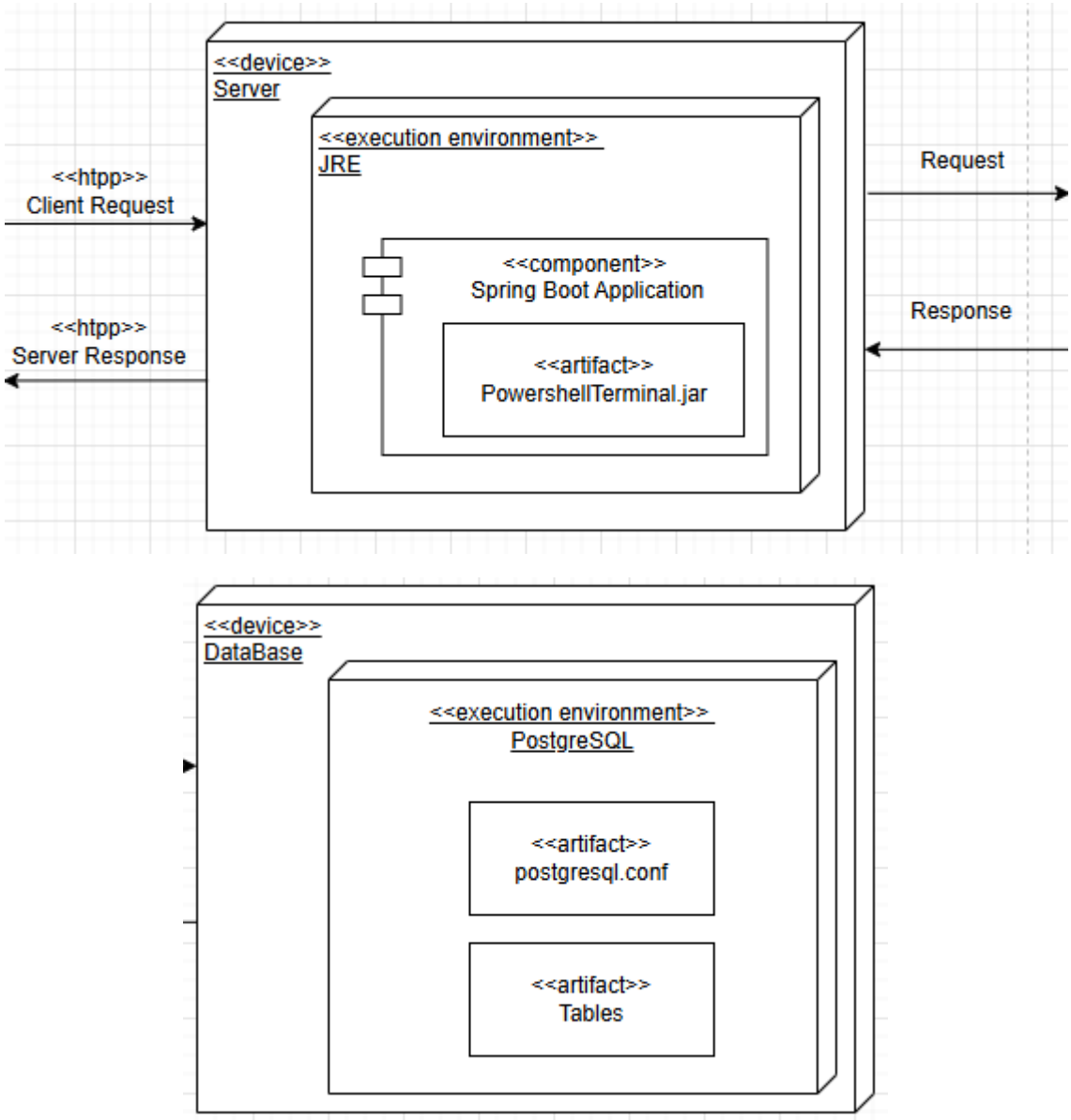
1. Ознайомитися з короткими теоретичними відомостями.
2. Розробити діаграму розгортання для проєктованої системи.
3. Розробити діаграму компонентів для проєктованої системи.
4. Розробити діаграму послідовностей для проєктованої системи.
5. Скласти звіт про виконану роботу.

Хід роботи

Діаграма розгортання







1. User PC (<<device>>):

- Це пристрій користувача, з якого здійснюється доступ до PowerShell терміналу.
- JavaFX Client (<<execution environment>>): середовище виконання JavaFX на клієнтському пристрої, яке відповідає за запуск інтерфейсу користувача.
- JavaFX UI (<<component>>): це компонент інтерфейсу користувача, який представляє графічний інтерфейс терміналу. Він надає користувачеві можливість взаємодіяти з терміналом — вводити команди, змінювати налаштування (кольори, розмір, фон) і переглядати результати виконання команд.
- style.css (<<artifact>>): артефакт, що представляє файл стилів CSS, використовується для налаштування зовнішнього вигляду інтерфейсу терміналу, зокрема стилів для синтаксичних конструкцій, фону, кольору тексту тощо.

2. Server (<<device>>):

- Це основний сервер, де виконуються всі серверні обчислення, обробляються команди терміналу і надсилаються налаштування.
- JRE (<<execution environment>>): Java Runtime Environment (JRE), необхідне для запуску застосунків на Java. Це середовище виконання, яке підтримує роботу Spring Boot на сервері, що забезпечує обробку запитів і виконання команд.
- Spring Boot Application (<<component>>): основний компонент, який обробляє запити клієнта. У рамках цього компонента відбувається обробка команд, надісланих із клієнта, виконується серверна логіка та комунікація з базою даних.
- PowershellTerminal.jar (<<artifact>>): зібраний артефакт застосунку у вигляді JAR-файлу, що містить весь код і залежності для запуску серверної частини терміналу. Цей файл розгорнутий на сервері й виконується в середовищі JRE.

3. DataBase (<<device>>):

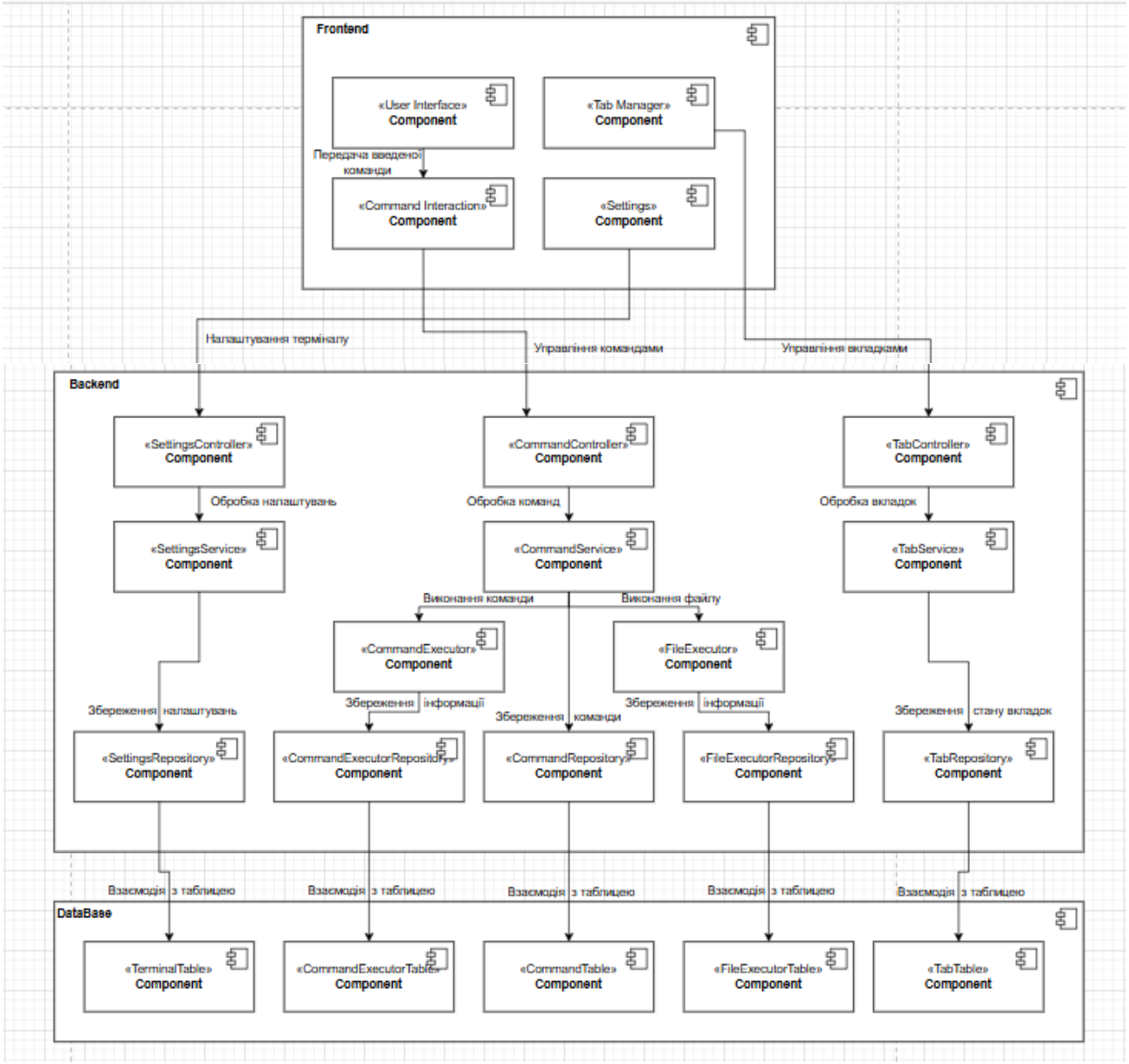
- Це сервер бази даних, який зберігає всі дані, необхідні для роботи PowerShell терміналу.
- PostgreSQL (<<execution environment>>): середовище виконання для бази даних PostgreSQL. Відповідає за обробку запитів на читання та запис даних.
- postgresql.conf (<<artifact>>): конфігураційний файл PostgreSQL, який містить параметри налаштування для бази даних, такі як параметри підключення, обсяг пам'яті, що використовується, і налаштування для оптимізації роботи бази даних.
- Tables (<<artifact>>): таблиці бази даних, у яких зберігаються дані про налаштування інтерфейсу, історію команд тощо. Цей артефакт відображає сховище, де зберігаються всі необхідні дані.

4. Зв'язки:

- Client Request (<<http>>) та Server Response (<<http>>): ці зв’язки представляють клієнт-серверну комунікацію між User PC і сервером.
    - Client Request: користувач із JavaFX UI відправляє запити на сервер (наприклад, для виконання команд або змін налаштувань).
    - Server Response: сервер обробляє запит, виконує потрібні команди або оновлення налаштувань і повертає результат або підтвердження виконання клієнту.
    - Використовується протокол HTTP або HTTPS, залежно від потреби у безпеці зв'язку.
- Request та Response між сервером і базою даних:

- Request: сервер надсилає запити до бази даних для отримання, збереження або оновлення даних (наприклад, зберігання налаштувань інтерфейсу, запис історії команд).
- Response: база даних відповідає на запити сервера, повертаючи результати запитів або підтвердження успішного запису даних.

Діаграма компонентів



Frontend (JavaFX):

- User Interface (UI): відповідає за графічний інтерфейс терміналу. Включає компоненти для відображення вікна, вкладок, налаштувань кольору, фону та розміру вікна.
- Settings Manager: керує налаштуваннями терміналу, такими як кольори синтаксису, фон, розмір вікна, і зберігає їх для майбутніх сесій.

- Tab Manager: обробляє відкриття, перемикання та видалення вкладок у терміналі, підтримує можливість одночасної роботи з декількома сесіями.
- Command Interaction Component: цей компонент відповідає за взаємодію користувача з терміналом, пов'язану з введенням та отриманням результатів команд.

Backend (Серверна частина):

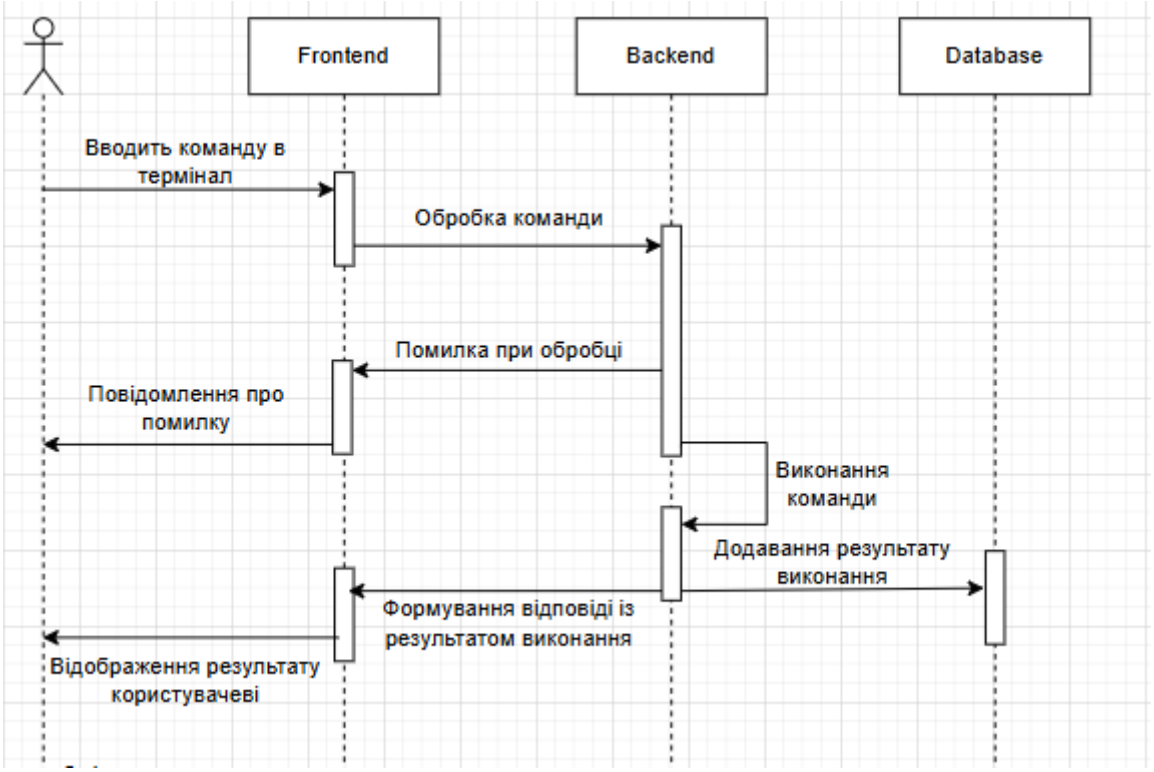
- CommandController Component – обробляє запити на виконання команд від клієнта, перевіряє їх на коректність і передає для виконання.
- SettingsController Component – обробляє запити щодо налаштувань терміналу, отримує та передає налаштування в сервіс налаштувань.
- TabController Component – обробляє запити, пов'язані з управлінням вкладками, забезпечує можливість зберігання сесій вкладок.
- CommandService Component – реалізує бізнес-логіку для виконання команд, перевіряє команди на коректність, виконує команди і повертає результати.
- CommandExecutor Component – обробляє виконання команд і зберігає результати в таблиці CommandExecutor.
- FileExecutor Component – відповідає за виконання команд, що стосуються файлів, і зберігає результати у таблиці FileExecutor.
- TabService Component – керує сесіями вкладок, обробляє додавання, видалення та збереження стану вкладок.
- SettingsService Component - реалізує бізнес-логіку налаштування параметрів терміналу, працює з TerminalRepository для збереження і оновлення даних налаштувань у таблиці Terminal.
- CommandRepository Component - забезпечує CRUD-операції для команд.
- CommandExecutorRepository Component - зберігає результати виконання команд, логи, повідомлення про помилки.
- FileExecutorRepository Component - зберігає результати виконання команд, пов’язаних з файлами, логи, повідомлення про помилки.
- TabRepository Component - зберігає інформацію про вкладки, їхній статус і команди, що виконувались у вкладках.
- SettingsRepository Component - забезпечує збереження налаштувань терміналу, включаючи розмір вікна, колір синтаксису, фон, список відкритих вкладок.

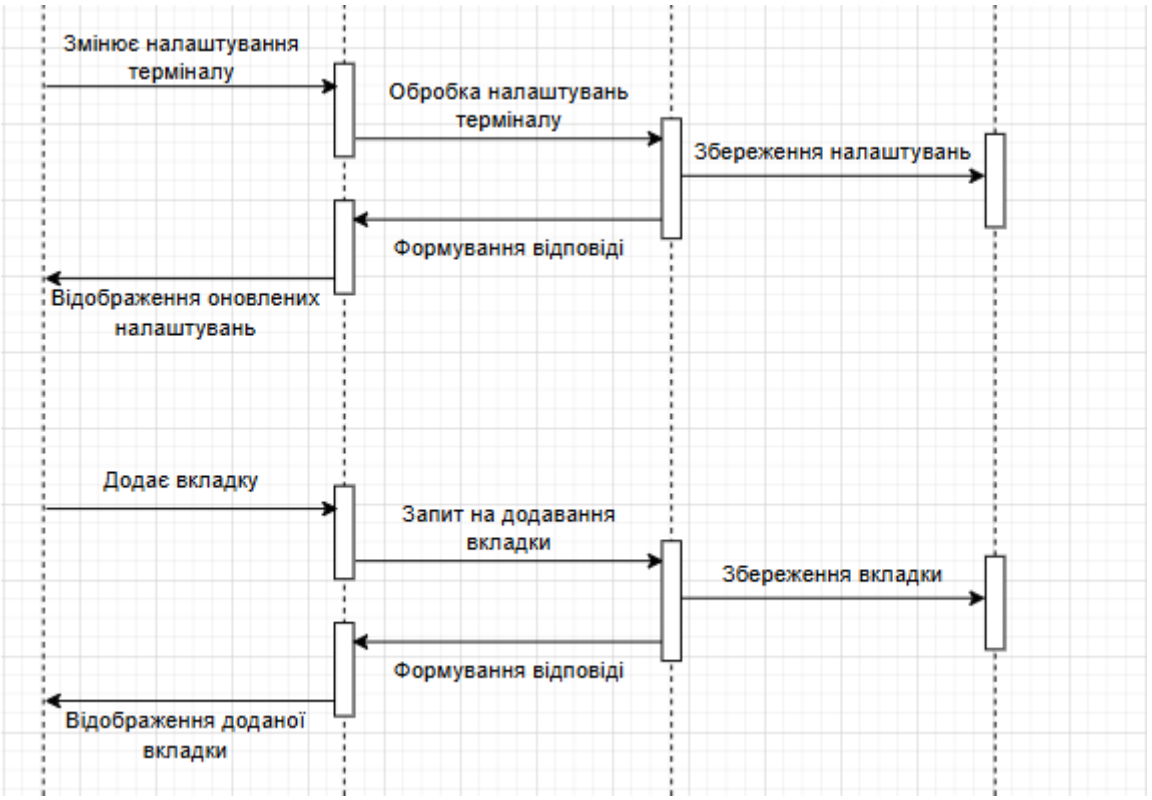
Таблиці:

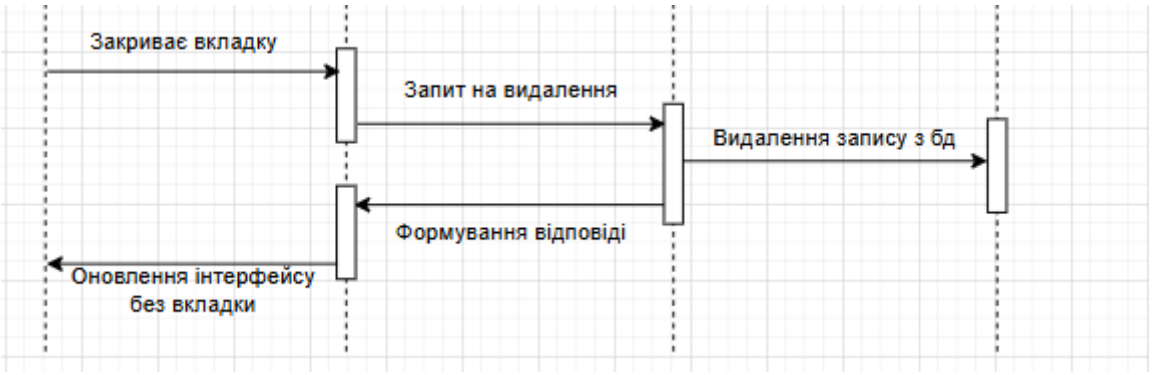
- CommandTable Component – таблиця для зберігання інформації про команди, включаючи їх назву, статус успішності виконання, опис та вказівку на те, чи є команда PowerShell.

- CommandExecutorTable Component – таблиця для зберігання інформації про виконання команд, зокрема результати виконання, текст помилок та лог виконання команд.
- FileExecutorTable Component – таблиця для збереження інформації про виконані файли, включаючи шлях до файлу, тип файлу, розмір, команду, пов’язану з файлом, та результат виконання.
- TabTable Component – таблиця, яка містить дані про вкладки, зокрема унікальний ідентифікатор, назву вкладки, статус активності та список команд, виконаних у межах кожної вкладки.
- TerminalTable Component – таблиця для зберігання налаштувань термінала, включаючи кольори синтаксису, розміри вікна, фон, а також список вкладок і команд, що використовуються в терміналі, з обмеженням на максимальну кількість вкладок.

Діаграма послідовностей







Виконання команди PowerShell:

1. Користувач вводить команду в інтерфейсі терміналу.
2. Frontend отримує команду та передає її на Backend для обробки.
3. Помилка при обробці (необов'язковий етап): Backend перевіряє команду на валідність і, якщо виникає помилка, надсилає інформацію про помилку назад на Frontend.
4. Повідомлення про помилку (необов'язковий етап): Frontend відображає користувачеві повідомлення про помилку, якщо команда не може бути виконана.
5. Якщо команда валідна, Backend починає виконання команди.
6. Після виконання команди Backend зберігає результат в базу даних.
7. Backend формує відповідь з результатом виконання команди та надсилає її на Frontend.

8. Frontend отримує відповідь та відображає результат виконання команди користувачеві.

Збереження налаштувань терміналу:

1. Користувач змінює налаштування терміналу (кольори, фон).
2. Frontend збирає оновлені налаштування та формує запит до Бекенду.
3. Бекенд обробляє та зберігає нові налаштування терміналу в таблиці Terminal в базі даних.
4. Фронтенд відображає оновленні налаштування.

Створення вкладки:

1. Користувач додає нову вкладку.
2. Фронтенд надсилає запит на додавання до Бекенду.
3. Бекенд створює вкладку та додає її до бази даних через TabRepository.
4. Фронтенд відображає вкладку користувачеві.

Видалення вкладки:

1. Користувач закриває вкладку.
2. Фронтенд надсилає запит на видалення до Бекенду.
3. Бекенд видаляє відповідний запис з бази даних.
4. Фронтенд оновлює інтерфейс користувача, закриваючи вкладку.

Висновки:

У ході виконання лабораторної роботи я дослідив та застосував на практиці різні типи діаграм UML для моделювання системи PowerShell терміналу. Зокрема, я створив діаграму розгортання, що відображає фізичне розміщення елементів системи, діаграму компонентів, яка описує структуру програми на рівні логічних модулів, та діаграму послідовностей, яка ілюструє взаємодію між компонентами під час виконання ключових операцій, таких як введення команд, зміна налаштувань та управління вкладками. Створені діаграми допомогли краще зрозуміти архітектуру та взаємодію компонентів проєктованої системи, що є важливим для подальшої розробки та підтримки застосунку.