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 №2

із дисципліни «*Технології розроблення програмного забезпечення*»
Тема: «Діаграма варіантів використання. Сценарії варіантів використання. Діаграми UML. Діаграми класів. Концептуальна

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 прецедентів

Діаграма прецедентів (use case diagram) — це один із типів діаграм в UML, який використовується для моделювання взаємодії між користувачами (акторами) і системою, що проєктується. Вона описує функціональні можливості системи з точки зору кінцевого користувача, показуючи, які завдання (прецеденти) користувачі можуть виконувати, і як вони взаємодіють із системою.

Діаграми варіантів використання призначені для:

1. Визначення загальної межі функціональності проектованої системи;
2. Сформулювати загальні вимоги до функціональної поведінки проектованої системи.
3. Розробка вихідної концептуальної моделі системи;
4. Створення основи для виконання аналізу, проектування, розробки та тестування.

Діаграма класів

Діаграма класів — це статична діаграма в UML, яка відображає структуру системи, моделюючи класи, їхні властивості (атрибути) та поведінку (методи), а також відносини між класами. Вона слугує для візуалізації об'єктно-орієнтованої моделі, показуючи, як різні частини системи взаємодіють на рівні об'єктів, дозволяючи детально проаналізувати архітектуру програмного забезпечення.

База даних та її структура

База даних — це організований набір даних, які зберігаються в структурованому вигляді, зазвичай у вигляді таблиць. Таблиці в базі даних складаються з рядків (записів) і стовпців (полів), що містять атрибути даних. Структура бази даних визначає, як дані взаємопов'язані між собою. Основними елементами є таблиці, ключі (первинні та зовнішні) і зв'язки між таблицями (один-до-одного, один-до-багатьох, багато-до-багатьох).

Сценарії використання

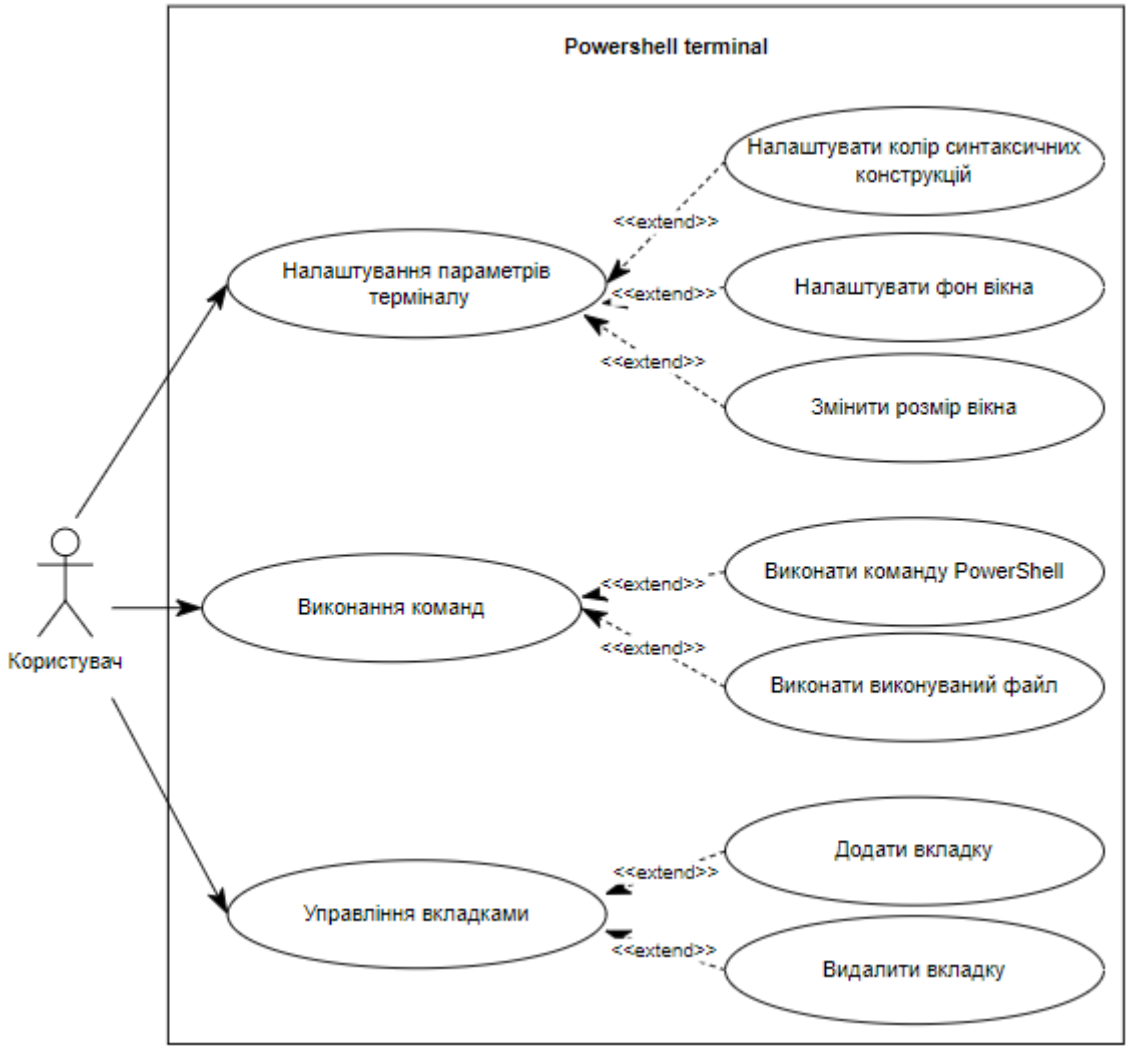
Сценарії використання - це текстові уявлення тих процесів, які відбуваються під час взаємодії користувачів системи та самої системи. Вони є чітко формалізованими, покроковими інструкціями, що описують той чи інший процес у термінах кроків досягнення мети. Сценарії використання однозначно визначають кінцевий результат.

Завдання

1. Ознайомитися з короткими теоретичними відомостями.
2. Проаналізуйте тему та намалюйте схему прецеденту, що відповідає обраній темі лабораторії.
3. Намалюйте діаграму класів для реалізованої частини системи.
4. Виберіть 3 прецеденти і напишіть на їх основі сценарії прецедентів.
5. Розробити основні класи і структуру системи баз даних.
6. Класи даних повинні реалізувати шаблон Репозиторію для взаємодії з базою даних.
7. Підготувати звіт про хід виконання лабораторних робіт. Звіт, що подається повинен містити: діаграму прецедентів, діаграму класів системи, вихідні коди класів системи, а також зображення структури бази даних.

Хід роботи

Діаграма прецедентів



Прецедент 1: *Налаштування параметрів терміналу*

Передумови: Користувач відкрив термінал.

Постумови: Параметри терміналу (колір, фон, розмір) налаштовані відповідно до вибору користувача.

Короткий опис: Користувач змінює налаштування кольору синтаксису, фону вікна та розміру вікна.

Основний хід подій:

1. Користувач вибирає опцію "Налаштування параметрів терміналу".
2. Система пропонує налаштувати колір синтаксису, фон вікна та розмір.
3. Користувач вказує потрібні значення для одного або декількох параметрів.
4. Система зберігає та застосовує нові налаштування.

Винятки:

- Якщо користувач не зберігає зміни, система скасовує всі налаштування і повертається до попередніх.

- Якщо введені дані некоректні, система повідомляє про помилку і повертається до попередніх налаштувань.

Примітки:

- Параметри можна змінити в будь-який момент без необхідності перезапуску терміналу.

Прецедент 2: *Виконання команди PowerShell*

Передумови: Користувач має доступ до терміналу і ввів команду PowerShell.

Постумови: Вказана команда PowerShell успішно виконана або система показала повідомлення про помилку.

Короткий опис: Користувач вводить команду PowerShell для виконання в терміналі.

Основний хід подій:

1. Користувач вводить команду PowerShell у терміналі.
2. Система перевіряє коректність команди.
3. Команда виконується, і результат виводиться на екран.
4. Система показує результат або повідомлення про помилку у випадку невдалої спроби.

Винятки:

- Якщо команда містить помилки, система виводить відповідне повідомлення із зазначенням помилки.

Прецедент 3*: Управління вкладками*

Передумови: Користувач працює з терміналом і має відкриту вкладку.

Постумови:

- Додана або видалена вкладка у терміналі.

Короткий опис: Користувач може додавати або видаляти вкладки для одночасної роботи з декількома сесіями терміналу.

Основний хід подій:

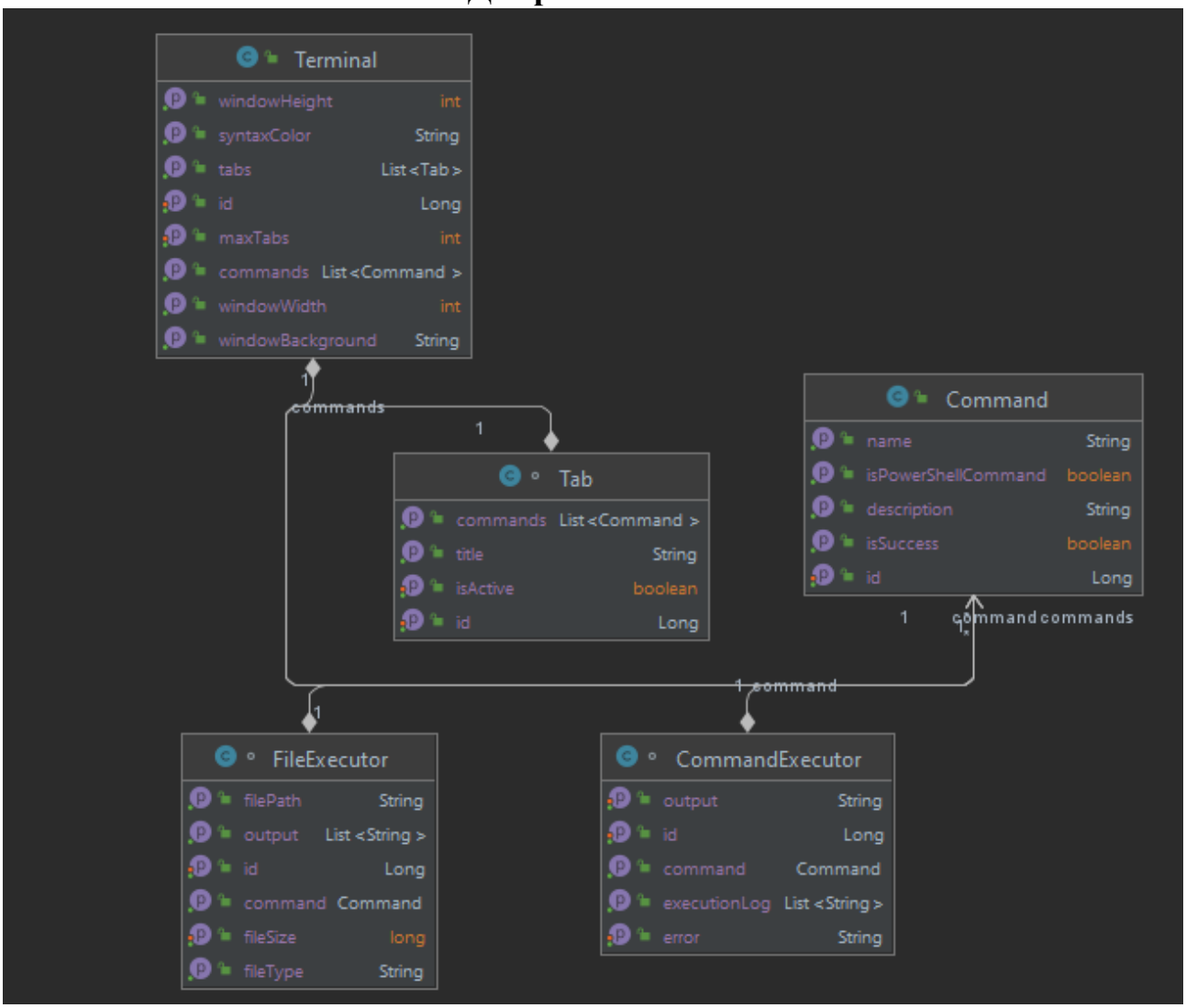
1. Користувач вибирає опцію "Додати вкладку".
2. Система відкриває нову вкладку в терміналі.
3. Користувач може переключатися між вкладками для роботи з різними сесіями.
4. Користувач може видалити вкладку, якщо вона більше не потрібна.

Винятки: Відсутні.

Примітки:

- Кількість вкладок може бути обмежено системними ресурсами.

Діаграма класів



Command (Команда)

- ID: Унікальний ідентифікатор команди.

- name: Назва команди.
- isPowerShellCommand: Вказує, чи є команда командою PowerShell.
- isSuccess: Вказує на успіх виконання команди.
- description: Опис команди.

CommandExecutor (Виконавець команди)

- ID: Унікальний ідентифікатор.
- command: Об'єкт класу Command, що представляє виконувану команду.
- output: Результат виконання команди (виведений текст).
- error: Текст помилки, якщо виконання завершилось з помилкою.
- executionLog: Лог виконання команди.

FileExecutor (Виконавець файлів)

- ID: Унікальний ідентифікатор.
- filePath: Шлях до файлу, який виконується.
- fileType: Тип файлу.
- command: Команда, пов'язана з файлом.
- fileSize: Розмір файлу.
- output: Список результатів виконання файлу.

Tab (Вкладка)

- ID: Унікальний ідентифікатор вкладки.
- title: Назва вкладки.
- isActive: Вказує, чи є вкладка активною в поточний момент.
- commands: Список команд, що були виконані у вкладці.

Terminal (Термінал)

- ID: Унікальний ідентифікатор термінала.
- syntaxColor: Колір синтаксису, що використовується в терміналі.
- windowWidth: Ширина вікна термінала.
- windowHeight: Висота вікна термінала.
- windowBackground: Фон вікна термінала.

- commands: Список усіх виконаних команд у терміналі.
- tabs: Список вкладок, що відкриті в терміналі.
- maxTabs: Максимальна кількість вкладок, яку можна відкрити одночасно.

Зв'язки між сутностями:

- Terminal має зв'язок один до багатьох з Tab (один термінал може мати кілька вкладок). Кожна вкладка є частиною терміналу, в якому вона відкривається.

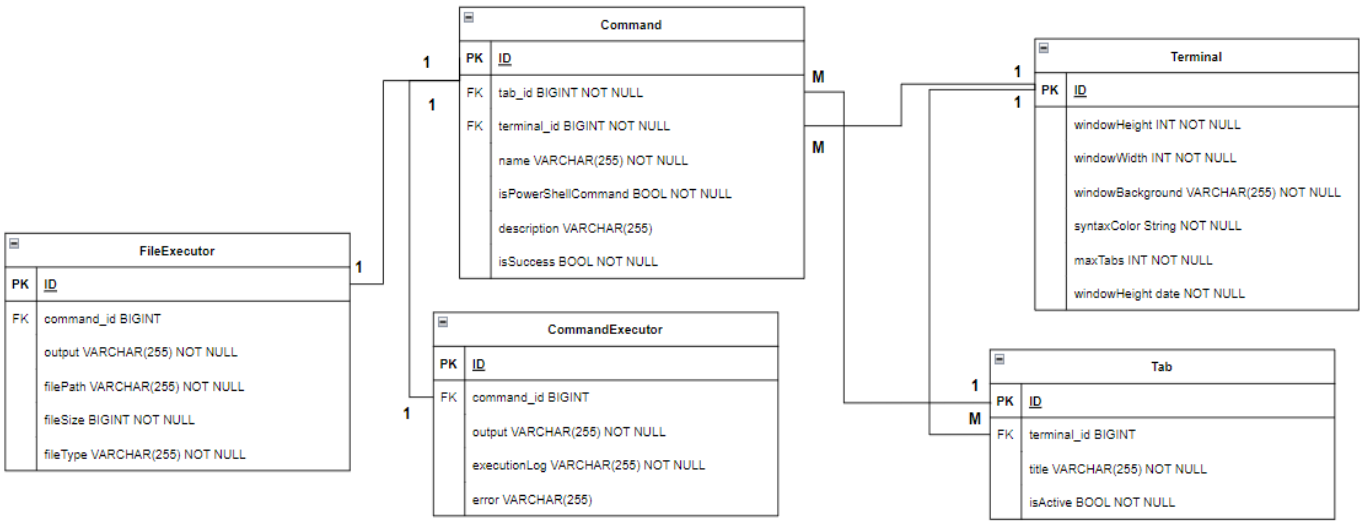
- Tab має зв'язок один до багатьох з Command (одна вкладка може мати кілька виконаних команд). Кожна вкладка містить історію команд, які виконувались у межах цієї вкладки.

- CommandExecutor має зв'язок один до одного з Command (один виконавець команди відповідає за виконання однієї команди). Виконавець команди обробляє і зберігає результат виконання відповідної команди.

- FileExecutor має зв'язок один до одного з Command (один виконавець файлів виконує одну команду, пов'язану з файлом). Виконання файлу може бути ініційоване командою.

- Terminal має зв'язок один до багатьох з Command (один термінал може мати кілька команд, виконаних загалом у різних вкладках). Це зберігає загальну історію виконання команд у терміналі.

Структура бази даних



Висновки:

Під час виконання даної лабораторної роботи я навчився моделювати програмні системи з використанням діаграм прецедентів, які описують взаємодію користувача із системою та діаграм класів, що відображають зв'язки між сутностями бази даних. Також створив модель структури бази даних моєї системи.