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: «Діаграма розгортання. Діаграма компонентів.
Діаграма взаємодій та послідовностей»

Варіант №18

Виконав:
студент групи ІА-24
Гуменюк К.Е.

Перевірив:
Мягкий Ю. М.

Київ 2025

Зміст

Мета.

Розробка та аналіз концептуальної моделі для обраного варіанту з використанням UML-діаграм. Зокрема створення діаграми розгортання, компонентів, взаємодій та послідовностей.

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.

Обрана тема.

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.

Короткі теоретичні відомості

1. Діаграма розгортання (Deployment Diagram)

Діаграми розгортання є важливим інструментом для візуалізації фізичного розміщення програмних компонентів у системі. Вони показують, як програмне забезпечення розгорнуте на апаратному забезпеченні, і які зв'язки існують між різними частинами системи. Основні елементи діаграми включають:

Вузли (Nodes): Це фізичні або віртуальні пристрої, які можуть містити програмне забезпечення. Вузли поділяються на:

Пристрої (Devices): Фізичні елементи, такі як сервери, комп'ютери, маршрутизатори. Вони представляють апаратне забезпечення, на якому запускається програмне забезпечення.

Середовища виконання (Execution Environments): Програмні платформи, такі як операційні системи або сервери додатків, які можуть містити інші програмні компоненти. Вони забезпечують середовище для виконання програмного забезпечення.

Зв'язки (Connections): Визначають, як вузли взаємодіють між собою, зазвичай через мережеві протоколи або інші технології зв'язку. Зв'язки можуть мати атрибути, такі як назва протоколу (наприклад, HTTP, IPC) або технології (наприклад, .NET Remoting, WCF).

Артефакти (Artifacts): Файли або інші фізичні прояви програмного забезпечення, такі як виконувані файли, бібліотеки, конфігураційні файли. Вони представляють програмне забезпечення, яке розгортається на вузлах.

Діаграми розгортання можуть бути описовими, без конкретних деталей про обладнання, або екземплярними, з конкретними деталями про апаратне забезпечення та програмне забезпечення. Описові діаграми корисні на ранніх етапах проектування, тоді як екземплярні діаграми використовуються на завершальних стадіях розробки.

2. Діаграма компонентів (Component Diagram)

Діаграми компонентів описують структуру системи через її модулі або компоненти. Вони допомагають зрозуміти, як різні частини системи взаємодіють одна з одною. Основні види діаграм компонентів:

Логічні: Відображають систему як набір автономних модулів, які взаємодіють між собою. Це допомагає візуалізувати архітектуру системи на концептуальному рівні. Кожен компонент може бути взаємозамінним і не обов'язково знаходиться в межах одного фізичного пристрою.

Фізичні: Показують, як компоненти розподілені між різними фізичними вузлами системи. Цей підхід застарів і зазвичай замінюється діаграмами розгортання.

Виконувані: Кожен компонент представляє собою файл або набір файлів, які можуть бути виконані, такі як .exe, бібліотеки або HTML-сторінки. Це дозволяє візуалізувати систему на рівні виконуваних файлів або процесів.

Діаграми компонентів допомагають візуалізувати загальну структуру коду, специфікувати виконувані варіанти системи та забезпечити повторне використання коду. Вони також можуть включати інтерфейси та схеми баз даних для більш детального уявлення про систему.

3. Діаграма взаємодій та послідовностей (Interaction and Sequence Diagrams)

Діаграми послідовностей використовуються для моделювання динамічної поведінки системи, показуючи, як об'єкти взаємодіють один з одним у певній послідовності. Вони допомагають зрозуміти, як різні частини системи взаємодіють у часі. Основні елементи включають:

Стан дії (Action State): Відображає виконання окремих дій або кроків алгоритму. Кожна дія має вхідний і вихідний перехід. Стан дії зазвичай моделює один крок виконання алгоритму або потоку управління.

Переходи: Нетриггерні переходи, які виконуються після завершення дії. Вони можуть бути умовними, з використанням сторожових умов. Переходи дозволяють моделювати розгалуження та паралельні процеси.

Дорожки (Swimlanes): Використовуються для розподілу дій між різними підрозділами організації, що дозволяє моделювати бізнес-процеси. Кожна

дорожка представляє окремий підрозділ або роль, відповідальну за виконання певних дій.

Діаграми взаємодій та послідовностей допомагають візуалізувати алгоритми виконання, потоки управління та бізнес-процеси, фокусуючи увагу на результатах і змінах стану системи. Вони є важливим інструментом для розуміння динамічної поведінки системи та взаємодії між її компонентами.

Діаграма розгортання

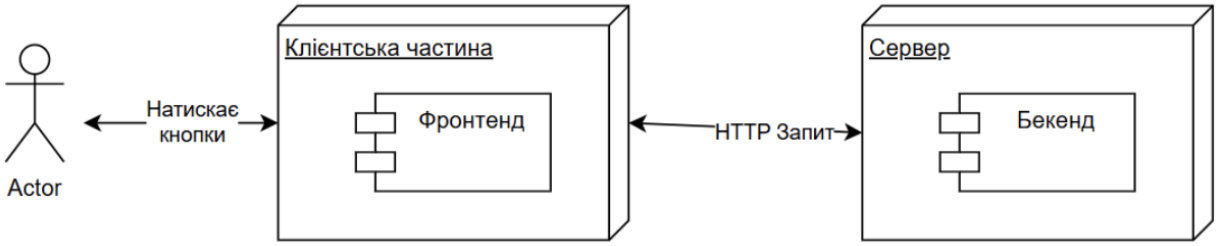

Рисунок 1. Діаграма розгортання

Архітектура системи розроблена таким чином, що взаємодія між клієнтською та серверною частинами відбувається через чітко визначені компоненти. Користувач взаємодіє з клієнтською частиною через вебінтерфейс, вводячи дані, переглядаючи їх результат та натискаючи відповідні кнопки. Клієнтська частина включає в себе компонент Фронтенд, який відповідає за відображення користувацького інтерфейсу та передачу запитів на сервер. Взаємодія з сервером здійснюється через HTTP-запити.

Серверна частина складається з Бекенду, який обробляє запити від клієнта та виконує відповідні файлові операції. Бекенд отримує команди від фронтенду, виконує файлові дії (перегляд, копіювання, переміщення, видалення, пошук, перемикання дисків) і повертає результати клієнту. Ця архітектура дозволяє розвантажити клієнтську частину, переклавши всю складну обробку запитів на сервер, що забезпечує стабільну та швидку роботу системи. Такий підхід дозволяє ефективно організувати взаємодію між компонентами та забезпечує надійну обробку запитів користувача

Опис діаграми:

1) Клієнт у браузері натискає на кнопки для взаємоії з певними елементами інтерфейсу .
2) Клієнтська частина формує HTTP запит після чого надсилає його до серверної частини
3) Сервер отримує запит і оброблює його згідно внутрішній логіці

Діаграма компонентів

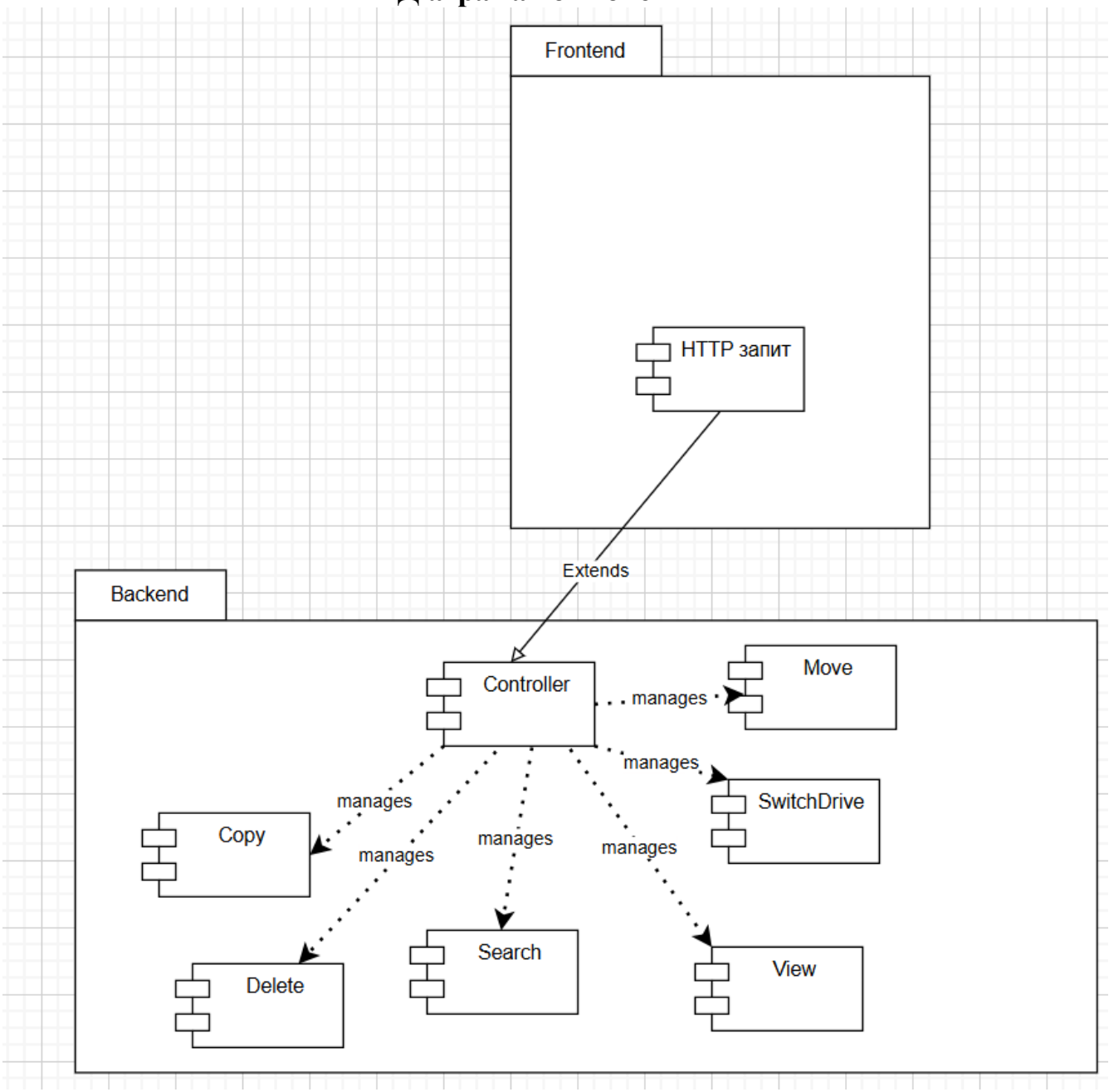

Рисунок 2. Діаграма компонентів

Основні компоненти:

1. Frontend: Користувач виконує певні дії за допомогою клієнтського інтерфейсу

2. Backend

   a) Controller – приймає запит від клієнту, оброблює його та направляє немобхідній частині коду

      a. Дії над файлами(копіювання, пощук, видалення, переміщення,

перегляд, зміна дисків) – окремі частини для реалізації конкретного функціоналу.

Взаємозв'язки між компонентами:

1. Use – зв'язок між користувачем та процесами входу і реєстрації.
2. Extends – зв'язок між функціями, що розширюють можливості роботи з файлами. Компонент "Робота з файлами" використовує розширені функції "File Manager" для видалення, перейменування та копіювання файлів.
3. Manages – зв'язки, що вказують на те, що "Обробник команд" керує діями, пов'язаними зі зміною дисків, пошуком та роботою з файлами.

Діаграма взаємодій та послідовностей

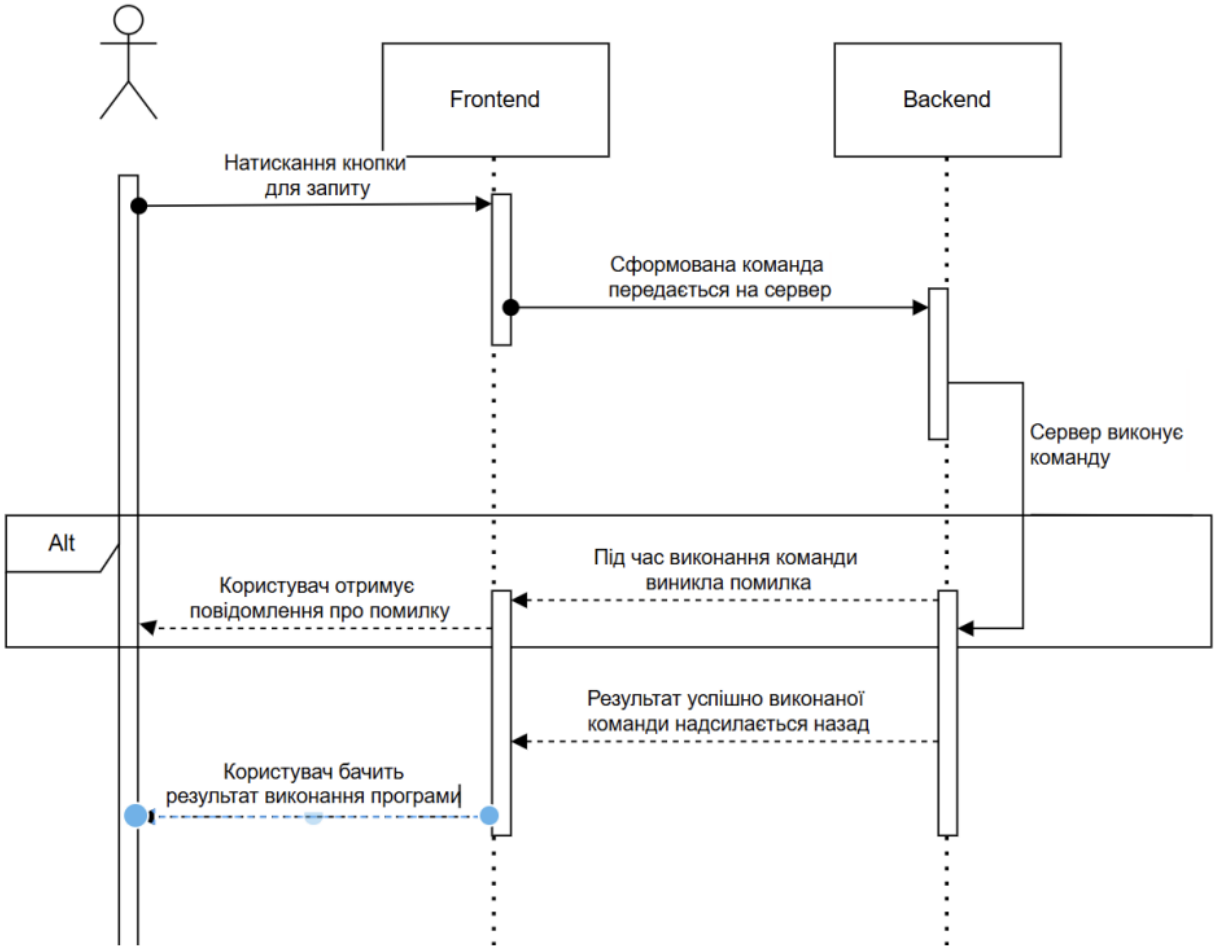

Рисунок 3. Діаграма взаємодій та послідовностей

Учасники:

1. Користувач – взаємодіє з системою для виконання операцій над файлами та папками.
2. Frontend – інтерфейс, через який користувач відправляє запити.
3. Backend – сервер який оброюлює запити реалізуючи функціонал застосунку.

Основні сценарії:

1. Перегляд файлів та папок:
    a. Користувач ініціює перегляд файлів та папок.
    b. Клієнтська частина надсилає запит до сервера.
    c. Сервер виконує поставлену задачау.
    d. Сервер повертає інформацію стосовно результатів виконання запиту.
2. Видалення файлу:
    a. Користувач ініціює видалення файлу або папки.
    b. Клієнтська частина надсилає запит до сервера.
    c. Сервер виконує поставлену задачау.
    d. Сервер повертає інформацію стосовно результатів виконання запиту.
3. Пошук файлів:
    a. Користувач ініціює пошук файлу у папці.
    b. Клієнтська частина надсилає запит до сервера.
    c. Сервер виконує поставлену задачау.
    d. Сервер повертає інформацію стосовно результатів виконання запиту.

Аналогічним чином працюють й інші дії

Висновки.

Запропонована система управління файлами та доступу до бази даних побудована на багаторівневій модульній архітектурі, що забезпечує зручність, ефективність і гнучкість.

1. **Модульність**: Система складається з окремих компонентів (Auth, Shell, File Manager, Server, Database), кожен із яких виконує чітко визначені функції. Це спрощує її підтримку, розширення та інтеграцію нових функцій.
2. **Чітка взаємодія**: Компоненти системи мають зрозумілі ролі у процесі обміну даними. Запити користувачів через Shell обробляються сервером і передаються до бази даних, як показано на діаграмах послідовності та розгортання.
3. **Гнучкість і масштабованість**: Завдяки чіткій структурі система легко адаптується до змін і може бути доповнена новими модулями чи функціями без порушення основної архітектури.

Ця архітектура поєднує безпеку, функціональність і зручність, забезпечуючи надійну основу для сучасних додатків із керування даними.

.