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

Лабораторна робота №3

з дисципліни «Технології розроблення програмного забезпечення»

Виконала:
студентка групи ІА-23
Проценко. В. І.

Перевірив:
Мягкий М. Ю

Київ 2024

Зміст

Тема: Діаграма розгортання. Діаграма компонентів. Діаграма 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.

1. Короткі теоретичні відомості

UML (Unified Modeling Language) є стандартною мовою для візуалізації, документування, побудови та аналізу систем. Вона широко використовується для створення моделей програмних систем, оскільки допомагає чітко структурувати процеси та зв’язки між компонентами. Основними типами діаграм, які ми розглянемо у даній лобораторній роботі це:

1. Діаграма розгортання (англ. deployment diagram) — діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент. Вона використовується для відображення архітектури системи на рівні розподілу компонентів між серверами, базами даних та іншими фізичними ресурсами. Це корисно для планування розгортання системи, особливо для великих проектів із розподіленою архітектурою.
2. Діаграма компонент (англ. Component diagram) — в UML, діаграма, на якій відображаються компоненти, залежності та зв'язки між ними. Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись. Вона дозволяє зрозуміти, як різні частини системи взаємодіють між собою. Діаграма компонентів корисна для планування архітектури системи та розподілу функцій між модулями.

Компонент — логічний блок системи, за рівнем абстракції трохи вищий, ніж «клас». Зображується прямокутником з зображенням вкладки у правому верхньому куті. Компоненти об'єднуються, разом використовуючи структурні зв'язки (англ. assembly connector), щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер». Інтерфейс постачання (англ. Provide interface) — набір відкритих атрибутів та операцій, які повинні бути надані класами, що реалізують даний інтерфейс. Інтерфейс вимоги (англ. Required interface) — набір відкритих атрибутів та операцій, які вимагаються класами, що залежать від даного інтерфейсу. Структурна взаємодія — «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту.»

При використанні діаграми компонент, щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

3. Діаграма послідовностей — ілюструє порядок обміну повідомленнями між різними об'єктами для виконання певного сценарію. Вона показує, як компоненти взаємодіють між собою з часом для реалізації конкретної функції або процесу. Ця діаграма є важливою для розуміння динамічної поведінки системи та допомагає ідентифікувати, які об'єкти беруть участь у кожному процесі.

   На діаграмі послідовності паралельними вертикальними лініями ("лініями життя" - англ. lifelines) показано різні процеси або об'єкти, які існують одночасно, а горизонтальними стрілками - повідомлення, якими вони обмінюються між собою, у порядку їх виникнення. Це дозволяє специфікувати прості сценарії виконання у графічній формі.

   Діаграма послідовності системи повинна визначати і показувати наступне:

   - Зовнішні актори
   - Об'єкти, які самі викликають методи
   - Повідомлення (методи), що викликаються цими акторами
   - Значення, що повертаються (якщо такі є), пов'язані з попередніми повідомленнями
   - Вказівка на будь-які цикли або області ітерацій

Дані діаграми використовуються для документування та моделювання системи моніторингу комп'ютерної активності, де кожен компонент відповідає за певну функцію, наприклад, збір даних про використання процесора, пам'яті, введення з клавіатури та миші, а також взаємодія з базою даних для зберігання історичних даних і створення звітів. Завдяки UML-діаграмам можна краще організувати розробку, обслуговування та тестування системи.

2. Хід роботи

Пригадаємо обрану індивідуальну тему для виконання лабораторних робіт.

..17 System activity monitor (iterator, command, abstract factory, bridge, visitor, SOA)

Монітор активності системи повинен зберігати і запам'ятовувати статистику використовуваних компонентів системи, включаючи навантаження на процесор, обсяг займаної оперативної пам'яті, натискання клавіш на клавіатурі, дії миші (переміщення, натискання), відкриття вікон і зміна вікон; будувати звіти про використання комп'ютера за різними критеріями (% часу перебування у веб-браузері, середнє навантаження на процесор по годинах, середній час роботи комп'ютера по днях і т.д.); правильно поводитися з «простоюванням» системи – відсутністю користувача.

2.1. Діаграма розгортання для проектованої системи.

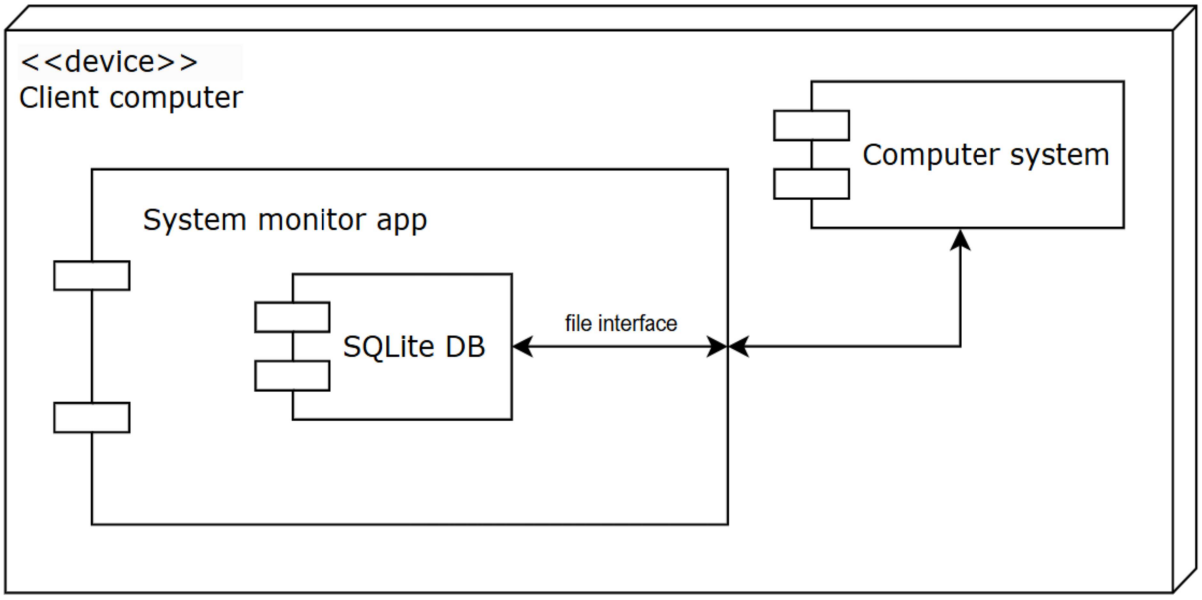

Рисунок 1.1 - Діаграма розгортання

Діаграма розгортання демонструє фізичну архітектуру системи моніторингу активності комп’ютера та розміщення її компонентів на клієнтському комп’ютері. Ця діаграма зображає, як різні компоненти взаємодіють між собою та з базою даних для зберігання інформації.

Компоненти та зв’язки:

- Client Computer.
  Це основний фізичний вузол, на якому працює система моніторингу. Він представляє комп’ютер користувача, де встановлені всі компоненти системи.
- System Monitor App.
  Основний компонент, який відповідає за відстеження та збір даних про використання комп’ютера(CPU, пам'ять, вікна тощо).
- Computer System.
  Представляє ресурси комп’ютера (процесор, пам’ять, відкриті вікна тощо), до яких звертається система моніторингу для збору інформації. Він забезпечує доступ до низькорівневих даних про завантаженість системи та інші системні показники.
- SQLite DB.
  База даних SQLite є внутрішнім компонентом додатку System Monitor App. Вона використовується для локального зберігання зібраних даних. Така структура забезпечує зручний доступ до даних безпосередньо з додатку без потреби в зовнішній базі даних чи віддаленому сервері.

З’єднання між компонентами:

- System Monitor App взаємодіє з Computer System через внутрішній інтерфейс комп'ютера для отримання актуальних даних про роботу системи.
- SQLite DB, як частина System Monitor App, забезпечує збереження та зчитування даних, зібраних системою моніторингу.

Варіант подальшого розширення:

Якщо у майбутньому виникне потреба масштабувати систему або розгортати її у хмарі, можна завжди повернутися до ідеї контейнеризації бази даних, але для простого локального рішення цей варіант не є обов'язковим.

2.2. Діаграма компонентів для проектованої системи.

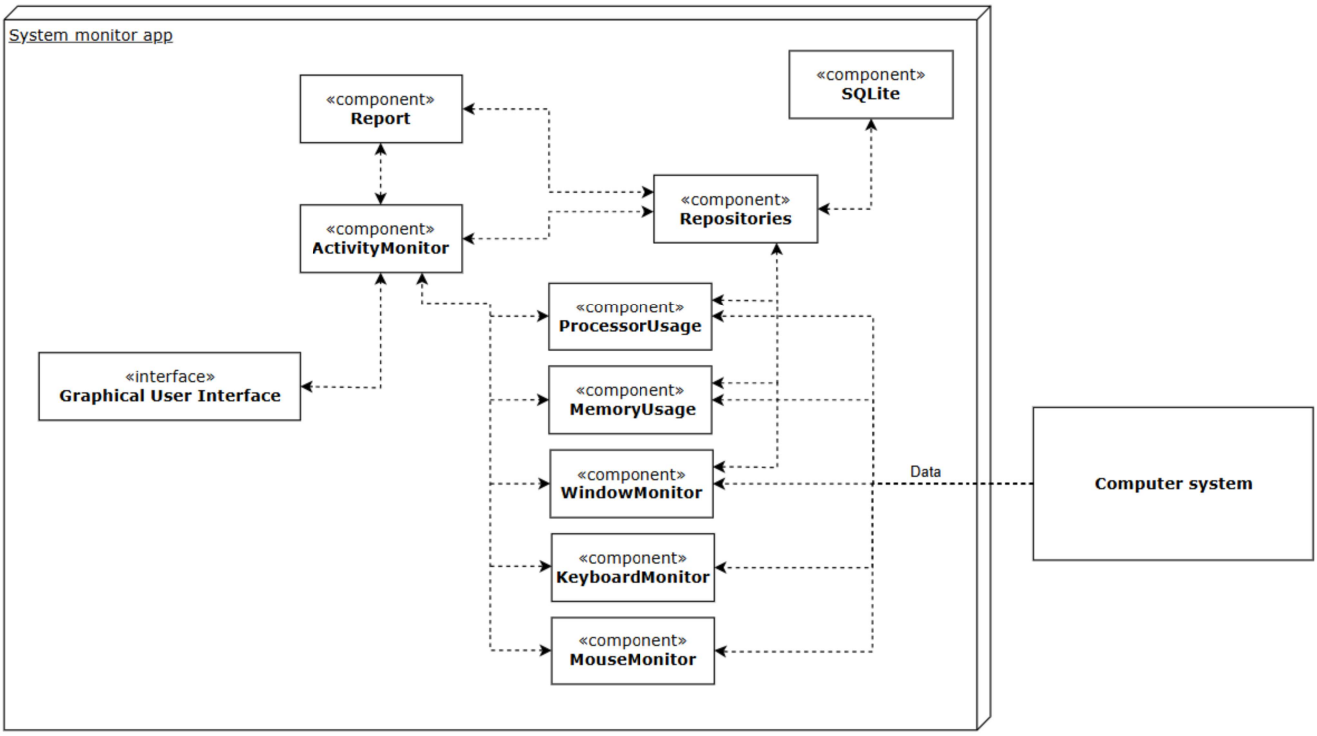

Рисунок 1.2 - Діаграма компонентів

Діаграма компонентів ілюструє архітектуру системи моніторингу активності комп'ютера. Ця діаграма відображає, як кожен компонент системи взаємодіє для досягнення загальної мети моніторингу і звітування про активність комп'ютера. Основні компоненти системи розташовані в рамках "System monitor app" та взаємодіють між собою для забезпечення збору даних, обробки і збереження інформації, а також для взаємодії з користувачем через графічний інтерфейс.

Компоненти діаграми:

- ActivityMonitor.
  Це основний компонент, який координує роботу системи.
  Він отримує дані від моніторів (ProcessorUsage, MemoryUsage, WindowMonitor, KeyboardMonitor, MouseMonitor) і заємодіє з Graphical User Interface для відображення даних у реальному часі. А також через компонент Repositories може отримати доступ до історичних даних із бази даних.
- ProcessorUsage.
  Відстежує навантаження на процесор, передає зібрані дані в ActivityMonitor для подальшої обробки, а також через відповідний репозиторій зберігає дані у базу даних.

- MemoryUsage.
  Відстежує використання оперативної пам’яті системи, передає дані в ActivityMonitor, а також через відповідний репозиторій зберігає дані у базу даних.
- WindowMonitor.
  Відстежує активні вікна та тривалість їхнього використання, передає зібрані дані в ActivityMonitor, а також через відповідний репозиторій зберігає дані у базу даних.
- KeyboardMonitor.
  Відстежує активність клавіатури, наприклад, натискання клавіш і передає дані в ActivityMonitor для подальшої обробки.
- MouseMonitor.
  Відстежує активність миші, наприклад, переміщення курсору та кліки
  Та інформує ActivityMonitor про зміну стану.
- Repositories.
  Компонент, що забезпечує інтерфейс для збереження та отримання даних у SQLite. Містить конкретні репозиторії, що зберігають дані від окремих моніторів (ProcessorUsage, MemoryUsage, WindowMonitor тощо).Взаємодіє з базою даних SQLite.
- SQLite.
  Локальна база даних, що зберігає історичні дані про активність системи. Використовує Repositories для збору інформації яку збирає система моніторингу та передачі інформації на основний монітор для створення звітів.
- Report.
  Компонент для генерації звітів на основі зібраних даних. Взаємодіє з ActivityMonitor і Repositories для отримання необхідної інформації.
- Graphical User Interface.
  Інтерфейс, що дозволяє користувачеві взаємодіяти із системою. Відображає поточні дані моніторингу в реальному часі. Дозволяє користувачеві переглядати історичні дані і ініціювати генерування звітів.
- Computer System.
  Джерело даних, що надає інформацію про стан системи, яка моніториться. Взаємодіє з компонентами моніторингу (ProcessorUsage, MemoryUsage тощо), передаючи їм необхідну інформацію.

2.3. Діаграма послідовностей для проектованої системи.

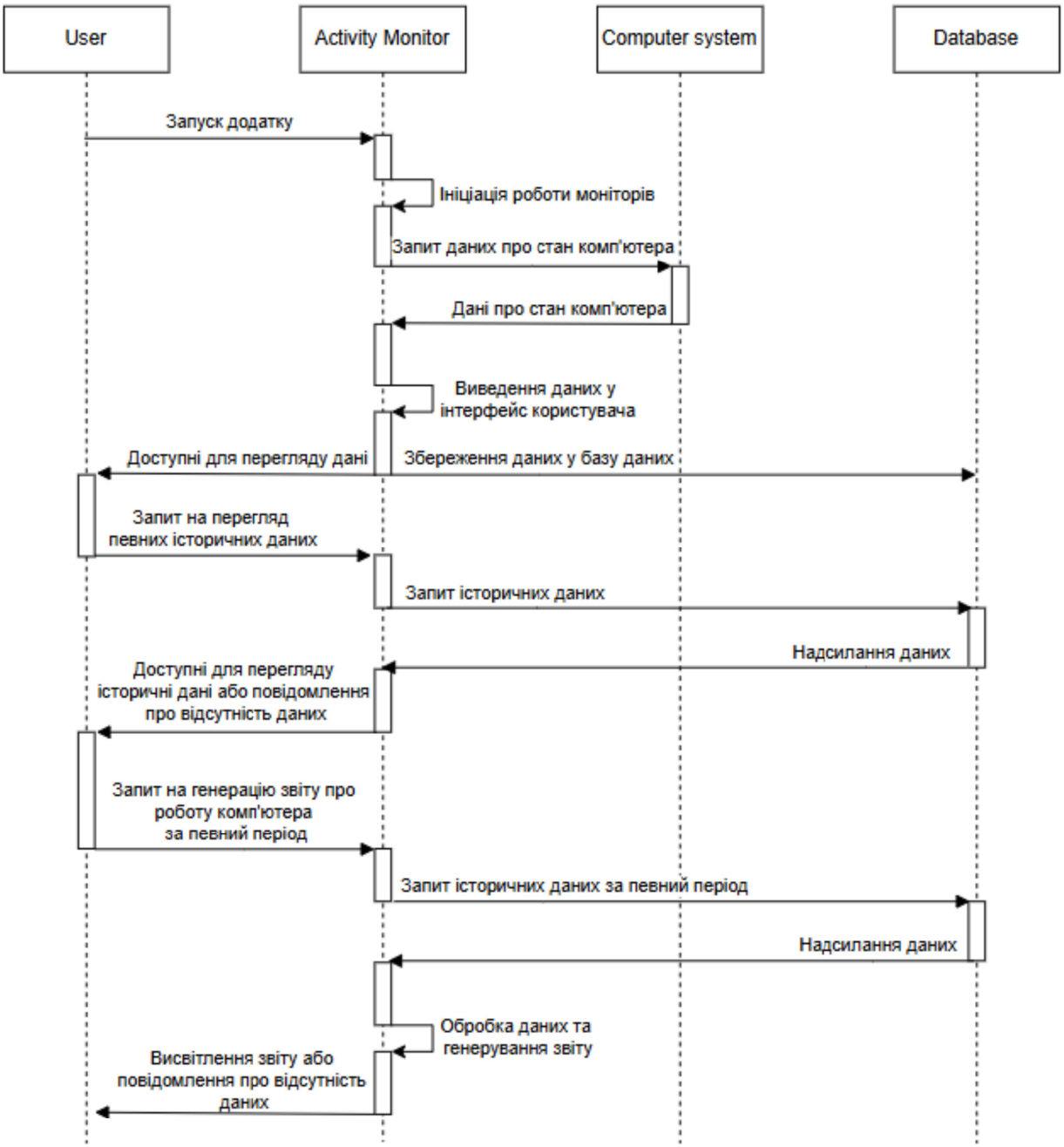

Рисунок 1.3 - Діаграма послідовностей

Діаграма послідовностей ілюструє взаємодію між користувачем, системою моніторингу активності (Activity Monitor), комп'ютерною системою (Computer system) та базою даних (Database) для відображення даних про стан комп'ютера, збереження історичних даних і генерацію звітів.

Основні кроки на діаграмі:

1. Запуск додатку.
   Користувач ініціює запуск додатку моніторингу активності.
2. Ініціація роботи моніторів
   Після запуску додатка, Activity Monitor ініціює роботу моніторів та запитує дані про стан комп'ютера у Computer system.
3. Отримання даних про стан комп'ютера
   Computer system відповідає, надаючи дані про поточний стан комп'ютера, такі як навантаження процесора, використання пам'яті тощо. Activity Monitor виводить ці дані в інтерфейсі користувача для відображення в реальному часі. А також може відбутись збереження даних у базу даних для створення історичних записів.
4. Запит на перегляд історичних даних
   Користувач може запитати перегляд певних історичних даних через Activity Monitor.
5. Запит на отримання даних із бази даних.
   Activity Monitor надсилає запит на отримання історичних даних до Database.
6. Отримання історичних даних
   Database надсилає потрібні історичні дані у відповідь, які потім відображаються користувачеві через Activity Monitor. Якщо даних немає, система повідомляє користувача про їхню відсутність.
7. Запит на генерацію звіту
   Користувач може також запросити звіт про роботу комп'ютера за певний період.
8. Запит на отримання даних із бази даних.
   Activity Monitor надсилає запит до Database для отримання даних за вказаний період.
9. Обробка даних та генерація звіту
   Database надає потрібні дані для звіту а Activity Monitor обробляє дані, генерує звіт та відображає його користувачеві. Якщо дані за певний період відсутні, система повідомляє користувача.

Діаграма показує, як Activity Monitor взаємодіє з Computer system і Database для збору, збереження, перегляду та аналізу даних. Користувач має можливість як переглядати поточні дані в реальному часі, так і запитувати історичні дані або звіти, щоб отримати повну картину активності комп'ютера.

2.4. Частина функціональності системи.

Частина функціональності системи, описана діаграмами з попередньої лабораторної роботи згідно обраної теми монітор активності знаходиться у спільному репозиторії, у папці system-activity-monitor за посиланням: https://github.com/protsenkoveronika/TRPZ/tree/test-branch/system-activity-monitor

3. Висновок

У ході даної лабораторної роботи було розроблено та описано кілька UML-діаграм для моніторингу комп'ютерної активності. Було створено діаграми компонентів, розгортання та послідовностей. Діаграма розгортання ілюструє розподіл компонентів системи на рівні фізичних серверів і показує, як база даних SQLite може бути розміщена разом із додатком і не потребує додаткових хмарних серверів. Діаграма компонентів відображає структуру системи моніторингу, зокрема взаємодію між основними модулями. Діаграма послідовностей демонструє покрокову взаємодію користувача з системою, а також обробку та збереження даних у базі, що дає повне уявлення про процеси, які відбуваються під час запиту історичних даних і генерації звітів. Завдяки цій роботі вдалося візуалізувати основні процеси в системі моніторингу комп'ютерної активності, забезпечуючи базу для подальшої реалізації та тестування програми.