Звіт

Лабораторна робота №1d

“Враження від інструменту моделювання чи малювання діаграм”

Виконала студентка групи ІПС-22

Чергінець Юлія Дмитрівна

1. **Чому було обрано саме цей інструмент, а не аналоги?**

   Для виконання лабораторної роботи було обрано інструмент Mermaid.js, оскільки він реалізує підхід «діаграми як код», що дозволяє описувати структуру схем за допомогою текстового синтаксису. Цей вибір був зумовлений передусім можливістю зберігати файли діаграм у системі контролю версій Git, що дозволяє відстежувати зміни через diff, на відміну від бінарних файлів зображень у графічних редакторах на кшталт Draw.io або StarUML. Крім того, інструмент має нативну підтримку на GitHub, де діаграми рендеряться автоматично прямо у файлах документації, та зручні плагіни для VS Code.

2. **Наскільки просто та зрозуміло було отримати, встановити, налаштувати та почати використовувати цей інструмент?**

   Процес отримання та налаштування інструменту виявився максимально простим, адже це JavaScript-бібліотека, яка не потребує встановлення важкого програмного забезпечення. Для початку роботи достатньо було встановити розширення Markdown Preview Mermaid Support у редакторі коду VS Code або скористатися онлайн-редактором Mermaid Live Editor для швидких експериментів, що дозволило розпочати створення діаграм миттєво без складних конфігурацій.

3. **Наскільки зрозумілою та корисною була документація інструмент?**

   Офіційна документація інструменту є доволі вичерпною для основних типів діаграм, таких як діаграми класів, послідовності чи блок-схем, і містить багато корисних прикладів коду, які легко адаптувати під свої потреби. Проте для вирішення специфічних задач, наприклад, стилізації вкладених пакетів або налаштування вигляду окремих зв'язків, документації іноді не вистачало, і доводилося шукати додаткові рішення на форумах спільноти або експериментувати з синтаксисом самостійно.

4. **Наскільки було зрозуміло, як саме використовувати інструмент, які функції/засоби/вікна/елементи керування використовувати для вирішення поставлених задач?**

   Логіка використання інструменту була інтуїтивно зрозумілою завдяки декларативному синтаксису, де структура коду безпосередньо відповідає логічній структурі діаграми, тому додаткові вікна чи елементи керування були не потрібні. Робота з інструментом зводилася до написання текстового опису сутностей та зв'язків між ними, що для розробника є природним процесом, схожим на написання звичайного програмного коду.

5. **Чи всі 14 типів діаграм з UML 2.5 підтримує інструмент? Якщо ні – вкажіть, які саме не підтримуються**

   Mermaid.js активно підтримує не всі 14 типів діаграм згідно зі специфікацією UML 2.5, забезпечуючи повноцінну роботу лише з найбільш популярними типами, такими як діаграми класів, станів, послідовності та варіантів використання. Деякі діаграми, наприклад, діаграми компонентів, розгортання чи комунікації, доводиться емулювати через розширені можливості блок-схем, а підтримка діаграм профілів чи композитних структур наразі є слабкою або відсутньою, що вимагає використання обхідних шляхів для їх реалізації.

6. **Чи всі можливості, доступні на кожному типі діаграм, підтримує інструмент? Якщо ні – вкажіть, що саме не підтримується, та для якого типу діаграм.**

   Не всі можливості стандарту UML доступні на кожному типі діаграм, зокрема, найбільшим обмеженням є автоматичне розташування елементів, яке не дозволяє користувачу точно задавати координати блоків або гнучко налаштувати стиль окремих елементів. Наприклад, у діаграмах класів складно керувати розміщенням методів та полів, а в діаграмах послідовності обмежені можливості для відображення складних фрагментів комбінування з нестандартними умовами виконання.

7. **Чи використовували якісь додаткові можливості інструменту, наприклад генерацію коду з діаграм чи відновлення діаграм з коду? Наскільки гарно та правильно працюють ці можливості?**

   У роботі використовувався підхід відновлення діаграм з існуючого коду вручну, а також генерація синтаксису Mermaid на основі JS-класів. Ця можливість працювала коректно та значно економила час розробки, дозволяючи автоматично створювати основу для складних схем, яку потім залишалося лише трохи відкоригувати для кращої читабельності.

8. **Наскільки зручно було використовувати інструмент, чи не треба було виконувати багато надлишкових дій?**

   Інструмент виявився дуже зручним у використанні для розробника, оскільки не вимагає виконання зайвих дій з ручного вирівнювання стрілок чи блоків мишкою, що є типовим для графічних редакторів. Рефакторинг схем, наприклад зміна назв класів або типів зв'язків, займав лічені секунди завдяки можливості простого редагування тексту, що значно пришвидшувало процес внесення змін.

9. **Наскільки зрозумілою була поведінка інструменту в різних ситуаціях? Чи не виникали ситуації, коли незрозуміло, чому були виконані якісь дії чи як досягли певного стану?**

   Поведінка інструменту була переважно передбачуваною та детермінованою, оскільки той самий код завжди генерував однакове зображення, що є важливою перевагою підходу Diagram as Code. Однак іноді виникали ситуації, коли алгоритм автоматичної маршрутизації проводив зв'язки неоптимальним або заплутаним шляхом через інші блоки, і виправити це було складно без зміни порядку рядків у коді або додавання прихованих зв'язків.

10. **Чи виникали якісь проблеми з використанням інструменту? Чи вдалось їх вирішити, як саме?**

    Під час роботи виникали технічні проблеми, наприклад, помилка рендерингу при використанні вкладених просторів імен у діаграмі пакетів, яку вдалося вирішити шляхом зміни типу діаграми на блок-схему з використанням підграфів для імітації пакетів. Також для коректного відображення часових діаграм довелося перейти на новий нативний синтаксис timingDiagram, оскільки старий підхід через діаграму Ганта виглядав некоректно та нагадував таблицю замість графіка зміни станів.

11. **Що хорошого можна сказати про цей інструмент, які були позитивні аспекти використання інструменту?**

    До позитивних аспектів використання можна віднести високу швидкість створення схем, яка порівнена зі швидкістю набору звичайного тексту, та естетичний вигляд діаграм за замовчуванням без додаткових зусиль. Також великою перевагою є можливість зберігання діаграм разом із кодом проекту в репозиторії, що спрощує командну роботу та контроль версій документації.

12. **Що поганого можна сказати про цей інструмент, які були негативні аспекти використання інструменту?**

    Негативним моментом є відсутність повного контролю над візуальним розташуванням елементів, що іноді призводить до створення заплутаних схем при великій кількості зв'язків. Також обмеження синтаксису для деяких специфічних UML-нотацій змушує шукати обхідні шляхи або спрощувати діаграми, що не завжди прийнятно для детального технічного проектування.

13. **Якби довелось вирішувати аналогічну задачу, але вже враховуючи досвід використання в цій лабораторній роботі, що варто було б зробити так само, а що змінити? Можливо, використати інший інструмент, чи використати інші можливості цього інструменту, чи інакше організувати процес розробки діаграм, чи ще щось?**

    Враховуючи набутий досвід, при вирішенні аналогічної задачі в майбутньому доцільно продовжувати використовувати Mermaid для діаграм, що тісно пов'язані з кодом, таких як діаграми класів та послідовності, оскільки вони ідеально лягають на текстовий опис. Водночас для високорівневих архітектурних схем, де важлива візуальна композиція та розташування блоків, варто розглянути використання графічних редакторів на кшталт Draw.io, а також від самого початку проекту організовувати окрему структуру папок для документації.