Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №3

Технології розроблення програмного забезпечення

«*Основи проектування розгортання.*»

«10. VCS all-in-one»

Виконав
студент групи ІА–32:
Варивода К. С.

Перевірив:

Мягкий Михайло Юрійович

Київ 2025

Зміст

Варіант

10. VCS all-in-one (iterator, adapter, factory method, facade, visitor, p2p)

Клієнт для всіх систем контролю версій повинен підтримувати основні команди і дії (commit, update, push, pull, fetch, list, log, patch, branch, merge, tag) для 3-х основних систем управління версіями (svn, git, mercurial), а також мати можливість вести реєстр репозиторіїв (і їх типів) і відображати дерева фіксації графічно.

Теоретичні відомості

Діаграми компонентів — показують модулі (компоненти), їхні залежності і артефакти (.jar, таблиці, html). Використовуються для логічного/фізичного поділу системи.

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.

Коли використовують логічне розбиття на компоненти, то у такому разі проєктовану систему віртуально уявляють як набір самостійних, автономних модулів (компонентів), що взаємодіють між собою.

Діаграми розгортання — показують фізичні вузли (devices) і середовища виконання (execution environments), на яких розгортаються артефакти;

зв’язки зазначають протоколи (HTTP, SQL/ODBC).

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).

Діаграми послідовностей — моделюють часову послідовність повідомлень між акторами/об’єктами (HTTP POST/GET: Browser – Server DB – Browser).

Діаграма послідовностей (Sequence Diagram) – це один із типів діаграм у моделюванні UML (Unified Modeling Language), який використовується для моделювання взаємодії між об'єктами системи у певній послідовності часу. Вона відображає, як об'єкти обмінюються повідомленнями, показуючи порядок і логіку виконання операцій. Діаграма складається з таких основних елементів:

Актори (Actors): Зазвичай позначаються піктограмами або назвами. Це користувачі чи інші системи, які взаємодіють із системою. Актори можуть бути

Об'єкти або класи: Розміщуються горизонтально на діаграмі. Вони позначаються прямокутниками з іменем об'єкта або класу під прямокутником. Кожен об'єкт має «життєвий цикл», який представлений вертикальною пунктирною лінією (лінія життя).

Повідомлення: Це лінії зі стрілками, які з'єднують об'єкти. Вони показують передачу повідомлень чи виклик методів. Стрілка може бути синхронною (звичайна стрілка) або асинхронною (лінія з відкритим трикутником) та з пунктирною лінією, що показує повернення результату.

Активності: Вказують періоди, протягом яких об'єкт виконує певну дію. На діаграмі це позначається прямокутником, накладеним на лінію життя.

Контрольні структури: Використовуються для відображення умов, циклів або альтернативних сценаріїв. Наприклад, блоки "alt" (альтернатива) або "loop" (цикл).

Хід роботи

Завдання

• Ознайомитись з короткими теоретичними відомостями.

• Проаналізувати діаграми створені в попередній лабораторній роботі а також тему системи та спроєктувати діаграму розгортання використання відповідно до обраної теми лабораторного циклу.

• Розробити діаграму компонентів для проєктованої системи.

• Розробити діаграму розгортання для проєктованої системи.

• Розробити як мінімум дві діаграми послідовностей для сценаріїв прописаних в попередній лабораторній роботі.

• На основі спроєктованих діаграм розгортання та компонентів доопрацювати програмну частину системи. Реалізація системи, додатково до попередньої реалізації, повинна містити як мінімум дві візуальні форми. В системі вже повинен бути повністю реалізована архітектура (повний цикл роботи з даними від вводу на формі до збереження їх в БД і подальшій виборці з БД та відображенням на UI).

• Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму розгортання з описом, діаграму компонентів системи з описом, діаграми послідовностей, а також вихідний код системи, який було додано в цій лабораторній роботі.

Діаграма розгортання системи

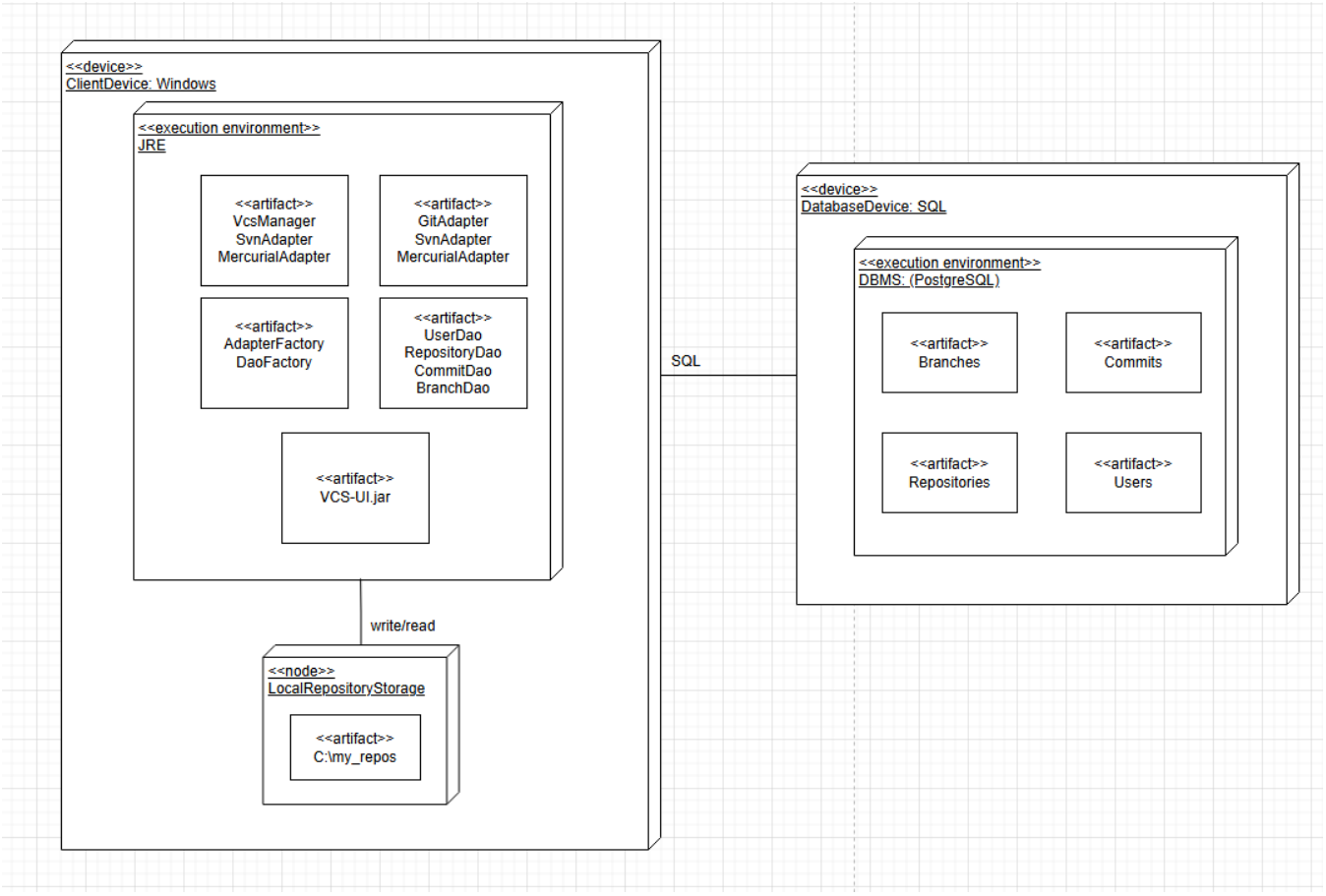

Рис 1. Діаграма розгортання системи

1. ClientDevice: Windows

- Це фізичний пристрій користувача
- Усередині нього розгорнуте середовище виконання JRE (<<execution environment>>).
- У JRE запускається артефакт "VCS-UI.jar", що відповідає за відображення графічного інтерфейсу користувача.
- Так як це монолітний застосунок, то у нас немає серверу і все відбувається в самому застосунку.
- Виконується JRE (<<execution environment>>), де розміщені всі «серверні» артефакти.
- VcsManager, SvnAdapter, GitAdapter, MercurialAdapter: Ключові компоненти бізнес-логіки.
- AdapterFactory, DaoFactory: Фабрики для створення адаптерів та об'єктів доступу до даних.
- UserDao, RepositoryDao, CommitDao, BranchDao: Артефакти, що реалізують логіку доступу до бази даних.
- Система розрахована на роботу з файлами, отже є модуль з підключенням файлової системи

2. DatabaseDevice: SQL

- Це сервер бази даних, що відповідає за збереження та обробку даних.
- Усередині нього виконується DBMS (PostgreSQL) як середовище виконання (<<execution environment>>).
- Розміщені артефакти - таблиці:
- Branches - таблиця для збереження гілок.
- Commits - таблиця для збереження комітів.
- Repositories - таблиця для збереження метаданих репозиторіїв.
- Users - таблиця для збереження інформації про користувачів.

3. Зв’язки

- ClientDevice - DatabaseDevice - взаємодія через SQL, коли серверні DAO (наприклад, CommitDao) звертаються до бази даних PostgreSQL для збереження чи отримання даних.
- ClientDevice – LocalRepositoryStorage – взаємодія через read/write, коли застосунок може взаємодіяти з файлами системи

Діаграма компонентів

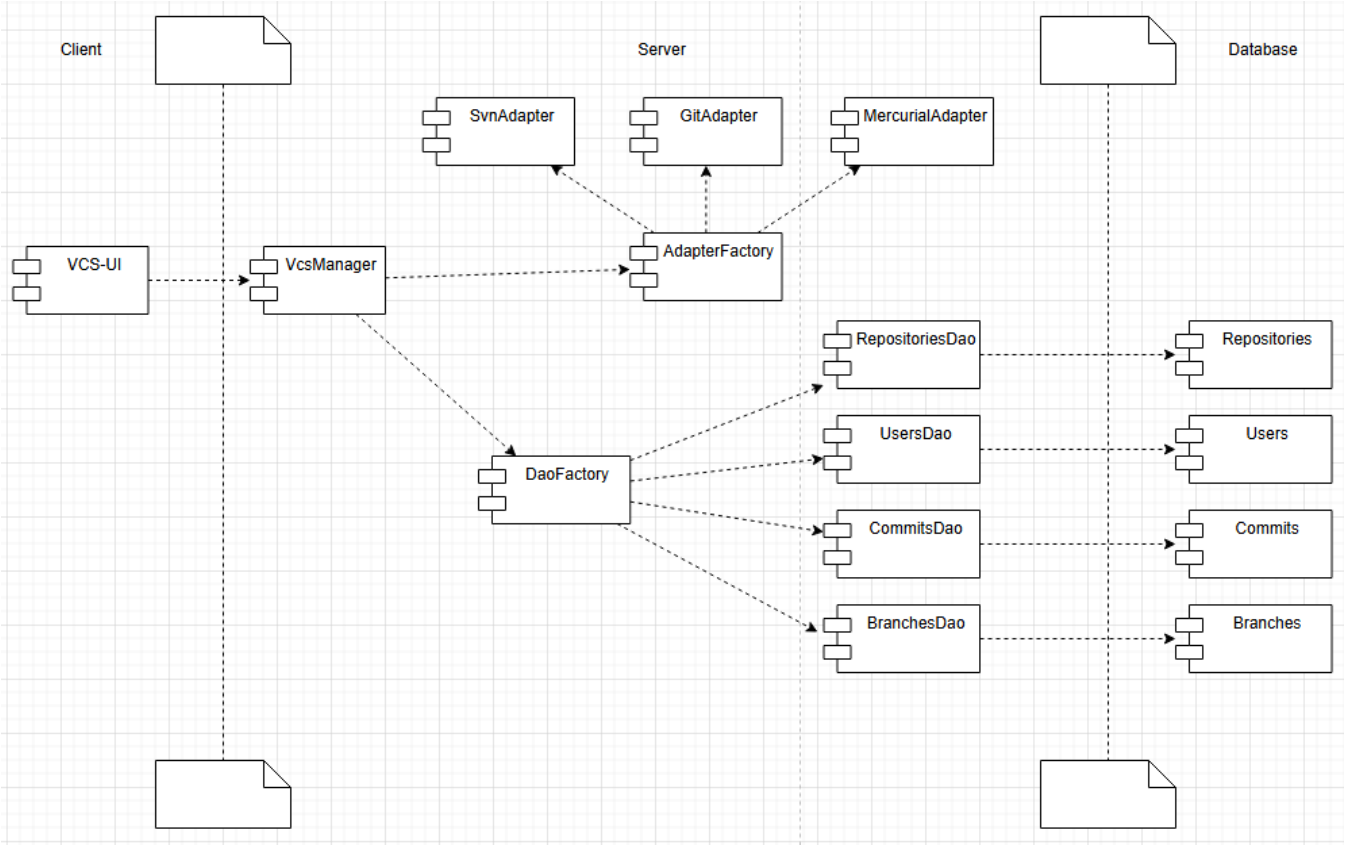

Рис 2. Діаграма компонентів

1. Client (Клієнтська частина):

    а. Компонент VCS-UI отримує інформацію та дії від користувача та передає їх на бекенд частину застосунку

2. Server (Серверна частина):

    а. VcsManager – Фасад для UI, який має в собі всі потрібні функції для візуальної частини

    б. AdapterFactory – Створення адаптерів для роботи з різними системами контролю

    в. DaoFactory – Створення DataAccessObjects з методами для роботи з базою даних

3. Database (База даних):

    а. Repositories – таблиця з інформацією про репозиторій

    б. Users – таблиця з інформацією про користувача

    в. Commits – таблиця з інформацією про коміти

г. Branches – таблиця з інформацією про гілки

4. Зв’язки:

а. VCS-UI взаємодіє через API всередині системи. VcsManager грає роль API

б. VcsManager створює та взаємодіє з factory для роботи з даними через DAO та операціями з системами контролю версій через адаптери

в. DAO об’єкти мають в собі методи для взаємодії з базою даних

Діаграма послідовностей

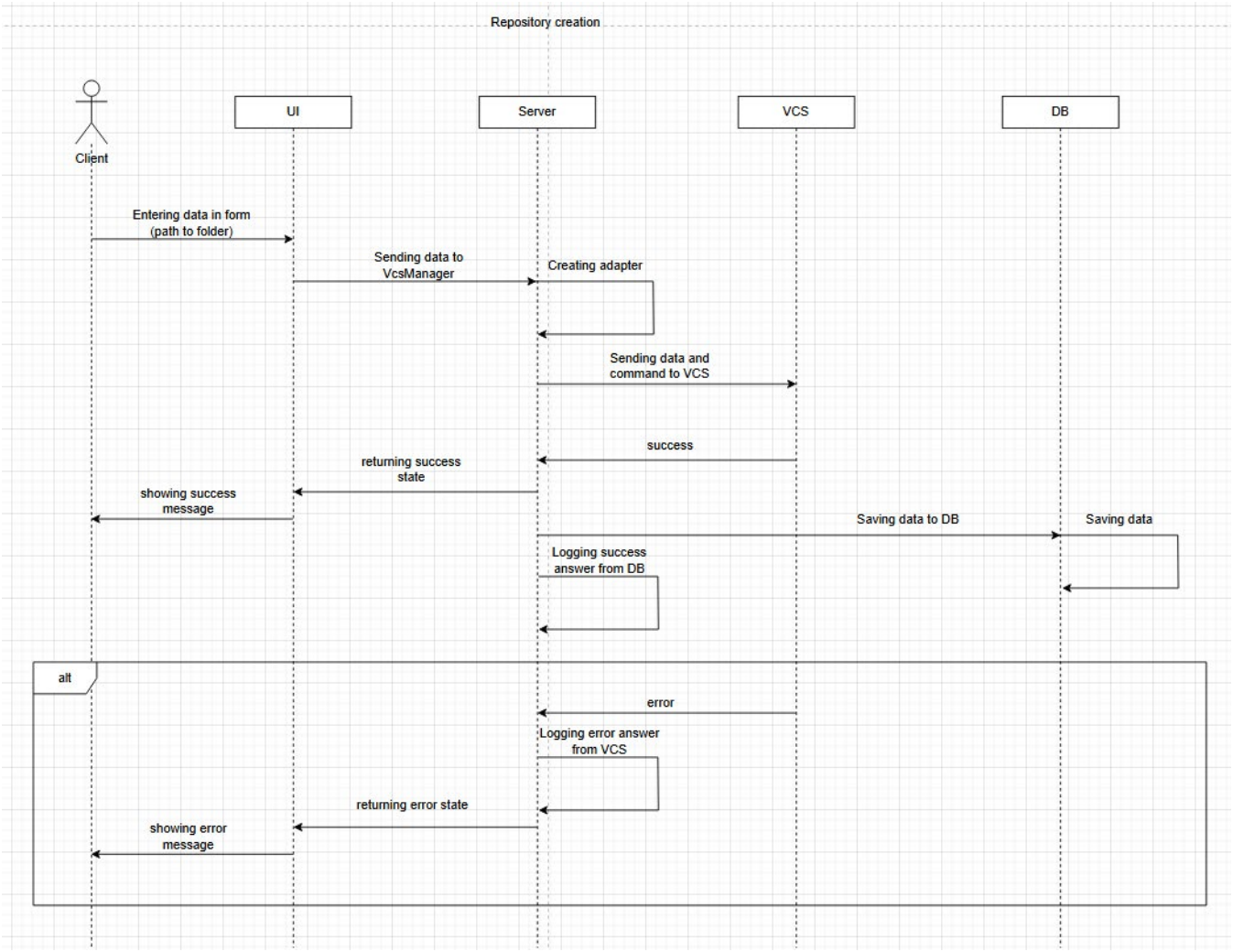

Рис 3.1 Діаграма послідовностей «Створення репозиторію»

Перебіг подій:

- Надсилання даних через UI
- Дані передаються на сервер частину, де викликається відповідний, до обраної системи контролю версій, адаптер
- Адаптер викликає метод з переданими даними
- Система контролю версій робить операцію та повертає один з двох станів – успіх чи помилка
- При успішному виконанні створюється відповідний запис у базі даних та відбувається логування.
- Передача інформації про успішне виконання до UI. При помилці виконується логування помилки та передається стан помилки на UI
- Вивід повідомлення про успішну операцію. При помилці вивід про неуспішне виконання та причини

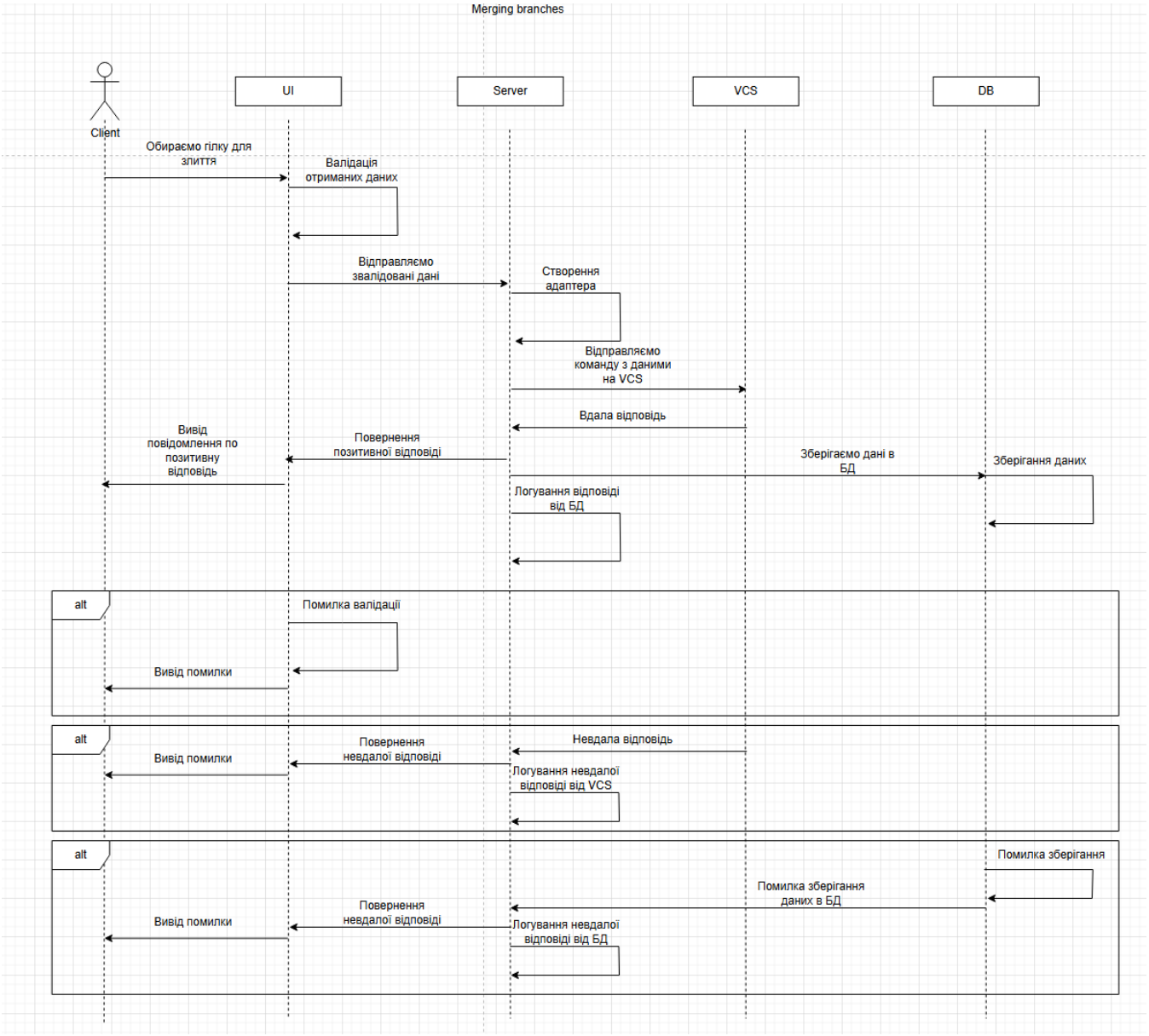

Рис 3.2 Діаграма послідовностей «Зливання гілок»

Перебіг подій:

- Користувач обирає гілку до якої треба робити злиття
- Дані передаються на сервер частину, де викликається відповідний, до обраної системи контролю версій, адаптер
- Адаптер викликає метод з переданими даними
- Система контролю версій робить операцію та повертає один з двох станів – успіх чи помилка
- При успішному виконанні створюється відповідний запис у базі даних та відбувається логування.
- Передача інформації про успішне виконання до UI. При помилці виконується логування помилки та передається стан помилки та конфліктуючі файли на UI

- Вивід повідомлення про успішну операцію. При помилці вивід про неуспішне виконання та причини

Код програми

https://github.com/mavr1iq/VCS-all-in-one

Висновки

Отже, під час виконання лабораторної роботи, досліджено процес проєктування діаграми розгортання системи разом з компонентами що проєктується задля подальшого розуміння процесу розгортання при деплої продукту. Для компонентів що проектується була зроблена діаграма компонентів, щоб наглядно було зрозуміло як різні елементи системи взаємодіють між собою. Також було досліджено розроблення діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі. Діаграма послідовностей явно показує, як перетікають події у системі з можливими виключеннями та/або альтернативними взаємодіями.

Контрольні питання

1. Що собою становить діаграма розгортання?
Діаграма розгортання (Deployment Diagram) у UML показує фізичне розміщення апаратних вузлів (серверів, клієнтів) і компонентів системи на цих вузлах. Вона відображає, як програмне забезпечення реалізується на апаратних елементах.

2. Які бувають види вузлів на діаграмі розгортання?
Фізичні вузли (Node): реальні апаратні пристрої (сервер, комп'ютер, мобільний пристрій). Вузли виконання (Execution Environment): середовище, у якому запускаються компоненти (операційна система, віртуальна машина, контейнер).

3. Які бувають зв'язки на діаграмі розгортання?
Ассоціація між вузлами (Communication Path): показує можливість обміну даними між вузлами. Залежність (Dependency): вказує, що один вузол або компонент залежить від іншого.

4. Які елементи присутні на діаграмі компонентів?
- Компоненти (Component): самостійні частини ПЗ.
- Інтерфейси (Interface): порти, через які компоненти взаємодіють.
- Пакети (Package): групування компонентів.
- Зв'язки (Dependency, Association): взаємозв'язки між компонентами.

5. Що становлять собою зв’язки на діаграмі компонентів?
Зв’язки відображають залежності між компонентами: хто на кого покладається, хто надає чи використовує інтерфейс іншого компонента.

6. Які бувають види діаграм взаємодії?
Діаграма послідовностей (Sequence Diagram) – показує порядок повідомлень між об’єктами. Діаграма комунікацій (Communication Diagram) – показує зв’язки між об’єктами та обмін повідомленнями. Діаграма часу (Timing Diagram) – показує зміни станів об’єктів у часі. Діаграма взаємодії (Interaction Overview Diagram) – поєднує елементи кількох діаграм взаємодії.

7. Для чого призначена діаграма послідовностей?
Вона описує порядок і часову послідовність взаємодії між об’єктами системи для реалізації певного сценарію чи варіанту використання.

8. Які ключові елементи можуть бути на діаграмі послідовностей?
- Об’єкти/Актори (Lifelines)
- Повідомлення (Messages) – виклики методів між об’єктами
- Активності (Activation bars) – час виконання дії об’єкта
- Події створення/знищення об’єктів

9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання?
Кожна діаграма послідовностей реалізує сценарій певного варіанту використання, деталізуючи, як актори взаємодіють із системою крок за кроком.

10. Як діаграми послідовностей пов’язані з діаграмами класів?
Повідомлення на діаграмі послідовностей зазвичай відповідають методам класів, а об’єкти на діаграмі є екземплярами класів із діаграми класів.