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

Лабораторна робота *№3*

із дисципліни ***«Технології розробки програмного забезпечення»***

Тема: ***«Основи проектування розгортання»***

Виконав:

Студент групи ІА-34

Ястремський Богдан

Перевірив:

асистент кафедри ІСТ

Мягкий Михайло Юрійович

Київ — 2025

Тема: Основи проектування розгортання.

Мета: Навчитися проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.

Короткі теоретичні відомості:

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).

Між вузлами можуть стояти зв'язки, які зазвичай зображують у вигляді прямої лінії. Як і на інших діаграмах, у зв'язків можуть бути атрибути множинності (для показання, наприклад, підключення 2x і більше клієнтів до одного сервера) і назва.

Вузли можуть містити артефакти (artifacts), які є фізичним уособленням програмного забезпечення; зазвичай це файли. Такими файлами можуть бути виконувані файли (такі як файли .exe, двійкові файли, файли DLL, файли JAR, збірки або сценарії) або файли даних, конфігураційні файли, HTML-документи тощо.

Артефакти можна зображати у вигляді прямокутників класів або перераховувати їхні імена всередині вузла. Якщо ви показуєте ці елементи у вигляді прямокутників класів, то можете додати значок документа або ключове слово «artifact».

Діаграма компонентів UML є представленням проєктованої системи, розбитої на окремі модулі.

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

Коли на діаграмі представляють фізичне розбиття, то в такому разі на діаграмі компонентів показують компоненти та залежності між ними. Залежності показують, що класи в з одного компонента використовують класи з іншого компонента. Фізична модель використовується для розуміння які компоненти повинні бути зібрані в інсталяційний пакет. Також така діаграма показує зміни в якому компоненті будуть впливати на інші компоненти.

Діаграма послідовностей (Sequence Diagram) – це один із типів діаграм у моделюванні UML, 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.

Хід роботи:

1. Розробити діаграму компонентів для проєктованої системи.

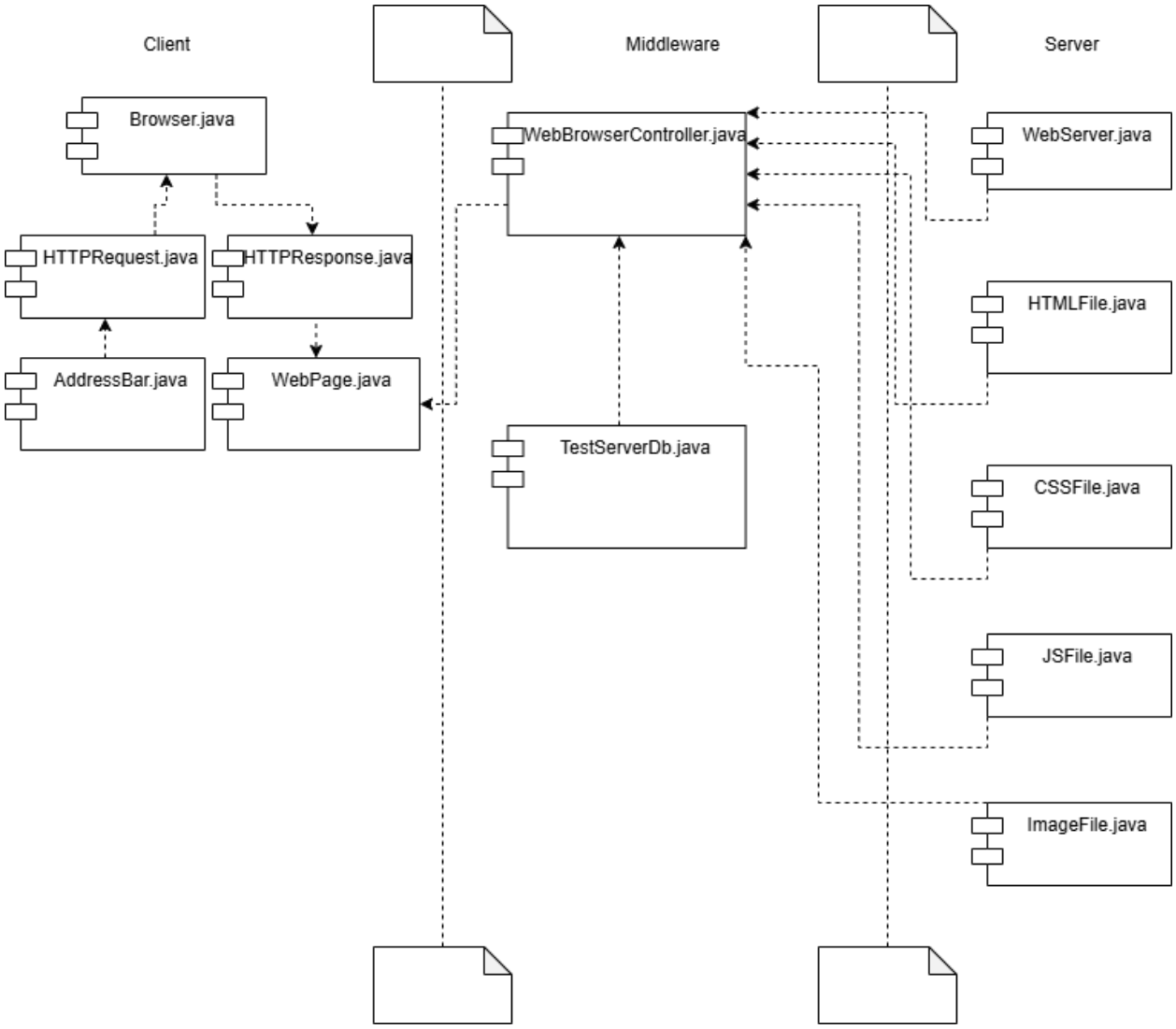

Рис. 3.1 – Діаграма компонентів системи

На діаграмі показано, з адресної строки ми направляємо запит у браузер, а браузер повертає відповідь у вигляді вебсторінки.

Також я додав тестовий сервер test.com для перевірки помилок 404, 502 і 503 і відображення відповідних сторінок. Всі компоненти сервера взаємодіють з контролером, який повертає необхідну веб сторінку.

Контролер ініціалізує JavaFX застосунок (робота з інтерактивними елементами: кнопка, поле вводу), тестовий сервер (для перевірки помилок 404, 502 і 503 і виводу відповідних сторінок), відповідає за

вивід інформації про сторінки в консоль, переміщення і перезавантаження.

В якості БД, я обрав БД для тестового серверу, який буде перевіряти помилки 404, 502 та 503. По суті, ця БД вбудована в контролер (на даному етапі), але можна її зробити окремо, по суті там лише html код сторінки.

Хоча БД можна також створити для збереження даних користувачів, наприклад ПІБ, е-пошти, паролю і т.д.

2. Розробити діаграму розгортання для проєктованої системи:

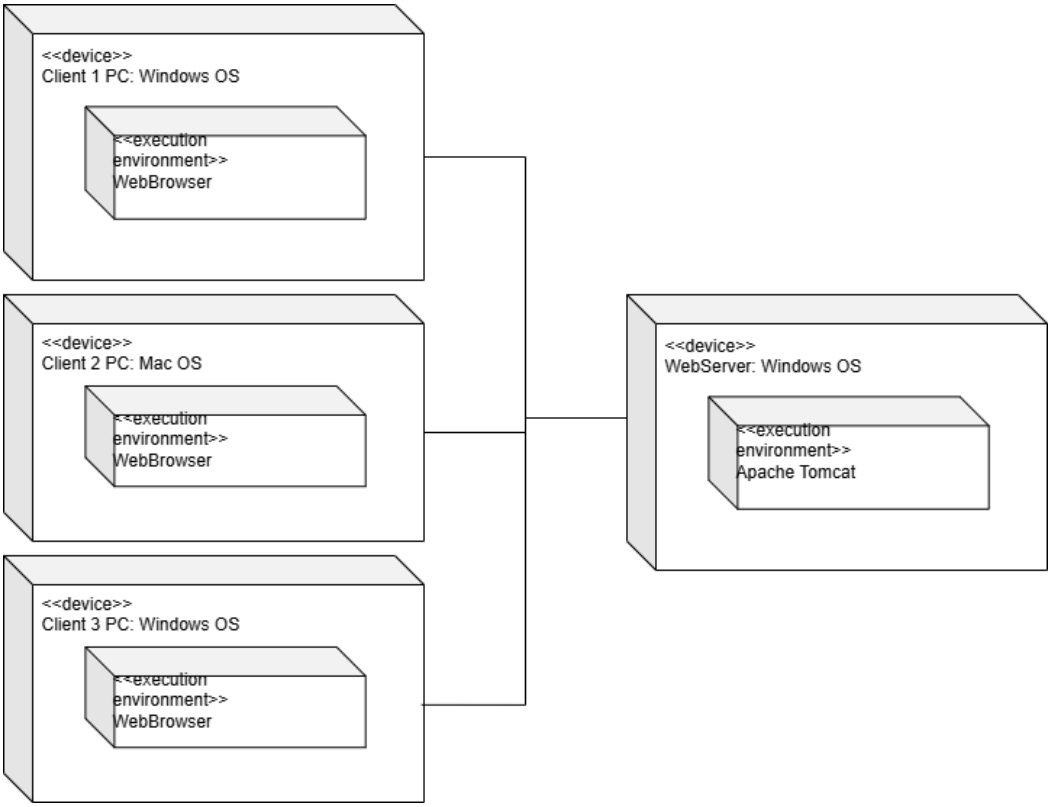

Рис. 3.2 – Діаграма розгортання системи

Для реалізації системи я обрав мову Java, оскільки вона є кросплатформною, і не доведедеться для кожної ОС підлаштовувати код на іншій мові програмування, тому на схемі я позначив 2 користувачів на Windows, і одного на Mac OS, хоча можна включити безліч користувачів на безлічі ОС.

Сервер по суті буде на моєму ПК, тому я задав девайс, як свою ОС – Windows. Але, оскільки застосунок десктопний, то сервером буде ОС, на якій користувач використовуватиме браузер.

3. Розробити як мінімум дві діаграми послідовностей для сценаріїв прописаних в попередній лабораторній роботі.
   1) Сценарій «Користувач – Введення URL»:

      Передумови: Користувач відкрив веб-браузер, має доступ до Інтернету та бажає перейти на конкретний сайт.

      Постумови: Браузер спробує підключитись до введеного URL, передавши запит на сервер, і почне завантажувати веб-сторінку.

      Сторони взаємодії: Користувач, Веб-браузер.

      Короткий опис: Користувач вводить адресу веб-сайту в адресний рядок браузера, після чого браузер ініціює HTTP-запит до серверу.

      Основний перебіг подій:

      - Користувач відкриває браузер.
      - Користувач вводить URL у адресний рядок.
      - Браузер обробляє введену адресу і здійснює HTTP-запит до веб-сервера для завантаження сторінки.

      Винятки: Користувач вводить неправильний або неповний URL або неіснуючу адресу сайту.

      Примітки: у разі відсутності протоколу, браузер автоматично його додає.

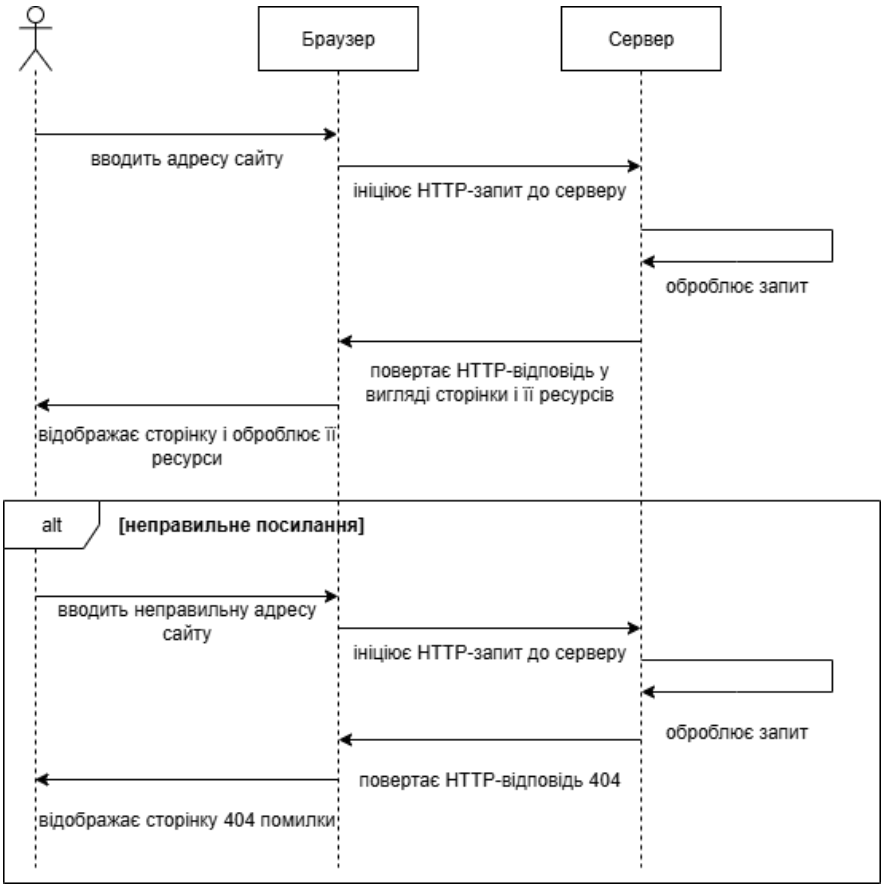

Рис. 3.3 – Діаграма послідовності для 1-го сценарію

2) Браузер – Обробка HTTP-відповіді:

Передумови: Браузер отримав відповідь від сервера у вигляді HTTP статусу та вмісту (HTML, CSS, зображення тощо).

Постумови: Браузер інтерпретує HTTP відповідь та відповідно відображає контент на екран. Якщо є помилка (наприклад, 404), браузер відображає відповідну сторінку помилки.

Сторони взаємодії: Браузер, Веб-сервер.

Короткий опис: Після отримання відповіді від сервера, браузер аналізує HTTP статус, інтерпретує його та відображає контент. Якщо відповідь містить помилку (наприклад, 404), браузер відображає сторінку з помилкою.

Основний перебіг подій:

- Браузер отримує HTTP-відповідь від сервера і перевіряє код статусу відповіді.

- Якщо відповідь успішна, браузер відображає сторінку з ресурсами.
- Якщо ні, то відображається сторінка з помилкою.

Винятки: Відповідь з помилкою або пошкодженим контентом.

Примітки: відсутні.

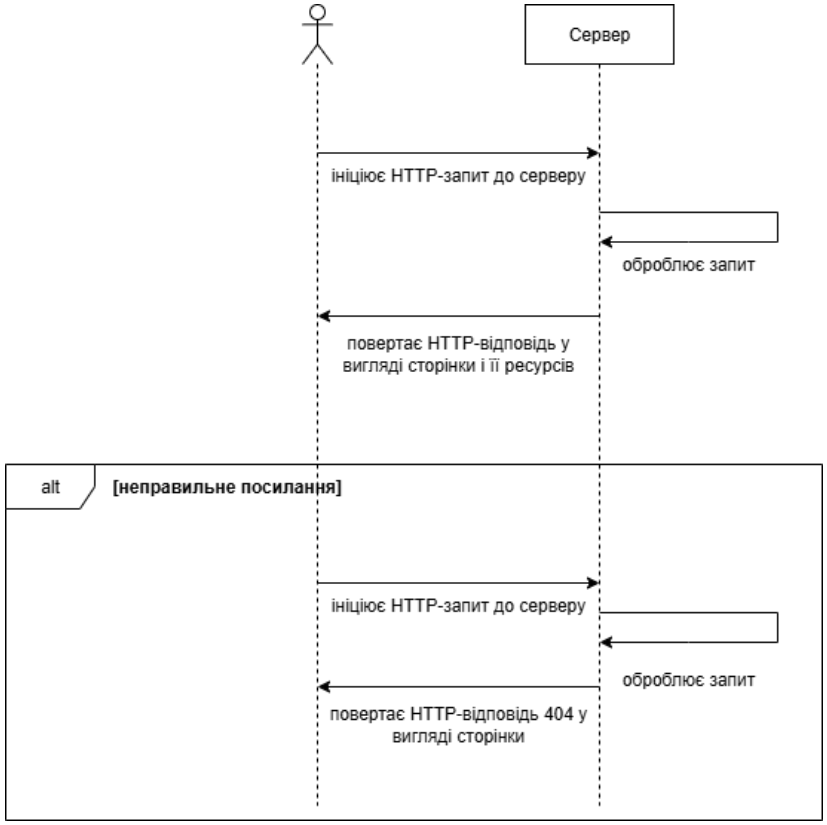

Рис. 3.4 – Діаграма послідовності для 2-го сценарію

4. 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). Посилання на github: https://github.com/bohdanyast/trpz

Висновок: на даному лабораторному занятті я продовжив роботу з мовою UML і навчився будувати діаграми компонентів, розгортання та послідовностей. Я зрозумів, що найважливішою діаграмою є діаграма

компонентів, оскільки на ній вже планується перші макети архітектури і вона визначає фундамент, як буде працювати застосунок. Діаграма розгортання по суті показує як буде працювати застосунок в цілому і потрібна вона більше для загального розуміння. Діаграма послідовності є не менш важливою, оскільки відображає у більш зрозумілій формі, як має працювати застосунок.

Відповіді на контрольні питання:

1. Що собою становить діаграма розгортання?

   Вона показує на якому фізичному обладнанні запускається та чи інша складова програмного забезпечення – власне система, сервер і т.д.

2. Які бувають види вузлів на діаграмі розгортання?

   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 (ОС, вебсервер).

3. Які бувають зв’язки на діаграмі розгортання?

   Зв’язок по протоколу: HTTP, HTTPS.

   Зв’язок, завдяки іншій технології для забезпечення взаємодії.

4. Які елементи присутні на діаграмі компонентів?

   На діаграмах компонентів з виконуваним поділом компонентів кожен компонент являє собою деякий файл – виконувані файли (.exe), файли вихідних кодів, сторінки html, бази даних і таблиці тощо.

5. Що становлять собою зв’язки на діаграмі компонентів?

   Залежності показують, що класи в з одного компонента використовують класи з іншого компонента.

6. Які бувають види діаграм взаємодії?
   - діаграма послідовності: показує взаємодію об'єктів у часовій послідовності
   - діаграма комунікації: фокусується на зв'язках між об'єктами та повідомленнях
   - діаграма огляду взаємодії: діаграма послідовності + діаграма комунікації
   - діаграма синхронізації: моделювання складних взаємодій

7. Для чого призначена діаграма послідовностей?

Вона відображає, як об'єкти обмінюються повідомленнями, показуючи порядок і логіку виконання операцій.

8. Які ключові елементи можуть бути на діаграмі послідовностей?
   - Актори (Actors): користувачі чи інші системи, які взаємодіють із системою.
   - 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).
9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання?

   Базуючись на діаграмі варіантів використання можна створити сценарій використання, а вже за ним створити діаграму послідовностей.
10. Як діаграми послідовностей пов’язані з діаграмами класів?

    Базуючись на діаграмі варіантів використання можна створити сценарій використання, а вже за ним виокремити класи, які прийматимуть участь та створити діаграму послідовностей.