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

Технології розроблення програмного забезпечення

*Тема роботи: «*ДІАГРАМА РОЗГОРТАННЯ. ДІАГРАМА КОМПОНЕНТІВ. ДІАГРАМА ВЗАЄМОДІЙ ТА ПОСЛІДОВНОСТЕЙ.*»*

Виконав студент групи ІА-23:

Каширов Д. О.

Перевірив:

Мягкий Михайло Юрійович

Київ 2024

Мета: Навчитися розробляти діаграму розгортання. діаграма компонентів. діаграма взаємодій та послідовностей.

Хід роботи

Варіант:

..6 Web-browser (proxy, chain of responsibility, factory method, template method, visitor, p2p) Веб-браузер повинен мати можливість зробити наступне: мати адресний рядок для введення адреси сайту, переміщатися і відображати структуру html документа, переглядати підключений javascript та css файли, перегляд всіх підключених ресурсів (зображень), коректна обробка відповідей з сервера (коди відповідей HTTP) - переходи при перенаправленнях, відображення сторінок 404 і 502/503.

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

Теоретичні відомості

• **Діаграма розгортання (Deployment Diagram)** - це один з типів діаграм UML (Unified Modeling Language), який використовується для моделювання фізичної структури та розміщення компонентів програмної системи на апаратному обладнанні, такому як сервери, комп'ютери, сенсори, мережеві пристрої тощо. Діаграма розгортання допомагає візуалізувати, як компоненти програмної системи розташовані на різних об'єктах обладнання та як вони взаємодіють між собою через мережу.

• Вузли (Nodes): Вузли представляють фізичні або віртуальні об'єкти обладнання, такі як сервери, робочі станції, маршрутизатори, бази даних, сенсори тощо. Кожен вузол може мати атрибути, такі як IP-адреси або характеристики обладнання.

• **Артефакти (Artifacts):** Артефакти представляють програмні компоненти або файли, які розгортаються на вузлах. Це можуть бути виконувані файли, конфігураційні файли, бази даних, додатки тощо.

• **Зв'язки (Connections):** Зв'язки показують, як компоненти взаємодіють між собою на різних вузлах. Зв'язки можуть позначати мережеві з'єднання, передачу даних, доступ до служб тощо.

Діаграми розгортання корисні для інженерів програмного забезпечення та системних архітекторів для візуалізації архітектурної концепції системи, розміщення компонентів на конкретних серверах чи обладнанні.

Діаграма розгортання:

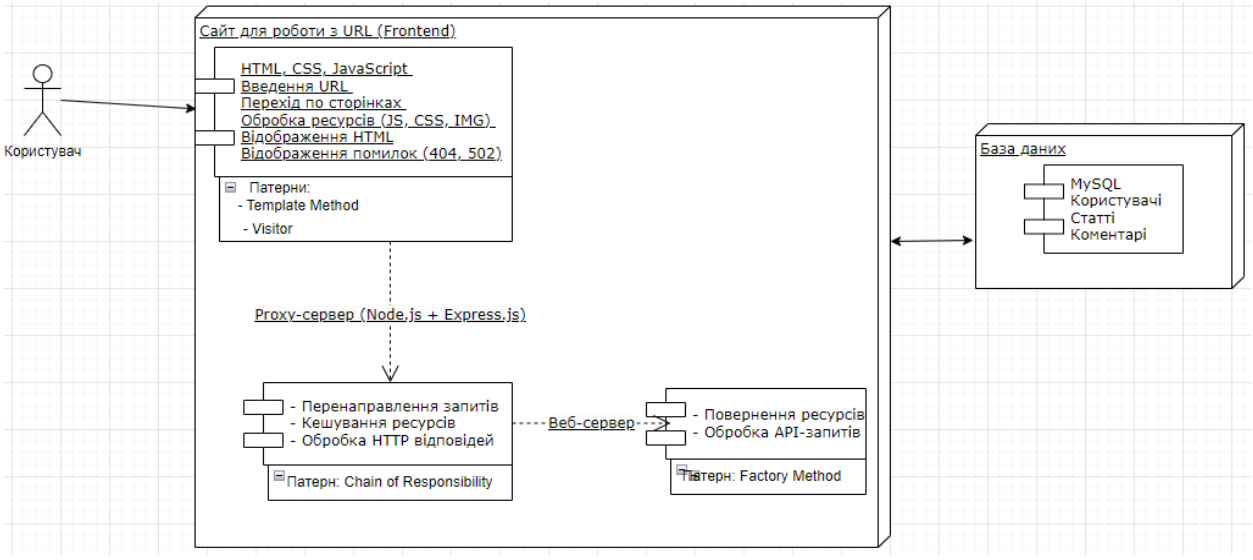

Рис 1 діаграма розгортання генератора інсталяційних пакетів

На діаграмі розгортання показано архітектуру веб-застосунку, яка складається з кількох компонентів, що взаємодіють між собою для забезпечення функціонування системи. Ось детальний опис того, що зображено на цій діаграмі:

1. Користувач:
   Це кінцевий користувач, який взаємодіє із системою через веб-браузер. Користувач може вводити URL-адресу, переходити між сторінками сайту та взаємодіяти з різними ресурсами.
2. Веб-браузер (Frontend):
   Веб-браузер відповідає за відображення веб-сторінок. Він обробляє HTML, CSS та JavaScript, які визначають структуру та стиль сторінок, а також функціональність через взаємодію з користувачем.
   - Введення URL: Користувач вводить адресу сайту в браузері.
   - Перехід по сторінках: Користувач може переміщатися між різними сторінками.
   - Обробка ресурсів (JS, CSS, IMG): Браузер завантажує й обробляє ресурси, необхідні для рендерингу сторінки (наприклад, зображення, стилі, скрипти).
   - Відображення HTML: Після обробки браузер відображає HTML-контент.
   - Відображення помилок (404, 502): Браузер показує повідомлення про помилки, наприклад, якщо сторінка не знайдена (404) або сервер недоступний (502).
3. Proxy-сервер (Node.js + Express.js):
   Цей сервер працює як проміжний рівень між клієнтом і веб-сервером. Він виконує кілька ключових функцій:
   - Перенаправлення запитів: Proxy-сервер перенаправляє запити від клієнта до відповідних серверів або ресурсів.
   - Кешування ресурсів: Сервер може зберігати ресурси в кеші для прискорення доступу до них у майбутньому.
   - Обробка HTTP відповідей: Proxy-сервер відповідає на запити клієнтів, обробляючи HTTP-відповіді від веб-сервера.
4. Веб-сервер:
   Веб-сервер обробляє запити на отримання ресурсів та API-викликів. Він

відповідає за забезпечення доступу до статичних та динамічних ресурсів:

- Повернення ресурсів: Веб-сервер віддає клієнту статичні файли (HTML, CSS, JavaScript, зображення тощо).
- Обробка API-запитів: Веб-сервер обробляє запити до API для взаємодії з базою даних або виконання інших функцій.

5. База даних (MySQL):
   База даних зберігає всю необхідну інформацію, таку як дані користувачів, статті та коментарі. Це основне місце для збереження даних, які використовуються та змінюються під час взаємодії користувача з веб-застосунком:
   - Користувачі: Інформація про користувачів, їхні облікові записи та налаштування.
   - Статті: Збереження контенту статей або публікацій.
   - Коментарі: Дані про коментарі, які користувачі можуть залишати під статтями або іншими публікаціями.

Ця діаграма розкриває, як веб-браузер взаємодіє з сервером через proxy, який опрацьовує запити та відповіді, а також як сервер підключається до бази даних для зберігання та отримання даних.

Діаграма компонентів

Діаграма компонентів (іноді також називається діаграмою компонентної структури) - це тип діаграми, яка використовується в інженерії програмного забезпечення для візуального представлення архітектури системи або програмного продукту та її компонентів (або модулів). Діаграми компонентів допомагають розуміти, як система складається з окремих частин (компонентів), як вони взаємодіють між собою та як вони спільно працюють для досягнення цілей системи.

• **Компоненти:** Це основні модулі або блоки, які складають систему або програмний продукт. Кожен компонент зазвичай виконує певну функцію і може бути реалізований як окремий програмний модуль або об'єкт.

• **Зв'язки:** Діаграми компонентів включають зв'язки між компонентами, що показують, як вони взаємодіють між собою. Зв'язки можуть представляти залежності між компонентами, наприклад, використання одного компонента

іншим, або вони можуть вказувати на комунікацію між компонентами через інтерфейси.

• **Інтерфейси:** Компоненти можуть мати інтерфейси, які описують спосіб взаємодії з іншими компонентами. Інтерфейси вказують, які методи або функції можуть бути викликані із зовнішніх компонентів.

• **Залежності:** Діаграми компонентів можуть також показувати залежності між компонентами, що вказують на те, як один компонент може впливати на інший.

Діаграма компонентів:

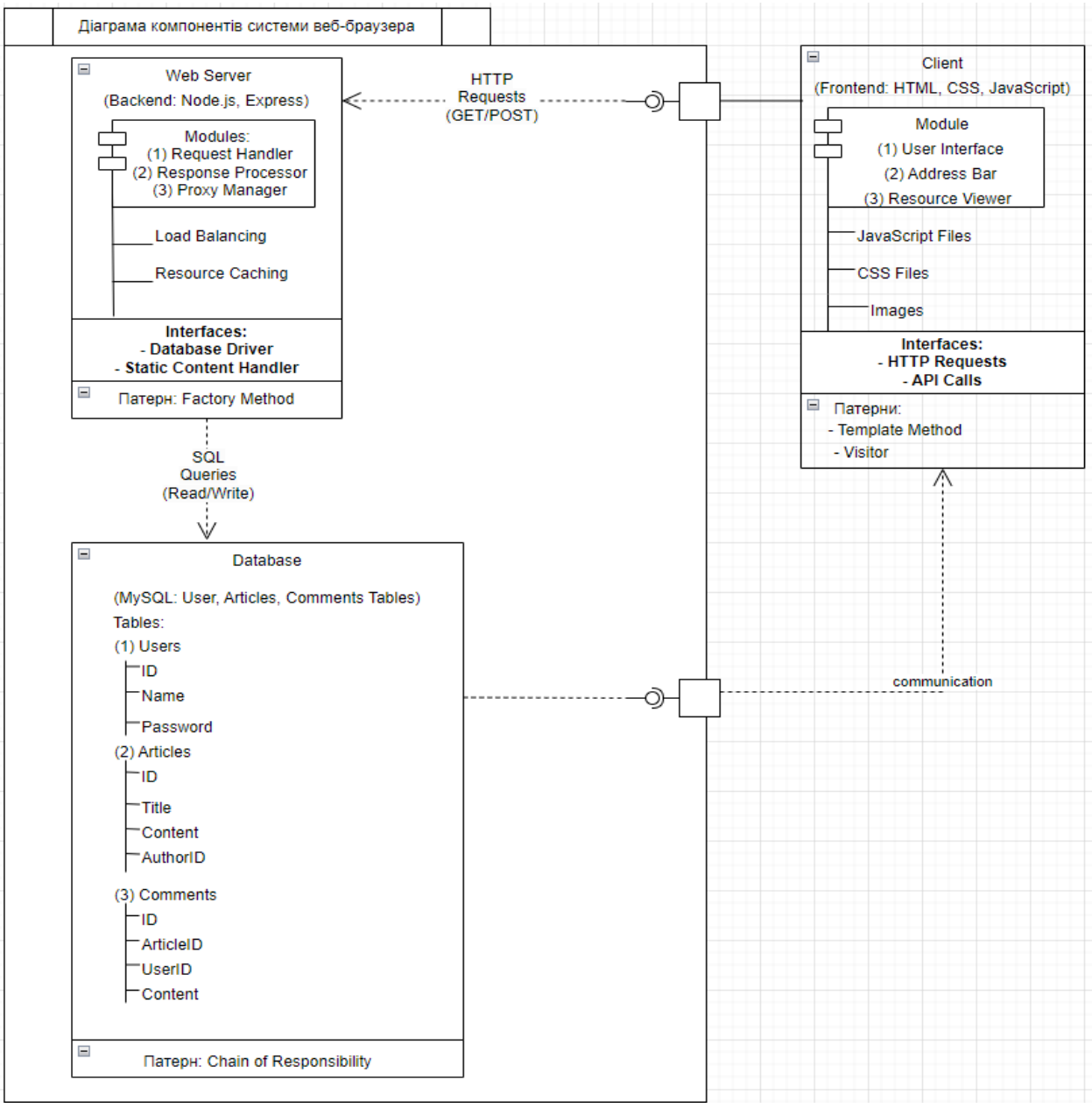

Рис 2 діаграма компонентів системи веб-браузера

На цій діаграмі компонентів ми бачимо архітектуру проектованої системи, яка складається з трьох основних рівнів:

1. **Frontend (Клієнтська частина):**
    - Компоненти для роботи з інтерфейсом користувача, такі як адресний рядок, переглядач ресурсів (зображення, CSS, JavaScript).
    - Взаємодія з сервером через HTTP-запити.
2. **Backend (Серверна частина):**
    - Компоненти для обробки запитів, відповідей та управління проксі-кешуванням.
    - Використання Node.js та Express.js для динамічного управління даними та логіки.
3. **Database (База даних):**
    - MySQL для зберігання інформації про користувачів, статті та коментарі.
    - Зв'язок із сервером через SQL-запити.

Загалом:

Діаграма показує, як клієнт, сервер та база даних взаємодіють, розділяючи відповідальність між модулями та забезпечуючи ефективну роботу системи.

Діаграми послідовностей:

Ці діаграми використовуються для візуалізації взаємодії між об'єктами або компонентами в системі впродовж певного проміжку часу. Вони дозволяють показати, як об'єкти взаємодіють між собою, обмінюючи повідомленнями або виконуючи методи, і як ця взаємодія впливає на стан системи. Основні елементи діаграм послідовностей включають:

• **Об'єкти (Objects):** Об'єкти або сутності, які беруть участь у взаємодії, представлені у верхній частині діаграми. Кожен об'єкт позначається ім'ям і може мати життєвий цикл, який відображається на діаграмі зліва направо.

• **Лінії життєвого циклу (Lifelines):** Лінії життєвого циклу об'єкта відображають, як триває його існування під час взаємодії. Вони 6

позначаються вертикальними лініями, на яких розташовані повідомлення, інтервали активації та інші моменти життєвого циклу об'єкта.

• **Повідомлення (Messages):** Повідомлення показують передачу інформації або виконання дій між об'єктами. Існують різні типи повідомлень, такі як синхронні (з блокуванням), асинхронні (без блокування), відповіді на повідомлення тощо. Діаграми послідовностей корисні для аналізу та проектування взаємодії в системі. Вони допомагають визначити порядок виконання операцій та повідомлень між об'єктами та можуть бути використані для детального опису архітектурних взаємодій в програмному забезпеченні перед його реалізацією.

Діаграми послідовностей:

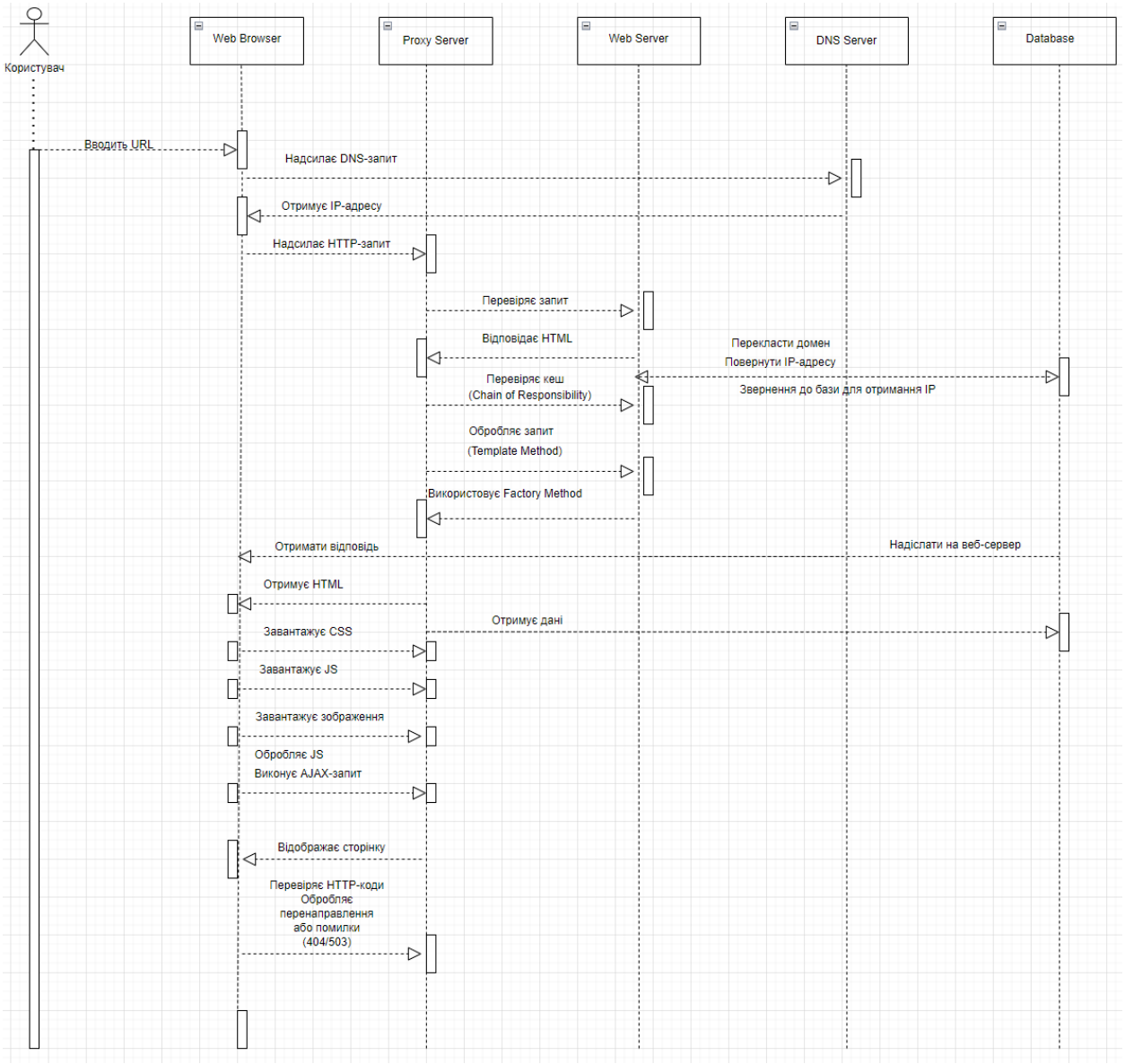

Рис 3 Діаграма послідовностей взаємодії клієнта з веб-системою

Ця діаграма описує весь процес взаємодії між різними компонентами системи при завантаженні веб-сторінки. Вона відповідає ключовим етапам вашого проекту, який стосується аналізу передачі даних через різні шари в мережі Інтернет. Нижче наведено пояснення:

Опис діаграми

1. **Користувач**:
    - Користувач вводить URL веб-сайту в адресний рядок браузера.
    - Цей етап ініціює процес взаємодії між усіма компонентами.
2. **Web Browser (Веб-браузер)**:
    - Браузер починає процес, надсилаючи DNS-запит для отримання IP-адреси веб-сайту. DNS-запит пересилається до DNS-сервера.
    - Після отримання IP-адреси браузер надсилає HTTP-запит через Proxy Server до веб-сервера.
    - Отримує відповідь від сервера (HTML-код сторінки) і починає завантаження додаткових ресурсів: CSS, JavaScript, зображень тощо.
    - Завершує обробку JavaScript, виконує AJAX-запити (якщо потрібні), і відображає сторінку користувачеві.
3. **Proxy Server (Проксі-сервер)**:
    - Виступає посередником між браузером і веб-сервером.
    - Перевіряє HTTP-запит (можливо, на предмет безпеки чи доступу) і пересилає його до веб-сервера.
    - Отримує відповідь від веб-сервера та передає її назад браузеру.
4. **Web Server (Веб-сервер)**:
    - Обробляє HTTP-запит від браузера, генерує відповідь у вигляді HTML-документа.
    - Відповідає браузеру HTML-кодом сторінки, а також надає доступ до додаткових ресурсів, таких як зображення, CSS та JavaScript.
5. **DNS Server (DNS-сервер)**:
    - Переводить доменне ім’я (наприклад, www.example.com) у відповідну IP-адресу.
    - Повертає цю IP-адресу браузеру, щоб браузер знав, до якого сервера надсилати HTTP-запит.
6. **Обробка помилок**:

- Якщо веб-сервер недоступний або сторінка не знайдена, сервер може повернути помилки (наприклад, 404 або 503).
- Браузер обробляє ці помилки і показує відповідні повідомлення користувачу.

Як ця діаграма відповідає проекту?

1. **Мета проекту**:
   - Вона чітко відображає весь процес завантаження веб-сторінки від введення URL до повного рендерингу сторінки у браузері.
   - Відображає ключові взаємодії між користувачем, браузером, проксі-сервером, веб-сервером і DNS-сервером.
2. **Деталізація**:
   - Усі основні етапи, включаючи отримання DNS-адреси, передачу запитів через Proxy Server, обробку запитів на веб-сервері, завантаження ресурсів і відображення сторінки, описані в діаграмі.
3. **Практичне застосування**:
   - Діаграма може бути використана для пояснення роботи Інтернету, взаємодії між клієнтом і сервером, а також для аналізу можливих точок збою, таких як затримки у відповіді DNS або помилки на веб-сервері.

Висновок:

Під час виконання лабораторної роботи я навчився створювати та аналізувати ключові діаграми, які відображають архітектуру і функціональність проектованої системи. Зокрема, було розроблено такі типи діаграм:

1. **Діаграма розгортання**:
   Ця діаграма допомогла зрозуміти, як фізично розташовані та взаємодіють між собою різні компоненти системи, такі як сервери, клієнтські пристрої, проксі-сервери, DNS-сервери та бази даних. Було наочно показано, як здійснюється комунікація між компонентами через мережу.

2. **Діаграма компонентів**:
   Цей тип діаграми дозволив візуалізувати логічну структуру системи, тобто взаємозв'язок між веб-браузером, веб-сервером, DNS-сервером та базою даних. Вона показала, як кожен компонент взаємодіє для забезпечення цілісного функціонування системи, включаючи опрацювання HTTP-запитів, отримання IP-адрес та повернення відповідей клієнту.

3. **Діаграма послідовностей**:
   На цій діаграмі було детально продемонстровано процеси обміну інформацією між основними складовими системи. Відображено послідовність дій, починаючи з введення користувачем URL-адреси у веб-браузері до відображення готової веб-сторінки. Також враховано обробку помилок, запити до бази даних та виконання AJAX-запитів.

Виконання роботи дало змогу краще зрозуміти принципи функціонування клієнт-серверної архітектури та механізмів роботи веб-додатків, таких як DNS-запити, HTTP-запити та отримання статичних і динамічних ресурсів. Окрім цього, я здобув практичні навички у створенні діаграм UML, які є важливими інструментами моделювання для проектування складних систем.

Результати роботи можуть бути застосовані для документування системи, аналізу її ефективності, а також для вдосконалення архітектури під час майбутньої розробки.