



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 №2
З дисципліни «Технології розроблення програмного забезпечення»
Тема: **«ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ. СЦЕНАРІЙ
ВАРІАНТІВ ВИКОРИСТАННЯ. ДІАГРАМИ UML. ДІАГРАМИ КЛАСІВ.
КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ»**
HTTP-сервер

Виконав:
Студент групи ІА-22
Білецький С. В.

Перевірив:
Мягкий М. Ю.

Київ-2024

Зміст

Тема:.....	3
Мета:.....	3
Хід роботи.....	3
1. Схема прецедентів.....	3
2. Деталі сценаріїв використання.....	3
3. Схема класів.....	5
4. Структура бази даних.....	6
Висновки:.....	7

Тема:

Діаграма варіантів використання. Сценарії варіантів використання. Діаграми uml. Діаграми класів. Концептуальна модель системи

Мета:

Дослідити та застосувати UML-діаграми для моделювання варіантів використання і концептуальної структури даних системи, зосереджуючись на діаграмах класів, прецедентів та опису їхніх сценаріїв використання.

Завдання:

HTTP-сервер. Сервер повинен мати можливість розпізнавати вхідні запити і формувати коректні відповіді (згідно протоколу HTTP), надавати сторінки chhtml (html сторінки з додаванням найпростіших C# конструкцій на розсуд студента), вести статистику вхідних запитів, обробку запитів у багатопотоковому/подієвому режимах.

Хід роботи

1. Схема прецедентів

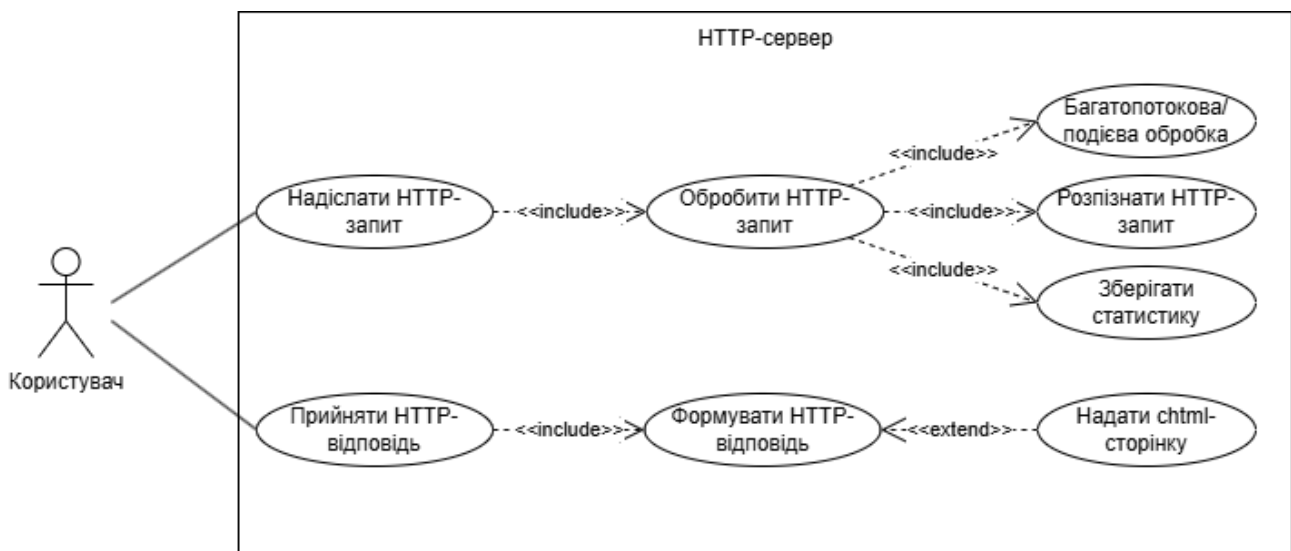


Рис. 3 - схема прецедентів

2. Деталі сценаріїв використання

Сценарій використання 1: Надіслати HTTP-запит

Передумови: Клієнтська програма готова для надсилання HTTP-запиту до сервера.

Післяумови: Сервер отримує та обробляє запит.

Актори: Клієнтська програма.

Опис: Цей сценарій описує процес надсилання HTTP-запиту від клієнтської програми до HTTP-сервера.

Основний хід подій:

1. Клієнтська програма створює HTTP-запит.
2. Клієнтська програма надсилає запит на сервер.
3. Сервер отримує HTTP-запит.

Винятки: Якщо з'єднання з сервером неможливе, клієнтська програма отримує повідомлення про помилку.

Сценарій використання 2: Обробити HTTP-запит

Передумови: Сервер отримав HTTP-запит від клієнта.

Післяумови: Запит оброблено, і сервер готовий до формування відповіді.

Актори: HTTP-сервер.

Опис: Цей сценарій описує процес обробки HTTP-запиту сервером, включаючи розпізнавання та багатопоточну обробку.

Основний хід подій:

1. Сервер розпізнає тип запиту (наприклад, GET або POST).
2. Сервер здійснює багатопоточну/подієву обробку для підвищення ефективності.
3. Сервер виконує необхідні операції для обробки запиту.
4. Сервер зберігає інформацію про запит у статистику.

Винятки: Якщо запит неможливо розпізнати, сервер повертає повідомлення про помилку.

Сценарій використання 3: Формувати HTTP-відповідь

Передумови: Сервер успішно обробив запит і готовий надіслати відповідь.

Післяумови: Відповідь сформована та надіслана клієнту.

Актори: HTTP-сервер.

Опис: Цей сценарій описує процес формування HTTP-відповіді сервером для клієнта.

Основний хід подій:

1. Сервер формує HTTP-відповідь на основі обробленого запиту.
2. Сервер додає вміст сторінки (HTML) до відповіді, якщо запитує html-сторінку.
3. Сервер надсилає відповідь клієнтській програмі.

Виятки: Якщо вміст сторінки не знайдено, сервер повертає повідомлення про помилку 404.

3. Схема класів

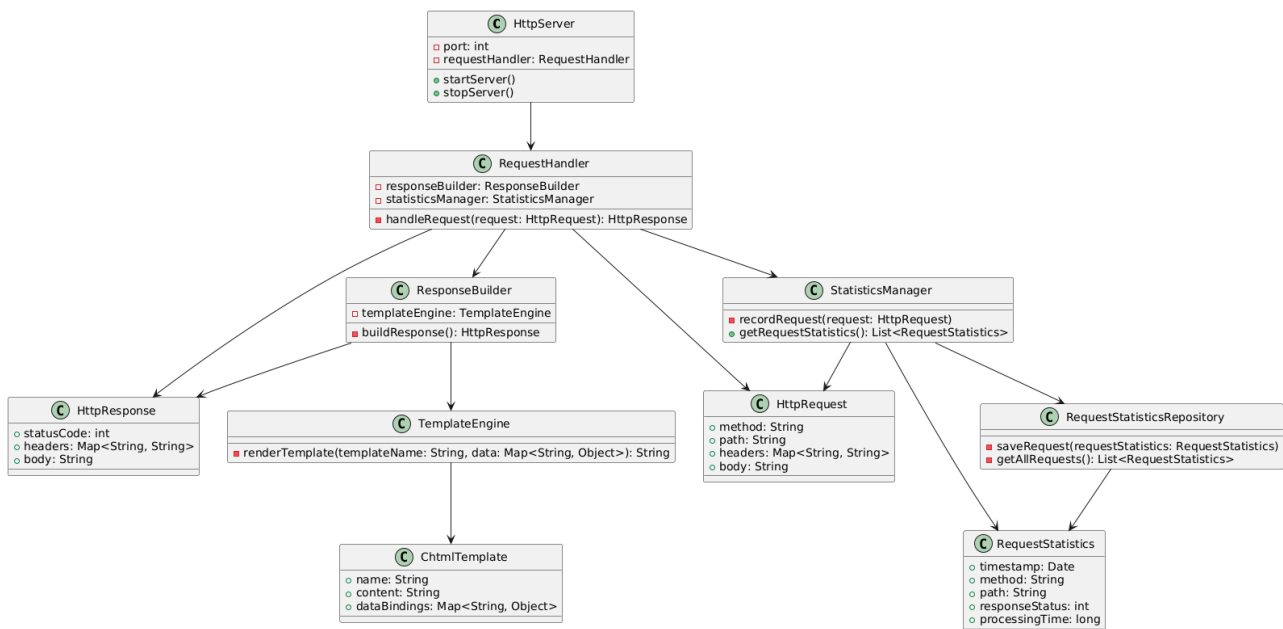


Рис. 2 - схема класів

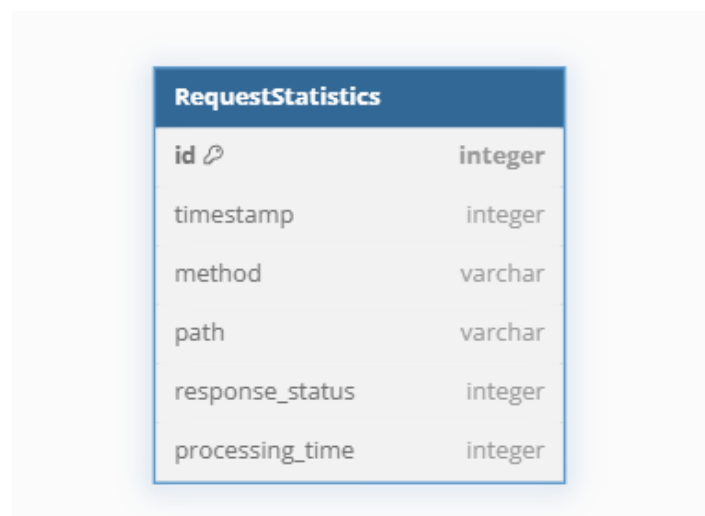
Ця діаграма класів відображає структуру HTTP-сервера, здатного обробляти запити, генерувати HTML-відповіді, та вести статистику запитів. Основні класи:

1. `HttpServer` — головний клас для запуску та зупинки сервера.
2. `RequestHandler` — обробляє HTTP-запити, використовуючи `ResponseBuilder` для формування відповідей та `StatisticsManager` для збору статистики.

3. `HttpRequest` та `HttpResponse` — представляють моделі HTTP-запиту та відповіді.
4. `ResponseBuilder` — будує HTTP-відповіді на основі шаблонів через `TemplateEngine`.
5. `TemplateEngine` — рендерить HTML-сторінки (CHTML) з підтримкою простих C# конструкцій.
6. `ChtmlTemplate` — модель HTML-шаблону для генерації динамічних сторінок.
7. `StatisticsManager` — збирає та зберігає статистику HTTP-запитів.
8. `RequestStatisticsRepository` — зберігає статистику запитів у базі даних.
9. `RequestStatistics` — модель для збереження деталей кожного HTTP-запиту.

Ця архітектура дозволяє серверу обробляти запити, генерувати HTML-відповіді з динамічним контентом, зберігати статистику у базі даних і працювати в багатопотоковому режимі, що підходить для хостингу API та веб-сторінок.

4. Структура бази даних



RequestStatistics	
id 🔗	integer
timestamp	integer
method	varchar
path	varchar
response_status	integer
processing_time	integer

Рис. 1 - структура бази даних

Ця структура бази даних є таблицею для збереження статистики запитів HTTP-сервера. Вона містить наступні поля:

- `id` (integer): Унікальний ідентифікатор кожного запису, який використовується для посилання на конкретний запит у таблиці.

- `timestamp (integer)`: Час, коли був зроблений запит, збережений у вигляді цілого числа (найімовірніше, у форматі UNIX-часу).
- `method (varchar)`: HTTP-метод запиту (наприклад, GET, POST, PUT, DELETE), який був використаний клієнтом.
- `path (varchar)`: Шлях ресурсу або URL, до якого було здійснено запит.
- `response_status (integer)`: Код статусу відповіді HTTP (наприклад, 200 для успішного запиту, 404 для помилки "не знайдено").
- `processing_time (integer)`: Час, витрачений на обробку запиту, вимірюється в мілісекундах.

У контексті HTTP-протоколу, який є *stateless* (тобто не зберігає стан між запитами), немає необхідності зберігати багато інформації про самі сеанси або дані користувачів, оскільки кожен запит незалежний від попереднього. Єдине, що потрібно зберігати для статистики, це базова інформація про кожен вхідний запит (час, метод, шлях, код статусу і час обробки). Це дозволяє здійснювати моніторинг продуктивності сервера, аналізувати частоту помилок та оптимізувати обробку запитів.

Отже, таблиця є достатньою для цього проекту, оскільки зберігається лише статистика, що відповідає вимогам протоколу HTTP.

Висновки:

Дослідили та застосували UML-діаграми для моделювання варіантів використання і концептуальної структури даних системи, зосереджуючись на діаграмах класів, прецедентів та опису їхніх сценаріїв використання.