НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Кафедра інформаційних систем та технологій

Тема «HTTP-сервер»

Курсова робота
З дисципліни «Технології розроблення програмного забезпечення»

Керівник
доц. Амонс О.А.
«Допущений до захисту»
_____
(Особистий підпис керівника)
« » _____2024р.
Захищений з оцінкою
_____
(оцінка)

Виконавець
ст. Машин Даниїл Ігорович
залікова книжка № ___ – _____
гр. ІА-31
_____
(особистий підпис виконавця)
«17» вересня 2025р.

Члени комісії:

_____ _____
(особистий підпис) (розшифровка підпису)

_____ _____
(особистий підпис) (розшифровка підпису)

Київ – 2025

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
(назва навчального закладу)
Кафедра ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ
Дисципліна «Технології розроблення програмного забезпечення»
Курс 3 Група ІА-31 Семестр 5

ЗАВДАННЯ

на курсову роботу студента

Машина Даниїла Ігоровича

(прізвище, ім’я, по батькові)

1.Тема роботи HTTP-сервер

2. Строк здачі студентом закінченої роботи

3. Вихідні дані до роботи:

_____

_____

_____

4. Зміст розрахунково – пояснювальної записки (перелік питань, що підлягають розробці)

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

Додатки:

_____

5.Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

_____

_____

_____

_____

6. Дата видачі завдання: 17.09.2025

КАЛЕНДАРНИЙ ПЛАН

№, п/п	Назва етапів виконання курсової роботи	Строк виконання етапів роботи	Підписи або примітки
1.	Підбір та вивчення літератури	30.09.2025	
2.	Проектування на написання розділу 1	31.10.2025	
3.	Розробка та написання розділу 2	20.11.2025	
4.	Подання курсової роботи на перевірку	25.11.2025	
5.	Захист курсової роботи	08.12.2025	

Студент _____ ___Машин Даниїл____

(підпис) (Ім’я ПРІЗВИЩЕ)

Керівник_____ __Олександр АМОНС__

(підпис) (Ім’я ПРІЗВИЩЕ)

«____»_____ 20___ р.

ЗМІСТ

ВСТУП

В світі веб-технологій HTTP-сервер відіграє роль незримного, але надзвичайно важливого моста, що з'єднує користувачів з веб-світом. Його функції виявляються не лише у взаємодії з вхідними запитами та формуванні відповідей, а й у вмінні тримати руку на пульсі кожного відвідувача. Проте, це тільки вершина айсберга. У цій курсовій роботі ми поглибимося у світ HTTP-серверів, досліджуючи їхню здатність розпізнавати й обробляти вхідні запити відповідно до протоколу HTTP, а також вміння працювати у різних режимах - від багатопотокового до подієвого.

При створенні HTTP-сервера важливо врахувати широкий спектр аспектів, що охоплюють розпізнавання вхідних запитів, формування правильних відповідей відповідно до протоколу HTTP та надання сторінок chtml – схрещення HTML та C# конструкцій, яке дозволяє реалізувати індивідуальний підхід до створення веб-інтерфейсів.

Однак, основний виклик полягає не лише у забезпеченні базового функціоналу. Справжня складність виявляється у підтримці багатопотокового/подієвого режимів, що вимагає не лише ефективної обробки запитів, але й відповідного управління станом сервера, координації дій та забезпечення безперервної роботи при навантажених умовах.

Такий сервер має не лише відповідати запитам користувачів, але й забезпечувати статистику вхідних запитів – важливий інструмент для аналізу та покращення ефективності роботи. Таким чином, розробка HTTP-сервера вимагає ретельного розгляду та впровадження широкого спектру технічних концепцій для досягнення бажаного функціоналу та продуктивності.

Ця курсова робота не обмежується вирішенням практичних завдань, вона також націлена на ілюстрацію того, як сучасні патерни проектування та принципи архітектури можуть бути використані для створення ефективних рішень у сфері автоматизації процесів розробки.

1 ПРОЄКТУВАННЯ СИСТЕМИ

1.1. Огляд існуючих рішень

Розробка HTTP-сервера вимагає уваги до деталей та урахування різних аспектів. Наприклад, сервери, які використовуються вже деякий час, такі як Apache чи Nginx, хоча й відомі своєю надійністю, можуть виявитися складними у конфігурації для новачків. Окрім цього, підтримка великого обсягу запитів може призвести до високого споживання ресурсів, що може вплинути на продуктивність сервера та швидкодію відповідей.

У випадку молодших технологій, наприклад, Node.js чи Python, основними мінусами можуть бути вимоги до додаткових бібліотек для підтримки певних функцій. Крім того, деякі фреймворки можуть мати власні особливості, які потребують часу для освоєння та розуміння.

Деякі інструменти можуть бути менш ефективними у використанні ресурсів або мати обмежену масштабованість при обробці великого потоку запитів. Крім того, підтримка певних функцій, наприклад, сторінок chtml або обробки запитів у багатопотоковому/подійному режимах, може бути менш оптимізованою або вимагати додаткових налаштувань, що може затримати час розробки та вплинути на загальну продуктивність системи.

Одним із основних аспектів є також складність у вивченні та освоєнні нового інструменту. Деякі системи можуть мати високий поріг входу для новачків, вимагаючи глибшого розуміння протоколів, мов програмування або конфігураційних файлів.

Тож при виборі інструменту для розробки HTTP-сервера важливо враховувати не лише його переваги, але й мінуси, які можуть вплинути на продуктивність, швидкодію, складність налаштування та час розробки.

Крім зазначених аспектів, ще однією проблемою може бути недостатня масштабованість деяких систем. Для великих проектів з великою кількістю одночасних користувачів чи великим обсягом даних, деякі HTTP-сервери можуть виявити себе не настільки ефективними, оскільки не всі з них можуть оптимально масштабуватись горизонтально (по кількості серверів) або вертикально (по потужності кожного сервера).

Це може призвести до обмеження можливостей масштабування проекту в майбутньому.

Додатково, багатопотокова обробка запитів може бути складною у реалізації для деяких систем, особливо якщо вона не оптимізована для конкретних потреб проекту. Це може призвести до ситуацій, коли сервер не ефективно використовує ресурси або має проблеми з керуванням потоками, що може вплинути на його стабільність та продуктивність в цілому.

Не менш важливою є проблема безпеки. Якщо сервер не має вбудованих або налагоджених механізмів безпеки, це може призвести до вразливостей системи перед атаками або несанкціонованим доступом до даних, що може значно підірвати довіру до системи.

Таким чином, наряду з вибором інструменту, важливо враховувати його здатність до ефективного масштабування, оптимізації роботи з потоками, а також вбудовані механізми безпеки для забезпечення надійності та захищеності системи.

1.2. Загальний опис проєкту

Загальна концепція проекту "HTTP-сервер"

Проект "HTTP-сервер" розроблений з метою створення потужного інструменту, який здатен розпізнавати вхідні запити згідно з протоколом HTTP і формувати відповіді відповідно до цього протоколу. Основною функціональністю сервера є надання сторінок у форматі chtml, що представляють собою HTML-сторінки з можливістю додавання базових конструкцій мови C#, відповідно до вимог і вибору користувача (наприклад, студента, який вивчає програмування), що дозволяє розвивати основні навички програмування.

Крім того, система статистики, що ведеться сервером, надає адміністраторам та користувачам значні можливості. Вона дозволяє збирати дані про вхідні запити, трафік, популярність конкретних запитів чи сторінок. Це надзвичайно корисна функціональність для оптимізації ресурсів і аналізу продуктивності. Зі збиранням статистики адміністратори можуть виявити найбільш запитувані ресурси. Це дає змогу оптимізувати сервер, наприклад, кешуючи часто використовувані сторінки для зменшення навантаження на сервер. Адміністратори можуть аналізувати, як сервер

працює під навантаженням. Вони можуть виявити проблемні аспекти роботи сервера, наприклад, деякі сторінки або запити, що спричиняють затримки чи викликають помилки.

Багатопотоковий/подієвий режими, ця функціональність - ключова для сервера, оскільки вона дозволяє йому ефективно керувати багато запитів одночасно. Основні переваги цього ефективність ресурсів, швидка відповідь, скалабельність. Ця функціональність дозволяє серверу розподіляти свої ресурси ефективно, щоб обслуговувати більше запитів одночасно без втрати продуктивності. Сервер може оперативно реагувати на запити навіть при великому навантаженні, що забезпечує користувачам швидку та стабільну роботу системи. Завдяки цій роботі у багатопотоковому/подієвому режимах, сервер може легко масштабуватися для відповіді на зростаючий обсяг запитів, що робить його більш гнучким у роботі з різними навантаженнями.

Проект "HTTP-сервер" має численні переваги, але також варто врахувати його можливі недоліки. Певні HTTP-сервери можуть вимагати складних налаштувань, особливо при великому обсязі конфігураційних опцій, що може збільшити час впровадження. Якщо сервер не налаштований належним чином, він може стати об'єктом атак, таких як DDoS або інші форми кібератак. Деякі сервери можуть мати обмеження щодо одночасного обслуговування великої кількості запитів, що може призвести до перевантаження у разі інтенсивного навантаження. Деякі сервери можуть не підтримувати оновлення протоколів, що може призвести до обмежень у роботі з новими клієнтами або додатками. Підтримка багатопотокового або подієвого режиму може займати багато системних ресурсів, що потребує великої кількості пам'яті та потужності процесора. Ці недоліки можуть вплинути на продуктивність та безпеку сервера, тому важливо ретельно вивчити їх при виборі конкретного рішення.

Отже, проект "HTTP-сервер" є потужним інструментом для обробки HTTP-запитів, надання сторінок у форматі chtml з можливістю використання елементів мови C#, ведення детальної статистики та оптимізації роботи в багатопотоковому/подієвому режимах для забезпечення ефективної та надійної роботи сервера.

Основні компоненти та функції:

CHtmlController: Обробляє HTTP-запити, що стосуються генерації HTML, такі як отримання сторінки, посилання, зображення чи таблиці.

RequestTrackingContoller: Управляє запитами для відстеження, зокрема отримання всіх запитів.

IApplicationDbContext (Інтерфейс контексту бази даних): Цей інтерфейс визначає контракти, які повинен реалізувати контекст бази даних. Він містить набори даних для обробки запитів про Request, SentRequest та RequestStatistic. Його реалізація (наприклад, клас ApplicationDbContext) включає в себе логіку доступу до бази даних.

HtmlGenerations Handlers (Обробники генерації HTML-елементів): Ці обробники відповідають за обробку запитів і генерацію різних HTML-елементів, таких як зображення, посилання, сторінки та таблиці.

HtmlGenerations Queries (Класи запитів для генерації HTML-елементів): Ці класи визначають параметри для генерації різних HTML-елементів. Вони можуть містити логіку формування запитів до бази даних або отримання даних для генерації HTML.

RequestTrackings (Обробник та запит для відстеження запитів): Цей компонент відповідає за отримання інформації про всі запити, що надходять до системи. Він може збирати статистику або іншу інформацію про запити для подальшого аналізу.

HtmlGenerator Service (Сервіс генерації HTML-елементів): Цей сервіс надає методи для створення різних HTML-елементів. Його функції можуть бути використані обробниками запитів для генерації вмісту сторінок.

Domain Entities (Сутності домену): Ці сутності, такі як Request, RequestStatistic, SentRequest, відображають структуру даних вашого додатку.

HtmlElement (Value Object для HTML-елемента): Цей об'єкт-значення представляє окремий HTML-елемент і може містити дані та властивості для його представлення.

ApplicationDbContext (Реалізація контексту бази даних): Це конкретна реалізація інтерфейсу контексту бази даних (IApplicationDbContext), яка успадковує DbContext і забезпечує зв'язок з реальною базою даних.

Обробники генерації HTML:

GetHtmlImageQueryHandler

GetHtmlLinkQueryHandler

GetHtmlPageQueryHandler

GetHtmlTableQueryHandler

Обробники відстеження запитів:

GetAllRequestsQueryHandler

Request, RequestStatistic та SentRequest, використовуються для відстеження та статистики.

Використані патерни проектування:

Mediator(Посередник)

CQRS(Розділення команд і запитів)

Builder(Будівельник)

Factory Method(Фабричний метод)

Додаткові можливості проекту "HTTP-server":

Кожна з цих можливостей може бути додатково розширена або покращена для забезпечення ще більш широкого функціоналу в рамках вашого проекту "HTTP-server". Ось додаткові можливості, які можна розглянути для розширення функціоналу вашого сервера:

Шаблонізація HTML: Додайте можливість використання шаблонів для створення більш складних HTML-сторінок, що дозволяє вам використовувати стандартні частини коду для забезпечення більшої повторюваності та зручності у розробці.

Розширені можливості маршрутизації: Додайте можливість встановлювати більш складні правила маршрутизації, наприклад, засновані на шаблонах URL або параметрах запиту, для кращого управління тим, як обробляються та генеруються сторінки.

Підтримка API: Додайте можливість обробки запитів з метою не лише генерації HTML, але й надання даних у форматі JSON або інших форматах, що дозволить використовувати ваш сервер як основу для розробки API.

Аутентифікація та авторизація: Вбудуйте систему авторизації, щоб керувати доступом до різних ресурсів на сервері, забезпечуючи безпеку та контроль доступу.

Кешування та оптимізація: Додайте можливості кешування для збереження попередньо згенерованих сторінок або ресурсів, що дозволить підвищити швидкодію відповідей сервера.

Логування та аналітика: Розширте можливості відстеження запитів для збереження подій, відмічення помилок або збору аналітики для подальшого аналізу та вдосконалення процесів роботи сервера.

1.3. Вимоги до застосунків системи

1.3.1. Функціональні вимоги до системи

Розпізнавання вхідних запитів і формування відповідей згідно з протоколом HTTP:

Сервер повинен мати можливість отримувати HTTP-запити від клієнтів і розпізнавати їх.

Потім формувати коректні відповіді на ці запити, що відповідають стандартам протоколу HTTP (наприклад, відповіді з кодами стану, заголовками тощо).

Надання сторінок chtml (html з додаванням простих C# конструкцій):

Сервер має підтримувати створення HTML-сторінок з можливістю додавання простих конструкцій мови C# для генерації динамічного вмісту на веб-сторінках.

Ведення статистики вхідних запитів:

Сервер повинен збирати та зберігати статистику про вхідні запити, можливо, такі дані, як типи запитів, час обробки, рівень навантаження сервера тощо.

Обробка запитів у багатопотоковому/подієвому режимах:

Для покращення продуктивності та можливості обробки багато запитів одночасно, сервер повинен підтримувати роботу у багатопотоковому або подійному режимах.

1.3.2. Нефункціональні вимоги до системи

Продуктивність:

Час відгуку: Максимальний час, за який сервер повинен відповідати на запити.

Масштабованість: Можливість сервера працювати ефективно при збільшенні навантаження.

Продуктивність багатопотоковості/подій: Ефективність обробки запитів у багатопотоковому/подійному середовищі без перевантаження.

Безпека:

Захист від атак: Захист сервера від різних видів атак, таких як SQL-ін’єкції, переповнення буфера, XSS тощо.

Аутентифікація і авторизація: Механізми перевірки доступу до ресурсів сервера.

Надійність:

Стабільність: Здатність сервера працювати без збоїв протягом тривалого періоду часу.

Відновлення після збоїв: Механізми автоматичного відновлення роботи після виникнення помилок або відмов.

Швидкодія:

Оптимізація ресурсів: Мінімізація використання пам’яті та обчислювальних ресурсів.

Сумісність:

Сумісність з різними браузерами: Забезпечення коректної роботи сервера на різних веб-платформах та браузерах.

Документація та підтримка:

Документація: Наявність чіткої та зрозумілої документації для користувачів та розробників.

Підтримка: Готовність надавати підтримку щодо виниклих проблем з сервером.

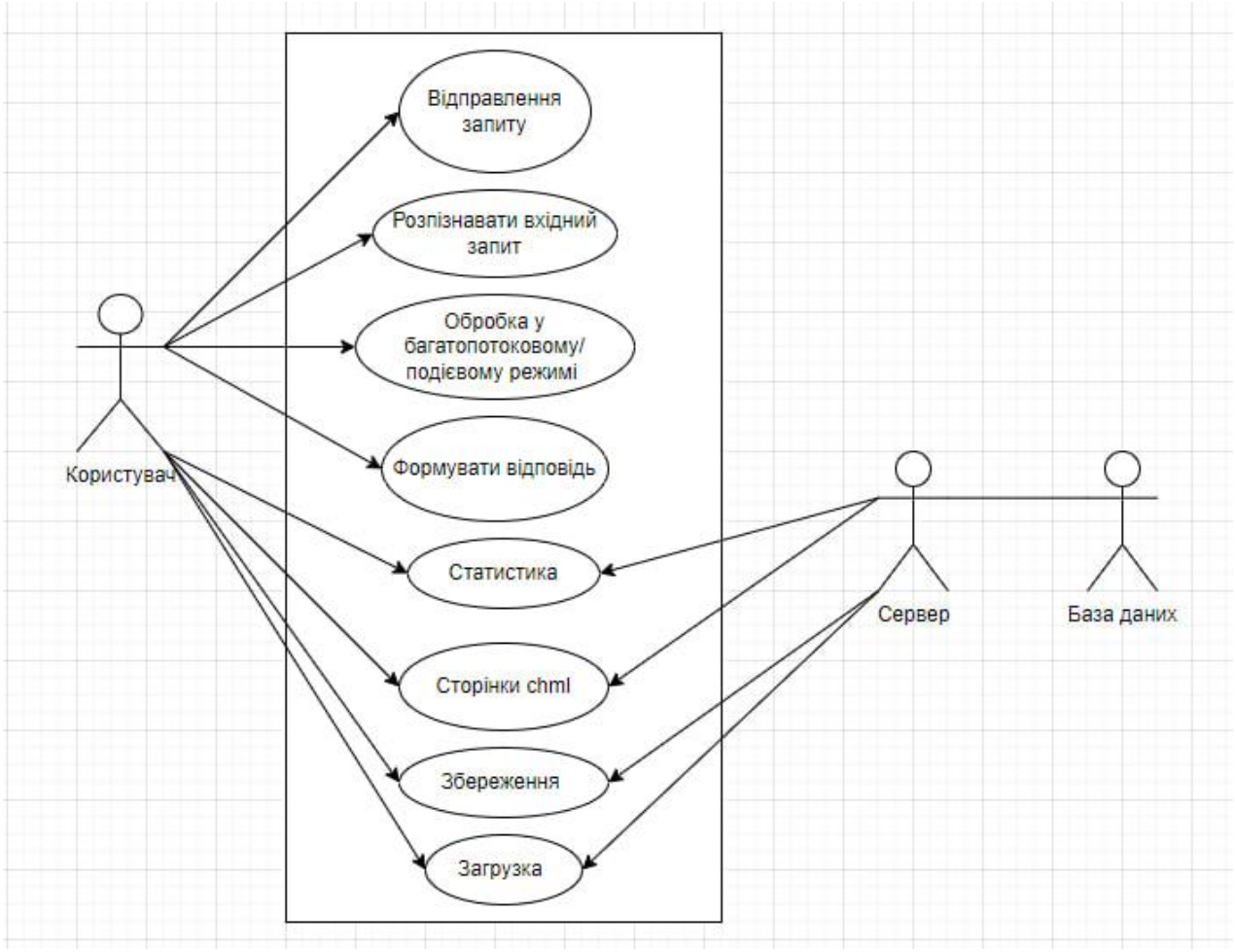
Розширюваність:

Модульність: Можливість легко розширювати та модифікувати функціонал сервера шляхом додавання нових модулів чи компонентів.

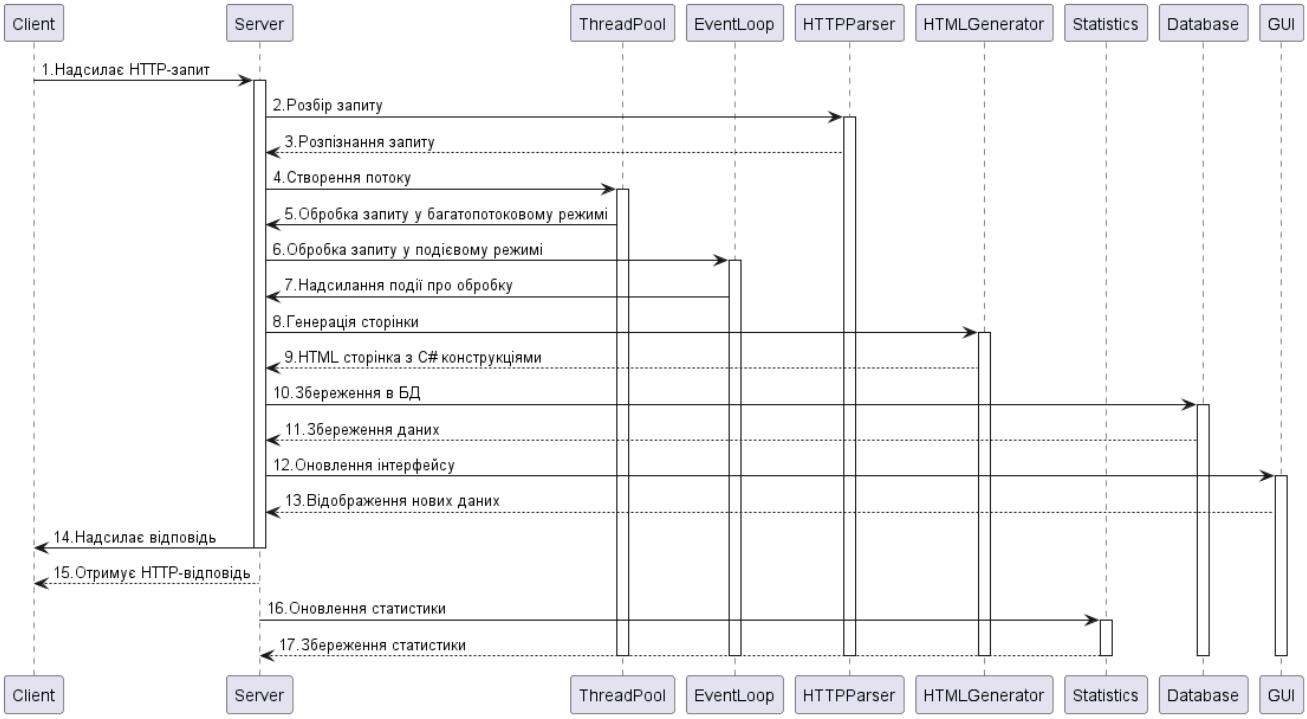
Відмовостійкість:

Мінімізація впливу помилок: Здатність сервера працювати навіть при виникненні помилок чи несправностей.

1.4. Сценарії використання системи



(Рис. 1.4.1 – Use Case diagram)



(Рис. 1.4.2 - Sequences diagram)

Сценарій використання системи "HTTP-server" починається з моменту, коли користувач запускає програму:

1) **HTTP-запит від клієнта:**

Коли клієнт відправляє запит на сервер, сервер активується та починає обробку цього запиту.

2) **HTTPParser:**

Отримавши запит, сервер використовує HTTPParser для розбору запиту, щоб зрозуміти його.

3) **Створення ThreadPool:**

Сервер створює пул потоків (ThreadPool), який використовується для обробки запитів у багатопотоковому режимі, що дозволяє обробляти кілька запитів одночасно.

4) **Багатопотокова обробка:**

Запускаються потоки з ThreadPool для обробки запиту. Це дозволяє серверу обробляти запити паралельно.

5) **Подієвий режим через EventLoop:**

Одночасно з обробкою у багатопотоковому режимі, сервер також може використовувати подієвий режим за допомогою EventLoop. EventLoop допомагає обробляти події асинхронно.

6) **HTML генерація через HTMLGenerator:**

Після обробки запиту, сервер викликає HTMLGenerator для генерації HTML-сторінки, використовуючи C# конструкції.

7) **Збереження в БД:**

Сформована HTML-сторінка зберігається в базі даних через взаємодію з Database.

8) **Оновлення GUI:**

Після збереження даних в БД, сервер оновлює інтерфейс користувача (GUI), щоб відобразити нові дані.

9) **Відправлення відповіді клієнту:**

Нарешті, сервер відправляє відповідь на HTTP-запит клієнту, повертаючи результат обробки.

10) **Оновлення статистики через Statistics:**

Крім того, сервер оновлює статистику через взаємодію з об'єктом Statistics, що може включати збір та оновлення даних про використання сервера або інші параметри.

1.5. Концептуальна модель системи

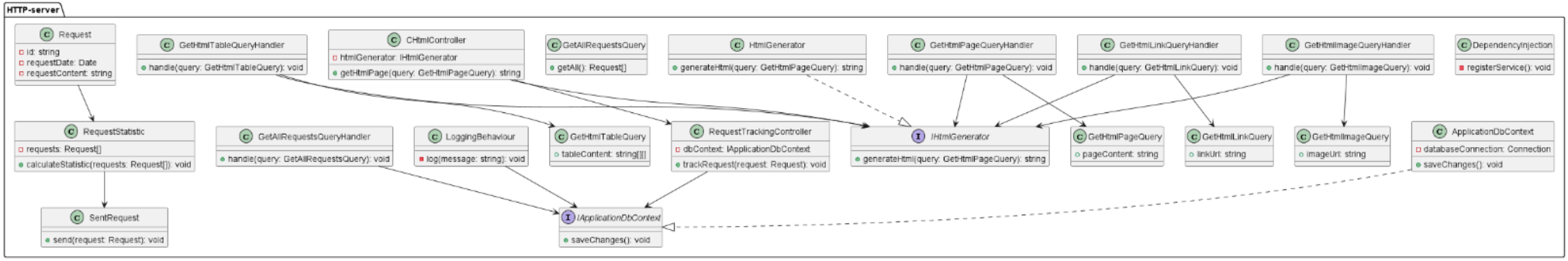


Рис. 1.5.1 - Classes diagram

1) **LoggingBehaviour:**

Відповідає за логування повідомлень. Має метод log, який приймає повідомлення у вигляді рядка.

2) **IApplicationDbContext:**

Інтерфейс, який описує контракт для збереження змін в додатковому контексті додатка.

3) **DependencyInjection:**

Клас, який, здається, відповідає за реєстрацію сервісів (ін'єкцію залежностей).

4) **GetHtmlImageQueryHandler,GetHtmlLinkQueryHandler, GetHtmlPageQueryHandler, GetHtmlTableQueryHandler:**

Класи, що обробляють певні запити для отримання зображення, посилання, сторінки HTML або таблиці HTML відповідно. Вони здаються мати методи handle, які обробляють відповідні запити.

5) **GetHtmlImageQuery,GetHtmlLinkQuery,GetHtmlPageQuery, GetHtmlTableQuery:**

Класи-запити з параметрами, які використовуються для передачі даних для відповідних запитів.

6) **GetAllRequestsQueryHandler:**

Клас, який обробляє запити для отримання всіх запитів. Має метод handle.

7) **HtmlGenerator, IHtmlGenerator:**

Клас та його інтерфейс, які відповідають за генерацію HTML-коду. Метод generateHtml приймає запит типу GetHtmlPageQuery та повертає рядок з HTML-кодом.

8) **Request, RequestStatistic, SentRequest:**

Класи, які, здається, відносяться до відстеження та обробки запитів.

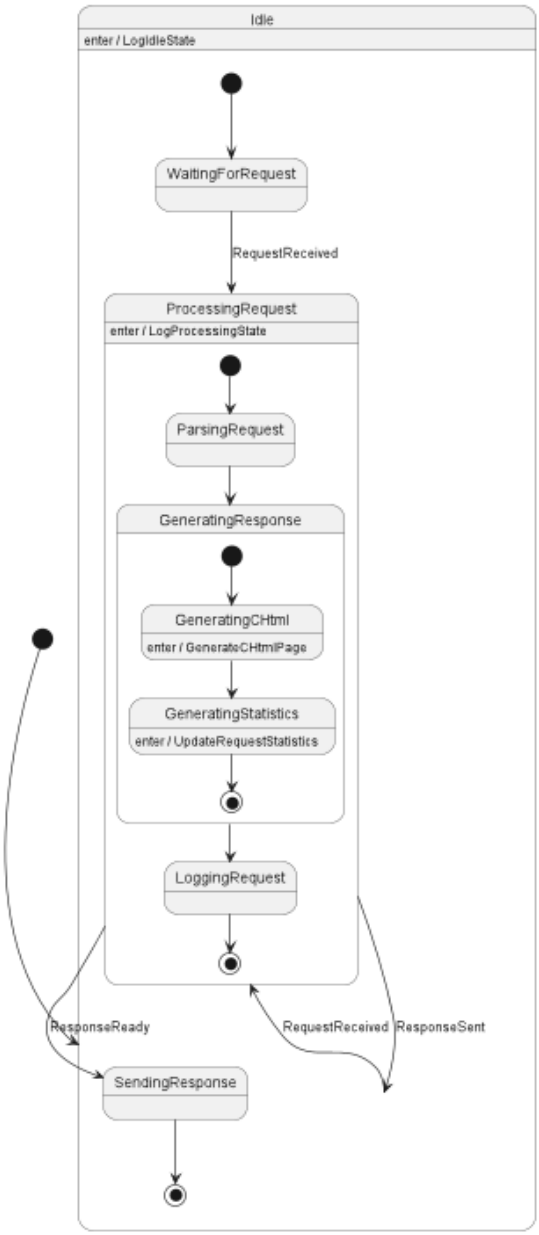
RequestStatistic обчислює статистику, SentRequest відповідає за відправку запитів.

9) **ApplicationDbContext:**

Клас, який, здається, відповідає за роботу з базою даних. Має метод saveChanges, який, очевидно, зберігає зміни.

10) **CHtmlController, RequestTrackingController:**

Контролери, які, здається, керують різними аспектами додатку. CHtmlController отримує HTML-сторінку через IHtmlGenerator, а RequestTrackingController відстежує запити через IApplicationDbContext.



(Рис. 1.5.2 - State diagram)

1) **Стан Idle (Простій):**

Цей стан означає, що сервер знаходиться в режимі очікування і готовий приймати запити. В цьому стані сервер не обробляє активних запитів і чекає на нові вхідні дані від клієнтів.

2) **Стан WaitingForRequest (Очікування запиту) підстатус стану "Idle":**

Сервер активно слухає мережу, чекаючи на вхідні запити від клієнтів.

3) **Стан ProcessingRequest (Обробка запиту):**

Сервер переходить у цей стан, коли отримує запит від клієнта. В цьому стані сервер аналізує запит, генерує відповідь та виконує необхідні дії, які вимагаються для обробки запиту.

4) **Стан ParsingRequest (Розбір запиту) підстатус стану "ProcessingRequest":**

Сервер аналізує вхідний запит, розбираючи його структуру та параметри для визначення подальших дій.

5) **Стан GeneratingResponse (Генерація відповіді) підстатус стану "ProcessingRequest":**

На цьому етапі сервер формує відповідь на запит, яка може включати генерацію chtml сторінок або інших даних.

6) **Стан LoggingRequest (Запис запиту) підстатус стану "ProcessingRequest":**

Сервер записує інформацію про запит у журнал для ведення статистики або аналітики.

7) **Стан SendingResponse (Відправлення відповіді):**

Сервер відправляє сформовану відповідь назад клієнту. Після відправлення відповіді сервер може повернутися до стану "Idle", готовий приймати наступний запит.

8) **Стан GeneratingCHtml (Генерація chtml) підстатус стану "GeneratingResponse":**

Сервер генерує chtml сторінки, які можуть включати елементи C#.

9) **Стан GeneratingStatistics (Генерація статистики) підстатус стану "GeneratingResponse":**

Сервер оновлює статистику вхідних запитів, збираючи дані про кількість та типи запитів.

Ці стани відображають загальний життєвий цикл HTTP-сервера від моменту очікування запиту до моменту відправлення відповіді та повернення до стану очікування.

1.6. Вибір Бази Даних

Вибір SQL Server Object Explorer у Visual Studio як системи управління базами даних (СУБД) для зберігання даних відправлених запитів та інформації про статистики системи може бути обґрунтований наступними причинами:

1. ***Візуалізація структури бази даних:***

SSOX дозволяє візуально оглядати та аналізувати структуру бази даних через графічний інтерфейс. Це включає:

- *Діаграми баз даних:* Схематичне відображення взаємозв'язків між таблицями, схемами, збереженими процедурами та іншими об'єктами, полегшуючи розуміння їхніх зв'язків.
- *Візуальне моделювання:* Можливість створення та модифікації структури бази даних через графічний інтерфейс, що дозволяє швидко розробляти та аналізувати схеми.

2. ***Підтримка версій баз даних:***

SSOX забезпечує сумісність з різними версіями SQL Server. Це важливо для:

- *Роботи з різними версіями сервера:* Дозволяє розробникам працювати з різними версіями SQL Server, зберігаючи сумісність і продуктивність.
- *Відновлення даних:* Підтримка різних версій спрощує процес відновлення даних на різних серверах.

3. ***Інтегрованість з іншими інструментами Visual Studio:***

SSOX гармонійно поєднується з іншими інструментами розробки у Visual Studio:

- *Загальний інтерфейс:* Інтеграція з іншими інструментами розробки (наприклад, з системою контролю версій) полегшує роботу та підвищує ефективність.

4. ***Синхронізація з кодом програм:***

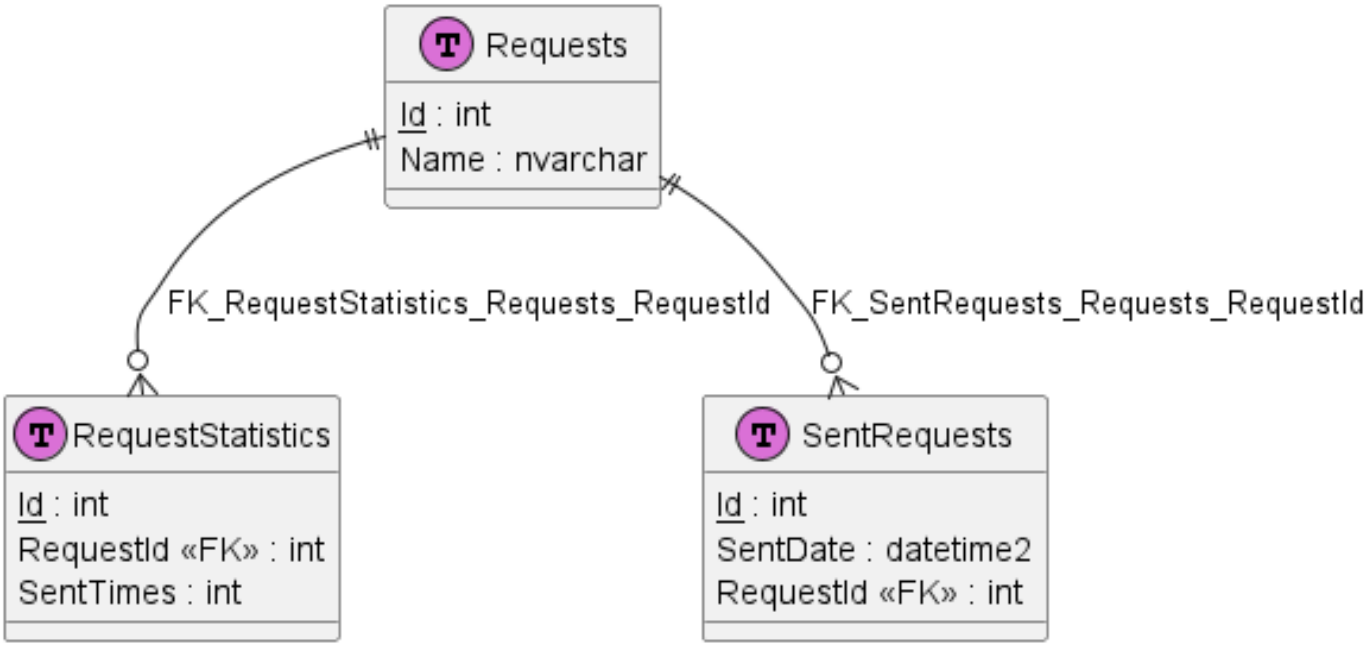
SSOX дозволяє впроваджувати зміни у структуру бази даних, синхронізуючи їх з кодом програм:

- *Інтеграція з проектами:* Зміни в структурі бази даних автоматично відображаються у відповідних програмних кодах, що спрощує розробку.

5. ***Розширені можливості налагодження:***

SSOX надає розширені можливості для налагодження SQL-запитів та бази даних:

- ***Інтерактивний інтерфейс налагодження:*** Дозволяє відлагоджувати складні запити та оптимізувати їх безпосередньо через інтерфейс.



(Рис. 1.6.1 – RequestDB)

Ми використовуємо БД , щоб зберігати і відстежувати різні запити, їх статистику та інформацію про відправлення цих запитів. Такі дані можна використовувати для аналізу виконання запитів, відстеження кількості відправлень тощо.

1.7. Вибір мови програмування та середовища розробки

Мова програмування: C#

C# було обрано як основну мову програмування для цього проекту з наступних причин:

1) ***Платформна універсальність:*** C# використовується в основному для розробки під платформу .NET, що дає змогу створювати різноманітні додатки, включаючи веб-додатки, десктопні програми, мобільні застосунки та навіть ігри.

2) ***Ефективність і продуктивність:*** Мова C# є високорівневою, що дозволяє швидко писати код, а вбудована збірка сміття допомагає управляти пам'яттю, спрощуючи роботу розробників.

3) ***Широкі можливості бібліотек і фреймворків:*** Екосистема .NET має багато готових бібліотек і фреймворків, які спрощують розробку, забезпечуючи готові рішення для різних задач.

4) ***Велика спільнота:*** C# має велику та активну спільноту розробників, що означає доступність допомоги, ресурсів і відповідей на питання.

5) ***Інструменти розробки:*** Visual Studio, основне середовище розробки для C#, надає широкий набір інструментів для роботи з цією мовою, включаючи відлагоджувач, аналізатори коду, підтримку рефакторингу та інше.

6) ***Підтримка популярних технологій:*** C# і .NET підтримують такі сучасні технології, як ASP.NET для веб-розробки, Xamarin для мобільних додатків та Unity для розробки ігор.

Середовище розробки: Visual Studio

Visual Studio було обрано як основне середовище розробки з таких причин:

1) ***Широкі можливості:*** Visual Studio підтримує багато мов програмування, серед яких C#, C++, Python, JavaScript, і багато інших. Це дозволяє розробникам працювати з різними технологіями в одному середовищі.

2) ***Інтегроване середовище розробки (IDE):*** Visual Studio має потужне інтегроване середовище, яке включає в себе редактор коду, відлагоджувач, систему керування версіями, інструменти для тестування, профілювання та багато іншого. Всі ці функції в одній програмі забезпечують зручність та продуктивність для розробників.

3) ***Підтримка:*** Visual Studio має велику спільноту користувачів, що робить підтримку та пошук відповідей на питання чи проблеми більш доступними.

4) ***Широкі можливості розширення:*** Visual Studio має велику кількість розширень, які дозволяють налаштувати робоче середовище для конкретних потреб розробника.

5) ***Інструменти для командної роботи:*** Величезний набір інструментів для роботи в команді, включаючи можливості інтеграції з різними системами керування версіями, спільної роботи над проектами тощо.

1.8. Проєктування розгортання системи

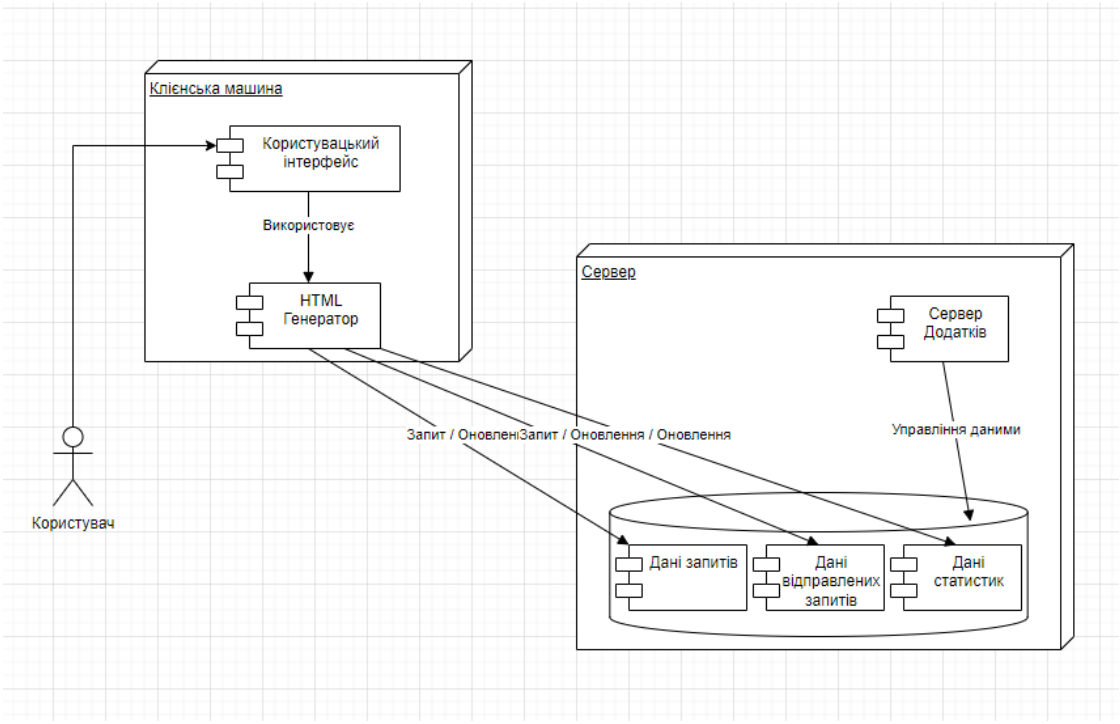
На клієнтській стороні розташований користувацький інтерфейс, що є основним входом для користувачів. Вони можуть взаємодіяти з програмою через цей інтерфейс. Також є HTML-генератор, який дозволяє користувачам створювати, редагувати та управляти файлами у форматі Chml. На серверній стороні, знаходиться логіка обробки цих файлів та забезпечення відповідей на запити користувачів. Ця діаграма розгортання відображає взаємодію між клієнтською та серверною частинами системи, показуючи, як вони співпрацюють для забезпечення функціональності програми та роботи з Chml-файлами.

На серверній стороні ми маємо базу даних, яка поділена на дві основні частини: дані запитів, дані відправлених запитів та дані статистик. Дані запитів(Requests) ця таблиця зберігає дані про запити. Кожен запис має унікальний ідентифікатор та ім'я. Ймовірно, це може бути список запитів або завдань, які потрібно виконати. Дані статистики запитів(RequestStatistics) тут зберігається статистика про кількість разів, коли запит з таблиці "Requests" був відправлений. Ця таблиця містить ідентифікатор для кожного запису, посилання на конкретний запит, та кількість відправлень. Зв'язок із таблицею "Requests" здійснюється через зовнішній ключ.. Дані запити що відправлено серверу(SentRequests), таблиця відображає дані про відправлені запити з таблиці "Requests". Кожен запис має унікальний ідентифікатор, дату відправлення і посилання на конкретний запит. Ця таблиця також має зовнішній ключ, який посилається на таблицю "Requests". Ці три частини бази даних взаємодіють з HTTP-server, дозволяючи зберігати та отримувати необхідну інформацію.

Сервер додатків представляє собою ключовий компонент на сервері, що керує взаємодією між базою даних та клієнтськими машинами. Він обробляє запити, що надходять від HTML-генератора, виконує необхідні операції з даними та повертає результати назад до HTML-генератора. Це забезпечує централізоване управління даними та високий рівень інтеграції між різними частинами системи.Цей сервер виконує роль посередника між фронтендом (HTML-генератором) та базою даних. Він отримує запити від користувачів через інтерфейс веб-сторінки, обробляє їх, взаємодіє з базою даних для отримання/збереження необхідної інформації та повертає результати у вигляді даних або HTML-сторінок.

Ця діаграма розгортання ілюструє взаємодію між різними компонентами системи, вказуючи на їх спільну роботу для забезпечення функціональності програми. Вона демонструє, як користувацький інтерфейс та HTML-генератор на клієнтській машині інтегруються з серверними компонентами, такими як база даних та сервер додатків, для забезпечення ефективного управління даними та взаємодії з користувачем.

Діаграма є важливим інструментом для візуалізації взаємодії та розташування компонентів системи, дозволяючи краще зрозуміти, як вони взаємодіють між собою та як кожен компонент виконує свої функції для забезпечення роботи програми.



(Рис. 1.8.1 - Deployment diagra

2 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

2.1. Структура бази даних

База даних проекту "HTTP-server" складається з таблиць: Requests, RequestStatistics та SentRequests. Ці таблиці взаємопов'язані та служать для зберігання інформації про запити та їхні статистику відповідно.

Таблиця Requests

Таблиця Requests призначена для зберігання інформації про окремі запити, що надходять на сервер.

Структура таблиці Requests наступна:

Id - це унікальний ідентифікатор кожного запиту, встановлений як ціле число (INT). Він є обов'язковим для кожного запису та є первинним ключем (PRIMARY KEY), що дозволяє однозначно ідентифікувати кожен запит.

Name - це поле, що містить ім'я або назву запиту. Тип даних NVARCHAR(MAX) дозволяє зберігати текстові дані задовільної довжини. Це поле вказує на характеристики або назву кожного запиту.

Це основна структура таблиці "Requests", яка дозволяє зберігати унікальні ідентифікатори та імена запитів у вашій базі даних.

Таблиця RequestStatistics

Таблиця RequestStatistics пов'язана з таблицею "Requests" і зберігає статистику про кожен збережений запит.

Структура таблиці RequestStatistics така:

Id - це унікальний ідентифікатор кожного запису у таблиці статистики, встановлений як ціле число (INT). Він є обов'язковим для кожного рядка таблиці та є первинним ключем (PRIMARY KEY).

RequestId - це поле, яке посилається на унікальний ідентифікатор запиту у таблиці "Requests". Це поле встановлює зв'язок між статистикою та конкретним запитом.

SentTimes - це поле, яке містить інформацію про кількість разів, коли даний запит був відправлений. Воно представлене як ціле число (INT) і використовується для зберігання статистики.

Така структура таблиці "RequestStatistics" дозволяє зберігати інформацію про кількість відправлень для кожного конкретного запиту, зв'язуючи ці дані з таблицею "Requests".

Таблиця SentRequests

Таблиця SentRequests ця таблиця відстежує інформацію про запити, які були відправлені.

Структура таблиці SentRequests така:

Id: Це унікальний ідентифікатор кожного запису у таблиці, який автоматично збільшується за допомогою ключового слова IDENTITY. Він використовується як первинний ключ (PRIMARY KEY) таблиці.

SentDate: Поле для зберігання дати та часу відправлення запиту, використовує тип даних DATETIME2 (7).

RequestId: Це поле, що посилається на ідентифікатор запиту у таблиці "Requests" через зовнішній ключ. Воно вказує на те, який саме запит був відправлений, забезпечуючи зв'язок між цією таблицею та таблицею запитів.

Також ви створюєте некластеризований індекс (NONCLUSTERED INDEX) на поле RequestId, що може бути корисним для оптимізації швидкості виконання певних запитів, особливо при частому пошуку за цим полем.

Ця структура дозволяє зберігати інформацію про відправлені запити, включаючи дату та час їхнього відправлення, і пов'язує ці дані з конкретними запитами, які зберігаються у таблиці "Requests".

Запити для створення таблиць в БД:

```
CREATE TABLE [dbo].[Requests] (
    [Id]   INT            IDENTITY (1, 1) NOT NULL,
    [Name] NVARCHAR (MAX) NOT NULL,
    CONSTRAINT [PK_Requests] PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

```
CREATE TABLE [dbo].[RequestStatistics] (
    [Id]        INT IDENTITY (1, 1) NOT NULL,
    [RequestId] INT NOT NULL,
    [SentTimes] INT NOT NULL,
    CONSTRAINT [PK_RequestStatistics] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_RequestStatistics_Requests_RequestId] FOREIGN KEY
([RequestId]) REFERENCES [dbo].[Requests] ([Id]) ON DELETE CASCADE
);

GO
CREATE NONCLUSTERED INDEX [IX_RequestStatistics_RequestId]
    ON [dbo].[RequestStatistics]([RequestId] ASC);

CREATE TABLE [dbo].[SentRequests] (
    [Id]        INT           IDENTITY (1, 1) NOT NULL,
    [SentDate]  DATETIME2 (7) NOT NULL,
    [RequestId] INT           NOT NULL,
    CONSTRAINT [PK_SentRequests] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_SentRequests_Requests_RequestId] FOREIGN KEY ([RequestId])
REFERENCES [dbo].[Requests] ([Id]) ON DELETE CASCADE
);

GO
CREATE NONCLUSTERED INDEX [IX_SentRequests_RequestId]
    ON [dbo].[SentRequests]([RequestId] ASC);
```

2.2. Архітектура системи

2.2.1. Специфікація системи

HTTP-сервер спроектований для ефективного розпізнавання та обробки вхідних HTTP-запитів, що дозволяє йому формувати відповіді, що відповідають стандартам протоколу HTTP. Окрім цього, сервер забезпечує

можливість відображення сторінок chtml, де студент може використовувати найпростіші конструкції мови C# для додавання функціональності до HTML-сторінок.

Окремо, він здатний збирати та аналізувати статистику вхідних запитів, що дозволяє відстежувати та оцінювати обсяг роботи сервера. Додатково, сервер обробляє запити у багатопотоковому або подійному режимах, що сприяє підвищенню ефективності та швидкодії обробки запитів для більш оптимальної роботи системи.

<u>Система включає в себе наступні ключові компоненти:</u>

Графічний Інтерфейс Користувача (GUI)

- *Відображення сторінок chtml:* GUI відповідає за відображення HTML-сторінок з додаванням C# конструкцій. Він може мати в собі інструменти для редагування, візуального створення та відображення сторінок, згенерованих серверною частиною.
- *Взаємодія з користувачем:* GUI надає можливість користувачам взаємодіяти з системою через інтерфейс: введення даних, навігація по сторінках, відправлення запитів на серверну частину тощо.
- *Відображення статистики та результатів:* GUI може також відображати статистику, яку збирає серверна частина, та результати обробки запитів, отримані від сервера.

Серверна Частина:

- *Обробка запитів та відправка відповідей:* Серверна частина відповідає за приймання вхідних запитів від GUI або інших клієнтів, їх обробку та генерацію відповідей згідно з протоколом HTTP.
- *Генерація сторінок chtml:* Ця частина системи формує сторінки chtml, вставляючи C# конструкції за необхідності для динамічної генерації контенту.
- *Збір та аналіз статистики:* Серверна частина веде облік вхідних запитів, збирає та аналізує статистику для подальшого використання.

Основні Функції Системи:

- *Обробка HTTP-запитів:* Система повинна розпізнавати та обробляти HTTP-запити, забезпечуючи відповідні відповіді відповідно до протоколу HTTP.
- *Генерація сторінок chtml:* Сервер має здатність створювати HTML-сторінки з використанням найпростіших конструкцій мови C#. Це включає в себе можливість динамічної генерації вмісту для надання користувачам більш персоналізованого доступу.
- *Збір та аналіз статистики запитів:* Система повинна вести облік вхідних запитів, зберігати статистику та надавати можливість аналізу даних про типи запитів, кількість запитів, IP-адреси користувачів тощо.
- *Багатопотокова/подійна обробка:* Сервер повинен мати можливість ефективно обробляти багато запитів одночасно за допомогою багатопотоковості або подійної моделі, забезпечуючи швидку та надійну відповідь на запити користувачів.
- *Забезпечення відповідності протоколу HTTP:* Система повинна виконувати усі вимоги стандарту HTTP для забезпечення сумісності з різними клієнтами та стабільної роботи системи в мережевому середовищі.
- *Безпечний Доступ та Управління Даними:* Система забезпечує високий рівень безпеки, використовуючи сучасні методи шифрування та аутентифікації.

2.2.2. Вибір та обґрунтування патернів реалізації

При розробці системи HTTP-server було використано кілька ключових патернів проектування, які забезпечують ефективність, гнучкість та масштабованість системи.

Основні використані патерни:

1) Mediator (Посередник):

Розділення відповідальності: Медіатор дозволяє зменшити залежність між компонентами системи, спрощуючи спосіб їх взаємодії. У вас GUI та серверна частина можуть комунікувати через цей посередник, що полегшує розширення та модифікацію системи без великих змін у внутрішній логіці кожного компонента.

2) CQRS (Розділення команд і запитів):

Оптимізація функціональності: Цей паттерн дозволяє краще розподілити завдання читання та запису. У вас це може бути корисно для розділення логіки, коли користувачі отримують вміст сторінок та надсилають команди для оновлення чи збереження даних.

3) Builder (Будівельник):

Генерація складних сторінок: Вибір цього паттерну дозволяє поетапно створювати складні об'єкти, такі як сторінки chtml з додаванням C# конструкцій. Це дозволяє вам гнучко формувати вміст сторінок та додавати необхідні елементи відповідно до потреб системи.

4) Factory Method (Фабричний метод):

Динамічне створення об'єктів: Фабричний метод може допомогти вам створювати об'єкти різних типів сторінок chtml в залежності від потреб системи. Це полегшує роботу зі створенням різноманітних сторінок та дозволяє швидко адаптувати систему до нових вимог.

Обґрунтування вибору патернів

Обрані патерни проектування - Mediator, CQRS, Builder, та Factory Method відіграють ключову роль у покращенні архітектури системи HTTP-сервера з GUI. Вони сприяють підвищенню рівня гнучкості, підтримуваності та розширюваності системи. Вони допомагають підвищити рівень абстракції та спрощують співпрацю між компонентами системи, що робить код більш гнучким, розширюваним та підтримуваним. Обрані патерни покликані оптимізувати розробку та забезпечити більшу прозорість та ефективність системи HTTP-сервера з GUI.

2.3. Інструкція користувача

Запуск сервера бази даних RequestDB

Для коректної роботи системи HTTP-server необхідно запустити сервер бази даних RequestDB. На сервері потрібно створити наступні таблиці:

Таблиця Requests - для зберігання запитів:

```
CREATE TABLE [dbo].[Requests] (
    [Id]   INT           IDENTITY (1, 1) NOT NULL,
    [Name] NVARCHAR (MAX) NOT NULL,
    CONSTRAINT [PK_Requests] PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Таблиця RequestStatistics – для статистики запитів:

```
CREATE TABLE [dbo].[RequestStatistics] (
    [Id]        INT IDENTITY (1, 1) NOT NULL,
    [RequestId] INT NOT NULL,
    [SentTimes] INT NOT NULL,
    CONSTRAINT [PK_RequestStatistics] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_RequestStatistics_Requests_RequestId] FOREIGN KEY
([RequestId]) REFERENCES [dbo].[Requests] ([Id]) ON DELETE CASCADE
);

GO
CREATE NONCLUSTERED INDEX [IX_RequestStatistics_RequestId]
    ON [dbo].[RequestStatistics]([RequestId] ASC);
```

Таблиця SentRequests - для зберігання запитів:

```
CREATE TABLE [dbo].[SentRequests] (
    [Id]        INT           IDENTITY (1, 1) NOT NULL,
    [SentDate]  DATETIME2 (7) NOT NULL,
    [RequestId] INT           NOT NULL,
    CONSTRAINT [PK_SentRequests] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_SentRequests_Requests_RequestId] FOREIGN KEY ([RequestId])
REFERENCES [dbo].[Requests] ([Id]) ON DELETE CASCADE
);
```

```
GO
CREATE NONCLUSTERED INDEX [IX_SentRequests_RequestId]
    ON [dbo].[SentRequests]([RequestId] ASC);
```

Налаштування підключення до бази даних

У SQL Server Object Explorer вибираю наш сервер баз даних “RequestDB”.
Нажимаємо на саму базу даних “RequestDB” та обираємо "Connect".
Вводимо облікові дані (ім'я користувача, пароль) для доступу до бази даних “RequestDB”.

Запуск

Після підключення до бази даних “RequestDB” ми виконуємо запити SQL, підлагоджуємо, оновлюємо та інше.
Для запуску можливої роботи з базою даних нам потрібно створити та виконати SQL-запити або робити це через код програми, за допомогою C#.

Робота із запитами

Створення запиту

- Формулювання запитів: Розробка або використання інструментів, що дозволяють користувачам формулювати HTTP-запити для отримання сторінок чи даних.
- Взаємодія з сервером: Забезпечення можливості коректної взаємодії з сервером через коректні URL-адреси та передачу параметрів запитів.

Редагування запиту

- Маніпулювання параметрами: Дозвіл користувачам змінювати параметри запитів, щоб отримувати різні дані або сторінки.
- Зміна типу запиту: Можливість вибирати різні типи запитів та обробляти їх коректно.

Зберігання запиту

- Зберігання журналу запитів: Ведення журналу всіх вхідних запитів для подальшого аналізу, включаючи дані про час, тип запиту, URL, параметри тощо.
- Можливість відновлення: Забезпечення можливості перегляду або відновлення попередніх запитів для з ручного повторення дій.

Створення статистики про запити

- Аналіз журналу запитів: Оброблення зібраних даних для створення звіту або графіків щодо типів запитів, їхньої кількості, часу виконання тощо.

ВИСНОВКИ

Під час розробки HTTP-сервера, кожен крок був направлений на створення високоефективної, гнучкої та надійної платформи для обробки запитів за протоколом HTTP. Інтеграція шаблонів проектування та використання бази даних у SQL Server Object Explorer у Visual Studio стали основними каменями в цьому процесі.Кожен використаний шаблон та інструмент виявився ключовим у досягненні поставлених цілей, що дозволило створити не лише сервер, здатний ефективно обробляти запити та формувати відповіді, але й платформу, що надає гнучкість, легкість налаштування та надійність у різних умовах використання.Дозвольте подивитися на досягнення цього проекту ближче, поглибитися у ключові аспекти та переваги, які він приніс у розробці веб-систем та забезпечивши високу якість обробки запитів у сучасному веб-середовищі.

Гнучкість та легкість налаштування завдяки Builder та Factory Method. Використання цих шаблонів проектування дозволило не лише ефективно розпізнавати вхідні запити та формувати відповіді, а й створило можливість швидко міняти та налаштовувати поведінку сервера за потреби.Покращена обробка запитів завдяки Mediator та CQRS. Ці підходи оптимізували роботу з багатопотоковими та подійними запитами, забезпечуючи ефективну взаємодію між компонентами серверу та швидку обробку даних.Гнучкість у розробці веб-сторінок через chtml із C# конструкціями. Можливість додавати найпростіші конструкції мови C# у chtml сторінки дозволяє студентам та користувачам легко розширювати функціонал веб-сторінок, забезпечуючи гнучкість у їхньому створенні та адаптації під власні потреби.Надійність та доступність даних через SQL Server Object Explorer. Інтеграція з SQL Server Object Explorer у Visual Studio дозволила надійно зберігати та швидко отримувати важливу статистику вхідних запитів, забезпечуючи високий рівень доступності до даних. Розширення можливостей сервера та підтримка. Завдяки використанню цих підходів та інструментів, сервер стає готовим до розширення функціоналу та легкої інтеграції з майбутніми технологіями та системами.

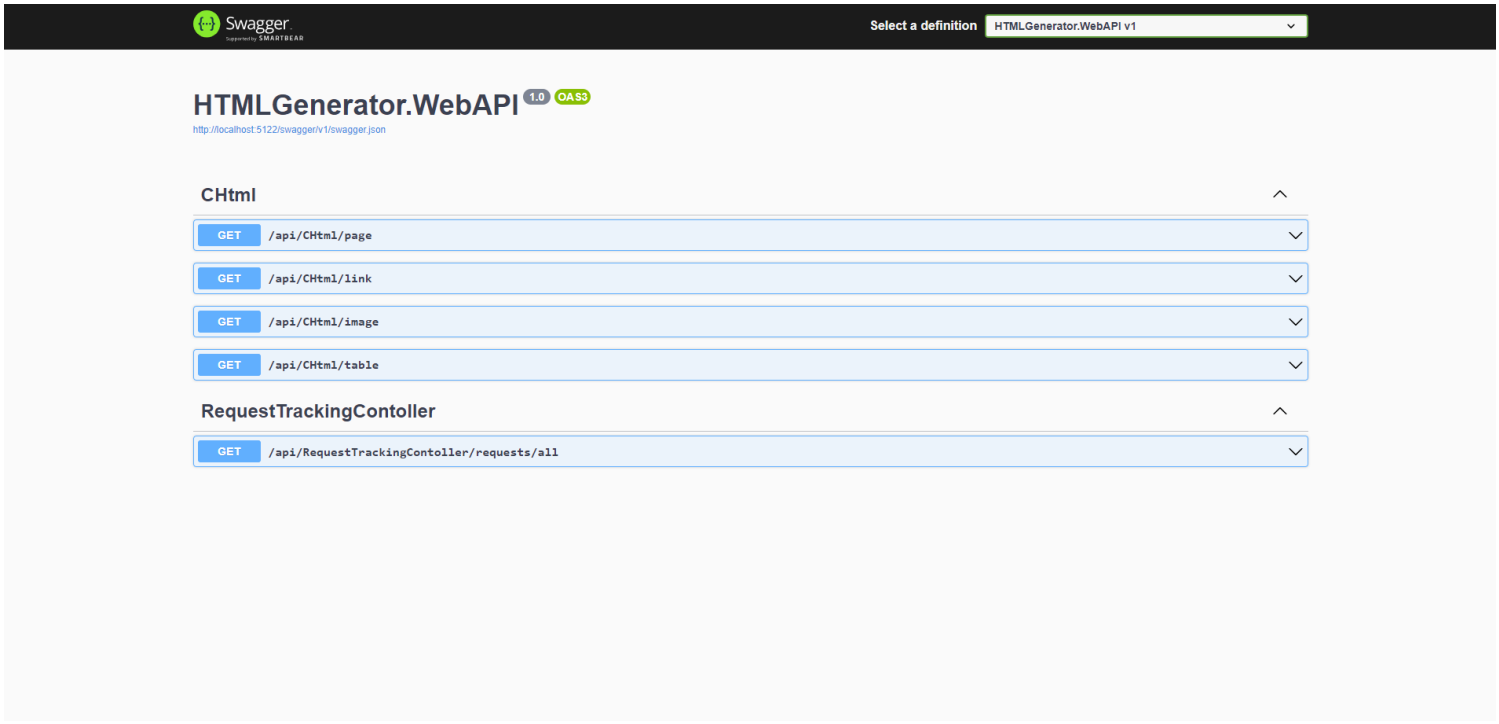
Цей HTTP-сервер став не просто ефективним інструментом обробки запитів, а й гнучкою, розширюваною платформою, що відповідає вимогам сучасного веб-середовища та забезпечує надійну роботу в різних умовах та сценаріях використання.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Andrew.T., & Philip.J. (2017). Pro C# 7 with .NET and .NET Core. United States: Apress.
2. Robert,C.M. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. United States: Pearson.
3. LINQ в C#. Огляд. (18.05.2021) . DEV. Retrieved from **LINQ в C#. Огляд - DEV Community**
4. Патерни проектування. Refactoring. Доступ: **Патерни/шаблони проектування (refactoring.guru)**
5. State Diagram. Plantuml. Доступ: **State Diagram syntax and features (plantuml.com)**
6. C# Interface. w3schools. Доступ: **C# Interface (w3schools.com)**
7. C# Polymorphism. w3schools. Доступ: **C# Polymorphism (w3schools.com)**
8. C# HttpListener. (05.07.2023). ZetCode. Доступ: **C# HttpListener - creating simple HTTP servers in C# (zetcode.com)**
9. Web server implementations in ASP.NET Core. (16.05.2023) . Microsoft. Доступ: **Web server implementations in ASP.NET Core | Microsoft Learn**
10. HttpServer Class. (28.10.2015) . Microsoft. Доступ: **HttpServer Class (System.Web.Http) | Microsoft Learn**

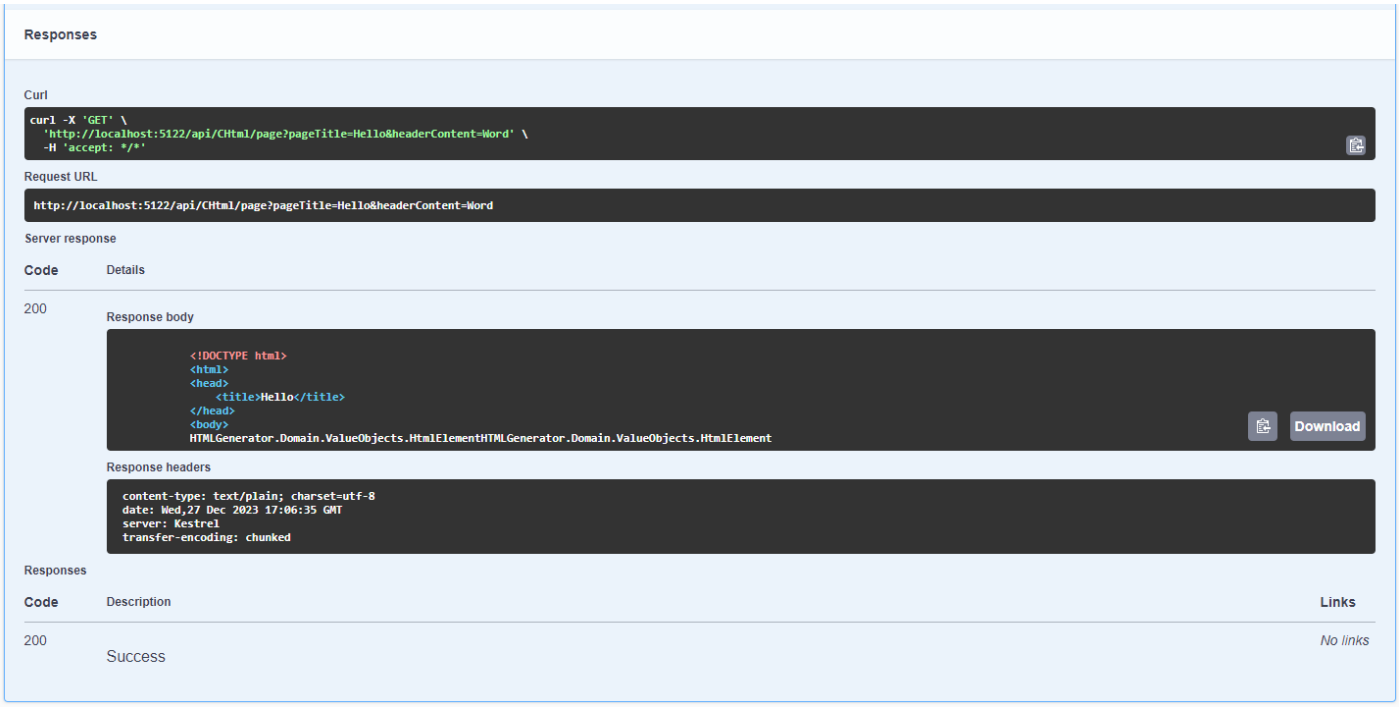
ДОДАТКИ

Web

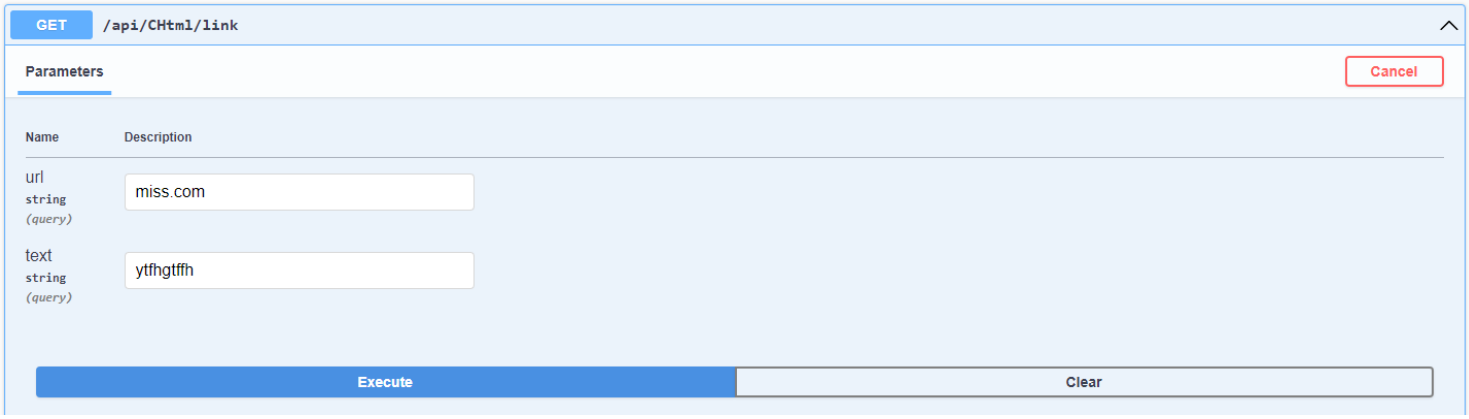

Swagger
Select a definition
HTMLGenerator.WebAPI v1
HTMLGenerator.WebAPI
1.0
OAS3
http://localhost:5122/swagger/v1/swagger.json
CHtml
GET /api/CHtml/page
GET /api/CHtml/link
GET /api/CHtml/image
GET /api/CHtml/table
RequestTrackingContoller
GET /api/RequestTrackingContoller/requests/all

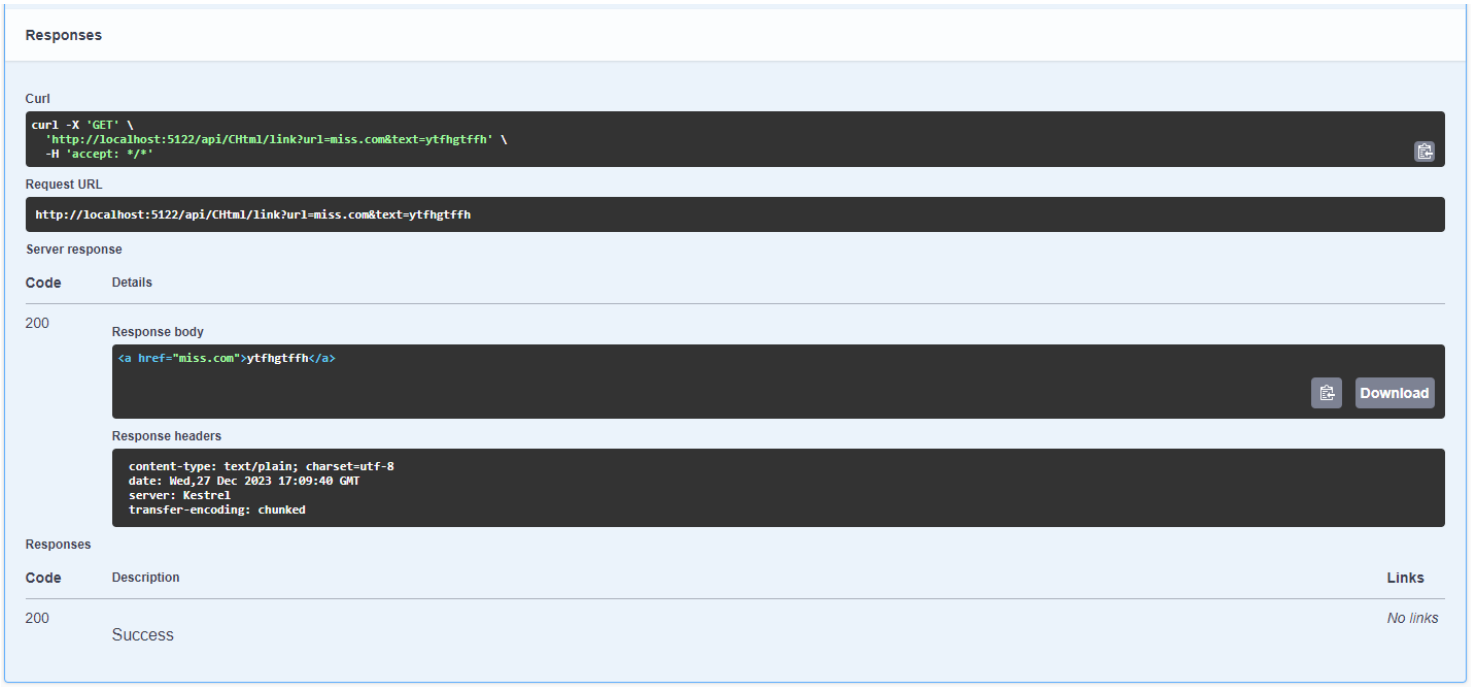
1) Page


GET
/api/CHtml/page
Parameters
Cancel
Name
Description
pageTitle
string
(query)
Hello
headerContent
string
(query)
Word
Execute
Clear

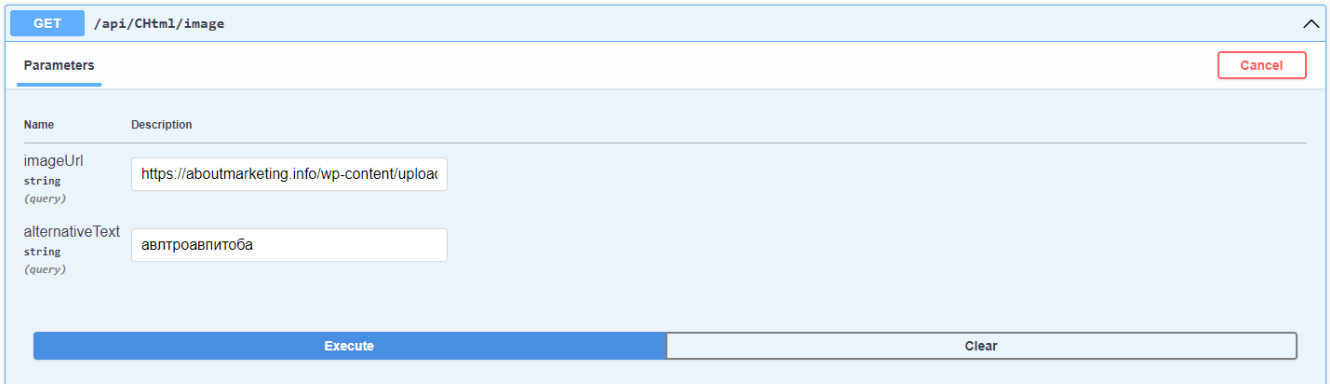

Responses
Curl
curl -X 'GET' \
'http://localhost:5122/api/CHtml/page?pageTitle=Hello&headerContent=Word' \
-H 'accept: */*'
Request URL
http://localhost:5122/api/CHtml/page?pageTitle=Hello&headerContent=Word
Server response
Code
Details
200
Response body
<!DOCTYPE html>
<html>
<head>
<title>Hello</title>
</head>
<body>
HTMLGenerator.Domain.ValueObjects.HtmlElementHTMLGenerator.Domain.ValueObjects.HtmlElement
Download
Response headers
content-type: text/plain; charset=utf-8
date: Wed,27 Dec 2023 17:06:35 GMT
server: Kestrel
transfer-encoding: chunked
Responses
Code
Description
Links
200
Success
No links

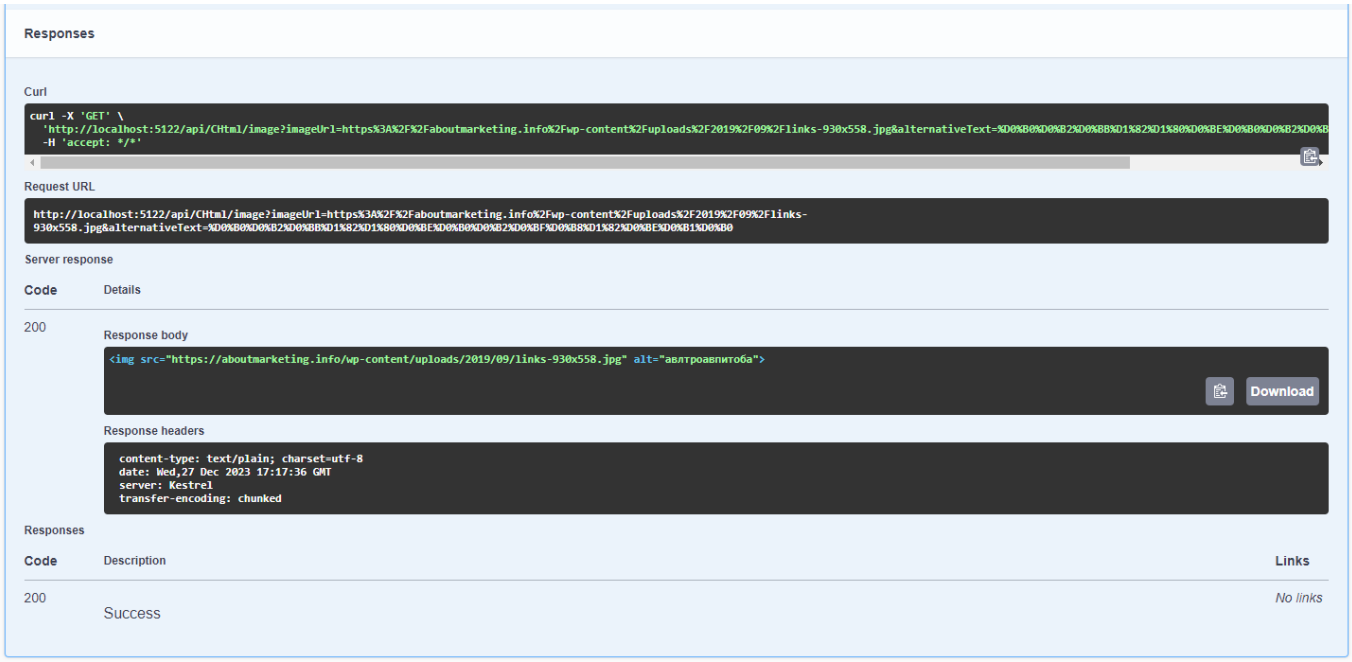
2) Link


GET
/api/CHtml/link
Parameters
Cancel
Name
Description
url
string
(query)
miss.com
text
string
(query)
ytfhgtffh
Execute
Clear

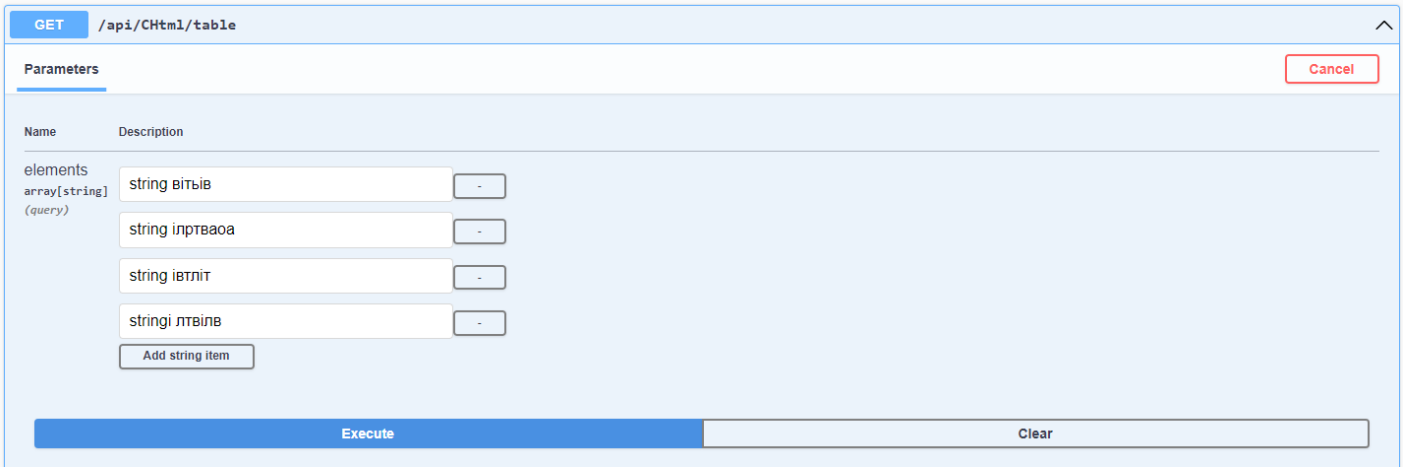

Responses
Curl
curl -X 'GET' \
'http://localhost:5122/api/CHtml/link?url=miss.com&text=ytfhgtffh' \
-H 'accept: */*'
Request URL
http://localhost:5122/api/CHtml/link?url=miss.com&text=ytfhgtffh
Server response
Code
Details
200
Response body
ytfhgtffh
Download
Response headers
content-type: text/plain; charset=utf-8
date: Wed,27 Dec 2023 17:09:40 GMT
server: Kestrel
transfer-encoding: chunked
Responses
Code
Description
Links
200
Success
No links

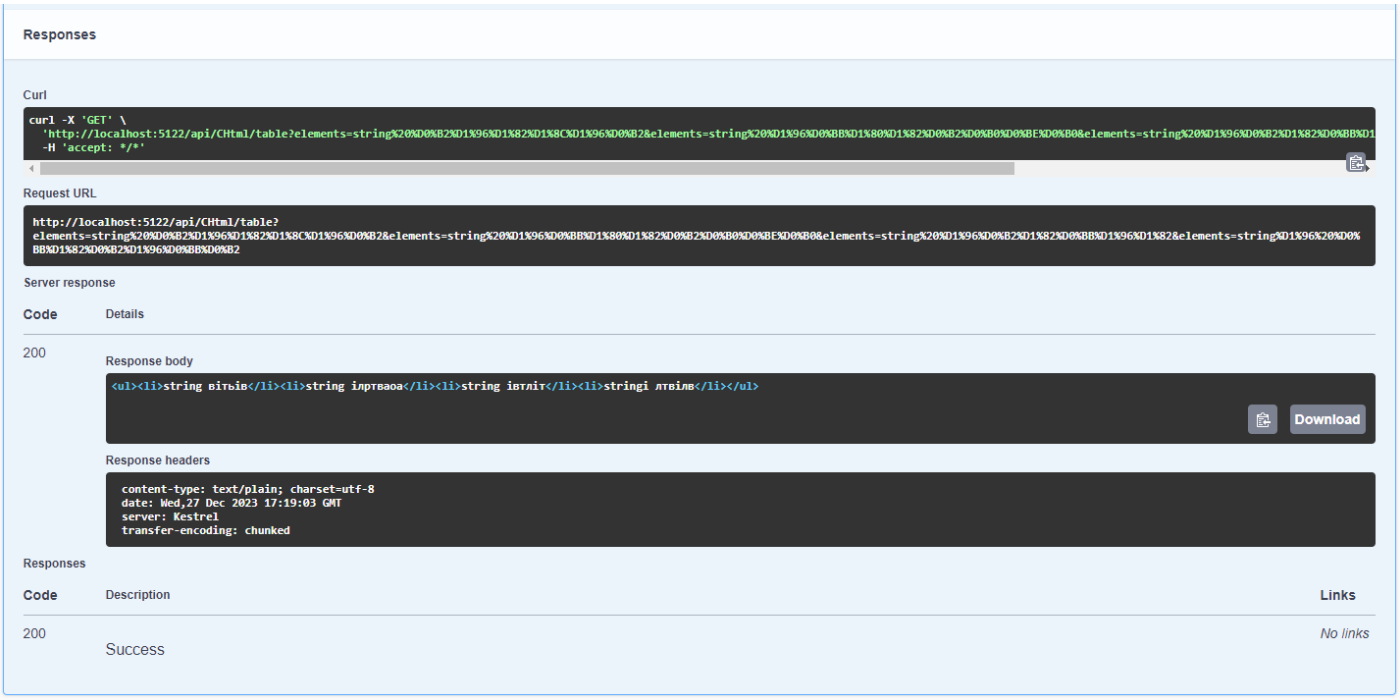
3) Image


GET
/api/CHtml/image
Parameters
Cancel
Name
Description
imageUrl
string
(query)
https://aboutmarketing.info/wp-content/uploa
alternativeText
string
(query)
авлтроавпитоба
Execute
Clear

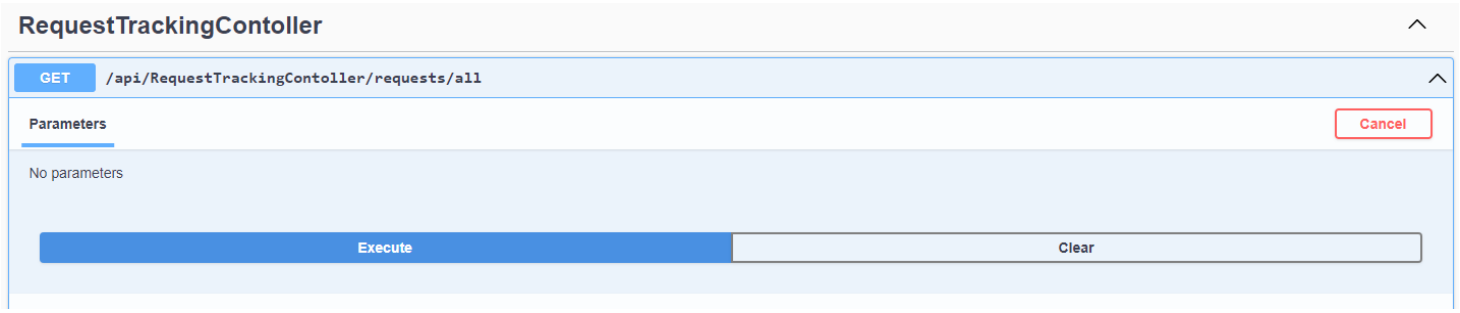

Responses
Curl
curl -X 'GET' \
'http://localhost:5122/api/CHtml/image?imageUrl=https%3A%2F%2Faboutmarketing.info%2Fwp-content%2Fuploads%2F2019%2F09%2Flinks-930x558.jpg&alternativeText=%D0%B0%D0%B2%D0%BB%D1%82%D1%80%D0%BE%D0%B0%D0%B2%D0%B
-H 'accept: */*'
Request URL
http://localhost:5122/api/CHtml/image?imageUrl=https%3A%2F%2Faboutmarketing.info%2Fwp-content%2Fuploads%2F2019%2F09%2Flinks-
930x558.jpg&alternativeText=%D0%B0%D0%B2%D0%BB%D1%82%D1%80%D0%BE%D0%B0%D0%B2%D0%BF%D0%B8%D1%82%D0%BE%D0%B1%D0%B0
Server response
Code
Details
200
Response body
<img src="https://aboutmarketing.info/wp-content/uploads/2019/09/links-930x558.jpg" alt="авлтроавпитоба">
Download
Response headers
content-type: text/plain; charset=utf-8
date: Wed,27 Dec 2023 17:17:36 GMT
server: Kestrel
transfer-encoding: chunked
Responses
Code
Description
Links
200
Success
No links

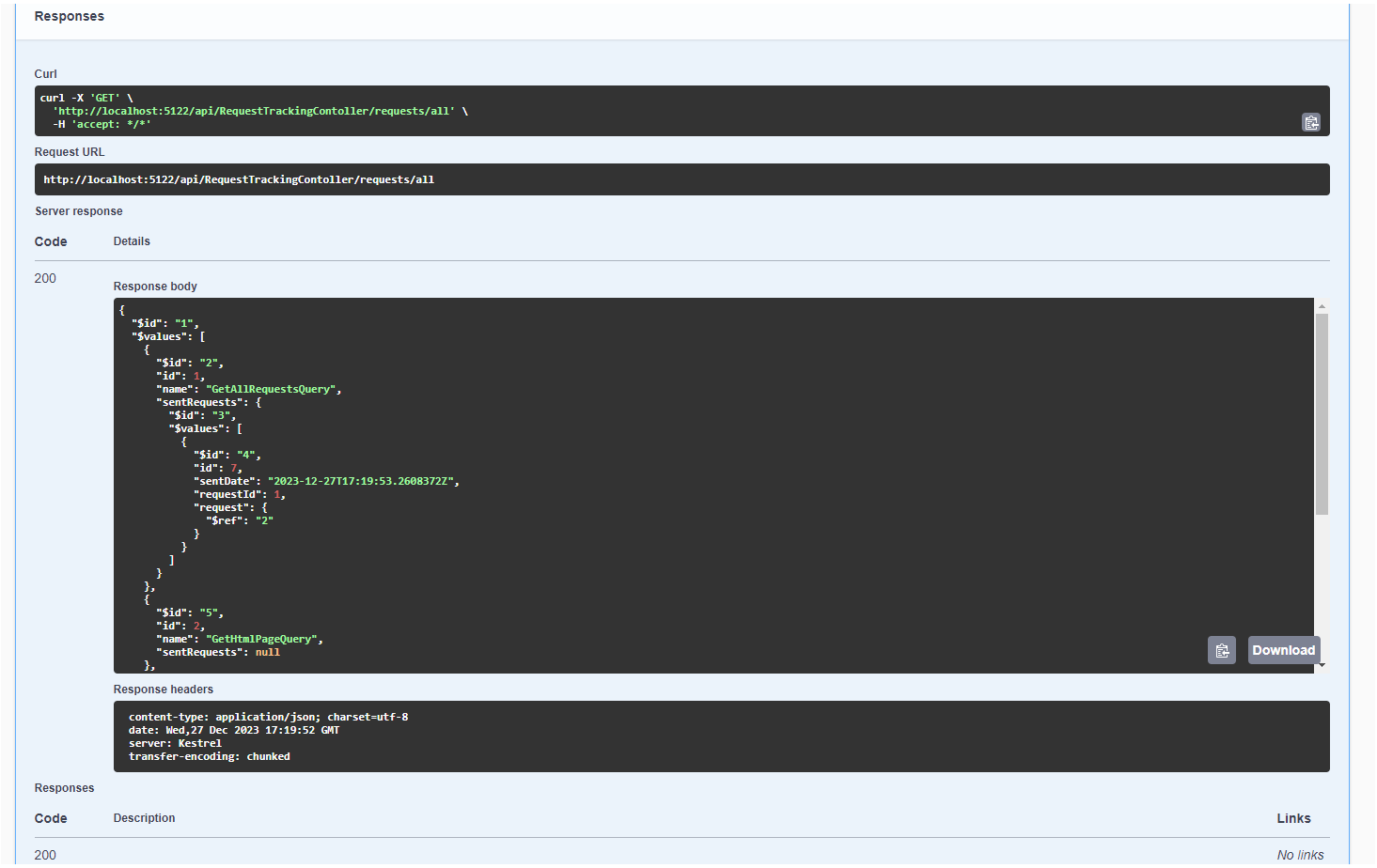
4) Table


GET
/api/CHtml/table
Parameters
Cancel
Name
Description
elements
array[string]
(query)
string вітьів
string ілртваоа
string івтліт
stringі лтвілв
Add string item
Execute
Clear


Responses
Curl
curl -X 'GET' \
'http://localhost:5122/api/CHtml/table?elements=string%20%D0%B2%D1%96%D1%82%D1%8C%D1%96%D0%B2&elements=string%20%D1%96%D0%BB%D1%80%D1%82%D0%B2%D0%B0%D0%BE%D0%B0&elements=string%20%D1%96%D0%B2%D1%82%D0%BB%D1
-H 'accept: */*'
Request URL
http://localhost:5122/api/CHtml/table?
elements=string%20%D0%B2%D1%96%D1%82%D1%8C%D1%96%D0%B2&elements=string%20%D1%96%D0%BB%D1%80%D1%82%D0%B2%D0%B0%D0%BE%D0%B0&elements=string%20%D1%96%D0%B2%D1%82%D0%BB%D1%96%D1%82&elements=string%D1%96%20%D0%
BB%D1%82%D0%B2%D1%96%D0%BB%D0%B2
Server response
Code
Details
200
Response body
string вітьів
string ілртваоа
string івтліт
stringі лтвілв
Download
Response headers
content-type: text/plain; charset=utf-8
date: Wed,27 Dec 2023 17:19:03 GMT
server: Kestrel
transfer-encoding: chunked
Responses
Code
Description
Links
200
Success
No links

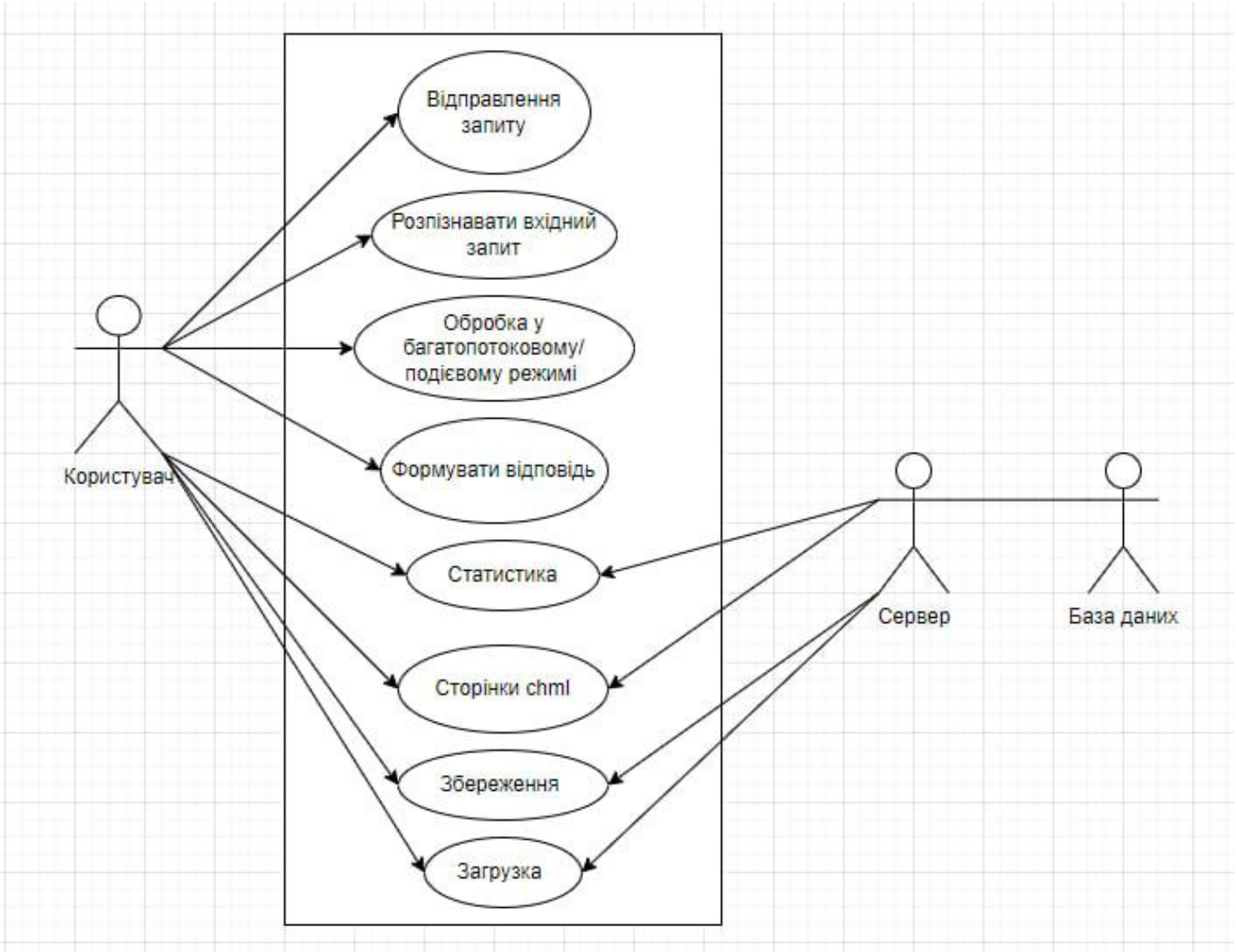
5) Statistics



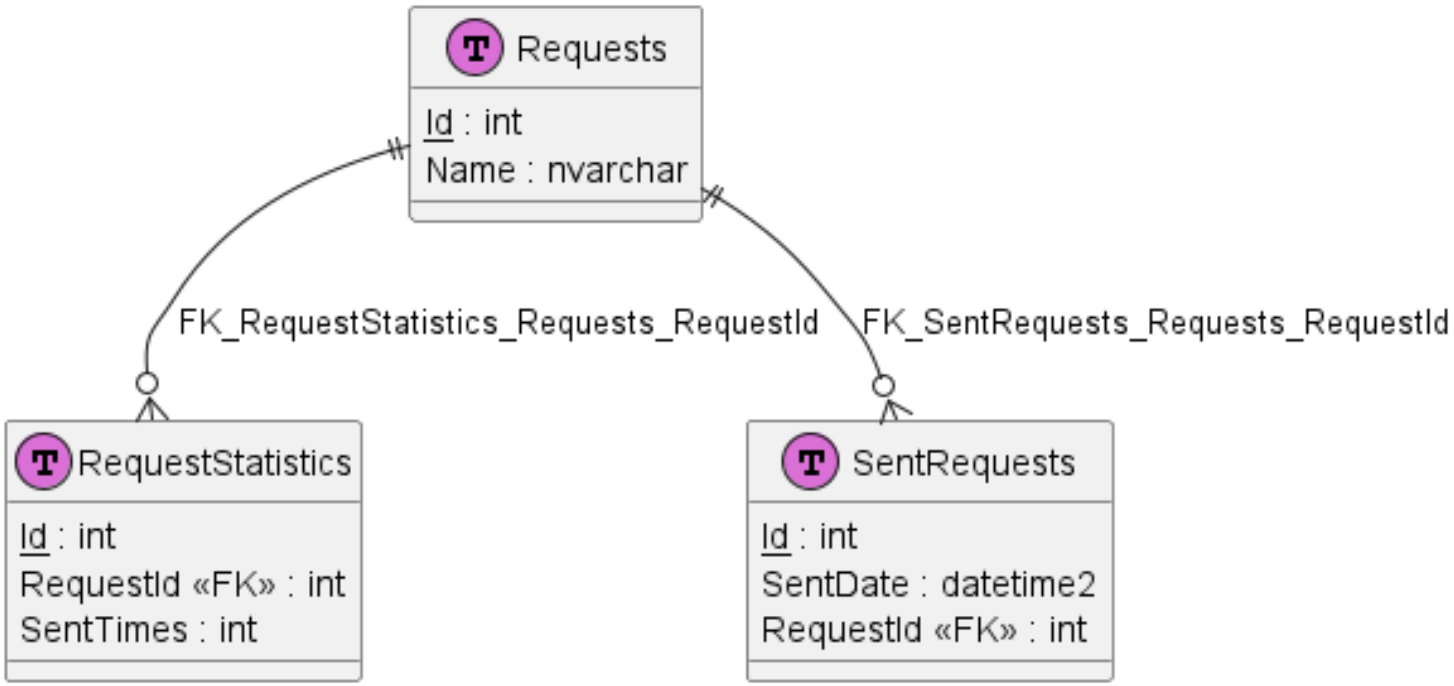


6) Diagrams

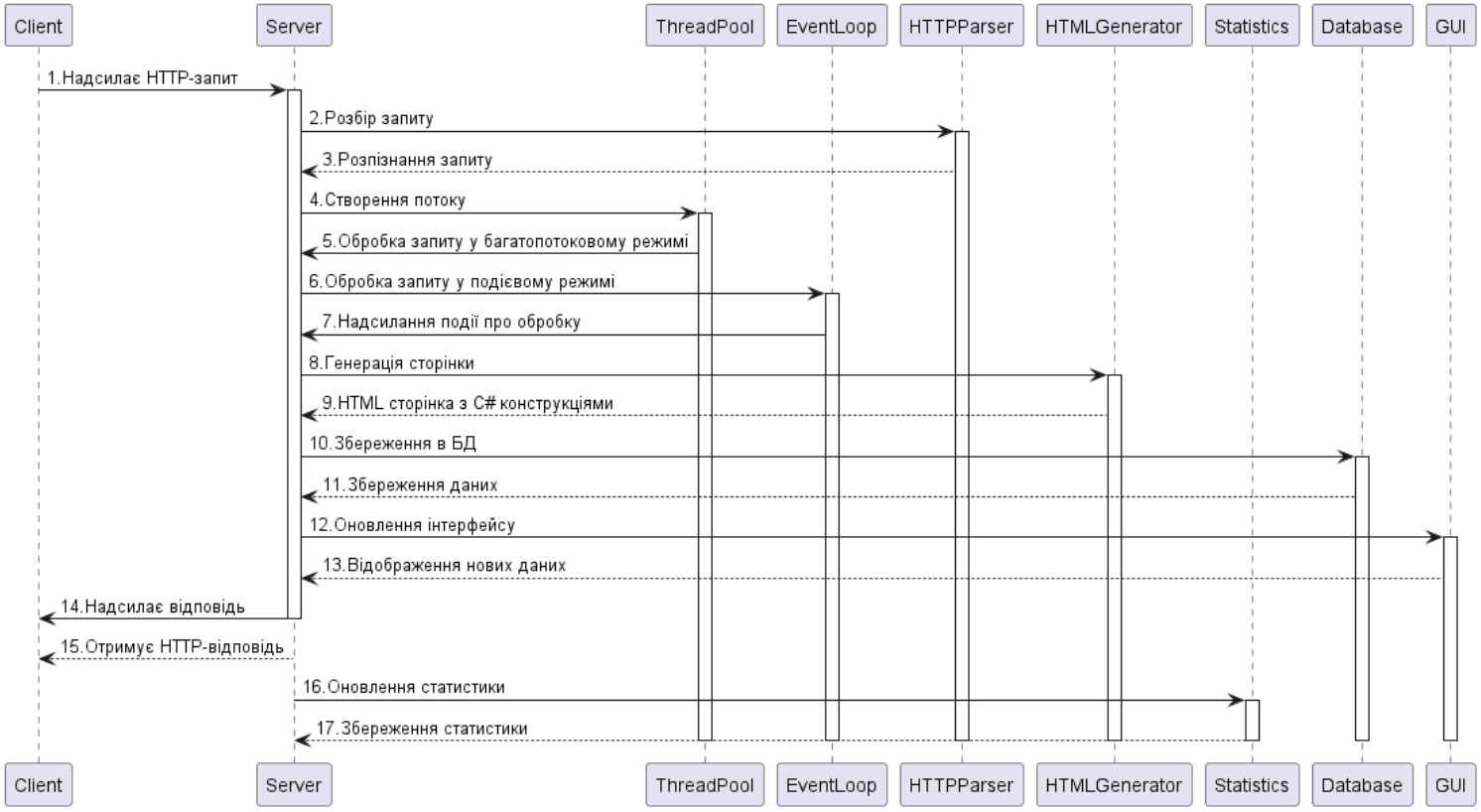
Use-Case



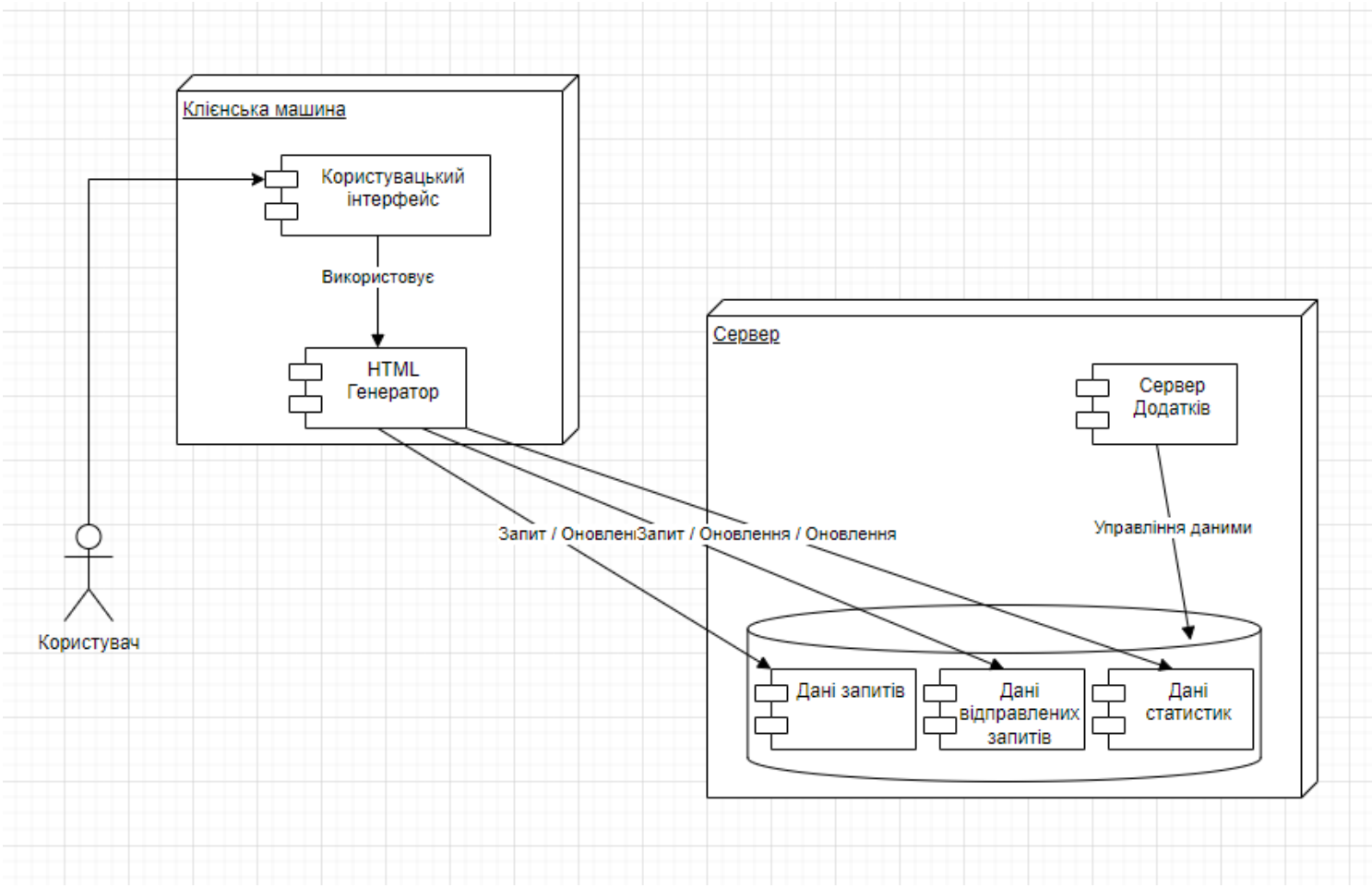
RequestDB



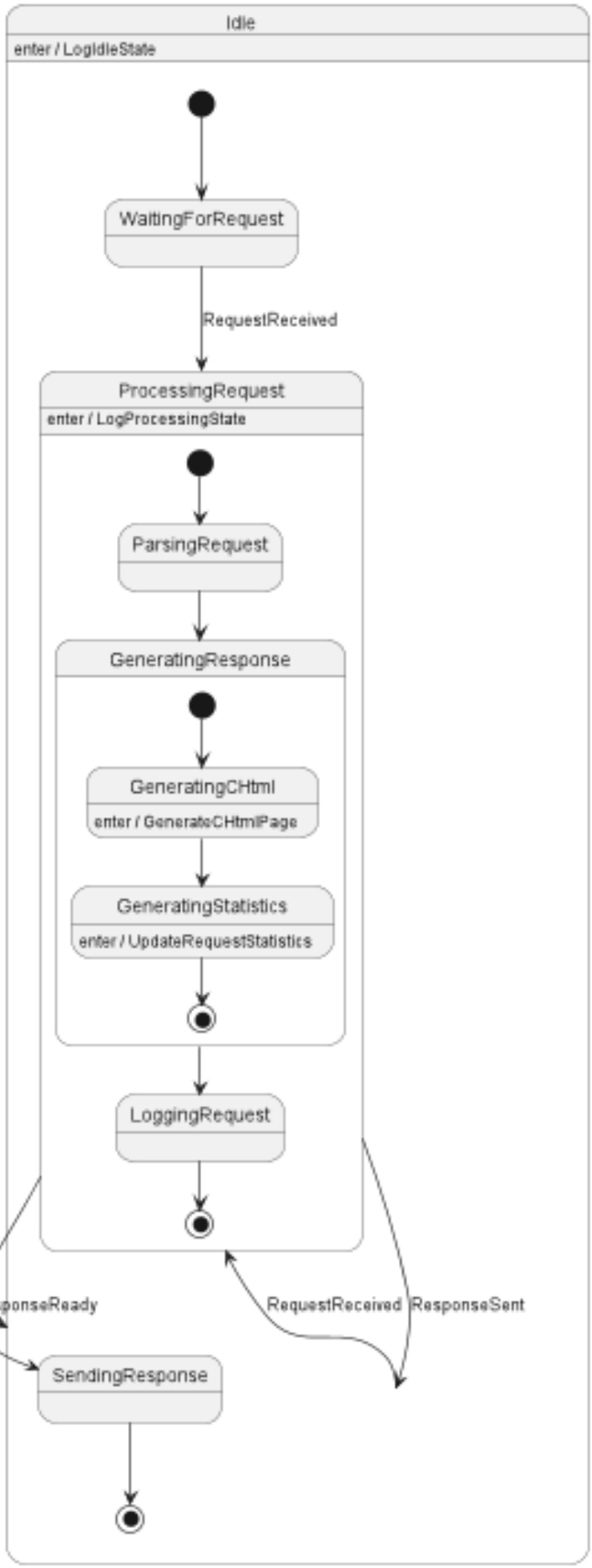
Sequence



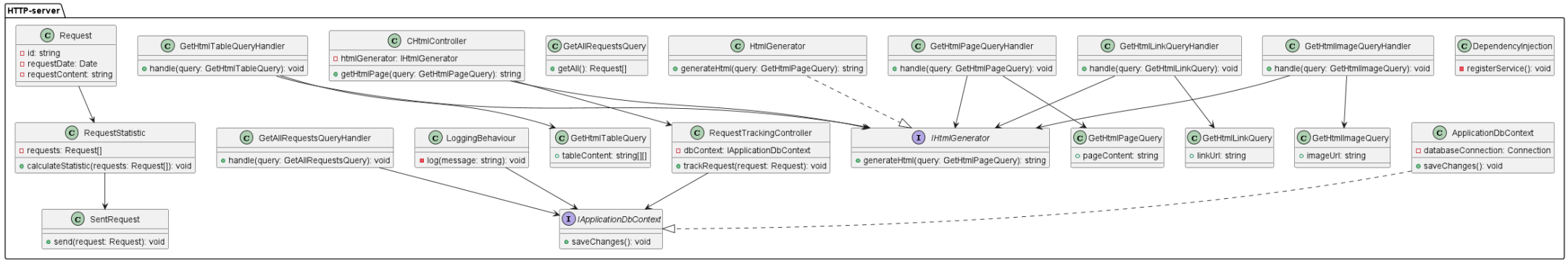
Deployment


Клієнська машина
Користувацький інтерфейс
Використовує
HTML Генератор
Сервер
Сервер Додатків
Управління даними
Запит / ОновленіЗапит / Оновлення / Оновлення
Користувач
Дані запитів
Дані відправлених запитів
Дані статистик

State


Idle
enter / LogIdleState
WaitingForRequest
RequestReceived
ProcessingRequest
enter / LogProcessingState
ParsingRequest
GeneratingResponse
GeneratingCHtml
enter / GenerateCHtmlPage
GeneratingStatistics
enter / UpdateRequestStatistics
LoggingRequest
ResponseReady
RequestReceived
ResponseSent
SendingResponse

Class


HTTP-server
Request
id: string
requestDate: Date
requestContent: string
GetHtmlTableQueryHandler
handle(query: GetHtmlTableQuery): void
CHtmlController
htmlGenerator: IHtmlGenerator
getHtmlPage(query: GetHtmlPageQuery): string
GetAllRequestsQuery
getAll(): Request[]
HtmlGenerator
generateHtml(query: GetHtmlPageQuery): string
GetHtmlPageQueryHandler
handle(query: GetHtmlPageQuery): void
GetHtmlLinkQueryHandler
handle(query: GetHtmlLinkQuery): void
GetHtmlImageQueryHandler
handle(query: GetHtmlImageQuery): void
DependencyInjection
registerService(): void
RequestStatistic
requests: Request[]
calculateStatistic(requests: Request[]): void
GetAllRequestsQueryHandler
handle(query: GetAllRequestsQuery): void
LoggingBehaviour
log(message: string): void
GetHtmlTableQuery
tableContent: string[][]
RequestTrackingController
dbContext: IApplicationDbContext
trackRequest(request: Request): void
IHtmlGenerator
generateHtml(query: GetHtmlPageQuery): string
GetHtmlPageQuery
pageContent: string
GetHtmlLinkQuery
linkUrl: string
GetHtmlImageQuery
imageUrl: string
ApplicationDbContext
databaseConnection: Connection
saveChanges(): void
SentRequest
send(request: Request): void
IApplicationDbContext
saveChanges(): void

7) DB

/DependencyInjection

```
using HTMLGenerator.Application.Common.Interfaces;
using HTMLGenerator.Infrastructure.Persistence;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;

namespace HTMLGenerator.Infrastructure
{
    public static class DependencyInjection
    {
        public static IServiceCollection AddInfrastructure(this IServiceCollection services,
IConfiguration configuration)
        {
            var databaseName = "requestdb";
            var connectionString =
$"Server=(localdb)\\mssqllocaldb;Database={databaseName};Trusted_Connection=True;MultipleActiv
eResultSets=true";

            services.AddDbContext<ApplicationDbContext>(options =>
                options.UseSqlServer(connectionString))
                .AddLogging(configure => configure.SetMinimumLevel(LogLevel.None));
            services.AddScoped<IApplicationDbContext>(provider =>
provider.GetRequiredService<ApplicationDbContext>());

            return services;
        }
    }
}
```

/ApplicationDbContext

```
using HTMLGenerator.Application.Common.Interfaces;
using HTMLGenerator.Domain.Entities;
using Microsoft.EntityFrameworkCore;

namespace HTMLGenerator.Infrastructure.Persistence
{
    public class ApplicationDbContext : DbContext, IApplicationDbContext
    {
        public DbSet<Request> Requests => Set<Request>();

        public DbSet<SentRequest> SentRequests => Set<SentRequest>();

        public DbSet<RequestStatistic> RequestStatistics => Set<RequestStatistic>();

        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) :
base(options)
        {
            Database.EnsureCreated();
        }
    }
}
```

/ IApplicationDbContext

```
using HTMLGenerator.Domain.Entities;
using Microsoft.EntityFrameworkCore;

namespace HTMLGenerator.Application.Common.Interfaces
{
    public interface IApplicationDbContext
    {
        DbSet<Request> Requests { get; }
        DbSet<SentRequest> SentRequests { get; }
        DbSet<RequestStatistic> RequestStatistics { get; }
        Task<int> SaveChangesAsync(CancellationToken cancellationToken);
    }
}
```