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

Лабораторна робота №2

Технології розроблення програмного забезпечення

«*Діаграма варіантів використання. Сценарії варіантів використання. Діаграми UML. Діаграми класів. Концептуальна модель системи.*»

Варіант 2

Виконала
студентка групи ІА–24:
Кійко А. О.

Перевірив
Мягкий М. Ю.

Київ 2024

ЗМІСТ

Тема: HTTP-сервер (state, builder, factory method, mediator, composite, p2p). 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.

Короткі теоритичні відомості

Прецеденти (Use Case Diagram)

Діаграма прецедентів використовується для візуалізації функціональних вимог до системи. Вона показує, як користувачі (актор або актори) взаємодіють із системою через певні сценарії використання (прецеденти). Основними елементами діаграми є актори, прецеденти, а також зв'язки між ними. Прецеденти допомагають виявити основні функції системи та забезпечують їх розуміння на високому рівні.

Діаграма класів (Class Diagram)

Діаграма класів використовується для моделювання статичної структури системи. Вона показує класи, атрибути класів, методи (операції), а також зв'язки між класами, такі як асоціації, агрегації, композиції та успадкування. Класи представляють основні компоненти системи, їхні характеристики (атрибути) та поведінку (методи). Зв'язки показують, як ці класи взаємодіють між собою.

База даних та її структура

База даних - це організований набір даних, які зберігаються в структурованому вигляді, зазвичай у вигляді таблиць. Таблиці в базі даних складаються з рядків (записів) і стовпців (полів), що містять атрибути даних. Структура бази даних визначає, як дані взаємопов'язані між собою.

Основними елементами є таблиці, ключі (первинні та зовнішні) і зв'язки між таблицями (один-до-одного, один-до-багатьох, багато-до-багатьох).

Шаблон Репозиторію (Repository Pattern)

Шаблон Репозиторію використовується для абстрагування доступу до даних. Він дозволяє взаємодіяти з базою даних через клас-репозиторій, що інкапсулює всі операції збереження, отримання, оновлення та видалення даних. Це забезпечує слабку залежність між бізнес-логікою та логікою доступу до даних, роблячи систему більш гнучкою для змін або модернізацій.

Вибір прецедентів та аналіз

Прецеденти дозволяють ідентифікувати сценарії використання, що найбільш підходять для вашої системи. Важливо вибрати і проаналізувати кілька подібних систем, щоб побачити, як їх функціональність відповідає вимогам вашого проекту. Це допомагає розробити найбільш ефективні та зручні рішення для користувачів.

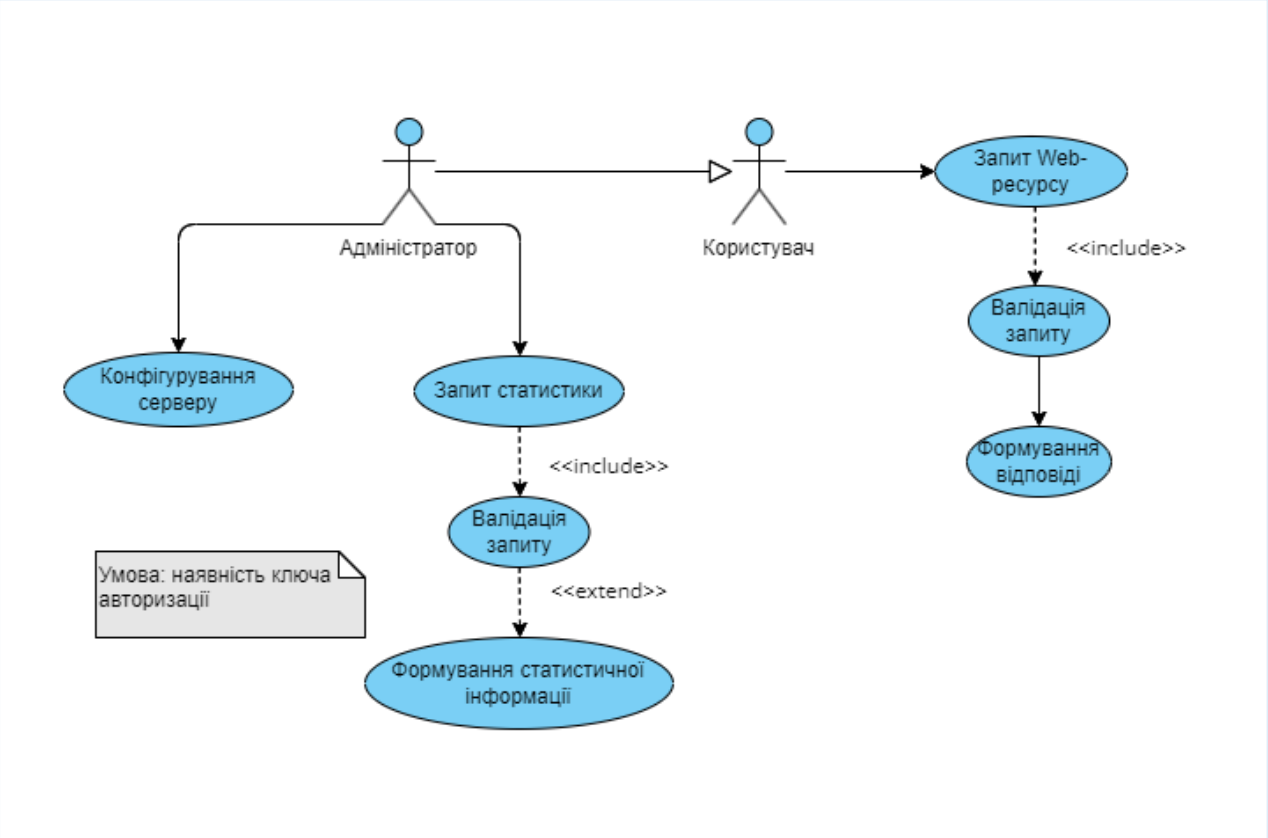
Хід роботи

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.
2. Проаналізувати тему та намалювати схему прецеденту, що відповідає обраній темі лабораторії.
3. Намалювати діаграму класів для реалізованої частини системи.
4. Вибрати 3 прецеденти і написати на їх основі прецеденти.
5. Розробити основні класи і структуру системи баз даних.
6. Класи даних повинні реалізувати шаблон Репозиторію для взаємодії з базою даних.
7. Підготувати звіт про хід виконання лабораторних робіт. Звіт, що подається повинен містити: діаграму прецедентів, діаграму класів

системи, вихідні коди класів системи, а також зображення структури бази даних.

Прецеденти:



Сценарії використання

Сценарій 1: Користувач - запит Web-ресурсу серверу.

Передумови	Наявність підключення до мережі Інтернет
Постумови	У разі успішної перевірки запиту сервер повертає користувачеві зміст сторінки, що запитувалась
Сторони, що взаємодіють	Користувач, HTTP-сервер
Короткий опис	Даний варіант використання описує запит користувача до певного ресурсу HTTP-сервера
Основний потік подій	1. Користувач надсилає HTTP-запит;

	2. Сервер перевіряє валідність запиту. Якщо запит не валідний, то виняткова ситуація №1; 3. Сервер перевіряє наявність ресурсу (сторінки), що запитується. Якщо сторінка не знайдена, то виняткова ситуація №2; 4. Сервер повертає користувачеві ресурс, що запитується; 5. Сервер записує інформацію до репозиторію про запит та результат обробки; 6. Сервер завершує сесію.
Винятки	1. Неправильний формат HTTP-запиту. Система формує повідомлення про помилку, відбувається перехід до п. 5 основного потоку подій. 2. Не знайдено ресурс (сторінку), що запитується. Система виводить повідомлення про відсутність ресурсу, після чого відбувається перехід до п. 5.

Сценарій 2: Адміністратор - запит статистики.

Передумови	Наявність підключення до мережі Інтернет
Постумови	У разі успішної перевірки запиту сервер повертає дані статистики запитів користувачів за обраний період часу
Сторони, що взаємодіють	Адміністратор серверу, HTTP-сервер
Короткий опис	Даний варіант використання описує запит адміністратора для отримання даних щодо статистики виконання запитів користувачів за обраний період часу
Основний потік подій	1. Actor надсилає запит для отримання статистики за обраний період часу. Якщо період часу не зазначений, за замовчуванням дані видаються за останні 24 год.;

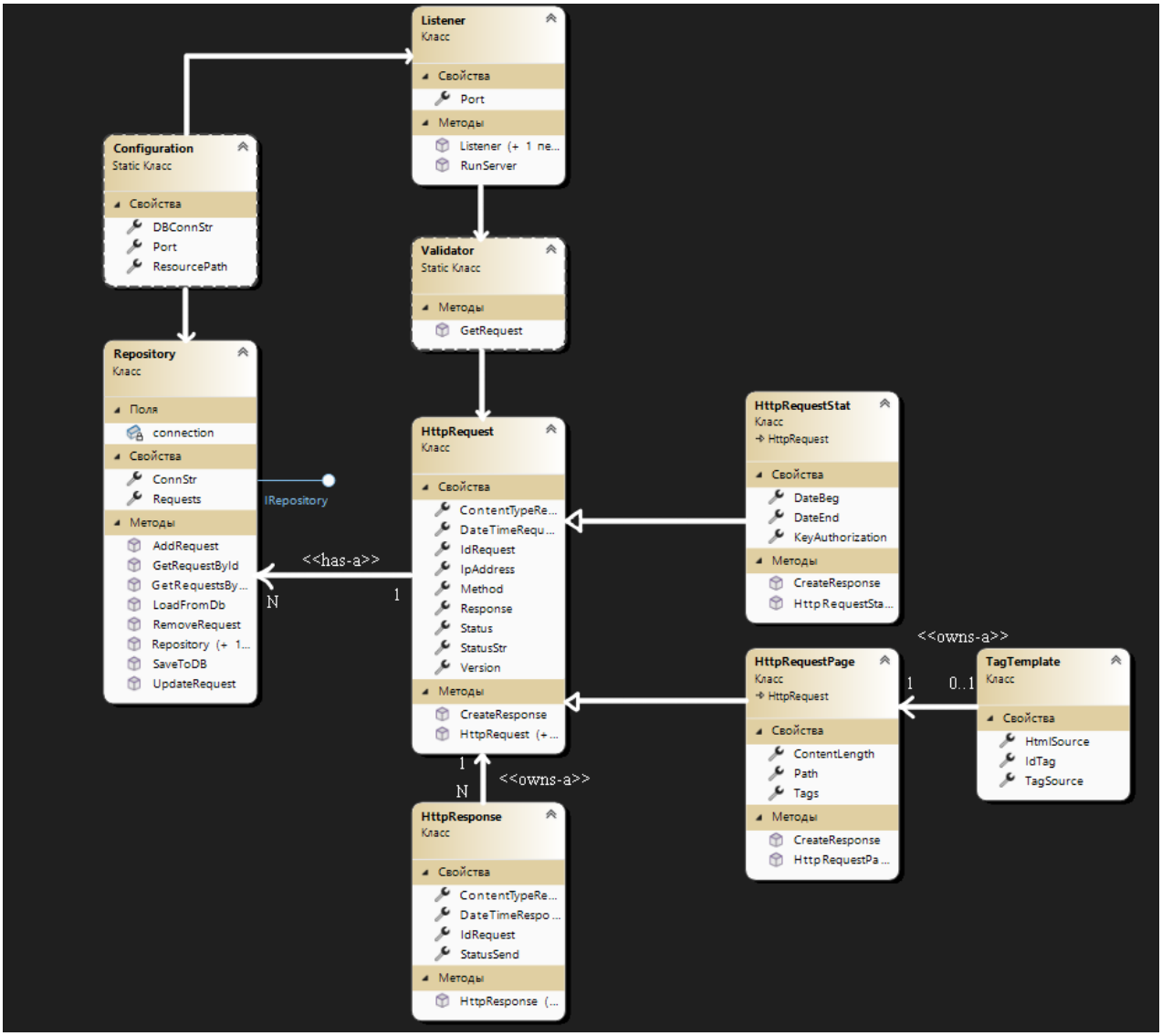
	2. Сервер перевіряє в заголовку запиту наявність ключа доступу адміністратора. Якщо ключ відсутній або не валідний, то виняткова ситуація №1; 3. Сервер формує запит до бази даних. Якщо доступ до бази даних відсутній, то виняткова ситуація №2; 4. Сервер повертає адміністратору дані статистики за обраний період часу; 5. Сервер завершує сесію.
Винятки	1. Ключ адміністратора системи в заголовку запиту відсутній або не валідний. Система формує повідомлення про помилку, відбувається перехід до п.5 основного потоку подій. 2. Відсутній доступ до бази даних. Система виводить повідомлення про відсутність доступу, після чого відбувається перехід до п. 5.

Сценарій 3: Адміністратор - адміністрування ресурсів (конфігурування серверу, додавання/видалення сторінок).

Передумови	Наявність підключення до мережі Інтернет
Постумови	Робота HTTP-серверу в оновленій конфігурації
Сторони, що взаємодіють	Адміністратор, сервер
Короткий опис	Даний варіант використання описує процес зміни конфігурації серверу, додавання/видалення ресурсів (сторінок) сервера.
Основний потік подій	1. Адміністратор, використовуючи механізми безпосереднього доступу до ресурсів серверу, змінює конфігурацію серверу (порт, шлях до репозиторію, перелік сторінок тощо);

	2. Адміністратор виконує перезавантаження HTTP-серверу; 3. При перезавантаженні сервер перевіряє оновлену конфігурацію. Якщо конфігурація не валідна, то виняткова ситуація №1; 4. Сервер працює в штатному режимі.
Винятки	1. Конфігурація не валідна. Аварійне завершення роботи серверу.

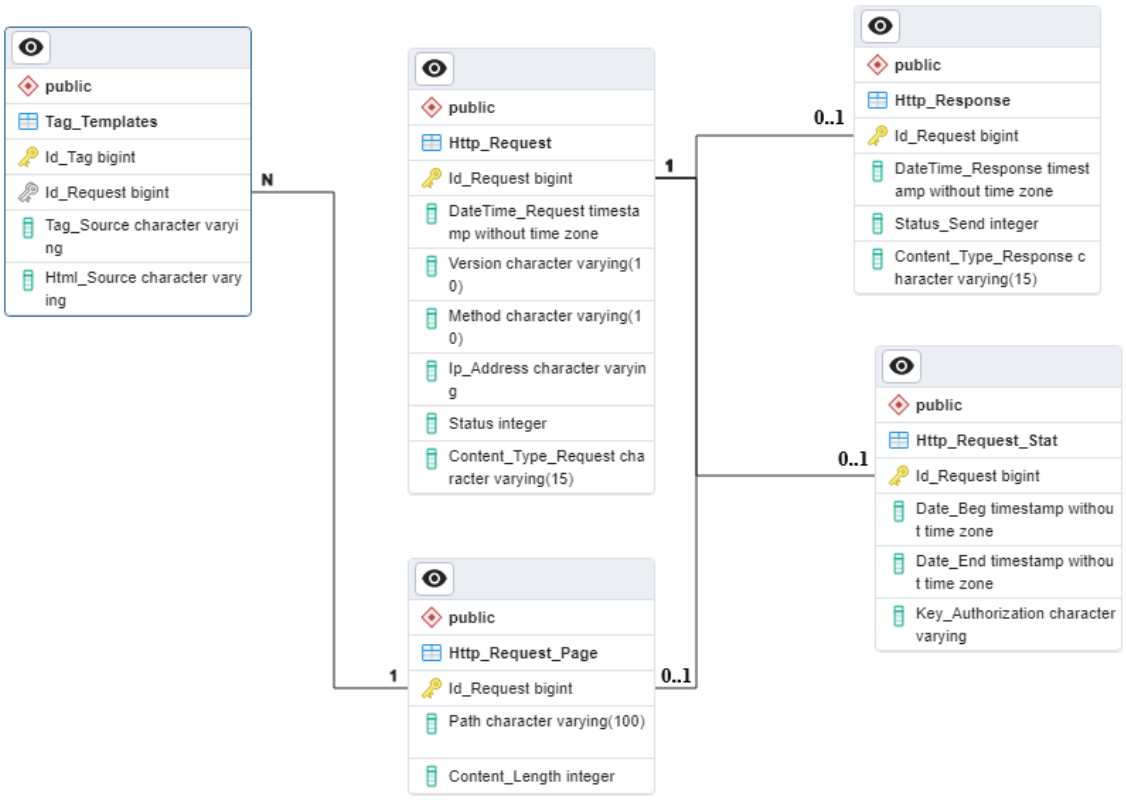
Діаграма класів:



Опис класів:

- **Listener** - управляючий циклом обробки клас, що забезпечує прослуховування відповідного сокету та організацію процесу обробки запитів;
- **Validator** - статичний клас, що призначений для парсингу вхідного потоку інформації, валідації та формування об’єкту HttpRequest або одного з його нащадків (HttpRequestPage, HttpRequestStat) в залежності від типу запиту;
- **Configuration** - статичний клас, що містить основні дані конфігурації серверу: порт, строка підключення до БД та шлях до місця збереження HTML-сторінок. Дана інформація використовується в класах Listener та Repository;
- **HttpRequest** - базовий клас, що описує об’єкт запиту до серверу. Він містить інформацію про основі дані запиту: метод та час запиту, версію, IP-адресу користувача тощо;
- **HttpRequestPage** - клас-нащадок (відношення узагальнення) HttpRequest, що містить додаткові атрибути запиту HTML-сторінки: адреса сторінки та довжина контенту в тілі запиту в байтах;
- **TagTemplate** - містить інформацію про конвертацію C# конструкцій у фрагменти HTML. Пов’язаний відношенням композиції (<<owns-a>>) з класом HttpRequestPage;
- **HttpRequestStat** - клас-нащадок (відношення узагальнення) HttpRequest, що містить додаткові атрибути запиту статистичної інформації: дати початку та кінця періоду запиту, ключ авторизації користувача, що є адміністратором системи;
- **HttpResponse** - клас, що описує об’єкт відповіді від сервера. Містить додаткові атрибути: час, статус відправки відповіді, формат відповіді. Пов’язаний відношенням асоціації з класом HttpRequest;
- **Repository** - клас, що реалізує інтерфейс IRepository. Інкапсулює всі операції взаємодії з сервером БД: отримання, додавання, оновлення, збереження та видалення даних. Включає в себе перелік об’єктів HttpRequest (відношення агрегації - <<has-a>>).

Структура бази даних:



Опис таблиць та зв'язків між ними:

- **Http_Request** - загальний перелік запитів до системи;
- **Http_Request_Page** - розширення таблиці Http_Request, описує дані запиту користувачем HTML-сторінки. Пов'язана відношенням "один-до-одного" з таблицею Http_Request;
- **Http_Request_Stat** - розширення таблиці Http_Request, описує дані запиту статистики за обраний період часу. Пов'язана відношенням "один-до-одного" з таблицею Http_Request;
- **Http_Response** - загальний перелік відповідей від системи. Пов'язана відношенням "один-до-одного" з таблицею Http_Request;
- **Tag_Templates** - описує конвертацію C# конструкцій у фрагменти HTML. Пов'язана відношенням "один-до-багатьох" з таблицею Http_Request_Page (заповнюється, ящко сторінка має у своєму складі C# вставки).

Висновок: у ході виконання даної лабораторної роботи було опрацьовано основи моделювання програмних систем з використанням діаграм прецедентів та діаграм класів. В цій роботі ми створили діаграми прецедентів, які описують взаємодію користувача із системою, побудували діаграму класів, що відображають зв'язки між сутностями бази даних, розробили структуру бази даних та побудували ER-діаграму.