

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра програмних систем і технологій

Дисципліна
« Архітектура та проектування ПЗ »

Лабораторна робота № 1
«Онлайн клієнт для погодних даних»

Виконав:	Гоша Д.О	Перевірів:	Берестов Д.С
Група	ІПЗ-23.1	Дата перевірки	
Форма навчання	Денна	Бали	
Спеціальність	121		

Мета практикуму – формування практичних навичок із застосування сучасних прийомів і засобів проектування ПЗ та оволодіння методами підтримки якості ПЗ.

Основні теоретичні визначення:

Архітектура ПЗ – фундаментальна конструкція всієї системи ПЗ, що визначає:

- які елементи включені в систему,
- яку функцію має кожен елемент,
- як кожен елемент пов'язаний один з одним

Архітектура ПЗ – загальна структура всієї системи – як все працює разом.

Архітектор ПЗ повинен враховувати наступні фактори:

- призначення системи,
- аудиторія або користувачі системи,
- якості з найбільшим значенням для користувачів;
- де система буде працювати.

Переваги застосування архітектури ПЗ:

- підвищення продуктивності (*productivity*) для команди – чітко визначена структура допомагає координувати роботу, реалізовувати індивідуальні функції та керувати обговоренням потенційних питань;
- поліпшена еволюція (*evolution*) ПЗ – принципи дизайну застосовуються для полегшення змін та/або легше знайти дефекти ПЗ;
- підвищення якості (*quality*) ПЗ за рахунок ретельного врахування потреб та перспектив усіх зацікавлених сторін (*stakeholders*).

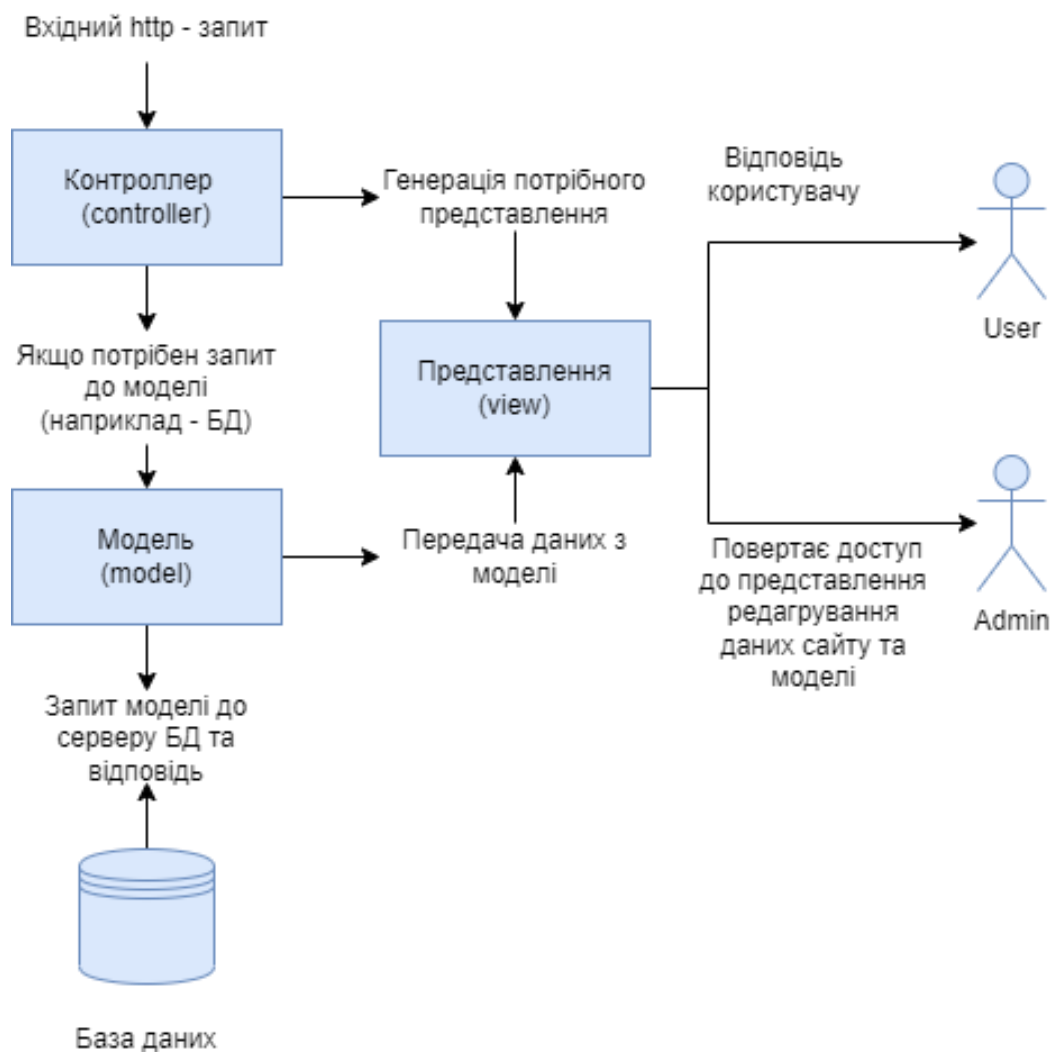
Архітектура ПЗ допомагає зробити систему простішою в обслуговуванні (*maintain*), повторному використанні (*reuse*) та адаптації (*adapt*).

- Архітектурна конфігурація – це специфікація з'єднань між з'єднувачами та компонентами.
- Нефункціональні властивості системи (НВС) – це обмеження на те, як ПЗ реалізує і доставляє свою функціональність. Наприклад: ефективність, складність, розширюваність, надійність. Будь-який високотехнологічний продукт, такий як телевізор чи мобільний телефон, продають на основі своїх функціональних можливостей.
- Ефективність – це якість, яка відображає здатність ПЗ до задоволення вимог продуктивності при одночасній мінімізації використання його ресурсів.
- Складність вказує до якої міри ПЗ або один з його компонентів, містить проектні рішення чи реалізацію, які важко зрозуміти і перевірити.

- Масштабованість ПЗ – це можливість системи бути зміненою з урахуванням нових вимог.
- Надійність – це набір властивостей ПЗ, що дозволяє розраховувати, що ПЗ буде функціонувати так, як було заплановано.
- Адаптованість - можливість використання системи без її зміни в тих галузях або середовищах, на які вона не була орієнтована безпосередньо.

Постановка задачі:

Створення ПЗ, який буде приймати погодні дані , передавати їх до нашого серверу, який буде містити сайт (графічний інтерфейс) та базу даних, та виводитиме їх у спеціальних блоках сайту.



Діаграма архітектури ПЗ

1. Опис застосування, що розробляється з точки зору користувача.

1.1. Інтерфейс , що друкує інформацію про поточний стан погоди. А саме:

- 1.1.1. Температура (Погодинно).
- 1.1.2. Швидкість вітру.
- 1.1.3. Відносна вологість.
- 1.1.4. Видимість.
- 1.1.5. Стан опадів (Йде сніг, пасмурно, тощо).

2. Опис основних функціональних вимог.

- 2.1. Отримання даних з серверу метеорологічного центру.
- 2.2. Сортуння та нормалізація даних.
- 2.3. Створення візуального зображення погодних умов.

3. Опис основних нефункціональних вимог.

Не функціональні вимоги можна поділити на дві категорії: покращення (безпека, надійність, швидкодія, зручність у використанні ...) та вдосконалення (масштабування, відновлюваність ...) властивостей системи.

1.1. Вимоги до Інтерфейсу (Interface Requirements)

1.1.1. Апаратні Інтерфейси (Hardware Interfaces)

Апаратні інтерфейси необхідні для підтримки системи, включаючи логічну структуру, фізичні адреси і очікувану поведінку.

1.1.2. Інтерфейси ПЗ (Software Interfaces)

Назви Інтерфейсів програмного забезпечення з якими аплікація повинна взаємодіяти.

1.1.3. Зв'язки Інтерфейсів (Communications Interfaces)

Зв'язки інтерфейсу з іншими системами або приладами.

1.2. Апаратні та Програмні Вимоги (Hardware/Software Requirements)

Опис апаратної та програмної платформ, необхідних для підтримки системи.

1.3. Operational Requirements

1.3.1. Безпека та Конфіденційність (Security and Privacy)

1.3.2. Надійність (Reliability)

1.3.3. Відновлювальність (Recoverability)

1.3.4. Продуктивність (Performance)

1.3.5. Потенціал (Capacity)

1.3.6. Збереження даних (Data Retention)

1.3.7. Керування помилками (Error Handling)

1.3.8. Правила Перевірки (Validation Rules)

4. Опис компонентів, які будуть використовуватися: сервіси, які вони забезпечують, основні архітектурні рішення, які вони втілюють, і припущення, які вони роблять. Обґрунтувати їх вибір.

4.1. Будемо використовувати багаторівневу архітектуру (На діаграмі зображено саме логічні шари архітектури).

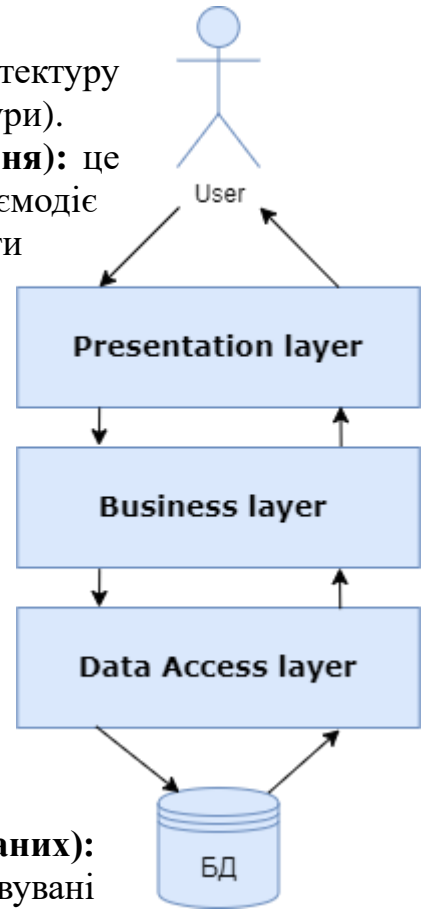
4.1.1. **Presentation layer (рівень представлення):** це той рівень, з яким безпосередньо взаємодіє користувач. Цей рівень включає компоненти інтерфейсу користувача, механізм отримання, введення від користувача. Стосовно asp.net mvc на даному рівні розташовані уявлення і всі ті компоненти, які складають інтерфейс користувача (стилі, статичні сторінки html, javascript), а також моделі уявлень, контролери, об'єкти контексту запиту.

4.1.2. **Business layer (рівень бізнес-логіки):** містить набір компонентів, що відповідають за обробку отриманих від рівня уявлень даних, реалізує всю необхідну логіку програми, всі обчислення, взаємодіє з базою даних та передає рівню представлення результат обробки.

4.1.3. **Data Access layer (рівень доступу до даних):** зберігає моделі, що описують використовувані сутності, також розміщуються специфічні класи для роботи з різними технологіями доступу до даних, наприклад, клас контексту даних Entity Framework. Тут також зберігаються репозиторії, якими рівень бізнес-логіки взаємодіє з базою даних.

4.1.4. **ASP.NET identity:** Вбудована технологія для аутентифікації та авторизації користувачів. Потрібна для делегування обов'язків та редагування даних у системі.

4.1.5. **Entity Framework:** представляє спеціальну об'єктно-орієнтовану технологію на базі фреймворку .NET для роботи з даними. Слугує сервісом для зв'язку та управління даними з БД.



5. Технології які будуть використовуватись:

- 5.1. Середовище розробки Visual Studio 2022
- 5.2. Сервер бд MS SQL Server 2019
- 5.3. Мова програмування C#
- 5.4. Entity Framework Core + Migrations + Identity
- 5.5. Тип застосунку ASP.NET Core MVC 3.1
- 5.6. HTML5 + SASS (CSS)
- 5.7. JavaScript + jQuery

Висновки:

Я обрав MVC (модель-представлення-контроллер) тому , що цей паттерн має модульну архітектуру. Хоча на первинну розробку застосунку витратиться більше часу, ніж в інших технологія , але значно спроститься супровід системи. Також це прямо вплине на швидкість масштабування . Ми не використовуємо непотрібних методів та все прописуємо самі.

На кшталт MS SQL Server 2019 , вибір пав саме тому , що вона має достатньо зручний інтерфейс та функціонал, спрощує розгортання , передачу та інтеграцію великих даних. Має підтримку постійної пам'яті. Інші технології являються класичними та гармонійно співпрацюють з мовою програмування C#