

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

**Дисципліна  
«Архітектура та проектування програмного забезпечення»**

**Кінцевий звіт проекту**

**на тему:  
«Онлайн клієнт для погодніх даних»**

Виконав:	Гоша Давід	Перевірів:	Берестов Д.С
Група	ІІЗ-23	Дата перевірки	
Форма навчання	денна	Оцінка	
Спеціальність	121		
2022			

## Вступ

### Зміст

Вступ.....	2
Завдання.....	2
Вимоги, висунуті до системи: .....	3
Вибір компонентів.....	4
UML діаграма обраних компонентів: .....	4
Технології які використовувалися .....	5
Опис обраного архітектурного стилю та шаблону:.....	5
Основні проекти рішення .....	6
Архітектурні рішення та з'єднувачі .....	6
Опис архітектури системи (стисло).....	7
Опис основних технічних викликів.....	8
Архітектурний аналіз .....	9
Початок програмної реалізації.....	12
Кінцевий етап розробки .....	13
Висновок.....	15

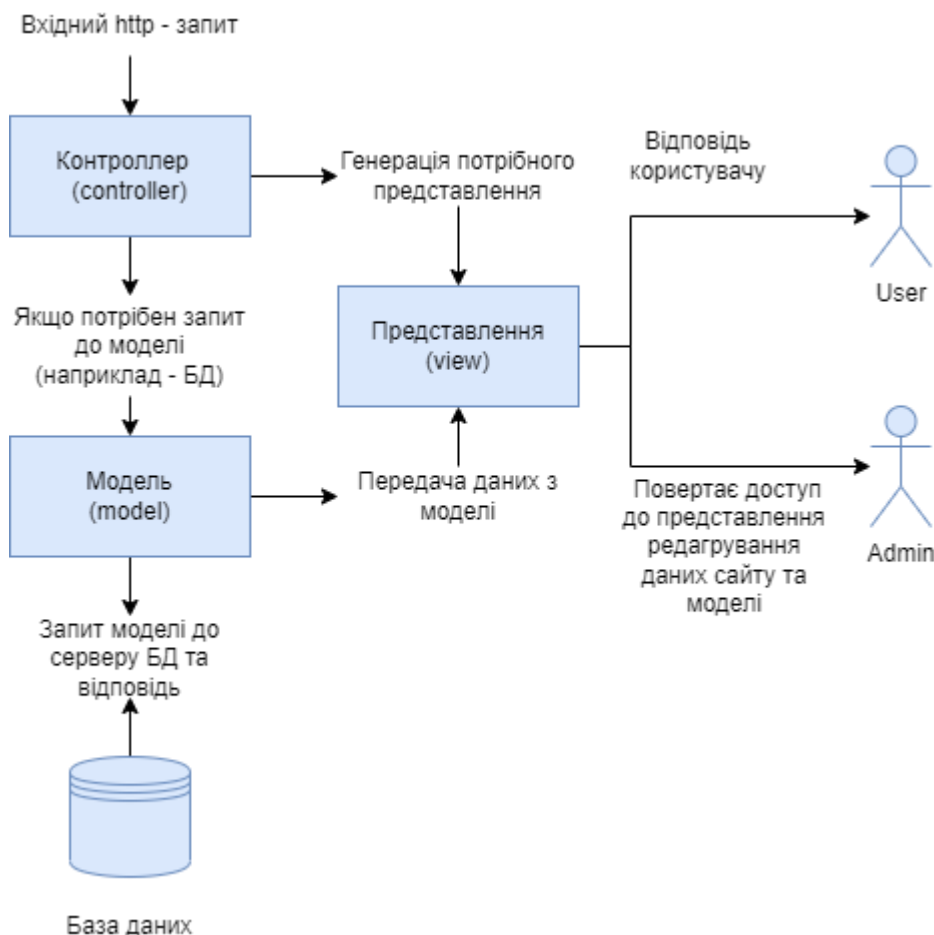
**Мета практикуму** – отримати практичні навички застосування шаблонів проектування для підвищення внутрішньої якості ПЗ.

**Тема проекту** – «Онлайн клієнт для погодних даних».

**Застосування проекту** - проект розробляється та проектується з метою публікації прогнозів погодних даних.

### Завдання

**Створення ПЗ**, яке 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.



## 1. Графічне зображення архітектури ПЗ

### Вимоги, висунуті до системи:

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.Сортування 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.

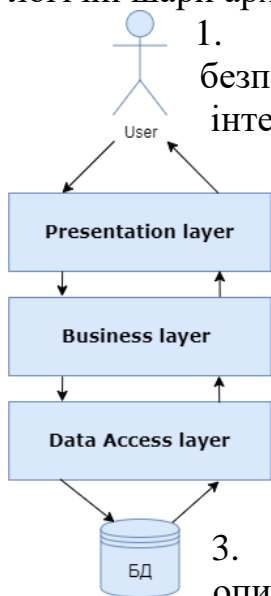
#### 3.1. Вимоги до Інтерфейсу (**Interface Requirements**).

- 3.1.1. Апаратні Інтерфейси (**Hardware Interfaces**) Апаратні інтерфейси необхідні для підтримки системи, включаючи логічну структуру, фізичні адреси і очікувану поведінку.

- 3.1.2. Інтерфейси ПЗ (**Software Interfaces**) Назви Інтерфейсів програмного забезпечення з якими аплікація повинна взаємодіяти.
- 3.1.3. Зв'язки Інтерфейсів (**Communications Interfaces**) Зв'язки інтерфейсу з іншими системами або приладами.
- 3.2. Апаратні та Програмні Вимоги (**Hardware/Software Requirements**):  
Опис апаратної та програмної платформ, необхідних для підтримки системи.
- 3.3. **Operational Requirements:**
  - 3.3.1. Безпека та Конфіденційність (**Security and Privacy**).
  - 3.3.2. Надійність (**Reliability**).
  - 3.3.3. Відновлювальність (**Recoverability**).
  - 3.3.4. Продуктивність (**Performance**).
  - 3.3.5. Потенціал (**Capacity**).
  - 3.3.6. Збереження даних (**Data Retention**).
  - 3.3.7. Керування помилками (**Error Handling**).
  - 3.3.8. Правила Перевірки (**Validation Rules**).

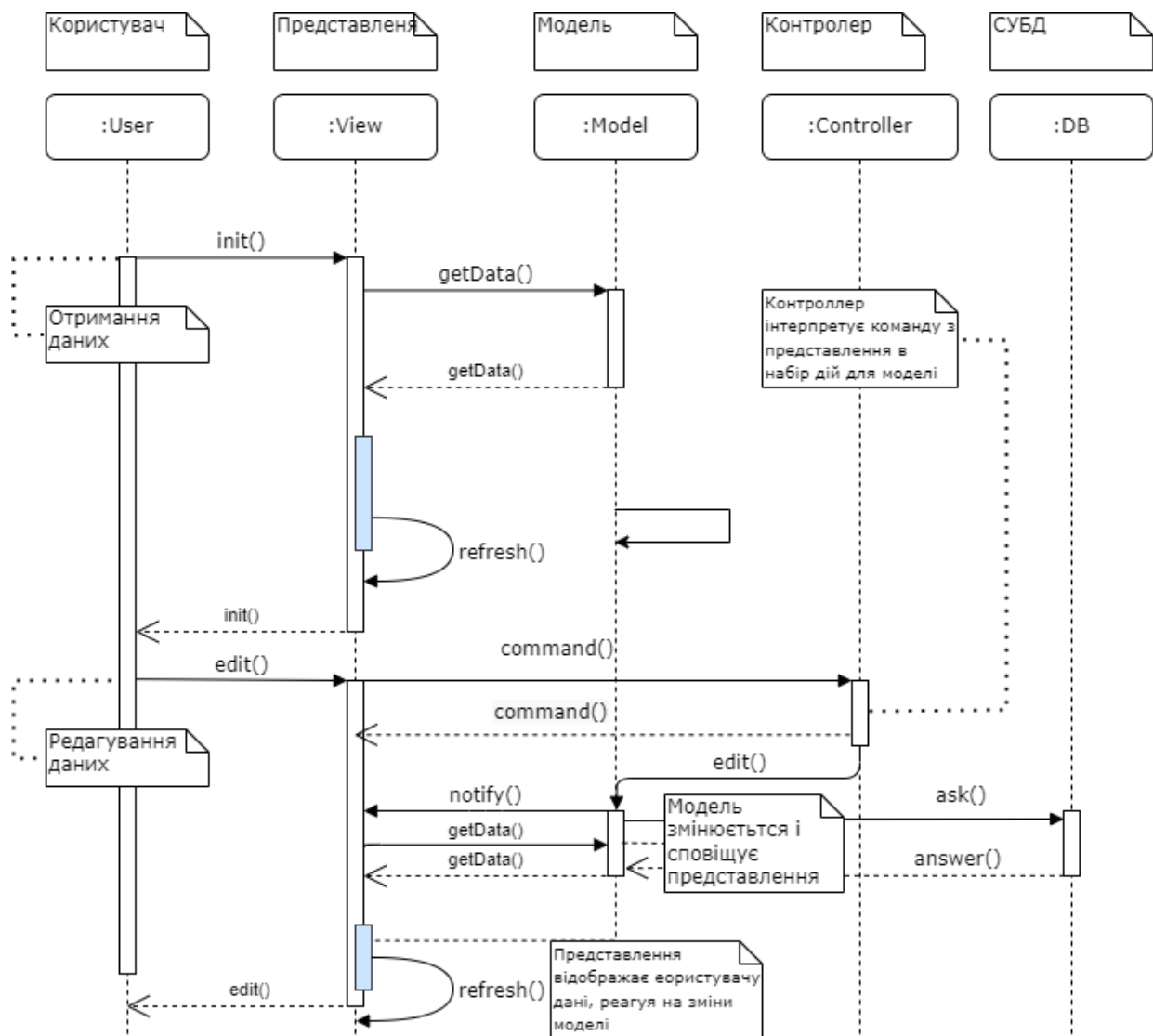
### Вибір компонентів

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



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.
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.
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. **ASP.NET identity**: Вбудована технологія для аутентифікації та авторизації користувачів. Потрібна для делегування обов'язків та редагування даних у системі.
5. **Entity Framework**: представляє спеціальну об'єктноорієнтовану технологію на базі фреймворку .NET для роботи з даними. Слугує сервісом для зв'язку та управління даними з БД.

### UML діаграма обраних компонентів:



## 2. UML діаграма обраних компонентів

### Технології які використовувалися

6. Середовище розробки Visual Studio 2022
7. Сервер бд MS SQL Server 2019
8. Мова програмування C#
9. Entity Framework Core + Migrations + Identity
10. Тип застосунку ASP.NET Core MVC 3.1 5.6. HTML5 + SASS (CSS)
11. JavaScript + jQuery

### Опис обраного архітектурного стилю та шаблону:

N-рівнева та 3-рівнева архітектура є стилями розгортання, що описують поділ функціональності на сегменти, багато в чому аналогічно багат шаровій архітектурі, але в даному випадку ці сегменти можуть фізично розміщуватися на різних комп'ютерах, їх називають рівнями. Дані архітектурні стилі було створено з урахуванням компонентно-орієнтовного підходу і, зазвичай, зв'язку використовують методи платформи, а чи не повідомлення.

Характеристиками N-рівневої архітектури програми є функціональна декомпозиція

програми, сервісні компоненти та їх розподілене розгортання, що забезпечує підвищену масштабованість, доступність, керованість та ефективність використання ресурсів. Кожен рівень абсолютно незалежний від решти, крім тих, з якими він безпосередньо сусідить. N-ному рівню потрібно лише знати, як обробляти запит від  $n+1$  рівня, як передавати цей запит на  $n-1$  рівень (якщо є), і як обробляти результати запита. Для забезпечення кращої масштабованості зв'язок між рівнями зазвичай асинхронний.

N-рівнева архітектура зазвичай має не менше трьох окремих логічних частин, кожна з яких фізично розміщується на різних серверах. Кожна частина відповідає за певну функціональність. При використанні багатошарового підходу шар розгортається на рівні, якщо функціональність, що надається цим шаром, використовується більш ніж одним сервісом або додатком рівня.

Прикладом N-рівневого /3-рівневого архітектурного стилю може бути типовий фінансовий Веб-додаток з високими вимогами до безпеки. Бізнес-шар у цьому випадку має бути розгорнутий за між мережевим екраном, через що доводиться розгортати шар представлення на окремому сервері у прикордонній мережі. Інший приклад – типовий насичений клієнт, у якому шар представлення розгорнуто на клієнтських комп'ютерах, а бізнес-шар та шар доступу до даних розгорнуті одному чи більше серверних рівнях.

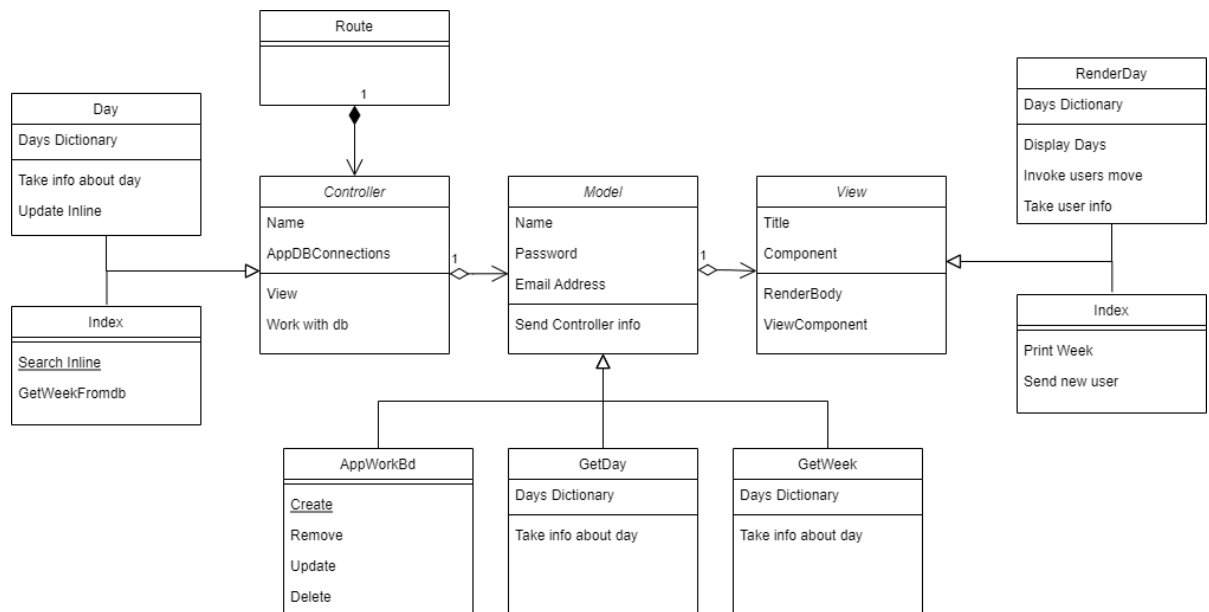
### **Основні проекти рішення**

Серед ресурсів третіх сторін було використано АПІ сайту openweathermap. Що надає можливість отримувати прогноз погоди на місяць.

Також використовувалась технологія GoogleLocationService, для отримання координат геолокації по назву вулиці, міста або населеного пункту, за допомогою АПІ Google Map.

Технологія IdentityUser, це готова бібліотека абстрактного представлення користувача. Identity є вбудованою в ASP.NET системою автентифікації та авторизації. Ця система дозволяє користувачам створювати облікові записи, автентифікувати, керувати обліковими записами або використовувати для входу на сайт облікові записи зовнішніх провайдерів, таких як Facebook, Google, Microsoft, Twitter та інші.

### **Архітектурні рішення та з'єднувачі**



### 3. Діаграма класів

Діаграма класів, показує як вибрані класи з'єднані. Тобто показує взаємодію класів у програмі. У даному випадку , ми бачимо , що кожен конкретну проблему , тобто розширення функціоналу, потрібно розкласти на 3 частини і вносити у кожний з базових класів. А саме , наприклад якщо знадобиться розширити функціонал та додати рівень тиску до кожного дня. Потрібно змінити частину бд де зберігається інформація про день та додати колонку , яка зберігатиме рівень тиску. Так як модель агрегує контроллер то у методі , що передає модель треба додати поле , що свідчить про рівень тиску. З цього випливає , що неминуча зміна інтерфейсу, де буде друкуватися рівень тиску для користувача. Ось така невелика зміна веде до купи роботи.

В той же час, для мене залишилося відкритим питання, наскільки можна застосувати концепцію MVC для розробки звичайних програм (наприклад, для Windows). На знижку можна, але не факт, що це буде оптимально. Ну і найголовніше питання: чи буду використовувати концепцію MVC, в наступних своїх проектах? Відповідь: думаю так.

### Опис архітектури системи (стисло)

#### 1. Wwroot

Папка у якій знаходиться шаблон (інтерфейсу) ,нашого застосунку

##### 1.1. CSS

У цій директорії знаходяться стилі нашого шаблону

##### 1.2. JS

У цій директорії знаходяться скрипти нашого шаблону , щоб він працював динамічно.

##### 1.3. Lib

Директорія , що містить бібліотеки

#### 1.4. Image

#### 1.5. Директорії для зображень , іконок.

### 2. Controlllers

Центральною ланкою (шаром) в архітектурі ASP.NET Core MVC є контролер. При отриманні запиту система маршрутизації вибирає обробки запиту потрібний контролер і передає йому дані запиту. Контролер обробляє ці дані та посилає назад результат обробки.

### 3. Models

Одним із ключових компонентів патерну MVC є моделі. Ключове завдання моделей - опис структури та логіки даних, що використовуються. Як правило, всі використовувані сутності у додатку виділяються в окремі моделі, які описують структуру кожної сутності. Залежно від завдань та предметної області, ми можемо виділити різну кількість моделей у додатку. Це найнижчий з шарів архітектури. Він на пряму взаємодіє з БД та надає інформацію до контроллеру.

### 4. Views

У більшості випадків при зверненні до веб-застосунку користувач очікує отримати веб-сторінку з будь-якими даними. У MVC для цього, як правило, використовуються уявлення, які формують зовнішній вигляд програми. В ASP.NET MVC Core уявлення - це файли з розширенням cshtml, які містять код інтерфейсу в основному на мові html, а також конструкції Razor - спеціального двигуна уявлень, який дозволяє переходити від коду html до коду мовою C#. Представлення містить кожну сторінку для контроллеру а саме Home – домашня директорія та Shared , що містить статичну частину шаблону. Це найвищий шар , що контактує з користувачем. Ні в якому разі він не повинен звертатися до моделі, тільки через контроллер!

### 5. Appsettings

JSON файл , що містить сатичні рядки типу підключення до БД

### 6. Program

Це головний файл нашої програми , з якого все стартує.

## **Опис основних технічних викликів**

### **Опис основних технічних викликів, ризиків та відкритих питань проекту:**

Основним ризиком та відкритим питанням залишається пошук метеорологічного центру, що буде надавати інформацію про поточні погодні умови у місті.

Також залишається питання парсингу цієї інформацію. У якому вигляді та як часто нам треба її оновлювати. На скільки широка сітка міст.

#### **Відкриті питання серверної частини:**

Запит , обмін та зберігання метеорологічних даних. У якому вигляді та яка кількість населених пунктів буде охоплена у базі даних. Як зазначалося раніше, ми будемо використовувати пропріитарну СУБД Microsoft SQL Server.

Також залишається питання з моделлю, на скільки багато інформації нам вдасться добувати , як саме публікувати її.

**Відкриті питання веб-застосунку:** –Головною проблемою є реалізація пошуку за



містом. Потреба у реалізації пошукової каретки за містом, та авто виправлення у разі введення не точної інформації.

### **Архітектурний аналіз**

На **початковому етапі** основна мета аналізу та проектування полягає в тому, щоб зрозуміти, чи можна в принципі реалізувати систему і вибрати технології для її реалізації (Операція: **синтез архітектури**). Якщо ризики для розробки невеликі (наприклад, при розробці системи у добре відомій предметній області, оновленні існуючої системи тощо), цю процедуру виконувати не обов'язково.

Це перший етап процесу RUP, основне завдання якого полягає у відповідності цілей життєвого циклу проекту, поставлених різними зацікавленими особами.

#### **Цілі:**

На початковому етапі переслідуються такі основні цілі:

Визначення змісту проекту та граничних умов, включаючи бачення, критерії приймання та розмежування того, що має входити в продукт, а що – ні.

Відмова від найскладніших варіантів, які максимально ускладнюють.

Розробка та демонстрація принаймні однієї потенційної архітектури для реалізації основних сценаріїв.

Оцінка загальної вартості та тривалості реалізації проекту (включаючи детальну оцінку витрат на етапі уточнення)

Оцінка потенційних ризиків (джерел невизначеності). розділ Концепція: ризик.

Підготовка середовища підтримки проекту.

#### **Основні операції:**

На початковому етапі виконуються такі основні операції:

Визначення змісту проекту.

Дана операція полягає у формалізації контексту та найважливіших вимог та обмежень у достатній мірі для розробки критеріїв приймання кінцевого продукту.

Планування та підготовка економічного обґрунтування проекту.

Оцінка альтернативних варіантів управління ризиками, підбору персоналу, планування проекту та компромісів між вартістю, графіком виконання та прибутковістю проектів.

Синтез потенційної архітектури

Оцінка компромісних рішень у проекті та у виборі способів реалізації окремих компонентів (купити, розробити, використовувати готовий) для прогнозування обсягу необхідних ресурсів та тривалості розробки. Мета цієї операції полягає у демонстрації можливості реалізації проекту на дослідному зразку. Роль дослідного зразка може виконувати модель, що імітує виконання необхідних функцій або початковий прототип, що дозволяє проаналізувати можливі джерела максимального ризику.

Створення прототипу на початковому етапі дозволяє переконатися в тому, що проект є справді реалістичним, і що його можна буде реалізувати на етапах уточнення та побудови.

Підготовка середовища для проекту.

У ході цієї операції проводиться аналіз проекту та організації, вибираються засоби розробки та приймаються рішення про оптимізацію окремих частин проекту.

Наступним етапом аналізу буде визначення **синтез архітектури**. До суттєвих для архітектури зазвичай входять питання, пов'язані з продуктивністю, масштабуванням, синхронізацією процесів та ниток, а також розподілу програмного забезпечення.

2. На початковому **етапі уточнення** основний пріоритет віддається створенню базової архітектури системи (Операція: **визначення гаданої архітектури**) , як відправної точки для повномасштабного аналізу. Якщо архітектура вже існує (наприклад, була створена в попередніх ітераціях або попередньому проєкті, або імпортована з середовища розробки), акцент зміщується на уточнення існуючої архітектури (Операція: **уточнення архітектури**). Створюється початковий набір елементів, які реалізують потрібну поведінку (Операція: аналіз поведінки).

#### **Визначення гаданої архітектури:**

- Створити початковий ескіз архітектури системи
- Визначити початковий набір суттєвих елементів архітектури як основу для аналізу
- Визначити початковий набір механізмів аналізу
- Визначити початкову структуру та організацію системи
- Визначити реалізацію прецедентів для поточної ітерації
- Визначити класи аналізу за варіантами, суттєвими для архітектури
- Додати взаємодію з класами аналізу у реалізації прецедентів

#### **Операція уточнення архітектури:**

- Забезпечує природний перехід від операцій аналізу до операцій проєктування та визначення наступних об'єктів:
- Елементів проєктування на основі елементів аналізу
- Механізмів проєктування на основі механізмів аналізу
- Дозволяє створити опис структури архітектури розгортання та середовища виконання системи
- Дозволяє систематизувати модель реалізації для спрощення переходу від проєктування до реалізації
- Дозволяє забезпечити цілісність та одноманітність архітектури та гарантувати виконання наступних умов:
- Нові елементи проєктування, створені або виявлені в поточній ітерації, поєднуються з елементами проєктування, що існували
- Якомога раніше досягається максимальний ефект від багаторазового використання доступних компонентів та елементів проєктування

3. Після ідентифікації початкових елементів починається їхнє уточнення. У процесі процедури (Операція: **проєктування компонентів**) створюється набір компонентів, реалізують необхідне поведінка системи. Якщо до складу системи входить база даних, паралельно з процедурою виконується процедура (Операція: **проєктування баз даних**). В результаті створюється початковий набір компонентів, які у подальшому уточнюються в (**Шаблоні можливостей: реалізація**).

## Проектування компонентів має такі цілі:

- Уточнити визначення елементів проекту деталями реалізації поведінки, що очікується від цих елементів.
- Уточнити та оновити реалізацію прецедентів на основі новостворених елементів проекту (іншими словами, регулярно оновлюйте реалізацію прецедентів)
- Уточнити проект

## Проектування баз даних передбачає виконання таких дій:

- Визначення постійних класів у проекті
- Проектування структур бази даних для зберігання постійних класів
- Опис механізмів та стратегій зберігання та вилучення постійних даних, що забезпечують досягнення необхідної продуктивності системи.
- База даних та механізми зберігання та вилучення постійних даних реалізуються

та тестуються разом з іншими компонентами та підсистемами додатка.

Для структурування процедур, що входять до складу **дисципліни реалізації**, сукупність операцій та продуктів роботи дисципліни називається шаблоном її можливостей.

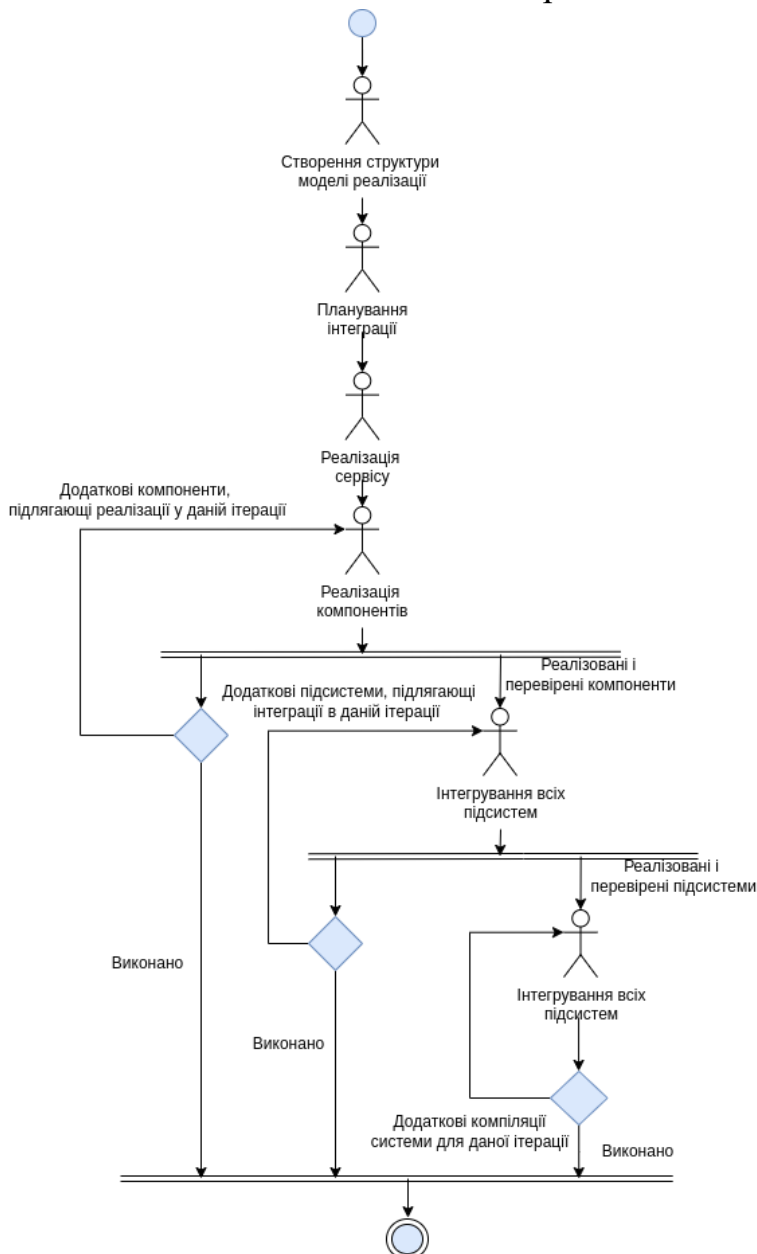
Кожна операція призначена для вирішення деякого загального завдання, без якого неможлива ефективна реалізація.

Структурування моделі реалізації виконується на початку етапу уточнення. У кожній ітерації етапу уточнення виконуватимуться такі операції: планування інтеграції, реалізація компонентів, інтеграція підсистем та інтеграція системи. Дві останні операції тісно пов'язані із тестуванням інтеграції.

## Обґрунтування вибору типів та методик архітектурного аналізу:

Я вибрав методологію розробки та аналізу програмного забезпечення RUP, тому що:

Ітераційна розробка програмного



## 4. Графічне зображення охоплює операції та потоки операцій дисципліни аналізу та проектування

забезпечення в RUP передбачає поділ проекту на кілька дрібних проектів, що виконуються послідовно, і кожна ітерація розробки чітко визначена набором цілей, які мають бути досягнуті наприкінці ітерації. Кінцева ітерація передбачає, що набір цілей ітерації повинен точно співпадати з набором цілей, зазначених замовником продукту, тобто всі вимоги повинні бути виконані.

RUP досить добре формалізований, і найбільша увага приділяється початковим стадіям розробки проекту – аналізу та моделювання. Отже, ця методологія спрямовано зниження комерційних ризиків (risk mitigating) у вигляді виявлення помилок на ранніх стадіях розробки. Технічні ризики (assesses) оцінюються і «розставляються» згідно з пріоритетами на ранніх стадіях циклу розробки, а потім переглядаються з часом та з розвитком проекту протягом наступних ітерацій. Нові цілі виникають залежно від пріоритетів цих ризиків. Релізи версій розподіляються таким чином, що найпріоритетніші ризики усуваються першими.

Результати проведеного архітектурного аналізу:

За вище перерахованими перевагами, ми можемо зробити висновки о результатах проведеної роботи. Визначивши цілі і характеристики проекту, ми уникнули головних ризиків під час роботи, проаналізувавши складність проекту і можливі методи реалізації поставлених задач.

Перевагою являється концентрація на виконанні вимог замовників до виконуваної програми (аналіз та побудова моделі прецедентів (варіантів використання)), що ми робили у минулій лабораторній.

Модель аналізу дозволяє очікування змін у вимогах, проектних рішеннях та реалізації у процесі розробки. Що робить проектування більш зручним і направленим на замовника.

Компонентна архітектура, що реалізується та тестується на ранніх стадіях проекту.

Постійне забезпечення якості на всіх етапах розробки проекту.

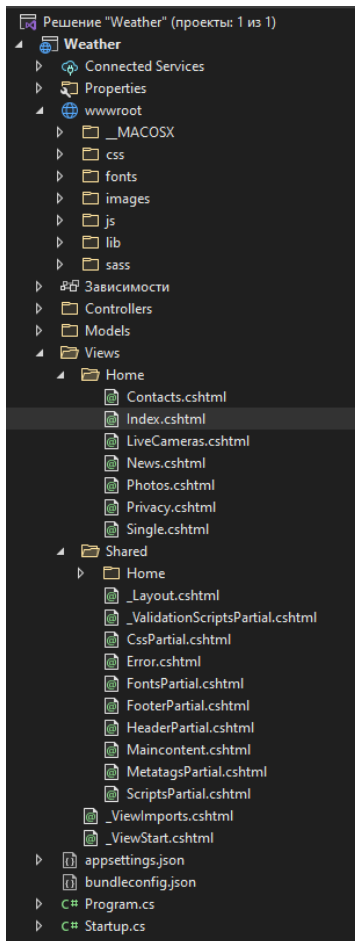
### **Початок програмної реалізації**

Наразі розроблено інтерфейс(frontend). Додано директорії для структури архітектури Asp-net core MVC. У кожен папку додано потрібні файли що до архітектурних вимог. Також було завантажено візуальний редактор SKeditor, щоб згодом працівники компанії могли у панелі адміністратора додавати або видаляти певний контент.

**Я вибрав платформу ASP.NET MVC тому що, вона має такі переваги.**

- Полегшує управління складними структурами шляхом поділу програми на модель, уявлення та контролер.
- Не використовує стан перегляду та серверні форми. Це робить платформу MVC ідеальною для розробників, які потребують повного контролю над поведінкою програми.

- Використовує схему основного контролера, коли запити веб-програми обробляються через один контролер. Це дозволяє створювати програми, які підтримують розширену інфраструктуру маршрутизації. Для отримання додаткових відомостей див. Інтерфейсний контролер на веб-сайті MSDN.



- Забезпечує розширену підтримку розробки з урахуванням тестування.

- Добре підходить для веб-додатків, які підтримуються великими командами розробників та веб-дизайнерами, які потребують високого рівня контролю над поведінкою програми.

Статус проекту:

Можна сказати, що візуально проект готовий, залишається тільки додати логіку парсингу даних про погодні умови і налаштувати доменну модель.

Інструмент Identity та спроектувати базу даних.

### Опис стан справ з розробкою конкретних архітектурних елементів із посиланнями на них:

Представлення повністю розроблені. Що до контролерів, будемо намагатися обмежитися двома контролерами(хоум та адмін ареа). Але в разі потреби додамо ще один. Зараз закінчено роботу над один з них.

Що до моделі, вона, разом із базою даних зовсім не

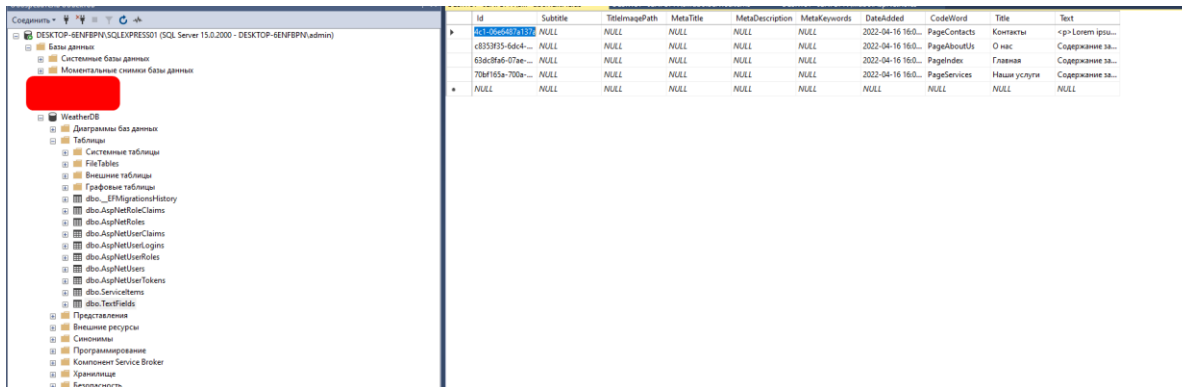
### 5. Приклад розгорнутої архітектури

готова. Залишається проектування бази та доменної моделі, зони адміністрації та логіка парсингу погодніх даних з метеорологічного центру.

### Кінцевий етап розробки

На кінцевому етапі розробки було додано адмін панель, для редагування сторінок сайту, зони користувачів такі як адміністраторська зона та зона користувача, що обмежені у доступі авторизацією і правами користувачів, що знаходяться у базі даних, так як і записи.

Було додано і спроектовано базу даних, внесені деякі користувачі та записи сторінок сайту.

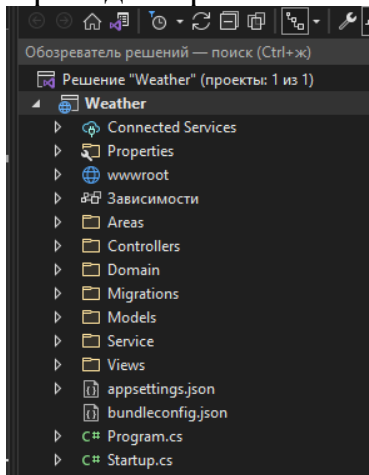


The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Database' folder is expanded, showing a list of tables including 'dbo.AspNetUsers', 'dbo.AspNetUserRoles', 'dbo.AspNetUserClaims', 'dbo.AspNetUserLogins', 'dbo.AspNetUserTokens', 'dbo.ServiceItems', 'dbo.TestFields', 'dbo.Users', 'dbo.Roles', 'dbo.Claims', 'dbo.Logins', 'dbo.Tokens', 'dbo.Items', 'dbo.Fields', 'dbo.Users', 'dbo.Roles', 'dbo.Claims', 'dbo.Logins', 'dbo.Tokens', 'dbo.Items', 'dbo.Fields'. On the right, a table view is displayed with columns: 'id', 'Subtitle', 'TitleImagePath', 'MetaTitle', 'MetaDescription', 'MetaKeywords', 'DateAdded', 'CodeWord', 'Title', 'Text'. The table contains several rows of data, including one with 'id' 1 and 'Title' 'PageContacts'.

id	Subtitle	TitleImagePath	MetaTitle	MetaDescription	MetaKeywords	DateAdded	CodeWord	Title	Text
1						2022-04-16 16:00:00	PageContacts	Контакты	«грі-Лоретт» про...
2						2022-04-16 16:00:00	PageAboutUs	О нас	Содержание та...
3						2022-04-16 16:00:00	PageIndex	Главная	Содержание та...
4						2022-04-16 16:00:00	PageServices	Наши услуги	Содержание та...
5						2022-04-16 16:00:00	PageServices	Наши услуги	Содержание та...

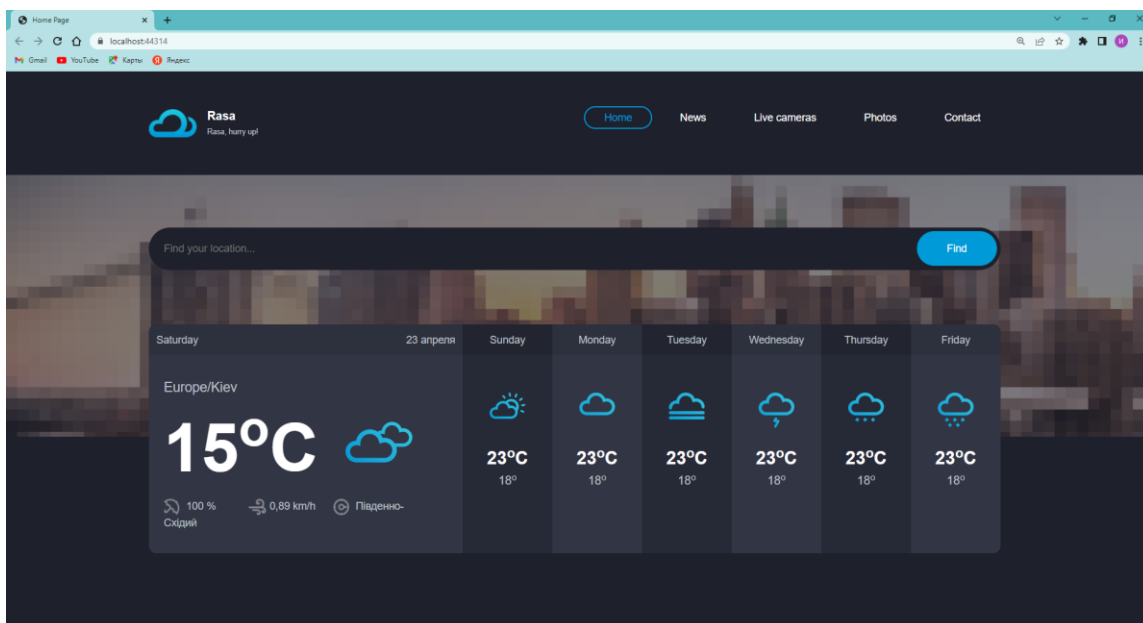
## 6. Архітектура БД та приклад одної з таблиць.

Проведен привентивний рефакторінг коду та компіляція і публікація на гітхаб.

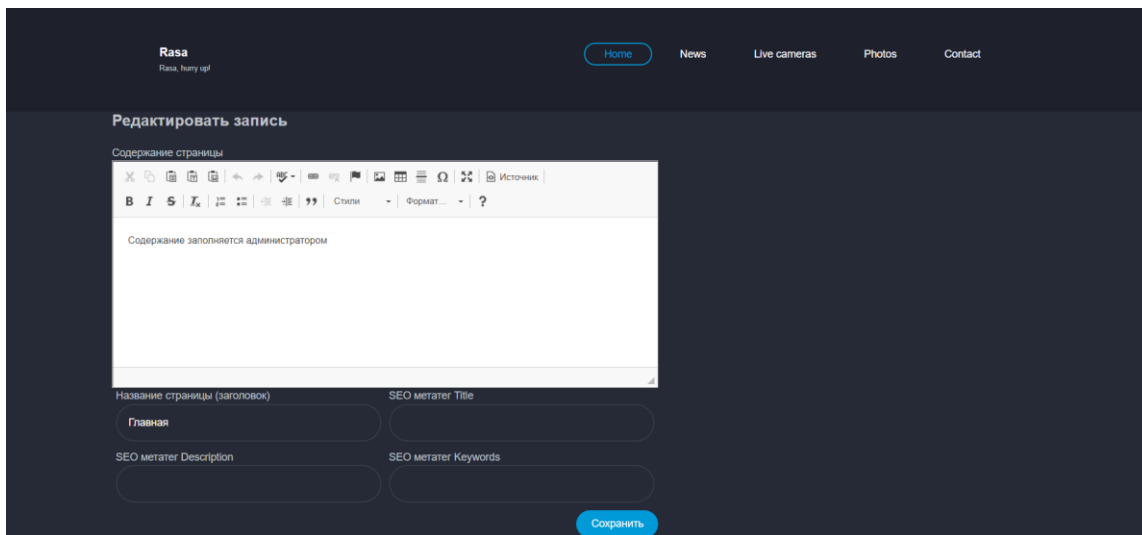


## 7. Максимально короткий зміст проекту програмного забезпечення

Далі декілька прикладів сторінок.



## 8. Головна сторінка



## 9. Адмін панель

### Висновок

В ході данного проекту був розроблений програмний подукт, що дозволе користувачам преревіряти погодні умови за містом. Проект виконано у вигляді сайту, що надає метереологічні дані. Сайт має простий, зрозумілий користувачеві інтерфейс. Також має власну базу даних, куди додаються записи адміністратором, чи нові користувачі за ролями. Було розроблено систему ролів як користувач або адміністратор, що мають різні права та обов'язки у системі.

Побудовано архітектурну модель розгортання ПЗ, яка підтримує реалізацію необхідної для вашого проекту якості сервісу, виконано аналіз моделі .Реалізовано розгортання розробленого застосунку, підготовано його до демонстрації. У ході роботи, обов'язково продокументовано будь-які зміни архітектури застосування.