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»

Лабораторна робота № 6
«Реалізація та розгортання програмного забезпечення»

на тему:
«Онлайн клієнт для погодних даних»

Виконав:	Гоша Давід	**Перевірив**:	Берестов Д.С
Група	ІПЗ-23	Дата перевірки	
Форма навчання	денна	Оцінка	
Спеціальність	121		

2022

Мета практикуму – отримати практичні навички реалізації та розгортання ПЗ.

Тема проекту – «Онлайн клієнт для погодних даних».

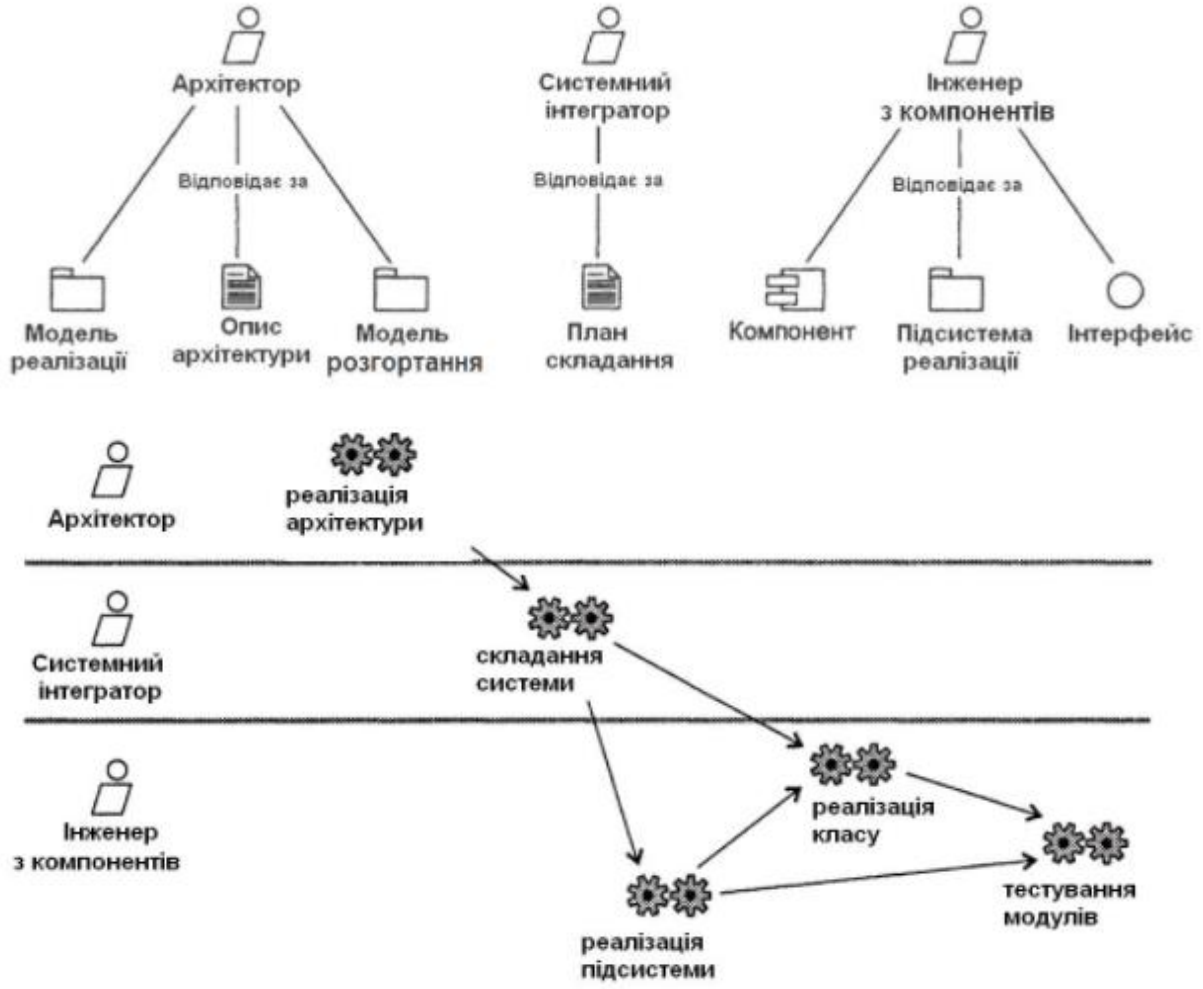
Застосування проекту - проект розробляється та проектується з метою публікації прогнозів погодних даних.

Опис основних архітектурних рішень:

1. **Архітектурне рішення:**
   Як визначалось у попередній роботі , застосунку не вистачало бази даних, парсингу погодних умов з сайту метереології та невеликої зміни візуальної частини. Тому з попередньої роботи було додано пошук за містом та парсинг JSON АПІ з сайту openweathermap, що є погодним провайдером по всій Європі. Але у момент укладення договору з метереологічної фірмою сталася проблема, безкоштовно інформація надається тільки на 2 дні , тому прогноз саме на сьогодні.
   Також було додано представленя та домена модель БД, яка у настуній лабораторній буде спроектована і готова до використання.
   На даний момент, дані з апі провайдера просто кешуються сервером і у такому вигляді зберігаються.
2. **Атрибути якості:**
   Саме вибір шаблону проектування а саме: ASP.NET Core MVC першочергово вплинув на якість продукту. Як зазначалось раніше, цей шаблон проектування має модульну архітектуру , тож кожний модуль або сервіс потрібно підключати в ручну. Ось наприклад саме отримання даних від серверу погодних прогнозів є сервісом , який я без проблем підключив у застосунок.
   Так можна робити з будь яким функціоналом, що забезпечує безперебійну роботу застосунку та простоту маштабування.
3. **Методики, техніки, шаблони, або ресурси третіх сторін, які використано:**
   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 пункта, за допомогою АПІ Google Map.
   Технологія IdentityUser, це готова бібліотека абстрактного представленя 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.
4. **Вплив на загальну архітектуру:**
   Загалом значних змін у представлені архітектури з першої лабораторної не сталося. Ні один з компонентів тою чи іншою мірою не вплинув на

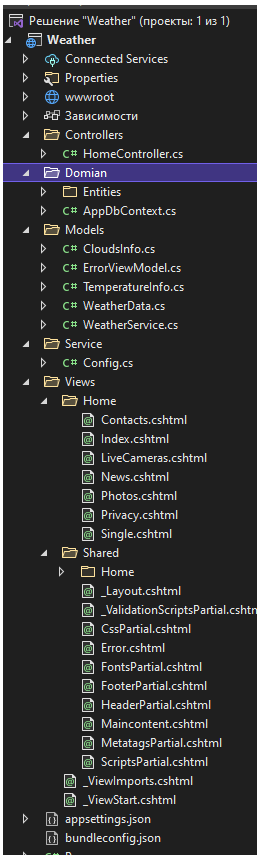
плани прозробки. Завдяки модульності та гнучкості шаблону проектування, можна проаналізувати і поетапно розбити розробку на сервіси і спрогнозувати термін, можливість та сутність архітектури.

Візуалізацію архітектурної моделі розгортання:
Розгортання є процесом, у ході якого розроблювальний продукт доставляється до кінцевого користувача. У ході цього процесу розробляється нова версія системи, поширення ПЗ, його установлення на боці кінцевого користувача та його навчання навичок ефективної роботи з поставленим ПЗ, надання послуг з технічного підтримання, бета-тестування і т. ін.
Після визначення складових компонентів моделі створюється опис інтерфейсів їх взаємозв'язку та розробляється ***план складання***, що дає опис послідовності ітерацій. Для кожного білду план містить опис:



Опис розвитку архітектури застосування протягом проекту:
З початку розробки програмного забезпечення планувалась архітектура MVC. Так як з початку розробки ніяких проблем при виконання ТЗ не виникло, було вирішено не змінювати архітектуру.

Один або два блоки коду:



Структура проекту 1

Приклад серівсу з обробки погодних даних

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using GoogleMaps.LocationServices;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Caching.Memory;
using System.Net;
using System.IO;
using Newtonsoft.Json;
using System.Web;
using Microsoft.AspNetCore.Http;
using Weather.Service;

namespace Weather.Models
{
    public class WeatherService : BackgroundService
    {
        private readonly IMemoryCache _memoryCache;
        public static string City;
        public WeatherService(IMemoryCache memoryCache)
        {
            this._memoryCache = memoryCache;
        }
        public MapPoint getCoordinats(string address)
        {
            try
            {

                MapPoint point;
                var locationService = new GoogleLocationService(Config.GoogleApi);
                return point = locationService.GetLatLongFromAddress(address);
            }
```

```
            catch (System.Net.WebException ex)
            {
                Console.WriteLine("Google Maps API Error {0}", ex.Message);
                return null;
            }

        }
        public WeatherData NewCity(string City)
        {
            try
            {
                Thread.CurrentThread.CurrentCulture = new
System.Globalization.CultureInfo("ru-RU");
                Encoding.RegisterProvider(CodePagesEncodingProvider.Instance);
                if (City == null)
                {
                    City = "Kyiv";
                }
                MapPoint geo = getCoordinats(City);
                if (geo == null)
                {
                    geo = getCoordinats("Kyiv");
                }
                string url =
$"https://api.openweathermap.org/data/2.5/onecall?lat={geo.Latitude}&lon={geo.Longitude}&ex
clude=hourly,daily&appid={Config.Weatherapi}&mode=json&units=metric";
                HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(url);
                HttpWebResponse httpWebResponse =
(HttpWebResponse)httpWebRequest.GetResponse();
                string responce;
                using (StreamReader streamReader = new
StreamReader(httpWebResponse.GetResponseStream()))
                {
                    responce = streamReader.ReadToEnd();
                }
                WeatherData weatherData =
JsonConvert.DeserializeObject<WeatherData>(responce);
                _memoryCache.Set($"Somekey", weatherData, TimeSpan.FromMinutes(1440));
                return weatherData;
            }
            catch (OperationCanceledException)
            {
                return null;
            }

        }
        protected async override Task ExecuteAsync(CancellationToken stoppingToken)
        {
            while (!stoppingToken.IsCancellationRequested)
            {
                try
                {
                    Thread.CurrentThread.CurrentCulture = new
System.Globalization.CultureInfo("ru-RU");
                    Encoding.RegisterProvider(CodePagesEncodingProvider.Instance);
                    if (City == null)
                    {
                         City = "Kyiv";
                    }
                    MapPoint geo = getCoordinats(City);

                    string url =
$"https://api.openweathermap.org/data/2.5/onecall?lat={geo.Latitude}&lon={geo.Longitude}&ex
clude=hourly,daily&appid=1883dea4cee28a3f1184c73d1cabe6da&mode=json&units=metric";
                    HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(url);
                    HttpWebResponse httpWebResponse =
(HttpWebResponse)httpWebRequest.GetResponse();
                    string responce;
                    using (StreamReader streamReader = new
StreamReader(httpWebResponse.GetResponseStream()))
```

```
                    {
                        responce = streamReader.ReadToEnd();
                    }
                    WeatherData weatherData =
JsonConvert.DeserializeObject<WeatherData>(responce);
                    _memoryCache.Set($"Somekey", weatherData, TimeSpan.FromMinutes(1440));
                }
                catch (OperationCanceledException)
                {

                }
                await Task.Delay(3000000, stoppingToken);
            }
        }
    }
}
```

Приклад моделі даних , що десереалізуються

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Weather.Models
{
    public class WeatherData
    {
        public string City { get; set; }
        public string timezone { get; set; }
        public TemperatureInfo Current { get; set; }

    }
}
```

Висновок

В ході даної лабораторної роботи був реалізований застосунок шляхом реалізації компонентів та проблемно-орієнтованих компонентів. 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.