|  |  |  |  |
| --- | --- | --- | --- |
| 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 № 5**  «Створення та дослідження архітектурного прототипу 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 підготовки та проведення архітектурного аналізу.

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

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

**Поточний статус проекту та обґрунтування вибору архітектурного каркасу:**

Наразі розроблено інтерфейс(frontend). Додано директорії для структури архітектури Asp-net core MVC. У кожну папку додано потрібні файли що до архітектурних вимог. Також було завантажено візулaьний редактор CKeditor, 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.

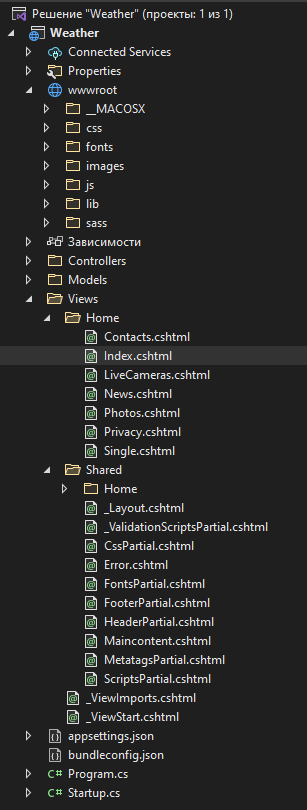
**Статус проекту:**

Можна сказати ,що візуально проект готовий, залишается тільки додати логіку парсингу даних про погодні умови і налаштувати доменну модель. Інструмент Identity та спроектувати базу даних.

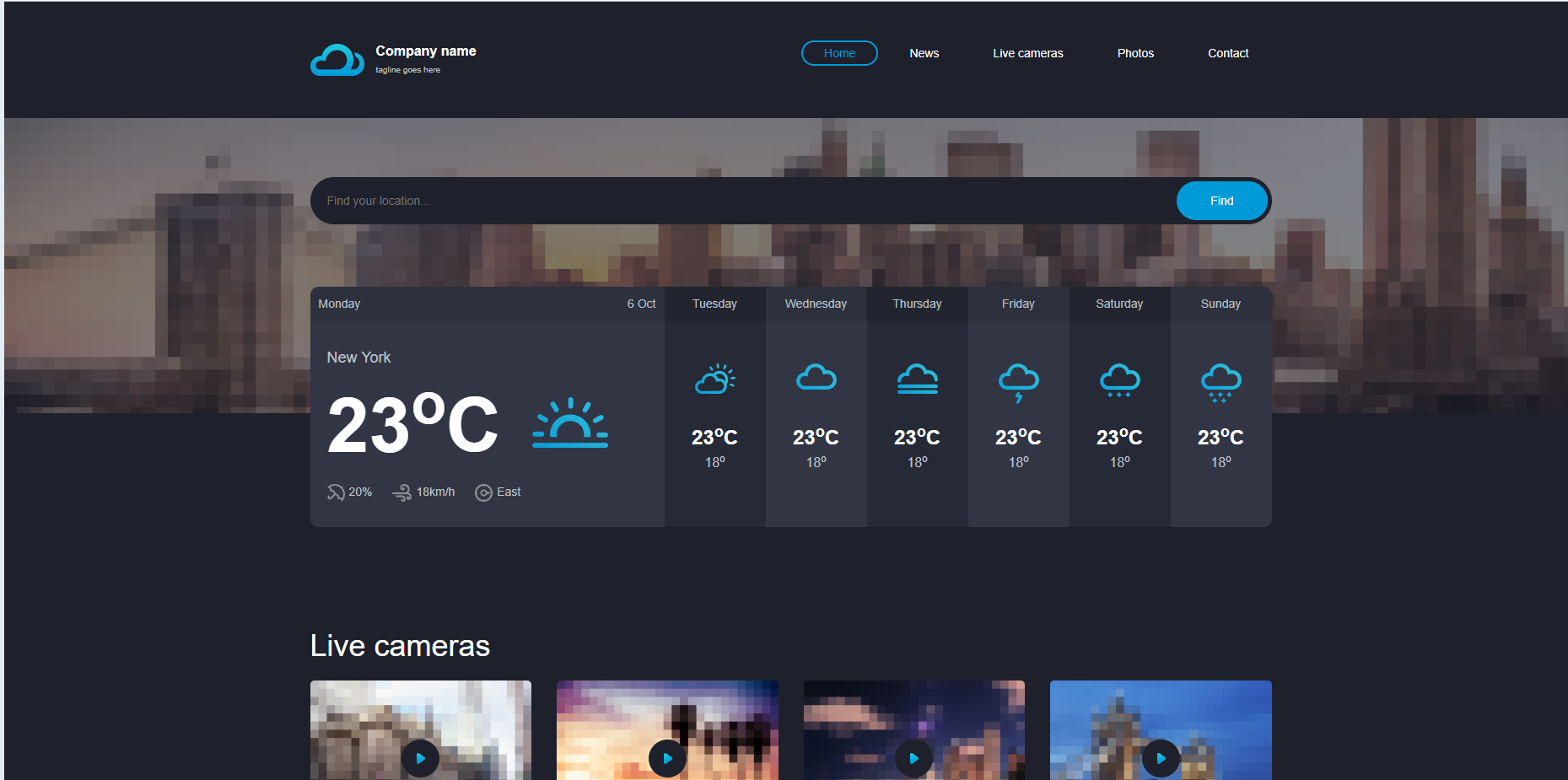
**Опис стан справ з розробкою конкретних архітектурних елементів із посиланнями на них:**

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 готова. Залишаеться проектування бази та доменої моделі , зони адміністрації та логіка парсингу погодних даних з метереологічного центру.

**Фрагменти коду:**



Структура проекту 1



Приклад головної сторінки 1

Приклад контролеру(Home). Створений для адресації та маршутизації нашого сайта , слугує посередником моделі та представлення.

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Threading.Tasks;

using Weather.Models;

namespace Weather.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

return View();

}

public IActionResult News()

{

return View();

}

public IActionResult LiveCameras()

{

return View();

}

public IActionResult Photos()

{

return View();

}

public IActionResult Single()

{

return View();

}

public IActionResult Contacts()

{

return View();

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

Головне представлення, що слугує викликом інших частин сторінки таких як Footer, Header, стилів та динамічної частини контенту.

<!DOCTYPE html>

<html lang="en">

<head>

@await Html.PartialAsync("MetatagsPartial")

<!-- Loading third party fonts -->

@await Html.PartialAsync("FontsPartial")

<!-- Loading main css file -->

@await Html.PartialAsync("CssPartial")

<!--[if lt IE 9]>

<script src="js/ie-support/html5.js"></script>

<script src="js/ie-support/respond.js"></script>

<![endif]-->

</head>

<body>

<div class="site-content">

<!-- .site-header -->

@await Html.PartialAsync("HeaderPartial")

@RenderBody()

<!-- .site-footer -->

@await Html.PartialAsync("FooterPartial")

</div>

<!-- .scripts -->

@await Html.PartialAsync("ScriptsPartial")

</body>

</html>

@{

ViewData["Title"] = "Home Page";

}

@await Html.PartialAsync("CssPartial")

<div class="hero" data-bg-image="images/banner.png">

<div class="container">

<**form** action="#" class="find-location">

<input type="text" placeholder="Find your location...">

<input type="submit" value="Find">

</**form**>

</div>

</div>

<div class="forecast-table">

<div class="container">

<div class="forecast-container">

<div class="today forecast">

<div class="forecast-header">

<div class="day">Monday</div>

<div class="date">6 Oct</div>

</div> <!-- .forecast-header -->

<div class="forecast-content">

<div class="location">New York</div>

<div class="degree">

<div class="num">23<sup>o</sup>C</div>

<div class="forecast-icon">

<**img** src="~/images/icons/icon-1.svg" alt="" width=90>

</div>

</div>

<span><**img** src="~/images/icon-umberella.png" alt="">20%</span>

<span><**img** src="~/images/icon-wind.png" alt="">18km/h</span>

<span><**img** src="~/images/icon-compass.png" alt="">East</span>

</div>

</div>

<div class="forecast">

<div class="forecast-header">

<div class="day">Tuesday</div>

</div> <!-- .forecast-header -->

<div class="forecast-content">

<div class="forecast-icon">

<**img** src="~/images/icons/icon-3.svg" alt="" width=48>

</div>

<div class="degree">23<sup>o</sup>C</div>

<small>18<sup>o</sup></small>

</div>

</div>

<div class="forecast">

<div class="forecast-header">

<div class="day">Wednesday</div>

</div> <!-- .forecast-header -->

<div class="forecast-content">

<div class="forecast-icon">

<**img** src="~/images/icons/icon-5.svg" alt="" width=48>

</div>

<div class="degree">23<sup>o</sup>C</div>

<small>18<sup>o</sup></small>

</div>

</div>

<div class="forecast">

<div class="forecast-header">

<div class="day">Thursday</div>

</div> <!-- .forecast-header -->

<div class="forecast-content">

<div class="forecast-icon">

<**img** src="~/images/icons/icon-7.svg" alt="" width=48>

</div>

<div class="degree">23<sup>o</sup>C</div>

<small>18<sup>o</sup></small>

</div>

</div>

<div class="forecast">

<div class="forecast-header">

<div class="day">Friday</div>

</div> <!-- .forecast-header -->

<div class="forecast-content">

<div class="forecast-icon">

<**img** src="~/images/icons/icon-12.svg" alt="" width=48>

</div>

<div class="degree">23<sup>o</sup>C</div>

<small>18<sup>o</sup></small>

</div>

</div>

<div class="forecast">

<div class="forecast-header">

<div class="day">Saturday</div>

</div> <!-- .forecast-header -->

<div class="forecast-content">

<div class="forecast-icon">

<**img** src="~/images/icons/icon-13.svg" alt="" width=48>

</div>

<div class="degree">23<sup>o</sup>C</div>

<small>18<sup>o</sup></small>

</div>

</div>

<div class="forecast">

<div class="forecast-header">

<div class="day">Sunday</div>

</div> <!-- .forecast-header -->

<div class="forecast-content">

<div class="forecast-icon">

<**img** src="~/images/icons/icon-14.svg" alt="" width=48>

</div>

<div class="degree">23<sup>o</sup>C</div>

<small>18<sup>o</sup></small>

</div>

</div>

</div>

</div>

</div>

<!-- .main-content -->

@await Html.PartialAsync("Maincontent")

@await Html.PartialAsync("ScriptsPartial")

Представленя за замовченням, головна або індексна сторінка сайту.

using System;

namespace Weather.Models

{

public class ErrorViewModel

{

public string RequestId { get; set; }

public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);

}

}

Приклад моделі , слугує як сортувальник помилок сайту. Згодом додам модель , що приймає погодні данні з бази.

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Weather

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews()

.SetCompatibilityVersion(Microsoft.AspNetCore.Mvc.CompatibilityVersion.Version\_3\_0).AddSessionStateTempDataProvider();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

});

}

}

}

Головний файл запуску сайту, який налагоджує протокол шифрування звязку між клієнтом та сервером. Встановлює котроллер за замовченням

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

Основним ризиком та відкритим питанням залишається пошук метереологічного центу , 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 метереологічних даних. У якому вигляді та яка кількість населених пунктів буде охоплена у базі даних. Як зазначалося раніше, ми будемо використовувати пропріоритарну 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.

**Висновок**

В ході даної лабораторної роботи був проведений архітектурний аналіз ПЗ. Визначені основні методики та етапи аналізу, та його проведення.