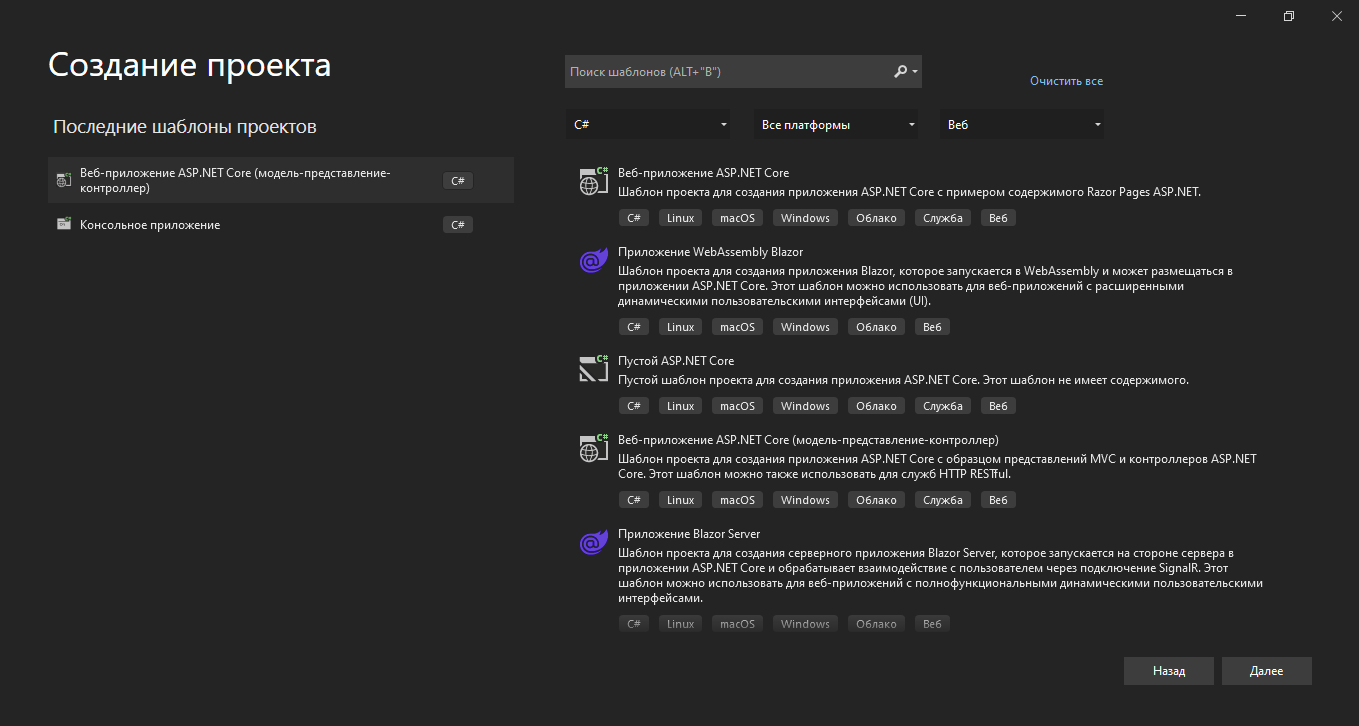
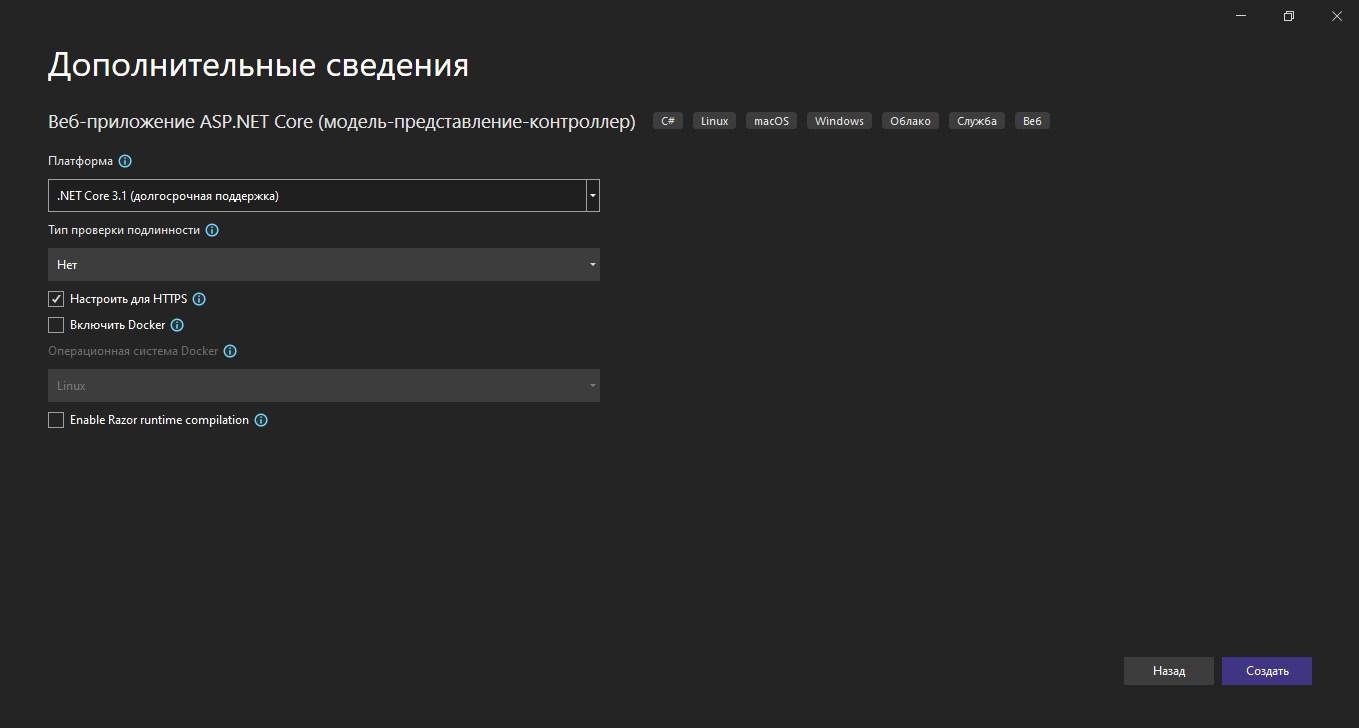
|  |  |  |  |
| --- | --- | --- | --- |
| МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ  **Кафедра програмних систем і технологій**  Дисципліна  **«Архітектура та проектування програмного забезпечення»**  **Лабораторна робота № 2**  **«Попереднє проектування архітектури програмного забезпечення»**  **на тему:**  Онлайн клієнт для погодних даних | | | |
| **Виконав:** | Гоша Д.О | **Перевірив**: | Берестов Д.С |
| Група | ІПЗ-23 | Дата перевірки |  |
| Форма навчання | денна | Оцінка |  |
| Спеціальність | 121 |
| 2022 | | | |

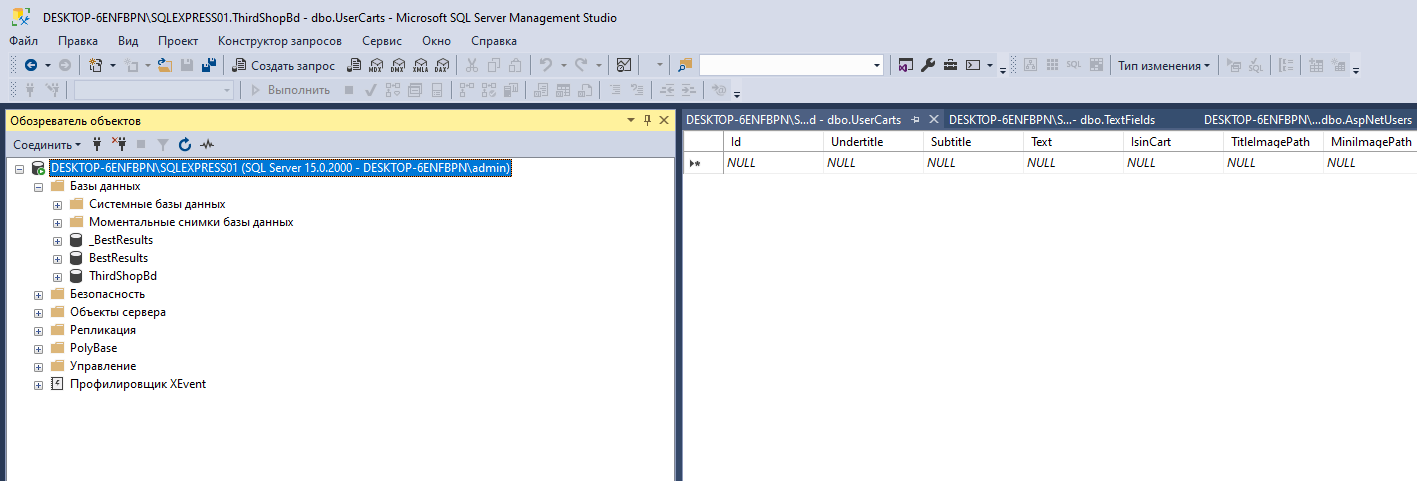
**Мета:** отримати практичні навички у підготовці існуючих компонентів до інтеграції у задане застосування, обґрунтованому виборі архітектурних стилів та шаблонів, описі архітектури ПЗ.

# Опис перевірки та тестування обраних компонентів



Стовремо веб застосунок на C# , та згенеруємо паттерн MVC. Виберемо назву та версію фреймворку .

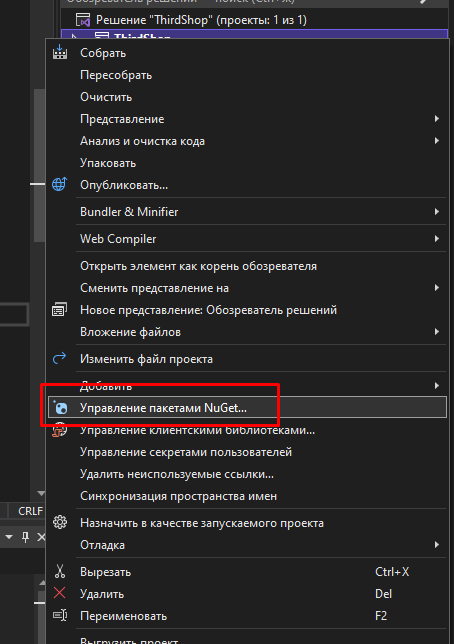


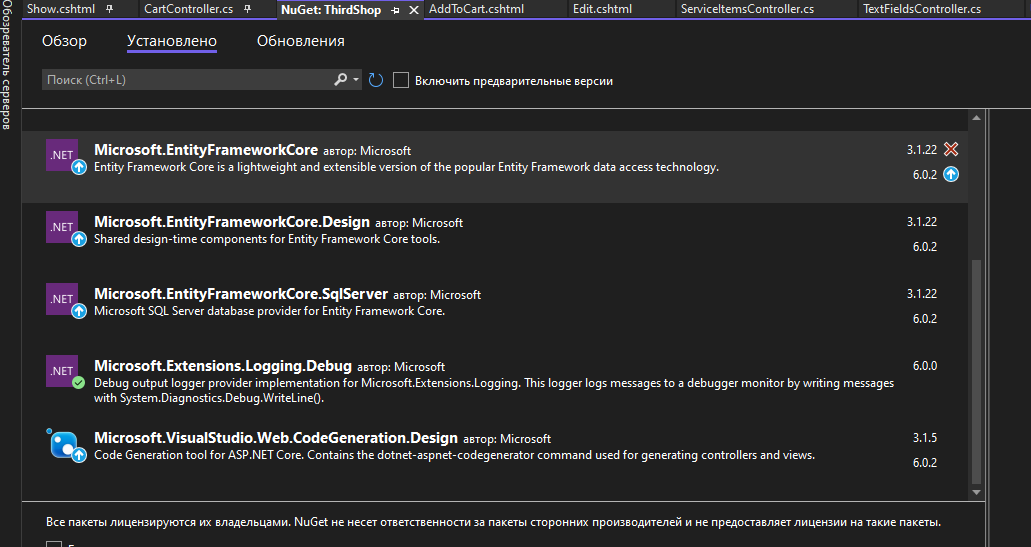


Завнтажимо MSSQl Та перевыримо створення баз. Все працює.

Тепер технології Identity та Entity Framework для аутентифікації та авторизації та роботи з БД.

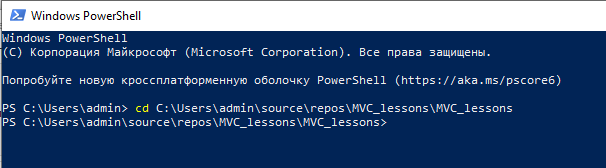
Перейдемо в менеджер пакетів NUGET та завантажимо їх.



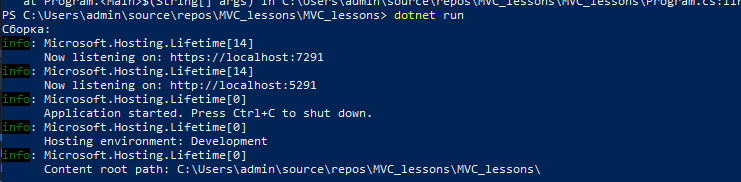


# Тестування компонентів

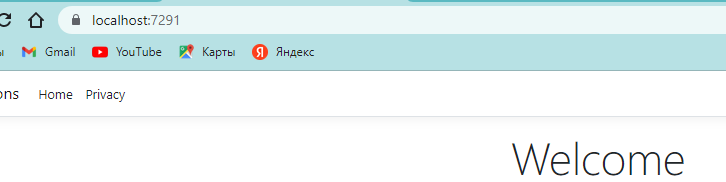
Будемо перевіряти проект , скомпілювавши його і запустивши через командний рядок PowerShell



Потрапимо в директорію проекта , де знаходиться виконуючий файл .csproj та скомпілюємо проект командою dotnet run

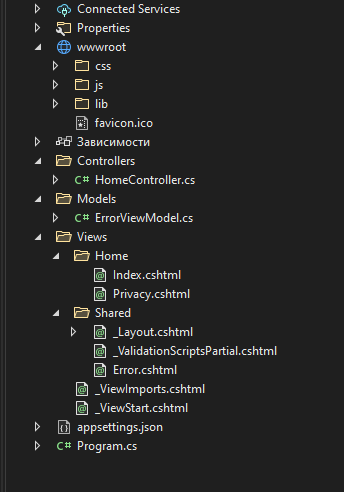


Бачимо , що наш проект скомпілювався і знаходиться на локалькому хосту під портом 5291 (http://localhost:5291). Спробуємо скористатися браузером , щоб потрапити на наш веб застосунок.



Переконавшись, що все працює як потрібно . Усі компоненти встановлені , оновлені до останніх версій. Пряцюють та виконують свою роботу за призначеням у повному обсязі. А саме пропрієтарна система управління базами даних Microsoft SQL , сприймає запити , зберігає дані та відповідає усім завданням ТЗ та правилам наійность описаних у попередній роботі. Наша IDE (Microsoft Visual Studio) компілює проеат та автоматично хостить на локальному сервері нашого компютера . Бразузер бачить наш сайт. Усі компоненти встановлені

# Опис архітектури системи



1. Wwwroot   
    Папка у якій знаходиться шаблон (інтерфейсу) ,нашого застосунку
   1. CSS  
      У цій директорії знаходяться стилі нашого шаблону
   2. JS  
      У цій директорії знаходться скрипти нашого шаблону , щоб він працював динамічно.
   3. Lib  
      Директорія , що містить бібліотеки
   4. Image
   5. Директорії для картинок , іконок.
2. Controllers  
   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 контрноллер!
5. Appsettings  
   JSON файл , що містить сатичні рядки типу підключення до БД
6. Program  
   Це мейн нашої програми , з якого все стартує.

# Архітектурні стилі та шаблони

# Клієнт/серверна

# Система поділяється на дві програми, де клієнт виконує запити до сервера. У багатьох випадках у ролі сервера виступає база даних, а логіка програми представлена ​​процедурами зберігання.

# Компонентна архітектура

# Дизайн програми розкладається на функціональні або логічні компоненти з можливістю повторного використання, що надають ретельно опрацьовані інтерфейси зв'язку.

# Багатошарова архітектура

# Функціональні області програми поділяються на багатошарові групи (рівні).

# Об'єктно-орієнтований архітектурний стиль

# Об'єктно-орієнтований архітектурний стиль, орієнтований на моделювання сфери ділової активності та визначальний бізнес-об'єкти на підставі сутності цієї сфери.

# Шина повідомлень

# Архітектурний стиль, що наказує використання програмної системи, яка може приймати та відправляти повідомлення по одному або більше каналах зв'язку, так що програми отримують можливість взаємодіяти, не маючи конкретних відомостей один про одного.

# N-рівнева/3-рівнева

# Функціональність виділяється в Функціональність виділяється в окремі сегменти, багато в чому аналогічно багатошаровому стилю, але в даному випадку сегменти фізично розташовуються на різних комп'ютерах.

# Сервісно-орієнтована

# Описує програми, що надають і архітектура (SOA) споживають функціональність у вигляді сервісів за допомогою контрактів та повідомлень.

**Опис обраного архітектурного стилю та шаблону:**

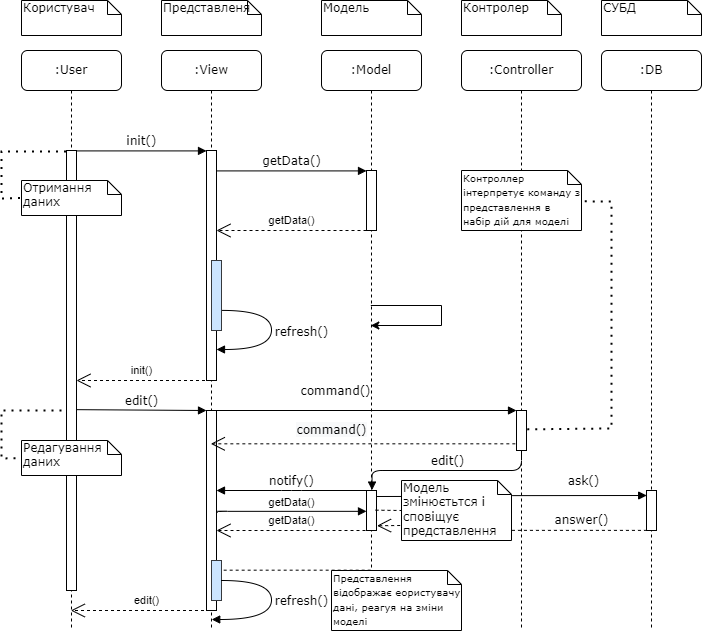
N-рівнева та 3-рівнева архітектура є стилями розгортання, що описують поділ функціональності на сегменти, багато в чому аналогічно багатошаровій архітектурі, але в даному випадку ці сегменти можуть фізично розміщуватися на різних комп'ютерах, їх називають рівнями. Дані архітектурні стилі було створено 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.

# UML діаграма обраних компонентів:



**Висновок**

У данному проекті було отримано практичні навички у підготовці

існуючих компонентів до інтеграції у дане завдання,також перебираючи архітерктурні стилі, дивлячись на плюси і мінуси, і його виборі.