КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп’ютерних наук та кібернетики

Курсова робота

За спеціальністю 122 Комп’ютерні науки

на тему:

ЗВІТ ДО ЛАБОРАТОРНОЇ РОБОТИ №2

Виконав студент 2 курсу

Юрченко Антон Андрійович

Науковий керівник:

асистент

Белова Анна Сергіївна

Засвідчую, що в цій курсовій роботі

немає запозичень з працю інших авторів

без відповідних посилань

Київ – 2022

Зміст

Реферат	**3**
Скорочення та умовні позначення	**4**
Вступ	**5**
Розділ 1. Огляд використаних технологій	**7**
Розділ 2. Призначення і цілі створення системи	**8**
Розділ 3. Опис організації інформаційної бази	**11**
Розділ 4. Реалізація системи	**15**
Розділ 5. Інструкція користувачу	**18**
Висновки	**20**
Перелік використаних джерел	**21**

Реферат

Ключові слова : ASP.NET CORE, ENTITY FRAMEWORK, MICROSOFT VISUAL STUDIO 2022, .NET, ВЕБ-ЗАСТОСУНОК, ПЗ, РОЗРОБКА ПЗ, ФОРМА, ЗАПИТ, ПРЕДСТАВЛЕННЯ, CODE FIRST, DB.

Об’єктом дослідження є процес реалізації сервісу перегляду інформації про страви, інгредієнти, меню, та сезони ресторану, надання можливості робити замовлення обраних страв.

Предметом дослідження є служба управління базою даних ресторану.

Метою даної роботи є розробка служби управлінням книжковими магазинами та їх складовими, а саме книгами, працівниками, сервісами доставки.

Методи дослідження: спостереження, порівняння та аналіз.

Інструментальні розроблення: платформа .NET Core, технологія роботи з даними Entity Framework Core, СКБД SQL Management Studio, база даних SQL Server, організація архітектури серверного застосунку на основі ASP.NET Core з використанням багаторівневої архітектури, середовище розробки Microsoft Visual Studio 2022.

Скорочення та умовні позначення

API – application programming interface, прикладний програмний інтерфейс;

ASP – Active Server Pages, технологія створення веб-застосунків і веб-сервісів від компанії Microsoft;

EF Core – Entity Framework Core, об'єктно-орієнтована, легковажна технологія;

HTML – HyperText Markup Language, мова гипертекстової розмітки;

IDE – Integrated Design Environment, інтегроване середовище розробки;

LINQ – Language Integrated Query, мова запитів для платформи програмування .NET;

.NET – платформа програмування компанії Microsoft;

NuGet – система керування пакетами для платформ Microsoft;

ORM – Object-Relational Mapping, об’єктно- реляційне відображення;

SQL – Structured query language, мова структурованих запитів;

WEB – World Wide Web, всесвітня мережа;

БД – база даних;

Вступ

Актуальність. Розробка інтернет-сервісу для замовлення їжі та його використання є актуальним питанням на сьогоднішній день, оскільки мільйони людей щодня, не виходячи з дому, користуються подібними сервісами, зокрема купують їжу. В світі і особливо в Україні, величезними темпами росте кількість користувачів Інтернет і, як наслідок, кількість «електронних» замовників.

Мета дослідження - розробка інтернет-сервісу для керування базою даних та створення замовлення.

Завдання дослідження – розробка алгоритму функціонування інтернет-сурвісута програмних засобів, проектування структури інтернет-сервісу, керування базою даних ресторану, внесених в базу їх персоналом записів; поглиблення та закріплення знань з технологій ASP.NET WEB API та Entity Framework Core – Code First

Предметом дослідження є веб-сервіс, що реалізує ідею сервісу для надання можливості створювати або замовляти страви.

Методи дослідження, використані у роботі:

- аналіз;
- моделювання;
- спостереження.

Використані інструменти для досягнення остаточного результату:

- платформа .NET;
- інтегроване середовище розробки Visual Studio 2022;
- мова програмування C#;
- додаток для створення веб-застосунків ASP.NET Core;
- WPS Office для створення даного звіту.

Практична цінність отриманих результатів полягає у використанні інтернет – сервісу для комерційних цілей. Взаємозв’язок з іншими роботами. В процесі роботи були

використані та поглиблені знання з створення та зміни баз даних за допомогою Code First Workflow, а також набуті знання в роботи з WEB API мовою програмування C#

Розділ 1. Огляд використаних технологій

1.1 Entity Framework Core Code First - ORM Framework

Об'єктно-орієнтована, розширювана технологія від компанії Microsoft для доступу до даних. EF Core є ORM-інструментом (object-relational mapping – відображення даних на реальні об'єкти). Тобто EF Core дозволяє працювати базами даних, але є більш високий рівень абстракції: EF Core дозволяє абстрагуватися від самої бази даних та її таблиць та працювати з даними незалежно від типу сховища. Якщо фізично ми оперуємо таблицями, індексами, первинними та зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працюємо з об'єктами.

1.2 ASP.NET WEB API

Фреймворк для створення сайтів та веб-додатків за допомогою реалізації APIб що передбачає створення набору чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення. API може бути використано для веббазованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

1.3 Bootswatch

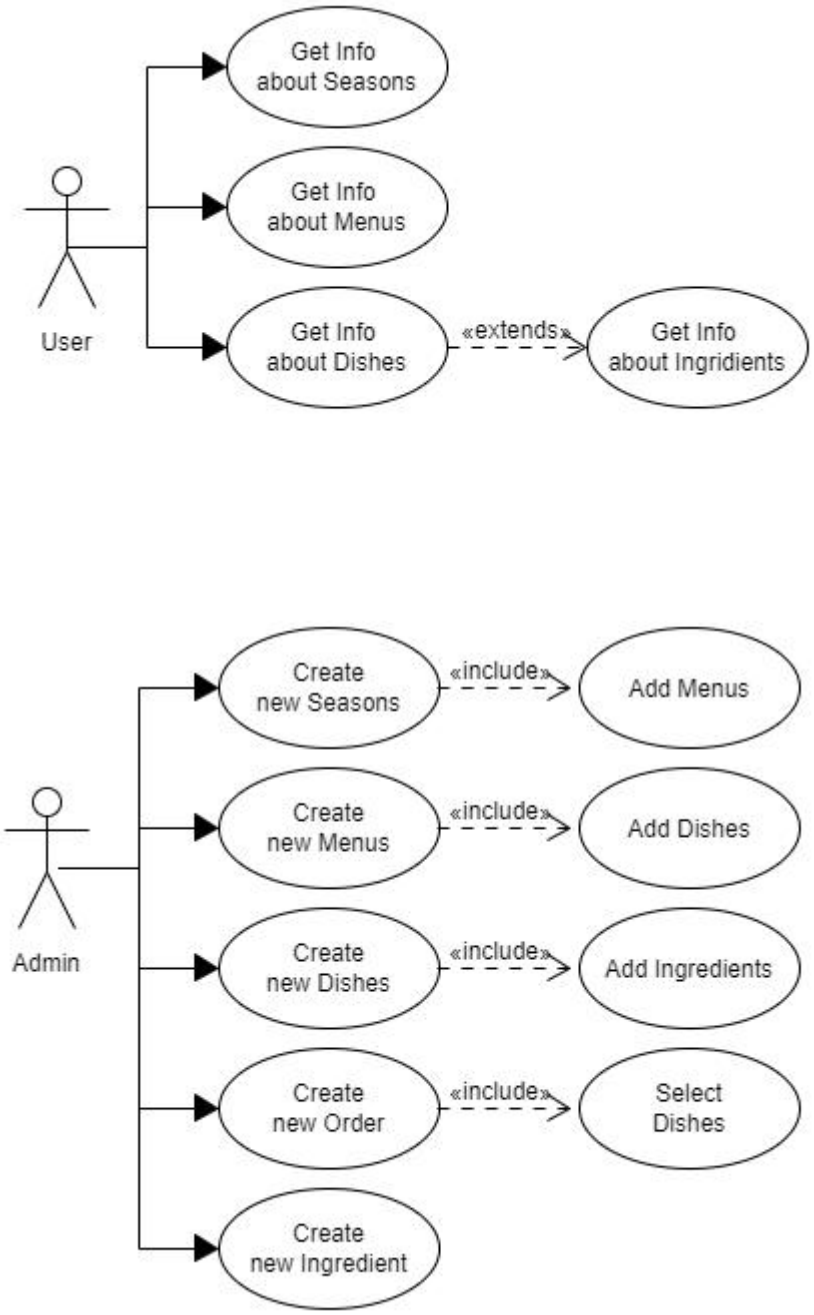
Вільний набір інструментів для створення сайтів та веб-додатків. Включає в себе HTML- й CSS-шаблони оформлення для типографії, веб-форм, кнопок, міток, блоків навігації та інших складових веб-інтерфейсу, включаючи JavaScript-розширення.

1.4 JavaScript

Потужна мова програмування, яка може додати веб-сайту інтерактивності. Його винайшов Брендан Айх (співзасновник проекту Mozilla, Mozilla Foundation і Mozilla Corporation). JavaScript є універсальним і зручним для початківців.

Розділ 2. Призначення і цілі створення системи

2.1 Призначення системи



Use-case діаграма

Призначення веб-системи «Restaurant API» є автоматизація процесу управління базою даних ресторану, що дає змогу ефективно керувати усіма задачами, від створення сезону, меню, страви або інгредієнту, до отримання замовлень та їх обробки.

Робота передбачає:

- Аналіз методів, методик і моделей, що застосовуються для розв’язання задач створення комплексних вебсистем
- Проектування та програмну реалізацію системи «Restaurant API»

2.2 Цілі створення системи

«Restaurant API» створюється з метою надання користувачам системи управління базою даних ресторану, організаціїї її в єдину систему, задля поліпшення та полегшення робочого процесу.

2.3 Вимоги до системи

Система повинна будуватись з використанням підходів ООП, з уніфікацією програмно-технічних засобів розробки прикладної функціональності з використанням сучасних технологій. А також система має бути реалізована з використанням технології API.

2.3.1 Вимоги до системи в цілому

Система «Restaurant API» повинна реалізовувати функції створення, редагування та видалення сезонів, меню, страв, інгредієнтів та замовлень.

В системі передбачається виділити наступні функціональні підсистеми:

- адміністративна, призначена для керування базою даних, зміною записів про страви, інгредієнти, меню, сезони та перегляд і обробка замовлень
- підсистема користувача, призначена для перегляду страви, інгредієнти, меню, сезони та створення замовлень

2.3.2 Вимоги до функції, які виконуються системою

Підсистема адміністратора

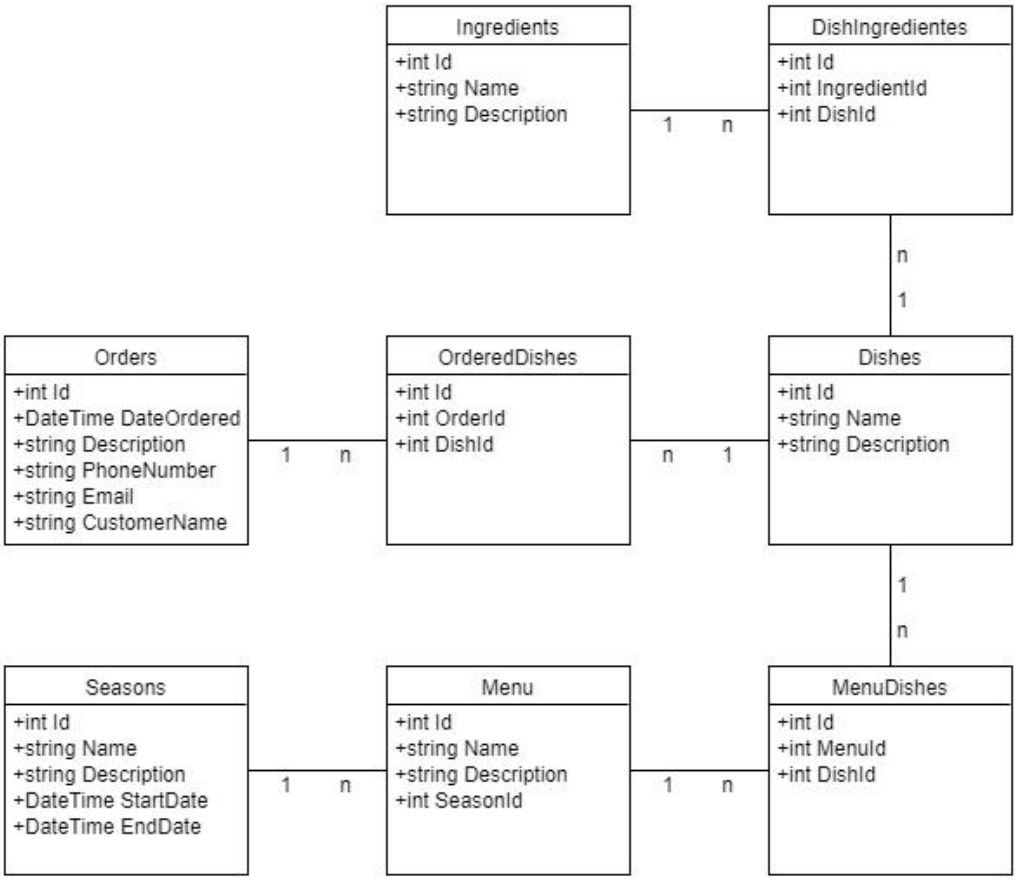
Функція	Задача
Робота з базою даних ресторану	Перегляд, створення, оновлення та видалення даних про страви, інгредієнти, меню та сезони
Робота з замовленнями	Перегляд та обробка замовлень

Підсистема користувача

Функція	Задача
Робота з базою даних ресторану	Перегляд, даних про страви, інгредієнти, меню та сезони
Робота з замовленнями	Створення замовлень

Розділ 3. Опис організації інформаційної бази

3.1 Логічна структура бази даних



Діаграма бази даних.

Перелік таблиць системи «Restaurant API»

Номер	Таблиця	Опис
1	Orders	Таблиця для збереження інформації про замовлення користувачів
2	OrderedDishes	Таблиця для збереження інформації про зв’язки страв і замовлень
3	Dishes	Таблиця для збереження інформації про страви
4	MenuDishes	Таблиця для збереження інформації про зв’язки страв і меню

5	Menu	Таблиця для збереження інформації про меню
6	Seasons	Таблиця для збереження інформації про сезони
7	DishIngredients	Таблиця для збереження інформації про зв’язки інгредієнтів і страв
8	Ingredients	Таблиця для збереження інформації про інгредієнти

3.2 Опис таблиць

Таблиця Orders

Атрибут	Тип	Опис
Id	int	Ідентифікатор замовлення
DateOrdered	datetime	Час замовлення
Description	Nvarchar(250)	Коментар до замовлення
PhoneNumber	Nvarchar(50)	Номер телефону замовника
Email	Nvarchar(50)	Адреса електронної пошти замовника
CustomerName	Nvarchar(50)	Ім’я замовника

Таблиця OrderedDishes

Атрибут	Тип	Опис
Id	int	Ідентифікатор зв’язку
OrderId	int	Ідентифікатор замовлення
DishId	int	Ідентифікатор страви

Таблиця Dishes

Атрибут	Тип	Опис
Id	int	Ідентифікатор страви
Name	Nvarchar(50)	Ім’я страви
Description	Nvarchar(225)	Опис страви

Таблиця MenuDishes

Атрибут	Тип	Опис
Id	int	Ідентифікатор зв’язку
MenuId	int	Ідентифікатор меню
DishId	int	Ідентифікатор страви

Таблиця Menu

Атрибут	Тип	Опис
Id	int	Ідентифікатор меню
Name	Nvarchar(50)	Назва меню
Description	Nvarchar(225)	Опис меню
SeasonId	int	Ідентифікатор сезону

Таблиця Season

Атрибут	Тип	Опис
Id	int	Ідентифікатор сезону
Name	Nvarchar(50)	Назва сезону
Description	Nvarchar(225)	Опис меню
StartDate	datetime	Дата початку сезону
EndDate	datetime	Дата завершення сезону

Таблиця DishIngredients

Атрибут	Тип	Опис
Id	int	Ідентифікатор зв’язку
IngredientId	int	Ідентифікатор інгредієнту
DishId	int	Ідентифікатор страви

Таблиця Ingredients

Атрибут	Тип	Опис
Id	int	Ідентифікатор інгредієнту
Name	Nvarchar(50)	Ім’я інгредієнту
Description	Nvarchar(225)	Опис інгредієнту

Розділ 4. Реалізація системи

4.1.Бази даних

Весь процес кодування виконувався в середовищі програмування Visual Studio 2022 на мові програмування C# із застосуванням технології Entity Framework Core та ASP.NET API протоколу для сайтів та веб-додатків. Був застосований підхід Code First.

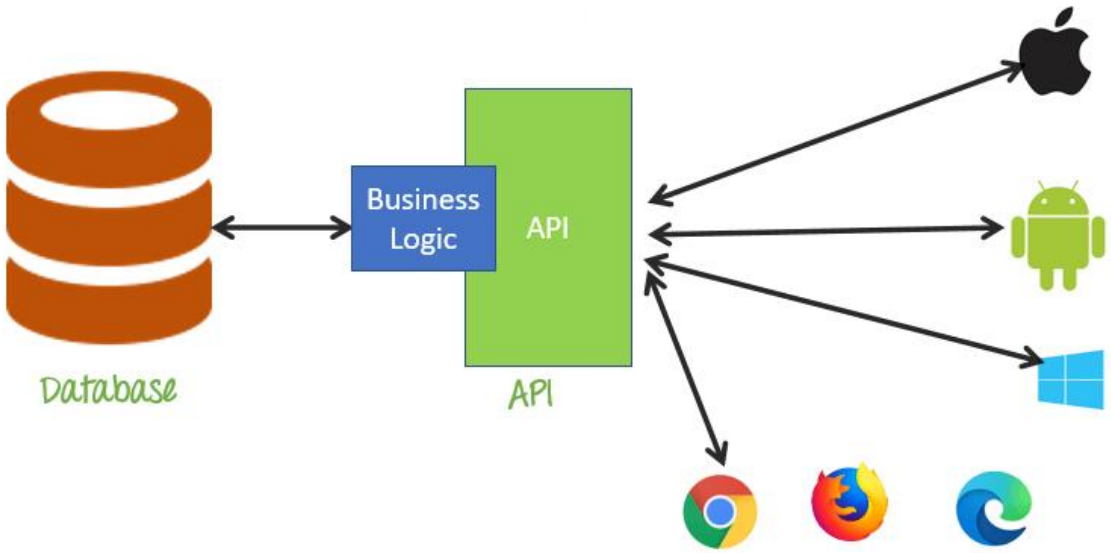


Схема API

Для того щоб використовувати БД у програмі, необхадно під’єднатися до SQL-сервера, вказавши ConnectionString, і створити необхадні моделі та контекст з цими моделями.

4.2 Створення моделей

За замовчуванням всі типи сутностей, для яких визначені в контексті даних набори DbSet, включаються в модель і надалі зіставляються з таблицями в базі даних. Але крім того, в модель також включаються типи, на які є посилання в сутності, які вже включені в модель, наприклад, через властивості DbSet.

За замовчуванням кожна сутність зіставляється з таблицею, яка називається за іменем властивості DbSet <T> в контексті даних, що представляє дану сутність. Якщо в контексті даних подібного властивості не визначено, то для назви таблиці використовується ім'я класу сутності.

```
using IEAPTLab2.Validation;
using System.ComponentModel.DataAnnotations;

namespace IEAPTLab2.Models
{
    5 references
    public class Season
    {
        3 references
        public int Id { get; set; }

        [Required(ErrorMessage = "Сезон обов'язково повинен мати назву")]
        [MinLength(3)]
        [MaxLength(50)]
        [Display(Name = "Назва сезону")]
        0 references
        public string Name { get; set; }

        [StartDateLessThanEndDate]
        [Required(ErrorMessage = "Сезон обов'язково повинен мати дату початку")]
        [Display(Name = "Дата початку сезону")]
        [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}")]
        1 reference
        public DateTime StartDate { get; set; }

        [StartDateLessThanEndDate]
        [Required(ErrorMessage = "Сезон обов'язково повинен мати дату кінця")]
        [Display(Name = "Дата кінця сезону")]
        [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}")]
        1 reference
        public DateTime EndDate { get; set; }

        [MaxLength(225)]
        [Display(Name = "Опис сезону")]
        0 references
        public string? Description { get; set; }
    }
}
```

Модель для зберігання сезону

```
using Microsoft.EntityFrameworkCore;

namespace IEAPTLab2.Models
{
    – references
    public class RestaurantAPIContext : DbContext
    {
        – references
        public virtual DbSet<Dish> Dishes { get; set; }
        – references
        public virtual DbSet<Ingredient> Ingredients { get; set; }
        – references
        public virtual DbSet<Menu> Menus { get; set; }
        – references
        public virtual DbSet<Season> Seasons { get; set; }
        – references
        public virtual DbSet<Order> Orders { get; set; }
        0 references
        public RestaurantAPIContext(DbContextOptions<RestaurantAPIContext> options):base(options)
        {
            Database.EnsureCreated();
        }
    }
}
```

Контекст

4.3 Створення контролера

До контролера застосовується атрибут [ApiController], який дозволяє використовувати низку додаткових можливостей, зокрема, в плані прив'язки моделі. Також до контролера застосовується

атрибут маршрутизації, який вказує, як контролер буде зіставлятися з запитами.

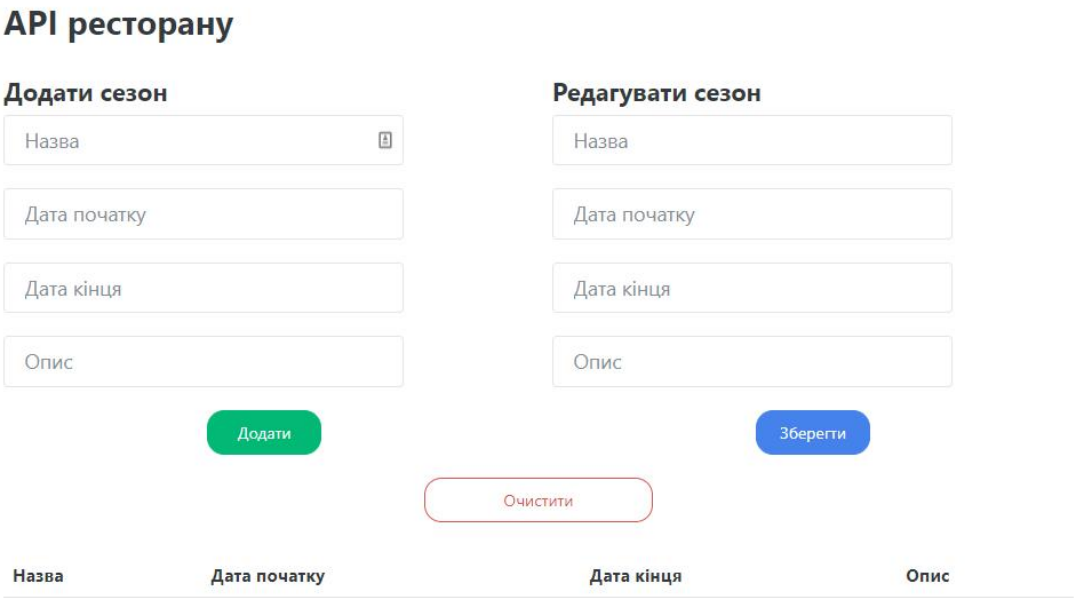
Контролер API призначений переважно для обробки запитів протоколу HTTP: Get, Post, Put, Delete, Patch, Head, Options. В даному випадку для кожного типу запитів в контролері визначено свої методи. Так, метод Get () обробляє запити типу GET і повертає колекцію об'єктів з БД.

```
[Route("api/[controller]")]
[ApiController]
1 reference
public class SeasonsController : ControllerBase
{
    private readonly RestaurantAPIContext _context;

    0 references
    public SeasonsController(RestaurantAPIContext context)
    {
        _context = context;
    }

    [HttpGet]
    0 references
    public async Task<IActionResult> Index()
    {
        var items = _context.Seasons;
        return Ok(items);
    }
    [HttpPost]
    0 references
    public async Task<IActionResult> Create([Bind("Name,StartDate, EndDate, Description")] Season Season)
    {
        if (ModelState.IsValid)
        {
            await _context.AddAsync(Season);
            await _context.SaveChangesAsync();
            return Ok(Season);
        }
        return BadRequest(Season);
    }
    [HttpPut]
    0 references
    public async Task<IActionResult> Edit([Bind("Id, Name,StartDate, EndDate, Description")] Season Season)
    {
        if (ModelState.IsValid)
        {
            if (SeasonExists(Season.Id))
            {
                _context.Update(Season);
                await _context.SaveChangesAsync();
                return Ok(Season);
            }
            return NotFound();
        }
        return BadRequest(Season);
    }
    [HttpDelete]
    0 references
    public async Task<IActionResult> Delete( int id)
    {
        Console.WriteLine(id);
        if (!SeasonExists(id))
        {
            return NotFound();
        }
        var Season = await _context.Seasons.FirstAsync(m => m.Id == id);
        _context.Seasons.Remove(Season);
        await _context.SaveChangesAsync();
        return Ok(Season);
    }
```

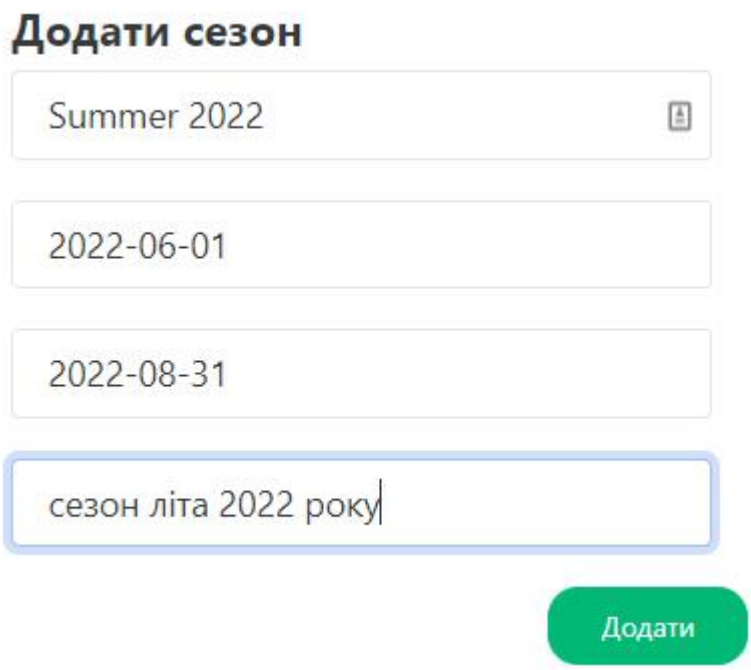
Контролер для обробки запитів до таблиці сезонів

Розділ 5. Інструкція користувачу

При вході на сайт, користувача зістручає сторінка для перегляду, додавання, редагуваня та видалення сезону.

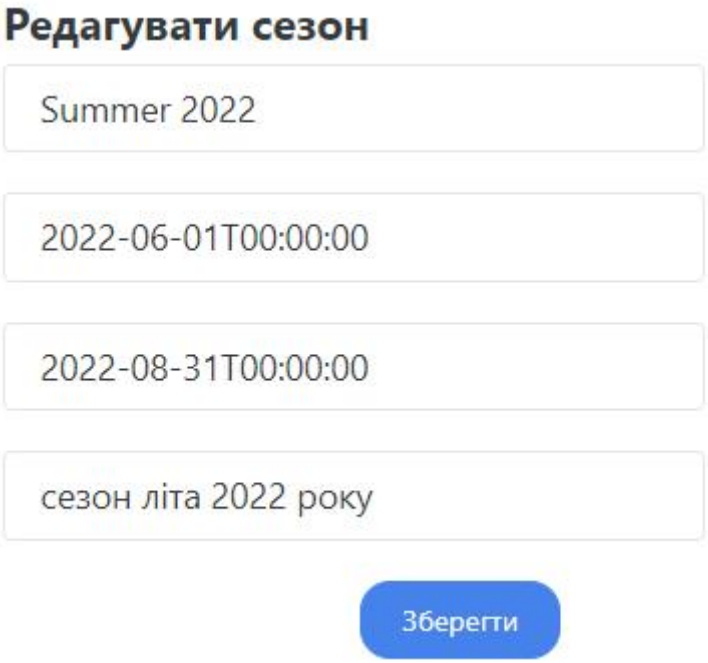


Для додавання дового сезону необхідно внести необхідні дані у форму “Додати сезон” на написнути кнопку “Додати”.



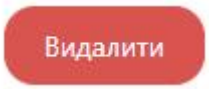
Після додавання нового сезону, у таблиці знизу буде наведено доданий запис

Для редагування записів необхідно натиснути кнопку “Редагувати” навпроти того запису, який користувач хоче редагувати. Тоді права форма, “Редагувати сезон” наповниться даними вибраного сезону, які можна буде редагувати. Для зберігання нових даних необхідно натиснути кнопку зберегти.



Для того щоб очистити форму після редагування необхідно натиснути кнопку “очистити”.

Для видалення записів необхідно натиснути кнопку “Видалити” навпроти того запиту, який користувач хоче видалити.



Висновки

Розробив систему, що надає користувачу можливість керування базою даних ресторану, зокрема переглядати, видаляти, створювати та редагувати сезони.

У процесі роботи над розробкою веб-додатку навчився новій технології з розробки веб-додатків, поглибив знання з HTML та CSS, а також ознайомився з мовою програмування JavaScript.

В результаті роботи над проектом було окремо розроблено backend та frontend частину, що в майбутньому дозволить реалізувати повноцінний програмний продукт «Restaurant API», що буде відповідати усім технічним вимогам. Цей досвід дуже важливий для підвищення технічних навичок, покращення знань у сфері інформаційних технологій та оформленні необхідної документації до проекту.

Перелік використаних джерел

1. Bootswatch [Електронний ресурс] – Режим доступу до ресурсу: https://bootswatch.com/
2. ASP.NET MVC documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en us/aspnet/mvc/overview/getting-started/introduction/getting-started/
3. Entity Framework documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/ef/
4. JavaScript documentation [Електронний ресурс] – Режим доступу до ресурсу: https://devdocs.io/javascript/