math_2024_056

Міністерство освіти і науки України

Чернівецький національний університет

імені Юрія Федьковича

Факультет математики та інформатики
(повна назва інституту/факультету)

Кафедра математичного моделювання
(повна назва кафедри)

"Розробка веб-сайту для бронювання спортзалів з використанням ASP.NET"

Кваліфікаційна робота

Рівень вищої освіти - перший (бакалаврський)

Виконав:
студент 4 курсу, 407 групи
Цюпа Олег Сергійович
Керівник:
асистент, канд. тех. наук
Шкільнюк Д.В.

До захисту допущено
на засіданні кафедри
математичного моделювання
протокол № _ від _____ 2023р.
Зав. кафедри _____ проф. Черевко І.М.

Чернівці – 2024

Анотація

У кваліфікаційній роботі описані етапи створення веб-сайту для бронювання спортивних залів, зокрема з організації спортивних ігор (волейбол, футбол та інші), за допомогою фреймворку ASP.NET MVC та мови програмування C#. У роботі розглядається інтеграція з Google Maps API для зручного відображення місцезнаходження залів, система відгуків через Telegram, додавання фотографій спортивних залів, відображення цін за кожен вид спорту, можливість ділитися інформацією через Telegram та інтеграція бронювання з календарем користувача за допомогою Google Calendar API.

Ключові слова: ASP.NET MVC, C#, Google Maps API, Google Calendar API, Telegram, бронювання, спортивні зали, база даних.

Annotation

This qualification work describes the stages of creating a website for booking sports halls, including the organization of sports games (volleyball, football, etc.), using the ASP.NET MVC framework and C# programming language. The work discusses integration with Google Maps API for convenient location display of halls, a feedback system via Telegram, adding photos of sports halls, displaying prices for each type of sport, the ability to share information via Telegram, and booking integration with the user's calendar using Google Calendar API.

Key words: ASP.NET MVC, C#, Google Maps API, Google Calendar API, Telegram, booking, sports halls, database.

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів наукових досліджень інших авторів мають посилання на відповідне джерело.

_____ Цюпа О.С.

(підпис)

Зміст

Вступ

У сучасному світі спорт стає невід'ємною частиною повсякденного життя, сприяючи не лише підтриманню фізичної форми, а й виконуючи важливу соціальну та психологічну функцію. Зростаючий інтерес до здорового способу життя створює потребу в інноваційних рішеннях, які б дозволяли легко та ефективно організовувати спортивні заходи.

Однією з головних проблем є відсутність зручних і ефективних систем для бронювання спортивних залів. Це стає серйозною перешкодою на шляху до регулярних занять спортом. У світі, де цифрові технології розвиваються стрімкими темпами, а попит на фітнес-послуги зростає, створення зручної та багатофункціональної системи бронювання спортивних залів є надзвичайно актуальним. Така система значно спростить процес пошуку та бронювання залів, зробивши його прозорим і доступним для широкого кола користувачів, сприяючи підвищенню загальної фізичної активності населення та покращенню якості життя.

Метою цього проекту є розробка веб-додатку "GymRent", який дозволить користувачам ефективно шукати та бронювати спортзали, адаптовані до їхніх індивідуальних потреб. Веб-додаток інтегруватиме різні функціональні можливості, включно з пошуком за географічним розташуванням, типом спорту, вартістю оренди та іншими параметрами. Основні завдання проекту включають:

- дослідження та аналіз ринку бронювання спортивних об'єктів для виявлення поточних трендів і недоліків;
- розроблення інтуїтивно зрозумілого користувацького інтерфейсу для полегшення процесу вибору і бронювання;
- впровадження системи управління базами даних для зберігання інформації про користувачів, спортзали та транзакції бронювання;
- тестування системи для гарантії її надійності та ефективності.

Таким чином, реалізація цього проекту не лише полегшить доступ до спортивних залів і підвищить якість життя користувачів, а й надасть власникам спортзалів можливість ефективно керувати своїми ресурсами та отримувати прибуток

Розділ 1. Теоретична частина

1.1 Важливість системи бронювання спортзалів

У сучасному світі спорт набуває все більшого значення, стаючи невід'ємною частиною життя багатьох людей. Однак, організація спортивних занять часто ускладнюється через відсутність зручних систем бронювання. Традиційні методи запису через телефонні дзвінки або особисті візити в спортивні заклади займають багато часу і не гарантують актуальність інформації про доступність залів. Відсутність єдиного ресурсу для бронювання спортивних залів створює значні перешкоди для людей, які прагнуть регулярно займатися спортом.

1.1.1 Вплив на здоров'я та соціальну активність

Спорт відіграє важливу роль у підтриманні фізичного здоров'я, зниженні стресу та покращенні соціальної активності. Доступ до зручної системи бронювання спортивних залів сприяє збільшенню кількості людей, що займаються спортом, що, в свою чергу, покращує загальний рівень здоров'я населення. Крім того, зручні та доступні платформи для бронювання сприяють соціалізації, оскільки люди можуть легше організовувати групові заняття та спортивні події.

1.2 Опис використаних технологій

У розробці сучасних веб-додатків вибір правильних технологій відіграє ключову роль. У даному проекті основою для створення системи бронювання спортзалів виступають ASP.NET MVC і C# .NET. Ці технології були обрані за їхню продуктивність, безпеку і широкі можливості зі створення масштабованих і надійних веб-додатків.

1.2.1 ASP.NET MVC

ASP.NET MVC - це потужний фреймворк для створення веб-додатків і веб-сайтів на базі .NET Framework, розроблений компанією Microsoft. Модель

MVC (Model-View-Controller) забезпечує чіткий поділ даних (модель), призначеного для користувача інтерфейсу (представлення) і бізнес-логіки (контролер), що підвищує керованість коду та спрощує тестування й підтримку програми.

Переваги використання ASP.NET MVC включають:

- тестованість: завдяки поділу логіки та інтерфейсу, розробники можуть більш ефективно організовувати тестування компонентів;
- гнучкість і контроль: фреймворк надає розробникам більшу контрольність над поведінкою додатка, на відміну від традиційного ASP.NET, де багато процесів автоматизовані і приховані від розробника;
- підтримка і розвиток: ASP.NET MVC активно підтримується і розвивається Microsoft, що гарантує доступність нових функцій і оновлень безпеки;

1.2.2 Мова програмування C#

C# - це об'єктно-орієнтована мова програмування, яка використовується для розробки серверної частини веб-додатків на платформі .NET. Цю мову обрано для проекту через її сувору типізацію, потужні можливості з опрацювання винятків, а також підтримку сучасних програмних патернів і принципів.

Безпека додатків, написаних на C#, забезпечується за рахунок:

- керованого коду: .NET Framework виконує код у керованому середовищі, що знижує ризики, пов'язані з витоками пам'яті та іншими низькорівневими вразливостями;
- вбудовані засоби безпеки: C# підтримує безліч функцій безпеки, як-от перевірка типів, автоматичне керування пам'яттю та виняткові ситуації, що допомагає запобігти безлічі поширених помилок програмування;
- широкі можливості для захисту даних: фреймворк .NET пропонує широкий набір інструментів для шифрування даних і безпечної роботи з

мережею, що критично важливо для веб-застосунків, які працюють із конфіденційною інформацією.

1.2.3 Razor Pages

Razor Pages є важливою частиною ASP.NET Core, надаючи розробникам простий і інтуїтивно зрозумілий підхід до створення веб-інтерфейсів. Ця технологія об'єднує логіку обробки та відображення даних в одному файлі, що значно полегшує підтримку та розширення функціональності [1].

Основні переваги:

- зручна структура: концентрація всіх елементів, пов'язаних з конкретною сторінкою, в одному місці сприяє кращій організації коду і спрощує його супровід.
- висока продуктивність: завдяки оптимізованому процесу обробки запитів, Razor Pages забезпечують швидке завантаження та обробку даних, що покращує загальну продуктивність веб-застосунку.
- простота у використанні: для розробників, які тільки починають працювати з ASP.NET Core, Razor Pages пропонують легкий для засвоєння підхід з чіткою структурою, що прискорює процес навчання.
- широкі можливості інтеграції: Razor Pages без проблем інтегруються з такими інструментами, як Entity Framework для роботи з базами даних та SignalR для реалізації функціоналу в реальному часі.

Використання Razor Pages дозволяє створювати адаптивні та інтерактивні веб-сторінки, що забезпечують інтуїтивно зрозумілий інтерфейс для користувачів. Це значно підвищує загальний користувацький досвід завдяки зручності та функціональності.

1.2.4 База даних SQL Express

Microsoft SQL Server Express є безкоштовною версією системи управління реляційними базами даних Microsoft SQL Server, призначеною для завантаження, розповсюдження та використання. Вона включає в себе базу

даних, спеціально призначену для вбудованих та меншомасштабних застосунків. Історія цього продукту сягає коріннями до продукту Microsoft Database Engine (MSDE), який був вперше представлений разом із випуском SQL Server 2000. Брендування "Express" використовується з моменту випуску SQL Server 2005 [2].

Microsoft SQL Server Express LocalDB є версією Microsoft SQL Server Express, керованим екземпляром движка SQL Server, який використовується за запитом. Він призначений для розробників і має наступні обмеження: розмір бази даних до 10 ГБ та лише локальні підключення (мережеві підключення не підтримуються).

Microsoft SQL Server Express і Microsoft SQL Server Express LocalDB надають розробникам можливість ефективно працювати з базами даних, не потребуючи великих витрат на ліцензії або складні налаштування. Вони є важливими інструментами для створення, розгортання та тестування програмних продуктів на базі баз даних Microsoft SQL Server. Завдяки їх простоті використання та обмеженим вимогам до ресурсів, вони стають популярними серед розробників, що працюють над проектами різних масштабів та складності.

1.2.5 Entity Framework

Entity Framework (EF) є потужним фреймворком для розробки баз даних, який відкриває перед розробниками можливості об'єктно-реляційного відображення (ORM) у середовищі ADO.NET. Починаючи свій шлях як невід'ємна частина .NET Framework, починаючи з версії 6.0, Entity Framework став окремим компонентом від .NET Framework, що дозволяє забезпечити більшу гнучкість у розробці та розгортанні програмних продуктів [3].

Останньою версією класичного фреймворку є Entity Framework 6.4. Незважаючи на те, що Entity Framework 6 продовжує підтримуватися, він більше не отримує нових функцій, а лише виправлення для проблем безпеки.

Це спонукало розробників до переходу на більш сучасні та продуктивні інструменти.

У 2016 році було представлено нове покоління фреймворку під назвою Entity Framework Core (EF Core). Ця версія відзначається значними поліпшеннями у порівнянні з попереднім варіантом, але не зберегла повну сумісність з ним. Почавши з версії 1.0, EF Core активно розвивається, і наразі останньою є версія 8.0, що пропонує розробникам ще більше можливостей для ефективної роботи з базами даних.

1.2.6 ASP .NET Identity

ASP.NET Core Identity - це потужний API, який надає можливості для управління автентифікацією та авторизацією користувачів у веб-додатках. Основні можливості включають [4]:

- управління користувачами: Identity дозволяє керувати реєстрацією, автентифікацією та авторизацією користувачів. Вона забезпечує можливість створення облікових записів, зберігання профільних даних, а також управління ролями та претензіями користувачів;
- підтримка зовнішніх постачальників входу: користувачі можуть створювати облікові записи або входити за допомогою різних зовнішніх постачальників, таких як Facebook, Google, Microsoft Account та Twitter;
- можливість автентифікації всіх користувачів: Identity надає можливість глобально вимагати автентифікації для всіх користувачів у додатку;
- легке налаштування та розгортання: завдяки можливостям налаштування та простоті розгортання, Identity дозволяє швидко і ефективно використовувати її у додатку;

Для розробників, що працюють з Identity, доступний вихідний код на GitHub, що дозволяє переглянути та налаштувати функціональність фреймворку. Також існує можливість використання різних сховищ для зберігання даних користувачів, таких як база даних SQL Server або Azure Table Storage.

Варто зазначити, що ASP.NET Core Identity є окремим від Microsoft Identity Platform, яка є еволюцією платформи розробника Azure Active Directory (Azure AD) і надає альтернативні можливості для автентифікації та авторизації в ASP.NET Core додатках.

1.2.7 Google Api Maps & Calendar

Google Maps API та Google Calendar API - це набори інструментів, які надають розробникам можливість інтегрувати картографічні функції та календарні події у свої веб-сайти та додатки.

Google Maps API дозволяє розробникам вбудовувати інтерактивні карти Google Maps у свої веб-сайти, що дозволяє користувачам переглядати карти, знаходити місцезнаходження, отримувати напрямки та багато іншого. Він забезпечує доступ до різноманітних функцій, таких як створення маршрутів, визначення місця на карті, відображення трафіку та інше.

Google Calendar API надає користувачам доступ до календарних подій та інформації про них, яка зберігається у Google Календарі. За допомогою цього API можна створювати, оновлювати та видаляти події у календарях користувачів, отримувати інформацію про події та їх деталі.

Інтеграція Google Maps API та Google Calendar API дозволяє створювати розширені функціональності для веб-сайтів та додатків, такі як реєстрація на події з Google Календаря, відображення маршрутів та подій на картах та багато іншого. Ці API відкривають широкі можливості для створення корисних та привабливих для користувачів функціональностей.

1.2.8 Telegram Discussion Widget

Discussion Widget створений для того, щоб вбудовувати обговорення з будь-якого публічного каналу безпосередньо на веб-сайт. Це зручне рішення для власників сайтів, які хочуть спростити спілкування зі своєю аудиторією. Необхідно лише мати посилання на повідомлення з коментарями, щоб вбудувати їх разом із усім обговоренням [5].

Цей віджет надає можливість показувати обговорення з телеграм-каналу безпосередньо на сторінках веб-сайту. Немає потреби в жодних додаткових налаштуваннях, адже обговорення будуть автоматично доступні на веб-сайті, одразу після публікації посилання в каналі.

1.3 Принципи проектування веб-додатків.

У сучасній веб-розробці принципи та методології проектування відіграють вирішальну роль у створенні надійних, масштабованих і легко підтримуваних додатків. Проект "GymRent" ґрунтується на моделі MVC (Model-View-Controller), яка є одним з основних шаблонів проектування для веб-додатків. Цей розділ докладно описує використання MVC у проекті, а також інші ключові аспекти, які було враховано під час проектування системи.

1.3.1 MVC патерн: модель, подання, контролер

MVC патерн розділяє додаток на три основні компоненти. Розглянемо кожен з них.

Компонент модель (Model) представляє структуру даних, бізнес-правила, логіку і функції. У проекті "GymRent", моделі використовуються для представлення даних про спортзали, користувачів і бронювання, а також для взаємодії з базою даних через Entity Framework.

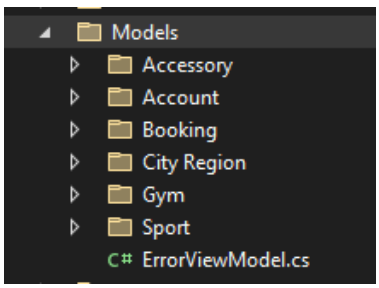


Рисунок 1.1 – Структура пакету Models

Подання (View) відповідає за відображення даних, отриманих від контролерів, користувачеві. У нашому застосунку використовуються різноманітні подання для відображення інформації про доступні спортзали, форм бронювання та профілів користувачів.

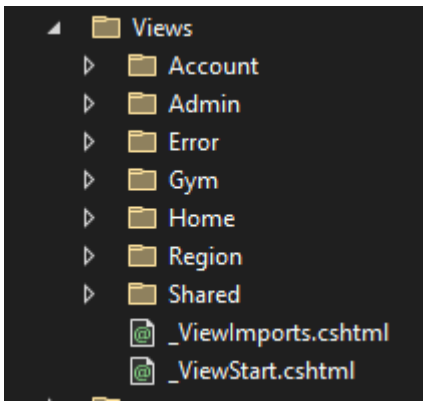


Рисунок 1.2 – Структура пакету Views

Контролер (Controller): забезпечує зв'язок між моделлю та поданням, обробляє користувацькі запити користувача, передаючи дані між моделлю та поданням. Контролери в "GymRent" керують логікою бронювання, реєстрації користувачів і обробки запитів на пошук спортзалів.

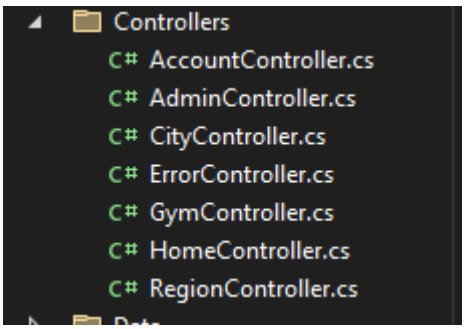


Рисунок 1.3 – Структура пакету Controllers

1.3.2 Безпека та оптимізація веб-додатків

Безпека є одним із найважливіших аспектів веб-додатків. У "GymRent" застосовуються такі заходи для забезпечення безпеки:

- автентифікація та авторизація: система використовує ASP.NET Identity для управління автентифікацією користувачів, що дає змогу забезпечити

безпечний доступ до особистої інформації та функцій управління бронюваннями;

- захист від XSS і CSRF атак: застосовуються вбудовані засоби ASP.NET, як-от антиформжері-токени, для захисту від міжсайтового скриптингу та підробки міжсайтових запитів.

Оптимізація продуктивності веб-додатка досягається за рахунок:

- кешування: використання кешування даних, часто запитуваних користувачами, знижує навантаження на сервер і прискорює завантаження сторінок;
- асинхронне програмування: асинхронні контролери в ASP.NET MVC дають змогу обробляти велику кількість запитів без блокування сервера, покращуючи загальну продуктивність системи.

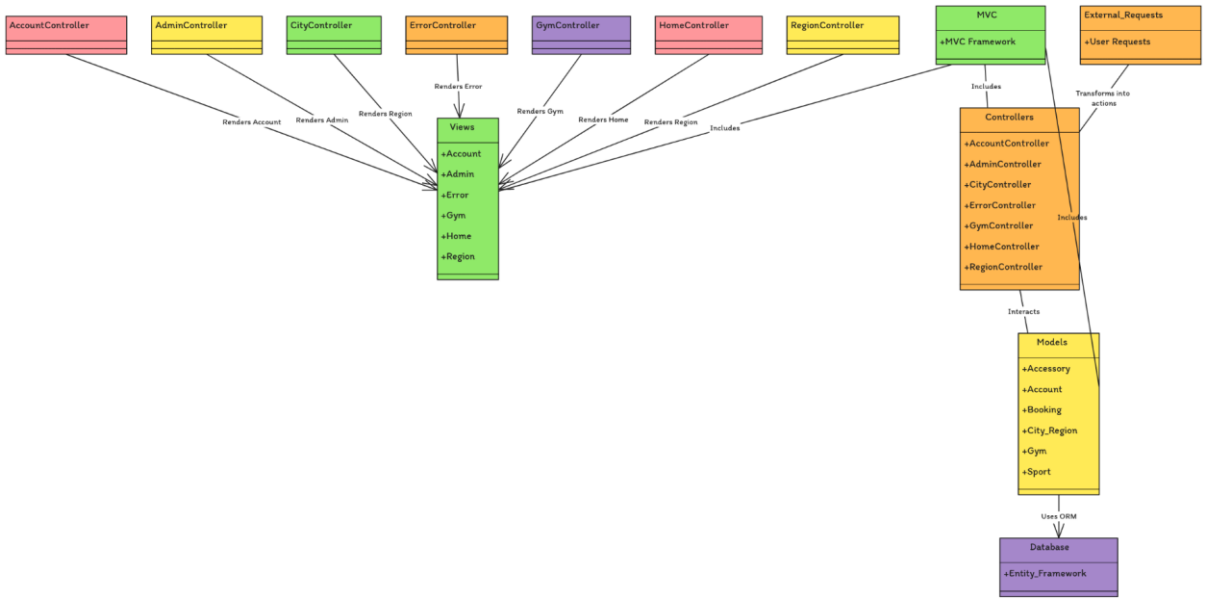


Рисунок 1.3 – Схема архітектури проекту

Розділ 2. Практична реалізація

2.1 Архітектура проекту.

Розробка веб-додатку для бронювання спортзалів "GymRent" заснована на чіткій багаторівневій архітектурі, використовуючи MVC патерн для ефективного розділення логіки даних, інтерфейсу користувача та керування даними. Цей підхід сприяє більшій модульності, легшій підтримці та масштабуванню проекту.

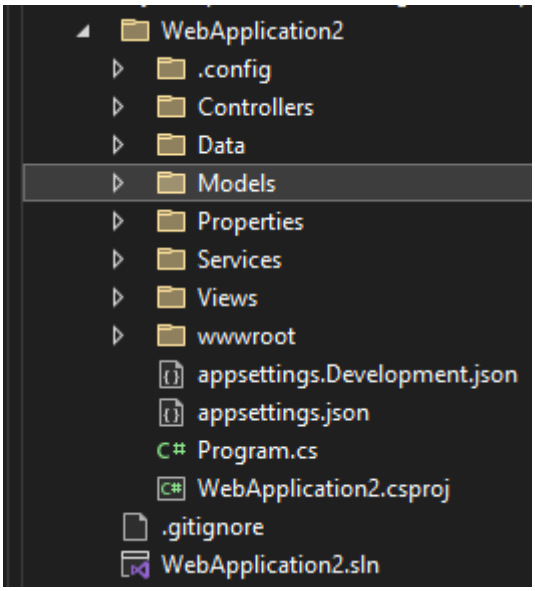


Рисунок 2.1 - Структура папок і основні компоненти проекту

2.1.1 Основні компоненти проекту

Проект організовано в кілька основних папок, кожна з яких виконує свою роль (рисунок 2.1). Розглянемо їх детальніше.

Пакет models (рисунок 2.2) містить класи, які представляють дані програми та логіку бізнес-правил. Включає в себе моделі, такі як Accessory, Account, Booking, CityRegion, Gym, і Sport.

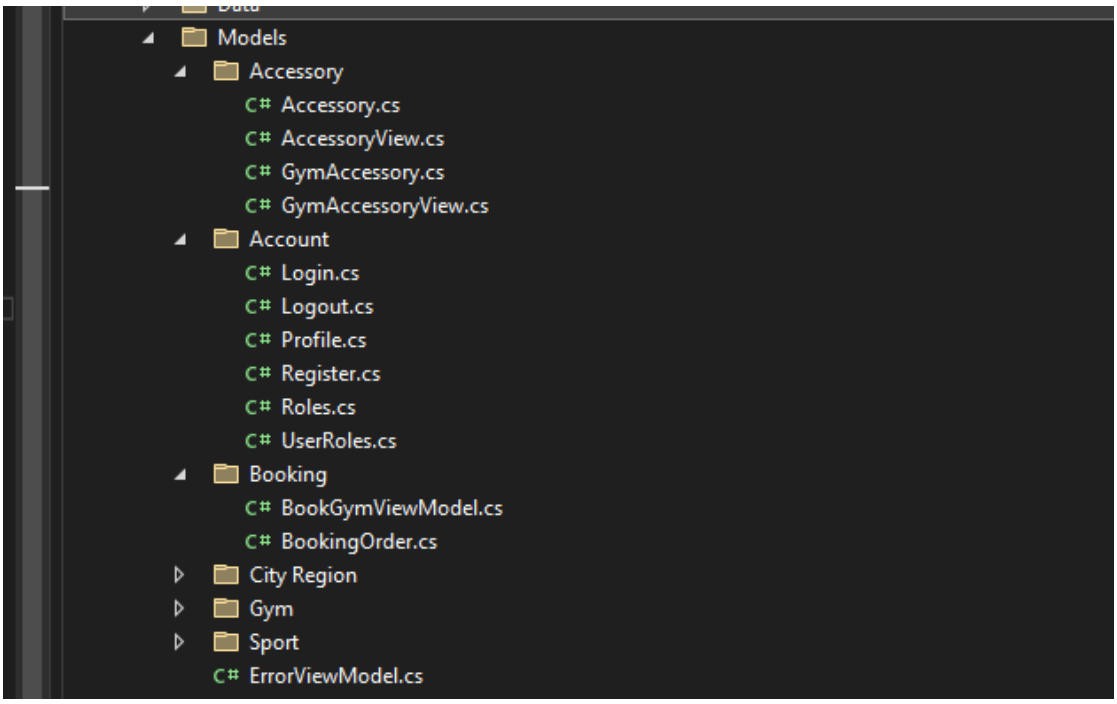


Рисунок 2.2 – Моделі

Accessory.cs, Gym.cs, BookingOrder.cs - представляють основні бізнес-об'єкти системи і включають в себе бізнес-логіку для валідації та обробки даних. Кожен клас моделі має пряме відображення в базу даних через Entity Framework, що забезпечує інтеграцію з базою даних і автоматизацію багатьох процесів управління даними.

Папка views містить файли подань, які відповідають за відображення даних користувачеві. Організована за підпапками, що відповідають контролерам, Account, Admin, Home. GymDetails.cshtml та BookGym.cshtml відповідають за представлення інтерфейсу користувача. Кожен файл View використовує Razor синтаксис для динамічного генерування HTML відповідно до даних, отриманих від контролерів.

Контролери (controllers) є ключовими компонентами, які обробляють вхідні запити, взаємодіють із моделями для оброблення даних і надсилають ці дані в подання для відображення.

Папка data містить усе, що пов'язано з доступом до даних. AppDbContext.cs — центральний компонент для керування з'єднаннями з

базою даних і конфігурацією Entity Framework. Клас контексту визначає, які моделі будуть використовуватися для генерації таблиць бази даних.

Сервіси (services) надають абстракцію бізнес-логіки і можуть включати класи, які допомагають контролерам у виконанні їхніх завдань. Це можуть бути сервіси для обробки складних запитів або виконання операцій, які не повинні бути безпосередньо вбудовані в контролери. GoogleCalendarService.cs — сервіс, що інтегрує систему з Google Calendar для управління подіями в календарях користувачів, надаючи зручний інтерфейс для відстеження та нагадувань про майбутні сесії в спортзалах.

2.1.2 Ключові файли у функціонуванні системи

Кожна директорія в проекті має чітко визначені обов'язки:

- config: містить конфігураційні файли, які налаштовують поведінку застосунку в різних середовищах, як-от розробка, тестування та продакшн;
- wwwroot: зберігає статичні файли, як-от CSS, JavaScript і зображення, які використовуються в поданнях;
- properties: включає файли, які містять метадані про проект, як-от версія та інші властивості, необхідні для складання програми;
- appsettings.json: містить параметри конфігурації, такі як рядки підключення до бази даних та інші важливі налаштування, які можуть бути змінені без необхідності перекомпіляції коду.

Ця структурована організація проекту (рисунок 2.3) покращує керованість і підтримку коду, даючи змогу розробникам і новим учасникам проекту швидше орієнтуватися в його структурі.

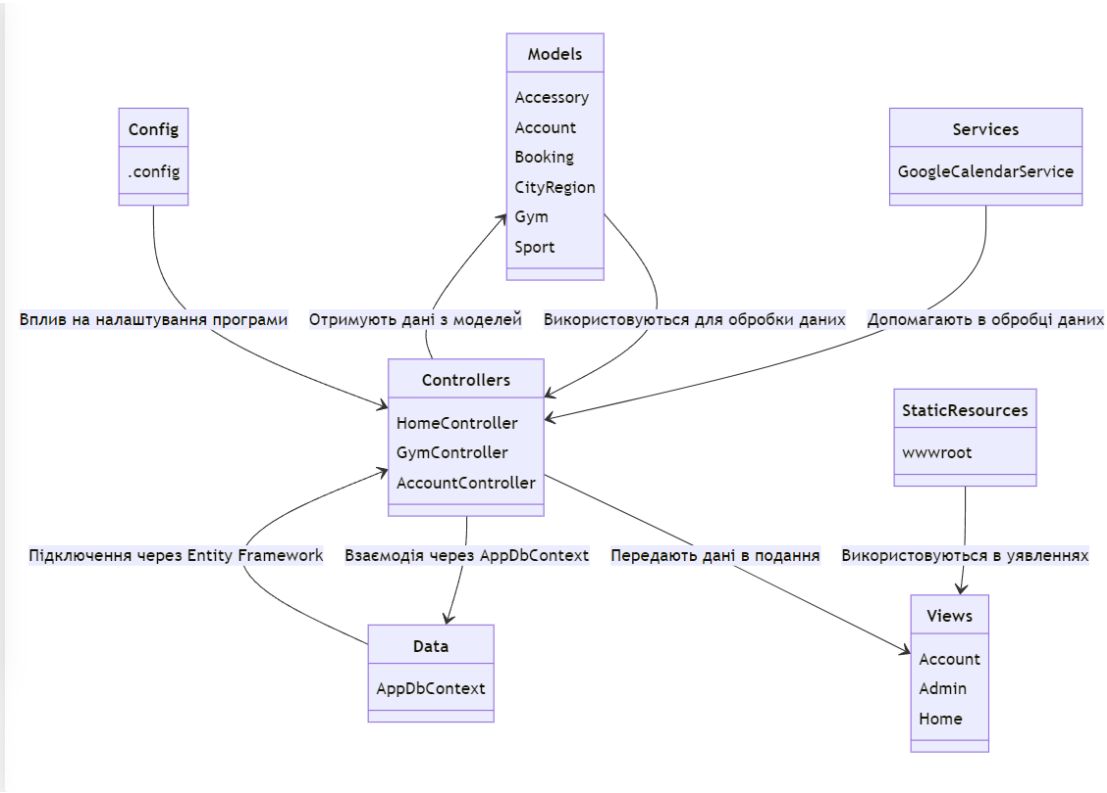


Рисунок 2.3 – Схема організації проекту

2.2 Розробка функціоналу бронювання

Розглянемо процес розробки ключових функцій бронювання в додатку "GymRent". Він охоплює розробку контролерів, моделей і уявлень, які забезпечують створення, управління та відображення бронювань спортзалів.

2.2.1 Розробка основного функціоналу системи

GymController відповідає за керування інформацією про спортзали, включно з додаванням, редагуванням і видаленням інформації про зали. Розглянемо методи цього контролера:

- реалізація функції відображення всіх спортзалів із можливістю фільтрації за містом і видами спорту (рисунок 2.2):

```
//Повертає список залів в місті в JSON
public async Task<IActionResult> Index(int cityId, int[] sportId)
{
    var gyms = await context.Gym
        .Where(g => g.city_id == cityId)
        .ToListAsync();
    if (sportId != null && sportId.Length > 0)
    {
        gyms = gyms
            .Where(gym => context.GymSport.Any(gs => gs.gym_id == gym.id && sportId.Contains(gs.sport_id)))
            .ToList();
    }
    return Json(gyms);
```

Рисунок 2.2 – Функція відображення спортзалів

- редагування інформації про зали, включно з керуванням фотографіями та доступними видами спорту (рисунок 2.3):

```
[HttpPost]
Ссылок: 0
public async Task<IActionResult> EditGym(EditGym model, IFormFile? photo)
{
    if (ModelState.IsValid)
    {
        var gym = await context.Gym.FindAsync(model.gym!.id);
        var gymAccessories = context.GymAccessory.Where(ga => ga.gym_id == model.gym.id);
        var gymAccessoryIds = gymAccessories.Select(ga => ga.accessory_id).ToList();
        var gymSports = context.GymSport.Where(ga => ga.gym_id == model.gym.id);
        var gymSportIds = gymSports.Select(ga => ga.sport_id).ToList();
        if (gym == null)
        {
            return NotFound();
        }
        // Delete old photo if it's not the default one
        if (!string.IsNullOrEmpty(gym.imagePath) && !gym.imagePath.Equals("/images/default-image.png") && photo != null)...

        // add photo
        if (photo != null && photo.Length > 0)...
        //else
        //{
        //    gym.imagePath = "/images/default-image.png";
        //}
        // Add new accessories
        foreach (var accessoryId in model.Accessories!)...
        // Add new sports
        foreach (var sportId in model.Sports!)...

        //Remove deleted accessories
        foreach (var gymAccessory in gymAccessories)...
        //Remove deleted sports
        foreach (var gymsport in gymSports)...
        gym.name = model.gym.name;
        gym.adress = model.gym.adress;
        gym.mapLocate = model.gym.mapLocate;
        gym.description = model.gym.description;
        gym.price = model.gym.price;
        gym.startwork = model.gym.startwork;
        gym.endwork = model.gym.endwork;

        await context.SaveChangesAsync();
        TempData["GoodMessage"] = "Gym edited!";
        return RedirectToAction("EditGym", new { gym.id });
```

Рисунок 2.3 – Функція редагування спортзалів

AccountController керує процесами автентифікації та реєстрації користувачів, а також надає функції керування профілем користувача.

Реалізацію реєстрації нових користувачів та їхню автоматичну автентифікацію після реєстрації наведено на рисунку 2.4.

```
[HttpPost]
public async Task<IActionResult> Register(RegisterModel model)
{
    if (ModelState.IsValid)
    {
        var user = new IdentityUser { UserName = model.Name};
        var result = await _userManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            await _userManager.AddToRoleAsync(user, "User");
            await _signInManager.SignInAsync(user, isPersistent: false);
            return RedirectToAction("Index", "Home");

        }
        foreach (var error in result.Errors)
        {
            ModelState.AddModelError(string.Empty, error.Description);
        }
    }
    return View(model);
```

Рисунок 2.4 – Функція реєстрації

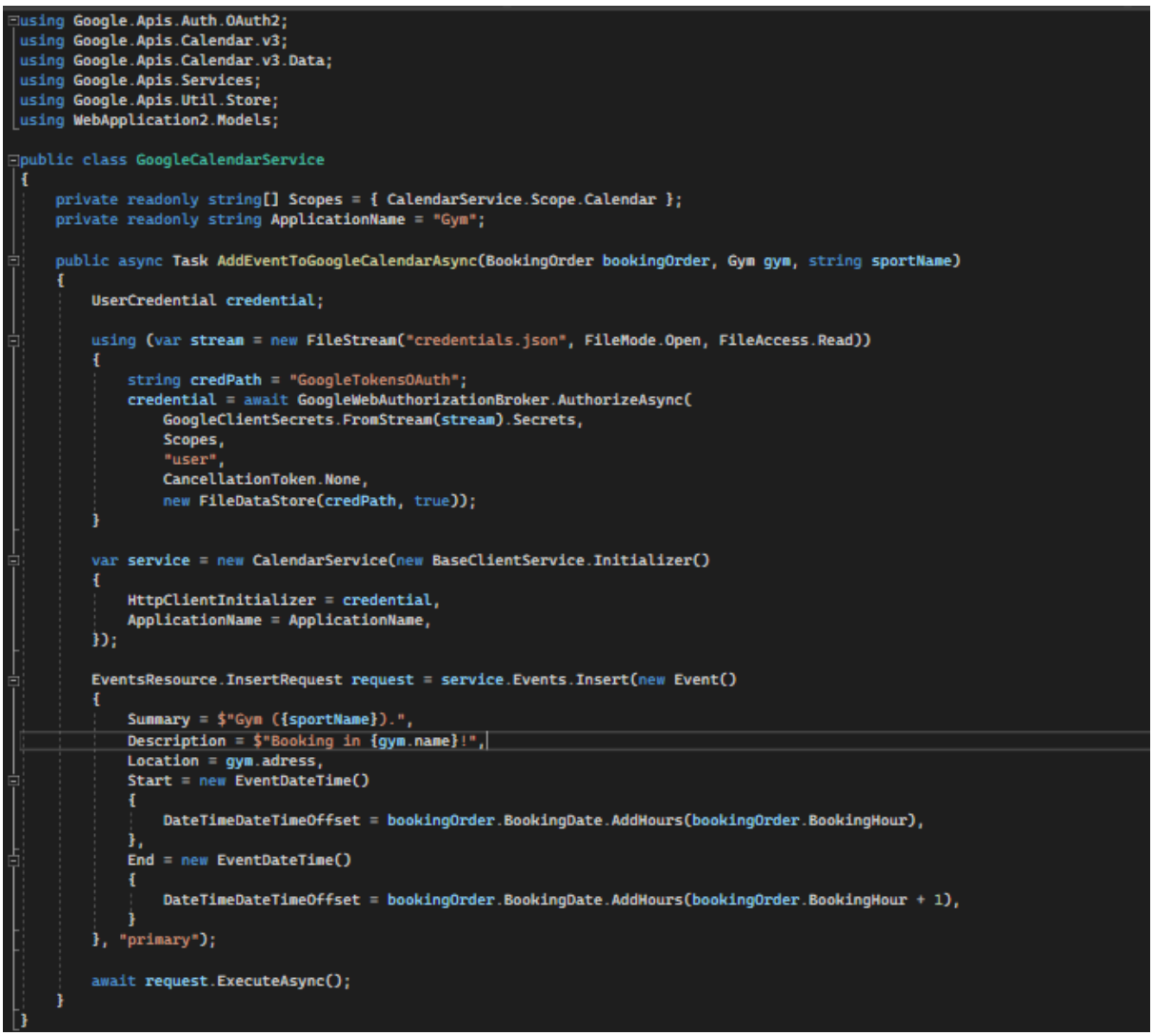
Взаємодія з базою даних відбувається через клас AppDbContext, який успадковується від IdentityDbContext і містить набори даних для кожної сутності (рисунок 2.5):

```
{
    public class AppDbContext : IdentityDbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

        public DbSet<Region> Region { get; set; }
        public DbSet<City> City { get; set; }
        public DbSet<Gym> Gym { get; set; }
        public DbSet<GymAccessory> GymAccessory { get; set; }
        public DbSet<Accessory> Accessory { get; set; }
        public DbSet<Sport> Sport   { get; set; }
        public DbSet<GymSport> GymSport { get; set; }
        public DbSet<BookingOrder> BookingOrders { get; set; }
    }
}
```

Рисунок 2.5 – Клас AppDbContext

GoogleCalendarService (рисунок 2.6) дає змогу додавати події до Google Календаря користувача на основі деталей бронювання, які включають інформацію про спортзал та обраний вид спорту. Робота сервісу починається з аутентифікації користувача через Google API.



```
using Google.Apis.Auth.OAuth2;
using Google.Apis.Calendar.v3;
using Google.Apis.Calendar.v3.Data;
using Google.Apis.Services;
using Google.Apis.Util.Store;
using WebApplication2.Models;

public class GoogleCalendarService
{
    private readonly string[] Scopes = { CalendarService.Scope.Calendar };
    private readonly string ApplicationName = "Gym";

    public async Task AddEventToGoogleCalendarAsync(BookingOrder bookingOrder, Gym gym, string sportName)
    {
        UserCredential credential;

        using (var stream = new FileStream("credentials.json", FileMode.Open, FileAccess.Read))
        {
            string credPath = "GoogleTokensOAuth";
            credential = await GoogleWebAuthorizationBroker.AuthorizeAsync(
                GoogleClientSecrets.FromStream(stream).Secrets,
                Scopes,
                "user",
                CancellationToken.None,
                new FileDataStore(credPath, true));
        }

        var service = new CalendarService(new BaseClientService.Initializer()
        {
            HttpClientInitializer = credential,
            ApplicationName = ApplicationName,
        });

        EventsResource.InsertRequest request = service.Events.Insert(new Event()
        {
            Summary = $"Gym ({sportName}).",
            Description = $"Booking in {gym.name}!",
            Location = gym.adress,
            Start = new EventDateTime()
            {
                DateTimeDateTimeOffset = bookingOrder.BookingDate.AddHours(bookingOrder.BookingHour),
            },
            End = new EventDateTime()
            {
                DateTimeDateTimeOffset = bookingOrder.BookingDate.AddHours(bookingOrder.BookingHour + 1),
            }
        }, "primary");

        await request.ExecuteAsync();
    }
}
```

Рисунок 2.6 - GoogleCalendarService

Розглянемо авторизацію користувача:

- використання OAuth 2.0 для Google API: сервіс використовує файл credentials.json для ініціалізації облікових даних користувача. Цей файл містить необхідні секрети для доступу до API;
- зберігання токенів: авторизація відбувається за допомогою GoogleWebAuthorizationBroker, який зберігає токени в директорію GoogleTokensOAuth. Це забезпечує можливість повторного використання токенів без необхідності повторної аутентифікації користувача.

Створення та надсилання події в календар:

- ініціалізація сервісу календаря: створюється екземпляр CalendarService з ініціалізованими обліковими даними та назвою програми;
- формування події: створюється об'єкт Event, який містить інформацію про захід, включно з назвою, описом, місцем розташування, часом початку та закінчення. Дані про час початку і закінчення формуються на основі часу бронювання;
- відправлення події: за допомогою методу Insert об'єкт Event відправляється в календар користувача на Google.

Цей сервіс значно спрощує процес керування розкладом відвідувань спортзалу, даючи змогу користувачеві бачити всі свої броні безпосередньо у своєму персональному календарі, що підвищує зручність використання веб-додатку.

Важливим аспектом проекту є інтеграція з Google Maps, що дозволяє користувачам візуально вибирати місцезнаходження спортзалів (рисунок 2.7), переглядаючи деталі прямо на карті. Реалізація цієї функції відбувається за допомогою Google Maps API, де кожен спортзал відображається як маркер на карті.

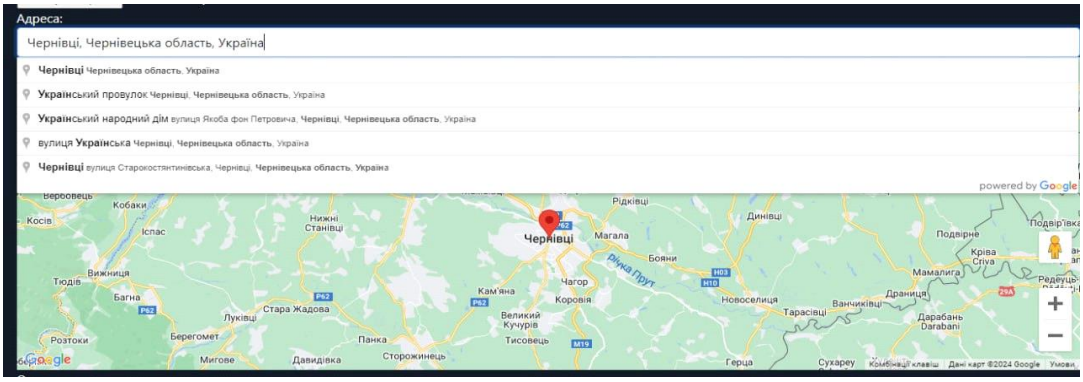


Рисунок 2.7 – Вибір адреси за допомогою Google Maps

Програмний код, який керує відображенням карт та маркерів, розміщений в файлі GymDetails.cshtml. В цьому Razor View файлі використовується JavaScript для ініціалізації та управління картою. Сценарій конфігурує карту, встановлює її центр у географічну позицію спортзалу і розміщує маркер на цій позиції з заголовком, що відображає назву спортзалу.

Такий підхід значно покращує інтерактивність додатку та забезпечує користувачам зручний спосіб оцінки локації спортзал.

На рисунку 2.8 представлена сторінка детальної інформації про тренажерний зал. Тут користувачі можуть :

- фотографії спортзалу, що демонструють його обладнання та інтер'єр;
- назву, опис і адресу спортзалу;
- ціни на послуги за кожним видом спорту (волейбол, футбол, баскетбол, бокс), що дає змогу користувачам оцінити вартість і вибрати відповідну активність;
- кнопка для бронювання, яка переведе користувача на сторінку вибору часу і підтвердження броні.

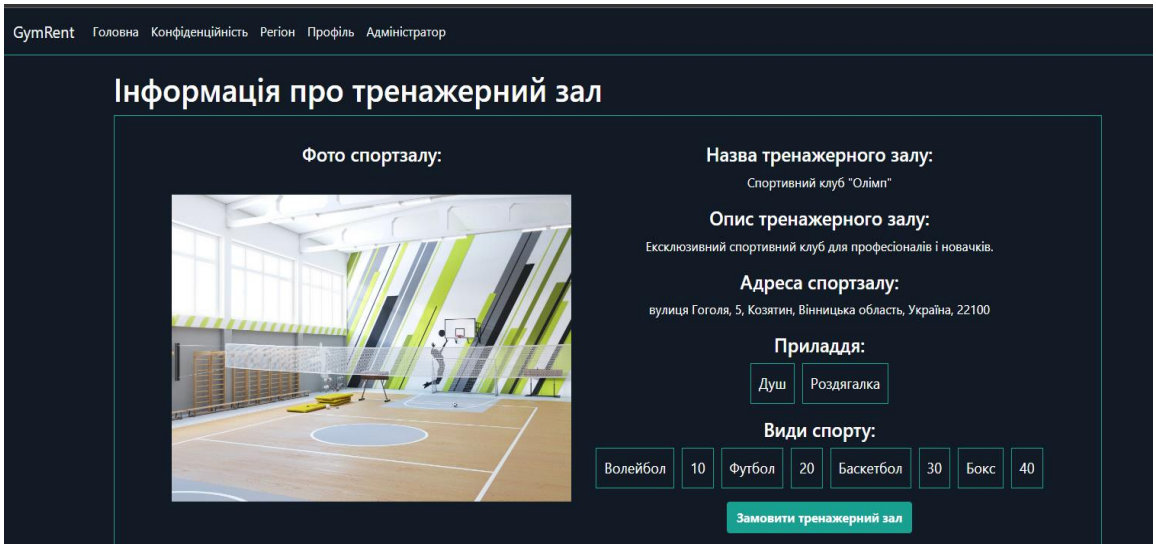


Рисунок 2.8 – Сторінка інформації про тренажерний зал

Код, пов'язаний із цією сторінкою, містить функції в GymController для отримання даних із бази даних і передавання їх у подання.

Детальний опис кожного елемента на сторінці дає змогу користувачам краще зрозуміти, що вони можуть робити на цій сторінці.

Взаємодія з базою даних через Entity Framework дає змогу динамічно оновлювати та відображати інформацію про тренажерні зали, що робить систему гнучкою та масштабованою.

Інтерфейс, зображений на рисунку 2.9 дає змогу користувачам переглядати список спортзалів, фільтруючи їх за містом і видом спорту. Це здійснюється через випадаючі списки на верхній панелі, що спрощує пошук відповідного залу в певному регіоні.

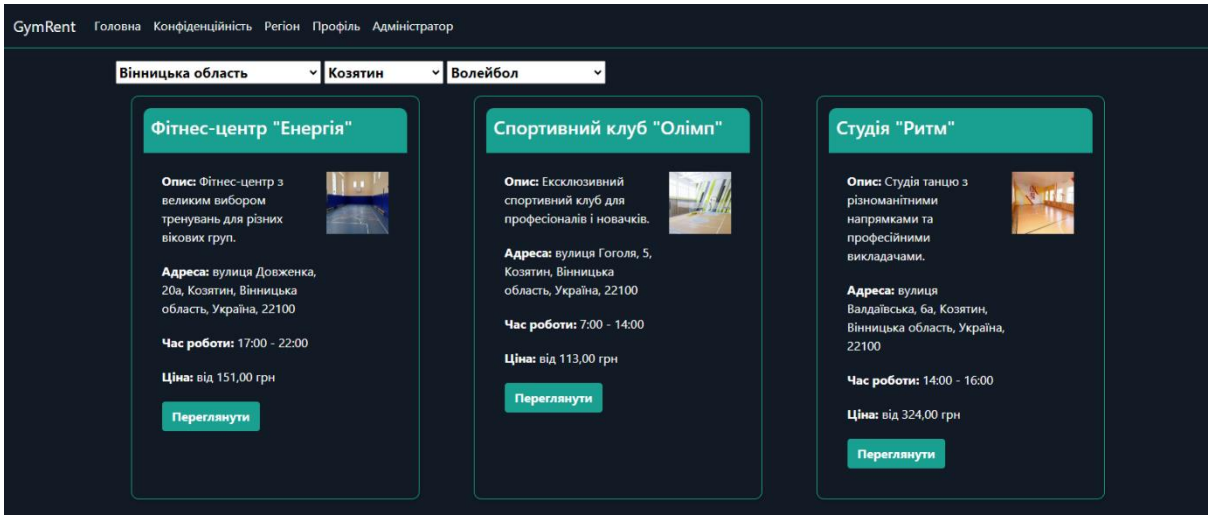
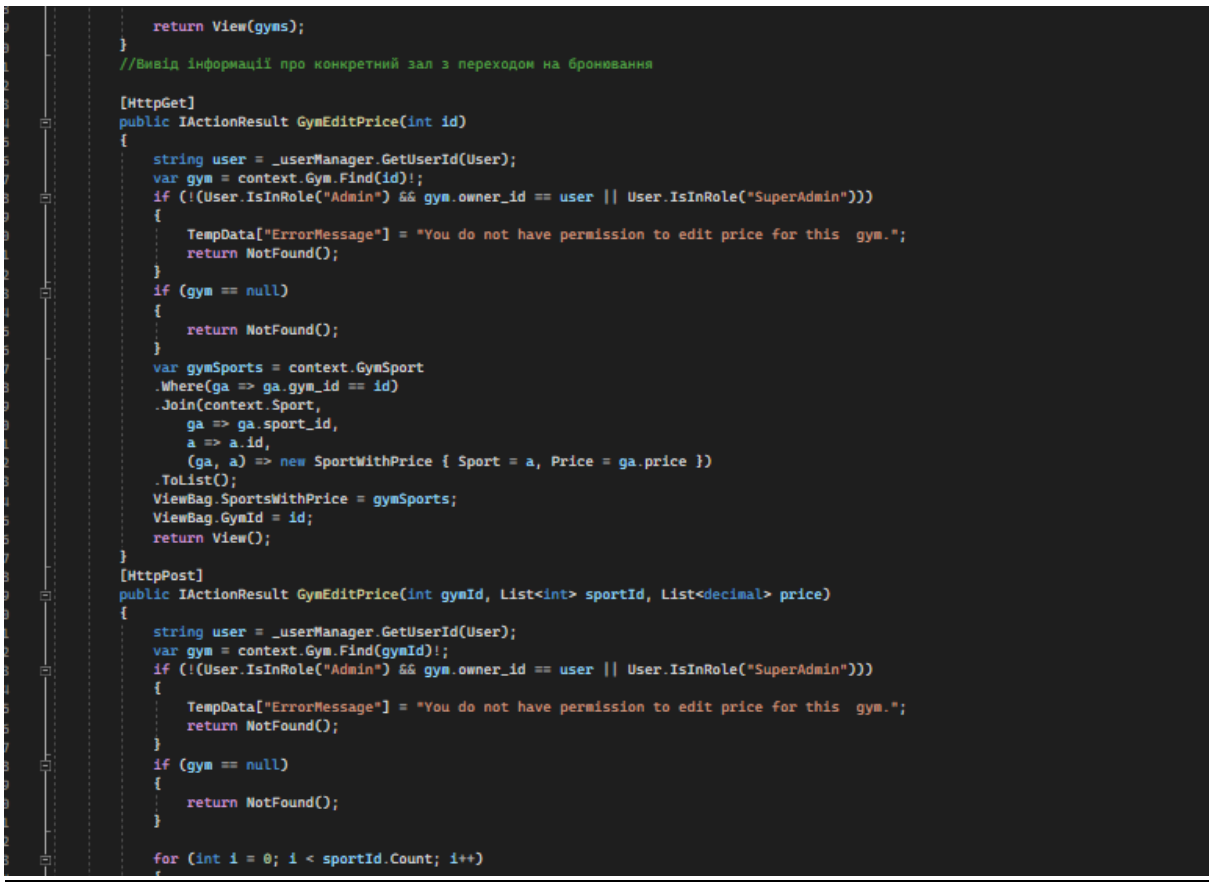


Рисунок 2.9 – Перегляд списку спортзалів

У коді це реалізовано через методи в GymController (рисунок 2.10), які фільтрують дані на основі обраних параметрів і повертають їх у вигляді списку для відображення в поданні.



```
            return View(gyms);
        }
        //Вивід інформації про конкретний зал з переходом на бронювання

        [HttpGet]
        public IActionResult GymEditPrice(int id)
        {
            string user = _userManager.GetUserId(User);
            var gym = context.Gym.Find(id)!;
            if (!(User.IsInRole("Admin") && gym.owner_id == user || User.IsInRole("SuperAdmin")))
            {
                TempData["ErrorMessage"] = "You do not have permission to edit price for this  gym.";
                return NotFound();
            }
            if (gym == null)
            {
                return NotFound();
            }
            var gymSports = context.GymSport
            .Where(ga => ga.gym_id == id)
            .Join(context.Sport,
                ga => ga.sport_id,
                a => a.id,
                (ga, a) => new SportWithPrice { Sport = a, Price = ga.price })
            .ToList();
            ViewBag.SportsWithPrice = gymSports;
            ViewBag.GymId = id;
            return View();
        }
        [HttpPost]
        public IActionResult GymEditPrice(int gymId, List<int> sportId, List<decimal> price)
        {
            string user = _userManager.GetUserId(User);
            var gym = context.Gym.Find(gymId)!;
            if (!(User.IsInRole("Admin") && gym.owner_id == user || User.IsInRole("SuperAdmin")))
            {
                TempData["ErrorMessage"] = "You do not have permission to edit price for this  gym.";
                return NotFound();
            }
            if (gym == null)
            {
                return NotFound();
            }

            for (int i = 0; i < sportId.Count; i++)
```

Рисунок 2.10 – Методи фільтрації даних

Розглянемо сторінку, яка являє собою інтерфейс для управління бронюваннями (рисунок 2.11). На ній можна:

- переглядати всі поточні бронювання за різними критеріями: дата, спорт тощо;
- використовувати фільтри для пошуку конкретних бронювань;
- подивитися детальну інформацію щодо кожного бронювання, включно з вартістю і часом.

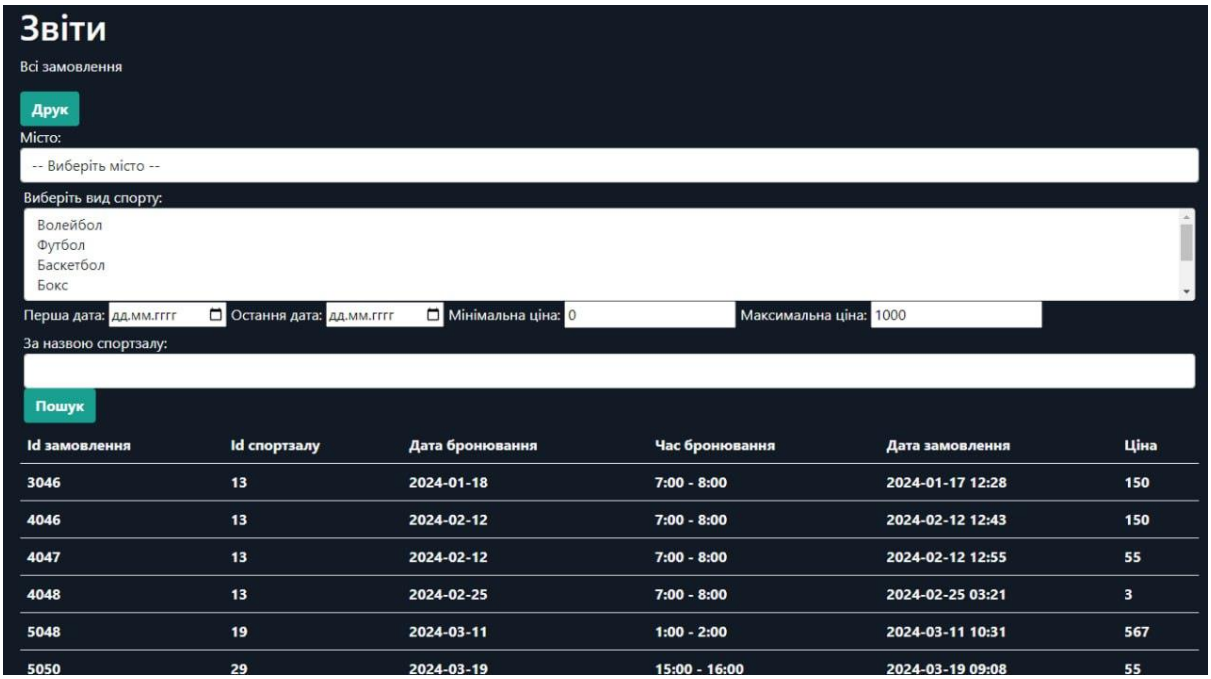


Рисунок 2.11 – Панель бронювань

У верхній частині сторінки присутні фільтри для вибору дати бронювання. Це дає змогу адміністратору обмежити відображувані записи обраним часовим інтервалом, забезпечуючи зручний пошук бронювань у певні дати. Користувач може також фільтрувати дані за ціною (від мінімальної до максимальної), що дає змогу швидко знаходити бронювання в заданому ціновому діапазоні.

Стовпці таблиці:

- ID бронювання: унікальний ідентифікатор бронювання, який дає змогу легко посилатися на конкретне бронювання;
- ID спортзалу: показує, в якому спортзалі було зроблено бронювання;
- дата бронювання: точна дата, на яку було здійснено бронювання;
- час початку: час початку заброньованого сеансу;
- час закінчення: час закінчення заброньованого сеансу;
- ціна: вартість бронювання.

Користувач може переходити сторінками таблиці для перегляду додаткових записів.

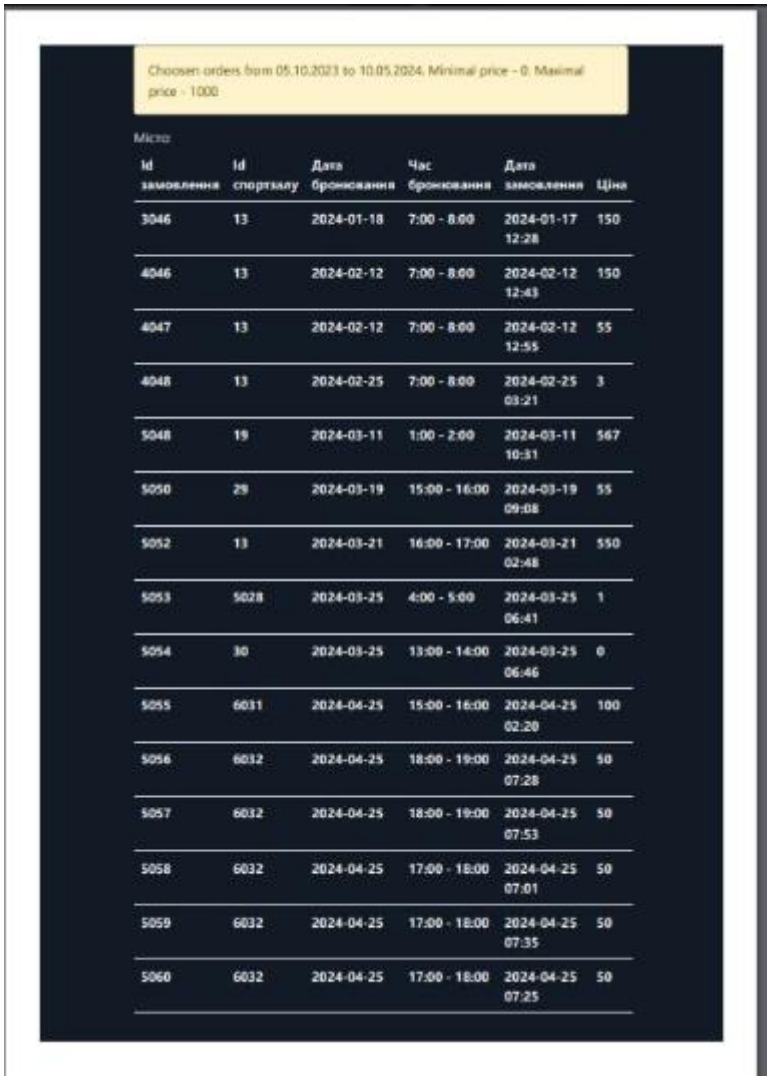


Рисунок 2.12 – Відфільтровані записи

Користувачі мають можливість надсилати відгуки про спортзали через Telegram (рисунок 2.13), що полегшує збір зворотного зв'язку і підвищує задоволеність клієнтів. Користувачі також можуть ділитися інформацією про зал зі своїми друзями через Telegram, що сприяє організації спільних занять і підвищує соціальну активність.

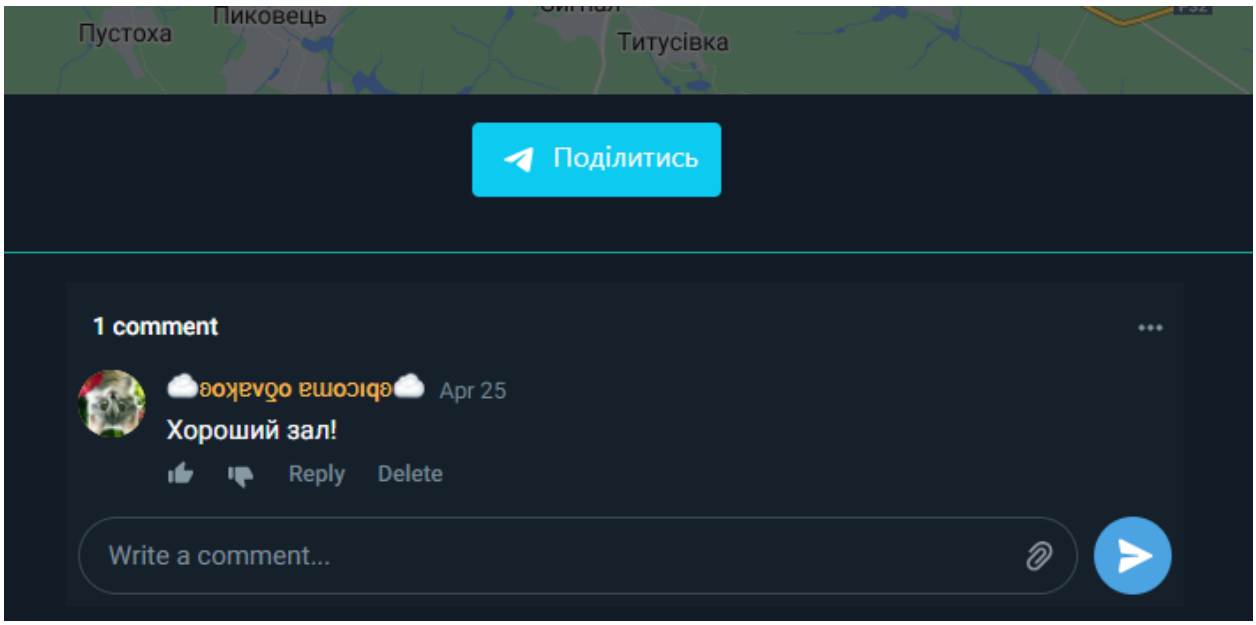


Рисунок 2.13 – Надсилання відгуку

2.2.2 Тестування системи

Тестування веб-додатка "GymRent" є важливим етапом розробки, оскільки воно гарантує стабільність, безпеку та відповідність додатка вимогам користувача. Процес тестування включає в себе кілька ключових аспектів

Юніт-тестування:

- юніт-тести написані для перевірки бізнес-логіки, оброблюваної моделями і методами контролерів;
- використовуються фреймворки, як-от xUnit і NUnit, для написання і виконання тестів.

Приклад юніт-тесту для методу бронювання:

```
public void TestBookingFunctionality() {
    var bookingController = new BookingController(context, userManager);
    var result = bookingController.Book("gymId", new BookingModel { /* */ });
    Assert.True(result is SuccessResult);
}
```

Розділ 3. Інтерфейс користувача та його взаємодія із системою

У сучасних інформаційних системах, особливо у веб-додатках, інтерфейс користувача (UI) відіграє критично важливу роль. Саме через інтерфейс користувачі взаємодіють з системою, виконують свої завдання та отримують необхідну інформацію. Ефективний дизайн UI забезпечує інтуїтивність, зручність та ефективність роботи користувачів, що сприяє задоволенню їхніх потреб та досягненню цілей бізнесу.

На рисунку 3.1 наведена діаграма можливої взаємодії з системою, яка ілюструє основні сценарії та потоки дій між користувачем та різними компонентами системи. Ця діаграма демонструє процеси від входу на сайт до завершення сеансу, включаючи реєстрацію, вибір спортивної зали, бронювання часу, підтвердження оплати та інтеграцію з Google Calendar для управління подіями.

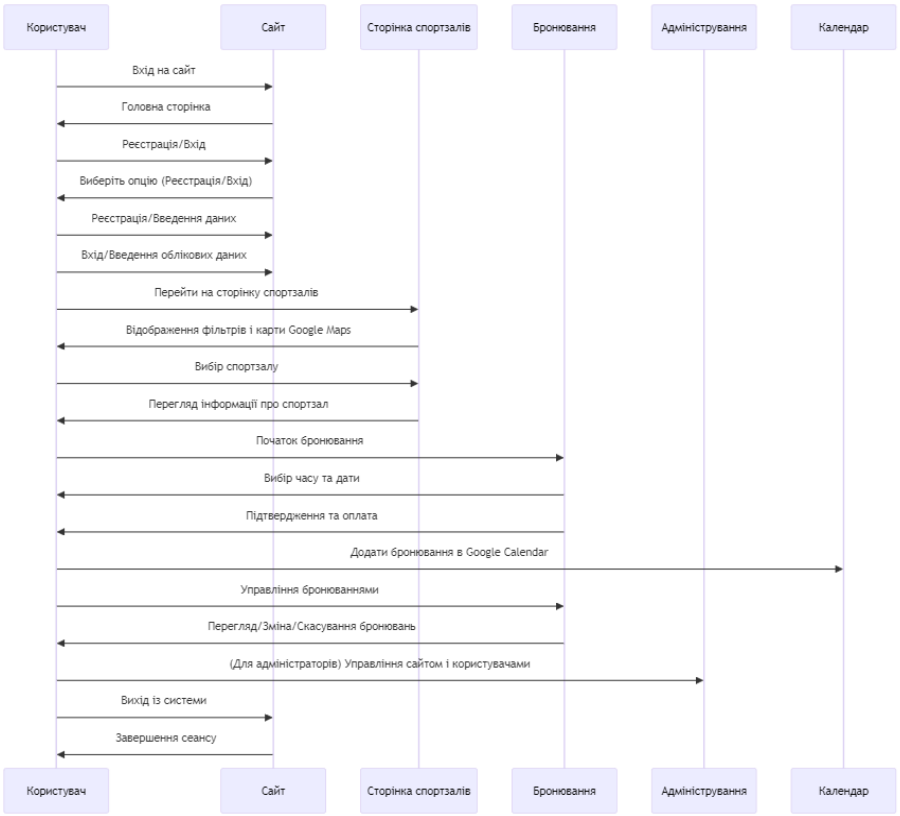


Рисунок 3.1 – Взаємодія користувача з системою

Розглянемо детальніше наведені на діаграмі процеси взаємодії користувача із системою.

Після входу на сайт користувач може побачити головну сторінку, на якій зображений перелік всіх можливих спортзалів. Є можливість відфільтрувати їх за областю, містом та видами спорту, доступними в спортзалі. Це здійснюється через випадаючі списки на верхній панелі, що спрощує пошук відповідного залу в певному регіоні. Панель навігації дозволяє переходити між головною, інформаційними та адміністративними сторінками.

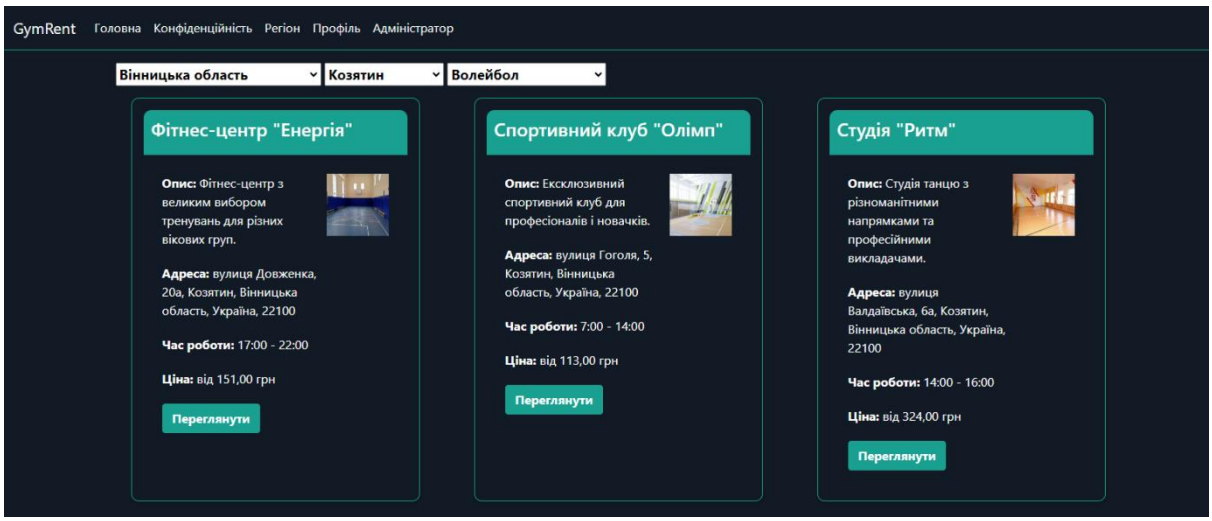


Рисунок 3.2 – Головна сторінка сайту

Для кожного спортзала є сторінка з детальною інформацією (рисунок 3.3), яка містить:

- фотографії спортзалу, що демонструють його обладнання та інтер'єр;
- назву, опис і адресу спортзалу;
- ціни на послуги за кожним видом спорту (волейбол, футбол, баскетбол, бокс), що дає змогу користувачам оцінити вартість і вибрати відповідну активність.

Це допомагає користувачу зробити поінформований вибір перед бронюванням.

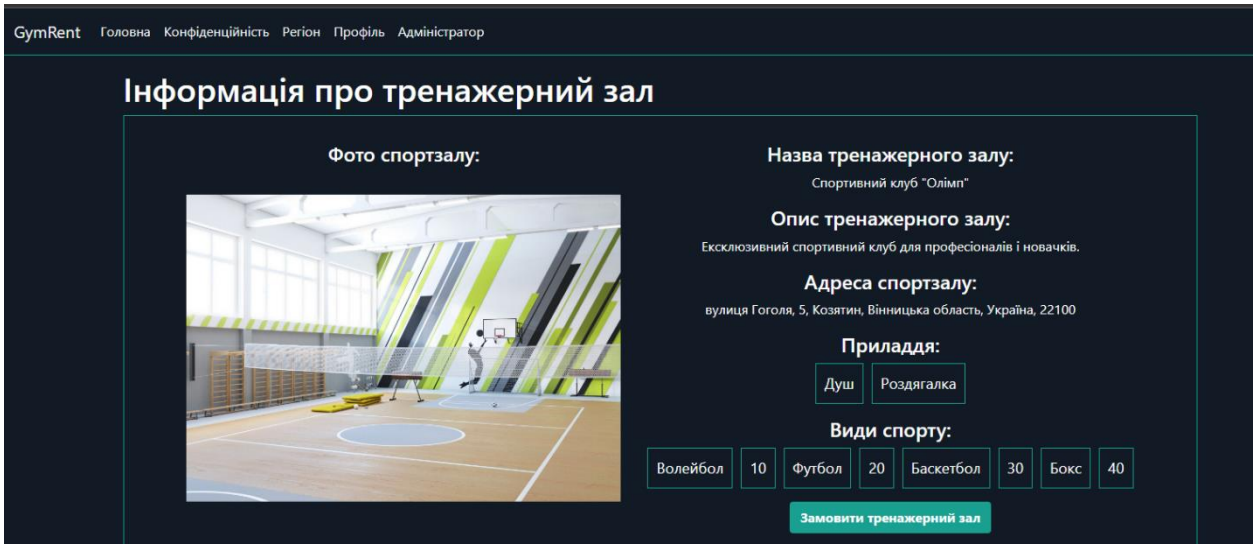


Рисунок 3.3 – Сторінка з детальною інформацією про спортзал

На цій сторінці є можливість забронювати зал, а також переглянути відгуки про спортзал та залишити свій через Telegram (рисунок 3.4), що полегшує збір зворотного зв'язку і підвищує задоволеність клієнтів. Користувачі також можуть ділитися інформацією про зал зі своїми друзями через Telegram, що сприяє організації спільних занять і підвищує соціальну активність.

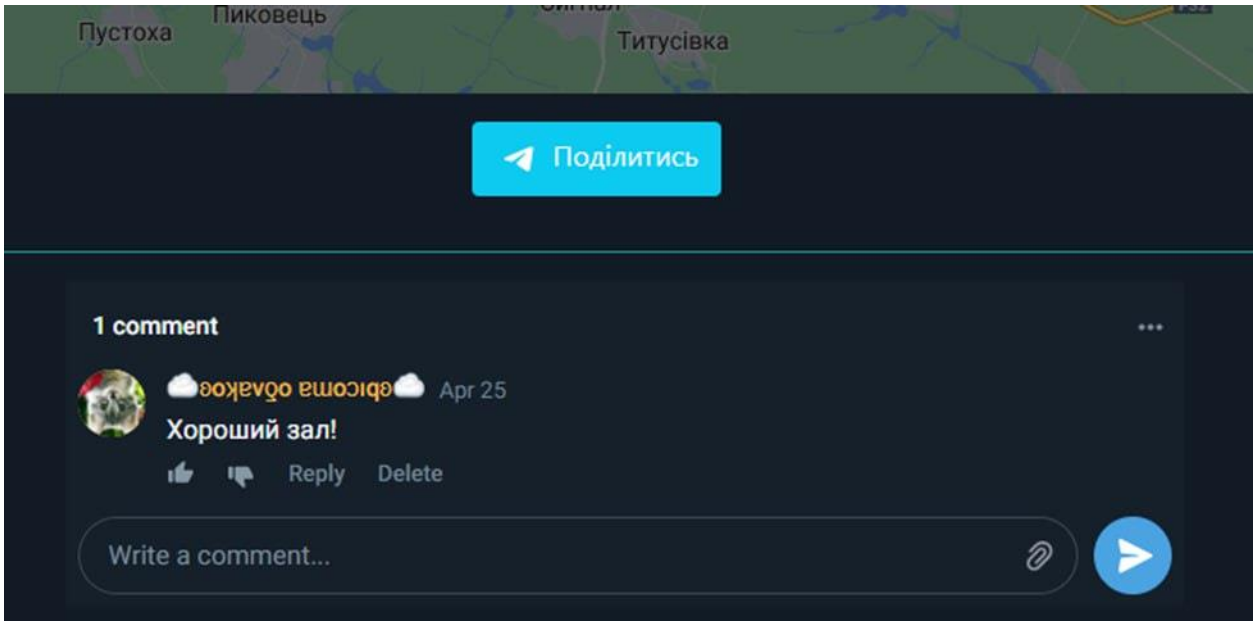


Рисунок 3.4 – Відгуки

Кнопка для бронювання переводить користувачів на сторінку вибору часу і підтвердження броні (рисунок 3.5). Вони мають можливість переглядати доступні часові слоти для різних видів спорту в обраному спортзалі, що робить процес бронювання інтуїтивно зрозумілим і зручним.

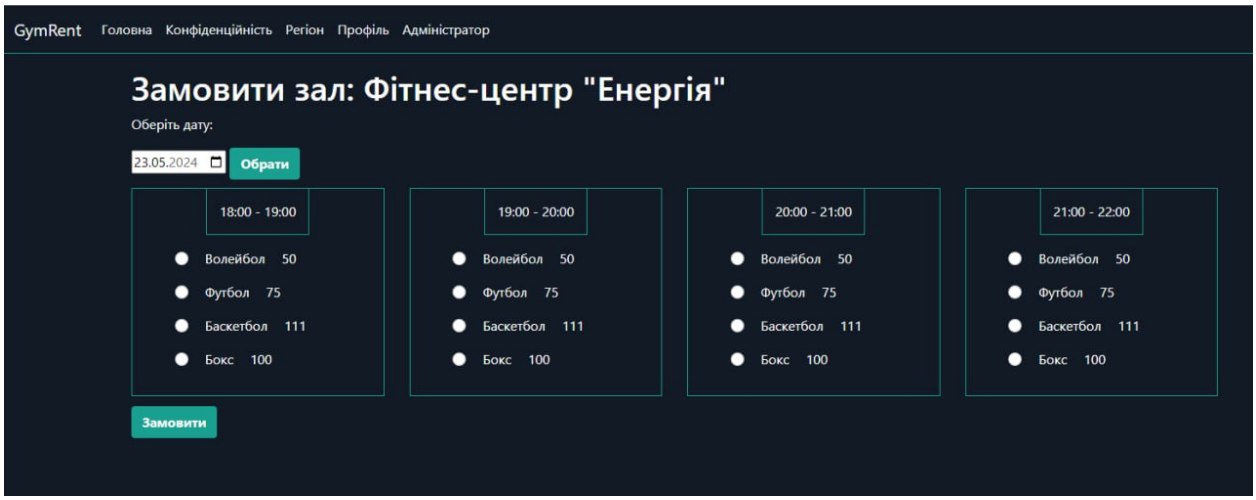


Рисунок 3.5 – Бронювання тренажерного зала

Спливаюче вікно з пропозицією додати подію в календар покращує зручність використання (рисунок 3.6), інтегруючи бронювання з особистим календарем користувача.

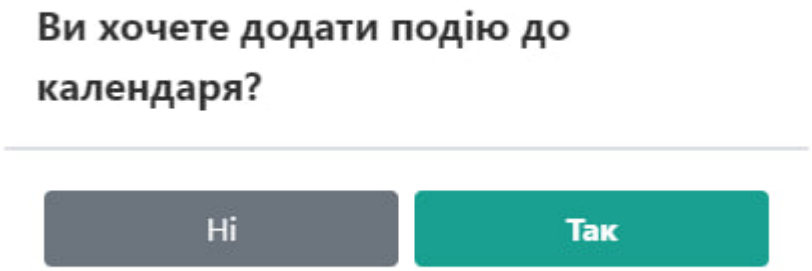


Рисунок 3.6 – Підтвердження додавання події в календар

Додана до календаря подія містить інформацію про обрану активність, час, назву спортзалу та його адресу (рисунок 3.7). Завдяки цьому користувачі можуть отримувати сповіщення з нагадуваннями.

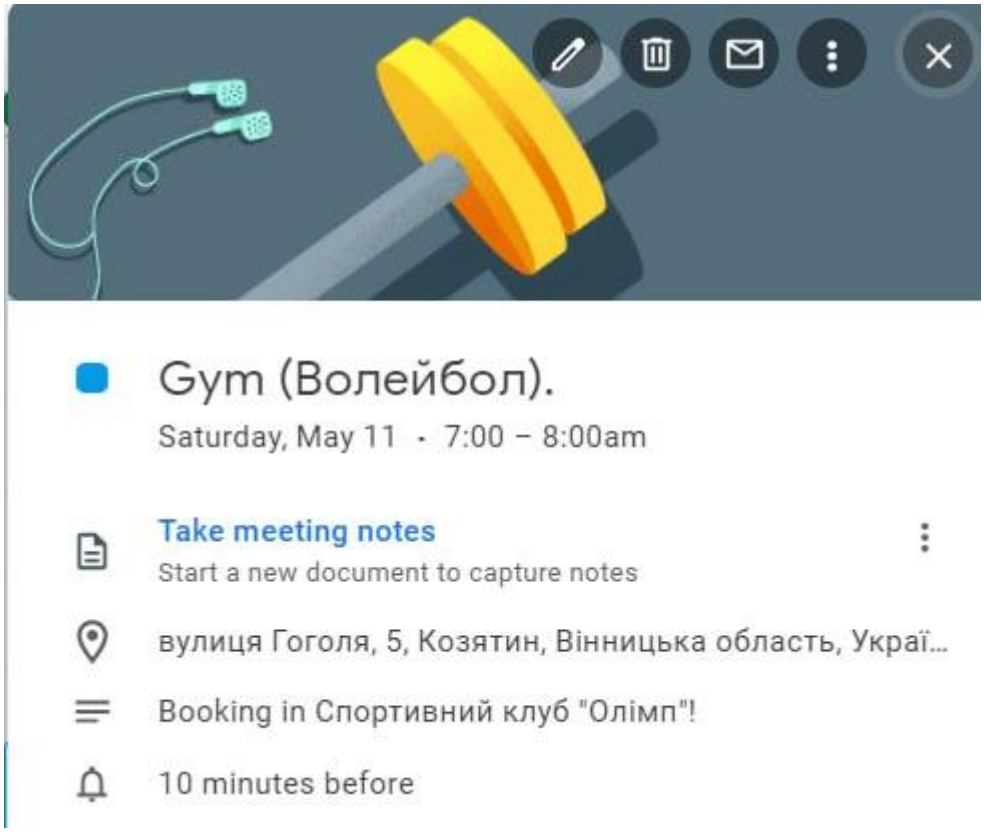


Рисунок 3.7 – Подія в календарі

Для того щоб отримати доступ до розширених можливостей адміністратор повинен авторизуватись. Сторінка входу у систему (рисунок 3.8) містить просту і чисту форму з полями для імені користувача і пароля, а також опцією "запам'ятати мене", що підкреслює зручність і безпеку доступу до функцій сайту.

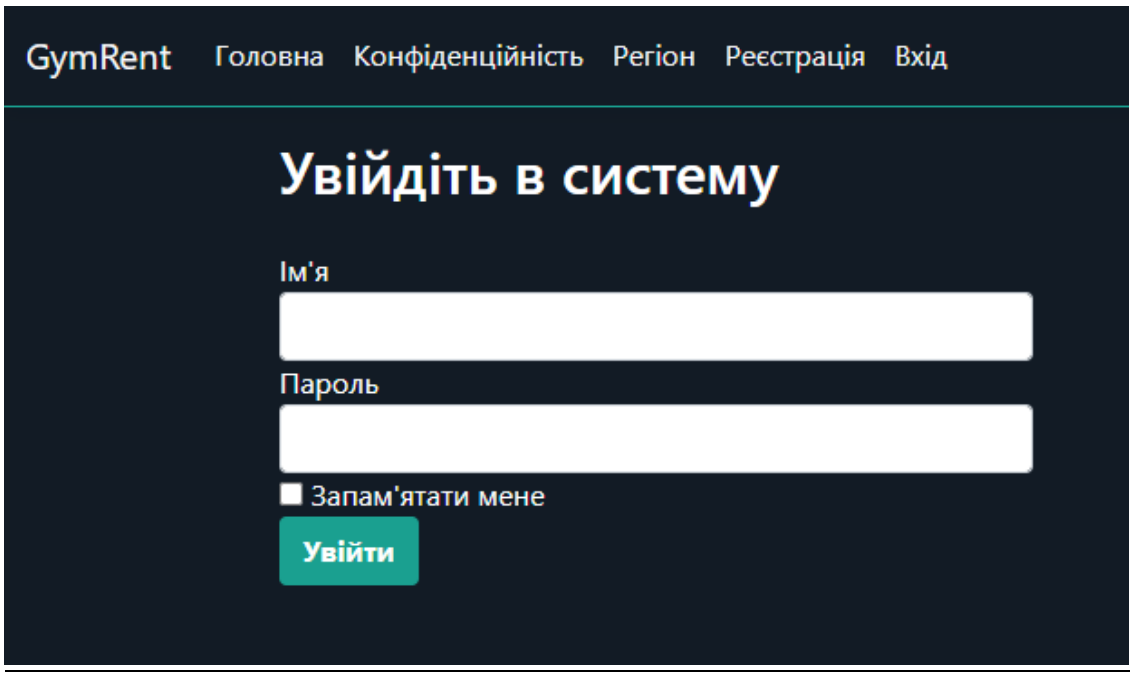


Рисунок 3.8 – Сторінка входу у систему

На панелі управління адміністратор має можливість керувати тренажерними залами, а також отримувати певну звітність. У звітах (рисунок 3.9) суперадмін може бачити історію всіх бронювань, а звичайний – лише тих залів, як він створив. На цій сторінці видно детальну інформацію про бронювання, включно з датою та часом.

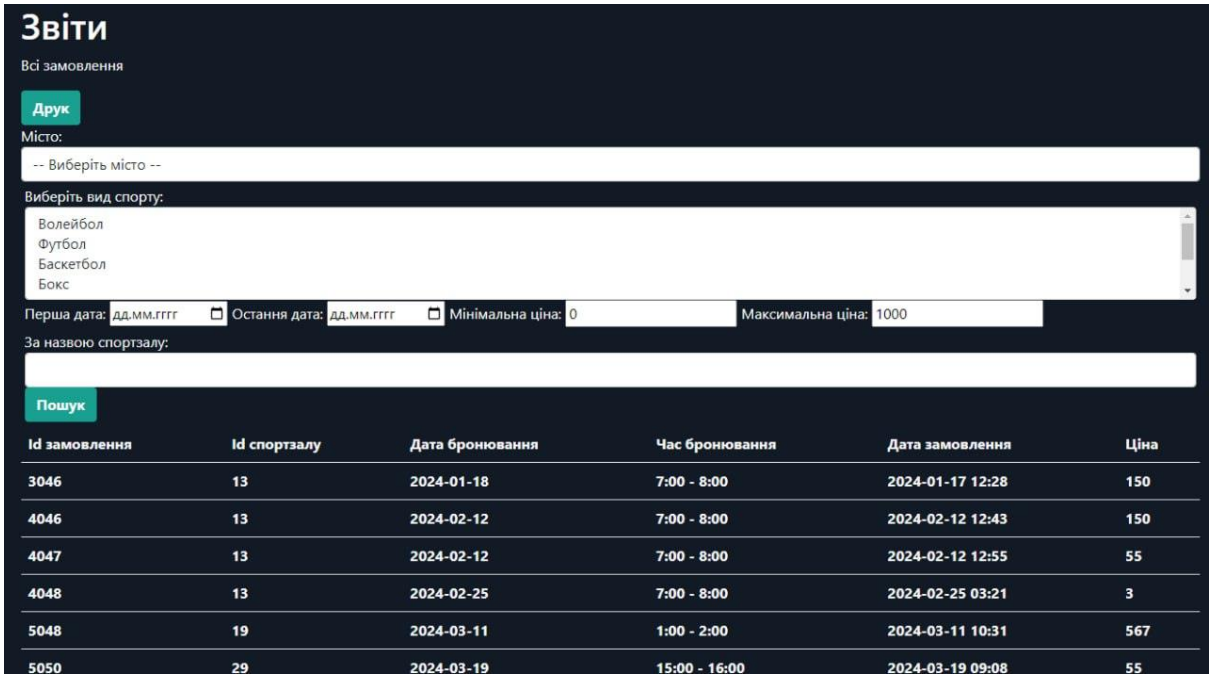


Рисунок 3.9 – Звітність

У верхній частині сторінки присутні фільтри для вибору дати бронювання. Це дає змогу обмежити відображувані записи обраним часовим інтервалом, забезпечуючи зручний пошук бронювань у певні дати. Можна також фільтрувати дані за ціною (від мінімальної до максимальної), що дає змогу швидко знаходити бронювання в заданому ціновому діапазоні.

Стовпці таблиці:

- ID бронювання: унікальний ідентифікатор бронювання, який дає змогу легко посилатися на конкретне бронювання;
- ID спортзалу: показує, в якому спортзалі було зроблено бронювання;
- дата бронювання: точна дата, на яку було здійснено бронювання;
- час початку: час початку заброньованого сеансу;
- час закінчення: час закінчення заброньованого сеансу;
- ціна: вартість бронювання.

Користувач може переходити сторінками таблиці для перегляду додаткових записів.

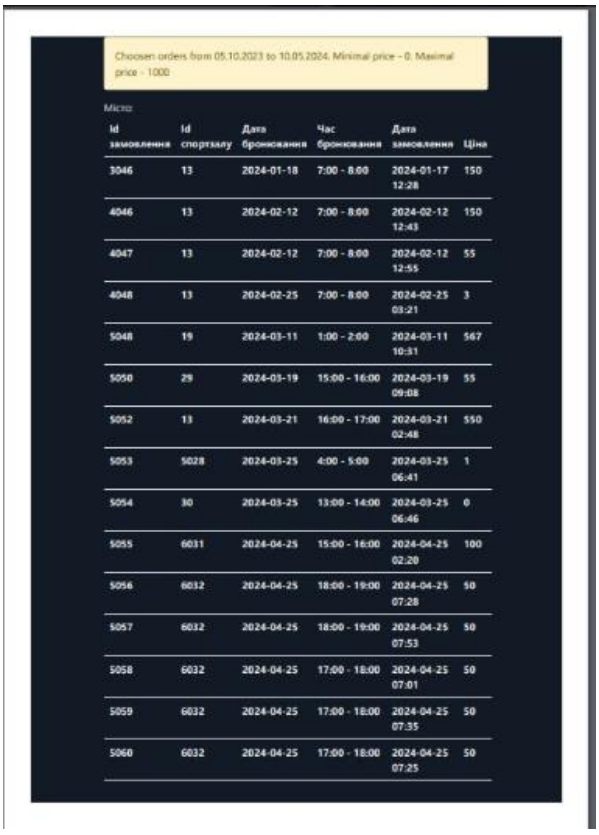


Рисунок 3.10 – Відфільтровані записи

Не менш важливою є можливість адміністратора керувати своїми спортзалами. Сторінка зі списком з доступними для управління залами має інтуїтивний дизайн, що дозволяє швидко виконувати CRUD-операції.

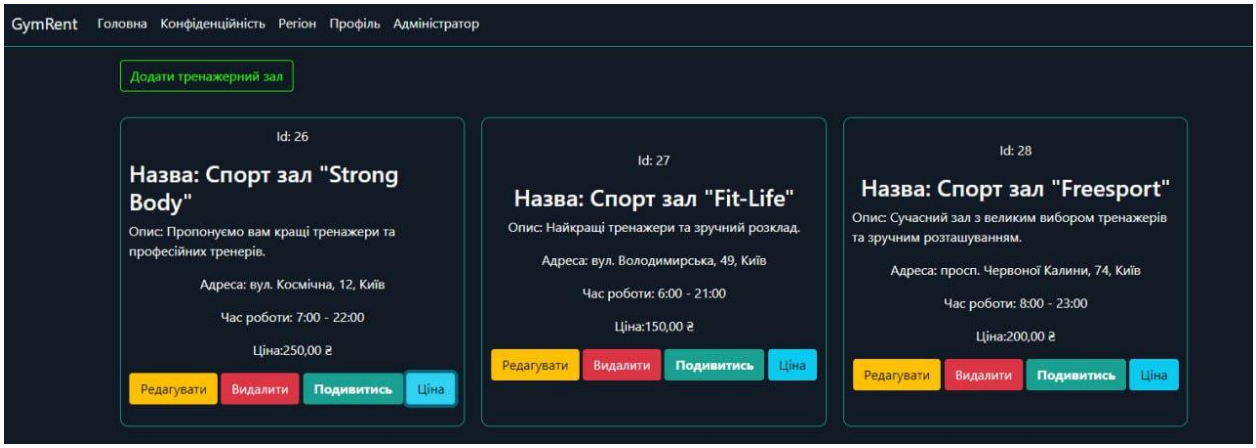


Рисунок 3.11 – Список залів адміністратора

Після натискання на кнопку «Додати тренажерний зал» система надає форму (рисунок 3.12), в якій потрібно ввести назву, адресу, опис, ціну та додати фотографію. Використання Google Maps для вибору адреси спрощує цей процес за рахунок візуального представлення місця розташування.

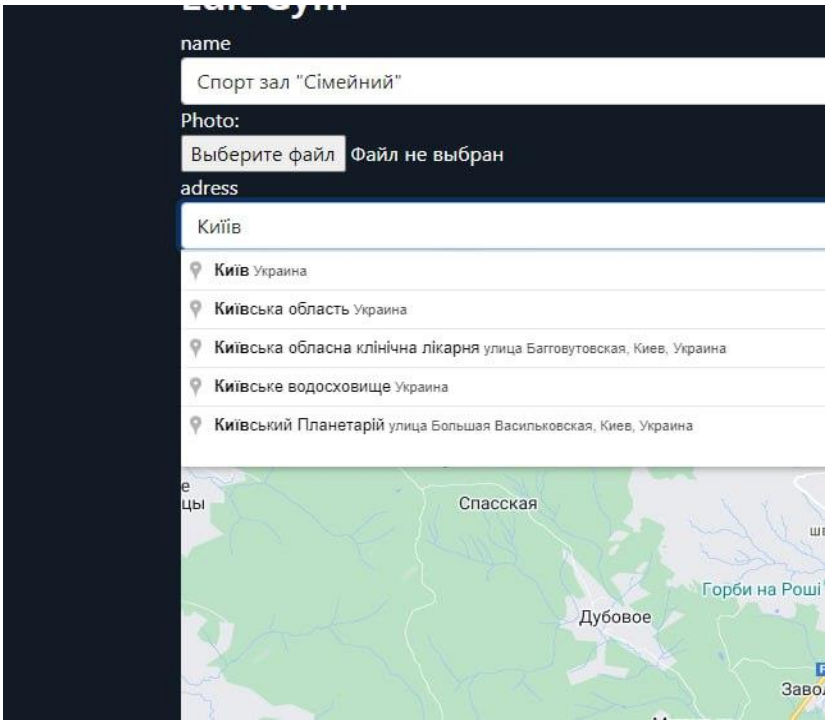


Рисунок 3.12 – Додавання нового спортзалу

Крім того, існує окрема сторінка для додавання та видалення видів спорту (рисунок 3.13), доступних у спортивному залі. Це дозволяє адміністраторам легко керувати переліком послуг, що надаються, та підтримувати актуальність інформації для користувачів.

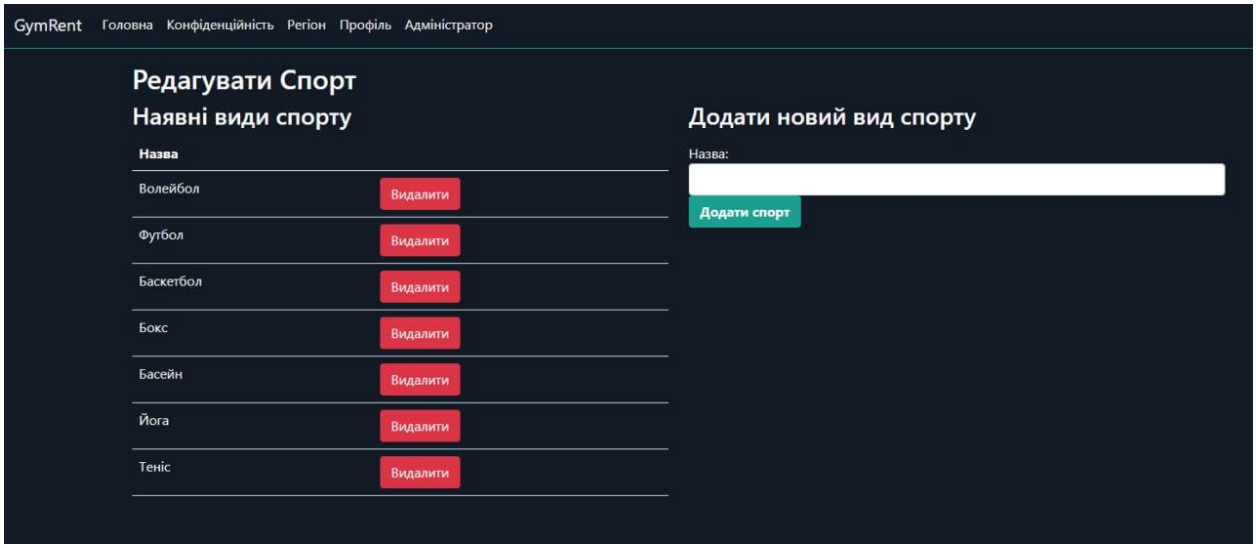


Рисунок 3.13 – Керування видами спорту

Також можна змінювати ціну на кожен вид спорту окремо (рисунок 3.14), що забезпечує гнучкість у ціноутворенні та дозволяє швидко реагувати на зміни попиту або ринкових умов.

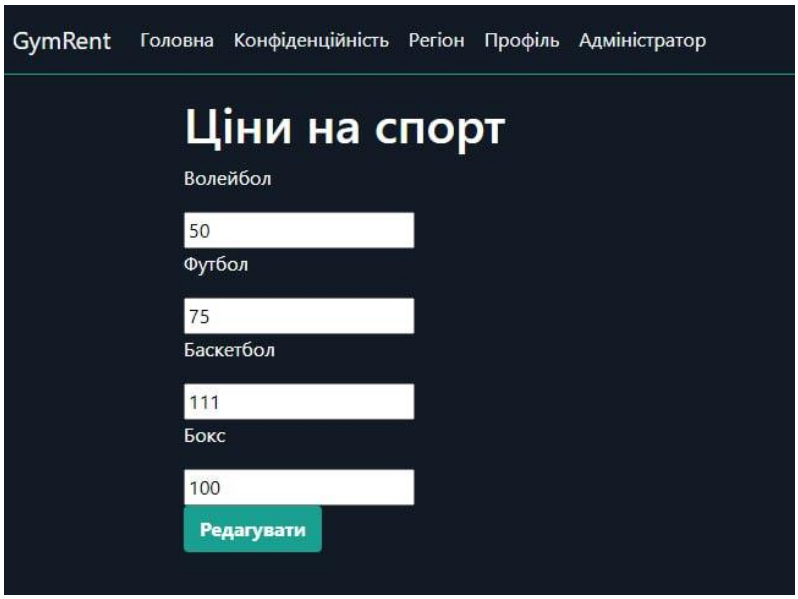


Рисунок 3.14 – Редагування цін на види спорту

Висновок

Проект "GymRent" успішно демонструє, як сучасні веб-технології можуть бути використані для вирішення практичних завдань у сфері фітнес-індустрії. Система бронювання спортзалів, розроблена в рамках цього проекту, є потужним інструментом, що спрощує процес пошуку та бронювання спортивних закладів для кінцевих користувачів, водночас надаючи зручні та ефективні інструменти управління для адміністраторів.

Основні досягнення проекту включають:

- інтеграція з Google Maps: використання Google Maps API дозволяє користувачам легко знаходити спортивні зали на карті, переглядати їх місцезнаходження та отримувати маршрути до них. Це значно підвищує зручність користування системою, надаючи візуальний контекст та полегшуючи процес планування відвідувань;
- відгуки за допомогою Telegram: система надає можливість користувачам надсилати відгуки про спортивні зали через Telegram, що полегшує збір зворотного зв'язку і підвищує задоволеність клієнтів. Це дозволяє оперативно отримувати інформацію про якість обслуговування та умови у залі;
- додавання фотографій: користувачі можуть переглядати фотографії спортивних залів, що демонструють їх обладнання та інтер'єр. Це допомагає користувачам прийняти поінформоване рішення про вибір залу, орієнтуючись на його візуальні характеристики;
- ціна за кожен вид спорту: система відображає детальну інформацію про ціни за кожен вид спорту, що дозволяє користувачам легко оцінити вартість послуг і вибрати відповідну активність. Це забезпечує прозорість цінової політики та полегшує процес порівняння різних залів;
- можливість поділитися через Telegram: користувачі можуть ділитися інформацією про заброньовані зали або майбутні події через Telegram, що сприяє соціалізації та допомагає організовувати спільні заняття зі своїми друзями та знайомими;

- бронювання в календарі: після успішного бронювання, система дозволяє додати подію до особистого календаря користувача. Це полегшує управління розкладом і допомагає користувачам не забувати про свої спортивні заняття. Інтеграція з календарем підвищує зручність використання та організованість.

Насамкінець, проект "GymRent" не тільки успішно вирішує завдання спрощення процесу бронювання спортивних залів та використовує сучасні технології для забезпечення зручності і ефективності користувачів, одночасно надаючи власникам спортзалів можливість ефективно керувати своїми ресурсами та отримувати прибуток. Проект є яскравим прикладом того, як технології можуть сприяти розвитку спорту та здорового способу життя і відкриває широкі перспективи для подальшого розвитку та вдосконалення проекту.

Список використаних джерел

1. Razor Pages [Електронний ресурс]. - URL: https://learn.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-8.0&tabs=visual-studio
2. SQL Server Express [Електронний ресурс]. - URL: https://en.wikipedia.org/wiki/SQL_Server_Express
3. Entity Framework [Електронний ресурс]. - URL: https://uk.wikipedia.org/wiki/Entity_Framework
4. ASP .NET Indentity [Електронний ресурс]. - URL: https://learn.microsoft.com/uk-ua/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio
5. Telegram Discussion Widget [Електронний ресурс]. - URL: https://core.telegram.org/widgets/discussion