

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОШУКУ ПАРТНЕРІВ ДЛЯ
ЗАНЯТЬ СПОРТОМ
SPORTS PARTNER SEARCH SOFTWARE

Виконав: студент(ка) 4 курсу, групи КНТ-138
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Терлецький С.В.

(прізвище та ініціали)

Керівник Льовкін В.М.

(прізвище та ініціали)

Рецензент Скрупський С.Ю.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Терлецького Станіслава Віталійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення пошуку партнерів для занять спортом
Sports Partner Search Software

керівник проєкту (роботи) Льовкін Валерій Миколайович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «27» квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) червень 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програмного забезпечення. 3. Розробка програми. 4. Експлуатація та тестування програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Льовкін В.М., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання « 4 » квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Проектування програми	3 тиждень	Розділ 2
4	Розробка програми	4-5 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	6 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
7	Захист роботи	8 тиждень	

Студент(ка)

(підпис)

Терлецький С.В.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Льовкін В.М.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 88 с., 25 рис., 3 табл., 3 дод., 10 джерел.

СПОРТ, СПОРТИВНИЙ МАЙДАНЧИК, СПІЛЬНЕ ЗАНЯТТЯ СПОРТОМ, ПАРТНЕРИ, ASP.NET CORE.

Об'єкт роботи – процес пошуку партнерів для занять спортом. Предмет роботи – програмне забезпечення пошуку партнерів для занять спортом. Мета роботи – розробити програмне забезпечення пошуку партнерів для спільних занять видами спорту, що потребують декількох учасників, у випадку відсутності партнерів.

Матеріали, методи та технічні засоби: мова програмування C#, вебфреймворк ASP.NET Core, шаблон проєктування Model-View-Controller, система керування реляційними базами даних Microsoft SQL Server та утиліта для адміністрування SQL Server Management Studio, фреймворк для роботи з даними Entity Framework Core, інтегроване середовище розробки Visual Studio, мови HTML і CSS, бібліотека Bootstrap.

Результати. Основним результатом є розроблений вебзастосунок, включаючи окрім самого застосунку ще результати його проєктування, розроблені в структурі програми класи моделей, представлень, контролерів, сервісів та інтерфейсів до них.

Висновки. Програма розроблена для підтримки спільноти людей, що займаються спортом, але тимчасово не мають партнерів, щоб реалізувати свої інтереси, надаючи їм інструменти для створення пропозицій і з іншого боку для пошуку серед пропозицій, створених іншими користувачами.

Галузь використання. Заняття спортом у вільний час у випадках, коли у людини є час, але немає партнерів або достатньої кількості партнерів для заняття видами спорту, що потребують партнерів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 88 pages, 25 figures, 3 tables, 3 appendixes, 10 sources.

SPORT, SPORT PLAYGROUND, JOINT SPORTS, PARTNERS, ASP.NET CORE.

The object of the work is the process of finding partners for sports. The subject of the work is sports partner search software. The purpose of the work is to provide software for joint sports that require individual participants in the case of lack of partners.

Materials, methods and technical tools: C# programming language, ASP.NET Core web framework, Model-View-Controller design template, Microsoft SQL Server relational database management system and SQL Server Management Studio administration utilities, Entity Framework Core for data presentation, Visual Studio development environment, HTML and CSS languages, Bootstrap library.

Results. The main result is the developed web application, including in addition to the application itself, the results of its design, developed in the structure of programs, classes of models presented, controllers, interface services to them.

Conclusions. The program is designed to support people who play sports, but temporarily do not have partners to pursue their interests, providing them with tools to create offers and, on the other hand, to search among offers created by other users.

Field of use. Leisure sports when a person has time but does not have partners or enough partners to play sports which require partners.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	7
Вступ	8
1 Аналіз предметної області	9
1.1 Огляд існуючих програмних аналогів	9
1.2 Порівняння існуючих програм	11
1.3 Висновки за розділом 1	12
2 Проєктування програмного забезпечення	14
2.1 Вибір засобів для проєктування і реалізації	14
2.2 Визначення функціональних вимог до програми	15
2.3 Проєктування бази даних	17
2.4 Структура програми	26
2.5 Висновки за розділом 2	27
3 Розробка програми	28
3.1 Алгоритм функціонування програми	28
3.2 Опис реалізованих класів	29
3.3 Висновки за розділом 3	41
4 Експлуатація та тестування програми	42
4.1 Призначення програми	42
4.2 Умови виконання програми	42
4.3 Експлуатація програми	42
4.4 Тестування програми	48
4.5 Висновки за розділом 4	49
Висновки	50
Перелік джерел посилання	51
Додаток А Технічне завдання	52
Додаток Б Текст програми	56
Додаток В Слайди презентації	81

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

EF – Entity Framework;

БД – база даних.

ВСТУП

Командні види спорту, а також індивідуальні, що потребують участі декількох партнерів, як наприклад, теніс, потребують не тільки власної участі, але і партнерів. Сучасний стиль життя, складні графіки роботи людей особливо у великих містах призводять до того, що може виникати бажання зайнятися спортом, але на потрібний часовий проміжок або у потрібному місці партнерів немає. Тоді як раз і потрібно знайти людину або людей, які можливо тимчасово, а можливо в майбутньому і постійно будуть виконувати роль партнерів. У дитинстві знайти друзів пограти в футбол набагато простіше ніж серед людей, які зайняти роботою, сім'ями, хоча в сучасному світі фізична активність для дітей також стає менш звичним явищем.

Міністерство охорони здоров'я та соціальних служб США рекомендує особам у віці від 18 до 64 років щотижня займатися помірною фізичною активністю принаймні 150 хвилин або 75 хвилин інтенсивною фізичною активністю, при цьому згідно з новим дослідженням лише 22,9 % дорослих дотримуються цих рекомендацій [1]-[3]. Це говорить про те, що сучасний рівень фізичної активності людей в багатьох країнах є достатньо низьким, а разом зі складними робочими графіками виходить, що невеликий відсоток людей готовий займатися спортом більш-менш регулярно. До того ж велика частка цих людей вже мають постійних партнерів для цього. Тому пошук таких партнерів для спільних занять стає достатньо важливою проблемою. До того ж підібрати потрібно партнерів під відповідний вид спорту, відповідний рівень у ньому можливо, під інтереси, а ще й найголовніше в потрібному місці і в потрібний час.

Враховуючи описану логіку і актуальність цієї проблеми, мету роботи було визначено як розробити програмне забезпечення пошуку партнерів для спільних занять видами спорту, що потребують декількох учасників, у випадку відсутності партнерів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих програмних аналогів

Застосунок RacketPal (рис. 1.1) [4] може використовуватися для пошук партнерів для гри в теніс, сквош, настільний теніс, бадмінтон.

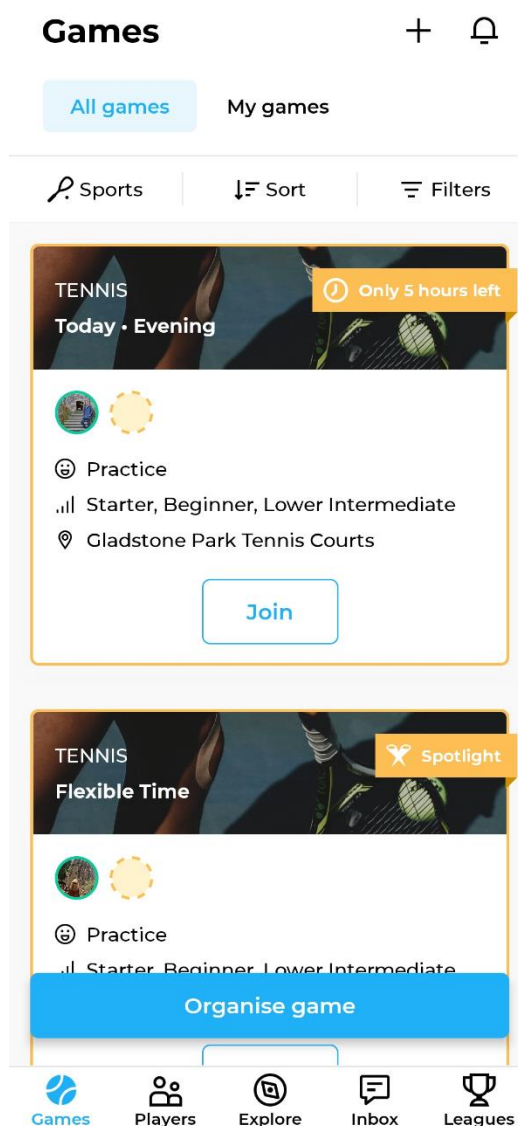


Рисунок 1.1 – Застосунок RacketPal

Програма доступна тільки як застосунок для мобільних пристроїв і потребує обов'язкового встановлення і авторизації для виконання будь-яких дій. Така характеристика в результаті визначає, що не кожен потенційний користувач захоче ознайомитись з можливостями програми особисто, а

значить база користувачів через це може бути меншою ніж це могло би бути потенційно. Для цієї предметної області важливою звичайно ж є велика кількість користувачів, які займаються спортом, застосовуючи цю програму.

Програма надає можливість створювати власні пропозиції, шукати пропозиції інших користувачів, відправляти їм повідомлення, виставляти оцінки партнерам. Процес взаємодії достатньо простий, проте заняття спортом може потребувати певної невеликої сплати. Знову ж таки, це може бути прийнятним для великою розвиненою країни, а в Україні це може бути проблемою і зменшувати базу користувачів.

Варто також зазначити, що програма надає доступ до обмеженої кількості фільтрів пошуку, а для використання повноцінного набору потрібно внести певну сплату.

Sporty (рис. 1.2) [5] є платформою для бронювання фактично тренувань на різних майданчиках та в спортивних клубах.

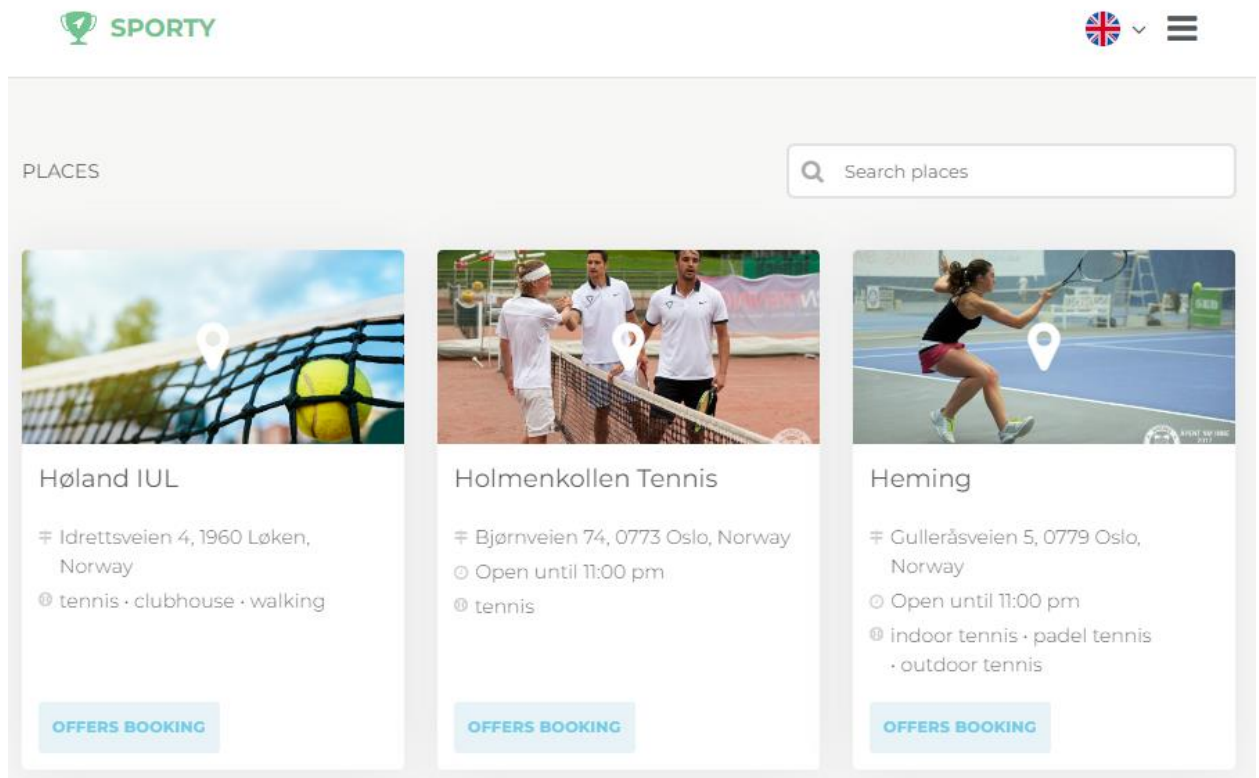


Рисунок 1.2 – Застосунок Sporty

Фактично ця програма не зовсім належить до засобів пошуку пропозицій партнерів, адже тут пошук відбувається місць, де можна займатися спортом і бронювання певного часу таких занять. Така платформа швидше може бути допоміжною під час пошуку партнерів для занять спортом ніж безпосередньо використовуватися для самого пошуку. Проте ця програма доступна як вебзастосунок, можуть оброблятися різні види спорту.

Існують також і інші програми в цій предметній області, наприклад, Matchup – це ще одна подібна платформа, заснована з метою спростити процес організації спортивного матчу, а для виконання пошуку потрібно повідомити програмі про час і місцезнаходження, а також про те, наскільки добре користувач займається спортом, ці вхідні дані використовуються для з'єднання із відповідними гравцями з однаковою доступністю [6]. Однак проблемою цієї програми є те, що вона знаходиться в тестовому стані і використати її на практиці майже неможливо. Ця програма доступна тільки як мобільний застосунок.

1.2 Порівняння існуючих програм

Проведений огляд було використано для порівняння цих програм з дипломною розробкою (табл. 1.1).

Серед розглянутих аналогів найближчою програмою до дипломної розробки є RacketPal. Це фактично на даний момент один активно використовуваний застосунок для пошуку партнерів для занять спортом, хоча підтримуються тільки види спорту, в яких використовуються ракетки. Оскільки це галузь, яка тільки починає динамічно розвиватися, то застосунки тільки починають створюватися, через що часто відрізняються нестабільністю і неможливістю практичного застосування.

На відміну від RacketPal розробку планується виконати як вебзастосунок та без платних можливостей, що повинно сприяти широті

застосування програми. Окрім того для простоти створення наступних пропозицій вирішено зберігати існуючі пропозиції як шаблони.

Таблиця 1.1 – Порівняння програмних аналогів

Критерій	RacketPal	Sporty	Дипломна розробка
Пошук пропозицій партнерів	Так	Ні	Так
Бронювання занять	Так	Так	Так
Платні функції пошуку	Так	Ні	Ні
Створення власних пропозицій	Так	Ні	Так
Платна участь в заняттях спортом	Так	Ні	Ні
Інформування про заняття спортом, що відбулись	Ні	Ні	Так
Створення шаблонів пропозицій	Ні	Ні	Так
Доступ з будь-якого типу пристрою	Ні	Так	Так
Доступ без облікового запису	Ні	Так	Так

1.3 Висновки за розділом 1

Виконаний огляд програмних аналогів продемонстрував, що область програмного забезпечення пошуку партнерів для занять спортом на даний момент активно розвивається, створюються нові застосунки, проте мало рішень таких, які можна вже використовувати.

Завданнями дипломної роботи є:

- аналіз предметної області;
- визначення вимог до програмного забезпечення пошуку партнерів для занять спортом;

- проєктування програмного забезпечення пошуку партнерів для занять спортом;
- реалізація і налагодження програмного забезпечення пошуку партнерів для занять спортом.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір засобів для проєктування і реалізації

Як було зазначено в першому розділі, створена програма повинна бути вебзастосунком. Попередньо було виконано відбір мов програмування для реалізації вебсистем, що включає достатньо широке число мов програмування, а з нього було виділено за основу мову C# [7], оскільки вона дозволяє реалізовувати великі вебсистеми, а потенційно дана система може бути такою, окрім того за рахунок платформи дозволяє об'єднувати в системі рішення для різних платформ.

Проте у випадку мови програмування C# критичним стає вибір платформи, адже це впливає і на те, на скільки багато переваг буде мати мова C#. Платформи .NET Framework і .NET Core [8] є різними версіями платформи, що дозволяє створювати проєкти зокрема і мовою C#, але не тільки. Платформа .NET Core була використана в підсумку (табл. 2.1), тому що вона дозволяє створювати кросплатформні рішення, які легше переносити, розгортати, а також вони не мають пропрієтарних обмежень на права використання.

Таблиця 2.1 – Вибір платформи розробки

Критерій	.NET Framework	.NET Core
Організація кросплатформних системи	Ні	Так
Обмежене розгортання	Ні	Можна розгортати окремі компоненти
Відкрите програмне забезпечення	Частково	Повністю
Ізоляція системи від інших застосунків операційної системи	Ні	Так

На основі обраної платформи використовувався фреймворк для розробки вебзастосунків ASP.NET Core [9]-[10].

2.2 Визначення функціональних вимог до програми

Звичайний користувач програми, який розглядається як партнер, повинен отримувати від програми такі функції (рис. 2.1):

- створення пропозиції про спільне заняття спортом;
- визначення інтересів партнера;
- створення зустрічі;
- створення власного переліку партнерів;
- визначення видів спорту, якими займається партнер;
- висунення зворотної пропозиції про заняття спортом;
- відправлення повідомлення щодо пропозиції про заняття спортом;
- перегляд повідомлень за пропозицією;
- збереження шаблону для створення пропозицій;
- уподобання зустрічі;
- пошук пропозицій;
- пошук спортивних майданчиків;
- перегляд пропозицій за партнером;
- перегляд пропозицій за спортивним майданчиком;
- перегляд зустрічей за спортивним майданчиком;
- перегляд зустрічей за партнером;
- пошук зустрічей;
- створення відгуку про партнера;
- створення відгуку про зустріч;
- перегляд спортивних майданчиків за районом міста;
- перегляд пропозицій за районом міста;
- перегляд пропозицій;
- перегляд зворотної пропозиції;

- перегляд даних спортивного майданчика;
- перегляд зустрічі.



Рисунок 2.1 – Діаграма прецедентів партнера

Адміністратор програми повинен отримувати від програми такі функції:

- додавання, відображення і редагування інтересів;
- додавання, відображення і редагування видів спорту;
- додавання, відображення і редагування речей, що використовуються для заняття спортом;
- додавання, відображення і редагування міст і районів міст;
- додавання, відображення і редагування спортивних майданчиків.

Виходячи з цих вимог, були виділена такі сценарії роботи адміністратора з системою (рис. 2.2):

- робота з доступними інтересами;
- робота з видами спорту;
- робота зі спортивними об'єктами;
- робота з містами;
- робота зі спортивними майданчиками.



Рисунок 2.2 – Діаграма прецедентів адміністратора

2.3 Проектування бази даних

У цьому підрозділі було визначено структуру бази даних (БД) на основі ідеї розроблюваної програми та сценаріїв роботи з нею користувачів.

Сутність Partner представляє партнера для занять спортом. Саме на пошук партнера направлена робота програмного забезпечення, а пошуком партнера відповідно займається інший користувач програми, який фактично також виступає партнером.

Сутність Proposition представляє початкову пропозицію, яку людина, що бажає у вільний час зайнятися спортом, розміщує в програмі, визначаючи таким чином всі власні можливості для такої зустрічі.

Таблиця Proposition має зв'язок з таблицею Partner. Це зв'язок один до багатьох зі сторони таблиці Partner, що можна пояснити тим, що кожен партнер, як користувач програми, може створювати велику кількість пропозицій щодо занять спортом у програмі, проте одне заняття може бути ініційовано тільки одним партнером.

Таблиця Proposition має зв'язок з таблицею Sport. Це зв'язок один до багатьох зі сторони таблиці Sport, що можна пояснити тим, що кожен вид спорту може бути задіяний у великій кількості пропозицій про заняття ним, проте одне заняття може стосуватися тільки одного виду спорту.

Таблиця Proposition має зв'язок з таблицею District. Це зв'язок один до багатьох зі сторони таблиці District, що можна пояснити тим, що у кожному районі міста може відбуватися велика кількість занять спортом, проте одне заняття може відбуватися тільки в одному місці одночасно.

Сутність Interest представляє інтерес партнера, що може детальніше розкривати партнера як особистість і використовуватися під час пошуку партнерів для занять спортом.

Сутність Meeting представляє зустрічі, тобто фактично самі заняття спортом. Вони розглядаються як реалізації пропозицій найчастіше, але передбачається підтримка програмою і більш розширеної інформації. Тобто у випадку, якщо користувач не створював пропозицію, але має бажання розповісти більше про свій власний досвід, то він може це зробити за допомогою зустрічей. Це також корисно у тому випадку, якщо користувач тільки почав працювати з програмою, даних про нього немає і перед тим як створювати пропозицію він хоче розповісти більше про себе фактично.

Таблиця Meeting має зв'язок з таблицею Partner. Це зв'язок один до багатьох зі сторони таблиці Partner, що можна пояснити тим, що кожен партнер може створити велику кількість занять спортом у різний час, проте одне заняття (одна зустріч) може бути створена тільки одним партнером. Наявність даного зв'язку необхідна для ситуацій, коли в програмі не було попередньо створено пропозиції про зустріч. Якщо ж була, то даний зв'язок

залишається незадіяним, тобто фактично з одного боку відношення від 0 до 1, а з іншого – від 0 до багатьох.

Таблиця Meeting має зв'язок з таблицею Proposition. Це зв'язок один до одного, що можна пояснити тим, що кожна пропозиція може мати 0 або 1 запис про зустріч і заняття спортом, відповідно кожна зустріч може мати 0 або 1 відповідний запис про пропозицію, при цьому значення 0 відповідає ситуації, коли вносяться дані зустрічі без створення попередньої пропозиції. За рахунок цієї логіки фактично потенційно підтримується створення невеликої мережі занять спортом, де не тільки користувачі шукають партнерів для занять спортом, але і діляться враженнями просто про звичайні заняття, для яких і не потрібно було шукати нових учасників.

Таблиця Meeting має зв'язок з таблицею Sportlocation. Це зв'язок один до багатьох зі сторони таблиці Sportlocation, що можна пояснити тим, що на кожному спортивному майданчику відбувається велика кількість зустрічей, коли люди займаються спортом, проте одна зустріч може реалізовуватися тільки на одному спортивному майданчику.

Таблиця Meeting має зв'язок з таблицею Sport. Це зв'язок один до багатьох зі сторони таблиці Sport, що можна пояснити тим, що кожен вид спорту звичайно має багато зустрічей в програмі, пов'язаних з ним, проте кожна зустріч стосується тільки одного виду спорту.

Сутність Partnerlist представляє перелік партнерів, який створюється для того, щоб швидше оперувати групою партнерів у командних видах спорту зокрема, або для того, щоб швидше звертатися до сторінок друзів у програмі.

Таблиця Partnerlist має зв'язок з таблицею Partner. Це зв'язок один до багатьох зі сторони таблиці Partner, що можна пояснити тим, що кожен партнер може створювати велику кількість переліків з власними друзями або просто користувачами програми, проте у кожному переліку може бути тільки один власник.

Сутність Sportlocation представляє майданчик для занять спортом.

Таблиця Sportlocation має зв'язок з таблицею Locationtype. Це зв'язок один до багатьох зі сторони таблиці Locationtype, що можна пояснити тим, що кожен тип майданчика може мати велику кількість різних конкретних майданчиків, проте кожен майданчик може відноситися тільки до одного типу.

Сутність Sport представляє вид спорту.

Сутність Locationtype представляє тип майданчика для занять спортом (зокрема торговий центр, спортивний зал, відкритий майданчик).

Сутність Thing представляє речі, які використовуються під час занять спортом. Наприклад, для гри в футбол потрібний м'яч. Для того щоб сформулювати повноцінно пропозицію під час гри в футбол, потрібно визначити, чи м'яч вже є і потрібно тільки знайти партнерів для гри, чи потрібно знайти і м'яч.

Сутність Saveproposition представляє шаблон для побудови пропозицій про заняття спортом. Оскільки таких зустрічей може бути багато, то для того щоб кожного разу все не повторювати заново, зручно використовувати шаблони. Шаблоном може бути власна пропозиція, а також пропозиція іншого користувача. Для того щоб розглядати пропозиції інших користувачів у якості шаблонів, відбувається їх збереження у цій сутності.

Таблиця Saveproposition має зв'язок з таблицею Proposition. Це зв'язок один до багатьох зі сторони таблиці Proposition, що можна пояснити тим, що кожна пропозиція може велику кількість разів зберігатися у вигляді шаблону, але в одному шаблоні може застосовуватися тільки одна пропозиція.

Таблиця Saveproposition має зв'язок з таблицею Partner. Це зв'язок один до багатьох зі сторони таблиці Partner, що можна пояснити тим, що кожен шаблон може бути утворений тільки одним партнером, проте один партнер може мати велику кількість шаблонів.

Сутність PartnerReview представляє відгук про партнера. Виконано розділення відгуків. Окремо розглядаються відгуки про зустріч, що можуть створюватися учасниками цієї зустрічі, і відгуки про партнера. Ці відгуки

важливі для того, щоб мати враження про людину, з якої доведеться займатися спортом, якщо це незнайомець.

Таблиця PartnerReview має зв'язок з таблицею Partner. Це зв'язок один до багатьох зі сторони таблиці Partner, що можна пояснити тим, що кожен відгук може бути створений тільки одним партнером, проте один партнер може створити велику кількість відгуків про інших партнерів. При цьому цей зв'язок повторюється між таблицями двічі через два різні поля, у другому випадку це пояснюється тим, що кожен відгук може стосуватися тільки одного партнера, проте кожен партнер може мати велику кількість відгуків про себе.

Сутність District представляє район міста. Вона необхідна для того, щоб розуміти, в яких околицях можливе заняття спортом для партнера, який створив пропозицію, і які саме майданчики знаходяться в цьому районі.

Таблиця District має зв'язок з таблицею City. Це зв'язок один до багатьох зі сторони таблиці City, що можна пояснити тим, що кожне місто може мати багато районів, проте один район розташовується у конкретному одному місті.

Сутність City представляє саме місто, де знаходяться спортивні майданчики.

Сутність Offer представляє зворотну пропозицію, яка є відповіддю на початкову пропозицію, а її може підтвердити або скасувати партнер, який є автором початкової пропозиції. Тобто таким чином користувач пропонує взяти участь в занятті спортом, ініційованому автором початкової пропозиції. У Proposition користувач говорить, що він хоче грати в футбол, а в Offer інший користувач говорить, що готовий виступити партнером для нього в грі в футбол.

Таблиця Offer має зв'язок з таблицею Proposition. Це зв'язок один до багатьох зі сторони таблиці Proposition, що можна пояснити тим, що кожна пропозиція може мати багато зворотних, проте одна зворотна пропозиція є зворотною тільки стосовно однієї початкової пропозиції.

Таблиця Offer має зв'язок з таблицею Partner. Це зв'язок один до багатьох зі сторони таблиці Partner, що можна пояснити тим, що кожен партнер може створити велику кількість зворотних пропозицій, проте сама зворотна пропозиція може бути створена тільки одним конкретним партнером.

Сутність PropositionMessage представляє повідомлення від потенційного партнера автору початкової пропозиції або навпаки, де вони за необхідності можуть уточнювати деталі майбутньої зустрічі.

Таблиця PropositionMessage має зв'язок з таблицею Proposition. Це зв'язок один до багатьох зі сторони таблиці Proposition, що можна пояснити тим, що кожна пропозиція про заняття спортом може мати велику кількість повідомлень, необхідних для уточнення позицій, проте одне повідомлення може стосуватися тільки однієї пропозиції.

Таблиця PropositionMessage має зв'язок з таблицею Partner. Це зв'язок один до багатьох зі сторони таблиці Partner, що можна пояснити тим, що кожен партнер може створити велику кількість повідомлень, але кожне з них може бути створене тільки одним партнером.

Сутність MeetingLike представляє уподобання зустрічі партнером.

Таблиця MeetingLike має зв'язок з таблицею Partner. Це зв'язок один до багатьох зі сторони таблиці Partner, що можна пояснити тим, що кожен партнер може уподобати велику кількість зустрічей, у яких він брав участь, проте одне уподобання може бути створене тільки одним партнером.

Таблиця MeetingLike має зв'язок з таблицею Meeting. Це зв'язок один до багатьох зі сторони таблиці Meeting, що можна пояснити тим, що кожна зустріч може мати багато уподобань зі сторони різних партнерів, проте одне уподобання може стосуватися тільки тільки до однієї конкретної зустрічі.

Сутність MeetingParticipant представляє участь партнерів у зустрічі.

Таблиця MeetingParticipant має зв'язок з таблицею Partner. Це зв'язок один до багатьох зі сторони таблиці Partner, що можна пояснити тим, що

кожен партнер може брати участь у великій кількості зустрічей, проте одна участь стосується тільки одного партнера.

Таблиця MeetingParticipant має зв'язок з таблицею Meeting. Це зв'язок один до багатьох зі сторони таблиці Meeting, що можна пояснити тим, що кожна зустріч може передбачати багато участей інших партнерів, проте одна така участь стосується конкретною зустрічі і не може визначатися одразу в декількох зустрічах.

Сутність Propositionparticipant визначає участь партнера в пропозиції щодо заняття спортом.

Таблиця Propositionparticipant має зв'язок з таблицею Partner. Це зв'язок один до багатьох зі сторони таблиці Partner, що можна пояснити тим, що кожен партнер може брати участь в різних пропозиціях про заняття спортом, проте одна участь стосується тільки одного конкретного партнера.

Таблиця Propositionparticipant має зв'язок з таблицею Proposition. Це зв'язок один до багатьох зі сторони таблиці Proposition, що можна пояснити тим, що кожна пропозиція може передбачати участь різних партнерів, проте одна участь стосується тільки однієї конкретної пропозиції.

Сутність PartnerSport визначає рівень, який на думку користувача він має у відповідному виді спорту.

Таблиця PartnerSport має зв'язок з таблицею Partner. Це зв'язок один до багатьох зі сторони таблиці Partner, що можна пояснити тим, що кожен партнер може бути зацікавлений у різних видах спорту, проте запис про одне таке зацікавлення стосується тільки одного партнера.

Таблиця PartnerSport має зв'язок з таблицею Sport. Це зв'язок один до багатьох зі сторони таблиці Sport, що можна пояснити тим, що у кожному виді спорту може бути зацікавлено багато партнерів, проте один такий запис про зацікавлення стосується тільки одного виду спорту.

Сутність Offerparticipant визначає участь партнера в зворотній пропозиції, тобто фактично формує участь цілої команди в занятті спортом.

Таким чином користувач може запропонувати для футбольного матчу участь своєї цілої команди або декількох друзів одночасно.

Таблиця Offerparticipant має зв'язок з таблицею Partner. Це зв'язок один до багатьох зі сторони таблиці Partner, що можна пояснити тим, що кожен партнер може брати участь в різних зворотних пропозиціях, проте кожний запис про таку участь стосується тільки одного конкретного партнера.

Таблиця Offerparticipant має зв'язок з таблицею Offer. Це зв'язок один до багатьох зі сторони таблиці Offer, що можна пояснити тим, що кожна зворотна пропозиція може визначати участь різних партнерів у ній, проте кожен запис про таку участь стосується тільки однієї зворотної пропозиції.

Сутність PropositionSportlocation визначає майданчик, на якому може відбуватися заняття спортом у відповідності з пропозицією, винесеною її автором.

Таблиця PropositionSportlocation має зв'язок з таблицею Proposition. Це зв'язок один до багатьох зі сторони таблиці Proposition, що можна пояснити тим, що кожна пропозиція може мати багато записів про те, що вона може відбуватися на якомусь спортивному майданчику, проте кожен такий запис стосується тільки однієї такої пропозиції.

Таблиця PropositionSportlocation має зв'язок з таблицею Sportlocation. Це зв'язок один до багатьох зі сторони таблиці Sportlocation, що можна пояснити тим, що кожен спортивний майданчик може бути запропонований в якості варіанта в різних пропозиціях про заняття спортом, проте кожен такий запис може стосуватися тільки одного конкретного спортивного майданчика.

Таблиця Partnerlistmember визначає партнера, який входить до складу переліку партнерів. Ця таблиця була створена для реалізації зв'язку багато до багатьох між таблицями Partnerlist і Partner, необхідність в ній виникла через те, що один партнер може входити в різні переліки партнерів, але і кожен перелік складається з різних партнерів.

Схема БД показана на рис. 2.3.

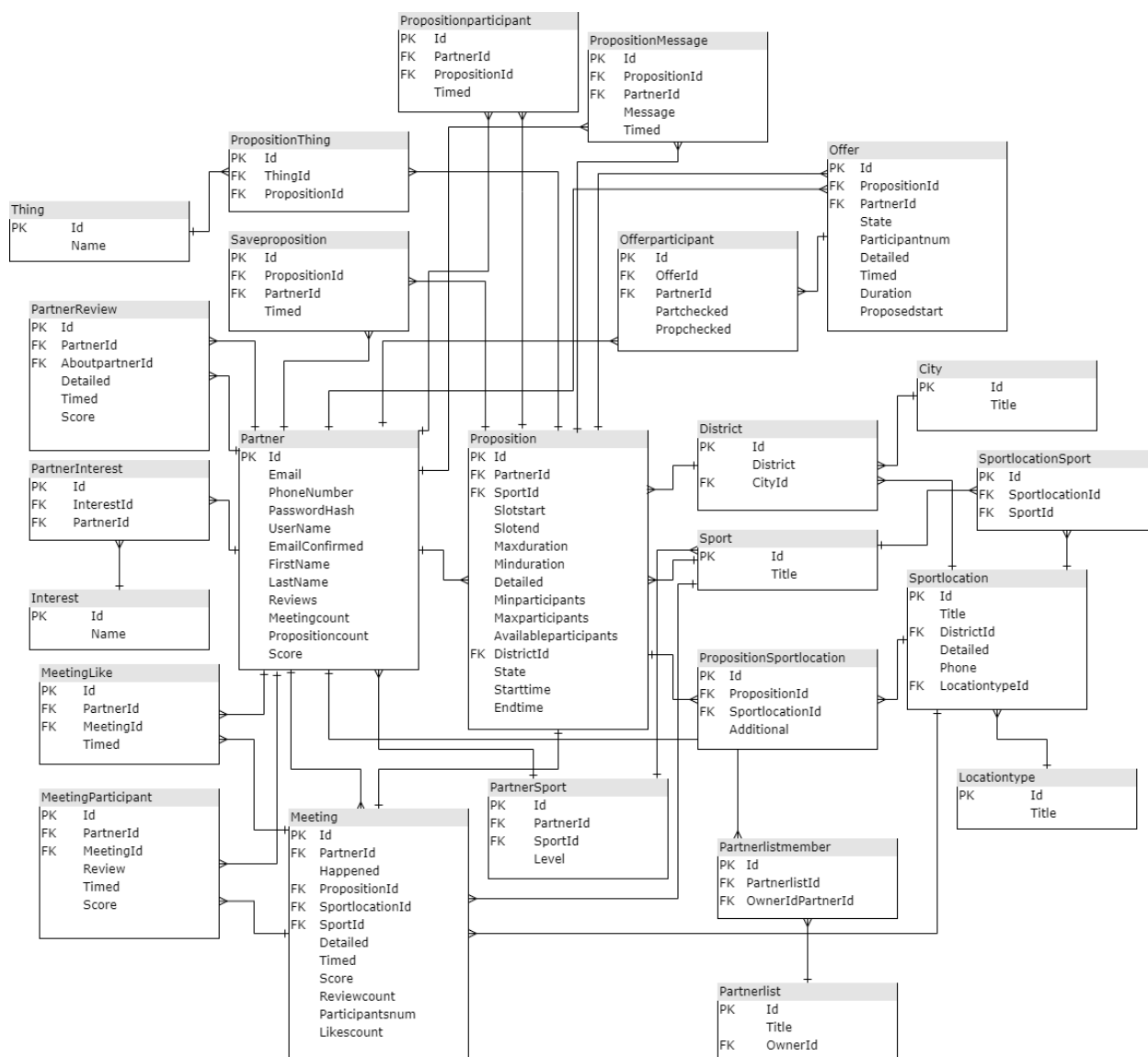


Рисунок 2.3 – Схема БД

Таблиця PropositionThing визначає речі, які необхідні для заняття спортом. Сюди можуть відноситись і ті, які вже має автор пропозиції, і ті, які потрібні від партнерів. Ця таблиця була створена для реалізації зв'язку багато до багатьох між таблицями Proposition і Thing, необхідність в ній виникла через те, що одна пропозиція може включати багато необхідних речей для заняття спортом, але і кожна річ може входити в різні пропозиції.

Таблиця PartnerInterest визначає інтереси партнерів. Ця таблиця була створена для реалізації зв'язку багато до багатьох між таблицями Partner і Interest, необхідність в ній виникла через те, що один партнер може мати

багато інтересів, проте кожен інтерес може характеризувати різних партнерів.

Таблиця SportlocationSport визначає види спорту, якими можна займатися на певному спортивному майданчику. Ця таблиця була створена для реалізації зв'язку багато до багатьох між таблицями Sportlocation і Sport, необхідність в ній виникла через те, що один спортивний майданчик може використовуватися для різних видів спорту, але і кожним видом спорту можна займатися на різних спортивних майданчиках.

Для роботи з БД було використано систему керування реляційними БД Microsoft SQL Server, утиліту для адміністрування SQL Server Management Studio, фреймворк для роботи з даними Entity Framework (EF) Core.

2.4 Структура програми

Структура програми була визначена з використанням шаблону проєктування Model-View-Controller (рис. 2.4).

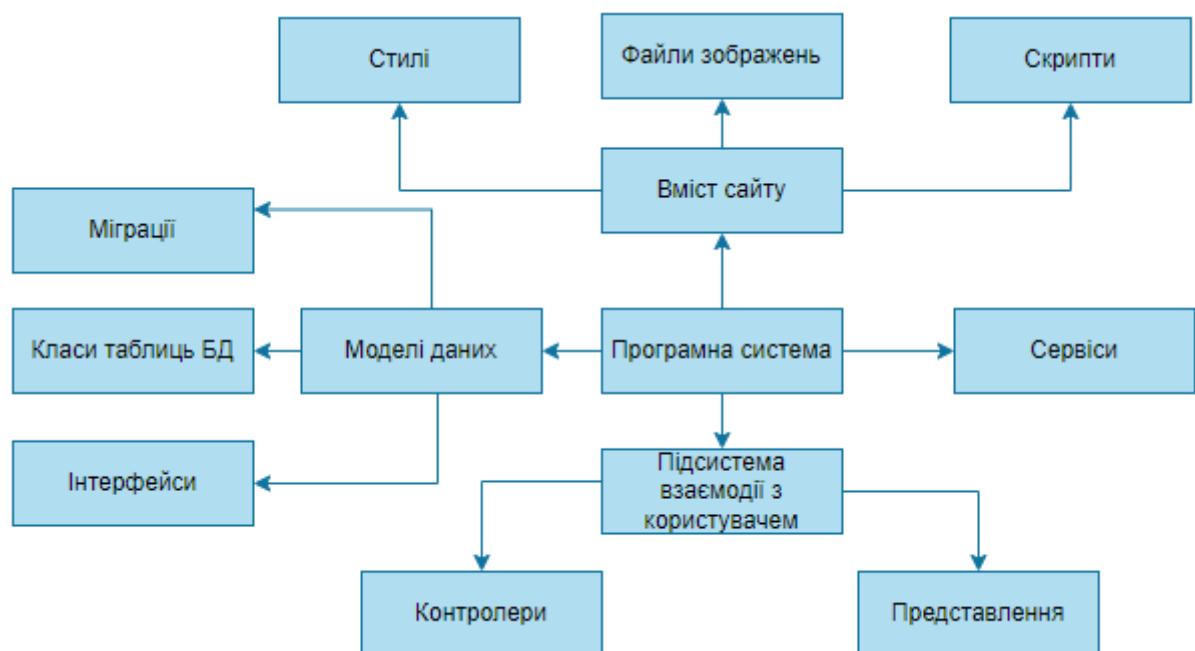


Рисунок 2.4 – Структурна схема програми

До основних складових програми в результаті входять моделі для визначення структури даних, сервіси для взаємодії з цими моделями і надання засобів базового вибирання і внесення даних в БД, контролери для наповнення даними, отриманими через сервіси, представлень у формі, готовій для відображення користувачу.

2.5 Висновки за розділом 2

Для реалізації програми було обрано мову програмування C#, вебфреймворк ASP.NET Core, систему керування реляційними базами даних Microsoft SQL Server, утиліту для адміністрування SQL Server Management Studio, фреймворк для роботи з даними EF Core, а також використано інтегроване середовище розробки Visual Studio, мови HTML і CSS, бібліотека Bootstrap. Сценарії роботи користувачів описані за допомогою діаграм прецедентів. Визначено сутності предметної області і представлено відповідні таблиці БД і зв'язки між ними. На основі шаблону проєктування Model-View-Controller визначена структура програми.

3 РОЗРОБКА ПРОГРАМИ

3.1 Алгоритм функціонування програми

Для опису алгоритму функціонування програми було використано основний порядок взаємодії звичайного користувача з програмою (рис. 3.1). Кожна з цих дій передбачає певну послідовність виконання у свою чергу.

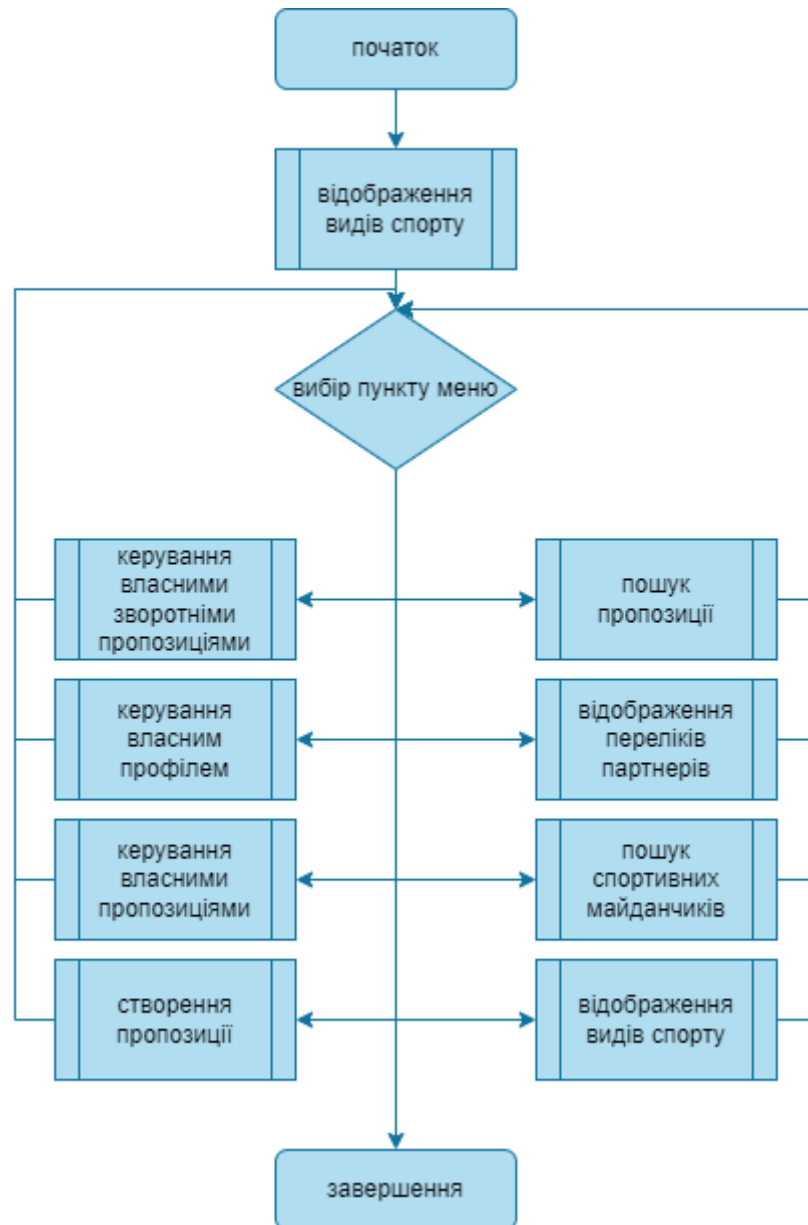


Рисунок 3.1 – Схема функціонування програми

3.2 Опис реалізованих класів

На основі використання EF розроблені класи моделей стали структурою БД.

Модель пропозиції була реалізована класом `Proposition`, до структури якого належать поля:

- `int Id` – ідентифікатор пропозиції;
- `int PartnerId` – ідентифікатор партнера, що створив пропозицію;
- `int SportId` – ідентифікатор виду спорту, до якого належить пропозиція;
- `DateTime Slotstart` – початок вільного часу;
- `DateTime Slotend` – завершення вільного часу;
- `TimeSpan Maxduration` – максимальна тривалість заняття;
- `TimeSpan Minduration` – мінімальна тривалість заняття;
- `string Detailed` – деталізований опис пропозиції;
- `short Minparticipants` – мінімальна необхідна кількість учасників;
- `short Maxparticipants` – максимальна необхідна кількість учасників;
- `short Availableparticipants` – вже наявна кількість учасників;
- `int DistrictId` – район міста, де є можливість провести заняття;
- `StateType State` – стан пропозиції;
- `DateTime Starttime` – фактичний час початку заняття;
- `DateTime Endtime` – фактичний час завершення заняття;
- `virtual Partner Partner` – віртуальне поле, яке визначає повністю партнера, який створив пропозицію;
- `virtual Sport Sport` – віртуальне поле, яке визначає повністю вид спорту;
- `virtual District District` – віртуальне поле, яке визначає повністю район міста;

- virtual ICollection<PropositionSportlocation> PropositionSportlocations – віртуальне поле, яке визначає запропоновані майданчики, на яких може відбуватися зустріч;
- virtual Meeting Meeting – віртуальне поле, яке визначає повністю зустріч, яка була проведена за пропозицією;
- virtual ICollection<Offer> Offers – віртуальне поле, яке визначає повністю зустрічні пропозиції від учасників про зустріч, тобто про їх участь у запланованому занятті;
- virtual ICollection<Thing> Things – віртуальне поле, яке визначає повністю речі, які потрібні для проведення заняття;
- virtual ICollection<PropositionMessage> PropositionMessages – віртуальне поле, яке визначає повністю повідомлення стосовно цієї пропозиції.

Модель зворотної пропозиції була реалізована класом Offer, до структури якого належать поля:

- int Id – ідентифікатор зворотної пропозиції;
- int PropositionId – ідентифікатор пропозиції, на яку створено цю зворотну пропозицію;
- int PartnerId – ідентифікатор партнера, що створив зворотну пропозицію;
- DateTime Timed – час створення пропозиції;
- DateTime Proposedstart – запропонований час початку;
- TimeSpan Duration – запропонована тривалість;
- string Detailed – текстовий опис зворотної пропозиції;
- short Participantnum – кількість учасників, яка є в групі, що готова долучитися;
- OfferStateType State – стан зворотної пропозиції;
- virtual Proposition Proposition – віртуальне поле, яке визначає повністю пропозицію, на яку створено цю зворотну пропозицію;

– virtual Partner Partner – віртуальне поле, яке визначає повністю партнера, який створив зворотну пропозицію.

Модель відгуку про партнера була реалізована класом PartnerReview, до структури якого належать поля:

- int Id – ідентифікатор відгуку про партнера;
- int PartnerId – ідентифікатор автора відгуку;
- int AboutpartnerId – ідентифікатор партнера, про якого створено відгук;
- DateTime Timed – час створення відгуку;
- string Detailed – текст відгуку;
- short Score – виставлена оцінка;
- virtual Partner Partner – віртуальне поле автора відгуку для повного доступу до даних про нього;
- virtual Partner Aboutpartner – віртуальне поле партнера, про якого створено відгук, для повного доступу до даних про нього.

Модель повідомлення до пропозиції була реалізована класом PropositionMessage, до структури якого належать поля:

- int Id – ідентифікатор повідомлення щодо пропозиції;
- int PartnerId – ідентифікатор автора повідомлення;
- int PropositionId – ідентифікатор пропозиції, якої стосується повідомлення;
- DateTime Timed – час створення повідомлення;
- string Message – текст повідомлення;
- virtual Partner Partner – віртуальне поле автора повідомлення;
- virtual Proposition Proposition – віртуальне поле пропозиції, якої стосується повідомлення.

Модель зустрічі для заняття спортом була реалізована класом Meeting, до структури якого належать поля:

- int Id – ідентифікатор зустрічі;
- int PartnerId – ідентифікатор партнера, що створив зустріч;

- int SportId – ідентифікатор виду спорту, до якого належить зустріч;
- DateTime Happened – час зустрічі;
- int PropositionId – пропозиція, до якої належить ця зустріч (якщо пропозиція попередньо створювалась);
- int SportlocationId – ідентифікатор майданчика;
- string Detailed – опис зустрічі;
- DateTime Timed – час створення зустрічі;
- float Score – середня оцінка зустрічі;
- short Participantsnum – кількість учасників;
- int Reviewcount – кількість відгуків від учасників;
- int Likescount – кількість уподобань зустрічі;
- virtual Partner Partner – партнер, який створив зустріч;
- virtual Sport Sport – вид спорту;
- virtual Sportlocation Sportlocation – майданчик;
- virtual Proposition Proposition – пропозиція, за якою відбулась зустріч
- virtual ICollection<MeetingParticipant> MeetingParticipants – учасники зустрічі;
- virtual ICollection< MeetingLike> MeetingLikes – уподобання зустрічі.

Модель участі партнера в зустрічі була реалізована класом MeetingParticipant, до структури якого належать поля:

- int Id – ідентифікатор участі партнера в зустрічі;
- int PartnerId – ідентифікатор партнера;
- int MeetingId – ідентифікатор зустрічі;
- DateTime Timed – час внесення;
- string Review – текст відгуку;
- int Score – оцінка зустрічі;
- virtual Partner Partner – віртуальне поле партнера;
- virtual Meeting Meeting – віртуальне поле зустрічі.

Модель партнера була реалізована класом `Partner` наслідуванням класу `IdentityUser` ASP.NET Core, серед полів якого використовувались зокрема наступні:

- поле `Id` – ідентифікатор користувача;
- поле `Email` – електронна пошта, що використовується для реєстрації спочатку і авторизації на її основі;

- поле `PhoneNumber` – номер телефону;
- поле `PasswordHash` – геш паролю;
- поле `UserName` – ім'я користувача;
- поле `EmailConfirmed` – підтверджена електронна пошта чи ні.

Також окремо визначено додаткові поля:

- `int Reviews` – кількість відгуків про даного партнера;
- `int Propositioncount` – кількість пропозицій, створених партнером;
- `int Meetingcount` – кількість зустрічей партнера;
- `float Score` – середня оцінка, виставлена даному партнеру;
- `virtual ICollection<Saveproposition> Savepropositions` – перелік збережених шаблонів пропозицій партнера;
- `virtual ICollection<PartnerSport> PartnerSports` – перелік видів спорту, якими займається цей партнер;
- `virtual ICollection<Proposition> Propositions` – перелік пропозицій, які створив партнер;
- `virtual ICollection<Meeting> Meetings` – перелік зустрічей для заняття спортом, які створив партнер;
- `virtual ICollection<MeetingLike> MeetingLikes` – перелік уподобань зустрічей партнером;
- `virtual ICollection<MeetingParticipant> MeetingParticipants` – перелік участі партнера в зустрічах;
- `virtual ICollection<Propositionparticipant> Propositionparticipants` – перелік участі партнера в пропозиціях;

- virtual ICollection<Offerparticipant> Offerparticipants – перелік участі партнера в зворотних пропозиціях;
- virtual ICollection<Offer> Offers – перелік зворотних пропозицій, створених партнером;
- virtual ICollection<PropositionMessage> PropositionMessages – перелік повідомлень від цього партнера;
- virtual ICollection<Interest> Interests – перелік інтересів партнера;
- virtual ICollection<PartnerReview> PartnerReviews – перелік відгуків цього партнера про інших партнерів;
- virtual ICollection<PartnerReview> AboutPartnerReviews – перелік відгуків про цього партнера;
- virtual ICollection<Partnerlist> Partnerlists – перелік з переліками партнерів, збереженими цим партнером.

Модель спортивного майданчика в пропозиції була реалізована класом PropositionSportlocation, до структури якого належать поля:

- int Id – ідентифікатор участі партнера в зустрічі;
- int PropositionId – ідентифікатор пропозиції;
- int SportlocationId – ідентифікатор спортивного майданчика;
- bool Additional – основний майданчик або в якості пропозиції;
- virtual Proposition Proposition – пропозиція;
- virtual Sportlocation Sportlocation – спортивний майданчик.

Модель інтересу була реалізована класом Interest, до структури якого належать поля:

- int Id – ідентифікатор інтересу;
- string Name – назва.

Модель переліку партнерів була реалізована класом Partnerlist, до структури якого належать поля:

- int Id – ідентифікатор переліку партнерів;
- string Title – назва переліку;
- int OwnerId – ідентифікатор власника;

– virtual Partner Owner – віртуальне поле з повними даними про партнера, який є власником цього переліку;

– virtual ICollection<Partner> Partnerlistmembers – партнери, що входять до переліку партнерів.

Модель спортивного майданчика була реалізована класом Sportlocation, до структури якого належать поля:

- int Id – ідентифікатор участі партнера в зустрічі;
- int DistrictId – ідентифікатор партнера;
- int LocationtypeId – ідентифікатор зустрічі;
- string Title – назва спортивного майданчика;
- string Detailed – текст відгуку;
- string Phone – оцінка зустрічі;
- virtual District District – віртуальне поле партнера;
- virtual Locationtype Locationtype – віртуальне поле зустрічі.
- virtual ICollection <PropositionSportlocation> PropositionSportlocations – перелік пропозицій, які можуть здійснюватися на цьому майданчику;
- virtual ICollection <Meeting> Meetings – перелік зустрічей на цьому майданчику;
- virtual ICollection <Sport> Sports – перелік видів спорту, якими можна займатися на цьому майданчику.

Модель виду спорту була реалізована класом Sport, до структури якого належать поля:

- int Id – ідентифікатор виду спорту;
- string Title – назва виду спорту;
- virtual ICollection <Proposition> Propositions – перелік пропозицій, які можуть здійснюватися на цьому майданчику;
- virtual ICollection <Meeting> Meetings – перелік зустрічей на цьому майданчику;
- virtual ICollection <Sportlocation> Sportlocations – перелік спортивних майданчиків;

– virtual ICollection <PartnerSport> PartnerSports – перелік партнерів, які займаються цим видом спорту.

Модель рівня заняття партнера видом спорту була реалізована класом PartnerSport, до структури якого належать поля:

- int Id – ідентифікатор;
- int PartnerId – ідентифікатор партнера;
- int SportId – ідентифікатор виду спорту;
- short Level – рівень, на якому за власною оцінкою знаходиться в цьому виді спорту партнер;
- virtual Partner Partner – віртуальне поле партнера;
- virtual Sport Sport – віртуальне поле виду спорту.

Модель типу спортивного майданчика була реалізована класом Locationtype, до структури якого належать поля:

- int Id – ідентифікатор типу спортивного майданчика;
- string Title – назва типу спортивного майданчика;
- virtual ICollection <Sportlocation> Sportlocations – перелік спортивних майданчиків цього типу.

Модель учасників заняття спортом за пропозицією була реалізована класом Propositionparticipant, до структури якого належать поля:

- int Id – ідентифікатор;
- int PartnerId – ідентифікатор партнера;
- int PropositionId – ідентифікатор пропозиції;
- DateTime Timed – час внесення партнера в пропозицію;
- virtual Partner Partner – віртуальне поле партнера;
- virtual Proposition Proposition – віртуальне поле пропозиції.

Модель учасників зворотної пропозиції була реалізована класом Offerparticipant, до структури якого належать поля:

- int Id – ідентифікатор;
- int PartnerId – ідентифікатор партнера;
- int OfferId – ідентифікатор зворотної пропозиції;

- bool Partchecked – підтверджено самим цим партнером;
- bool Propchecked – підтверджено автором пропозиції;
- virtual Partner Partner – віртуальне поле партнера;
- virtual Offer Offer – віртуальне поле зворотної пропозиції.

Модель речі була реалізована класом Thing, до структури якого належать поля:

- int Id – ідентифікатор речі;
- string Name – назва речі.
- virtual ICollection <Proposition> Propositions – перелік пропозицій.

Модель збереженої в якості шаблону пропозиції була реалізована класом Saveproposition, до структури якого належать поля:

- int Id – ідентифікатор шаблону;
- int PartnerId – ідентифікатор партнера;
- int PropositionId – ідентифікатор пропозиції, з якої створено шаблон;
- DateTime Timed – час збереження пропозиції як шаблону;
- virtual Partner Partner – віртуальне поле партнера;
- virtual Proposition Proposition – віртуальне поле пропозиції, з якої

створено шаблон.

Модель району міста була реалізована класом District, до структури якого належать поля:

- int Id – ідентифікатор шаблону;
- string Title – назва типу спортивного майданчика;
- int CityId – ідентифікатор партнера;
- virtual City City – віртуальне поле партнера;
- virtual ICollection <Sportlocation> Sportlocations – перелік спортивних майданчиків у цьому районі міста;
- virtual ICollection <Proposition> Propositions – перелік пропозицій

про заняття спортом, які реалізуються в цьому районі.

Модель міста була реалізована класом City, до структури якого належать поля:

- int Id – ідентифікатор міста;
- string Title – назва міста;
- virtual ICollection <District> Districts – перелік районів міста.

Модель уподобання зустрічі була реалізована класом MeetingLike, до структури якого належать поля:

- int Id – ідентифікатор уподобання;
- int PartnerId – ідентифікатор партнера, який вподобав цю зустріч;
- int MeetingId – ідентифікатор зустрічі, яку було уподобано;
- DateTime Timed – час уподобання;
- virtual Partner Partner – віртуальне поле партнера;
- virtual Meeting Meeting – віртуальне поле зустрічі.

Під дані класи моделей були реалізовані класи сервісів. Вони побудовані приблизно за однаковим принципом з внесенням функціональних можливостей, які потребують безпосередніх запитів даних, тому нижче буде описано декілька з них.

Клас сервісу пропозиції PropositionService забезпечує роботу таких методів.

Метод Proposition GetById (int proposId) надає повні дані пропозиції з номером proposId.

Метод Proposition GetByIdForOwner (int proposId) надає повні дані пропозиції для її власника, включаючи повідомлення від інших партнерів, усі зворотні пропозиції.

Метод IEnumerable<Proposition> GetAll() формує перелік всіх пропозицій.

Метод IEnumerable<Proposition> GetPropositionsByPartner(int partnerId, StateType? state) формує перелік всіх пропозицій за партнером partnerId, який їх створив, та станом state, який може не задаватися.

Метод IEnumerable<Offer> GetOffersByProposition (int propositionId) формує перелік всіх зворотних пропозицій за пропозицією з номером propositionId.

Метод `Proposition GetLastProposition(int? partnerId)` повертає останню пропозицію загалом (якщо не встановлено значення параметра `partnerId`) або у заданого партнера `partnerId`.

Метод `IEnumerable<Proposition> GetPropositionsByParameters (DateTime start, DateTime end, int sportId, TimeSpan minduration, TimeSpan maxduration, short minparticip, int maxparticip, int districtId, float minscore, float maxscore)` реалізує пошук пропозицій за параметрами: `start` – початок вільного часу, `end` – закінчення вільного часу, `sportId` – вид спорту, `minduration` – мінімальна тривалість, `maxduration` – максимальна тривалість, `minparticip` – мінімальна кількість учасників, `maxparticip` – максимальна кількість учасників, `districtId` – номер району, `minscore` – мінімальна прийнятна оцінка, `maxscore` – максимальна оцінка.

Метод `Task Add(Proposition propos)` реалізує додавання до БД пропозиції.

Метод `Task Update(Proposition propos)` реалізує оновлення в БД пропозиції.

Метод `IEnumerable<Proposition> GetPropositionsBySportlocation (int sportlocationId)` повертає список активних пропозицій за заданим майданчиком `sportlocationId`.

Клас сервісу зустрічі `MeetingService` забезпечує роботу таких методів.

Метод `Meeting GetById (int meetingId)` надає повні дані зустрічі з номером `meetingId`.

Метод `IEnumerable<Meeting> GetAll()` формує перелік всіх зустрічей.

Метод `IEnumerable<Meeting> GetMeetingsByPartner(int partnerId)` перелік всіх зустрічей за партнером `partnerId`, який їх створив.

Метод `DateTime GetFirstMeetingTime()` повертає час найдавнішої зустрічі.

Метод `DateTime GetLastMeetingTime()` повертає час останньої зустрічі.

Метод `Meeting GetLastMeeting(int? partnerId)` повертає останню внесену зустріч.

Метод `Task Add(Meeting meet)` реалізує додавання до БД зустрічі.

Метод `Task Update(Meeting meet)` реалізує оновлення в БД зустрічі.

Метод `IEnumerable<Meeting> GetMeetingsBySportlocation (int sportlocationId)` повертає список зустрічей за заданим майданчиком `sportlocationId`.

Метод `IEnumerable<Meeting> GetMeetingsWithPartner (int partnerId)` повертає список зустрічей, в яких брав участь учасник з номером `partnerId`.

Метод `IEnumerable<Meeting> GetMeetingsByParameters (DateTime start, DateTime end, int sportlocationId, float minscore, float maxscore, int sportId)` реалізує пошук зустрічей за параметрами: `start` – початок часового інтервалу, `end` – закінчення часового інтервалу, `sportId` – вид спорту, `minscore` – мінімальна прийнятна оцінка, `maxscore` – максимальна оцінка, `sportlocationId` – спортивний майданчик, де відбувалась зустріч.

Метод `Task CalculateMeeting(int meetingId)` виконує завдання переобчислення середнього балу зустрічі та кількості відгуків і уподобань цієї зустрічі.

У свою чергу контролери реалізують формування даних під конкретне представлення вебзастосунку. Під кожен сторінку, що відповідає фактично базовому вираженню кожної функції програми, створено відповідний метод відповідного представлення, а самі представлення організовані у відповідності зі структурою функцій, що в основному відповідає сутностям, представленим в моделях за виключенням доповміжних таких класів.

Клас контролера `MeetingController` допомагає заповнювати даними представлення вебзастосунку.

Метод `Task<IActionResult> Index(int meetingId)` підтримує представлення перегляду зустрічі з заданим номером `meetingId`.

Метод `Task<IActionResult> Create([Bind("Id, PartnerId, SportId, Happened, PropositionId, SportlocationId, Detailed, Participantsnum")] Meeting`

meeting) підтримує представлення створення зустрічі з наповненням даними доступних видів спорту, спортивних майданчиків та пропозицій авторизованого партнера.

Метод `Task<ActionResult> Edit(int meetingId, [Bind("Id, PartnerId, SportId, Happened, PropositionId, SportlocationId, Detailed, Participantsnum")] Meeting meeting)` підтримує представлення редагування зустрічі з наповненням даними доступних видів спорту, спортивних майданчиків та пропозицій авторизованого партнера.

Метод `ActionResult SportlocationMeetings(int sportlocationId)` підтримує створення представлення для перегляду зустрічей за спортивним майданчиком `sportlocationId`.

Метод `ActionResult GetMeetingsWithPartner (int partnerId)` підтримує представлення перегляду зустрічей партнера `partnerId`.

Метод `ActionResult GetMeetingsByPartner (int partnerId)` підтримує представлення перегляду зустрічей, створених партнером `partnerId`.

Метод `ActionResult SearchMeetings(DateTime? start, DateTime? end, int? sportlocationId, float? minscore, float? maxscore, int? sportId)` підтримує представлення пошуку зустрічей, що починаються в момент не раніше `start`, а закінчуються в момент не пізніше `end`, відбуваються на майданчику `sportlocationId`, маючи оцінки не менше `minscore` та не більше `maxscore`, а також передбачали заняття за видом спорту `sportId`.

3.3 Висновки за розділом 3

У третьому розділі реалізовано вебзастосунок, включаючи зокрема класи моделей, класи інтерфейсів, класи сервісів, класи представлень, а також самі представлення під усі функції, що визначалися функціональними вимогами, описано послідовність роботи програми щодо основних користувацьких функцій.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програма призначена для підтримки спільноти людей, що займаються спортом, але тимчасово не мають партнерів, щоб реалізувати свої інтереси, надаючи їм інструменти для створення пропозицій і з іншого боку для пошуку серед пропозицій, створених іншими користувачами.

4.2 Умови виконання програми

Для забезпечення підтримки роботи вебсервера та сервера БД необхідно надати апаратне забезпечення, що включає мінімально процесор, що має не менше ніж 4 ядра з тактовою частотою не менше 2,1 ГГц, підтриманий оперативною пам'яттю обсягом не менше 8 Гб та жорстким диском з доступними 10 Гб простору на ньому.

4.3 Експлуатація програми

Експлуатація програми зі сторони звичайного користувача є достатньо інтуїтивно зрозумілим процес, проте через велику кількість посилань і функцій потрібно вмісти в них орієнтуватися.

У той же час процес роботи адміністратора системи ще простіший, адже у відповідності зі сценаріями його роботи під кожний сценарій виділено відповідний розділ і з нього адміністратор отримує перелік відповідних даних і може редагувати їх, створювати нові записи. Адміністратор виконує глобальну роботу, визначаючи дані, які є базовими для роботи звичайного користувача.

Звичайний користувач, або партнер, відкриваючи вебсайт, потрапляє на домашню сторінку, яка для ознайомлення містить перелік наявних видів спорту (рис. 4.1). Відкриваючи один з них, можна перейти до сторінки

пошуку пропозицій, але під час пошуку вже буде визначено одразу вид спорту.

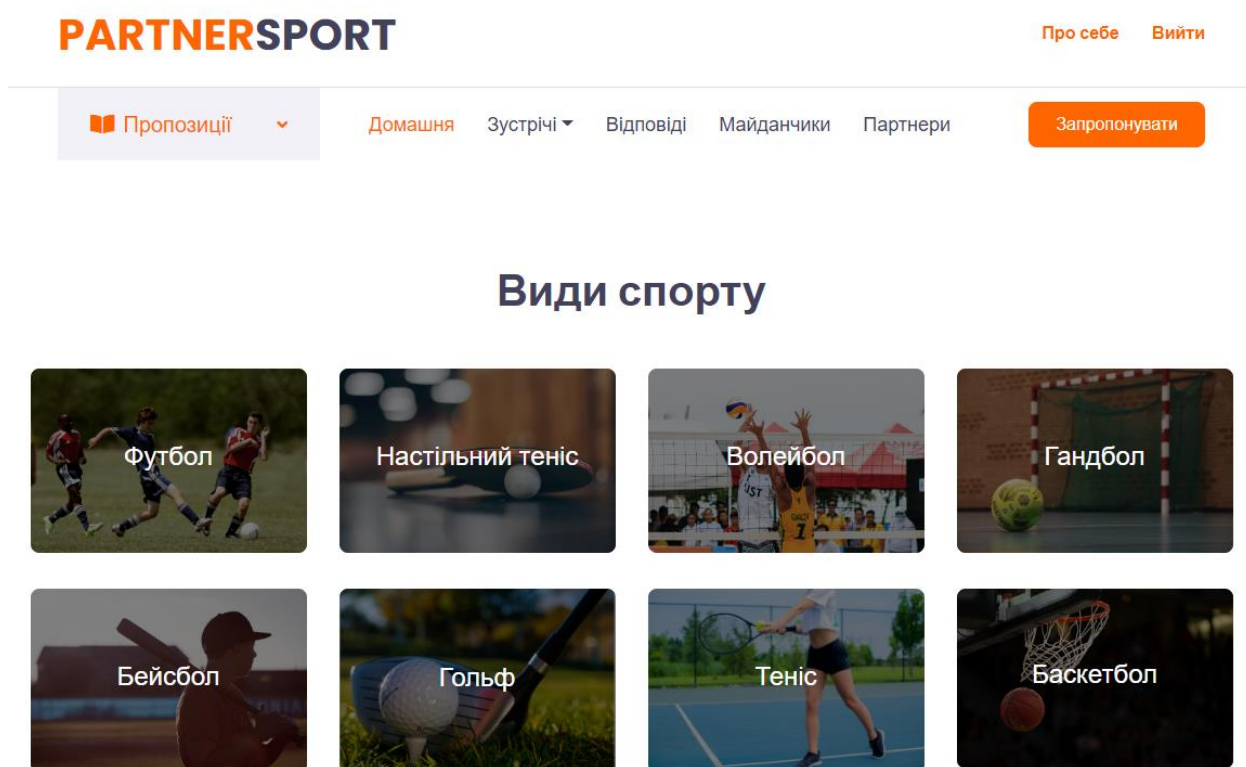


Рисунок 4.1 – Перегляд видів спорту

Основним пунктом меню, винесеним окремо, є «Пропозиції», адже саме робота з пропозиціями є базовою необхідністю, заради якої програма створювалась. Даний пункт містить два підпункти, які дозволяють або переглянути перелік всіх своїх власних пропозицій, або перейти до пошуку пропозицій.

Пошук пропозицій (рис. 4.2) організований на окремій сторінці, яка зверху містить форму для визначення параметрів пошуку, що включають дату майбутнього заняття, час, у який найраніше хоче почати користувач займатися, найпізніший час заняття, мінімальна і максимальна тривалості заняття, вид спорту, місто і район, де буде відбуватися заняття, мінімальну і максимальну оцінку на основі відгуків про організатора заняття, мінімальну та максимальну кількість учасників.

Шукати пропозиції

Дата 14.06.2022 Розпочати з 06:00 Закінчити по 22:00 Мінімальна тривалість 01:00 Максимальна тривалість 02:00 Вид спорту Теніс	Місто Запоріжжя Район Шевченківський Мінімальна оцінка організатора 2 Максимальна оцінка організатора 5 Мінімально учасників 2 Максимально учасників 4 Показати
--	--



👤 2 ⌚ 14.06.22 07:00–09:00

Організатор: Шеховцов Антон

Взяти участь



👤 2–4 ⌚ 14.06.22 14:00–16:00

Організатор: Ляшенко Віктор

Взяти участь



👤 2 ⌚ 14.06.22 17:00–20:00

Організатор: Петренко Павло

Взяти участь

Рисунок 4.2 – Пошук пропозицій

Визначивши всі потрібні параметри, потрібно натиснути на кнопку «Шукати», нижче форми будуть відображатися всі знайдені пропозиції. Кожна пропозиція включає фотографію (дозволяє перейти до сторінки

пропозиції), ім'я організатора (дозволяє перейти до сторінки профілю з інтересами, відгуками та видами спорту, якими цікавиться). Кнопка «Взяти участь» дозволяє перейти одразу до створення зворотної пропозиції. За пропозицією також показується кількість учасників максимальна і мінімальна та вільний проміжок часу для заняття у організатора.

Сторінка пропозиції містить всю інформацію про неї (рис. 4.3).

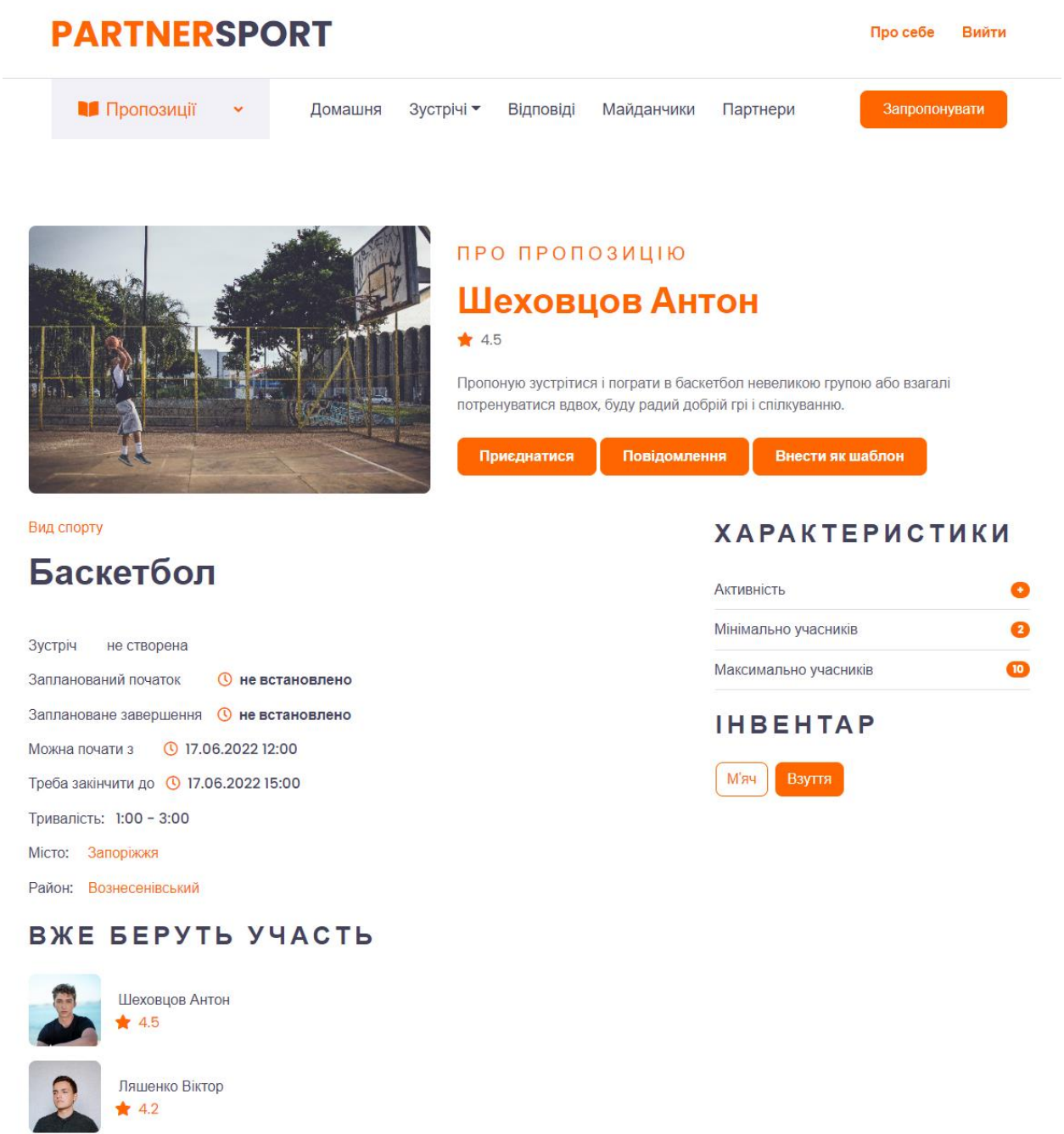


Рисунок 4.3 – Перегляд пропозиції про заняття спортом

Інформація про заняття включає дані організатора з можливістю перейти до його профілю, а також визначеною його середньою оцінкою, опис зустрічі, фотографію для ілюстрації зустрічі (у разі відсутності використовується фотографія профілю організатора).

Кнопка «Приєднатися» дозволяє створити зворотну пропозицію.

Кнопка «Повідомлення» дозволяє переглянути або відправити повідомлення в спілкування між користувачами щодо цієї пропозиції.

Кнопка «Внести як шаблон» дозволяє під час створення пропозиції використовувати цю як шаблон.

Сторінка також містить перелік уже наявних учасників, перелік наявних (відображається на світлому фоні) і потрібних (відображається на помаранчевому фоні) речей (або інвентаря) для заняття.

Також на сторінці нижче знаходиться перелік спортивних майданчиків, на яких пропонує провести заняття організатор.

На всіх сторінках відображається зверху кнопка «Запропонувати», яка для авторизованого користувача дозволяє перейти до створення власної початкової пропозиції. Якщо користувач ще неавторизований, то його у випадку таких дій буде перенаправлено до авторизації.

Також сторінки містять пункт меню «Зустрічі», що дозволяє відобразити власні зустрічі або шукати зустрічі. Процес роботи аналогічний по суті як і для пропозицій. Варто також зазначити, що на сторінка пропозиції відображається посилання на зустріч, якщо вона була створена за цією пропозицією. Але це відбувається тоді, коли пропозиція стала неактивною (відображається мінус в полі активності). Так само як і в пропозиціях, при відкритті власних зустрічей надається їх перелік з доступною кнопкою редагування. Але в пропозиціях також надається кнопка відповіді, що дозволяє переглянути перелік поданих зворотних пропозицій, в якому відображаються кнопки для прийняття та скасування зворотної пропозиції.

4.4 Тестування програми

Тестування програми реалізовувалось спочатку під час реалізації програми і дозволяло одразу виявляти основні проблеми і змінювати відповідним чином код. Для цього використовувався набір модульних тестів. Після завершення реалізації був створений чеклист з переліком функцій програми і було перевірено коректність їх виконання (табл. 4.1). Перевірка була успішною.

Таблиця 4.1 – Чеклист тестування програми

Перевірка	Результат
Створення і редагування пропозиції про заняття спортом	+ / +
Визначення інтересів партнера	+
Створення і редагування зустрічі	+
Створення власного переліку партнерів	+
Визначення видів спорту, якими займається партнер	+
Додавання і редагування зворотної пропозиції про заняття спортом	+
Відправлення повідомлення щодо пропозиції про заняття спортом	+
Перегляд повідомлень за пропозицією	+
Збереження шаблону для створення пропозицій	+
Уподобання зустрічі	+
Пошук пропозицій	+
Пошук спортивних майданчиків	+
Перегляд пропозицій за партнером	+
Перегляд пропозицій за спортивним майданчиком	+
Перегляд зустрічей за спортивним майданчиком	+
Перегляд зустрічей за партнером	+

Продовження таблиці 4.1

Перевірка	Результат
Пошук зустрічей	+
Створення відгуку про партнера	+
Створення відгуку про зустріч	+
Перегляд відгуків про партнера	+
Перегляд відгуків про зустріч	+
Перегляд спортивних майданчиків за районом міста	+
Перегляд пропозицій за районом міста	+
Перегляд пропозиції	+
Перегляд зворотної пропозиції	+
Перегляд даних спортивного майданчика	+
Перегляд зустрічі	+
Додавання, відображення і редагування інтересів	+ / + / +
Додавання, відображення і редагування видів спорту	+ / + / +
Додавання, відображення і редагування інвентаря	+ / + / +
Додавання, відображення і редагування міст	+ / + / +
Додавання, відображення і редагування районів міст	+ / + / +
Додавання, відображення і редагування спортивних майданчиків	+ / + / +

4.5 Висновки за розділом 4

Представлено основні інтерфейси роботи користувачів з програмою. Виконано тестування програми з перевіркою коректності реалізації всіх запланованих функцій, що підтвердило коректність.

ВИСНОВКИ

За результатом роботи розроблено програмне забезпечення пошуку партнерів для спільних занять видами спорту, що потребують декількох учасників, у випадку відсутності партнерів.

Виконаний аналіз предметної області дозволив встановити, що проблема пошуку партнерів для занять спортом є актуальною, а наявні на даний момент інструменти мають недоліки, які повинні бути вирішені в програмі, що розроблялась в роботі.

Для реалізації програми було обрано мову програмування C#, вебфреймворк ASP.NET Core, систему керування реляційними базами даних Microsoft SQL Server, утиліту для адміністрування SQL Server Management Studio, фреймворк для роботи з даними EF Core, а також використано інтегроване середовище розробки Visual Studio, мови HTML і CSS, бібліотека Bootstrap. Сценарії роботи користувачів описані за допомогою діаграм прецедентів. Визначено сутності предметної області і представлено відповідні таблиці БД і зв'язки між ними. На основі шаблону проєктування Model-View-Controller визначена структура програми.

Реалізовано вебзастосунок, включаючи зокрема класи моделей, класи інтерфейсів, класи сервісів, класи представлень, а також самі представлення під усі функції, що визначалися функціональними вимогами, описано послідовність роботи програми щодо основних користувацьких функцій.

Представлено основні інтерфейси роботи користувачів з програмою. Виконано тестування програми з перевіркою коректності реалізації всіх запланованих функцій, що підтвердило коректність.

Програма розроблена для підтримки спільноти людей, що займаються спортом, але тимчасово не мають партнерів, щоб реалізувати свої інтереси, надаючи їм інструменти для створення пропозицій і з іншого боку для пошуку серед пропозицій, створених іншими користувачами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Dominic, A. Only 23 Percent of Americans Meet National Exercise Guidelines [Electronic resource] / A. Dominic. – Access mode : <https://www.clubindustry.com/fitness-studies/only-23-percent-americans-meet-national-exercise-guidelines>.
2. Rakicevic, M. 21 Fascinating Exercise Statistics You Should Know in 2022 [Electronic resource] / M. Rakicevic. – Access mode : <https://disturbmenot.co/exercise-statistics/>.
3. Ducharme, J. Only 23% of Americans Get Enough Exercise, a New Report Says [Electronic resource] / J. Ducharme. – Access mode : <https://time.com/5324940/americans-exercise-physical-activity-guidelines/>.
4. RacketPal [Electronic resource]. – Access mode : <https://racketpal.co.uk/>.
5. Sporty. Activities with People Nearby [Electronic resource]. – Access mode : <https://sportyapp.com/>.
6. Matchup [Electronic resource] : Sport Partner Nearby Mobile App. The Best Mobile App Awards. – Access mode : <https://bestmobileappawards.com/app-submission/matchup-sport-partner-nearby>.
7. Albahari, J. C# 10 in a Nutshell [Text] : The Definitive Reference / J. Albahari. – Massachusetts : O'Reilly Media, 2022. – 1058 p.
8. ASP.NET documentation. Microsoft Docs [Electronic resource]. – Access mode : <https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-6.0>.
9. Lock, A. ASP.NET Core in Action, Second Edition [Text] / A. Lock. – Manning, 2021. – 832 p.
10. Murach, J. Murach's ASP.NET Core MVC [Text] / J. Murach. – Mike Murach & Associates, 2020. – 780 p.

ДОДАТОК А
Технічне завдання

Вступ

Командні види спорту, а також індивідуальні, що потребують участі декількох партнерів, потребують не тільки власної участі, але і партнерів. Сучасний стиль життя, складні графіки людей особливо у великих містах призводять до того, що може виникати бажання зайнятися спортом, але на потрібний часовий проміжок або у потрібному місці партнерів немає. Тоді як раз і потрібно знайти людину або людей, які можливо тимчасово, а можливо в майбутньому і постійно будуть виконувати роль партнерів.

А.1 Підстави для розробки

Дипломна кваліфікаційна робота виконується за темою «Програмне забезпечення пошуку партнерів для занять спортом» у відповідності з наказом № 102 від 27 квітня 2022 р. за Національним університетом «Запорізька політехніка».

А.2 Призначення розробки

Програма, яка розробляється, призначена для підтримки спільноти людей, що займаються спортом, але тимчасово не мають партнерів, щоб реалізувати свої інтереси, надаючи їм інструменти для створення пропозицій і з іншого боку для пошуку серед пропозицій, створених іншими користувачами.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Звичайний користувач програми повинен бути забезпечений наступними функціями:

- створення пропозиції про спільне заняття спортом;
- визначення інтересів партнера;
- створення зустрічі;
- створення власного переліку партнерів;
- визначення видів спорту, якими займається партнер;
- висунення зворотної пропозиції про заняття спортом;
- відправлення повідомлення щодо пропозиції про заняття спортом;
- перегляд повідомлень за пропозицією;
- збереження шаблону для створення пропозицій;
- уподобання зустрічі;
- пошук пропозицій;
- пошук спортивних майданчиків;
- перегляд пропозицій за партнером;
- перегляд пропозицій за спортивним майданчиком;
- перегляд зустрічей за спортивним майданчиком;
- перегляд зустрічей за партнером;
- пошук зустрічей;
- створення відгуку про партнера;
- створення відгуку про зустріч;
- перегляд спортивних майданчиків за районом міста;
- перегляд пропозицій за районом міста;
- перегляд пропозицій;
- перегляд зворотної пропозиції;
- перегляд даних спортивного майданчика;
- перегляд зустрічі.

Адміністратор програми повинен бути забезпечений наступними функціями:

- додавання, відображення і редагування інтересів;
- додавання, відображення і редагування видів спорту;

- додавання, відображення і редагування речей, що використовуються для заняття спортом;
- додавання, відображення і редагування міст і районів міст;
- додавання, відображення і редагування спортивних майданчиків.

А.3.2 Умови експлуатації

Для експлуатації програмного забезпечення потрібно забезпечити окремий сервер, на якому буде запущено вебсервер і сервер бази даних для постійного доступу користувачів до сервісу бронювання паркувальних місць.

А.3.3 Вимоги до складу та параметрів технічних засобів

Для забезпечення підтримки роботи вебсервера та сервера бази даних необхідно надати апаратне забезпечення, що включає мінімально процесор, що має не менше ніж 4 ядра з тактовою частотою не менше 2,1 ГГц, підтриманий оперативною пам'яттю обсягом не менше 8 Гб та жорстким диском з доступними 10 Гб простору на ньому.

А.4 Порядок контролю та приймання

Для контролю виконання дипломної кваліфікаційної роботи повинен використовуватися відповідний календарний план, який затверджено викладачем у завданні на роботу. Приймання роботи відбувається на основі проведення процедури захисту роботи.

ДОДАТОК Б
Текст програми

Б.1 Текст файла Proposition.cs

```

using System;
using System.Collections.Generic;
using SportPartnershipApp.Data.Enums;

namespace SportPartnershipApp.Data.Models
{
    public class Proposition
    {
        public int Id //ідентифікатор пропозиції
        { get; set; }
        public int PartnerId //ідентифікатор партнера, що
створив пропозицію
        { get; set; }
        public int SportId //ідентифікатор виду спорту, до
якого належить пропозиція
        { get; set; }
        public DateTime Slotstart //початок вільного часу
        { get; set; }
        public DateTime Slotend //завершення вільного часу
        { get; set; }
        public TimeSpan Maxduration //маскимальна тривалість
заняття
        { get; set; }
        public TimeSpan Minduration //мінімальна тривалість
заняття
        { get; set; }
        public string Detailed //деталізований опис
пропозиції
        { get; set; }
        public short Minparticipants //мінімальна необхідна
кількість учасників
        { get; set; }
        public short Maxparticipants //максимальна необхідна
кількість учасників
        { get; set; }
        public short Availableparticipants //вже наявна
кількість учасників
        { get; set; }
        public int DistrictId //район міста, де є можливість
провести заняття
        { get; set; }
        public StateType State //стан пропозиції
        { get; set; }
        public DateTime Starttime //фактичний час початку
заняття

```

```

        { get; set; }
        public DateTime Endtime //фактичний час завершення
заняття
        { get; set; }
        public virtual Partner Partner //партнер, який
створив пропозицію
        { get; set; }
        public virtual Sport Sport //вид спорту
        { get; set; }
        public virtual District District //район міста
        { get; set; }
        public virtual ICollection<PropositionSportlocation>
PropositionSportlocations //запропоновані майданчики, на яких може
відбуватися зустріч
        { get; set; }
        public virtual Meeting Meeting //зустріч, яка була
проведена за пропозицією
        { get; set; }
        public virtual ICollection<Offer> Offers //зустрічні
пропозиції від учасників про зустріч, тобто про їх участь у
запланованому занятті
        { get; set; }
        public virtual ICollection<Thing> Things //речі, які
потрібні для проведення заняття
        { get; set; }
        public virtual ICollection<PropositionMessage>
PropositionMessages //повідомлення стосовно цієї пропозиції
        { get; set; }
    }
}

```

Б.2 Текст файла Offer.cs

```

using System;
using System.Collections.Generic;
using SportPartnershipApp.Data.Enums;

namespace SportPartnershipApp.Data.Models
{
    public class Offer
    {
        public int Id //ідентифікатор зворотної пропозиції
        { get; set; }
        public int PropositionId //ідентифікатор пропозиції,
на яку створено цю зворотню пропозицію
        { get; set; }
    }
}

```

```

        public int PartnerId //ідентифікатор партнера, що
створив зворотню пропозицію
        { get; set; }
        public DateTime Timed //час створення пропозиції
        { get; set; }
        public DateTime Proposedstart //пропонований час
початку
        { get; set; }
        public TimeSpan Duration //пропонована тривалість
        { get; set; }
        public string Detailed //текстовий опис зворотньої
пропозиції
        { get; set; }
        public short Participantnum //кількість учасників,
яка є в групі, що готова долучитися
        { get; set; }
        public OfferStateType State //стан зворотньої
пропозиції
        { get; set; }
        public virtual Proposition Proposition //пропозиція,
на яку створено цю зворотню пропозицію
        { get; set; }
        public virtual Partner Partner //партнер, який
створив зворотню пропозицію
        { get; set; }
    }
}

```

Б.3 Текст файла PartnerReview.cs

```

using System;
using System.Collections.Generic;
using SportPartnershipApp.Data.Enums;

namespace SportPartnershipApp.Data.Models
{
    public class PartnerReview
    {
        public int Id //ідентифікатор відгуку про партнера
        { get; set; }
        public int PartnerId //ідентифікатор автора відгуку
        public int AboutpartnerId //ідентифікатор партнера,
про якого створено відгук
        { get; set; }
        public DateTime Timed //час створення відгуку
        { get; set; }
        public string Detailed //текст відгуку
    }
}

```

```

        { get; set; }
        public short Score //виставлена оцінка
        { get; set; }
        public virtual Partner Partner //автор відгуку
        { get; set; }
        public virtual Partner Aboutpartner //партнер, про
якого створено відгук
        { get; set; }
    }
}

```

Б.4 Текст файла PropositionMessage.cs

```

using System;
using System.Collections.Generic;
using SportPartnershipApp.Data.Enums;

namespace SportPartnershipApp.Data.Models
{
    public class PropositionMessage
    {
        public int Id //ідентифікатор повідомлення щодо
пропозиції
        { get; set; }
        public int PartnerId //ідентифікатор автора
повідомлення
        public int PropositionId //ідентифікатор пропозиції,
якої стосується повідомлення
        { get; set; }
        public DateTime Timed //час створення повідомлення
        { get; set; }
        public string Message //текст повідомлення
        { get; set; }
        public virtual Partner Partner //автор повідомлення
        { get; set; }
        public virtual Proposition Proposition //пропозиція,
якої стосується повідомлення
        { get; set; }
    }
}

```

Б.5 Текст файла Meeting.cs

```

using System;
using System.Collections.Generic;

```

```

using SportPartnershipApp.Data.Enums;

namespace SportPartnershipApp.Data.Models
{
    public class Meeting
    {
        public int Id //ідентифікатор зустрічі
        { get; set; }
        public int PartnerId //ідентифікатор партнера, що
створив зустріч
        { get; set; }
        public int SportId //ідентифікатор виду спорту, до
якого належить зустріч
        { get; set; }
        public DateTime Happened //час зустрічі
        { get; set; }
        public int PropositionId //пропозиція, до якої
належить ця зустріч (якщо прпозиція попередньо створювалась)
        { get; set; }
        public int SportlocationId //ідентифікатор майданчика
        { get; set; }
        public string Detailed //опис зустрічі
        { get; set; }
        public DateTime Timed //час створення зустрічі
        { get; set; }
        public float Score //середня оцінка зустрічі
        { get; set; }
        public short Participantsnum //кількість учасників
        { get; set; }
        public int Reviewcount //кількість відгуків від
учасників
        { get; set; }
        public int Likescount //кількість уподобань
        { get; set; }
        public virtual Partner Partner //партнер, який
створив зустріч
        { get; set; }
        public virtual Sport Sport //вид спорту
        { get; set; }
        public virtual Sportlocation Sportlocation
//майданчик
        { get; set; }
        public virtual Proposition Proposition //пропозиція,
за якою відбулась зустріч
        { get; set; }
        public virtual ICollection<MeetingParticipant>
MeetingParticipants //учасники зустрічі
        { get; set; }
    }
}

```

```

        public virtual ICollection<MeetingLike> MeetingLikes
//уподобання зустрічі
        { get; set; }
    }
}

```

Б.6 Текст файла MeetingParticipant.cs

```

using System;
using System.Collections.Generic;
using SportPartnershipApp.Data.Enums;

namespace SportPartnershipApp.Data.Models
{
    public class MeetingParticipant
    {
        public int Id //ідентифікатор участі партнера в
зустрічі
        { get; set; }
        public int PartnerId //ідентифікатор партнера
        public int MeetingId //ідентифікатор зустрічі
        { get; set; }
        public DateTime Timed //час внесення
        { get; set; }
        public string Review //текст відгуку
        { get; set; }
        public int Score //оцінка зустрічі
        { get; set; }
        public virtual Partner Partner //партнер
        { get; set; }
        public virtual Meeting Meeting //зустріч
        { get; set; }
    }
}

```

Б.7 Текст файла PropositionService.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SportPartnershipApp.Data;

```

```

using SportPartnershipApp.Data.Interfaces;
using SportPartnershipApp.Data.Models;
using Microsoft.EntityFrameworkCore;

namespace SportPartnershipApp.Service
{
    public class PropositionService : IProposition
    {
        private readonly DataContext _accesstodata;

        public PropositionService (DataContext accesstodata)
        {
            _accesstodata = accesstodata;
        }

        //метод, який надає повні дані пропозиції
        public Proposition GetById (int proposId)
        {
            return _accesstodata.Propositions.Where(prop =>
prop.Id == proposId)
                .Include(prop =>
prop.Partner).ThenInclude(part => part.PartnerSports)
                .Include(prop =>
prop.Partner).ThenInclude(part => part.Interests)
                .Include(prop => prop.Sport)
                .Include(prop => prop.District)
                .Include(prop => prop.Meeting)
                .Include(prop => prop.Sportlocations)
                .Include(prop => prop.Things)
                .First();
        }

        //метод, який надає повні дані пропозиції для її
власника
        public Proposition GetByIdForOwner (int proposId)
        {
            return _accesstodata.Propositions.Where(prop =>
prop.Id == proposId)
                .Include(prop => prop.Partner)
                .Include(prop => prop.Sport)
                .Include(prop => prop.District)
                .Include(prop => prop.Meeting)
                .Include(prop => prop.Sportlocations)
                .Include(prop => prop.Things)
                .Include(prop => prop.Offers)
                .Include(prop => prop.PropositionMessages)
                .First();
        }
    }
}

```

```

//метод, який формує перелік всіх пропозицій
public IEnumerable<Proposition> GetAll()
{
    return _accesstodata.Propositions
        .Include(prop => prop.Partner)
        .Include(prop => prop.Sport)
        .Include(prop => prop.District);
}

//метод, який формує перелік всіх пропозицій за
партнером partnerId, який її створив, та станом state, який може не
здаватися
public IEnumerable<Proposition>
GetPropositionsByPartner(int partnerId, StateType? state)
{
    if (state)
    {
        return
        _accesstodata.Propositions.Where(prop => prop.PartnerId ==
partnerId)
            .Where(prop => prop.State == state)
            .Include(prop => prop.Partner)
            .Include(prop => prop.Sport)
            .Include(prop => prop.District);
    }
    else
    {
        return
        _accesstodata.Propositions.Where(prop => prop.PartnerId ==
partnerId)
            .Include(prop => prop.Partner)
            .Include(prop => prop.Sport)
            .Include(prop => prop.District);
    }
}

//метод, який формує перелік всіх зворотніх
пропозицій за пропозицією
public IEnumerable<Offer> GetOffersByProposition (int
propositionId)
{
    return _accesstodata.Offers.Where(prop =>
prop.PropositionId == propositionId);
}

//метод, який повертає останню пропозицію загалом
(якщо не встановлено параметр) або у заданого партнера
public Proposition GetLastProposition(int? partnerId)

```

```

    {
        if (partnerId)
        {
            return
GetPropositionsByPartner(partnerId).Last();
        }
        else
        {
            return GetAll().Last();
        }
    }

    //метод, який реалізує пошук пропозицій за
    параметрами: start - початок вільного часу, end - закінчення
    вільного часу, sportId - вид спорту, minduration - хочу грати
    мінімум, maxduration - можу грати максимум, minparticip -
    мінімальні кількість учасників, maxparticip - максимальна кількість
    учасників, districtId - номер району, minscore - мінімальна
    прийнятна оцінка, maxscore - максимальна оцінка
    public IEnumerable<Proposition>
GetPropositionsByParameters (DateTime start, DateTime end, int
sportId, TimeSpan minduration, TimeSpan maxduration, short
minparticip, int maxparticip, int districtId, float minscore, float
maxscore)
    {
        return GetAll()
            .Where(prop
                => (prop.State == StateType.Free)
                && (prop.SportId == sportId)
                && (prop.Partner.Score >= minscore)
                && (prop.Minduration <= minduration)
                && (prop.Maxduration <= maxduration)
                && (prop.Slotstart <= start)
                && (prop.Slotend >= end)
                && (prop.Maxparticipants <= maxparticip)
                && (prop.Minparticipants >= minparticip)
                && (prop.DistrictId == districtId)
                && (prop.Partner.Score <= maxscore));
    }

    //метод реалізує додавання до БД пропозиції
    public async Task Add(Proposition propos)
    {
        _accesstodata.Add(propos);
        await _accesstodata.SaveChangesAsync();
    }

    // метод реалізує оновлення в БД пропозиції

```

```

        public async Task Update(Proposition propos)
        {
            _accesstodata.Update(propos);
            await _accesstodata.SaveChangesAsync();
        }

        //метод повертає список пропозицій активних за
        заданим майданчиком
        public IEnumerable<Proposition>
        GetPropositionsBySportlocation (int sportlocationId)
        {
            return GetAll()
                .Where(prop => (prop.State ==
StateType.Free))
                .Where(prop =>
prop.PropositionSportlocations.Any(sl => (sl.Id == sportlocationId)
&& (sl.Additional == true)));
        }
    }
}

```

Б.8 Текст файла IProposition.cs

```

using System;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;
using SportPartnershipApp.Data.Models;

namespace SportPartnershipApp.Data.Interfaces
{
    public interface IProposition
    {
        Proposition GetById(int propos_id);
        Proposition GetByIdForOwner (int proposId);
        IEnumerable<Proposition> GetAll();
        IEnumerable<Proposition> GetPropositionsByPartner(int
partnerId, StateType? state);
        IEnumerable<Offer> GetOffersByProposition (int
propositionId);
        Proposition GetLastProposition(int? partnerId);
        Proposition GetPropositionsByParameters (DateTime
start, DateTime end, int sportId, TimeSpan minduration, TimeSpan
maxduration, short minparticip, int maxparticip, int districtId,
float minscore, float maxscore);
    }
}

```

```

        IEnumerable<Proposition>
        GetPropositionsBySportlocation (int sportlocationId);
        Task Add(Proposition propos);
        Task Task Update(Proposition propos);
    }
}

```

Б.9 Текст файла MeetingService.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SportPartnershipApp.Data;
using SportPartnershipApp.Data.Interfaces;
using SportPartnershipApp.Data.Models;
using Microsoft.EntityFrameworkCore;

namespace SportPartnershipApp.Service
{
    public class MeetingService : IMeeting
    {
        private readonly DataContext _accesstodata;

        public MeetingService (DataContext accesstodata)
        {
            _accesstodata = accesstodata;
        }

        //метод, який надає повні дані про зустріч
        public Meeting GetById (int meetingId)
        {
            return _accesstodata.Meetings.Where(meet =>
meet.Id == meetingId)
                .Include(meet => meet.Partner)
                .Include(meet =>
meet.Partner).ThenInclude(part => part.PartnerSports)
                .Include(meet =>
meet.Partner).ThenInclude(part => part.Interests)
                .Include(meet => meet.Proposition)
                .Include(meet => meet.Sportlocation)
                .Include(meet => meet.Sport)
                .First();
        }

        //метод, який формує перелік всіх зустрічей
    }
}

```

```

public IEnumerable<Meeting> GetAll()
{
    return _accesstodata.Meetings
        .Include(meet => meet.Partner)
        .Include(meet => meet.Proposition)
        .Include(meet => meet.Sportlocation)
        .Include(meet => meet.Sport);
}

//метод, який формує перелік всіх зустрічей за
партнером partnerId, який їх створив
public IEnumerable<Meeting> GetMeetingsByPartner(int
partnerId)
{
    return _accesstodata.Meetings
        .Where(meet => (meet.PartnerId == partnerId)
        || (meet.Proposition.PartnerId == partnerId))
        .Include(meet => meet.Partner)
        .Include(meet => meet.Proposition)
        .Include(meet => meet.Sportlocation)
        .Include(meet => meet.Sport);
}

//метод, який повертає час найстарішої зустрічі
public DateTime GetFirstMeetingTime()
{
    var meet = GetAll().OrderByDescending(meet =>
meet.Happened).Last();
    return meet.Happened;
}

//метод, який повертає час останньої зустрічі
public DateTime GetLastMeetingTime()
{
    var meet = GetAll().OrderByDescending(meet =>
meet.Happened).First();
    return meet.Happened;
}

public Meeting GetLastMeeting(int? partnerId)
{
    if (partnerId)
    {
        return
GetMeetingsByPartner(partnerId).Last();
    }
    else
    {

```

```

        return GetAll().Last();
    }
}

//метод реалізує додавання до БД зустрічі
public async Task Add(Meeting meet)
{
    _accesstodata.Add(meet);
    await _accesstodata.SaveChangesAsync();
}

// метод реалізує оновлення в БД зустрічі
public async Task Update(Meeting meet)
{
    _accesstodata.Update(meet);
    await _accesstodata.SaveChangesAsync();
}

//метод повертає список зустрічей за заданим
майданчиком
public IEnumerable<Meeting>
GetMeetingsBySportlocation (int sportlocationId)
{
    return GetAll()
        .Where(meet => meet.SportlocationId ==
sportlocationId);
        .OrderByDescending(meet => meet.Happened);
}

//метод повертає список зустрічей, в яких брав участь
учасник
public IEnumerable<Meeting> GetMeetingsWithPartner
(int partnerId)
{
    return GetAll()
        .Where(meet =>
meet.MeetingParticipants.Any(part => part.PartnerId == partnerId));
}

//метод, який реалізує пошук зустрічей за
параметрами: start - початок часового інтервалу, end - закінчення
часового інтервалу, sportId - вид спорту, minscore - мінімальна
прийнятна оцінка, maxscore - максимальна оцінка, sportlocationId -
спортивний майданчик, де відбувалась зустріч
public IEnumerable<Meeting> GetMeetingsByParameters
(DateTime start, DateTime end, int sportlocationId, float minscore,
float maxscore, int sportId)
{

```

```

if ((sportId == -1) && (sportlocationId == -1))
{
    return GetAll()
        .Where(meet
            => (meet.Happened >= start)
            && (meet.Score >= minscore)
            && (meet.Happened <= end)
            && (meet.Score <= maxscore));
}
else if (sportlocationId == -1)
{
    return GetAll()
        .Where(meet
            => (meet.Happened >= start)
            && (meet.SportId == sportId)
            && (meet.Score >= minscore)
            && (meet.Happened <= end)
            && (meet.Score <= maxscore));
}
else if (sportId == -1)
{
    return GetAll()
        .Where(meet
            => (meet.Happened >= start)
            && (meet.Score >= minscore)
            && (meet.Happened <= end)
            && (meet.SportlocationId ==
sportlocationId)
            && (meet.Score <= maxscore));
}
else
{
    return GetAll()
        .Where(meet
            => (meet.Happened >= start)
            && (meet.SportId == sportId)
            && (meet.Score >= minscore)
            && (meet.Happened <= end)
            && (meet.SportlocationId ==
sportlocationId)
            && (meet.Score <= maxscore));
}
}

public async Task CalculateMeeting(int meetingId)
{
    var meet = GetById (meetingId);
    var participants = meet.MeetingParticipants;

```

```

        var partreviews = participants.Where(p => p.Score
> 0);

        var partList = partreviews.ToList();
        float av = 0;
        int particCount = partList.Count;
        foreach (var partic in partList)
        {
            if (partic.Score > 0)
                av = av + partic.Score / particCount;
        }
        meet.Score = av;
        meet.Reviewcount = particCount;
        var likes = meet.MeetingLikes.ToList();
        meet.Likescount = likes.Count;
        _accesstodata.Update(meet);
        await _accesstodata.SaveChangesAsync();
    }

}

```

Б.10 Текст файла IMeeting.cs

```

using System;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;
using SportPartnershipApp.Data.Models;

namespace SportPartnershipApp.Data.Interfaces
{
    public interface IMeeting
    {
        Meeting GetById (int meetingId);
        IEnumerable<Meeting> GetAll();
        IEnumerable<Meeting> GetMeetingsByPartner(int
partnerId);

        MeetingGetLastMeeting(int? partnerId);
        Task Add(Meeting meet);
        Task Update(Meeting meet);
        IEnumerable<Meeting> GetMeetingsBySportlocation (int
sportlocationId);
        IEnumerable<Meeting> GetMeetingsWithPartner (int
partnerId);

        IEnumerable<Meeting> GetMeetingsByParameters
(DateTime start, DateTime end, int sportlocationId, float minscore,
float maxscore, int sportId);
    }
}

```

```

        Task CalculateMeeting(int meetingId);
    }
}

```

Б.11 Текст файла SportlocationController.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using SportPartnershipApp.Data.Models.Sportlocation;
using SportPartnershipApp.Data.Models.Meeting;
using SportPartnershipApp.Data.Models.District;
using SportPartnershipApp.Data.Models.Locationtype;

namespace SportPartnershipApp.Controllers
{
    public class SportlocationController : Controller
    {
        private readonly ISportlocation
_sportlocationService;
        private readonly IMeeting _meetingService;
        private readonly IDistrict _districtService;
        private readonly ILocationtype _locationtypeService;

        public SportlocationController (ISportlocation
sportlocationService, IDistrict districtService, ILocationtype
locationtypeService)
        {
            _sportlocationService = sportlocationService;
            _districtService = districtService;
            _locationtypeService = locationtypeService;
            _meetingService = meetingService;
        }

        //метод, який підтримує заповнення представлення
перегляду спортмайданчика, вносячи додатково кількість зустрічей на
майданчику
        public async Task<IActionResult> Index(int
sportlocationId)
        {
            var sportlocation = _sportlocationService.GetById
(sportlocationId);
            ViewData["sloc"] = sportlocation;
            ViewData["meetcount"] = _meetingService
GetMeetingsBySportlocation(sportlocationId);

```

```

        return View();
    }

    public IActionResult Create()
    {
        ViewData["DistrictId"] = new
SelectList(_districtService.GetAll(), "Id", "Title");
        ViewData["LocationtypeId"] = new
SelectList(_locationtypeService.GetAll(), "Id", "Title");
        return View();
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    //метод, який підтримує створення спортивного
майданчика
    public async Task<IActionResult> Create([Bind("Id,
DistrictId, LocationtypeId, Title, Detailed, Phone")] Sportlocation
sportlocation)
    {
        if (ModelState.IsValid)
        {
            try
            {
                _sportlocationService.Add(sportlocation);
                return RedirectToAction(nameof(Index));
            }
            catch (Exception ex)
            {
                return View();
            }
        }
        ViewData["DistrictId"] = new
SelectList(_districtService.GetAll(), "Id", "Title",
sportlocation.DistrictId);
        ViewData["LocationtypeId"] = new
SelectList(_locationtypeService.GetAll(), "Id", "Title",
sportlocation.LocationtypeId);
        return View(sportlocation);
    }
    public async Task<IActionResult> Edit(int
sportlocationId)
    {
        if (sportlocationId == null)
        {
            return NotFound();
        }
    }

```

```

        var sportlocation = _sportlocationService.GetById
(sportlocationId);
        if (sportlocation == null)
        {
            return NotFound();
        }
        ViewData["DistrictId"] = new
SelectList(_locationtypeService.GetAll(), "Id", "Title",
sportlocation.LocationtypeId);
        ViewData["LocationtypeId"] = new
SelectList(_locationtypeService.GetAll(), "Id", "Title");
        return View(sportlocation);
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    //метод, який підтримує редагування спортивного
майданчика
    public async Task<IActionResult> Edit(int
sportlocationId, [Bind("Id, DistrictId, LocationtypeId, Title,
Detailed, Phone")] Sportlocation sportlocation)
    {
        if (sportlocationId != sportlocation.Id)
        {
            return NotFound();
        }

        if (ModelState.IsValid)
        {
            try
            {
                _sportlocationService.Update(sportlocation);
                return RedirectToAction("Index");
            }
            catch (DbUpdateConcurrencyException)
            {
                return NotFound();
            }
        }
        ViewData["DistrictId"] = new
SelectList(_districtService.GetAll(), "Id", "Title",
sportlocation.DistrictId);
        ViewData["LocationtypeId"] = new
SelectList(_locationtypeService.GetAll(), "Id", "Title",
sportlocation.LocationtypeId);
        return View(sportlocation);
    }

```

//метод, який підтримує відтворення спортивних майданчиків за районом міста districtId (typeop true) або містом cityId (typeop false)

```
public IActionResult DistrictSportlocations(bool
typeop, int? districtId, int? cityId)
{
    if (typeop)
    {
        var sportlocations =
        _sportlocationService.GetSportlocationsByDistrict (districtId);
        ViewData["sportlocations"] =
        sportlocations.ToList();
        return View();
    }
    else
    {
        var sportlocations =
        _sportlocationService.GetSportlocationsByCity (cityId);
        ViewData["sportlocations"] =
        sportlocations.ToList();
        return View();
    }
}
```

//метод, який підтримує пошук спортивних майданчиків за видом спорту sportId та районом міста districtId, який може бути пропущений

```
public IActionResult SearchSportlocations(int
sportId, int? districtId, int? cityId)
{
    if (districtId)
    {
        var sportlocations =
        _sportlocationService.GetSportlocationsByParameters (sportId,
        districtId, 0);
        ViewData["sportlocations"] =
        sportlocations.ToList();
        return View();
    }
    else
    {
        var sportlocations =
        _sportlocationService.GetSportlocationsByDistrict (sportId, 0,
        cityId);
        ViewData["sportlocations"] =
        sportlocations.ToList();
        return View();
    }
}
```

```

    }
}
}

```

Б.12 Текст файла MeetingController.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Globalization;
using SportPartnershipApp.Data.Models.Meeting;
using SportPartnershipApp.Data.Models.Sportlocation;
using SportPartnershipApp.Data.Models.District;
using SportPartnershipApp.Data.Models.Locationtype;
using Microsoft.AspNet.Identity;

namespace SportPartnershipApp.Controllers
{
    public class MeetingController : Controller
    {
        private readonly IMeeting _meetingService;
        private readonly ISportlocation
_sportlocationService;
        private readonly ISport _sportService;

        public SportlocationController (ISportlocation
sportlocationService, IMeeting meetingService, ISport sportService)
        {
            _sportlocationService = sportlocationService;
            _meetingService = meetingService;
            _sportService = sportService;
        }

        public async Task<IActionResult> Index(int meetingId)
        {
            var meeting = _meetingService.GetById
(meetingId);

            ViewData["sloc"] = meeting;
            return View();
        }

        public IActionResult Create()
        {

```

```

        ViewData["PropositionId"] = new
SelectList(_propositionService.GetMeetingsByPartner(User.Identity.G
etUserId()), "Id", "Id");
        ViewData["SportlocationId"] = new
SelectList(_sportlocationService.GetAll(), "Id", "Title");
        ViewData["SportId"] = new
SelectList(_sportService.GetAll(), "Id", "Title");
        return View();
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    //метод, який підтримує представлення створення
зустрічі з наповненням даними доступних видів спорту, спортивних
майданчиків та пропозицій авторизованого партнера, з яких може
створюватися зустріч
    public async Task<IActionResult> Create([Bind("Id,
PartnerId, SportId, Happened, PropositionId, SportlocationId,
Detailed, Participantsnum")] Meeting meeting)
    {
        if (ModelState.IsValid)
        {
            try
            {
                meeting.Timed = DateTime.Now;
                meeting.Score = 0;
                meeting.Reviewcount = 0;
                meeting.Likescount = 0;
                meeting.PartnerId =
User.Identity.GetUserId();
                _meetingService.Add(meeting);
                return RedirectToAction(nameof(Index));
            }
            catch (Exception ex)
            {
                return View();
            }
        }
        ViewData["PropositionId"] = new
SelectList(_propositionService.GetMeetingsByPartner(User.Identity.G
etUserId()), "Id", "Id", meeting.Id);
        ViewData["SportlocationId"] = new
SelectList(_sportlocationService.GetAll(), "Id", "Title",
meeting.SportlocationId);
        ViewData["SportId"] = new
SelectList(_sportService.GetAll(), "Id", "Title", meeting.SportId);
        return View(meeting);
    }

```

```

public async Task<IActionResult> Edit(int meetingId)
{
    if (meetingId == null)
    {
        return NotFound();
    }

    var meeting = _meetingService.GetById
(meetingId);

    if (meeting == null)
    {
        return NotFound();
    }

    ViewData["PropositionId"] = new
SelectList(_propositionService.GetMeetingsByPartner(User.Identity.G
etUserId()), "Id", "Id", meeting.Id);
    ViewData["SportlocationId"] = new
SelectList(_sportlocationService.GetAll(), "Id", "Title",
meeting.SportlocationId);
    ViewData["SportId"] = new
SelectList(_sportService.GetAll(), "Id", "Title", meeting.SportId);
    return View(meeting);
}

[HttpPost]
[ValidateAntiForgeryToken]
//метод, який підтримує представлення редагування
зустрічі з наповненням даними доступних видів спорту, спортивних
майданчиків та пропозицій авторизованого партнера
public async Task<IActionResult> Edit(int meetingId,
[Bind("Id, PartnerId, SportId, Happened, PropositionId,
SportlocationId, Detailed, Participantsnum")] Meeting meeting)
{
    if (meetingId != meeting.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _meetingService.Update(meeting);
            return RedirectToAction("Index");
        }
        catch (DbUpdateConcurrencyException)
        {
            return NotFound();
        }
    }
}

```

```

        }
    }
    _meetingService.CalculateMeeting(meetingId);
    ViewData["PropositionId"] = new
SelectList(_propositionService.GetMeetingsByPartner(User.Identity.G
etUserId()), "Id", "Id", meeting.Id);
    ViewData["SportlocationId"] = new
SelectList(_sportlocationService.GetAll(), "Id", "Title",
meeting.SportlocationId);
    ViewData["SportId"] = new
SelectList(_sportService.GetAll(), "Id", "Title", meeting.SportId);
    return View(meeting);
}

//метод, який підтримує створення представлення для
перегляду зустрічей за сопртивним майданчиком sportlocationId
public IActionResult SportlocationMeetings(int
sportlocationId)
{
    var meetings =
_meetingService.GetMeetingsBySportlocation (sportlocationId);
    ViewData["meetings"] = meetings.ToList();
    return View();
}

//метод, який підтримує представлення перегляду
зустрічей партнера partnerId
public IActionResult GetMeetingsWithPartner (int
partnerId)
{
    var meetings =
_meetingService.GetMeetingsWithPartner (partnerId);
    ViewData["meetings"] = meetings.ToList();
    return View();
}

//метод, який підтримує представлення перегляду
зустрічей, створених партнером partnerId
public IActionResult GetMeetingsByPartner (int
partnerId)
{
    var meetings =
_meetingService.GetMeetingsWithPartner (partnerId);
    ViewData["meetings"] = meetings.ToList();
    return View();
}

```

```

        //метод, який підтримує представлення пошуку
зустрічей
        public IActionResult SearchMeetings(DateTime? start,
DateTime? end, int? sportlocationId, float? minscore, float?
maxscore, int? sportId)
        {
            if (!start)
            {
                start =
                _meetingService.GetFirstMeetingTime();
            }
            if (!end)
            {
                end = _meetingService.GetLastMeetingTime();
            }
            if (!minscore)
            {
                minscore = 0;
            }
            if (!maxscore)
            {
                maxscore = 5;
            }
            if (!sportlocationId)
            {
                sportlocationId = -1;
            }
            if (!sportId)
            {
                sportId = -1;
            }
            var meetings =
            _meetingService.GetMeetingsByParameters (start, end,
            sportlocationId, minscore, maxscore, sportId);
            ViewData["meetings"] = meetings;
            return View();
        }
    }
}

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»

кафедра програмних засобів

Дипломна кваліфікаційна робота

**Програмне забезпечення пошуку партнерів
для занять спортом
Sports Partner Search Software**

Виконав
ст. гр. КНТ-138

С. В. Терлецький

Керівник
к.т.н., доцент

В. М. Льовкін

2022

Рисунок В.1 – Слайд № 1

Об'єкт роботи – процес пошуку партнерів для занять спортом.

Предмет роботи – програмне забезпечення пошуку партнерів для занять спортом.

Мета роботи – розробити програмне забезпечення пошуку партнерів для спільних занять видами спорту, що потребують декількох учасників, у випадку відсутності партнерів.

(2)

Рисунок В.2 – Слайд № 2

Завдання роботи

- аналіз предметної області;
- визначення вимог до програмного забезпечення пошуку партнерів для занять спортом;
- проєктування програмного забезпечення пошуку партнерів для занять спортом;
- реалізація і налагодження програмного забезпечення пошуку партнерів для занять спортом.

(3)

Рисунок В.3 – Слайд № 3

Результати порівняння аналогів

Критерій	RacketPal	Sporty	Дипломна розробка
Пошук пропозицій партнерів	Так	Ні	Так
Бронювання занять	Так	Так	Так
Платні функції пошуку	Так	Ні	Ні
Створення власних пропозицій	Так	Ні	Так
Платна участь в заняттях спортом	Так	Ні	Ні
Інформування про заняття спортом, що відбулись	Ні	Ні	Так
Створення шаблонів пропозицій	Ні	Ні	Так
Доступ з будь-якого типу пристрою	Ні	Так	Так
Доступ без облікового запису	Ні	Так	Так

(4)

Рисунок В.4 – Слайд № 4

Порівняння платформ розробки

Критерій	.NET Framework	.NET Core
Організація кросплатформних системи	Ні	Так
Обмежене розгортання	Ні	Можна розгортати окремі компоненти
Відкрите програмне забезпечення	Частково	Повністю
Ізоляція системи від інших застосунків операційної системи	Ні	Так

(5)

Рисунок В.5 – Слайд № 5

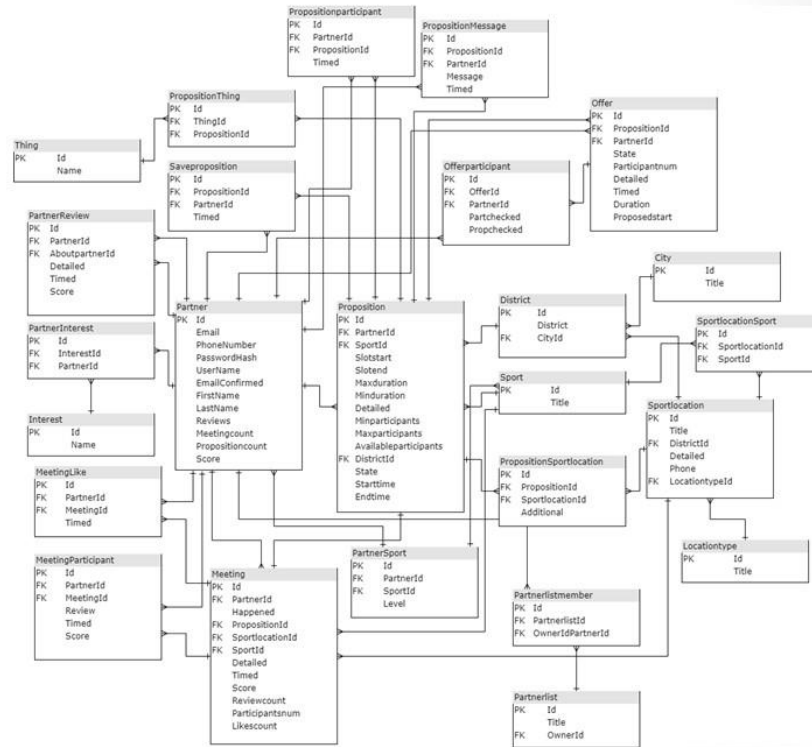
Діаграма прецедентів



(6)

Рисунок В.6 – Слайд № 6

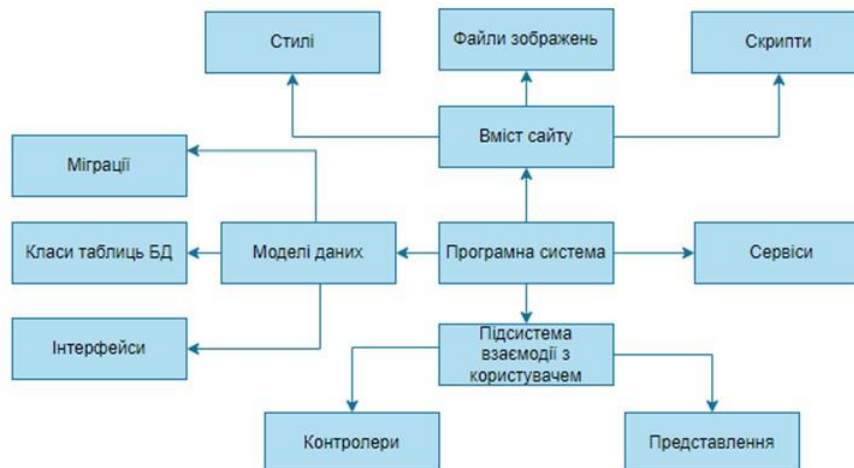
Схема бази даних



7

Рисунок В.7 – Слайд № 7

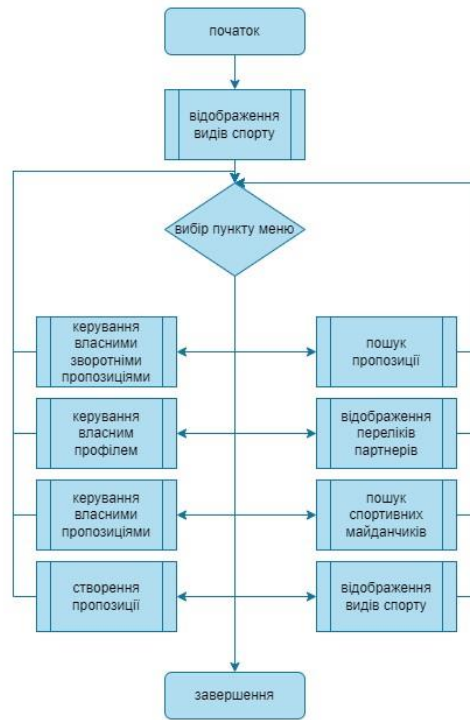
Структурна схема програми



8

Рисунок В.8 – Слайд № 8

Схема функціонування програми



9

Рисунок В.9 – Слайд № 9

Робота з програмою

PARTNERSPORT

Про себе Вийти

Пропозиції

Домашня

Зустрічі

Відповіді

Майданчики

Партнери

Запропонувати

Шукати пропозиції

Дата	Місце
14.06.2022	Запоріжжя
06:00	Шевченківський
22:00	2
01:00	5
02:00	2
Теніс	4

Показати



2 14.06.22 07:00-09:00
Організатор: Шеховцов Антон
Взяти участь



2-4 14.06.22 14:00-16:00
Організатор: Лященко Віктор
Взяти участь



2 14.06.22 17:00-20:00
Організатор: Петренко Павло
Взяти участь

10

Рисунок В.10 – Слайд № 10

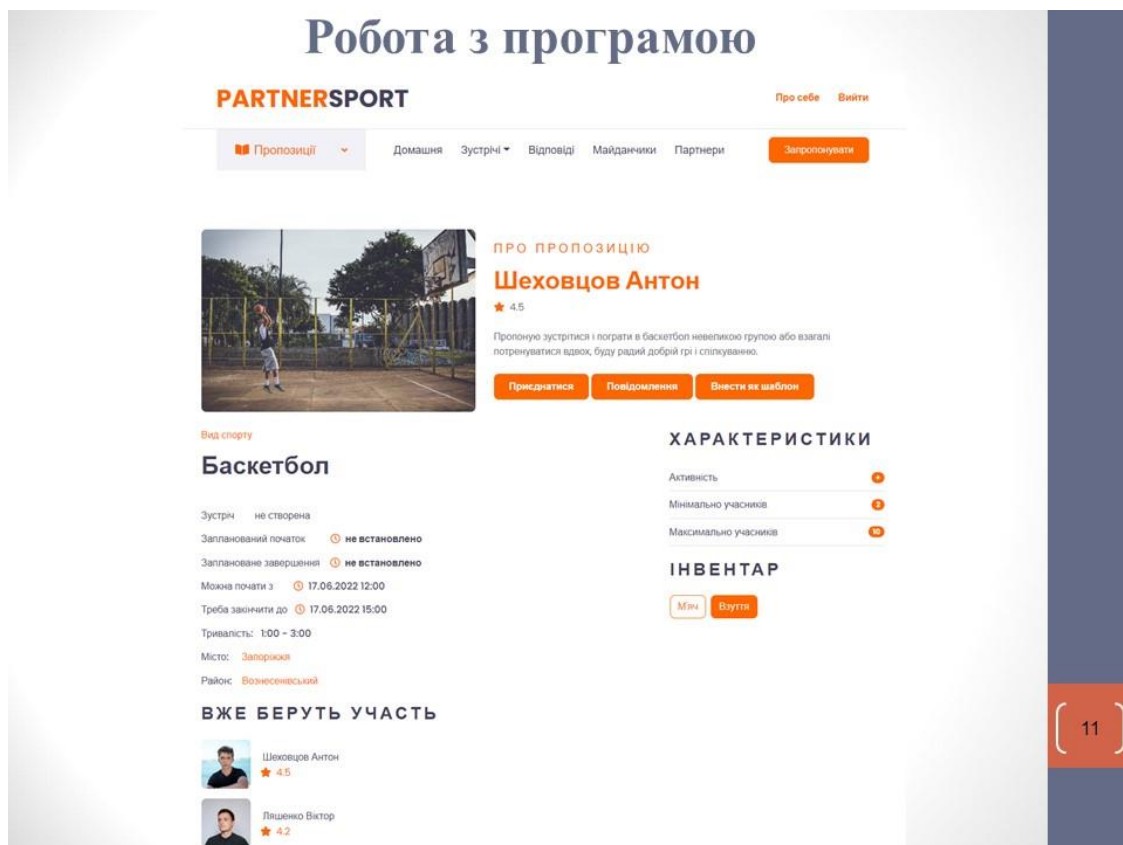


Рисунок В.11 – Слайд № 11

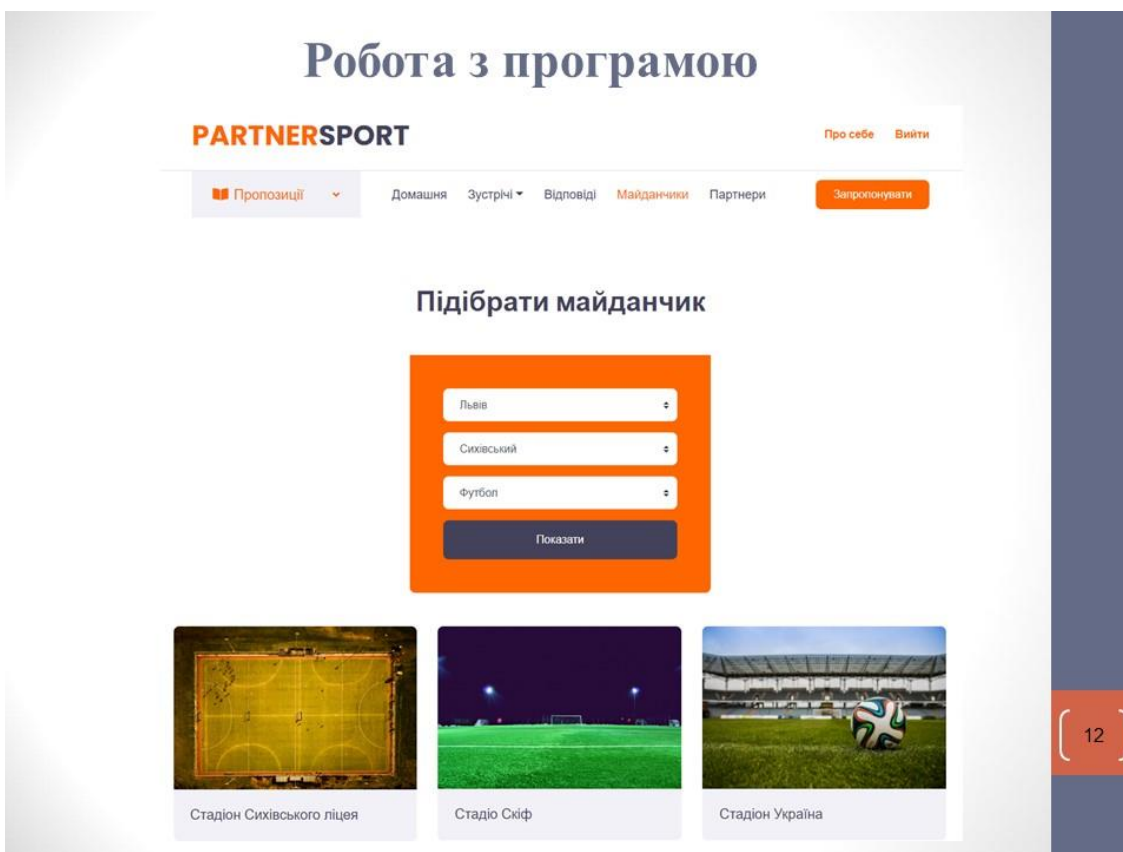


Рисунок В.12 – Слайд № 12

Результати тестування програми

Перевірка	Результат	Перевірка	Результат
Створення і редагування пропозиції про заняття спортом	+ / +	Пошук зустрічей	+
Визначення інтересів партнера	+	Створення відгуку про партнера	+
Створення і редагування зустрічі	+	Створення відгуку про зустріч	+
Створення власного переліку партнерів	+	Перегляд відгуків про партнера	+
Визначення видів спорту, якими займається партнер	+	Перегляд відгуків про зустріч	+
Додавання і редагування зворотної пропозиції про заняття спортом	+	Перегляд спортивних майданчиків за районом міста	+
Відправлення повідомлення щодо пропозиції про заняття спортом	+	Перегляд пропозицій за районом міста	+
Перегляд повідомлень за пропозицією	+	Перегляд пропозиції	+
Збереження шаблону для створення пропозицій	+	Перегляд зворотної пропозиції	+
Уподобання зустрічі	+	Перегляд даних спортивного майданчика	+
Пошук пропозицій	+	Перегляд зустрічі	+
Пошук спортивних майданчиків	+	Додавання, відображення і редагування інтересів	+ / + / +
Перегляд пропозицій за партнером	+	Додавання, відображення і редагування видів спорту	+ / + / +
Перегляд пропозицій за спортивним майданчиком	+	Додавання, відображення і редагування інвентаря	+ / + / +
Перегляд зустрічей за спортивним майданчиком	+	Додавання, відображення і редагування міст	+ / + / +
		Додавання, відображення і редагування районів міст	+ / + / +
		Додавання, відображення і редагування	+ / + / +

Рисунок В.13 – Слайд № 13

Висновки

Для реалізації програми було обрано мову програмування C#, вебфреймворк ASP.NET Core, систему керування реляційними базами даних Microsoft SQL Server, утиліту для адміністрування SQL Server Management Studio, фреймворк для роботи з даними EF Core. Сценарії роботи користувачів описані за допомогою діаграм прецедентів. Визначено сутності предметної області і представлено відповідні таблиці БД і зв'язки між ними. На основі шаблону проєктування Model-View-Controller визначена структура програми.

Реалізовано вебзастосунок, включаючи зокрема класи моделей, класи інтерфейсів, класи сервісів, класи представлень, а також самі представлення під усі функції, що визначалися функціональними вимогами, описано послідовність роботи програми щодо основних користувацьких функцій.

Програма розроблена для підтримки спільноти людей, що займаються спортом, але тимчасово не мають партнерів, щоб реалізувати свої інтереси, надаючи їм інструменти для створення пропозицій і з іншого боку для пошуку серед пропозицій, створених іншими користувачами.

Рисунок В.14 – Слайд № 14