

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ

(назва теми)

ПРОГНОЗУВАННЯ ПОКАЗНИКІВ ДІЯЛЬНОСТІ КЛУБУ ДЛЯ

ЗАНЯТЬ СПОРТОМ. RESEARCH AND SOFTWARE

IMPLEMENTATION OF METHODS TO FORECAST THE SPORT

CLUB PERFORMANCE INDICATORS

Виконав(ла): студент(ка) 2 курсу, групи КНТ-122м

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

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)

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет _____ ФКНТ _____
Кафедра _____ програмних засобів _____
Ступінь вищої освіти _____ магістр _____
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.
Сергій СУББОТІН
« _____ » _____ 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ТЕРЛЕЦЬКОГО Станіслава Віталійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів прогнозування показників діяльності клубу для занять спортом. Research and Software Implementation of Methods to Forecast the Sport Club Performance Indicators

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

затверджені наказом закладу вищої освіти від « 14 » листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації програмного забезпечення. 5. Експлуатація, тестування та експериментальне дослідження програми.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ЛЬОВКІН В.М., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання « 04 » вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Срок виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Дослідження проблеми	3-5 тижні	Розділ 2
4	Розробка архітектури програми	6 тиждень	Розділ 3
5	Розробка програми	7-8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програми	9-10 тижні	Розділ 5
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	11 тиждень	Додатки
8	Захист роботи	12 тиждень	

Студент(ка)

Stanislav Terletskyi

(підпис)

СТАНІСЛАВ ТЕРЛЕЦЬКИЙ

(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Валерій ЛЬОВКІН

(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
131 с., 55 рис., 6 табл., 3 дод., 15 джерел.

СПОРТИВНИЙ КЛУБ, АБОНЕМЕНТ, ПРОГНОЗУВАННЯ,
ЧАСОВИЙ РЯД, НЕЙРОННА МЕРЕЖА, LSTM.

Об'єкт роботи – процес керування діяльністю клубу для занять спортом. Предмет роботи – методи прогнозування показників діяльності клубу для занять спортом. Мета роботи – розробити програмне забезпечення підтримки роботи клубу для занять спортом з клієнтами для прийняття рішень щодо подальшої діяльності на основі результатів прогнозування кількості активних абонементів для занять спортом.

Матеріали, методи та технічні засоби: мова Python як основна мова розробки, фреймворк Django для веброзробки, система PostgreSQL для організації і керування базою даних.

Результати. Розроблено вебзастосунок, який дозволяє популяризувати спортивні активності, тренерів у спортивному клубі, організувати підтримку бронювання тренувань, продажу абонементів, забезпечити прогнозування кількості абонементів у розрізі видів спортивних активностей та груп за відсотком відвідуваності клієнтами спортивного залу.

Висновки. Наукова новизна роботи полягає в методі прогнозування кількості активних абонементів для занять спортом, який дозволяє прогнозувати кількість абонементів у майбутні дні за групою відвідуваності та спортивною активністю за кількістю абонементів у минулі дні, застосовуючи для побудови нейронних мереж архітектуру LSTM та застосовуючи правила нормування даних.

Галузь використання. Клуб для занять спортом, фітнес-центр, спортивний зал.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 131 pages, 55 figures, 6 tables, 3 appendixes, 15 sources.

SPORT CLUB, SUBSCRIPTION, PREDICTION, TIME SERIES, NEURAL NETWORK, LSTM.

The object of the work is the process of managing activities of the sports club. The subject of the work is methods for prediction of sport club performance indicators. The purpose of the work is to develop software to support the work of a sport club with clients to make decisions on further activities based on the results of predicting the number of active subscriptions for sports.

Materials, methods and technical means: Python language as the main development language, Django framework for web development, PostgreSQL system for database organization and management.

The results. Web application has been developed that allows to popularize sports activities, trainers in a sports club, organize training booking support, season ticket sales and provide prediction of the number of subscriptions in terms of types of sports activities and groups based on the percentage of customers visiting sports hall.

Conclusions. The scientific novelty of the work consists in the method of prediction of the number of active subscriptions for sport club, which allows to predict the number of subscriptions for the future days by attendance group and sports activity, using the number of subscriptions in the past days, using the LSTM architecture to build neural networks and applying data normalization rules.

Field of use. Sport club, fitness center, gym.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Огляд програмного забезпечення підтримки роботи з клієнтами клубу для занять спортом	11
1.2 Порівняння аналогів	14
1.3 Висновки за розділом 1	16
2 Матеріали і методи.....	17
2.1 Задача прогнозування кількості активних абонементів для занять спортом та методи її вирішення.....	17
2.2 Метод прогнозування кількості активних абонементів для занять спортом	20
2.3 Висновки за розділом 2	23
3 Проектування програмного забезпечення	24
3.1 Вибір засобів розробки	24
3.2 Моделювання програмного забезпечення на основі функціональних вимог	25
3.3 Проектування структури бази даних.....	35
3.4 Висновки за розділом 3	45
4 Основні рішення щодо реалізації програмного забезпечення	46
4.1 Порядок роботи програмного забезпечення	46
4.2 Рішення з реалізації вебзастосунку	47
4.3 Висновки за розділом 4	65
5 Експлуатація, тестування та експериментальне дослідження програми	67
5.1 Опис програми.....	67
5.2 Експлуатація програми.....	67
5.3 Експериментальне дослідження	75
5.4 Висновки за розділом 5	89

Висновки	91
Перелік джерел посилання	92
Додаток А Технічне завдання	94
Додаток Б Текст програми	101
Додаток В Слайди презентації	122

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

JSON – JavaScript Object Notation;

LSTM – Long Short-Term Memory;

MVT – Model View Template;

БД – база даних.

ВСТУП

Здоровий образ життя є достатньо популярним у світі, але окрім того популярним є образ і людини з гарним тілом. Це сприяє тому, щоб велика кількість людей долучалась до занять спортом. В результаті кількість функціонуючих клубів для занять спортом у багатьох містах є достатньо великою. Популярність образів не завжди призводить до того, що кожна людина після зацікавлення заняттями спортом та подальшого придбання абонементів такі заняття відвідує. Нерідко далі придбання абонементів справи не заходять. Навіть сплачена сума не додає стабільності відвідування, а то і відвідування в принципі.

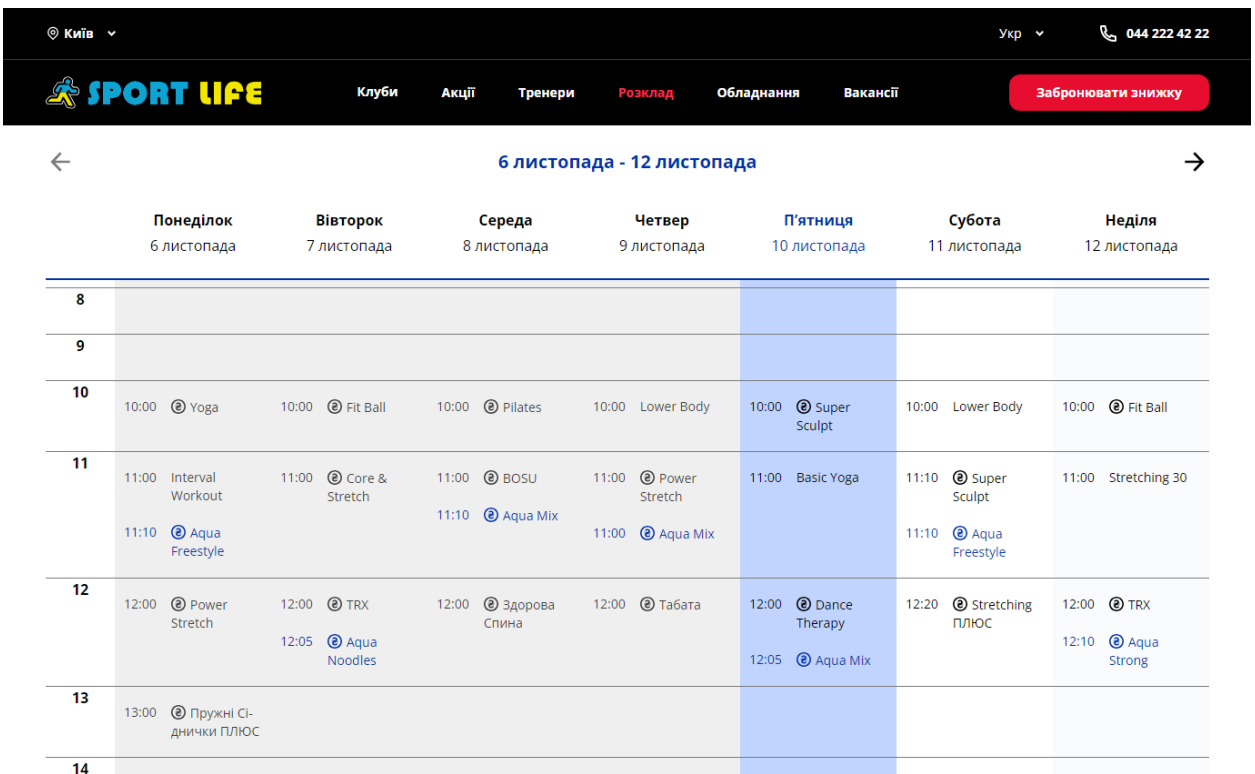
Для спортивного клубу це призводить до проблем з оптимальним використанням наявних засобів. Тобто кількість придбаних абонементів може бути такою, що не всі люди зможуть одночасно перебувати у спортивному клубі, якщо відвідають його саме в один момент. Проте не всі відвідують клуб в один і той самий момент. Відвідування відбувається в різні дні, різний час. Але не всі власники абонементів взагалі відвідують спортивний клуб в підсумку, інші роблять це зрідка. У підсумку важливо не тільки зацікавити відвідувачів доєднуватися до спортивного клубу, але і далі розуміти, як кількість абонементів буде змінюватися, при чому змінюватися в різних категоріях відвідуваності та за різними активностями, адже вони потребують різних приміщень, різного обладнання, різних тренерів. Реагувати ж на зміни потрібно якомога швидше. Це стосується і випадків, коли кількість абонементів зменшується, і коли кількість абонементів збільшується. Окрім раннього прийняття стратегічних рішень при цьому можна приймати також рішення про те, щоб зменшити або збільшити кількість тренерів за певними активностями. Це звичайно не єдине рішення, яке може бути прийняте, це фактично один з варіантів, але головне мати підґрунтя для подальшого прийняття рішень.

Тож саме таке програмне забезпечення, яке з одного боку має дозволити популяризувати спортивні активності, тренерів у спортивному клубі, підтримувати бронювання тренувань, а з іншого боку забезпечити прогнозування кількості абонементів у широкому розрізі для подальшого визначення показників діяльності клубу для занять спортом, необхідно розробити.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд програмного забезпечення підтримки роботи з клієнтами клубу для занять спортом

Sportlife (рис. 1.1) [1] є фактично найбільшою мережею фітнес-центрів, а загалом і спортивних закладів загалом, тому логічно розпочати огляд саме з його вебсайту. Вебсайт присвячений одразу всій мережі, тому по-перше, можна отримати інформацію про всі заклади клуби. Основною функцією є як раз інформування. Надається інформація про розклад занять у конкретному клубі на необхідний проміжок дат, про тренерів за конкретним клубом. Але за тренерами можна тільки отримати інформацію, побачити конкретний графік, забронювати заняття через вебсайт не можна.



	Понеділок 6 листопада	Вівторок 7 листопада	Середа 8 листопада	Четвер 9 листопада	П'ятниця 10 листопада	Субота 11 листопада	Неділя 12 листопада
8							
9							
10	10:00 @ Yoga	10:00 @ Fit Ball	10:00 @ Pilates	10:00 Lower Body	10:00 @ Super Sculpt	10:00 Lower Body	10:00 @ Fit Ball
11	11:00 Interval Workout 11:10 @ Aqua Freestyle	11:00 @ Core & Stretch	11:00 @ BOSU 11:10 @ Aqua Mix	11:00 @ Power Stretch 11:00 @ Aqua Mix	11:00 Basic Yoga	11:10 @ Super Sculpt 11:10 @ Aqua Freestyle	11:00 Stretching 30
12	12:00 @ Power Stretch	12:00 @ TRX 12:05 @ Aqua Noodles	12:00 @ Здорова Спина	12:00 @ Табата	12:00 @ Dance Therapy 12:05 @ Aqua Mix	12:20 @ Stretching ПЛЮС	12:00 @ TRX 12:10 @ Aqua Strong
13	13:00 @ Пружні Сіднички ПЛЮС						
14							

Рисунок 1.1 – Розклад занять на вебсайті Sportlife

На вебсайті надається інформація про вартість абонементів (рис. 1.2), можна оформити фактично заявку на купівлю абонементу. Тобто сплатити через вебсайт за абонемент не можна. Також немає і авторизації

користувачів, відповідно якоїсь персоналізованої функціональності немає. На вебсайті можна отримати також інформацію про наявні спортивні активності, про обладнання, яке доступне в мережі загалом для занять.

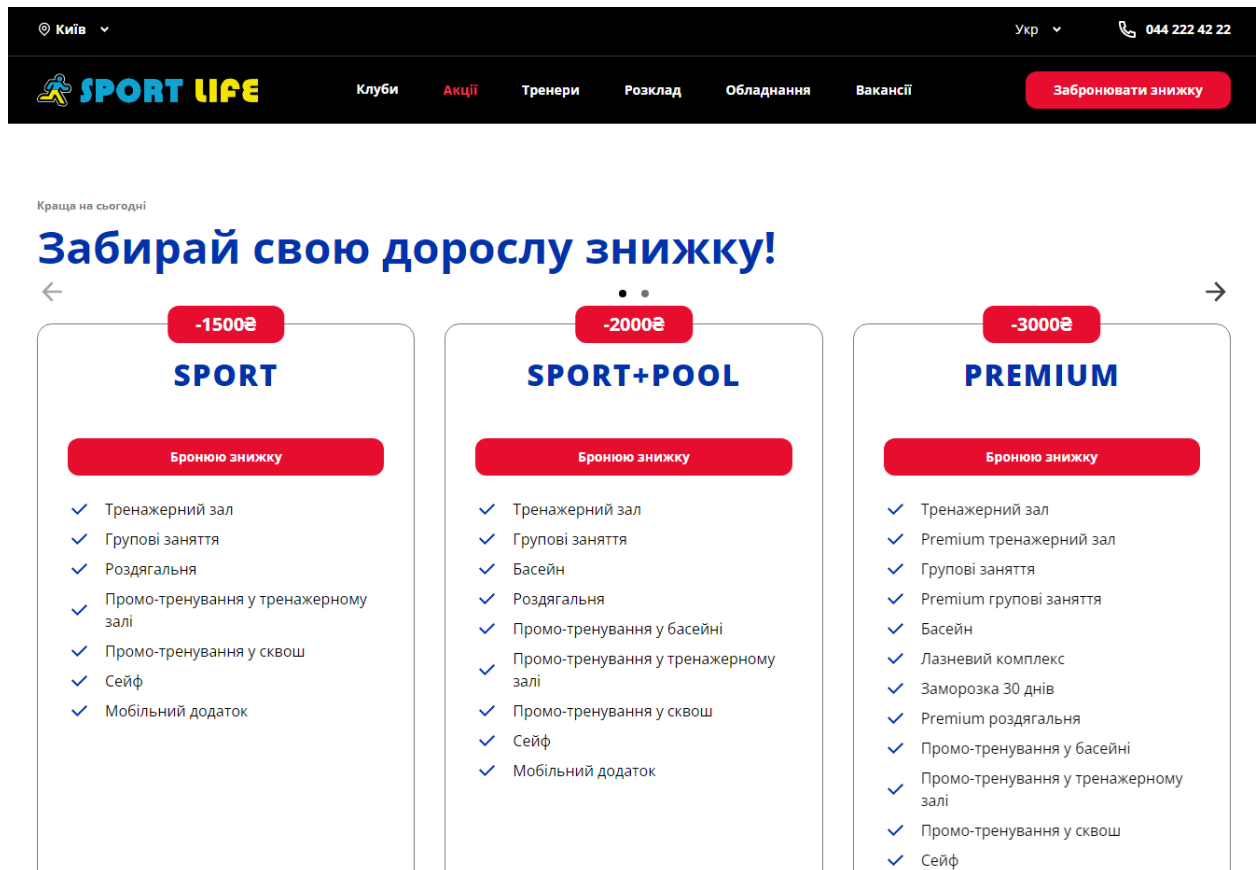


Рисунок 1.2 – Вартість абонементів на вебсайті Sportlife

Окрім того було розглянуто ще декілька вебсайтів різних клубів або подібних закладів для занять спортом. До прикладу в підсумку виділено вебсайт оздоровчого центру «Три стихії» (рис. 1.3) [2].

Оздоровчим центром надається цілий ряд спортивних активностей, відповідно вебсайт надає інформацію про них. Є широка система абонементів на відвідування, але окрім того за них можна також і заплатити на вебсайті. Для цього потрібно пройти через реєстрацію. Але сплатити можна за обмежений набір варіантів абонементів, що стосуються басейну, тренажерного залу, занять фітнесом.

три стихії

Послуги

Ціни

Про нас

Фотогалерея

Акції та новини

Блог Здоров'я та краси

Часті запитання

Контакти

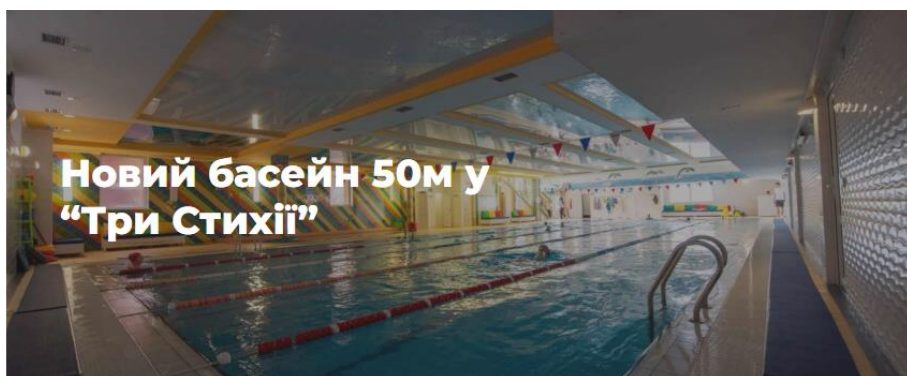
Форма реєстрації

Оплата на сайті

**Замовити послуги
онлайн**

Відгуки про оздоровчий
центр Три Стихії →

Блог
Здоров'я та краси



[Головна](#) / [Послуги](#) / [Новий басейн 50м у "Три Стихії"](#)

Ціни - басейн

Абонемент на повний день 08:00-20:30 (дорослий) "8 візитів"

Разовий візит	300,00 грн.
1 місяць	1300,00 грн.
3 місяці	3705,00 грн.
6 місяців	7020,00 грн.
12 місяців	13260,00 грн.

Рисунок 1.3 – Вебсайт оздоровчого центру «Три стихії»

На відміну від Sportlife вебсайт фітнес-клубу Palestra (рис. 1.4) [3] дозволяє придбати абонемент. Вебсайт також надає інформацію про тренерів, але як такого бронювання тренувань немає, є тільки форма для подання заявки. Тобто фактично можна залишити інформацію про себе, а далі вже контактування відбувається не через вебсайт, адже подальшої функціональності вебсайт не має. Наявний також у структурі блог, але він стосується загалом здоров'я, а не певних вправ.

Вебсайт Livestrong (рис. 1.5) [4] не є вебсайтом клубу для занять спортом або мережі, але він має у своєму складі розширену кількість порад, комплексів вправ для занять спортом та загалом щодо здоров'я, включаючи також правильне харчування, вибір взуття тощо.

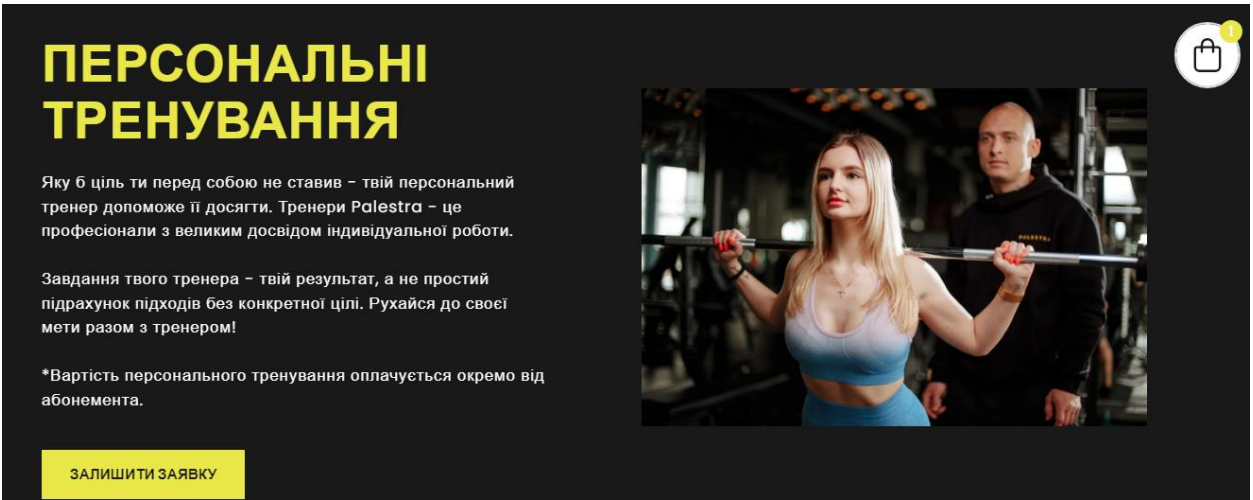


Рисунок 1.4 – Вебсайт фітнес-клубу Palestra

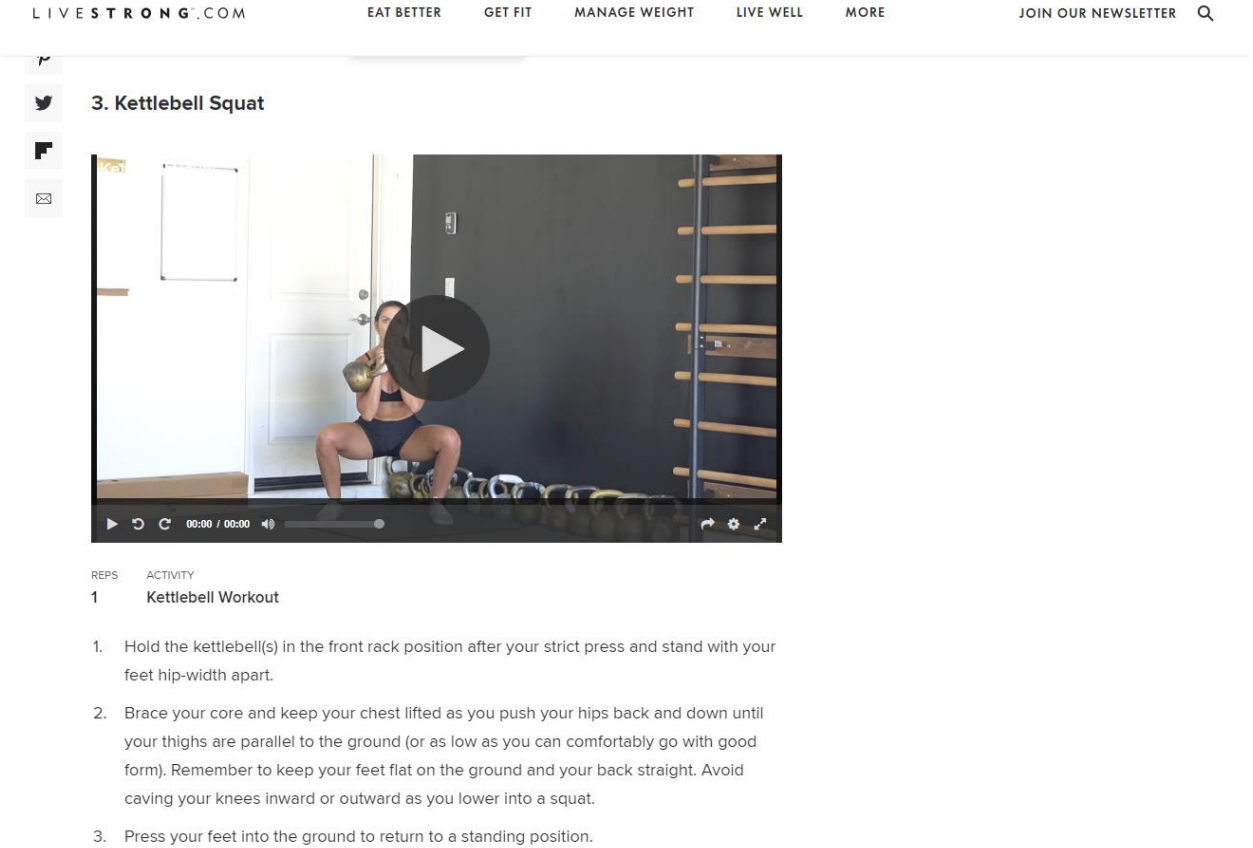


Рисунок 1.5 – Вебсайт Livestrong

1.2 Порівняння аналогів

До порівняння виокремлено Sportlife, Palestra як вебсайти клубів та Livestrong як вебсайт супутньої інформації про заняття спортом (табл. 1.1).

Таблиця 1.1 – Порівняння існуючих програм підтримки роботи з клієнтами клубу для занять спортом

Показник / програма	Sportlife	Livestrong	Palestra	Програма, що розробляється
Інформація про активності клубу	Так	Ні	Так	Так
Інформація про тренерів	Так	Ні	Так	Так
Бронювання занять з тренером	Ні	Ні	Тільки оформлення загальної заявки	Так
Придбання абонементів	Тільки оформлення заявки	Ні	Так	Так
Персоналізація	Ні	Ні	Ні	Так
Поради щодо занять спортом	Ні	Так	Ні	Так
Комплекси вправ	Ні	Так	Ні	Так
Прогнозування кількості абонементів	Ні	Ні	Ні	Так

Відмінностями програми, що розробляється, є можливість прогнозування кількості абонементів, персоналізація, тобто можливість авторизації, реєстрації в різних ролях у програмі, а також наявність на одному вебсайті і комплексів вправ, порад і засобів для клієнтів клубу для занять спортом.

1.3 Висновки за розділом 1

Виконаний початковий огляд проблеми дозволив сформулювати завдання роботи:

- проаналізувати існуючі програми підтримки роботи з клієнтами клубу для занять спортом;
- розробити метод прогнозування кількості активних абонементів для занять спортом;
- виконати проектування програми підтримки роботи з клієнтами клубу для занять спортом;
- реалізувати вебзастосунок підтримки роботи з клієнтами клубу для занять спортом;
- виконати експериментальне дослідження прогнозування кількості активних абонементів для занять спортом.

Результати виконання першого розділу стосуються першого завдання та включають особливості існуючих програм, порівняння їх між собою та з програмою-розробкою.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Задача прогнозування кількості активних абонементів для занять спортом та методи її вирішення

При прогнозуванні показників діяльності клубу для занять спортом одним з найважливіших базових показників є кількість активних абонементів для занять спортом. Це можна пояснити тим, що кожен абонемент – це певна сплачена сума за роботу клубу. Звичайно, що ці суми різні, а вага їх різна. Тобто один абонемент оформлено на місяць, інший на рік, значить вага таких абонементів різна. Тому фактично важливим є те, скільки саме абонементів діє на певну дату. Так, в залежності від того, на який строк було оформлено абонемент, вартість заняття буде відрізнятися, відповідно певна вага у кожного абонементу фактично все одно залишається, але це вже дозволяє більш точно і конкретно враховувати вплив абонементів. Саме тому в підсумку для прогнозування показників діяльності клубу для занять спортом було використано саме кількість активних абонементів для занять спортом, яка має бути встановлено на кожну дату роботи клубу.

Кількість абонементів дозволяє за необхідності оцінити дохід клубу, адже загалом зазвичай відома середня вартість доходу від одного абонементу, тож для конкретного клубу для занять спортом зазвичай не є проблемою перевести кількість абонементів у отриманий у підсумку дохід від занять спортом. Так само відомі зазвичай для клубу для занять спортом і середні витрати на одне заняття. Тож якщо відома кількість абонементів, то можна розрахувати і витрати, а вже з відомих витрат та доходів обчислити очікуваний прибуток. При цьому варто відзначити, що знаючи витрати на одне відвідування клубу, не зовсім коректно одразу з кількості абонементів робити висновок про те, якими саме будуть витрати. Щоб зробити такий висновок, потрібно знати, як перевести кількість абонементів у кількість відвідувань. Загалом може бути відомо на поточний момент середню відвідуваність, але при цьому далі це число може змінюватися. Тобто воно не

залишається статичним. Можна приблизно розрахувати загальні витрати, використовуючи певний середній відсоток відвідуваності клубу. Якщо ж робити більш точні розрахунки, то потрібно прогнозувати не тільки сумарну кількість абонементів на певний момент, а хоча би те, в якій групі відвідуваності яка саме кількість абонементів діє на певний момент. Тоді кількість активних абонементів визначається таким чином, що встановлюються групи відвідуваності клубу для занять спортом. Тобто визначається для кожної групи інтервал значень відсотка відвідуваності занять спортом. Тоді для кожного абонементу має бути визначено, скільки занять за певний період міг відвідати максимально відвідувач у клубі та скільки занять він фактично відвідав. Тоді далі визначенням співвідношення між цими числами (перше – знаменник, друге – чисельник) та множенням отриманого значення на 100 визначається відсоток відвідуваності за абонементом. Далі визначається, до якої саме групи відвідуваності належить кожний абонемент, а для кожної групи визначається кількість абонементів, які цій групі належать.

Прогнозування доходів, витрат, а відповідно і кількості активних абонементів може відбуватися на довгий період часу (наприклад, рік), однак не завжди потрібне тільки таке прогнозування. Адже прогнозування на рік дозволяє виробити певну стратегію, визначити очікування, сформулювати політику. Але далі ці встановлені значення показників зазнають зіштовхування з практикою. А тоді важливо вчасно реагувати на наявні тенденції. Тобто фактично виконувати моніторинг і реагувати на ситуацію. Реакцією на ситуацією може бути розуміння того, що в певний момент зали можуть не вмістити всіх бажаючих або що їх навпаки буде надто мало, тоді можна проводити додаткові заходи, привертати увагу, якщо відвідувачів стає тимчасово менше, або стимулювати відвідувачів приходити в інший час на заняття, якщо кількість відвідувачів надто збільшується.

Тому в роботі задача прогнозування кількості активних абонементів для занять спортом розглядається в розрізі, по-перше, активностей, тобто

фактично залів, які використовуються в клубі для занять спортом, а по-друге, груп за відсотком відвідуваності занять. Для кожної такої групи за певною спортивною активністю має бути визначено кількість активних абонементів на кожен з майбутніх 7 днів, знаючи кількість активних абонементів за минулі 7 днів.

За цією постановкою зрозуміло, що задача прогнозування кількості активних абонементів для занять спортом є задачею прогнозування часового ряду [5]. Для розв'язання таких задач можуть використовуватися різні моделі і методи, включаючи зокрема:

- лінійну регресію;
- дерева рішень, а зокрема їх ансамблі на основі методу Random Forest [6]-[9];
- нейронні мережі;
- рекурентні нейронні мережі, що є спеціалізованим варіантом нейронних мереж;
- нейронні мережі з архітектурою Long Short-Term Memory (LSTM) [10], що є спеціалізацією рекурентних нейронних мереж.

Звичайно, що цей набір можна розширяти, додаючи зокрема і різні варіанти моделей на основі авторегресії, але зазвичай у підсумку, особливо за прогнозування на короткий строк, саме дерева рішень та LSTM демонструють найкращі результати для багатьох задач.

Random Forest може використовуватися для розв'язання і інших задач, зокрема задач класифікації, відповідно універсальність застосування є однією з особливостей цього методу машинного навчання. Він належить до алгоритмів навчання з учителем, а в результаті створює ансамбль (фактично ліс з дерев), для навчання дерев у якому використовується метод на основі створення пакетів (bagging), використовуючи ці пакети для навчання окремих дерев, досягаючи таким чином спеціалізації на певних даних, а потім поєднуючи окремі дерева у ансамбль, а відповідно і результати їхнього навчання і подальшого використання для прогнозування [6]-[9].

LSTM має іншу логіку роботи, яка забезпечує роботи з часовим рядом певної тривалості за рахунок спеціальної структури комірки пам'яті, де шлюзи використовуються для керування потоком інформації, що дозволяє комірці вибірково зберігати або відкидати інформацію за потреби та таким чином уникати проблеми зникнення градієнта, яка характерна для класичного варіанта рекурентної нейронної мережі [10]. Загалом у комірці використовуються три шлюзи, які названі вхідним шлюзом, який відкривається тоді, коли вхід важливий, і закривається тоді, коли він не важливий, шлюзом забуття, який відкривається, коли інформація більше не важлива, і закривається, коли інформація навпаки є важливою і має бути збережена на довший період, і вихідним шлюзом, який відповідно відкривається тоді, коли інформація важлива, і закривається, коли вона неважлива [10]. Усі шлюзи працюють на основі сигмоїдних функцій, які створюють вихідні значення між 0 і 1 [10].

Для оцінювання результатів прогнозування в розрізі задач такого типу використовується середньоквадратична похибка.

2.2 Метод прогнозування кількості активних абонементів для занять спортом

Для розв'язання задачі прогнозування кількості активних абонементів для занять спортом, яку було описано в попередньому підрозділі, представлено метод.

На першому етапі виконання методу потрібно визначити, з якої саме дати розпочав роботу клуб для занять спортом або з якої саме дати розпочато продаж абонементів, або з якої саме дати доступні дані про продаж абонементів. З цих трьох дат визначена або остання за часом має бути визначена як початок періоду навчання. Остання дата, доступна для навчання, відповідно завершує цей період. Це можуть бути всі дані, накопичені в програмі, або це можуть бути тільки виділені для навчання дані.

При проведенні експериментального дослідження до навчальних належала тільки частина даних.

За кожним абонементом має бути встановлено, в який період часу цей абонемент був активним. Тоді для кожного місяця з цього періоду має бути визначено відсоток відвідуваності як співвідношення кількості відвідувань клубу для занять спортом за цей місяць до максимальної кількості відвідувань за цим абонементом за цей період. Якщо в абонементі кількість занять визначена на місяць, а дата початку дії абонементу не є першим числом місяця, то розподіл максимальної кількості на місяця відбувається пропорційно.

Далі має бути визначено кількість активних абонементів за кожною групою відвідуваності за кожною спортивною активністю для кожної дати з періоду навчання. За кожною датою за кожною групою визначається кількість абонементів, які діяли на цю дату та за відсотком відвідуваності занять за цією спортивною активністю на місяць, до якого ця дата належить, потрапляють до цієї групи відвідуваності. Таким чином визначена кількість у підсумку встановлюється для кожної групи за кожною спортивною активністю за кожною датою з періоду навчання.

Якщо за абонементами не доступні дані про відвідування за період (місяць), то тоді дані можуть узагальнюватися за весь час дії абонементу, тобто тоді розділення на місяці не відбувається.

Це в підсумку формує дані, необхідні для формування навчальної вибірки.

На другому етапі створюється набір моделей, кожна з яких буде використовуватися для роботи з конкретною групою відвідуваності занять та конкретною спортивною активністю.

Групи відвідуваності занять спортом:

- ті, хто взагалі не відвідував спортивний центр, маючи абонемент;
- ті, хто відвідував заняття, але відвідав не більше ніж 25 %;
- ті, хто відвідав більше 25 % занять, але не більше ніж 50 %;

– ті, хто відвідав більше ніж 50 % занять.

У процесі дослідження будуть використовуватися такі спортивні активності:

- легка атлетика;
- водні види спорту (басейн);
- фітнес;
- командні види спорту;
- види спорту з ракеткою (ракетні);
- бойові мистецтва;
- заняття людей з обмеженими можливостями.

Моделі створюються на основі одного LSTM-шару та одного повнозв'язного шару. Під час експериментального дослідження 8 LSTM-комірок використано в шарі.

На третьому етапі формуються навчальні вибірки даних для кожної з моделей. Навчальні вибірки визначаються таким чином, що кожен екземпляр складений з 14 значень, кожен з яких визначає нормовану кількість абонементів. 7 днів – минулі значення, наступні 7 днів – майбутні. Кожен наступний екземпляр – це зсув на добу вперед проміжку з 14 діб. Нормування виконується або стандартним чином, або якщо кількість монотонно збільшується, то має визначатися через максимальне значення, встановлене на основі експертного оцінювання.

На четвертому етапі застосовується навчання окремо для кожної з моделей на основі відповідної сформованої вибірки даних.

П'ятий етап застосовується кожен день (раз на добу) для всіх навчених моделей. Для цього формується вхід моделі, що визначається аналогічно формуванню екземпляра навчальної вибірки даних за минулі 7 діб. Тоді виконується прогнозування.

2.3 Висновки за розділом 2

Результати виконання другого розділу стосуються постановки задачі прогнозування кількості активних абонементів для занять спортом, визначення методів її вирішення та формулювання конкретного методу прогнозування кількості активних абонементів для занять спортом.

Наукова новизна роботи полягає в методі прогнозування кількості активних абонементів для занять спортом, який дозволяє прогнозувати кількість абонементів у майбутні дні за групою відвідуваності та спортивною активністю за кількістю абонементів у минулі дні, застосовуючи для побудови нейронних мереж архітектуру LSTM та застосовуючи правила нормування даних.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір засобів розробки

Основний вибір засобів виконувався через вибір мови програмування для реалізації програмного забезпечення підтримки роботи з клієнтами клубу для занять спортом, адже це визначало далі наступні рішення і було фактично основним рішенням. Мова Python [11] обрана для створення програми (табл. 3.1), оскільки вона інтегрує до веброзробки засоби машинного навчання, що потрібно для реалізації методу з розділу 2, автоматизації, що потрібно для визначення статистики абонементів та прогнозування на наступні дні, роботи з даними, що потрібно для формування самої статистики, оброблення наборів даних у програмі, а також враховуючи додаткові переваги стосовно швидкості підтримки і відлагодження.

Таблиця 3.1 – Вибір мови програмування для реалізації програмного забезпечення підтримки роботи з клієнтами клубу для занять спортом

Показник / мова програмування	PHP	Python
Галузі застосування	Веброзробка	Машинне навчання, робота з даними, автоматизація та їх застосування у веброзробці
Відлагодження	Порівняно більший час	Менший час
Швидкість роботи результуючих програм	Вища	Порівняно менша
Організація підтримки результуючих програм	Порівняно складніша	Простіша

3.2 Моделювання програмного забезпечення на основі функціональних вимог

Моделювання програмного забезпечення виконувалось з розрахунку на наявність таких ролей користувачів:

- неавторизований відвідувач, який є звичайним користувачем, який переглядає вебсайт і може авторизуватися для того, щоб отримати притаманні іншій ролі можливості;
- відвідувач, який є авторизованим користувачем і розглядається фактично як користувач, який відвідує клуб для занять спортом або принаймні займається спортом або має бажання розпочати такі заняття;
- тренер, який є авторизованим користувачем і використовує програму для популяризації власних активностей та керування бронюваннями тренувань, які ним проводяться;
- менеджер, який керує і отримує доступ до загальної інформації стосовно клубу для занять спортом та його діяльності.

Сценарії неавторизованого відвідувача (рис. 3.1):

- авторизація облікового запису;
- відображення інформації про тренера;
- відображення фотографій з тренувань;
- відображення інформації про спортивний клуб;
- відображення інформації про вартість абонементів;
- відображення відгуків про тренера;
- відображення комплексу вправ;
- відображення комплексів вправ на групу м'язів;
- пошук комплексів вправ;
- відображення даних зали клубу;
- відображення комплексів вправ від тренера;
- відображення активностей спортивного клубу;
- відображення порад щодо занять спортом від тренера;

- відображення поради щодо занять спортом повністю;
- пошук порад щодо занять спортом.



Рисунок 3.1 – Діаграма варіантів використання програми неавторизованим відвідувачем

Альтернативний сценарій для неавторизованого відвідувача представлений тільки один – це реєстрація облікового запису, після якої він фактично отримує можливість стати відвідувачем (або авторизованим

відвідувачем у повній назві), що реалізується як альтернативний сценарій для авторизації облікового запису.

Відвідувачем неавторизований користувач стає після авторизації (рис. 3.2).



Рисунок 3.2 – Діаграма варіантів використання програми відвідувачем

Сценарії, забезпечувані для роботи відвідувача клубу для занять спортом:

- визначення даних профілю;
- реєстрація тренування в спортивному клубі;
- створення відгуку про тренера;
- відображення власних тренерів відвідувача;
- створення групи тренування;

- відображення власних груп тренування відвідувача;
- відображення історії тренувань в спортивному клубі;
- відображення стану абонементу;
- придбання абонементу для тренування в спортивному клубі;
- відображення власного плану тренувань в спортивному клубі;
- планування індивідуального тренування в спортивному клубі;
- планування спільного тренування в спортивному клубі;
- внесення поради до власної скарбнички;
- відображення власної скарбнички порад;
- подання заявки на тренування з тренером;
- відображення доступних слотів для тренувань у тренера;
- сплата за персональне тренування;
- бронювання заняття з тренером;
- реєстрація стану тренування;
- рекомендування комплексу вправ;
- рекомендування поради від тренера.

Альтернативні сценарії, забезпечувані для роботи відвідувача клубу для занять спортом:

- відмовлення у тренуванні в спортивному клубі, що реалізується як альтернативний сценарій для реєстрації тренування в спортивному клубі за умов, коли відвідувач не має активного абонементу і не може отримати доступ у клуб для занять спортом;

- видалення поради з власної скарбнички, що реалізується як альтернативний сценарій для внесення поради до власної скарбнички за умов, коли порада вже була збережена до скарбнички, але відвідувач не бажає більше її зберігати;

- скасування заняття з тренером, що реалізується як альтернативний сценарій для бронювання заняття з тренером за умов, коли відвідувач забронював заняття, але до моменту його початку зрозумів, що не зможе його відвідати;

– деактивування групи тренування, що реалізується як альтернативний сценарій для створення групи тренування за умов, коли група для спільного тренування вже більше не буде використовуватися, хоча була раніше створена, тобто відвідувачів не мають більше бажання відвідувати заняття разом;

– редагування групи тренування, що реалізується як альтернативний сценарій для створення групи тренування за умов, коли група тренування була створена, але її дані потребують оновлення, зокрема можливо змінюється склад відвідувачів;

– скасування індивідуального тренування в спортивному клубі, що реалізується як альтернативний сценарій для планування індивідуального тренування в спортивному клубі за умов, коли відвідувач запланував відвідування клубу, але більше не має бажання або можливості зробити це у зазначений час;

– скасування спільного тренування в спортивному клубі, що реалізується як альтернативний сценарій для планування спільного тренування в спортивному клубі за умов, коли відвідувач зі складу групи тренування не зможе відвідати спільне тренування, яке раніше було заплановано;

– скасування рекомендації комплексу вправ, що реалізується як альтернативний сценарій для рекомендування комплексу вправ за умов, коли відвідувач виставив рекомендацію комплексу вправ, але більше не бажає вважати цей комплекс рекомендованим;

– скасування рекомендації поради від тренера, що реалізується як альтернативний сценарій для скасування поради від тренера за умов, коли відвідувач виставив рекомендацію поради від тренера, але більше не бажає вважати цю пораду рекомендованою.

Сценарії, забезпечувані для роботи тренера:

– авторизація;

– визначення даних профілю;

- реєстрація стану тренування тренером;
- відображення зареєстрованих на тренуванні на поточний момент;
- відображення задіяності тренера;
- відображення відгуків про тренера;
- відображення відвідувачів у тренера;
- відображення плану тренувань в спортивному клубі;
- відображення комплексу вправ;
- відображення власних комплексів вправ;
- публікація комплексу вправ;
- відображення власних порад щодо занять спортом;
- публікація поради щодо занять спортом;
- додавання слоту для тренування;
- відображення заявок на тренування тренеру;
- відображення історії тренувань відвідувача у тренера;
- підтвердження заявки;
- відображення доступних слотів для тренувань;
- відображення поради щодо занять спортом повністю;
- визначення тарифу на персональне заняття.

Альтернативні сценарії (рис. 3.3), забезпечувані для роботи тренера:

- редагування комплексу вправ від тренера, що реалізується як альтернативний сценарій для публікації комплексу вправ за умов, коли комплекс вправ вже було опубліковано, але тренер має потребу внести зміни;
- редагування поради щодо занять спортом, що реалізується як альтернативний сценарій для публікації поради щодо занять спортом за умов, коли пораду щодо занять спортом вже було опубліковано, але тренер має потребу внести зміни;
- деактивація поради щодо занять спортом, що реалізується як альтернативний сценарій для публікації поради щодо занять спортом за умов, коли порада більше не є актуальною;

- редагування слоту для тренування, що реалізується як альтернативний сценарій для додавання слоту для тренування за умов, коли слот вже було створено, але тепер має бути змінено його параметри;
- вилучення слоту для тренування, що реалізується як альтернативний сценарій для додавання слоту для тренування за умов, коли слот вже було створено, але тренер не може його виконати;
- скасування заявки, що реалізується як альтернативний сценарій для підтвердження заявки за умов, коли тренер не може або не бажає тренувати відповідного відвідувача.



Рисунок 3.3 – Діаграма варіантів використання програми тренером

Сценарії, забезпечувані для роботи менеджера:

- авторизація;
- визначення даних профілю;
- відображення інформації про тренера;
- реєстрація відвідувача;
- реєстрація менеджера;
- реєстрація тренера;
- відображення тренерів у спортивному клубі;
- відображення менеджерів у спортивному клубі;
- пошук відвідувачів;
- додавання активності спортивного клубу;
- відображення зареєстрованих на тренуванні на поточний момент;
- відображення фотографій з тренувань;
- додавання фотографії з тренувань;
- відображення інформації про спортивний клуб;
- відображення інформації про вартість абонементів;
- додавання пропозиції абонементу;
- відображення задіяності тренера;
- відображення статистики використання абонементів;
- відображення статистики відвідування спортивної активності;
- відображення статистики активних абонементів;
- прогнозування кількості активних абонементів;
- відображення відгуків про тренера;
- відображення відвідувачів у тренера;
- відображення історії тренувань в клубі за відвідувачем;
- відображення історії тренувань в спортивному клубі за активністю;
- відображення історії тренувань в спортивному клубі за тренером;
- відображення стану абонементу відвідувача;
- відображення плану тренувань в спортивному клубі за відвідувачем;
- відображення плану тренувань в спортивному клубі за активністю;

- відображення плану тренувань в спортивному клубі за тренером;
- відображення комплексу вправ;
- відображення комплексів вправ на групу м'язів;
- пошук комплексів вправ;
- відображення комплексів вправ від тренера;
- відображення активностей спортивного клубу;
- відображення порад щодо занять спортом від тренера;
- відображення доступних слотів для тренувань у тренера;
- відображення поради щодо занять спортом повністю;
- пошук порад щодо занять спортом.

Альтернативні сценарії (рис. 3.4), забезпечувані для роботи менеджера:

- деактивація облікового запису тренера, що реалізується як альтернативний сценарій для реєстрації тренера за умов, коли тренер більше не працює в спортивному клубі або порушує правила роботи;
- деактивація облікового запису менеджера, що реалізується як альтернативний сценарій для реєстрації менеджера за умов, коли менеджер більше не працює в спортивному клубі або порушує правила роботи;
- деактивація облікового запису відвідувача, що реалізується як альтернативний сценарій для реєстрації відвідувача за умов, коли відвідувач порушив правила роботи або йому заборонено доступ у спортивний клуб;
- редагування активності спортивного клубу, що реалізується як альтернативний сценарій для додавання активності спортивного клубу за умов, коли певна спортивна активність в клубі надається, але змінилися умови;
- видалення фотографії з тренувань, що реалізується як альтернативний сценарій для додавання фотографії з тренувань за умов, коли фотографія вже була додана, але вона не має з'являтися у вебзастосунку більше;
- редагування інформації про спортивний клуб, що реалізується як

альтернативний сценарій для відображення інформації про спортивний клуб за умов, коли інформації не відповідає дійсності.

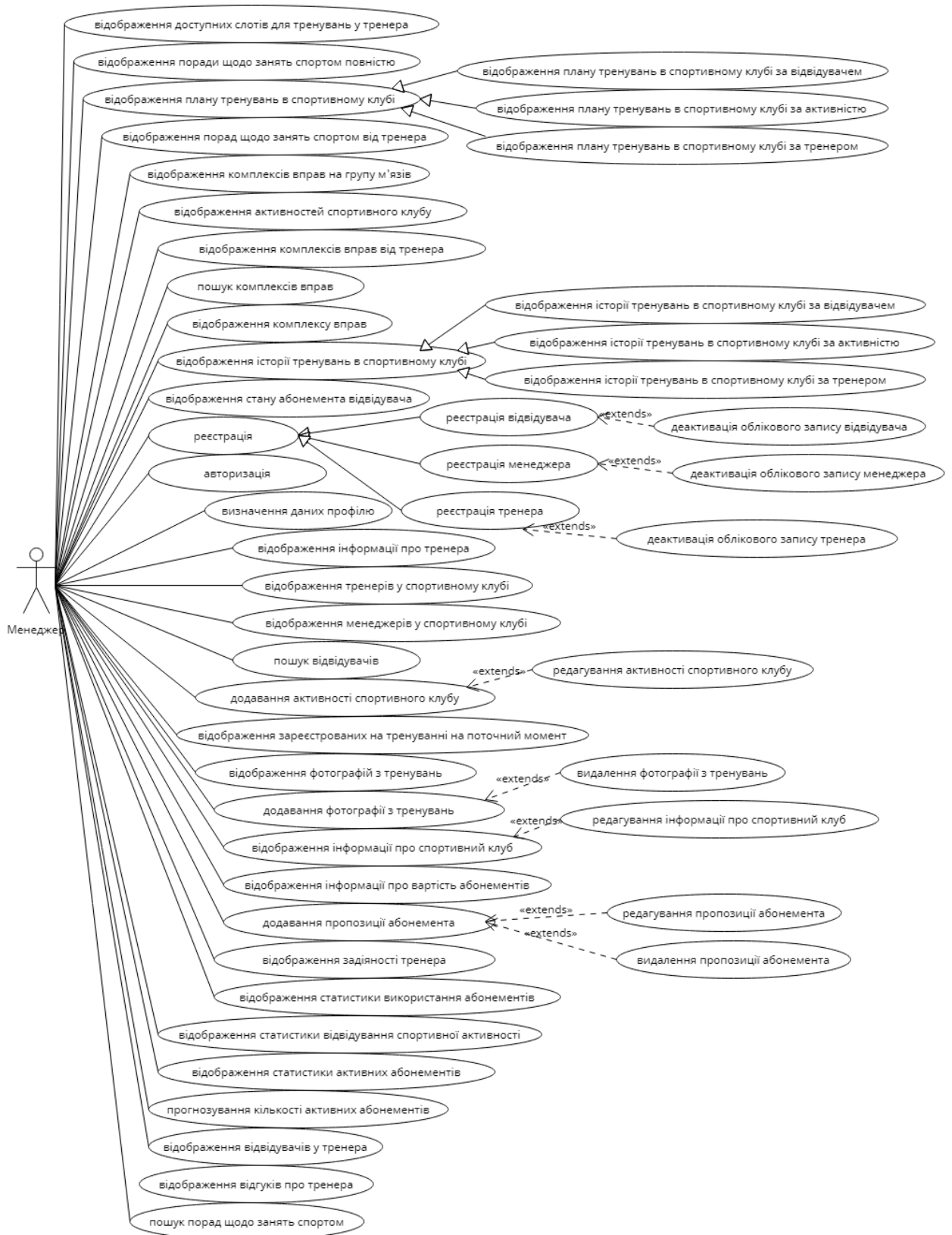


Рисунок 3.4 – Діаграма варіантів використання програми менеджером

Останніми двома альтернативними сценаріями для менеджера є:

- видалення пропозиції абонементу, що реалізується як альтернативний сценарій для додавання пропозиції абонементу за умов, коли існуюча пропозиція більше не доступна і не може бути замовлена;
- редагування пропозиції абонементу, що реалізується як альтернативний сценарій для додавання пропозиції абонементу за умов, коли існуюча пропозиція починає надаватися на нових умовах.

3.3 Проектування структури бази даних

Робота з даною організована в програмі через створення бази даних (БД) за допомогою системи PostgreSQL та використання JavaScript Object Notation (JSON)-файлу. JSON-файл необхідний для збереження даних про клуб для занять спортом. Оскільки такий клуб один, а вся діяльність у програмі відбувається навколо його активностей, можливостей, то немає необхідності зберігати дані декількох клубів, відповідно немає необхідності створювати таблицю БД. Саме тому ці дані винесені в JSON-файл.

Однак, не зважаючи на те, що клуб для занять спортом тільки один, але він може складатися з декількох складових одиниць: наприклад, спортивного залу, тенісного корту, басейну, стадіону тощо. Інформація про ці зали знаходиться в таблиці `stansport_hall`.

Клуб для занять спортом надає можливість займатися різними активностями. Ці активності можуть включати заняття фітнесом, плаванням, різними командними видами спорту тощо. Інформація про ці активності, які надаються клубом, знаходиться в таблиці БД `stansport_activity`.

Кожна активність виконується в певному залі, проте не в декількох, а тільки в одному, а відповідно в одному залі може реалізовуватися декілька активностей. Тож зв'язок між таблицями `stansport_hall` і `stansport_activity` є зв'язком, тип якого встановлений як один-до-багатьох.

Дані про всіх користувачів відповідають таблиці БД `stansport_user`, в якій зберігаються облікові дані користувачів у відповідності з базовою моделлю Django, яку було розширено. Оскільки ця модель включає поле щодо активності облікового запису або його деактивації (`is_active`), доступу до адміністративної панелі, яку має менеджер (`is_staff`), а також адміністратор, але адміністратор у полі `is_superuser` має істинне значення. Базова користувацька інформація включає логін в полі `username`, ім'я в полі `first_name`, прізвище в полі `last_name`, електронну пошту в полі `email` та пароль в полі `password`. Також час останньої авторизації за користувачем визначається в полі `last_login`, а час реєстрації в полі `date_joined`. Фотографія людини зберігається в полі `photo`, а секретний унікальний номер в полі `code`. У свою чергу тренери та відвідувачі мають облікові записи, які за цими параметрами нічим не відрізняються. Оскільки про звичайного відвідувача інші дані іншим користувачам не потрібні, то тренер був відділений в окрему сутність, де має знаходитися додаткова інформація, що включає його досвід, опис тощо. Відповідно якщо для користувача є додатковий запис у таблиці тренерів `stansport_trainer`, то він є тренером. Якщо ні, то відвідувачем.

Кожен тренер повинен мати обліковий запис для роботи з програмою, але він не може мати декілька таких записів. Кожен обліковий запис стосується в свою чергу лише одного тренера. Тож зв'язок між таблицями `stansport_trainer` і `stansport_user` є зв'язком, тип якого встановлений як один-до-одного.

Тренери можуть виконувати тренування не за будь-якою активністю, яка є в клубі, а тільки за тими, за якими менеджер встановив для них можливість проведення тренувань. Проте кожен тренер може проводити тренування за однією або декількома активностями, але і тренування за різними активностями можуть проводитися різними тренерами, тож зв'язок виходить багато-до-багатьох, а реалізовано його через таблицю `stansport_traineractivity`.

Реєстрація є однією з сутностей. Це не дія з реєстрації облікового запису, а це реєстрація в системі під час відвідування певної активності в клубі для занять спортом. Тобто кожне відвідування спортивного клубу фіксується. Ці дані зберігаються в таблиці `stansport_enrollment`.

Кожна реєстрація відбувається стосовно тільки одного відвідувача, але відвідувачі можуть за весь час роботи декілька реєстрацій. Тож зв'язок між таблицями `stansport_user` і `stansport_enrollment` є зв'язком, тип якого встановлений як один-до-багатьох. Кожна реєстрація може стосуватися тільки однієї конкретної активності, але одна спортивна активність передбачає багаторазові реєстрації різних відвідувачів. Тож зв'язок між таблицями `stansport_activity` і `stansport_enrollment` є зв'язком, тип якого встановлений як один-до-багатьох.

Але перед тим, як мати можливість зареєструватися, потрібно отримати доступ до клубу. Для цього надаються абонементи, один з яких має придбати відвідувач для того, щоб мати можливість зареєструвати своє відвідування. Інформація про доступні варіанти абонементів знаходиться в таблиці `stansport_subscription`. Варіанти абонементів відрізняються тривалістю (кількістю місяців) та кількістю тренувань, які доступні за місяць. Окрім того окремий вид абонементу – разовий.

Абонементи можуть надаватися на всі види активностей, а можуть стосуватися тільки однієї або декількох. Наприклад, це може бути абонемент на відвідини басейну, а інші активності при цьому не включаються. Тож виходить, що один абонемент може стосуватися різних активностей, а одна активність може бути визначена в різних варіантах абонементів, тому зв'язок виходить багато-до-багатьох, а реалізовано його таблицею `stansport_subscriptionactivity`.

Однак варіанти абонементів є тільки пропозиціями для придбання, а саме придбання відвідувачем абонементу є окремою сутністю, дані якої зберігаються в таблиці `stansport_subscriptionpurchase`. Відвідувач у різні періоди часу може придбати різні варіанти абонементів, при цьому кожне

окреме придбання абонементу стосується лише одного конкретного відвідувача. Тож зв'язок між таблицями `stansport_user` і `stansport_subscriptionpurchase` є зв'язком, тип якого встановлений як один-до-багатьох.

Після придбання абонементу відвідувач також може винайняти персональне тренування. Саме для цього в клубі і присутні тренери. Персональні тренування також мають свою вартість. Її встановлюють кожен тренер окремо. І вартість залежить від кількості тренувань. Конкретні пропозиції від тренерів знаходяться в таблиці `stansport_trainingtariff`. Тренери можуть мати по декілька пропозицій щодо вартості тренувань, але кожна пропозиція визначається лише одним тренером. Тож зв'язок між таблицями `stansport_trainer` і `stansport_trainingtariff` є зв'язком, тип якого встановлений як один-до-багатьох.

Для того щоб забронювати тренування, спочатку потрібно оформити заявку, а тренер має її підтвердити. Інформація про заявки зберігається в таблиці `stansport_personalapplication`. Після підтвердження заявки потрібно сплатити за тренування. Оскільки варіантів оплати в таблиці `stansport_trainingtariff` декілька за кожним тренером або щонайменше один, то придбання конкретного варіанта зберігається в таблиці `stansport_trainingpurchase`.

Один відвідувач може мати багато заявок до різних тренерів щодо занять спортом на основі персональних тренувань, але кожна заявка оформлюється одним відвідувачем. Тож зв'язок між таблицями `stansport_user` і `stansport_personalapplication` є зв'язком, тип якого встановлений як один-до-багатьох. Один відвідувач може придбати декілька разів тренування з тренером, але кожне придбання оформлюється тільки стосовно одного конкретного тренера, то зв'язок між таблицями `stansport_user` і `stansport_subscriptionpurchase` є зв'язком, тип якого встановлений як один-до-багатьох.

Тренери проводять тренування не в будь-який можливий момент, а самостійно визначають часові рамки тренувань. Ці часові рамки представляються у вигляді слотів для занять, які потім обираються відвідувачі. Слоти для занять знаходяться в таблиці БД `stansport_trainingslot`.

Слоти для тренування стосуються навіть не самого тренера, а тренера за певною активністю. Тобто слот визначається не на тренування за будь-якою активністю, тренування за якою здійснює цей тренер, а не конкретну активність, яку визначив тренер. За кожною активністю тренера може бути визначено багато слотів, але кожен слот стосується лише однієї спортивної активності тренера. Тож зв'язок між таблицями `stansport_traineractivity` і `stansport_trainingslot` є зв'язком, тип якого встановлений як один-до-багатьох.

Безпосередньо бронювання оформлюються відвідувачами у межах конкретного слоту. Ці бронювання визначаються в таблиці `stansport_slotbooking`. До бронювання тренер або відвідувач можуть визначити, які саме вправи та скільки разів виконувались під час заняття, якщо вважають це за потрібне. Відповідні вправи за заняттям з бронювання містяться в таблиці `stansport_slotbookingdetail`.

Бронювання завжди має автора, яким є один відвідувач. Кожен відвідувач може мати багато різних бронювань. Тож зв'язок між таблицями `stansport_user` і `stansport_slotbooking` є зв'язком, тип якого встановлений як один-до-багатьох. Відповідно кожен слот для тренування може містити декілька бронювань, адже тренер може визначити можливість тренування декількох відвідувачів одночасно під час персонального тренування, але кожне бронювання може стосуватися лише одного слота. Тож зв'язок між таблицями `stansport_trainingslot` і `stansport_slotbooking` є зв'язком, тип якого встановлений як один-до-багатьох.

Про тренера відвідувач після заняття у нього може залишити відгук. Відгуки про тренерів вносяться в таблицю `stansport_trainereview`. Кожен відвідувач може мати різні відгуки про тренерів, але кожен відгук має лише одного автора. Тож зв'язок між таблицями `stansport_user` і

stansport_trainereview є зв'язком, тип якого встановлений як один-до-багатьох. Кожен тренер може мати різні відгуки про себе, але кожен відгук визначає думку тільки про одного тренера. Тож зв'язок між таблицями stansport_trainer і stansport_trainereview є зв'язком, тип якого встановлений як один-до-багатьох.

Тренери окрім проведення занять у програмі можуть визначати власні комплекси вправ. Комплекси вправ від тренерів зберігаються в таблиці stansport_workout. Кожен комплекс вправ має тільки одного автора, яким є тренер, але один тренер може мати декілька комплексів вправ. Тож зв'язок між таблицями stansport_trainer і stansport_workout є зв'язком, тип якого встановлений як один-до-багатьох.

Окрім комплексів вправ тренери також можуть визначати поради щодо тренувань, створюючи для цього статті, де описують ці поради. Для збереження поради в БД використовується таблиця stansport_advice. Кожна порада про заняття спортом має тільки одного автора, яким є тренер, але один тренер може мати декілька порад. Тож зв'язок між таблицями stansport_trainer і stansport_advice є зв'язком, тип якого встановлений як один-до-багатьох.

І поради, і комплекси вправ потрібно певним чином впорядковувати, щоб потім можна було організувати пошук, структурувати ці дані. Для цього необхідно було ввести дані про м'язи. Таким чином можна і поради, і комплекси вправ співвідносити з м'язами. Люди відповідно бажають розвинути, підтримувати необхідний стан м'язів, а значить це і один із основних критеріїв пошуку комплексів вправ та порад. Інформація про внесені в програму м'язи знаходиться в таблиці stansport_muscle.

Оскільки кожна порада тренера може стосуватися декількох м'язів, а так само і кожен комплекс вправ задіює декілька м'язів, а кожен м'яз відповідно може бути представлений в різних порадах та комплексах вправ, то для реалізації цих зв'язків між відповідними таблицями створено окремі таблиці stansport_workoutmuscle і stansport_advicemuscle. Ці таблиці містять і

додаткові характеристики про те, як саме задіяно відповідні м'язи у комплексі вправ і поради, але за своєю суттю це реалізація зв'язку багато-до-багатьох.

Кожен комплекс вправ може містити декілька вправ. Але для зручності визначення переліку вправ потрібно спочатку визначити конкретні вправи, а потім пропонувати їх додати під час визначення комплексу вправ, пришвидшуючи процес внесення даних. Доступні вправи з занять спортом знаходяться в таблиці `stansport_exercise`. Інформація ж про те, які саме вправи, в якій саме кількості задіяні у відповідному комплексі вправ знаходиться в таблиці `stansport_workoutplan`. Кожна вправа може входити в різні комплекси вправ, а кожен комплекс вправ може складатися з різних вправ, тож виходить, що сама по самі таблиця `stansport_workoutplan` є реалізацією зв'язку багато-до-багатьох.

Тренування можуть відбуватися самостійно в індивідуальному форматі. Для цього достатньо придбати абонемент. Тоді відвідувач далі може планувати, коли саме він бажає відвідати клуб для занять спортом. Надаючи таку інформацію, відвідувачі можуть отримувати інформацію про те, скільки ще людей планують відвідати клуб в цей час. Ці заплановані відвідини в індивідуальному форматі зберігаються в таблиці `stansport_planenrollment`. Кожне індивідуальне тренування стосується тільки одного відвідувача, але відвідувач може в різний час мати різні такі тренування. Тож зв'язок між таблицями `stansport_user` і `stansport_planenrollment` є зв'язком, тип якого встановлений як один-до-багатьох. За кожною активністю можуть плануватися різні індивідуальні тренування, а кожне заплановане тренування стосується тільки однієї конкретної спортивної активності. Тож зв'язок між таблицями `stansport_activity` і `stansport_planenrollment` є зв'язком, тип якого встановлений як один-до-багатьох.

Однак окрім того відвідувачі можуть займатися в групах. Це необхідно, щоб запланувати потім відвідини клубу для занять спортом з друзями. Для таких відвідин спочатку має бути визначено групу для занять

спортом. Самі групи зберігаються в таблиці `stansport_traininggroup`. Кожен відвідувач може створити декілька груп для тренування, але одна група для тренування має тільки одного відвідувача як власника цієї групи. Тож зв'язок між таблицями `stansport_user` і `stansport_traininggroup` є зв'язком, тип якого встановлений як один-до-багатьох.

Оскільки кожна група тренування може, а фактично логічно і має містити певну кількість відвідувачів (2 і більше), а кожен відвідувач може входити в різні групи, адже в різні дні, в різні моменти часу може відвідувати клуб для занять спортом з різними іншими людьми, то виходить, що зв'язок має тип багато-до-багатьох. Тому реалізацію цього зв'язку виконано через таблицю `stansport_traininggroupperson`.

Заплановані тренування у складі групи знаходяться в таблиці `stansport_plangroupenrollment`. Однак, для того щоб створити тренування в групі, запланувати його, недостатньо тільки визначити час тренування. Потрібно, щоб інші учасники тренування підтвердили свою участь. Ці дані зберігаються в таблиці `stansport_plangroupenrollmentdetail`. Кожне заплановане тренування у складі групи має бути ініційовано одним з учасників групи, але кожен відвідувач може планувати різні заняття в групі в різний час. Тож зв'язок між таблицями `stansport_user` і `stansport_plangroupenrollment` є зв'язком, тип якого встановлений як один-до-багатьох. Кожне заняття в групі може стосуватися тільки однієї спортивної активності. Кожна активність може мати різні заплановані заняття в групах у різний час. Тож зв'язок між таблицями `stansport_activity` і `stansport_plangroupenrollment` є зв'язком, тип якого встановлений як один-до-багатьох. Окрім загального плану тренування в групі визначається деталізація участі відвідувача з групи, тобто кожен відвідувач може мати різні такі деталізації, але кожна деталізація стосується тільки одного відвідувача. Тож зв'язок між таблицями `stansport_user` і `stansport_plangroupenrollmentdetail` є зв'язком, тип якого встановлений як один-до-багатьох. Кожне тренування в групі може мати декілька деталізацій

щодо участі різних відвідувачів у тренуванні, але кожна деталізація може стосуватися лише одного тренування в групі. Тож зв'язок між таблицями `stansport_plangroupenrollment` і `stansport_plangroupenrollmentdetail` є зв'язком, тип якого встановлений як один-до-багатьох.

Поради, які надаються тренерами, можна зберігати, створюючи власну скарбничку порад. Збереження порад відбувається через внесення їх за відвідувачем до таблиці `stansport_box`. Окрім збереження в скарбничку поради відвідувач також може рекомендувати раду. Такі рекомендації знаходяться в таблиці `stansport_advicerecommendation`. Кожна рекомендація може мати тільки одного автора, яким є відвідувач, але кожен відвідувач може давати різні рекомендації різним порадам. Тож зв'язок між таблицями `stansport_user` і `stansport_advicerecommendation` є зв'язком, тип якого встановлений як один-до-багатьох. Кожна рекомендація стосується тільки однієї поради, але кожна рада може мати багато рекомендацій. Тож зв'язок між таблицями `stansport_advice` і `stansport_advicerecommendation` є зв'язком, тип якого встановлений як один-до-багатьох.

Оскільки прогнозування кількості активних абонементів для занять спортом, а також і надання статистики про відвідування спортивної активності і активні абонементи може відбуватися в різні моменти часу для різних користувачів, то не варто всі ці обчислення весь час повторювати, адже вони охоплюють по декілька таблиць БД, передбачають витягання великого масиву даних, що є ресурсовитратними операціями. Якщо прогнозування має відбуватися раз на день, то якщо результати прогнозування не зберігати, його потрібно виконувати кожен раз, коли менеджер звертається за відповідними даними. Окрім того, щоб виконати таке прогнозування, потрібно накопичити вибірку даних для навчання. Навіть якщо навчання відбувається один раз, то подальше прогнозування потребує виділення деякого часового ряду для подання на вхід, а це вже охоплює фактично аналіз всіх абонементів, які взагалі присутні в БД. Якщо розглядати не прогнозування, а вже минулі дані, тобто статистику за тим, що

відбулось, то кожне звернення менеджера буде призводити до виконання таких ресурсовитратних операцій, що буде перевантажувати БД. Щоб це постійно не виконувати, а доступ до даних отримувати швидко, було введено таблицю `stansport_statisticssubscription` для збереження даних щодо статистика абонементів та прогнозування їх кількості.

Схематичне представлення відповідних таблиць та залежностей між ними приведено на рис. 3.5.

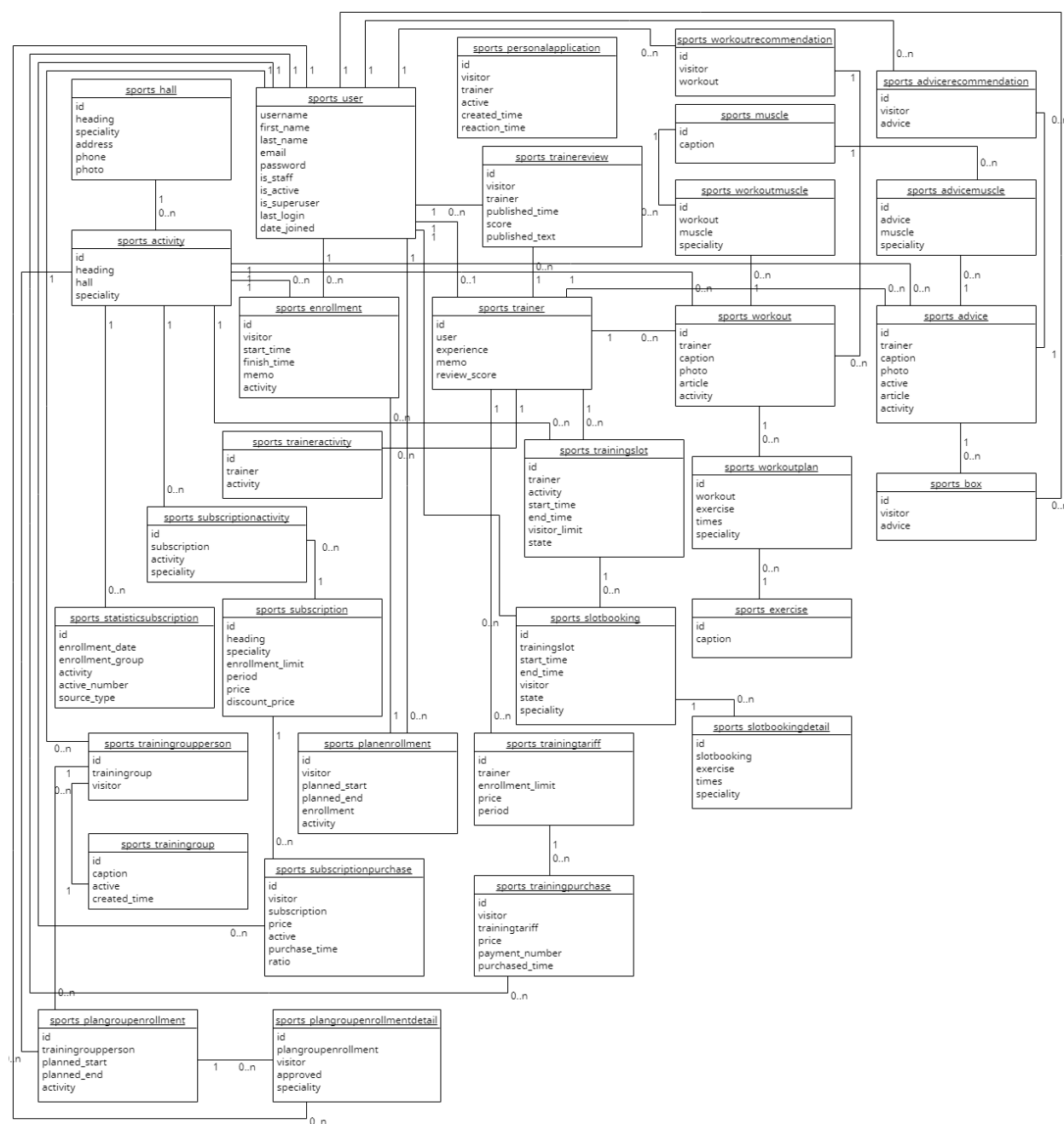


Рисунок 3.5 – Схема БД

Кожен запис щодо абонементів на день стосується тільки однієї активності, а одна активність може мати багато записів на різні дні. Тож зв'язок між таблицями `stansport_activity` і `stansport_statisticssubscription` є зв'язком, тип якого встановлений як один-до-багатьох.

Рекомендації можуть стосуватися також і комплексів вправ. Ці рекомендації знаходяться в таблиці `stansport_workoutrecommendation`. Кожна рекомендація може мати тільки одного автора, яким є відвідувач, але кожен відвідувач може давати різні рекомендації різним комплексам вправ. Тож зв'язок між таблицями `stansport_user` і `stansport_workoutrecommendation` є зв'язком, тип якого встановлений як один-до-багатьох. Кожна рекомендація стосується тільки одного комплексу вправ, але кожен комплекс вправ може мати багато рекомендацій. Тож зв'язок між таблицями `stansport_workout` і `stansport_workoutrecommendation` є зв'язком, тип якого встановлений як один-до-багатьох.

3.4 Висновки за розділом 3

Результати виконання третього розділу стосуються опрацювання вимог до програми зі створенням діаграм варіантів використання для всіх основних ролей користувачів, що включають неавторизованого відвідувача, відвідувача, тренера і менеджера, визначення на основі інтерпретації загальної логіки програми та вимог до програми структури даних за допомогою БД, для якої було сформульовано сутності, виділено на їх основі таблиці, зв'язки між ними, вибору мови програмування python для реалізації програми.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Порядок роботи програмного забезпечення

Порядок роботи з програмою визначається послідовностями взаємодії між різними учасниками тренувального процесу в розрізі програмних користувацьких інтерфейсів та сценаріїв роботи. Послідовність взаємодії між відвідувачем та тренером охоплює зокрема і варіанти, коли відвідувач тренується самостійно і використовує програму індивідуально. Таку взаємодію зручно показувати у вигляді діаграми послідовності (рис. 4.1).

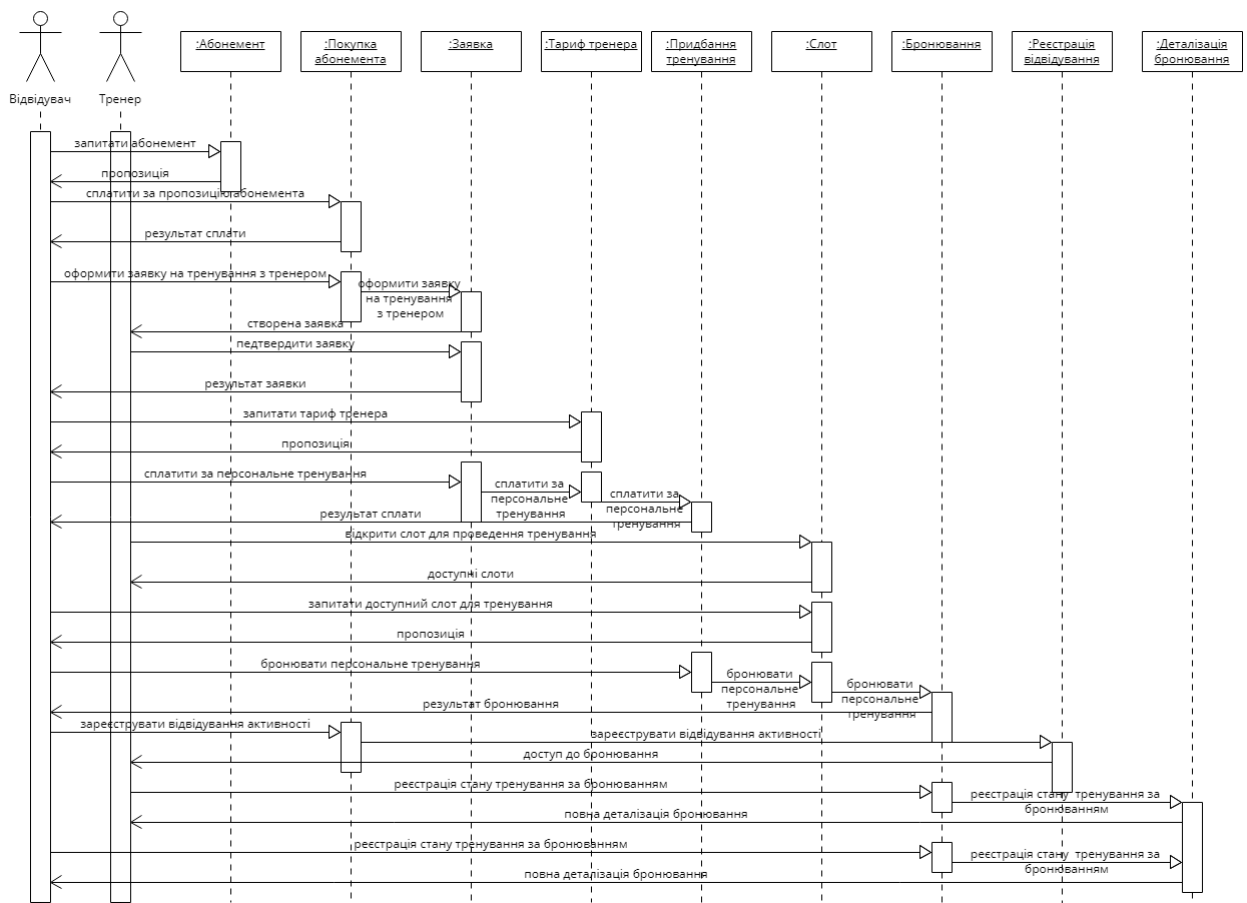


Рисунок 4.1 – Діаграма послідовності для потоку взаємодії між відвідувачем і тренером стосовно проведення індивідуального персонального тренування

У процесі взаємодії спочатку відвідувач має визначитись з абонементом, купити один з варіантів абонента, тільки тоді він зможе подавати заявки тренеру. Якщо тренер схвалив заявку, то він отримує можливість обрати тариф для персональних тренувань та сплатити за нього. Після того, як тариф сплачено, поки він діє, відвідувач отримує можливість бронювати тренування. Бронювання відбуваються в межах тих слотів, які визначив тренер. Після того, як за наявності бронювання, відвідувач зареєструвався в клубі, то тренер отримує можливість деталізувати бронювання, вносячи вправи, які виконуються під час тренування. Після тренування відвідувач може самостійно додати дані до деталізації або змінити їх.

У випадку тренувань без тренера відвідувач має абонемент, а тоді вже планує відвідини, якщо бажає це зробити, реєструється в клубі. У випадку тренувань у групі за наявності абонементу спочатку має бути створено групу, підтверджено участь всіх запрошених учасників, а тоді заплановано тренування, підтверджено участь всіх або частини учасників, а тоді вже відбувається реєстрація в клубі та саме тренування.

4.2 Рішення з реалізації вебзастосунку

Всі прийняті рішення для реалізації вебзастосунку включають використання фреймворку Django, який базується на використанні шаблону проєктування Model View Template (MVT) [12]. Це відповідно визначає структуру вебзастосунку, до якої також додаються скрипти для автоматичного підведення статистики кожен день та прогнозування кількості активних абонементів, побудови моделей прогнозування.

Шаблони вебсторінок за MVT створювались головним чином за тими сценаріями, які були представлені в розділі 3, з визначенням макетів та стилів сторінок у відповідності з безкоштовним вільним шаблоном [13].

Клас `Trainer` визначає реалізацію моделі тренера, яка включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором тренера, який визначено через тип `BigAutoField`.

Поле моделі `user` формує обліковий запис, який використовує тренер для роботи з програмою і який визначено через тип `OneToOneField`.

Поле моделі `experience` формує кількість років досвіду роботи тренера, який визначено через тип `PositiveSmallIntegerField`.

Поле моделі `memo` формує візитівку про тренера і визначено через тип `CharField`.

Поле моделі `review_score` формує оцінку, яку в середньому було виставлено тренеру за всіма відгуками про нього, визначене через тип `DecimalField`.

Поле моделі `activities` формує активності, за якими проводить заняття тренер, реалізовано через тип `ManyToManyField`.

Клас `Enrollment` визначає реалізацію моделі реєстрації відвідування клубу для занять спортом, яка включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором реєстрації відвідування клубу для занять спортом, який визначено через тип `BigAutoField`.

Поле моделі `visitor` формує дані відвідувача, який зареєстрував відвідини спортивного клубу, визначено через тип `ForeignKey`.

Поле моделі `start_time` формує час реєстрації, тобто початку відвідування клубу для занять спортом, визначено через тип `DateTimeField`.

Поле моделі `finish_time` формує час закінчення відвідування клубу для занять спортом, визначено через тип `DateTimeField`.

Поле моделі `memo` формує додатковий опис тренування, визначено через клас `CharField`.

Поле моделі `activity` формує активність, за якою відбувається заняття спортом, визначено через тип `ForeignKey`.

Клас Subscription визначає реалізацію моделі абонементу на заняття спортом, яка включає наступні поля.

Поле моделі id є унікальним ідентифікатором пропозиції абонементу, який визначено через тип BigAutoField.

Поле моделі heading формує назву абонементу, визначено через тип CharField.

Поле моделі speciality формує детальний опис пропозиції абонементу, визначено через тип CharField.

Поле моделі enrollment_limit формує максимальну кількість занять спортом, яку можна відвідати за цією пропозицією абонементу протягом місяця, визначено через тип PositiveSmallIntegerField.

Поле моделі period формує кількість місяців, протягом якої діє абонемент (відповідно за кожен місяць максимальна кількість занять дорівнює enrollment_limit, а у випадку, якщо в полі period значення 0, то відповідно абонемент на одне відвідування клубу для занять спортом), визначено через тип PositiveSmallIntegerField.

Поле моделі price формує суму, яку потрібно сплатити за цей абонемент, визначено через тип IntegerField.

Поле моделі discount_price формує суму, яку потрібно сплатити за цю пропозицію абонементу у випадку наявності знижки, визначено через тип IntegerField.

Клас Hall визначає реалізацію моделі зали клубу для занять спортом, яка включає наступні поля.

Поле моделі id є унікальним ідентифікатором зали для занять спортом, який визначено через тип BigAutoField.

Поле моделі heading формує назву зали для занять спортом, визначено через тип CharField.

Поле моделі speciality формує детальний опис зали для занять спортом, визначено через тип CharField.

Поле моделі address є представленням адреси зали для занять спортом, визначено через тип CharField.

Поле моделі phone є представленням номеру телефону зали клубу для занять спортом.

Поле моделі photo формує фотографію, якою ілюструється зала клубу, визначено через тип ImageField.

Клас Activity визначає реалізацію моделі спортивної активності, яка включає наступні поля.

Поле моделі id є унікальним ідентифікатором спортивної активності, який визначено через тип BigAutoField.

Поле моделі heading формує назву спортивної активності, визначено через тип CharField.

Поле моделі hall формує залу клубу для занять спортом, де відбувається ця спортивна активність, визначено через тип ForeignKey.

Поле моделі speciality формує детальний опис спортивної активності, визначено через тип CharField.

Клас Subscriptionactivity визначає реалізацію моделі спортивної активності стосовно конкретного абонементу, яка включає наступні поля.

Поле моделі id є унікальним ідентифікатором спортивної активності стосовно конкретного абонементу, який визначено через тип BigAutoField.

Поле моделі subscription є пропозицією абонементу, стосовно якої забезпечується ця спортивна активність та конкретизується, визначено через тип ForeignKey.

Поле моделі activity формує спортивну активність, яка забезпечується в цьому клубі для занять спортом за цією пропозицією абонементу, визначено через тип ForeignKey.

Поле моделі speciality формує особливості спортивної активності в розрізі цієї пропозиції абонементу, визначено через тип CharField.

Клас `StatisticsSubscription` визначає реалізацію моделі узагальнення даних про абонементи в клубі для занять спортом, яка включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором узагальнення даних про абонементи в клубі для занять спортом, який визначено через тип `BigAutoField`.

Поле моделі `enrollment_date` формує дату, в яку узагальнено дані про кількість абонементів, визначено через тип `DateField`.

Поле моделі `enrollment_group` формує групу за відсотком відвідування занять спортом, визначено через тип `PositiveSmallIntegerField`. Розглядаються групи:

- за нульової відвідуваності;
- за відвідуваності від 0 до 25 % включно;
- за відвідуваності від 25 до 50 % включно;
- за відвідуваності від 50 до 75 % включно;
- за відвідуваності від 75 до 100 % включно.

Тобто поле `enrollment_group` зберігає номер такої групи відвідуваності.

Поле моделі `activity` формує дані спортивної активності, за якою визначається кількість активних на задану дату абонементів. Визначено через тип `ForeignKey`.

Поле моделі `active_number` формує кількість активних абонементів, тобто абонементів, які можуть бути використані відвідувачами для відвідування заняття в клубі для занять спортом у задану дату. Визначено через тип `BigIntegerField`.

Поле моделі `source_type` формує джерело даних, тобто визначення кількості активних абонементів. Можливі два варіанти: обчислення з БД для дат, які вже пройшли, та прогнозування за допомогою навченої моделі для дат, які є поточними або ще не настали. Визначено через тип `BooleanField`.

Клас `Trainereview` визначає реалізацію моделі відгуку про тренера, яка включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором відгуку про тренера, який визначено через тип `BigAutoField`.

Поле моделі `visitor` формує дані про відвідувача, який написав відгук про тренера, визначено через тип `ForeignKey`.

Поле моделі `trainer` формує дані про тренера, якого стосується відгук, визначено через тип `ForeignKey`.

Поле моделі `published_time` формує дату і час публікації відгуку про тренера, визначено через тип `DateTimeField`.

Поле моделі `score` формує бали, які визначені відгуком, визначено через тип `PositiveSmallIntegerField`.

Поле моделі `published_text` формує текстове представлення думки з відгуку тренера, визначено через тип `CharField`.

Клас `Traininggroup` визначає реалізацію моделі групи для спільного тренування, яка включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором групи для спільного тренування, який визначено через тип `BigAutoField`.

Поле моделі `caption` формує назву групи для спільного тренування, визначено через тип `CharField`.

Поле моделі `active` формує стан групи для спільного тренування (активна чи ні), визначено через тип `BooleanField`.

Поле моделі `created_time` формує дату і час, коли ця група тренування була створена, визначено через тип `DateTimeField`.

Поле `grouppersons` формує перелік осіб, які є відвідувачами, що долучені до цієї групи тренування, визначено через тип `ManyToManyField`.

Клас `Personalapplication` визначає реалізацію моделі заявки на персональне тренування від відвідувача до тренера, яка включає наступні поля. Це саме не бронювання слоту тренування, а загальна заявка на те, щоб тренуватися з певним тренером від відвідувача.

Поле моделі `id` є унікальним ідентифікатором заявки на персональне тренування від відвідувача до тренера, який визначено через тип `BigAutoField`.

Поле моделі `visitor` формує дані про відвідувача, який подав заявку на тренування, визначено через тип `ForeignKey`.

Поле моделі `trainer` формує дані про тренера, до якого подано заявку, визначено через тип `ForeignKey`.

Поле моделі `active` формує стан заявки, що може бути поданим, підтвердженим, скасованим, визначено через тип `PositiveSmallIntegerField`.

Поле моделі `created_time` формує дату і час, коли ця заявка була подана, визначено через тип `DateTimeField`.

Поле моделі `reaction_time` формує дату і час, коли ця заявка була оброблена тренером, визначено через тип `DateTimeField`.

Клас `Subscriptionpurchase` визначає реалізацію моделі придбання абонементу на відвідування клубу для занять спортом, яка включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором придбання абонементу на відвідування клубу для занять спортом, який визначено через тип `BigAutoField`.

Поле моделі `visitor` формує дані про відвідувача, який придбав абонемент, визначено через тип `ForeignKey`.

Поле моделі `price` формує вартість придбаного абонементу, визначено через тип `IntegerField`.

Поле моделі `payment_number` формує номер платежу, за яким було виконано сплату, визначено через тип `CharField`.

Поле моделі `active` формує стан абонементу, тобто чи він ще діє, чи вже є неактивним, визначено через тип `BooleanField`.

Поле моделі `purchase_time` формує дату і час придбання абонементу, визначено через тип `DateTimeField`.

Поле моделі `ratio` формує відсоток відвідування занять спортом за цим абонементом, визначено через тип `FloatField`. Це відсоток, який відповідає співвідношенню кількості відвіданих тренувань до доступної кількості тренувань за цим абонементом за час, який вже минув. Тобто якщо абонемент на 2 місяці, минув місяць, то відсоток буде відповідати кількості відвіданих занять спортом до максимальної доступної кількості занять за цим абонементом.

Клас `Planenrollment` визначає реалізацію моделі запланованого індивідуального заняття спортом, яка включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором запланованого індивідуального заняття спортом, який визначено через тип `BigAutoField`.

Поле моделі `visitor` формує дані про відвідувача, який запланував це тренування для себе, визначено через тип `ForeignKey`.

Поле моделі `planned_start` формує дату і час, коли заняття має розпочатися, визначено через тип `DateTimeField`.

Поле моделі `planned_end` формує дату і час, коли заняття має закінчитися, визначено через тип `DateTimeField`.

Поле моделі `enrollment` формує дані про реєстрацію відвідування спортивного клубу, яку було здійснено за цим запланованим заняттям, визначено через тип `ForeignKey`. Якщо реєстрації не було, то значить в підсумку відвідувач не здійснив це заняття.

Поле моделі `activity` формує дані про спортивну активність, за якою заплановано заняття, визначено через тип `ForeignKey`.

Клас `Plangroupenrollment` визначає реалізацію моделі запланованого заняття спортом у групі, яка включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором запланованого заняття спортом у групі, який визначено через тип `BigAutoField`.

Поле моделі `trainingoupperson` формує дані про відвідувача з певної групи тренування, який запланував це заняття в групі, визначено через тип `ForeignKey`.

Поле моделі `planned_start` формує дату і час, коли заняття має розпочатися, визначено через тип `DateTimeField`.

Поле моделі `planned_end` формує дату і час, коли заняття має закінчитися, визначено через тип `DateTimeField`.

Поле моделі `traininggroup` формує дані про групу для спільного заняття спортом, визначено через тип `ForeignKey`.

Поле моделі `activity` формує дані про спортивну активність, за якою заплановано заняття, визначено через тип `ForeignKey`.

Клас `Plangroupenrollmentdetail` визначає реалізацію моделі деталізації запланованого заняття спортом у групі, яка включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором деталізації запланованого заняття спортом у групі, який визначено через тип `BigAutoField`.

Поле моделі `plangroupenrollment` формує дані запланованого заняття спортом у групі, визначено через тип `ForeignKey`.

Поле моделі `visitor` формує дані про відвідувача з групи тренування, якого стосується ця деталізація, визначено через тип `ForeignKey`.

Поле моделі `approved` формує стан підтвердження цієї деталізації. Це поле необхідно тому, що деталізація створюється, як тільки один з відвідувачів, який є учасником групи тренування, запланував проведення спільного тренування. Далі кожен має підтвердити або скасувати свою участь. Інакше деталізація знаходиться в стані невизначеності. Поле визначено через тип `BooleanField`.

Поле моделі `speciality` формує позначку або повідомлення стосовно участі цього відвідувача в запланованому занятті спортом у групі, визначено через тип `CharField`.

Клас `Muscle` визначає реалізацію моделі м'яза тіла людини, яка включає наступні поля. Поле моделі `id` є унікальним ідентифікатором м'яза тіла людини, який визначено через тип `BigAutoField`. Поле моделі `caption` формує саму назву м'яза, визначено через тип `CharField`.

Клас `Workout` визначає реалізацію моделі комплексу вправ для занять спортом, яка включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором комплексу вправ для занять спортом, який визначено через тип `BigAutoField`.

Поле моделі `trainer` формує дані про тренера, який створив цей комплекс вправ, визначено через тип `ForeignKey`.

Поле моделі `caption` формує назву комплексу вправ, визначено через тип `CharField`.

Поле моделі `photo` формує фотографію, якою ілюструється комплекс вправ, визначено через тип `ImageField`. Для комплексу може додаватися ще фотографія, відео, але вони вже подаються як відповідне посилання в тексті комплексу вправ, який визначається в наступному полі.

Поле моделі `article` виконує формулювання комплексу вправ, визначено через тип `TextField`.

Поле `recommendations` формує перелік всіх відвідувачів, які рекомендували цей комплекс вправ, визначено через тип `ManyToManyField`.

Поле моделі `activity` формує дані про спортивну активність, якої стосується цей комплекс вправ, визначено через тип `ForeignKey`.

Клас `Workoutmuscle` визначає реалізацію моделі, яка конкретизує м'язи, задіяні в комплексі вправ, включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором конкретизації м'язу в комплексі вправ для занять спортом, який визначено через тип `BigAutoField`.

Поле моделі `workout` формує дані про комплекс вправ, у якому цей м'яз задіяно, визначено через тип `ForeignKey`.

Поле моделі `muscle` формує дані про м'яз, який задіяно у комплексі вправ, визначено через тип `ForeignKey`.

Поле моделі `speciality` формує опис того, як задіяно цей м'яз у цій вправі, визначено через тип `CharField`.

Клас `Workoutplan` визначає реалізацію моделі, яка конкретизує вправи з комплексу і включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором конкретизації м'язу в комплексі вправ для занять спортом, який визначено через тип `BigAutoField`.

Поле моделі `workout` формує дані про комплекс вправ, до якого відноситься ця вправа, визначено через тип `ForeignKey`.

Поле моделі `exercise` формує дані про вправу, яка належить до цього комплексу, визначено через тип `ForeignKey`.

Поле моделі `times` формує кількість разів, яку потрібно повторити дану вправу в цьому комплексі, визначено через тип `PositiveSmallIntegerField`.

Поле моделі `speciality` формує деталі щодо виконання вправи у складі комплексу вправ з занять спорту, визначено через тип `CharField`.

Клас `Exercise` визначає реалізацію моделі вправи, яка включає наступні поля. Поле моделі `id` є унікальним ідентифікатором вправи, який визначено через тип `BigAutoField`. Поле моделі `caption` формує саму назву вправи, визначено через тип `CharField`.

Клас `Advice` визначає реалізацію моделі поради щодо занять спортом від тренера, яка включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором поради щодо заняття спортом від тренера, який визначено через тип `BigAutoField`.

Поле моделі `trainer` формує дані про тренера, який надав цю пораду, визначено через тип `ForeignKey`.

Поле моделі `caption` формує назву поради або фактично назву статті про цю пораду, визначено через тип `CharField`.

Поле моделі `photo` формує фотографію, якою ілюструється порада, визначено через тип `ImageField`. Для поради може додаватися ще фотографія, відео, але вони вже подаються як відповідне посилання в тексті статті з описом поради, який визначається в наступному полі.

Поле моделі `active` формує дані активності поради. Якщо порада неактивна, то вона не відображається відвідувачам. Поле визначено через тип `BooleanField`.

Поле моделі `article` виконує формулювання поради, визначено через тип `TextField`.

Поле `recommendations` формує перелік всіх відвідувачів, які рекомендували цю пораду, визначено через тип `ManyToManyField`.

Поле моделі `activity` формує дані про спортивну активність, за якою описано цю пораду, визначено через тип `ForeignKey`.

Клас `Advicemuscle` визначає реалізацію моделі, яка конкретизує м'язи, яких стосується порада від тренера, включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором конкретизації м'язу в пораді щодо занять спортом від тренера, який визначено через тип `BigAutoField`.

Поле моделі `advice` формує дані про пораду, в якій цей м'яз задіяно, визначено через тип `ForeignKey`.

Поле моделі `muscle` формує дані про м'яз, який зачеплено в цій пораді, визначено через тип `ForeignKey`.

Поле моделі `speciality` формує опис того, як задіяно цей м'яз у цій пораді, визначено через тип `CharField`.

Клас `Advicemuscle` визначає реалізацію моделі, яка додає пораду від тренера до скарбнички, включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором внесення поради від тренера до скарбнички, визначено через тип `BigAutoField`.

Поле моделі `visitor` формує дані про відвідувача, який зберіг собі в скарбничку цю пораду, визначено через тип `ForeignKey`.

Поле моделі `advice` формує дані про пораду, яку саме зберіг до скарбнички цей відвідувач, визначено через тип `ForeignKey`.

Клас `Trainingslot` визначає реалізацію моделі, яка визначає часовий слот для проведення персонального тренування у тренера, включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором часового слоту для проведення персонального тренування у тренера, визначено через тип `BigAutoField`.

Поле моделі `trainer` формує дані про тренера, який визначив цей слот для проведення тренування, визначено через тип `ForeignKey`.

Поле моделі `activity` формує дані про активність, за якою це тренування буде проводитись тренером, визначено через тип `ForeignKey`.

Поле моделі `start_time` формує дату і час, коли цей часовий слот має розпочатися, визначено через тип `DateTimeField`.

Поле моделі `end_time` формує дату і час, коли цей часовий слот має закінчитися, визначено через тип `DateTimeField`.

Поле моделі `visitor_limit` формує максимальну кількість відвідувачів, які можуть одночасно займатися з цим тренером під час даного часового слоту, визначено через тип `PositiveSmallIntegerField`.

Поле моделі `state` формує стан слоту, визначено через тип `PositiveSmallIntegerField`. Слот може бути відкритим, заповненим або скасованим. У випадку скасування слоту, коли вже додалися інші відвідувачі, вони мають бути про це повідомленні у власному плані, а їхнє бронювання тренування відповідно скасовано.

Клас `Slotbooking` визначає реалізацію моделі, яка визначає бронювання заняття в межах персонального тренування за доступним у тренера часовим слотом, включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором бронювання заняття в межах персонального тренування за доступним у тренера часовим слотом, визначено через тип `BigAutoField`.

Поле моделі `trainingslot` формує дані про часовий слот, за яким має це заброньоване тренування проводитись, визначено через тип `ForeignKey`.

Поле моделі `visitor` формує дані про відвідувача, який виконав це бронювання, визначено через тип `ForeignKey`.

Поле моделі `start_time` формує дату і час, коли заняття за бронюванням має розпочатися, визначено через тип `DateTimeField`.

Поле моделі `end_time` формує дату і час, коли заняття за бронюванням має закінчитися, визначено через тип `DateTimeField`.

Поле моделі `speciality` формує особливості проведення тренування або участі в тренуванні відвідувача, визначено через тип `CharField`.

Поле моделі `state` формує стан бронювання, визначено через тип `PositiveSmallIntegerField`. Бронювання може бути або запланованим, або виконаним, або скасованим. Скасованим воно стає у випадку скасування самим відвідувачем або у випадку, якщо тренер скасував цей часовий слот.

Клас `Slotbookingdetail` визначає реалізацію моделі, яка визначає деталізацію тренування за заброньованим заняттям, включає наступні поля. Деталізація полягає у визначенні того комплексу вправ, які було виконано під час тренування.

Поле моделі `id` є унікальним ідентифікатором бронювання заняття в межах персонального тренування за доступним у тренера часовим слотом, визначено через тип `BigAutoField`.

Поле моделі `slotbooking` формує дані про бронювання заняття в межах персонального тренування за доступним у тренера часовим слотом, за яким було проведено заняття та яке деталізовано, визначено через тип `ForeignKey`.

Поле моделі `exercise` формує дані про вправу, яка була виконана під час тренування, визначено через тип `ForeignKey`.

Поле моделі `times` формує кількість повторень вправи, визначено через тип `PositiveSmallIntegerField`.

Поле моделі `speciality` формує, визначено через тип

Клас `Trainingtariff` визначає реалізацію моделі, яка визначає тариф на персональне тренування у тренера, включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором тарифу на персональне тренування у тренера, визначено через тип `BigAutoField`.

Поле моделі `trainer` формує дані про тренера, який визначив тариф на свої тренування, визначено через тип `ForeignKey`.

Поле моделі `enrollment_limit` формує максимальну кількість окремих тренувань, протягом яких може взяти участь відвідувач, якщо сплатить за цей тариф, визначено через тип

Поле моделі `price` формує вартість тарифу, визначено через тип `IntegerField`.

Поле моделі `period` формує кількість місяців, протягом якої діє цей тариф, визначено через тип `PositiveSmallIntegerField`.

Клас `Trainingpurchase` визначає реалізацію моделі, яка визначає придбання ряду персональних тренувань у тренера відвідувачем, включає наступні поля.

Поле моделі `id` є унікальним ідентифікатором придбання тарифу на персональні тренування у тренера, визначено через тип `BigAutoField`.

Поле моделі `visitor` формує дані про відвідувача, який придбав цю серію тренувань, визначено через тип `ForeignKey`.

Поле моделі `trainingtariff` формує дані про тариф, який було сплачено, визначено через тип `ForeignKey`.

Поле моделі `price` формує суму, яку було сплачено за цей тариф, визначено через тип `IntegerField`.

Поле моделі `payment_number` формує номер платежу, за яким було виконано сплату, визначено через тип `CharField`.

Поле моделі `purchased_time` формує дату і час, коли було виконано сплату, визначено через тип `DateTimeField`.

Реалізація сценаріїв за необхідними функціональними вимогами забезпечена представленнями за MVT, які розділені на такі файли:

- `authviews`, де внесені представлення, які стосуються авторизації у програмі;
- `unvisitorviews`, де внесені представлення, які стосуються неавторизованих відвідувачів;

- `visitorviews`, де внесені представлення, які стосуються відвідувачів;
- `trainerviews`, де внесені представлення, які стосуються тренерів;
- `managerviews`, де внесені представлення, які стосуються менеджерів.

Реалізація всіх представлень виконана функціями. Частина з цих функцій описана нижче.

Відображення власних порад щодо занять спортом тренера реалізовано функцією представлення `trainer_advices_view`. Ця функція не має додаткових параметрів, тому дані про тренера беруться за авторизованим обліковим записом, при цьому нові поради розташовуються раніше, а перелік поради містить всі поради, в тому числі неактивні. А функція представлення `std_trainer_advices_view` має параметр `trainer_id`, який визначає ідентифікатор тренера, поради якого необхідно відобразити, але результати містять тільки активні поради.

Відображення заявок на тренування тренеру реалізовано функцією представлення `trainer_applications_view`, яка має параметр `application_state`, що визначає стан заявок, які мають бути відображені авторизованому тренеру. Серед результатів, які повертаються у переважній більшості представлень, є повідомлення за ключем `message`. У випадку помилок це повідомлення містить проблему. Якщо помилок немає, то повідомлення порожнє.

Функція `application_approve_view` реалізує представлення підтвердження заявки з номером `application_id`. Підтвердження реалізується тільки тоді, коли авторизований тренер є адресатом заявки з цим номером.

Функція `application_cancel_view` реалізує представлення скасування заявки з номером `application_id`. Якщо заявка знаходиться в стані підтвердженої, то скасування дозволяється тільки тоді, коли відвідувач не сплатив раніше за персональне тренування за одним з тарифів. Якщо він це зробив, то скасування вже не допускається. Тоді це питання має розв'язуватися за участю менеджера або адміністратора, який зокрема через адміністративну панель може виконувати будь-якій дії зі зміни даних у БД.

Відображення відвідувачів у тренера реалізовано функцією представлення `trainer_visitors_view`. За кожним відвідувачем, який придбав один з тарифів на персональні заняття з тренером, який авторизувався, повертається також перелік його виконаних бронювань, що дозволяє на тій же вебсторінці відобразити відповідні дані за конкретним відвідувачем, якщо це необхідно.

Відкриття окремої сторінки з відображенням історії тренувань відвідувача у тренера за логіном відвідувача `visitor_username` реалізовано через функцію представлення `trainer_visitor_history_view`.

Відображення зареєстрованих на тренуванні на поточний момент за тренером, який авторизувався, реалізовано функцією представлення `trainer_enrolled_view`. При цьому відображаються всі відвідувачі, які мають бронювання на тренування з цим тренером на поточну дату і чисе відвідування клубу для занять спортом на поточний момент зареєстровано, але все ще не завершено.

Відображення задіяності тренера виконано функцією представлення `trainer_engagement_view`, яка має параметри `period`, що визначає мінімальний період визначення задіяності (0 відповідає року, 1 – місяцю, 2 – тижню, 3 – дню), `day_start`, що визначає день початку періоду розрахунків, `month_start`, що визначає місяць початку періоду розрахунків, `year_start`, що визначає рік початку періоду розрахунків, `day_end`, що визначає день завершення періоду розрахунків, `month_end`, що визначає місяць завершення періоду розрахунків, `year_end`, що визначає рік завершення періоду розрахунків.

Публікація комплексу вправ реалізована через функцію представлення `workout_publish_view` на основі взаємодії з формою, що відповідає моделі комплексу вправ. Авторство публікації встановлюється автоматично за даними авторизації тренера. Якщо це не тренер, то публікація неможлива. Публікація поради щодо занять спортом виконується за допомогою функції представлення `advice_publish_view`.

Подібно влаштоване, наприклад, і додавання слоту для тренування, а ось редагування слоту має свої особливості. Воно реалізовано функцією представлення `slot_update_view`. Параметр `trainingslot_id` визначає ідентифікатор слоту для тренування. Оскільки в результаті збереження нових даних слоту може виявитися, що слот, який був збережений раніше, з ним перекривається, то реалізовано перевірку всіх слотів, які перекриваються з новим періодом слоту, який редагується. Якщо час одного зі слотів міститься в іншому, то менший слот вилючається. Якщо час слотів перекривається, то один зі слотів розширяється за рахунок іншого.

Пошук комплексів вправ реалізовано функцією представлення `workouts_search_view`. Усі параметри виділяються з запиту (адреси звернення), включаючи максимально:

- перелік назв м'язів, які задіяні у цьому комплексі вправ, у параметрі `muscles`, де ці назви розділяються дефісами з окличним знаком;
- назву комплексу вправ у параметрі `name`;
- перелік назв окремих вправ, які задіяні у цьому комплексі вправ, у параметрі `exercises`, де ці назви розділяються дефісами з окличним знаком;
- прізвище тренера у параметрі `trainer`;
- назва активності в параметрі `activity`;
- мінімальна оцінка тренера в параметрі `score`;
- максимальна оцінка тренера в параметрі `score_max`.

Результати пошуку впорядковуються за кількістю рекомендацій у комплексу вправ. Подібним чином організовано представлення пошуку порад щодо занять спортом у функції `advice_search_view`.

Відображення зареєстрованих на тренуванні на поточний момент реалізовано функцією представлення `enrolled_view`. При цьому відображаються всі відвідувачі, які зареєстрували своє відвідування клубу для занять спортом на поточний момент, але не завершили його.

Відображення статистики використання абонементів виконано функцією представлення `subscription_usage_view`, яка має параметри

day_start, що визначає день початку періоду розрахунків, month_start, що визначає місяць початку періоду розрахунків, year_start, що визначає рік початку періоду розрахунків, day_end, що визначає день завершення періоду розрахунків, month_end, що визначає місяць завершення періоду розрахунків, year_end, що визначає рік завершення періоду розрахунків. Розрахунки виконуються за кожною пропозицією абонементу. За кожною пропозицією визначається, яка кількість занять була доступна відвідувачів, які купили абонемент такого типу, та кількість занять, яку вони фактично відвідали. Подібно влаштовано функцію представлення activity_usage_view, де відбуваються подібні дії з тими самими параметрами, але групування виконується не за пропозиціями абонементів, а за спортивними активностями.

Відображення статистики активних абонементів реалізовано разом з прогнозуванням через функцію представлення statistics_view, у якій задається окрім інтервалу дат ще спортивна активність. Дані беруться з БД. А у БД вносяться заново щоденно, додаючи день, що минув, та виконуючи прогноз на 7 наступних днів. Для реалізації моделей було використано пакет keras, включаючи класи з підпаketу keras.layers для визначення шарів, зокрема LSTM (такий фактично один шар) і Dense (підтримка повнозв'язного шару з кількістю нейронів, що дорівнює кількості днів, на які виконується прогнозування), а також клас базової моделі Sequential для визначення входів моделі. Під час експериментального дослідження Random Forest реалізовано через використання пакету sklearn, з якого з підпаketу ensemble застосовано клас RandomForestRegressor для побудови регресійної моделі. Прогнозування виконувалось на 7 днів.

4.3 Висновки за розділом 4

Результати виконання четвертого розділу стосуються реалізації програмного забезпечення, використовуючи для цього фреймворк Django,

визначаючи всі моделі у відповідності зі структурою БД, розмітку всіх шаблонів у відповідності зі сценаріями користувачів та визначаючи потрібні для цього представлення, формулюючи потік взаємодії між відвідувачем і тренером стосовно проведення індивідуального персонального тренування за допомогою діаграми послідовності.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Опис програми

Програма, яку розроблено, дозволяє популяризувати спортивні активності, тренерів у спортивному клубі, підтримувати бронювання тренувань з персональним тренером, планувати індивідуальні тренування та тренування в групах відвідувачів, визначати статистику відвідуваності та забезпечувати прогнозування кількості абонементів у широкому розрізі для подальшого визначення показників діяльності клубу для занять спортом.

5.2 Експлуатація програми

Коли користувач заходить на вебсайт, то він працює з мінімальним набором функцій, що розташовані в меню у розділі Клуб, що включає підрозділи Інформація з основною інформацією про клуб, Фото з фотографіями з тренувань, Активності з переліком спортивних активностей клубу для занять спортом (рис. 5.1), звідки можна перейти до сторінки відповідної зали, у розділі Тренери, де знаходиться перелік тренерів, які працюють у клубі, звідки доступний перелік відгуків про кожного тренера, перелік порад кожного тренера (рис. 5.2), де при наведенні відображаються також задіяні м'язи, активність і тренер, та перелік комплексів вправ, що відкриваються на окремій сторінці, розділі Вправи, де потрібно спочатку обрати м'яз, а за ним відображається перелік статей про комплекси вправ за цим м'язом, розділі Абонементи з вартістю доступних абонементів.

Кнопка з зображенням лупи призводить до того, що над сторінкою спливає форма для організації пошуку. Кнопка дозволяє перемкнутися між пошуком порад, який може полягати у визначенні назв м'язів, назви статті про пораду, імені тренера, конкретної активності, мінімальної і максимальної оцінок тренера, та пошуку комплексів вправ.

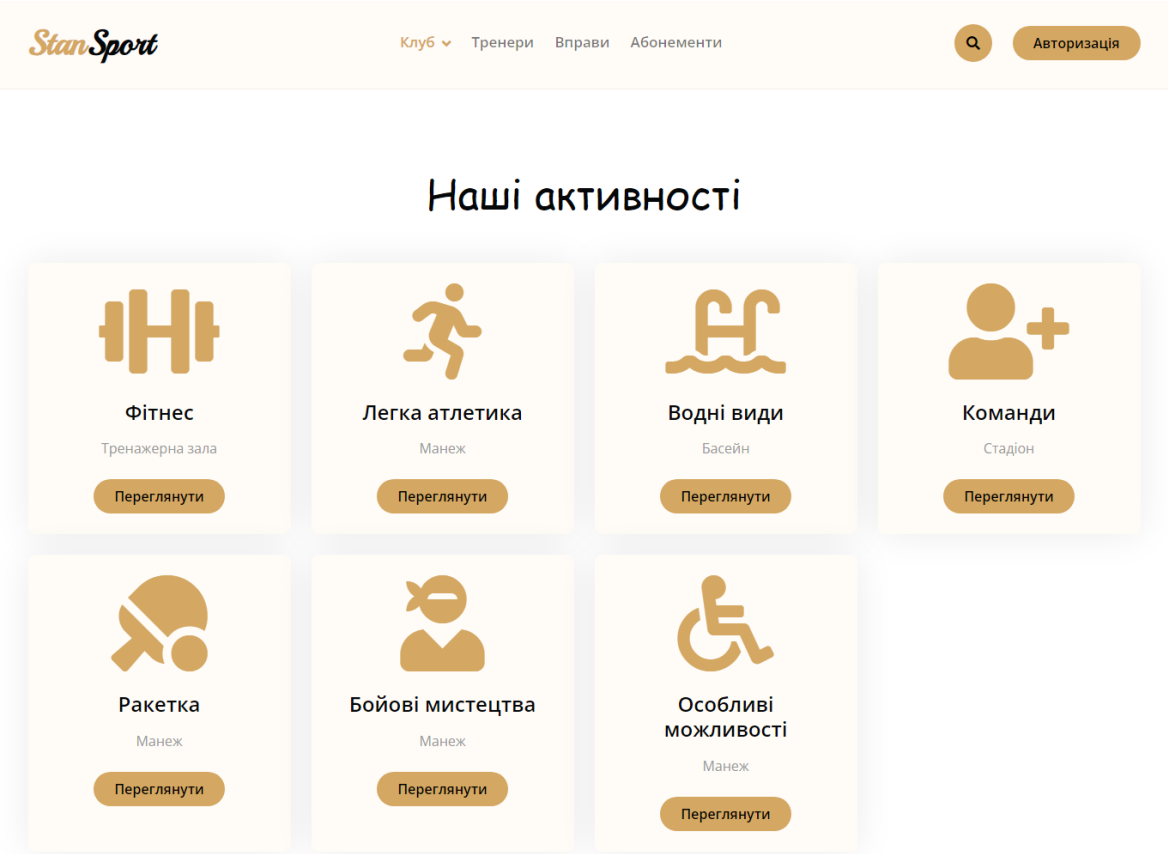


Рисунок 5.1 – Активності клубу для занять спортом

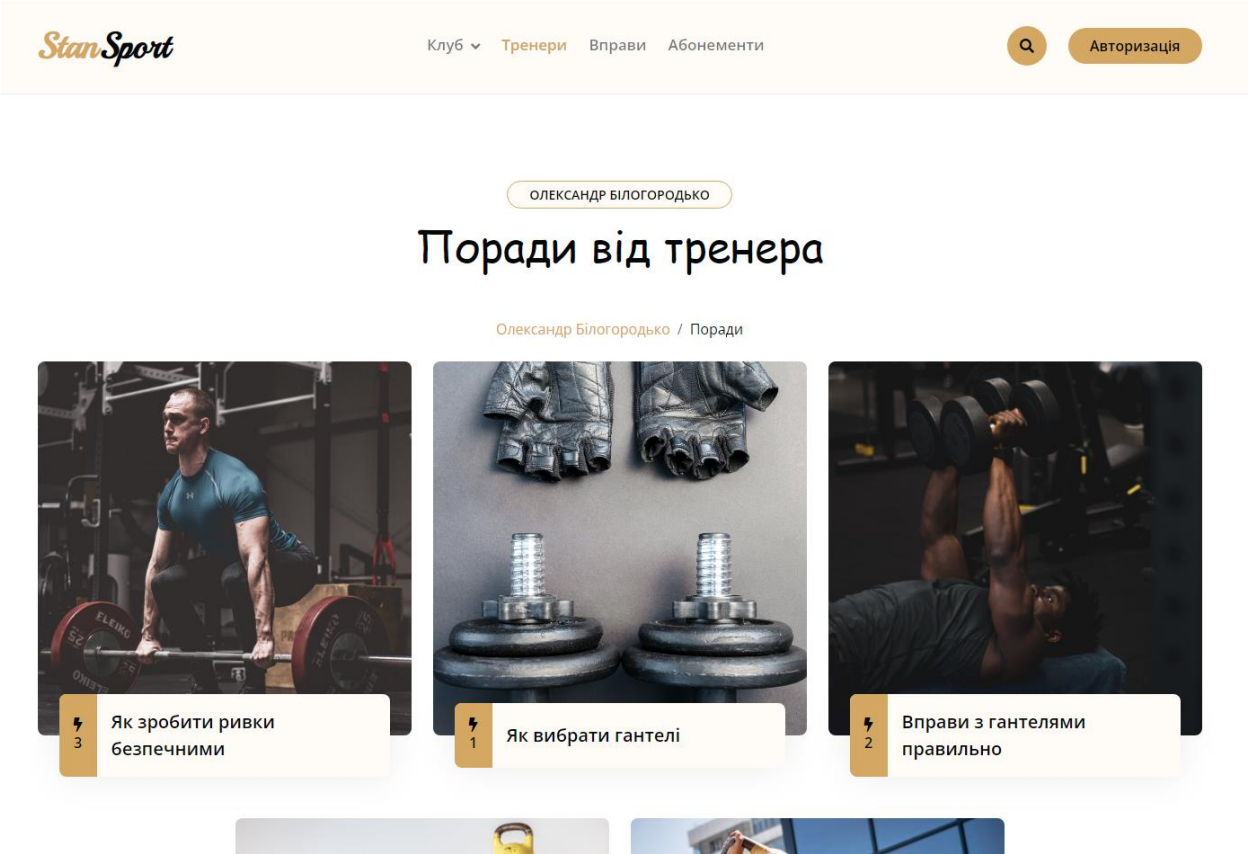


Рисунок 5.2 – Поради від тренера про заняття спортом

Пошук комплексів вправ може полягати у визначенні назв м'язів, перелічених через кому, які мають бути задіяні у комплексі вправ, назви статті про комплекс вправ, перелічених через кому назв вправ, імені тренера, конкретної активності, мінімальної та максимальної оцінок тренера.

Кнопка Авторизація дозволяє відкрити форму авторизації, звідки можна перейти до реєстрації відвідувачів. За авторизації відвідувача йому відкривається меню з додатковим розділами, зокрема Групи, що включає підрозділи перелік для відображення всіх груп, до яких було додано відвідувача, Планувати, що дозволяє перейти до планування спільного тренування. Для групи можна відобразити всі заплановані тренування, а за тренуванням продемонструвати конкретне з них (рис. 5.3). Якщо відвідувач не є автором планування, то йому доступна кнопка, яка виглядає як галочка (для підтвердження участі, яка після цього зникає, а за наведення на неї розширюється до відображення другої кнопки для скасування участі).

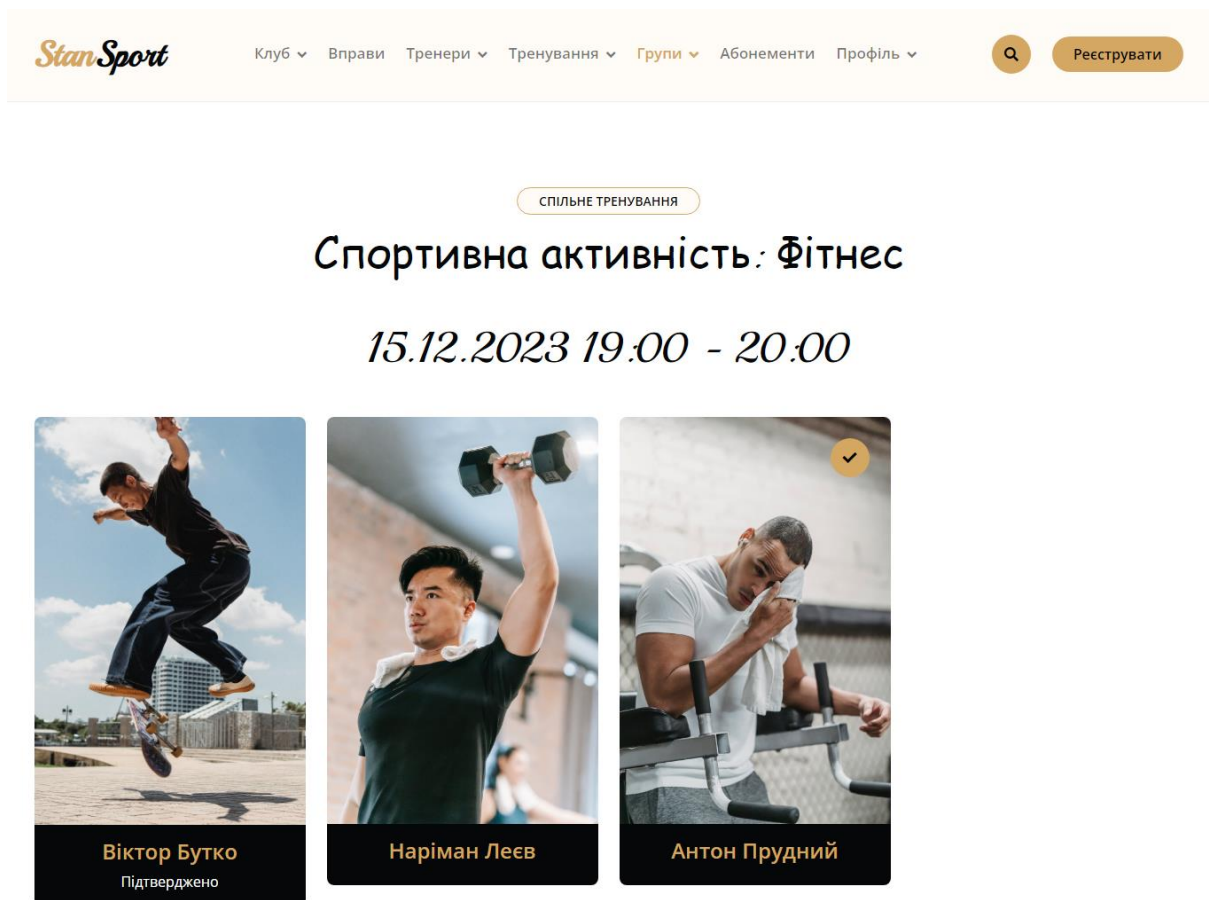
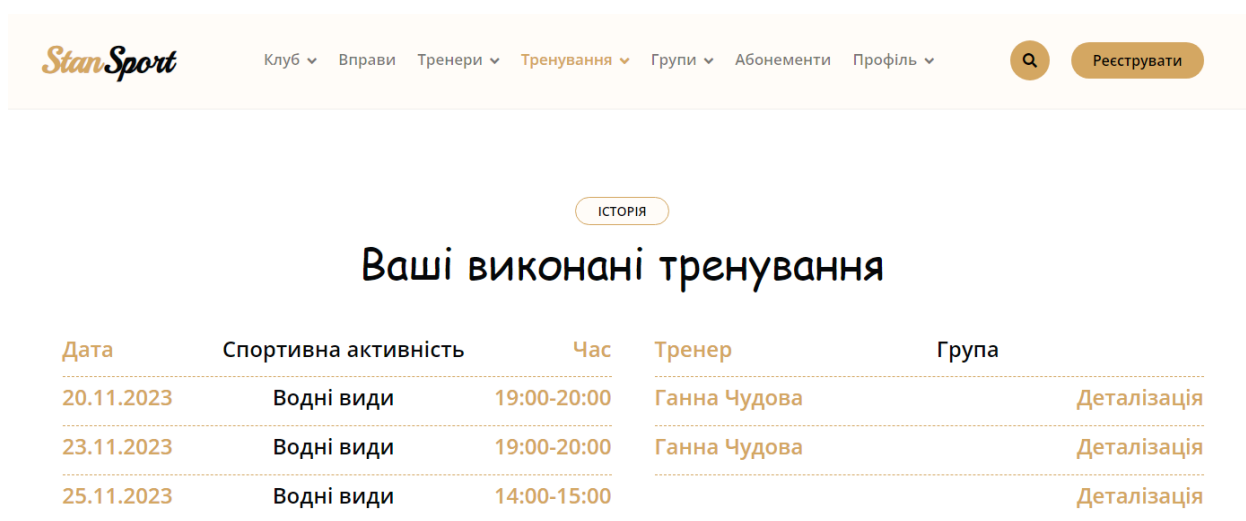


Рисунок 5.3 – Планування спільного тренування

Додатковий розділ Тренування містить підрозділи Історія, що дозволяє перейти до перегляду тих тренувань, які вже відбулись і на яких був зареєстрований у клубі відвідувач (рис. 5.4), План, що дозволяє переглянути всі заплановані індивідуальні та групові тренування, Бронювання, що дозволяє переглянути всі заброньовані персональні тренування, Індивідуальний план, що дозволяє створити план нового індивідуального тренування, Груповий план, що дозволяє створити план нового тренування у складі однієї з груп тренування, до якої внесено відвідувача.



Дата	Спортивна активність	Час	Тренер	Група
20.11.2023	Водні види	19:00-20:00	Ганна Чудова	Деталізація
23.11.2023	Водні види	19:00-20:00	Ганна Чудова	Деталізація
25.11.2023	Водні види	14:00-15:00		Деталізація

Рисунок 5.4 – Історія тренувань відвідувача

Зі сторінки реєстрації за посиланням Деталізація можна відкрити перелік вправ на тренуванні (рис. 5.5), звідки зокрема відвідувач може редагувати вправи, їхню кількість та додавати нові.

Кнопка Реєструвати дозволяє виконати реєстрацію відвідування в клубі. Відбувається перехід до сторінки, на якій формується QR-код. Зі своєї сторінки менеджер має перевірити цей QR-код, тоді реєструється відвідування клубу. Код формується з ідентифікатора відвідувача у програмі, спеціального унікального коду, який формується для користувача і який більше ніде не надається, та з ідентифікатора покупки абонементу, який діє

на даний момент. Якщо ці дані є коректними, тобто вони будуть розпізнані зі сторінки менеджера і виявляться правильними, то відбудеться реєстрація.

StanSport

Клуб ▾ Вправи Тренери ▾ Тренування ▾ Групи ▾ Абонементи Профіль ▾

Q

Реєструвати

ДЕТАЛІЗАЦІЯ

Перелік виконаних вправ

Фітнес 22.11.2023 20:00-21:00

Стартовий набір, повторювати

Додати вправу

Номер	Вправа	Кількість	
1	Підтягування	5	Змінити
2	Тяга верхнього блоку	10	Змінити
3	Тяга блоку до поясу	8	Змінити
4	Підйоми на біцепс	8	Змінити
5	Тяга верхнього блоку двома руками	10	Змінити
6	Біцепс на нижніх блоках	10	Змінити

Рисунок 5.5 – Деталізація переліку вправ за бронюванням

Тренер має меню, що включає розділи Відвідувачі з підрозділами Заявки для підтвердження або скасування поданих заявок, Активні для визначення тих відвідувачів, які мають підтверджені заявки та сплачений тариф на тренування у цього тренера, розділ Тренування з підрозділами Історія, що дозволяє задати логін відвідувача, а далі відобразити всі виконані бронювання (тренування) цього відвідувача у цього тренера, Слоти, що дозволяє управляти переліком доступних слотів за тренером на вказану дату, виконуючи зміну тривалості слоту (при зміні часу слоту виконується за можливості об’єднання слотів, якщо вони перетинаються), видаляти слот

(рис. 5.6), Зареєстровані, що дозволяє переглянути всіх відвідувачів, які мають на поточний момент заброньоване тренування та їхнє відвідування клубу зареєстроване, План, що дозволяє переглянути всі заброньовані тренування з цим тренером, розділ Публікації з підрозділами Вправи, що дозволяє відкрити перелік власних статей з комплексами вправ, створеними тренером, та Поради, що дозволяє відкрити перелік власних статей з порадами щодо занять спортом, створеними тренером, Тарифи з переліком існуючих та можливістю додавати, змінювати тарифи на персональні тренування з цим тренером, та Профіль з підрозділами Власний, що дозволяє переглянути і змінити дані профілю, Задіяність, що дозволяє переглянути дані за період про кількість успішних бронювань (фактично тренувань) у тренера за заданий період часу щоденно, щотижнево, щомісячно або щорічно, та Відгуки, що дозволяє переглянути всі відгуки про тренера. Кнопка Реєструвати тренера дозволяє внести за тренуванням вправи та кількість їх виконання до тієї деталізації, яку потім може відвідувач змінювати.

StanSport

Відвідувачі ▾ Тренування ▾ Публікації ▾ Тарифи Профіль ▾

🔍 Реєструвати

слоти

Наявні слоти на задану дату

05.12.2023 📅

Показати

Додати слот

Час	Спортивна активність	Стан	Місце		
10:00-12:00	Фітнес	Активний	1	Змінити	Видалити
14:00-15:00	Фітнес	Активний	1	Змінити	Видалити
18:00-22:00	Фітнес	Активний	2	Змінити	Видалити

Рисунок 5.6 – Управління слотами для занять у тренера

Створити відгук може сам відвідувач зі сторінки тренера, якщо у нього були в цього тренера виконані бронювання.

Менеджер за авторизації має розділи меню Клуб, що включає ті самі дані, що і для відвідувача, але менеджер може їх редагувати, Вправи, що дозволяє переглядати вправи за групою м'язів, Тренування, що включає підрозділи Історія, План, Зареєстровані (рис. 5.7), аналогічні описаним вище, але вже стосовно всього клубу, а не тільки одного тренера, Користувачі, що містить підрозділи Відвідувачі, Тренери та Менеджери, що дозволяє переглянути перелік всіх відповідних користувачів або виконати пошук відвідувачів, зареєструвати нових тренерів або менеджерів, що можуть зробити тільки менеджери, та зареєструвати відвідувачів за необхідності, Статистика, що включає підрозділи Активності, Абонементи та Активні абонементи за відповідними сценаріями роботи менеджера, Абонементи, Профіль. Зареєстровані відвідини містять перелік всіх відвідувачів, які зареєстрували свої відвідини у клубі, але не зареєстрували вихід (або дата не вийшла), розташовані від найновішої реєстрації.

 Клуб ▾ Вправи Тренування ▾ Користувачі ▾ Статистика ▾ Абонементи Профіль ▾  Регіструвати					
ТАБЛО РЕЕСТРАЦІЇ					
Зареєстровані відвідини на даний момент					
Реєстрація	Прізвище відвідувача	Логін	Тренер	Спортивна активність	
19.11.2023 17:31	Віктор Бутко	butko		Фітнес	
19.11.2023 17:28	Андрій Мочупан	rap		Водні види	
19.11.2023 17:26	Наріман Леев	nariman		Фітнес	
19.11.2023 17:22	Антон Прудний	prood		Фітнес	

Рисунок 5.7 – Зареєстровані відвідини клубу для занять спортом у відображення для менеджера

Відповідно підрозділ Активні абонементи з розділу Статистика за визначення відповідних дат містить кількість абонементів, отриману за результатами прогнозування (рис. 5.8). У даному випадку ситуацію було змодельовано для відображення наступних значень за значеннями, наявними у вибірці. Стандартно дані одразу вносяться на збереження кожен день, а наступні 7 днів стають результатами прогнозування. Під час відображення статистики за активними абонементами потрібно задати проміжок дат, обрати спортивну активність, після цього відображається для кожної дати кількість активних абонементів за кожною групою відвідування, які було описано в розділі 2.

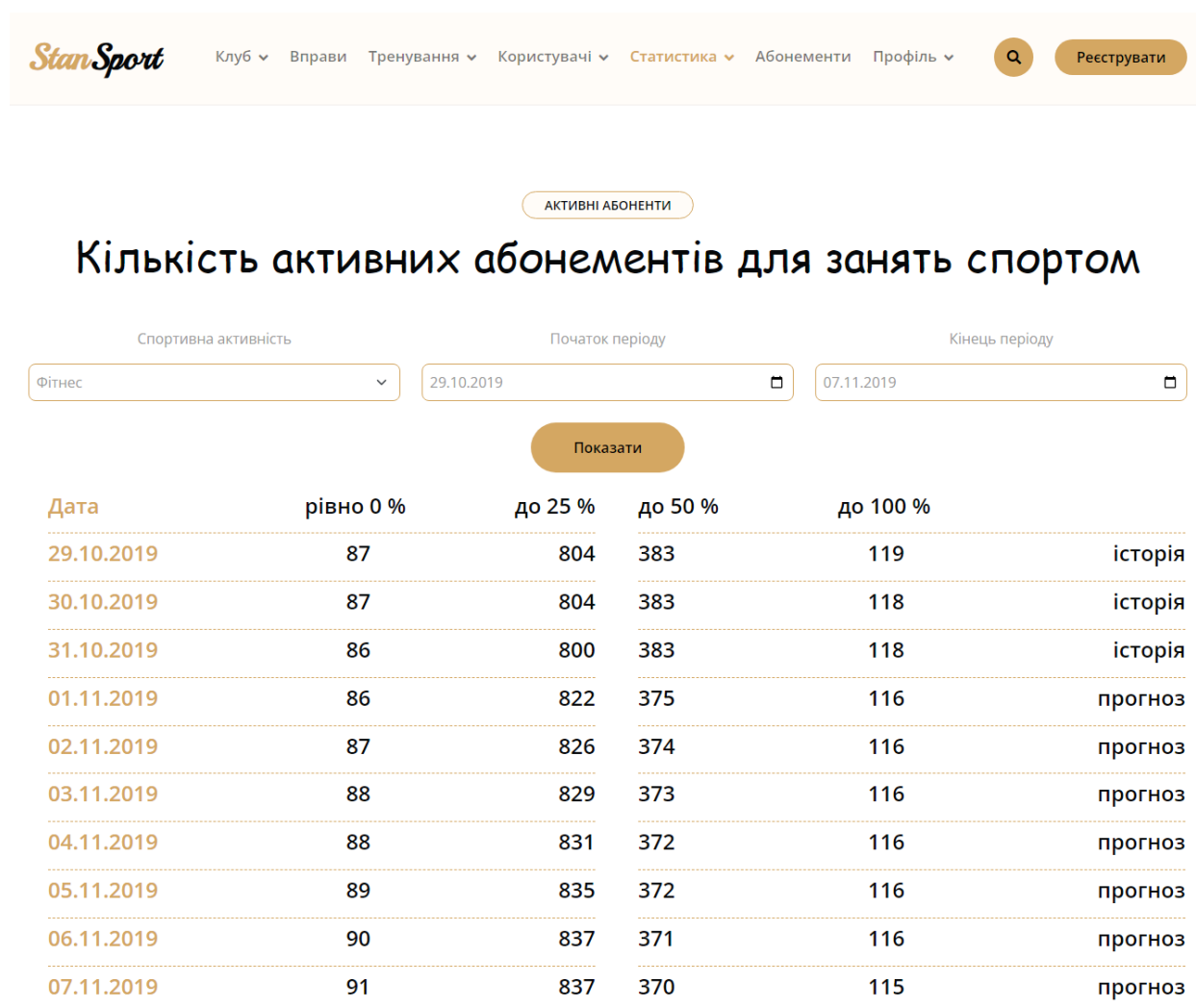


Рисунок 5.8 – Відображення статистики активних абонементів і прогнозування кількості активних абонементів

5.3 Експериментальне дослідження

Експериментальне дослідження виконувалось за допомогою використання вибірки даних [14], яка відкрито доступна для проведення досліджень і представляє дані, зібрані в спортивному центрі в місті Лісабон, що знаходиться в Португалії. Дані були зібрані з 1 червня 2014 року до 31 жовтня 2019 року і мають достатньо докладний опис, приведений в роботі [15]. Дані мають цілий ряд характеристик, серед яких були відібрані тільки ті, які відповідають принципам, закладеним у метод прогнозування показників діяльності клубу для занять спортом. Вибірка містить також характеристики стосовно відвідуваності видів спорту, пов'язаних з танцями та відвідуванням природи, але фактичних даних за цими характеристиками не представлено. Тому дані стосуються в підсумку легкої атлетики, водних видів спорту, фітнесу, командних, ракетних видів спорту, бойового мистецтва, занять для людей з обмеженими можливостями.

За кожним абонементом присутні дані про рівень відвідуваності занять як співвідношення відвіданих занять до максимальної можливої кількості. Відповідний відсоток розділено на групи:

- ті, хто взагалі не відвідував спортивний центр, маючи абонемент;
- ті, хто відвідував заняття, але відвідав не більше ніж 25 %;
- ті, хто відвідав більше 25 % занять, але не більше ніж 50 %;
- ті, хто відвідав більше ніж 50 %: дані розділені початково на групи від 50 до 75 % та від 75 до 100 %, але зважаючи на те, що кількість людей у цих групах зазвичай є малою, то їх було об'єднано в єдину групу.

Розрахувавши кількість абонементів, яка в середньому була активною на кожен рік дослідження, було встановлено, скільки цих абонементів використовувалось людьми, які не відвідували заняття, які відвідали до 25 % занять включно, більше 25 і до 50 % занять включно та більшу кількість. Далі для різних видів спорту було побудовано відповідні гістограми, зокрема для легкої атлетики (рис. 5.9) та відвідування басейну (рис. 5.10).

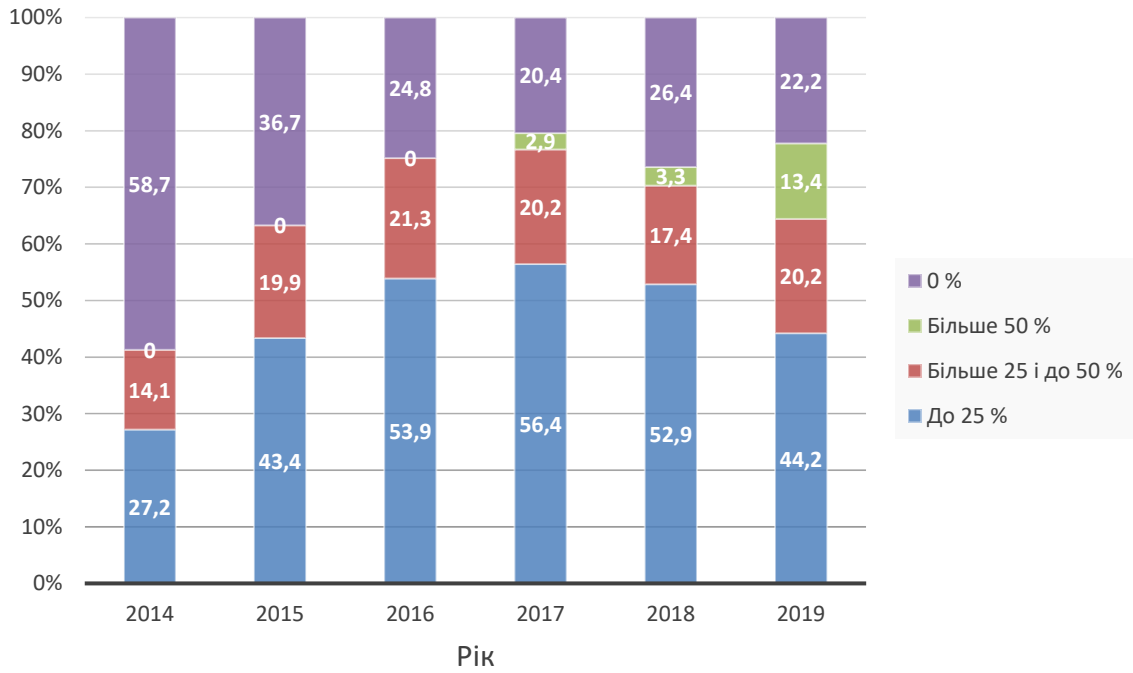


Рисунок 5.9 – Розподіл відвідуваності за роками для занять легкою атлетикою

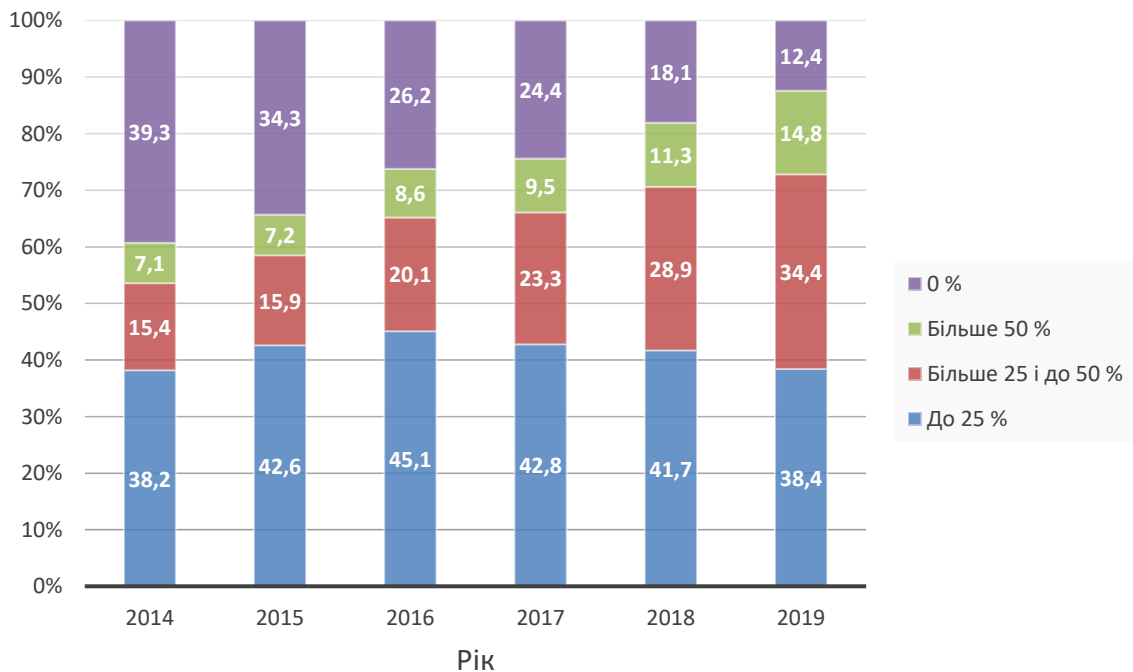


Рисунок 5.10 – Розподіл відвідуваності за роками для занять у басейні

Аналогічні гістограми побудовані для занять фітнесом (рис. 5.11) та командних видів спорту (рис. 5.12).

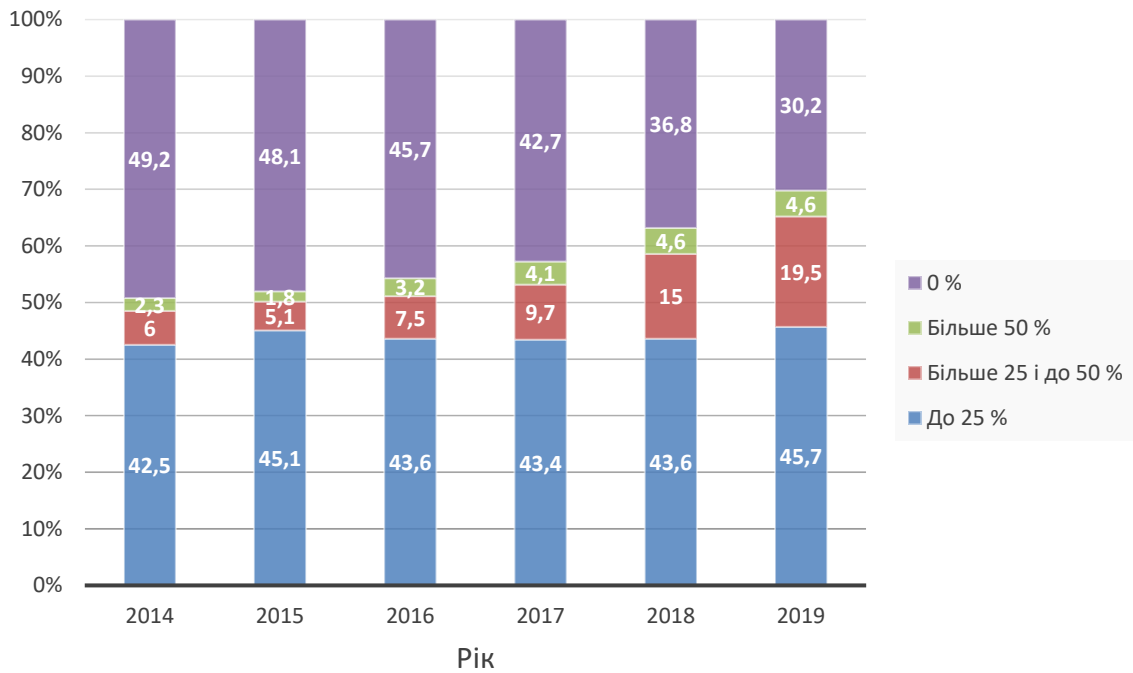


Рисунок 5.11 – Розподіл відвідуваності за роками для занять фітнесом

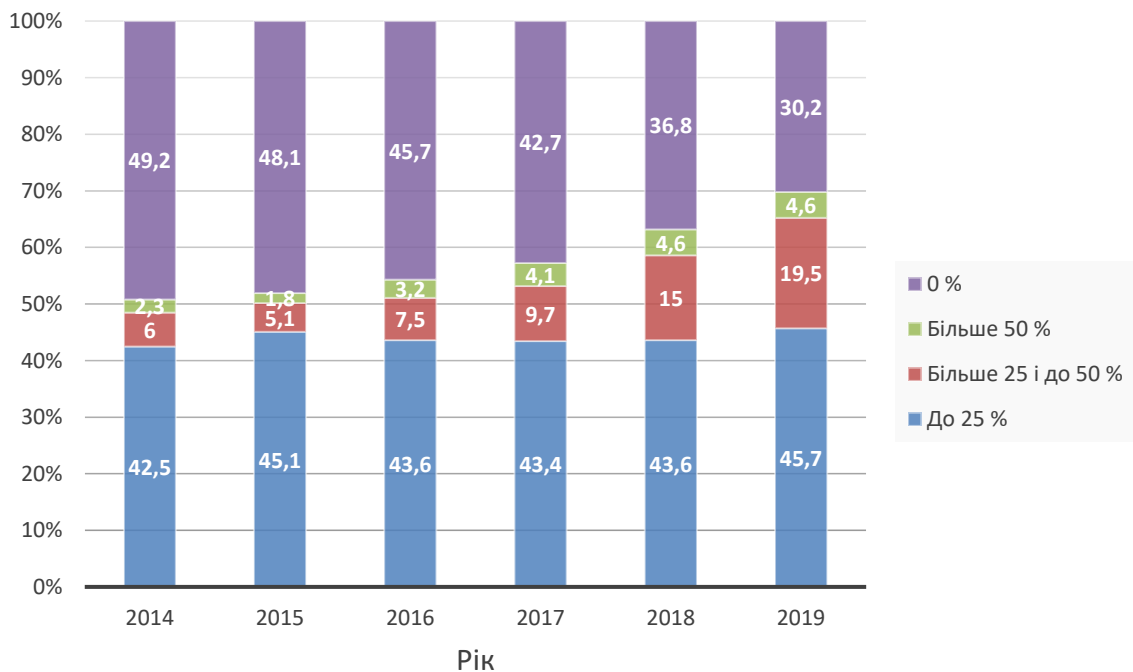


Рисунок 5.12 – Розподіл відвідуваності за роками для занять командними видами спорту

Аналогічні гістограми побудовані для занять командними видами спорту з ракетками (рис. 5.13) та занять бойовим мистецтвом (рис. 5.14).

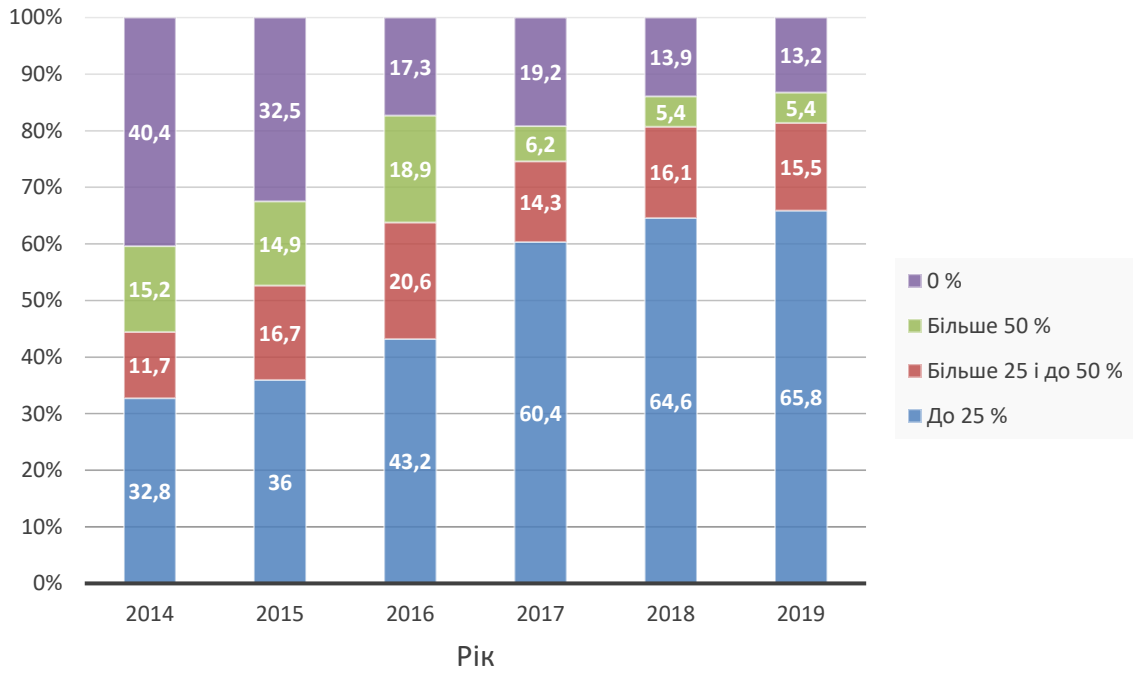


Рисунок 5.13 – Розподіл відвідуваності за роками для занять ракетними видами спорту

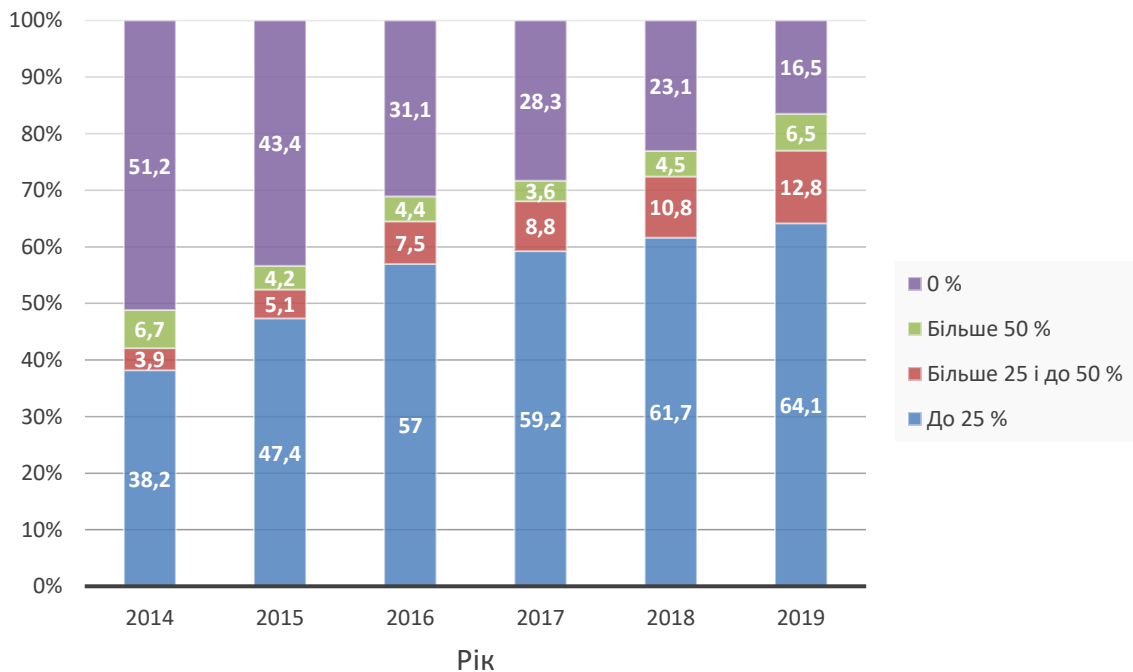


Рисунок 5.14 – Розподіл відвідуваності за роками для занять бойовим мистецтвом

Аналогічна гістограма побудована для занять спортом серед людей з

обмеженими можливостями (рис. 5.15).

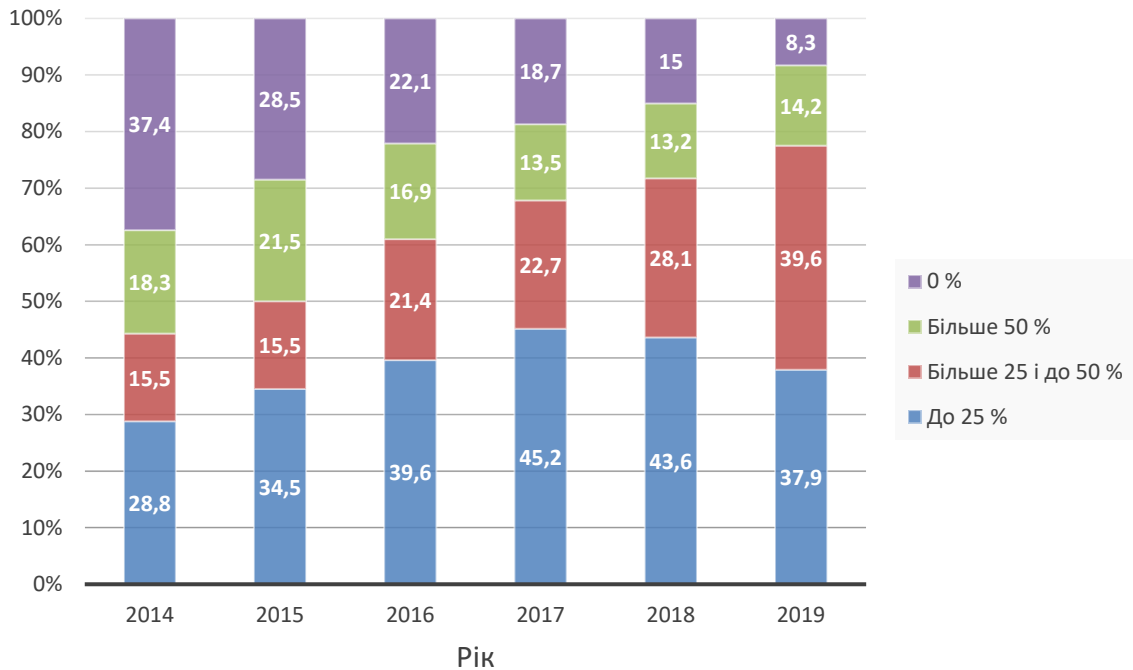


Рисунок 5.15 – Розподіл відвідуваності за роками для занять спортом серед людей з обмеженими можливостями

Представлені гістограми вказують на те, що з часом змінюється не тільки кількість учасників у кожній групі відвідувачів занять спортом, але і співвідношення між групами. Як помітно з гістограм, відсоток людей, які купляють абонемент, але не займаються спортом, значно знизився за фактично всіма видами спортивних активностей. При тому на перший рік відсоток таких людей складав від 37,4 % для людей з обмеженими можливостями до 58,7 % для людей, які займаються легкою атлетикою.

Далі було проаналізовано зміну кількості активних абонементів на день за місяцями. Ці дії було виконано тільки для водних видів спорту (басейн), фітнесу та бойових мистецтв, адже вони мають біль співмірну кількість абонементів, що дозволяє їх адекватно відображати разом. Зміну кількості абонементів було проаналізовано у групах абонементів, де відвідано більше 0 і не більше ніж 25 % занять (рис. 5.16) та до 50 % (рис. 5.17).

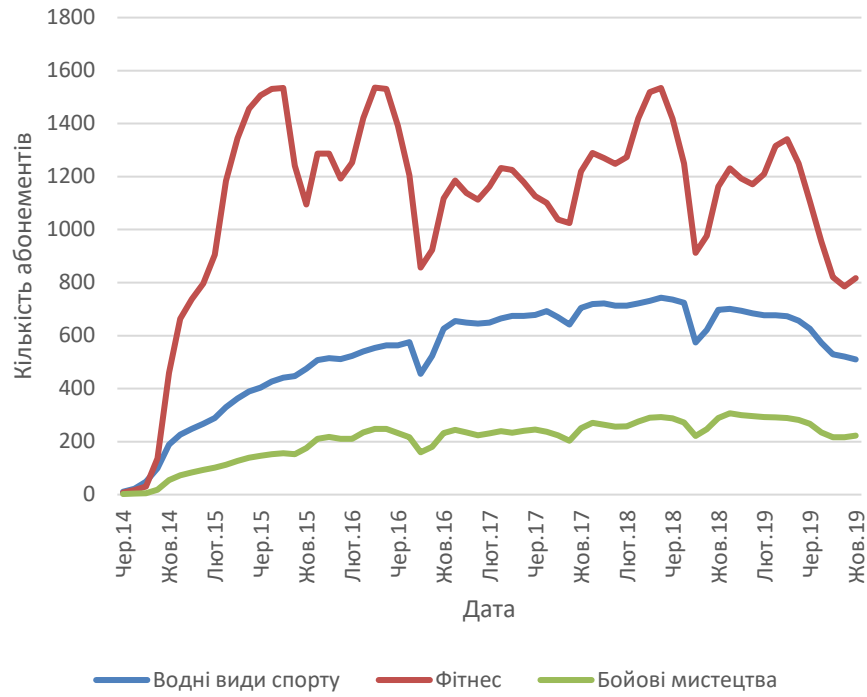


Рисунок 5.16 – Зміна кількості абонементів за місяцями за відвідуваності більше 0 і до 25 % включно

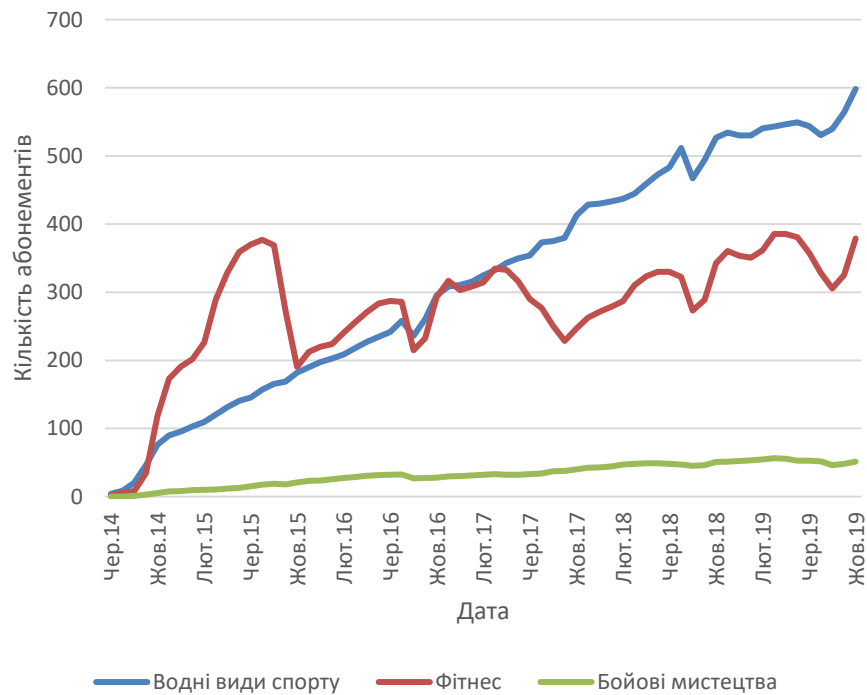


Рисунок 5.17 – Зміна кількості абонементів за місяцями за відвідуваності більше 25 і до 50 % включно

Аналогічні графіки побудовані для відвідуваності більше 50 %

(рис. 5.18) та для тих, хто взагалі не відвідував центр (рис. 5.19).

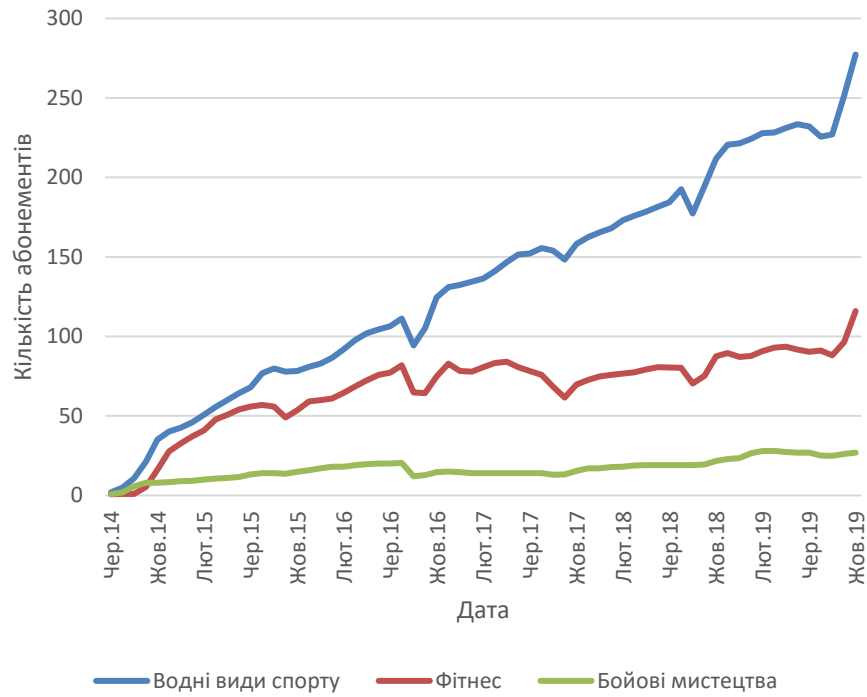


Рисунок 5.18 – Зміна кількості абонементів за місяцями за відвідуваності більше 50 %

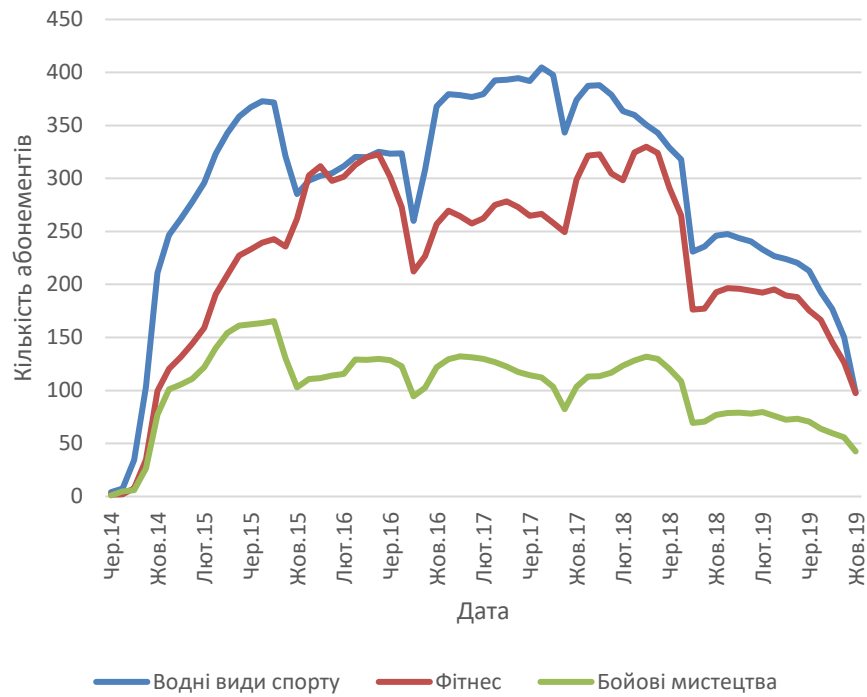


Рисунок 5.19 – Зміна кількості абонементів за місяцями за невідвідуваності

Побудовано аналогічний графік, але вже для всіх абонементів разом (рис. 5.20) без розбиття на групи за відвідуваністю.



Рисунок 5.20 – Загальна зміна кількості абонементів за місяцями

З графіків помітно, що кількість абонементів постійно змінюється як всередині груп, так і разом, при чому не тільки збільшується, але і зменшується часом. Це достатньо суттєві зміни, адже вони помітні і в місячному вимірі. Прослідкувати такі зміни можна і на графіках, де вже наведено кількість абонементів на кожен день, наприклад, для фітнесу (рис. 5.21). На цих графіках помітно різке зменшення кількості абонементів у певний момент (рис. 5.22), що може бути більш або менш вираженим для різних груп для різних спортивних активностей. Однак, це не завжди так. Іноді такі стрибки відсутні протягом місяця або є менш вираженими.

Тож загалом дані вказують на те, що кількість абонементів постійно змінюється. Для планування поточних робіт, обмежень важливим є розуміння очікуваної кількості абонементів на кожен момент, а до того ж того, на скільки активно ці люди користуються абонементом.

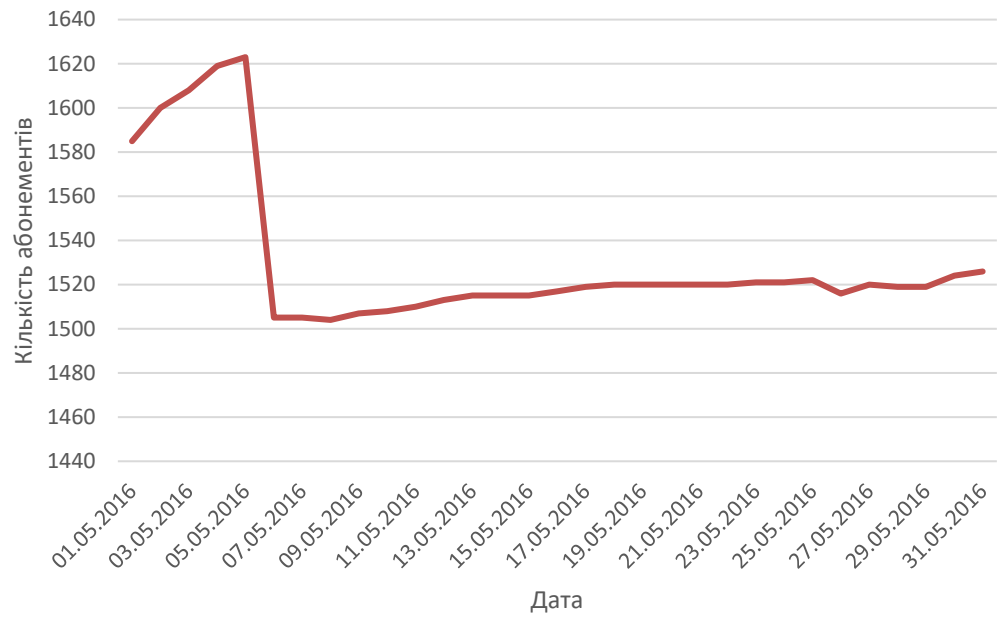


Рисунок 5.21 – Щоденна кількість активних абонементів для занять фітнесом за травень 2016 року

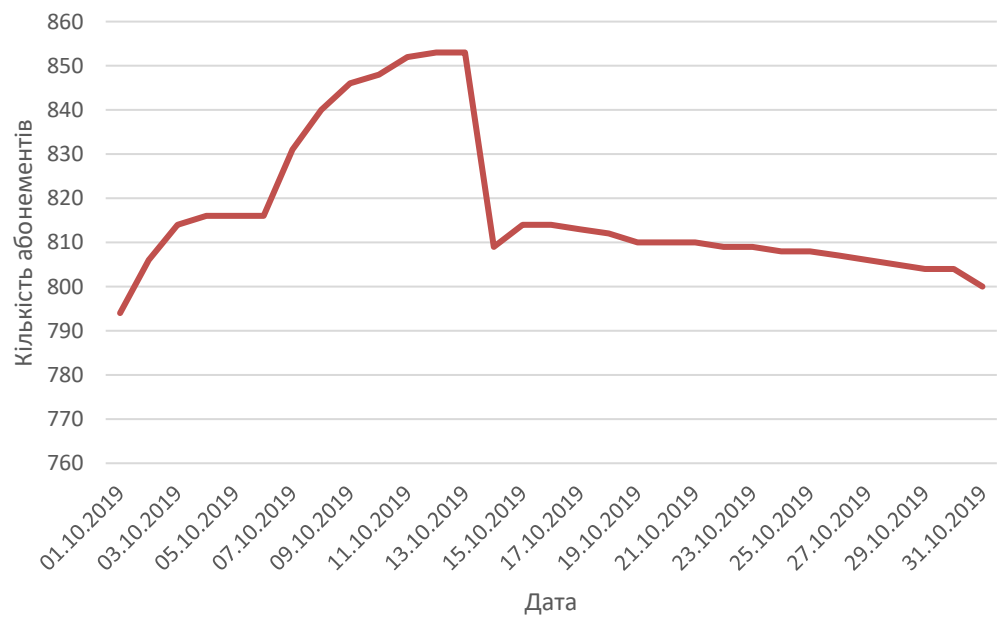


Рисунок 5.22 – Щоденна кількість активних абонементів для занять фітнесом за жовтень 2019 року

Приклад місяця, коли зменшення кількості абонементів не відбувалось, можна помітити на рис. 5.23.

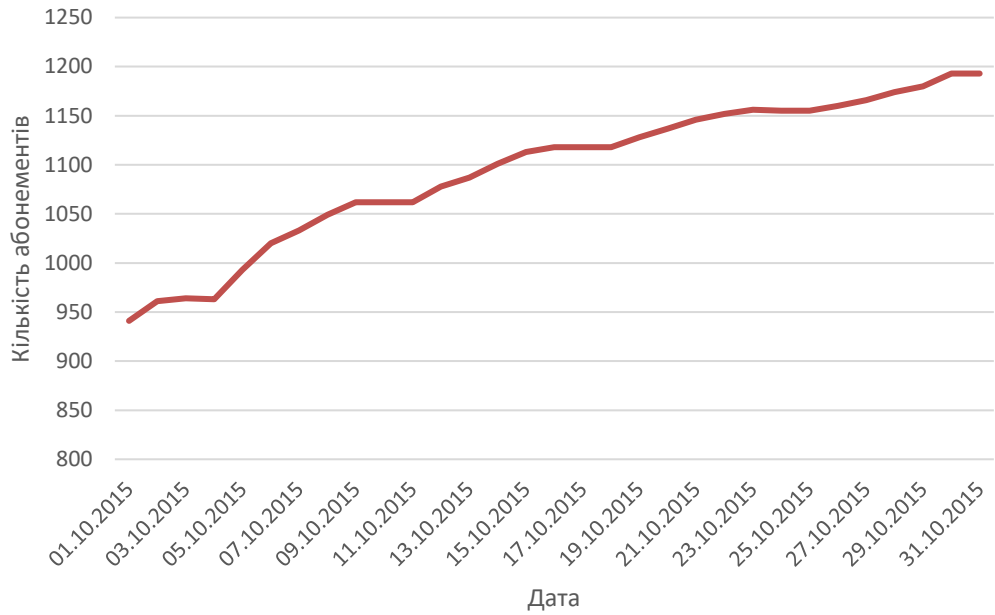


Рисунок 5.23 – Щоденна кількість активних абонементів для занять фітнесом за жовтень 2015 року

На рис. 5.24 показано зміну кількості абонементів для занять командними видами спорту за відвідуваності більше 25 % та не більше 50 %. У цьому випадку кількість фактично весь час збільшується. Тоді нормування призводить до того, що дані під час тестування весь час більші за 1.

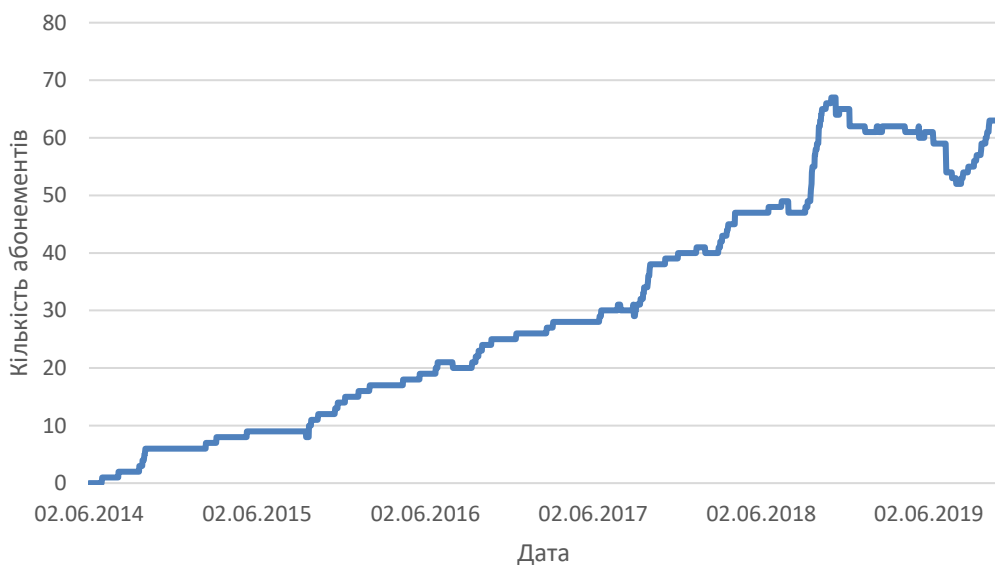


Рисунок 5.24 – Щоденна кількість активних абонементів для занять командними видами спорту за відвідуваності більше 25 % та не більше 50 %

На рис. 5.25 і 5.26 показано щоденну кількість абонементів для людей з обмеженими можливостями за відвідуваності більше 25 % та не більше 50 % та для занять фітнесом за відвідуваності більше 0 % та не більше 25 %.

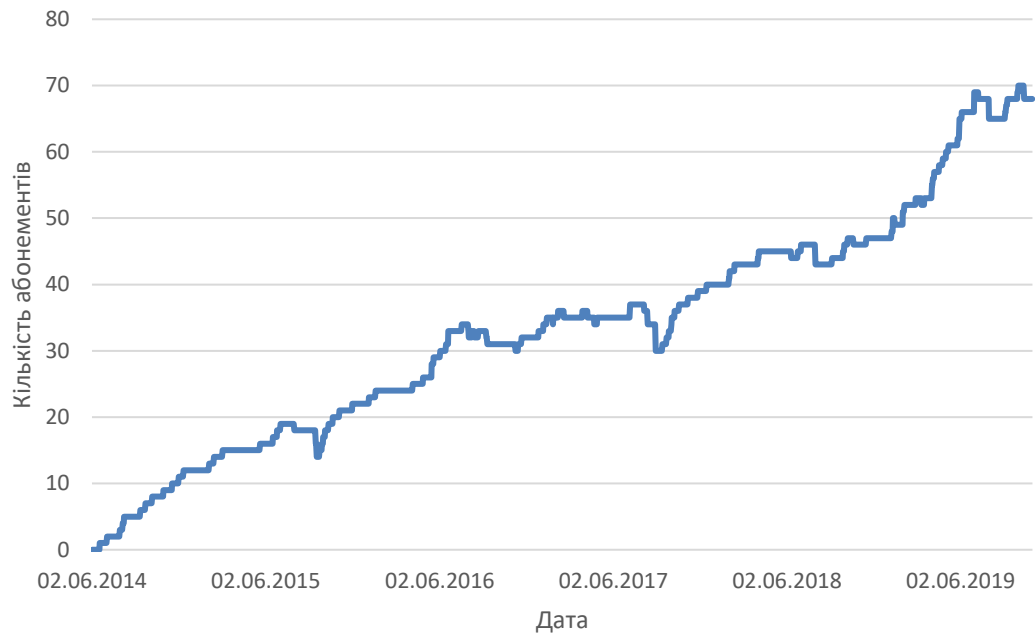


Рисунок 5.25 – Щоденна кількість активних абонементів для людей з обмеженими можливостями за відвідуваності більше 25 % та не більше 50 %

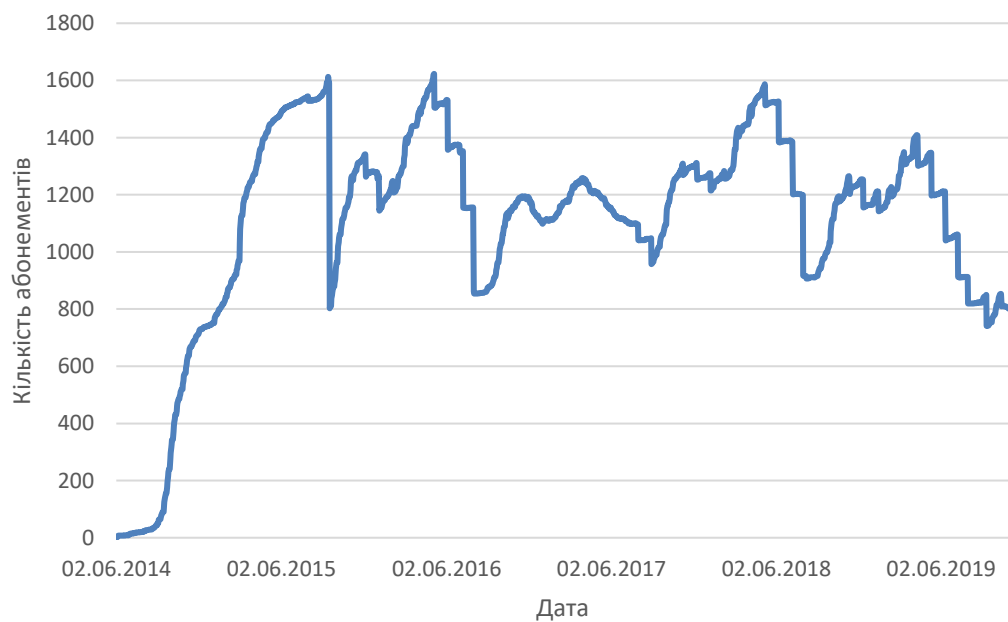


Рисунок 5.26 – Щоденна кількість активних абонементів для занять фітнесом за відвідуваності більше 0 % та не більше 25 %

На рис. 5.26 видно, що не завжди, а радше як виключення, відбувається постійне збільшення кількості абонементів. У стандартних випадках наявні певні коливання. Тому експертне обмеження майбутніх значень потрібно саме у випадках постійного збільшення значень за весь час спостережень.

Результати експериментального дослідження було оцінено за допомогою середньоквадратичної похибки. Ці результати були отримані для всіх моделей для груп відвідуваності в 4 варіантах зокрема для моделей Random Forest (табл. 5.1).

Таблиця 5.1 – Середньоквадратична похибка моделей Random Forest

Вид спортивних активностей / відсоток відвідуваності	(0; 25]	(25; 50]	(50;100)	0
Легка атлетика	0,00131	0,00076	0,13641	0,00136
Водні види спорту (басейн)	0,00053	0,02092	0,02892	0,00252
Фітнес	0,00106	0,05931	0,01443	0,00079
Командні види спорту	0,00185	0,0149	0,00141	0,001
Види спорту з ракетками	0,00127	0,00049	2,72E-08	8,93E-06
Бойові мистецтва	0,00094	0,007	0,23407	0,00285
Спорт для людей з обмеженими можливостями	0,00062	0,10088	0,00018	0,00265

Моделі LSTM спочатку було побудовано за 100 ітерацій навчання (без скорочення кількості) та також оцінено середньоквадратичною похибкою (табл. 5.2).

Таблиця 5.2 – Середньоквадратична похибка моделей LSTM за навчання у 100 ітерацій

Вид спортивних активностей / відсоток відвідуваності	(0; 25]	(25; 50]	(50;100)	0
Легка атлетика	0,0013	0,00098	0,00521	0,0028
Водні види спорту (басейн)	0,00054	0,00279	0,00654	0,00119
Фітнес	0,00115	0,00205	0,00444	0,00181
Командні види спорту	0,00055	0,02584	0,00272	0,00049
Види спорту з ракетками	0,00014	0,00006	0,00003	0,00051
Бойові мистецтва	0,00218	0,00217	0,0116	0,00036
Спорт для людей з обмеженими можливостями	0,0006	0,02099	0,00021	0,00083

Також було створено, навчено моделі LSTM у відповідності з методом, який запропоновано (табл. 5.3). Максимально використовувалось 500 ітерацій. Результати виявились кращими ніж при використанні 100 ітерацій. Окрім того з результатів помітно, що вони є кращими і за використання Random Forest при створенні ансамблю з 1000 елементів. Хоча в певних варіантах комбінацій виду спортивних активностей та групи за відсотком відвідуваності Random Forest зміг продемонструвати кращі результати. Це спостерігалось у 4 випадках з 28. У всіх інших випадках кращими виявились моделі, створені у відповідності з методом, який запропоновано.

Таблиця 5.3 – Середньоквадратична похибка моделей LSTM, створених у відповідності з методом, який запропоновано

Вид спортивних активностей / відсоток відвідуваності	(0; 25]	(25; 50]	(50;100)	0
Легка атлетика	0,00115	0,00068	0,00392	0,00155
Водні види спорту (басейн)	0,00052	0,00221	0,00295	0,00178
Фітнес	0,00081	0,00078	0,00067	0,00044
Командні види спорту	0,00061	0,00174	0,0011	0,00049
Види спорту з ракетками	0,00003	0,00002	0,00005	0,00003
Бойові мистецтва	0,00086	0,00079	0,00274	0,00096
Спорт для людей з обмеженими можливостями	0,00109	0,00376	0,00009	0,00089

Підсумкові результати експериментального дослідження (табл. 5.4) включають результати моделей Random Forest та моделей LSTM, створених у відповідності з методом, який запропоновано. Оцінювання відбувалось шляхом усереднення середньоквадратичної похибки серед 4 груп за відвідуваністю для кожного виду спортивних активностей. Також використано оцінку, яку названо зваженою. Для зваженої оцінки знайдено не усереднене значення, а враховано ваги. Ваги відповідають відсотку для кожної групи відвідуваності за 2019 рік. Отримані результати у другому випадку дещо відрізняються, але в підсумку за обох показників для всіх видів активностей Random Forest призвели до отримання гірших результатів порівняно з моделями LSTM у відповідності з методом, який запропоновано в роботі.

Таблиця 5.4 – Підсумкові результати експериментального дослідження

Вид спортивних активностей / відсоток відвідуваності	Random Forest		LSTM за запропонованим методом	
	Середньо-квадратична похибка	Зважена середньо-квадратична похибка	Середньо-квадратична похибка	Зважена середньо-квадратична похибка
Легка атлетика	0,03496	0,01931	0,00183	0,00152
Водні види спорту (басейн)	0,01322	0,01199	0,00187	0,00162
Фітнес	0,0189	0,01401	0,00067	0,00076
Командні види спорту	0,00479	0,00412	0,00098	0,00082
Види спорту з ракетками	0,00044	9,16E-04	3,1573E-05	2,81715E-05
Бойові мистецтва	0,06122	0,01719	0,00134	0,00099
Спорт для людей з обмеженими можливостями	0,02608	0,04043	0,00146	0,00199
Середнє значення	0,0228	0,01542	0,00117	0,0011

5.4 Висновки за розділом 5

Описані результати п'ятого розділу включають принципи роботи над виконанням функцій, що забезпечуються програмою, результати проведеного експериментального дослідження на основі даних, зібраних у спортивному центрі в Лісабоні з 1 червня 2014 року до 31 жовтня 2019 року. Моделі було побудовано на основі використання методу Random Forest і методу, який описано в розділі 2. Результати вказують на ефективність застосування

моделей LSTM, створених у відповідності з методом, який запропоновано, для всіх видів спортивних активностей як за використання середньоквадратичної похибки, так і у випадку зважування значень, отриманих для кожної групи відвідуваності (нульова відвідуваність, від 0 до 25 % включно, більше 25 % і до 50 % включно та більше 50 %).

ВИСНОВКИ

У дипломній кваліфікаційній роботі розроблено програмне забезпечення підтримки роботи клубу для занять спортом з клієнтами для прийняття рішень щодо подальшої діяльності на основі результатів прогнозування кількості активних абонементів для занять спортом. Розроблено вебзастосунок, який дозволяє популяризувати спортивні активності, тренерів у спортивному клубі, організувати підтримку бронювання тренувань, продажу абонементів, забезпечити прогнозування кількості абонементів у розрізі видів спортивних активностей та груп за відсотком відвідуваності клієнтами спортивного залу. Для розробки використано мову Python як основну мову розробки, фреймворк Django для веброзробки, систему PostgreSQL для організації і керування базою даних.

Наукова новизна роботи полягає в методі прогнозування кількості активних абонементів для занять спортом, який дозволяє прогнозувати кількість абонементів у майбутні дні за групою відвідуваності та спортивною активністю за кількістю абонементів у минулі дні, застосовуючи для побудови нейронних мереж архітектуру LSTM та застосовуючи правила нормування даних.

Виконано експериментальне дослідження на основі даних, зібраних у спортивному центрі в Лісабоні з 1 червня 2014 року до 31 жовтня 2019 року. Було побудовано моделі на основі використання методу Random Forest і методу, запропонованого в дипломній кваліфікаційній роботі. Результати вказують на ефективність застосування моделей LSTM, створених у відповідності з методом, який запропоновано, для всіх видів спортивних активностей як за використання середньоквадратичної похибки, так і у випадку зважування значень, отриманих для кожної групи відвідуваності (нульова відвідуваність, від 0 до 25 % включно, більше 25 % і до 50 % включно та більше 50 %).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Sport Life – мережа фітнес-клубів № 1 в Україні. Sport Life [Електрон. ресурс]. – Режим доступу : <https://sportlife.ua/uk/>.
2. Оздоровчий центр «ТРИ СТИХІЇ» [Електрон. ресурс]. – Режим доступу : <https://trystyhii.com.ua/>.
3. Palestra Fitness Club [Електрон. ресурс]. – Режим доступу : <https://palestra.ua/>.
4. Simple Healthy Living. Livestrong [Electronic resource]. – Access mode : <https://www.livestrong.com/>.
5. Kotu, V. Data Science Concepts and Practice [Text] / Vijay Kotu, Bala Deshpande. – Cambridge : Morgan Kaufmann Publishers, 2019. – 568 p.
6. What is random forest? IBM [Electronic resource]. – Access mode : <https://www.ibm.com/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems.>
7. Donges, N. Random Forest [Electronic resource] : A Complete Guide for Machine Learning / N. Donges. – Access mode : <https://builtin.com/data-science/random-forest-algorithm>.
8. Random Forest Algorithm [Electronic resource]. – Access mode : <https://www.simplilearn.com/tutorials/machine-learning-tutorial/random-forest-algorithm>.
9. Rafaat, Y. Ensemble Learning Techniques [Electronic resource] : A Walkthrough with Random Forests in Python / Y. Rafaat. – Access mode : <https://www.kdnuggets.com/ensemble-learning-techniques-a-walkthrough-with-random-forests-in-python>.
10. Introduction to Long Short-Term Memory (LSTM) [Electronic resource]. – Access mode : <https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/lstm>.
11. Python Tutorial [Electronic resource]. – Access mode : <https://www.w3schools.com/python/>.

12. Django Project MVT Structure [Electronic resource]. – Access mode : <https://www.geeksforgeeks.org/django-project-mvt-structure/>.
13. CaterServ – Free Bootstrap 5 Business Landing Page Template [Electronic resource]. – Access mode : <https://themewagon.com/themes/caterserv/>.
14. Sport Services Dataset – Mendeley Data [Electronic resource]. – Access mode : <https://data.mendeley.com/datasets/yprk4jdgnv/1>.
15. Pinheiro, P. Regular sports services: Dataset of demographic, frequency and service level agreement [Electronic resource] / P. Pinheiro, L. Cavique // Data in Brief. – 2021. – Volume 36. – Article 107054. – Access mode : <https://www.sciencedirect.com/science/article/pii/S2352340921003383>.

ДОДАТОК А
Технічне завдання

Вступ

Здоровий образ життя є достатньо популярним у світі, але окрім того популярним є образ і людини з гарним тілом. Це сприяє тому, щоб велика кількість людей долучалась до занять спортом. В результаті кількість функціонуючих клубів для занять спортом у багатьох містах є достатньо великою.

Однак популярність образів не завжди призводить до того, що кожна людина після зацікавлення заняттями спортом та подальшого придбання абонементів такі заняття відвідує. Нерідко далі придбання абонементів справи не заходять. Для спортивного клубу це призводить до проблем з оптимальним використанням наявних засобів. Тобто кількість придбаних абонементів може бути такою, що не всі люди зможуть одночасно перебувати у спортивному клубі, якщо відвідають його саме в один момент.

Не всі відвідують клуб в один і той самий момент. Відвідування відбувається в різні дні, різний час. Але не всі власники абонементів взагалі відвідують спортивний клуб в підсумку, інші роблять це зрідка. У підсумку важливо не тільки зацікавити відвідувачів доєднуватися до спортивного клубу, але і далі розуміти, як кількість абонементів буде змінюватися, при чому змінюватися в різних категоріях відвідуваності та за різними активностями, адже вони потребують різних приміщень, різного обладнання, різних тренерів.

Тож саме таке програмне забезпечення необхідно розробити.

A.1 Підстави для розробки

Наказ № 437 від 14 листопада 2023 р. за Національним університетом «Запорізька політехніка» з зазначенням теми роботи «Дослідження та програмна реалізація методів прогнозування показників діяльності клубу для занять спортом» та завдання на роботу є підставами для розробки.

A.2 Призначення розробки

Програма, яку потрібно розробити, має дозволити популяризувати спортивні активності, тренерів у спортивному клубі, підтримувати бронювання тренувань з персональним тренером, а з іншого боку визначати статистику відвідуваності та забезпечити прогнозування кількості абонементів у широкому розрізі для подальшого визначення показників діяльності клубу для занять спортом.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Забезпечення функціональних характеристик повинно вкладатися в наступний набір функцій:

- авторизація;
- визначення даних профілю;
- відображення інформації про тренера;
- реєстрація відвідувачів;
- реєстрація тренерів менеджерами;
- реєстрація менеджерів менеджерами;
- реєстрація тренування в спортивному клубі;
- відмовлення у тренуванні в спортивному клубі;
- відображення зареєстрованих на тренуванні на поточний момент;
- відображення фотографій з тренувань;
- додавання фотографії з тренувань;
- видалення фотографії з тренувань;
- відображення інформації про спортивний клуб;
- редагування інформації про спортивний клуб;
- відображення інформації про вартість абонементів;
- додавання пропозиції абонементу;

- видалення пропозиції абонементу;
- редагування пропозиції абонементу;
- відображення задіяності тренера;
- відображення статистики використання абонементів;
- відображення статистики відвідування спортивної активності;
- відображення статистики активних абонементів;
- прогнозування кількості активних абонементів;
- створення відгуку про тренера;
- відображення даних зали клубу;
- відображення відгуків про тренера;
- відображення власних тренерів відвідувача;
- відображення відвідувачів у тренера;
- створення групи тренування;
- редагування групи тренування;
- деактивування групи тренування;
- відображення заявок на тренування тренеру;
- відображення власних груп тренування відвідувача;
- відображення історії тренувань в спортивному клубі;
- відображення стану абонементу;
- придбання абонементу для тренування в спортивному клубі;
- відображення плану тренувань в спортивному клубі, в тому числі за відвідувачем, за активністю, за тренером;
- планування індивідуального тренування в спортивному клубі;
- скасування індивідуального тренування в спортивному клубі;
- планування спільного тренування в спортивному клубі;
- скасування спільного тренування в спортивному клубі;
- відображення комплексу вправ;
- відображення комплексів вправ на групу м'язів;
- пошук комплексів вправ;
- публікація комплексу вправ;

- відображення комплексів вправ від тренера;
- редагування комплексу вправ від тренера;
- відображення активностей спортивного клубу;
- відображення порад щодо занять спортом від тренера;
- внесення поради до власної скарбнички;
- видалення поради з власної скарбнички;
- відображення власної скарбнички;
- реєстрація стану тренування тренером;
- публікація поради щодо занять спортом;
- редагування поради щодо занять спортом;
- деактивація поради щодо занять спортом;
- визначення доступного розкладу тренувань тренера, включаючи додавання нових слотів, редагування існуючих слотів та вилучення існуючих слотів;
- подання заявки на тренування з тренером;
- підтвердження або скасування заявки тренером;
- відображення доступних слотів для тренувань у тренера;
- відображення поради щодо занять спортом повністю;
- сплата за персональне тренування;
- визначення тарифу на персональне заняття;
- бронювання заняття з тренером;
- скасування заняття з тренером;
- відображення тренерів у спортивному клубі;
- відображення менеджерів у спортивному клубі;
- деактивація облікового запису тренера;
- деактивація облікового запису менеджера;
- деактивація облікового запису відвідувача;
- пошук відвідувачів;
- редагування активності спортивного клубу;
- додавання активності спортивного клубу;

- пошук порад щодо занять спортом;
- рекомендування комплексу вправ;
- скасування рекомендації комплексу вправ;
- рекомендування поради від тренера;
- скасування рекомендації поради від тренера.

A.3.2 Умови експлуатації

Програмне забезпечення в підсумку має підтримуватись під час експлуатації через:

- організацію хостингу для роботи з сервером;
- розміщення на сервері вебзастосунку з базовим налаштуванням Django;
- наявність інтерпретатора python 3;
- встановлення пакетів Tensorflow, Keras.

Або у складі цього серверу, або як окремий сервер БД має бути використано і налаштовано систему PostgreSQL.

A.3.3 Вимоги до складу та параметрів технічних засобів

Вимоги до організації технічних засобів сервера включають мінімально:

- процесор Dual Intel Xeon, який має частоту 3,2 ГГц;
- 4 Гб доступної оперативної пам'яті;
- 10 Гб доступної пам'яті на жорсткому диску.

A.4 Порядок контролю та приймання

Строки виконання дипломної кваліфікаційної роботи магістра визначені в завданні на роботу безпосередньо, включаючи строк подання

роботи, календарний план виконання роботи з зазначеними етапами, результатами та строками.

Захист роботи відбувається за оголошеним завчасно графіком захисту дипломних кваліфікаційних робіт і відповідно графіком роботи екзаменаційної комісії.

ДОДАТОК Б
Текст програми

Б.1 Текст файлу trainerviews.py

```

from django.shortcuts import render
from .models import Trainer, Workout, Advice,
Personalapplication, Trainingpurchase, User, Trainereview,
Slotbooking, Trainingslot, Enrollment, Trainingtariff
from django.shortcuts import get_object_or_404
from django.http import HttpResponseRedirect
from .forms import WorkoutForm, AdviceForm,
TrainingslotForm, TrainingtariffForm, SlotbookingdetailForm
from django.shortcuts import redirect

def trainer_view(request):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        return render(request, "stansport/profile.html",
{"trainer": auth_trainer, "user": request.user, "type":
"trainer" })
    else:
        return render(request, "stansport/profile.html",
{"trainer": None, "user": None, "type": "Сторінка доступна
тільки для авторизованого користувача, якщо він є тренером
клубу. Авторизуйтеся спочатку відповідним чином"})

def trainer_workouts_view(request):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        trainer_workouts = Workout.objects.filter(trainer =
auth_trainer).annotate(Count("recommendations",
distinct=True).select_related("activity").prefetch_related("work
outmuscles").prefetch_related("workoutmuscles_muscles").order_by
("-id"))
        return render(request, "stansport/workouts.html",
{"trainer": auth_trainer, "workouts": trainer_workouts.values(),
"count": trainer_workouts.count(), "message": ""})
    else:
        return render(request, "stansport/workouts.html",
{"trainer": None, "workouts": None, "count": 0, "message":
"Сторінка доступна тільки для авторизованого користувача, якщо
він є тренером клубу. Авторизуйтеся спочатку відповідним
чином"})

def trainer_advices_view(request):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)

```

```

        trainer_advices = Advice.objects.filter(trainer =
auth_trainer).annotate(Count("recommendations",
distinct=True)).annotate(Count("boxes",
distinct=True)).select_related("activity").prefetch_related("advi
cemuscles").prefetch_related("advicemuscles_muscles").order_by("-
id")

        return render(request, "stansport/advices.html",
{"trainer": auth_trainer, "advices": trainer_advices.values(),
"count": trainer_advices.count(), "message": ""})
    else:
        return render(request, "stansport/advices.html",
{"trainer": None, "advices": None, "count": 0, "message":
"Сторінка доступна тільки для авторизованого користувача, якщо
він є тренером клубу. Авторизуйтеся спочатку відповідним
чином"})

def trainer_applications_view(request, application_state):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        trainer_applications =
Personalapplication.objects.filter(trainer = auth_trainer,
active = application_state).order_by("-created_time")
        return render(request,
"stansport/applications.html", {"trainer": auth_trainer,
"applications": trainer_applications.values(), "count":
trainer_applications.count(), "message": ""})
    else:
        return render(request,
"stansport/applications.html", {"trainer": None, "applications":
None, "count": 0, "message": "Сторінка доступна тільки для
авторизованого користувача, якщо він є тренером клубу.
Авторизуйтеся спочатку відповідним чином"})

def application_approve_view(request, application_id):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        trainer_application =
get_object_or_404(Personalapplication, trainer = auth_trainer,
id = application_id)
        if trainer_application.active == 0 or
trainer_application.active == 2:
            trainer_application.active = 1
            trainer_application.save()
            return render(request,
"stansport/application_result.html", {"trainer": auth_trainer,
"application": trainer_application, "message": "Заявку було
успішно підтверджено."})
    else:

```

```

        return render(request,
            "stansport/application_result.html", {"trainer": None,
            "application": None, "message": "Сторінка доступна тільки для
авторизованого користувача, якщо він є тренером клубу.
Авторизуйтеся спочатку відповідним чином"})

    def application_cancel_view(request, application_id):
        if request.user.is_authenticated:
            auth_user_object = request.user
            auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
            trainer_application =
get_object_or_404(Personalapplication, trainer = auth_trainer,
id = application_id)
            if trainer_application.active == 0:
                trainer_application.active = 2
                trainer_application.save()
                return render(request,
                    "stansport/application_result.html", {"trainer": auth_trainer,
                    "application": trainer_application, "message": "Заявку було
успішно скасовано."})
            elif trainer_application.active == 1:
                application_visitor =
trainer_application.visitor
                application_time =
trainer_application.reaction_time
                purchases =
Trainingpurchase.objects.filter(visitor = application_visitor,
trainingtariff__trainer = auth_trainer, purchased_time__gte =
application_time).order_by("-purchased_time")
                if purchases and purchases.count() > 0:
                    trainer_application.active = 2
                    trainer_application.save()
                    return render(request,
                        "stansport/application_result.html", {"trainer": auth_trainer,
                        "application": trainer_application, "message": "Заявку було
успішно скасовано."})
                else:
                    return render(request,
                        "stansport/application_result.html", {"trainer": None,
                        "application": None, "message": "Сторінка доступна тільки для
авторизованого користувача, якщо він є тренером клубу.
Авторизуйтеся спочатку відповідним чином"})

    def advice_view(request, advice_id):
        if request.user.is_authenticated:
            auth_user_object = request.user
            auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
            advices =
Advice.objects.prefetch_related('recommendations').select_relate
d("activity")

```

```

        trainer_advice = get_object_or_404(advice, id =
advice_id)
        return render(request, "stansport/advice.html",
{"trainer": auth_trainer, "advice": trainer_advice, "message":
""})
    else:
        return render(request, "stansport/advice.html",
{"trainer": None, "advice": None, "message": "Сторінка доступна
тільки для авторизованого користувача, якщо він є тренером
клубу. Авторизуйтеся спочатку відповідним чином"})

def workout_view(request, workout_id):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        workouts =
Workout.objects.prefetch_related('recommendations').select_relat
ed("activity")
        trainer_workout = get_object_or_404(workouts, id =
workout_id)
        return render(request, "stansport/workout.html",
{"trainer": auth_trainer, "workout": trainer_workout, "message":
""})
    else:
        return render(request, "stansport/workout.html",
{"trainer": None, "workout": None, "message": "Сторінка доступна
тільки для авторизованого користувача, якщо він є тренером
клубу. Авторизуйтеся спочатку відповідним чином"})

def trainer_reviews_view(request, min_score, max_score):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        trainer_reviews =
Trainerreview.objects.filter(trainer = auth_trainer, score__gte =
min_score, score__lte =
max_score).prefetch_related("visitor").order_by("-
published_time")
        return render(request, "stansport/reviews.html",
{"trainer": auth_trainer, "reviews": trainer_reviews.values(),
"count": trainer_reviews.count(), "message": ""})
    else:
        return render(request, "stansport/reviews.html",
{"trainer": None, "reviews": None, "count": 0, "message":
"Сторінка доступна тільки для авторизованого користувача, якщо
він є тренером клубу. Авторизуйтеся спочатку відповідним
чином"})

def trainer_visitors_view(request):
    if request.user.is_authenticated:
        auth_user_object = request.user

```

```

        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        visitors =
Trainingpurchase.objects.filter(trainingtariff__trainer =
auth_trainer).order_by("-
purchased_time").values_list('visitor').distinct()
        visitors_count = visitors.count()
        trainer_visitors = []
        for visitor in visitors:
            current_visitor = {}
            current_visitor['visitor'] = visitor
            current_visitor['visits'] =
Slotbooking.objects.filter(visitor = visitor, state = 1,
trainingslot__trainer = auth_trainer).count()
            trainer_visitors.append(current_visitor)
        return render(request,
"stansport/trainer_visitors.html", {"trainer": auth_trainer,
"visitors": trainer_visitors, "count": visitors_count,
"message": ""})
    else:
        return render(request,
"stansport/trainer_visitors.html", {"trainer": None,
"application": None, "message": "Сторінка доступна тільки для
авторизованого користувача, якщо він є тренером клубу.
Авторизуйтеся спочатку відповідним чином"})

def trainer_visitor_history_view(request,
visitor_username):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        for_user = get_object_or_404(User, username =
visitor_username)
        bookings = Slotbooking.objects.filter(visitor =
for_user, state = 1, trainingslot__trainer =
auth_trainer).order_by("-end_time")
        bookings_count = bookings.count()
        return render(request,
"stansport/trainer_visitors.html", {"trainer": auth_trainer,
"bookings": bookings.values(), "count": bookings_count,
"message": ""})
    else:
        return render(request,
"stansport/trainer_visitors.html", {"trainer": None, "bookings":
None, "count": 0, "message": "Сторінка доступна тільки для
авторизованого користувача, якщо він є тренером клубу.
Авторизуйтеся спочатку відповідним чином"})

def trainer_slotbookings_view(request, min_date, max_date):
    if request.user.is_authenticated:
        auth_user_object = request.user

```

```

        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        trainer_slots = Trainingslot.objects.filter(trainer
= auth_trainer).order_by("-end_time")
        slot_bookings = []
        for trainer_slot in trainer_slots:
            slot_start = trainer_slot.start_time
            slot_end = trainer_slot.end_time
            trainer_bookings =
Slotbooking.objects.filter(trainingslot__trainer = auth_trainer,
start_time__gte = slot_start, end_time__lte =
slot_end).select_related("visitor")
            slot_bookings.append(trainer_bookings)
        return render(request,
"stansport/trainer_slots.html", {"trainer": auth_trainer,
"slots": trainer_slots.values(), "bookings":
slot_bookings.values(), "message": ""})
    else:
        return render(request, "stansport/advice.html",
{"trainer": None, "slots": None, "bookings": None, "message":
"Сторінка доступна тільки для авторизованого користувача, якщо
він є тренером клубу. Авторизуйтеся спочатку відповідним
чином"})

import datetime
from dateutil.relativedelta import relativedelta

def trainer_enrolled_view(request):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        moment = datetime.datetime.now()
        day = moment
        day.hour = 0
        day.minute = 1
        trainer_bookings =
Slotbooking.objects.filter(trainingslot__trainer = auth_trainer,
start_time__lte = moment, end_time__gte = moment, state =
1).select_related("visitor")
        enrolled = []
        for trainer_booking in trainer_bookings:
            current_visitor = trainer_booking.visitor
            enrollments = Enrollment.objects.filter(visitor
= current_visitor, start_time__gte = day, finish_time__isnull =
False)

            if enrollments:
                enrolled.append(current_visitor)
        return render(request,
"stansport/trainer_enrolled.html", {"trainer": auth_trainer,
"visitors": enrolled.values(), "message": ""})
    else:

```

```

        return render(request,
            "stansport/trainer_enrolled.html", {"trainer": None, "visitors":
            None, "message": "Сторінка доступна тільки для авторизованого
            користувача, якщо він є тренером клубу. Авторизуйтеся спочатку
            відповідним чином"})

    def trainer_engagement_view(request, period, day_start,
month_start, year_start, day_end, month_end, year_end):
        if request.user.is_authenticated:
            auth_user_object = request.user
            auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
            if year_start == 0 or year_end == 0:
                bookings = Slotbooking.objects.filter(state =
1, trainingslot__trainer = auth_trainer).order_by("start_time")
                if year_start == 0:
                    start_booking = bookings[0]
                    start_time = start_booking.start_time
                    start_time.hour = 0
                    start_time.minute = 0
                    start_time.second = 0
                if year_end == 0:
                    end_booking = bookings[-1]
                    end_time = end_booking.start_time
                    end_time.hour = 23
                    end_time.minute = 59
                    end_time.second = 59
            else:
                start_time = datetime.datetime(year_start,
month_start, day_start, 0, 0, 0)
                end_time = datetime.datetime(year_end,
month_end, day_end, 23, 59, 59)
                engagement = []
                start = True
                while end_time <= start_time:
                    start_time_begin = start_time
                    if period == 0:
                        start_time += relativedelta(years = 1)
                    if period == 1:
                        start_time += relativedelta(months = 1)
                    if period == 2:
                        start_time += relativedelta(weeks = 1)
                    if period == 3:
                        start_time += relativedelta(days = 1)
                    if start:
                        trainer_bookings =
Slotbooking.objects.filter(trainingslot__trainer = auth_trainer,
start_time__gte = start_time_begin, end_time__lte = start_time,
state = 1).count()
                    else:
                        trainer_bookings =
Slotbooking.objects.filter(trainingslot__trainer = auth_trainer,

```

```

start_time__gt = start_time_begin, end_time__lte = start_time,
state = 1).count()
        engagement.append(trainer_bookings)
        return render(request,
"stansport/trainer_engagement.html", {"trainer": auth_trainer,
"engagement": engagement, "message": ""})
    else:
        return render(request,
"stansport/trainer_engagement.html", {"trainer": None,
"engagement": None, "message": "Сторінка доступна тільки для
авторизованого користувача, якщо він є тренером клубу.
Авторизуйтеся спочатку відповідним чином"})

def workout_publish_view(request):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        if request.method == "POST":
            workoutform = WorkoutForm(request.POST)
            if workoutform.is_valid():
                workout_instance = workoutform.save(commit
= False)

                workout_instance.trainer = auth_trainer
                workout_instance.save()
                workout_instance.save_m2m()
                return
            redirect('stansport.views.trainerworkoutview', workout_id =
workout_instance.id)
        else:
            workoutform = None
            return render(request,
"stansport/workout_publish.html", {"form": workoutform,
"message": ""})
    else:
        return render(request,
"stansport/workout_publish.html", {"form": None, "message":
"Сторінка доступна тільки для авторизованого користувача, якщо
він є тренером клубу. Авторизуйтеся спочатку відповідним чином"
})

def advice_publish_view(request):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        if request.method == "POST":
            adviceform = AdviceForm(request.POST)
            if adviceform.is_valid():
                advice_instance = adviceform.save(commit =
False)

                advice_instance.trainer = auth_trainer
                advice_instance.save()

```

```

        advice_instance.save_m2m()
        return
    redirect('stansport.views.traineradviceview', advice_id =
advice_instance.id)
    else:
        adviceform = None
        return render(request,
"stansport/advice_publish.html", {"form": adviceform, "message":
""})
    else:
        return render(request,
"stansport/advice_publish.html", {"form": None, "message":
"Сторінка доступна тільки для авторизованого користувача, якщо
він є тренером клубу. Авторизуйтеся спочатку відповідним
чином"})

def slot_add_view(request):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        if request.method == "POST":
            trainslotform = TrainingslotForm(request.POST)
            if trainslotform.is_valid():
                trainingslot_instance =
trainslotform.save(commit = False)
                trainingslot_instance.trainer =
auth_trainer
                trainingslot_instance.state = 0
                trainingslot_instance.save()
                return
            redirect('stansport.views.traineradviceview', advice_id =
advice_instance.id)
        else:
            trainslotform = None
            return render(request, "stansport/slot_add.html",
{"form": trainslotform, "message": ""})
        else:
            return render(request, "stansport/slot_add.html",
{"form": None, "message": "Сторінка доступна тільки для
авторизованого користувача, якщо він є тренером клубу.
Авторизуйтеся спочатку відповідним чином"})

def slot_update_view(request, trainingslot_id):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
    else:
        return render(request,
"stansport/slot_update.html", {"form": None, "message":
"Сторінка доступна тільки для авторизованого користувача, якщо

```

він є тренером клубу. Авторизуйтеся спочатку відповідним чином"}})

```

    try:
        trainingslot = Trainingslot.objects.get(id =
trainingslot_id).select_related("trainer")
    except Trainingslot.DoesNotExist:
        trainingslot = None
    message = ""
    if trainingslot.trainer == auth_trainer:
        if request.method == "POST":
            trainslotform = TrainingslotForm(request.POST,
instance = trainingslot)
            if trainslotform.is_valid():
                slots =
Trainingslot.objects.get(start_time__gte =
trainslotform.start_time, end_time__lte =
trainslotform.end_time, trainer = auth_trainer)
                trainslotform.delete()
                if slots:
                    message = "Зміни внесені. Вказаний слот
об'єднано з існуючим"
                return
            redirect('stansport.views.trainerslotsview', trainer_id =
auth_trainer)
                slots =
Trainingslot.objects.get(start_time__lte =
trainslotform.start_time, end_time__gte =
trainslotform.end_time, trainer = auth_trainer).values()[0]
                slots.delete()
                trainslotform.save()
                if slots:
                    message = "Зміни внесені. Вказаний слот
об'єднано з існуючим"
                return
            redirect('stansport.views.trainerslotsview', trainer_id =
auth_trainer)
                slots =
Trainingslot.objects.get(start_time__lte =
trainslotform.start_time, end_time__gte =
trainslotform.start_time, end_time__lte = trainslotform.end_time,
trainer = auth_trainer).values()[0]
                slots.start_time = trainslotform.start_time
                slots.save()
                if slots:
                    message = "Зміни внесені. Вказаний слот
об'єднано з існуючим"
                return
            redirect('stansport.views.trainerslotsview', trainer_id =
auth_trainer)
                slots =
Trainingslot.objects.get(start_time__gte =
trainslotform.start_time, start_time__lte =

```

```

trainslotform.end_time, end_time_gte = trainslotform.end_time,
trainer = auth_trainer).values()[0]
        slots.end_time = trainslotform.end_time
        slots.save()
        if slots:
            message = "Зміни внесені. Вказаний слот
об'єднано з існуючим"
            return
redirect('stansport.views.trainerslotsview', trainer_id =
auth_trainer)
        trainslotform.save()
        return
redirect('stansport.views.trainerslotsview', trainer_id =
auth_trainer)
    else:
        trainslotform = TrainingslotForm(None, instance
= trainingslot)
    else:
        trainslotform = None
        message = "Сторінка доступна тільки для
авторизованого користувача, якщо він є тренером клубу.
Авторизуйтеся спочатку відповідним чином"
        return render(request, "stansport/slot_update.html",
{"form": trainslotform, "message": message})

def workout_update_view(request, workout_id):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
    else:
        return render(request,
"stansport/workout_update.html", {"form": None, "message":
"Сторінка доступна тільки для авторизованого користувача, якщо
він є тренером клубу. Авторизуйтеся спочатку відповідним
чином"})
    try:
        workout = Workout.objects.get(id =
workout_id).select_related("trainer")
    except Workout.DoesNotExist:
        workout = None
    message = ""
    if workout.trainer == auth_trainer:
        if request.method == "POST":
            workoutform = WorkoutForm(request.POST,
instance = workout)
            if workoutform.is_valid():
                workoutform.save()
                return
redirect('stansport.views.trainerworkoutview', workout_id =
workoutform.id)
        else:

```

```

        workoutform = WorkoutForm(None, instance =
workout)
    else:
        workoutform = None
        message = "Сторінка доступна тільки для
авторизованого користувача, якщо він є тренером клубу.
Авторизуйтеся спочатку відповідним чином"
        return render(request, "stansport/workout_update.html",
{"form": workoutform, "message": message})

    def advice_update_view(request, advice_id):
        if request.user.is_authenticated:
            auth_user_object = request.user
            auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        else:
            return render(request,
"stansport/advice_update.html", {"form": None, "message":
"Сторінка доступна тільки для авторизованого користувача, якщо
він є тренером клубу. Авторизуйтеся спочатку відповідним
чином"})
        try:
            advice = Advice.objects.get(id =
advice_id).select_related("trainer")
        except Advice.DoesNotExist:
            advice = None
            message = ""
            if advice.trainer == auth_trainer:
                if request.method == "POST":
                    adviceform = AdviceForm (request.POST, instance
= advice)
                    if adviceform.is_valid():
                        adviceform.save()
                        return
                    redirect('stansport.views.traineradviceview', advice_id =
adviceform.id)
                else:
                    adviceform = AdviceForm(None, instance =
advice)
            else:
                adviceform = None
                message = "Сторінка доступна тільки для
авторизованого користувача, якщо він є тренером клубу.
Авторизуйтеся спочатку відповідним чином"
                return render(request, "stansport/advice_update.html",
{"form": adviceform, "message": message})

    def advice_deactivate_view(request, advice_id):
        if request.user.is_authenticated:
            auth_user_object = request.user
            auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        try:

```

```

        advice = Advice.objects.get(id =
advice_id).select_related("trainer")
        except Advice.DoesNotExist:
            advice = None
        if advice and advice.trainer == auth_trainer and
advice.active = True:
            advice.active = False
            advice.save()
        return
    redirect('stansport.views.traineradviceview', advice_id =
advice.id)
    else:
        return render(request, "stansport/problem.html",
{"message": "Сторінка доступна тільки для авторизованого
користувача, якщо він є тренером клубу. Авторизуйтеся спочатку
відповідним чином"})

    def trainingslot_delete_view(request, slot_id):
        if request.user.is_authenticated:
            auth_user_object = request.user
            auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
            try:
                trainingslot = Trainingslot.objects.get(id =
slot_id).select_related("trainer")
            except Trainingslot.DoesNotExist:
                trainingslot = None
            if trainingslot and trainingslot.trainer ==
auth_trainer:
                trainingslot.delete()
                return redirect('stansport.views.trainerslotsview',
trainer_id = auth_trainer)
            else:
                return render(request, "stansport/problem.html",
{"message": "Сторінка доступна тільки для авторизованого
користувача, якщо він є тренером клубу. Авторизуйтеся спочатку
відповідним чином"})

    def trainingtariff_add_view(request):
        if request.user.is_authenticated:
            auth_user_object = request.user
            auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
            if request.method == "POST":
                trainingtariffform =
TrainingtariffForm(request.POST)
                if trainingtariffform.is_valid():
                    trainingtariff_instance =
trainingtariffform.save(commit = False)
                    trainingtariff_instance.trainer =
auth_trainer
                    trainingslot_instance.save()

```

```

        return
    redirect('stansport.views.trainertariffsvview')
    else:
        trainingtariffform = None
        return render(request, "stansport/tariff_add.html",
{"form": trainingtariffform, "message": ""})
    else:
        return render(request, "stansport/tariff_add.html",
{"form": None, "message": "Сторінка доступна тільки для
авторизованого користувача, якщо він є тренером клубу.
Авторизуйтеся спочатку відповідним чином"})

def trainingtariff_update_view(request, trainingtariff_id):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
    else:
        return render(request,
"stansport/trainingtariff_update.html", {"form": None,
"message": "Сторінка доступна тільки для авторизованого
користувача, якщо він є тренером клубу. Авторизуйтеся спочатку
відповідним чином"})
    try:
        trainingtariff = Trainingtariff.objects.get(id =
trainingtariff_id).select_related("trainer")
    except Trainingtariff.DoesNotExist:
        trainingtariff = None
    message = ""
    if trainingtariff.trainer == auth_trainer:
        if request.method == "POST":
            trainingtariffform =
TrainingtariffForm(request.POST, instance = trainingtariff)
            if trainingtariffform.is_valid():
                trainingtariffform.save()
                return
    redirect('stansport.views.trainertariffsvview', trainer =
auth_trainer)
    else:
        trainingtariffform = TrainingtariffForm(None,
instance = trainingtariff)
    else:
        trainingtariffform = None
        message = "Сторінка доступна тільки для
авторизованого користувача, якщо він є тренером клубу.
Авторизуйтеся спочатку відповідним чином"
        return render(request,
"stansport/trainingtariff_update.html", {"form":
trainingtariffform, "message": message})

def trainer_slotbookingdetail_add_view(request,
slotbooking_id):
    if request.user.is_authenticated:

```

```

        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
        try:
            slotbooking = Slotbooking.objects.get(id =
slotbooking_id).select_related("trainingslot").select_related("t
rainingslot__trainer")
        except Slotbooking.DoesNotExist:
            slotbooking = None
        if request.method == "POST":
            slotbookingdetailform =
SlotbookingdetailForm(request.POST)
            if slotbookingdetailform.is_valid() and
slotbooking.trainingslot.trainer == auth_trainer:
                slotbookingdetail_instance =
slotbookingdetailform.save(commit = False)
                slotbookingdetail_instance.slotbooking =
slotbooking
                slotbookingdetail_instance.trainer =
auth_trainer
                slotbookingdetail_instance.save()
                return
            redirect('stansport.views.slotbookingview', slotbooking_id =
slotbooking.id)
        else:
            slotbookingdetailform = None
            return render(request,
"stansport/booking_detail_add.html", {"form":
slotbookingdetailform, "message": ""})
        else:
            return render(request,
"stansport/booking_detail_add.html", {"form": None, "message":
"Сторінка доступна тільки для авторизованого користувача, якщо
він є тренером клубу. Авторизуйтеся спочатку відповідним
чином"})

def trainer_slotbookingdetail_update_view(request,
slotbookingdetail_id):
    if request.user.is_authenticated:
        auth_user_object = request.user
        auth_trainer = get_object_or_404(Trainer, user =
auth_user_object)
    else:
        return render(request,
"stansport/booking_detail_update.html", {"form": None,
"message": "Сторінка доступна тільки для авторизованого
користувача, якщо він є тренером клубу. Авторизуйтеся спочатку
відповідним чином"})
    try:
        slotbookingdetail =
Slotbookingdetail.objects.get(id =
slotbookingdetail_id).select_related("slotbooking").select_relat

```

```

ed("slotbooking__trainingslot").select_related("slotbooking__tra
iner")
        except Slotbookingdetail.DoesNotExist:
            slotbookingdetail = None
            message = ""
            if slotbookingdetail.slotbooking.trainingslot.trainer
== auth_trainer:
                if request.method == "POST":
                    slotbookingdetailform =
SlotbookingdetailForm(request.POST, instance =
slotbookingdetail)
                    if slotbookingdetailform.is_valid():
                        slotbookingdetailform.save()
                        return
            redirect('stansport.views.slotbookingview', slotbooking_id =
slotbookingdetail.slotbooking)
        else:
            slotbookingdetailform = SlotbookingdetailForm
(None, instance = slotbookingdetail)
        else:
            slotbookingdetailform = None
            message = "Сторінка доступна тільки для
авторизованого користувача, якщо він є тренером клубу.
Авторизуйтеся спочатку відповідним чином"
            return render(request, "stansport/
booking_detail_update.html", {"form": slotbookingdetailform,
"message": message})

```

Б.2 Текст файлу unvisitorviews.py

```

from django.shortcuts import render
from .models import Trainer, Trainereview, Activity,
Subscription, Muscle, Exercise, Hall
from django.shortcuts import get_object_or_404

def std_trainer_view(request, trainer_id):
    trainer = get_object_or_404(Trainer, id = trainer_id)
    return render(request, "stansport/trainer_info.html",
{"trainer": trainer})

def std_trainer_reviews_view(request, trainer_id):
    trainer = get_object_or_404(Trainer, id = trainer_id)
    trainer_reviews = Trainereview.objects.filter(trainer =
trainer).prefetch_related("visitor").order_by("-published_time")
    return render(request,
"stansport/trainer_reviews.html", {"trainer": trainer,
"reviews": trainer_reviews.values(), "count":
trainer_reviews.count()})

def std_trainer_workouts_view(request, trainer_id):
    trainer = get_object_or_404(Trainer, id = trainer_id)

```

```

        trainer_workouts = Workout.objects.filter(trainer =
trainer).annotate(Count("recommendations",
distinct=True)).prefetch_related("workoutmuscles").prefetch_relat
ed("workoutmuscles_muscles").select_related("activity").order_by
("-id")
        return render(request,
"stansport/trainer_workouts.html", {"trainer": trainer,
"workouts": trainer_workouts.values(), "count":
trainer_workouts.count()})

        def std_trainer_advices_view(request, trainer_id):
            trainer = get_object_or_404(Trainer, id = trainer_id)
            trainer_advices = Advice.objects.filter(trainer =
trainer).annotate(Count("recommendations",
distinct=True)).prefetch_related("advicemuscles").prefetch_relate
d("advicemuscles_muscles").select_related("activity").order_by("
-id")
            return render(request,
"stansport/trainer_advices.html", {"trainer": trainer,
"advices": trainer_advices.values(), "count":
trainer_advices.count()})

        def activities_view(request, trainer_id):
            activities =
Activity.objects.all().select_related("hall").order_by("heading"
)
            return render(request, "stansport/activities.html",
{"activities": activities})

        def subscriptions_view(request, trainer_id):
            subscriptions =
Subscription.objects.all().prefetch_related("activities").order_
by("price")
            return render(request, "stansport/subscriptions.html",
{"subscriptions": subscriptions })

        def std_advice_view(request, advice_id):
            advices =
Advice.objects.select_related("trainer").prefetch_related("advic
emuscles").prefetch_related("advicemuscles_muscles").select_rela
ted("activity").annotate(Count("recommendations",
distinct=True))
            advice = get_object_or_404(advices, id = advice_id,
active = True)
            return render(request, "stansport/advice_info.html",
{"advice": advice})

        def std_workout_view(request, workout_id):
            workouts =
Workout.objects.select_related("trainer").prefetch_related("work
outmuscles").prefetch_related("workoutmuscles_muscles").prefetch
_related("workoutplans").prefetch_related("workoutplans__exercis

```

```

e").select_related("activity").annotate(Count("recommendations",
distinct=True))
        workout = get_object_or_404(workouts, id = workout_id)
        return render(request, "stansport/workout_info.html",
{"workout": workout})

    def muscle_workouts_view(request, muscle_id):
        muscle = get_object_or_404(Muscle, id = muscle_id)
        muscle_workouts =
Workout.objects.select_related("trainer").prefetch_related("work
outmuscles").prefetch_related("workoutmuscles_muscles").annotate
(recommendations_count = Count("recommendations",
distinct=True)) .select_related("activity").order_by("-
recommendations_count")
        return render(request,
"stansport/muscle_workouts.html", {"workouts": muscle_workouts})

    def workouts_search_view(request):
        muscles_names = self.request.GET.get('muscles')
        workout_caption = self.request.GET.get('name')
        exercises_names = self.request.GET.get('exercises')
        trainer_name = self.request.GET.get('trainer')
        activity = self.request.GET.get('activity')
        trainer_min = self.request.GET.get('score')
        trainer_max = self.request.GET.get('score_max')
        if muscles_names:
            muscles = muscles_names.partition('-!-')
            muscles_full = []
            for muscle in muscles:
                muscle = Muscle.objects.get(caption__icontains
= muscle)
                if muscle:
                    muscles_full.append(muscle)
        if exercises_names:
            exercises = exercises_names.partition('-!-')
            exercises_full = []
            for exercise in exercises:
                exercise =
Exercise.objects.get(caption__icontains = exercise)
                if exercise:
                    exercises_full.append(exercise)
        workouts =
Workout.objects.all().select_related("trainer").prefetch_related
("workoutmuscles").prefetch_related("workoutmuscles_muscles").an
notate(recommendations_count = Count("recommendations",
distinct=True))
        if workout_caption:
            workouts = workouts.filter(caption__icontains =
workout_caption)
        if trainer_name:
            workouts =
workouts.filter(trainer__user__last_name__icontains =
trainer_name)

```

```

        if activity:
            activity = Activity.objects.get(heading__icontains
= activity)
            workouts = workouts.filter(activity = activity)
        if trainer_min:
            workouts =
workouts.filter(trainer__review_score__gte = trainer_min)
        if trainer_max:
            workouts =
workouts.filter(trainer__review_score__lte = trainer_max)
        if muscles_names:
            workouts =
workouts.filter(workoutmuscles_muscle__in = muscles_full)
        if exercises_names:
            workouts =
workouts.filter(workoutplans_exercise__in = exercises_full)
        workouts = workouts.order_by("-
recommendations_count").values()
        return render(request,
"stansport/search_workouts.html", {"workouts": workouts})

def advices_search_view(request):
    muscles_names = self.request.GET.get('muscles')
    advice_caption = self.request.GET.get('name')
    trainer_name = self.request.GET.get('trainer')
    activity = self.request.GET.get('activity')
    trainer_min = self.request.GET.get('score')
    trainer_max = self.request.GET.get('score_max')
    if muscles_names:
        muscles = muscles_names.partition('-!-')
        muscles_full = []
        for muscle in muscles:
            muscle = Muscle.objects.get(caption__icontains
= muscle)

            if muscle:
                muscles_full.append(muscle)
        advices = Advice.objects.filter(active =
True).select_related("trainer").prefetch_related("advicemuscles"
).prefetch_related("advicemuscles_muscles").annotate(recommendat
ions_count = Count("recommendations", distinct=True))
        if advice_caption:
            advices = advices.filter(caption__icontains =
advice_caption)
        if trainer_name:
            advices =
advices.filter(trainer__user__last_name__icontains =
trainer_name)
        if activity:
            activity = Activity.objects.get(heading__icontains
= activity)
            advices = advices.filter(activity = activity)
        if trainer_min:

```

```

        advices = advices.filter(trainer__review_score__gte
= trainer_min)
        if trainer_max:
            advices = advices.filter(trainer__review_score__lte
= trainer_max)
        if muscles_names:
            advices = advices.filter(workoutmuscles_muscle__in
= muscles_full)
        advices = advices.order_by("-
recommendations_count").values()
        return render(request, "stansport/search_advices.html",
{"advices": advices})
    import json

    def sports_club_view(request):
        with open("../static/stansport/sports_club.json", "r")
as description:
            sports_club_info = json.load(description)
            sports_context = {}
            sports_context["heading"] =
sports_club_info["heading"]
            sports_context["description"] =
sports_club_info["description"]
            sports_context["address"] =
sports_club_info["address"]
            sports_context["phones"] =
sports_club_info["phones"]
            sports_context["instagram"] =
sports_club_info["instagram"]
            sports_context["facebook"] =
sports_club_info["facebook"]
            sports_context["logo"] = sports_club_info["logo"]
            return render(request,
"stansport/sports_club.html", sports_context)

    def hall_view(request, hall_id):
        hall = get_object_or_404(Hall, id = muscle_id)
        return render(request, "stansport/hall.html", {"hall":
hall})

```

ДОДАТОК В
Слайди презентації



Рисунок В.1 – Слайд № 1 (Титульний слайд)

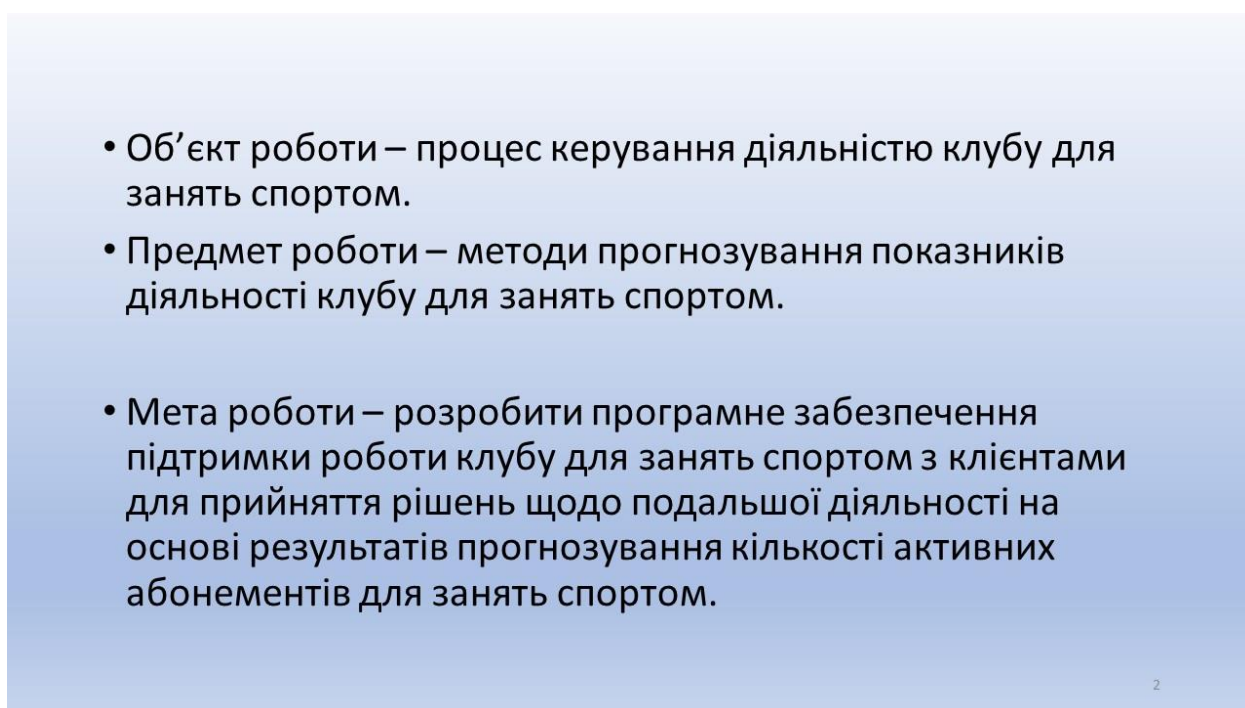


Рисунок В.2 – Слайд № 2 (Об'єкт, предмет, мета дипломної кваліфікаційної роботи)

Завдання роботи

- проаналізувати існуючі програми підтримки роботи з клієнтами клубу для занять спортом;
- розробити метод прогнозування кількості активних абонементів для занять спортом;
- виконати проектування програми підтримки роботи з клієнтами клубу для занять спортом;
- реалізувати вебзастосунок підтримки роботи з клієнтами клубу для занять спортом;
- виконати експериментальне дослідження прогнозування кількості активних абонементів для занять спортом.

3

Рисунок В.3 – Слайд № 3 (Завдання роботи)

Порівняння існуючих програм підтримки роботи з клієнтами клубу для занять спортом

Показник / програма	Sportlife	Livestrong	Palestra	Програма, що розробляється
Інформація про активності клубу	Так	Ні	Так	Так
Інформація про тренерів	Так	Ні	Так	Так
Бронювання занять з тренером	Ні	Ні	Тільки оформлення загальної заявки	Так
Придбання абонементів	Тільки оформлення заявки	Ні	Так	Так
Персоналізація	Ні	Ні	Ні	Так
Поради щодо занять спортом	Ні	Так	Ні	Так
Комплекси вправ	Ні	Так	Ні	Так
Прогнозування кількості абонементів	Ні	Ні	Ні	Так

4

Рисунок В.4 – Слайд № 4 (Порівняння існуючих програм підтримки роботи з клієнтами клубу для занять спортом)

Метод прогнозування кількості активних абонементів для занять спортом

- визначення періоду навчання, відсотку активності за базовим періодом за кожним активним абонентом, узагальнення кількості активних абонементів за кожною групою відвідуваності за кожною спортивною активністю для кожної дати з періоду навчання;
- створення набору моделей, кожна з яких буде використовуватися для роботи з конкретною групою відвідуваності занять та конкретною спортивною активністю;
- формування навчальних вибірок даних для кожної з моделей за допомогою екземплярів з 14 нормованих значень;
- застосування навчання окремо для кожної з моделей на основі відповідної сформованої вибірки даних;
- застосування кожен день всіх навчених моделей для прогнозування кількості активних абонементів.

5

Рисунок В.5 – Слайд № 5 (Метод прогнозування кількості активних абонементів для занять спортом)

Вибір мови програмування для реалізації програмного забезпечення підтримки роботи з клієнтами клубу для занять спортом

Показник / мова програмування	RНР	Python
Галузі застосування	Веброзробка	Машинне навчання, робота з даними, автоматизація та їх застосування у веброзробці
Відлагодження	Порівняно більший час	Менший час
Швидкість роботи результатуючих програм	Вища	Порівняно менша
Організація підтримки результатуючих програм	Порівняно складніша	Простіша

6

Рисунок В.6 – Слайд № 6 (Вибір мови програмування для реалізації програмного забезпечення підтримки роботи з клієнтами клубу для занять спортом)

Діаграма варіантів використання програми неавторизованим відвідувачем та відвідувачем



Рисунок В.7 – Слайд № 7 (Діаграма варіантів використання програми неавторизованим відвідувачем та відвідувачем)

Діаграма варіантів використання програми тренером та менеджером

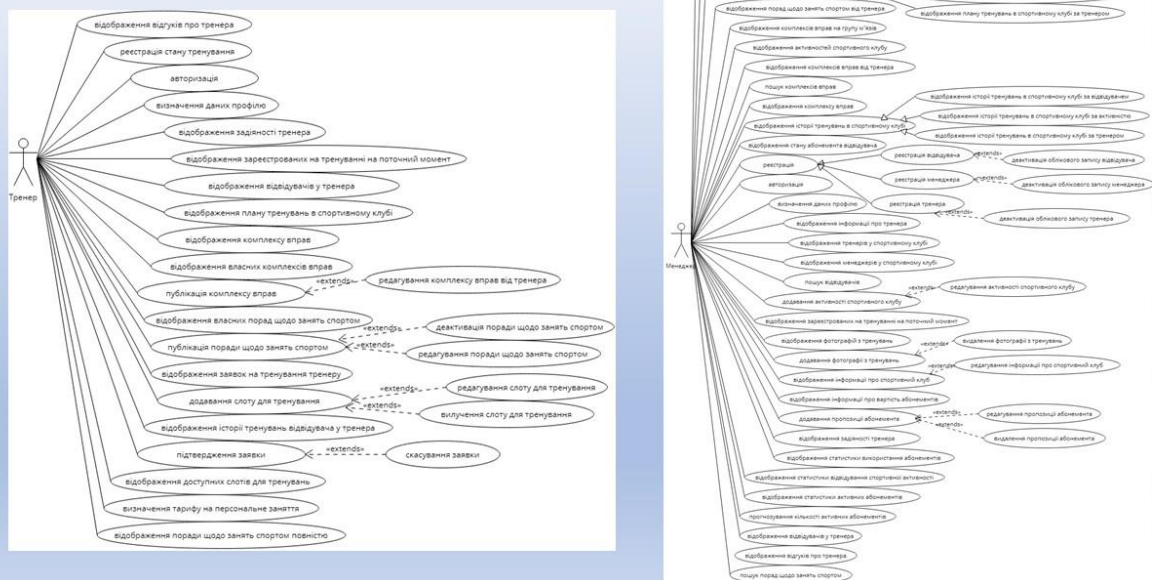


Рисунок В.8 – Слайд № 8 (Діаграма варіантів використання програми тренером та менеджером)

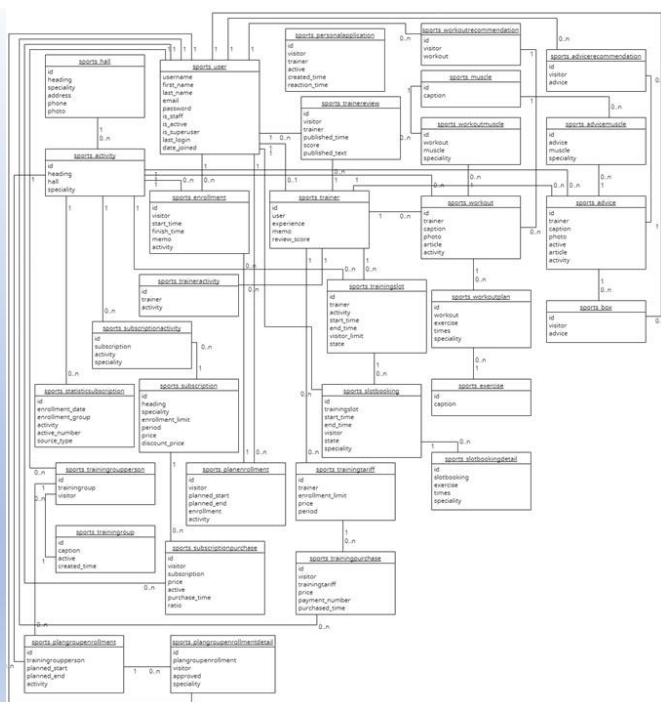


Рисунок В.9 – Слайд № 9 (Схема БД)

Діаграма послідовності для потоку взаємодії між відвідувачем і тренером стосовно проведення індивідуального персонального тренування

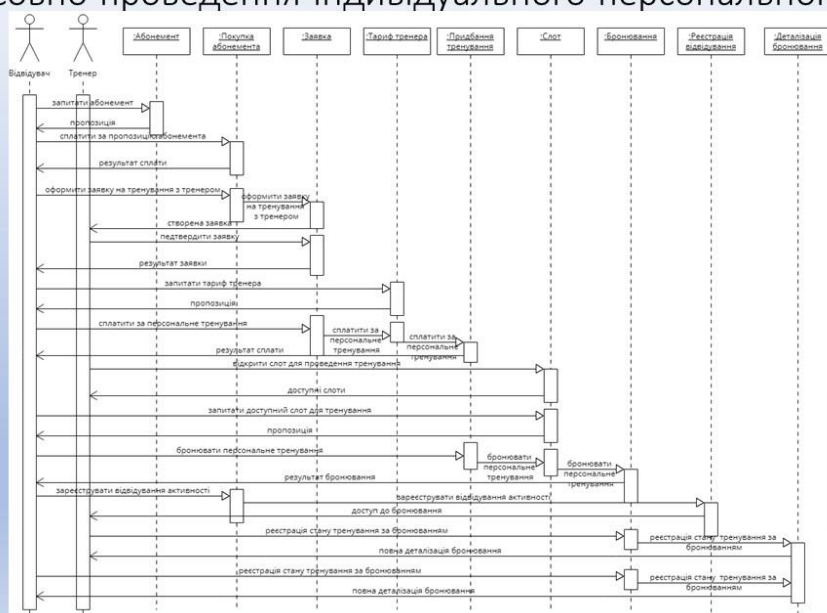
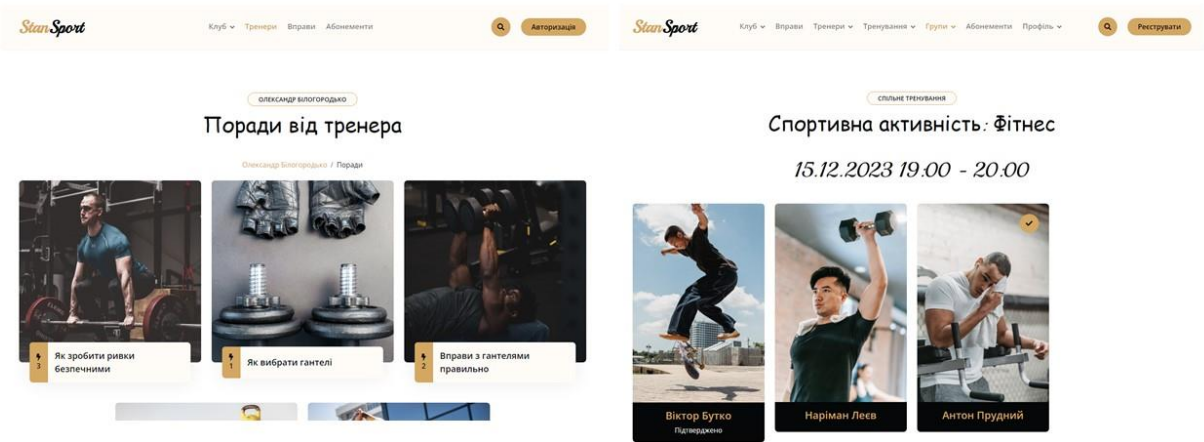


Рисунок В.10 – Слайд № 10 (Діаграма послідовності для потоку взаємодії між відвідувачем і тренером стосовно проведення індивідуального персонального тренування)

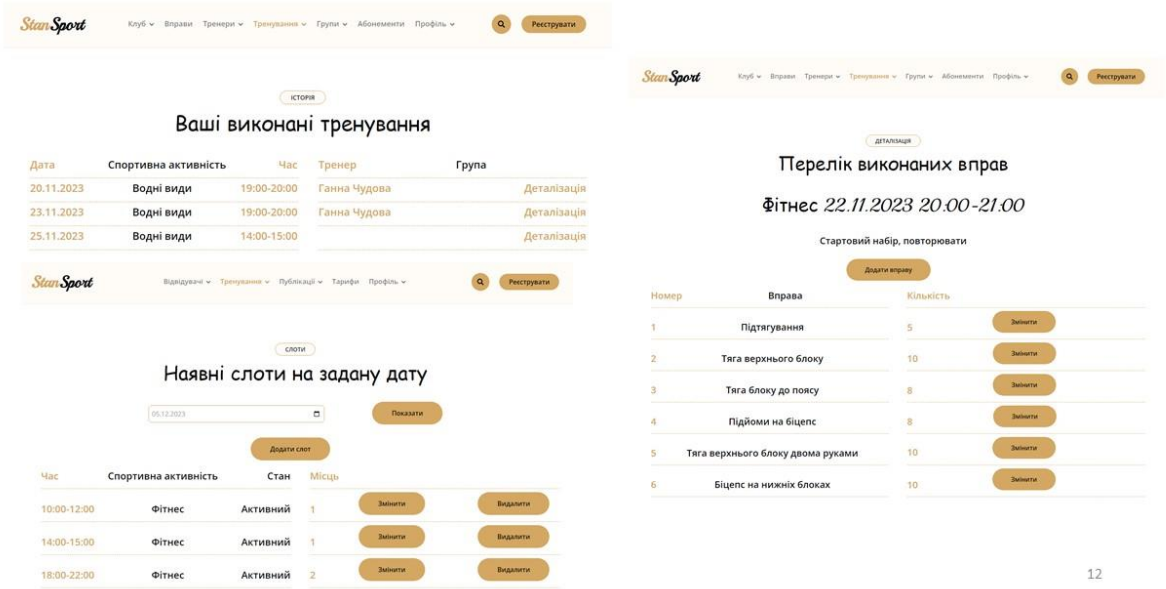
Робота неавторизованого відвідувача та відвідувача з програмою



11

Рисунок В.11 – Слайд № 11 (Робота неавторизованого відвідувача та відвідувача з програмою)

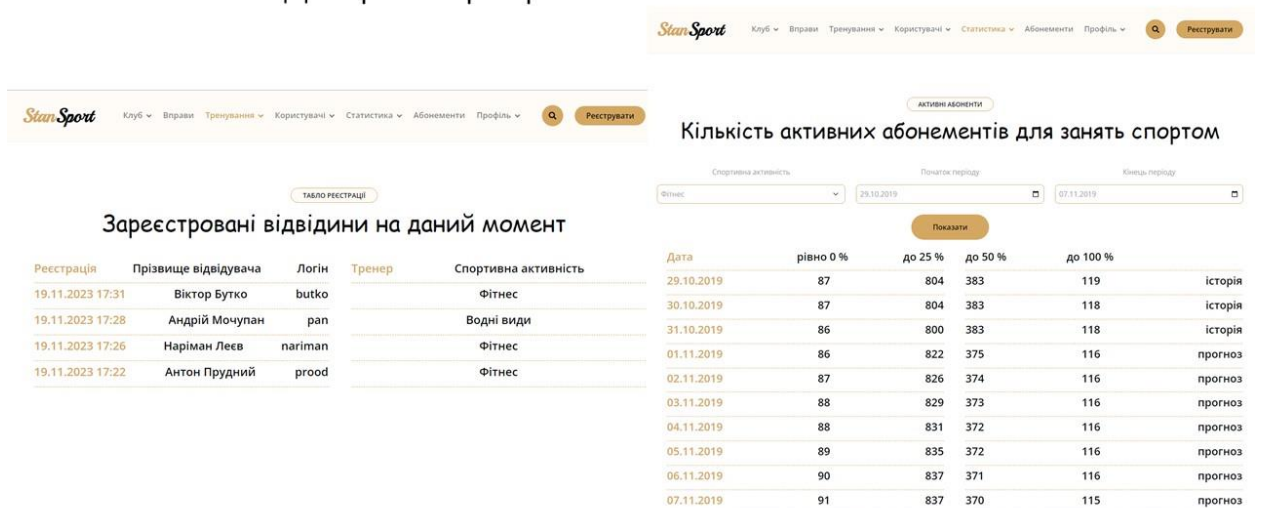
Робота відвідувача та тренера з програмою



12

Рисунок В.12 – Слайд № 12 (Робота відвідувача та тренера з програмою)

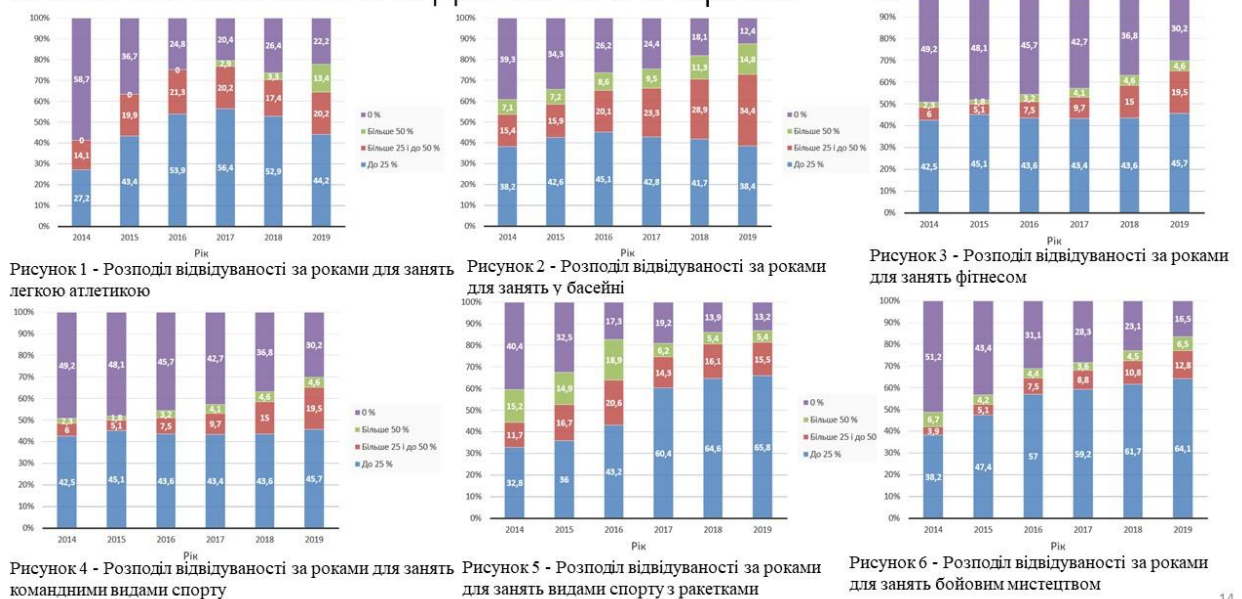
Робота менеджера з програмою



13

Рисунок В.13 – Слайд № 13 (Робота менеджера з програмою)

Експериментальне дослідження прогнозування кількості активних абонементів для занять спортом



14

Рисунок В.14 – Слайд № 14 (Експериментальне дослідження прогнозування кількості активних абонементів для занять спортом)

Експериментальне дослідження прогнозування кількості активних абонементів для занять спортом



Рисунок 7 - Зміна кількості абонементів за місяцями за відвідуваності більше 0 і до 25 % включно



Рисунок 8 - Зміна кількості абонементів за місяцями за відвідуваності більше 25 і до 50 % включно



Рисунок 9 - Зміна кількості абонементів за місяцями за відвідуваності більше 50 %



Рисунок 10 - Зміна кількості абонементів за місяцями за невідвідуваності



Рисунок 11 - Загальна зміна кількості абонементів за місяцями

15

Рисунок В.15 – Слайд № 15 (Експериментальне дослідження прогнозування кількості активних абонементів для занять спортом)

Результати експериментального дослідження прогнозування кількості активних абонементів для занять спортом

Таблиця 1 - Середньоквадратична похибка моделей Random Forest

Вид спортивних активностей / відсоток відвідуваності	(0; 25]	(25; 50]	(50; 100]	0
Легка атлетика	0,00131	0,00076	0,13641	0,00136
Водні види спорту (басейн)	0,00053	0,02092	0,02892	0,00252
Фітнес	0,00106	0,05931	0,01443	0,00079
Командні види спорту	0,00185	0,0149	0,00141	0,001
Види спорту з ракетками	0,00127	0,00049	2,72E-08	8,93E-06
Бойові мистецтва	0,00094	0,007	0,23407	0,00285
Спорт для людей з обмеженими можливостями	0,00062	0,10088	0,00018	0,00265

Таблиця 2 - Середньоквадратична похибка моделей LSTM, створених у відповідності з методом, який запропоновано

Вид спортивних активностей / відсоток відвідуваності	(0; 25]	(25; 50]	(50; 100]	0
Легка атлетика	0,00115	0,00068	0,00392	0,00155
Водні види спорту (басейн)	0,00052	0,00221	0,00295	0,00178
Фітнес	0,00081	0,00078	0,00067	0,00044
Командні види спорту	0,00061	0,00174	0,0011	0,00049
Види спорту з ракетками	0,00003	0,00002	0,00005	0,00003
Бойові мистецтва	0,00086	0,00079	0,00274	0,00096
Спорт для людей з обмеженими можливостями	0,00109	0,00376	0,00009	0,00089

16

Рисунок В.16 – Слайд № 16 (Результати експериментального дослідження прогнозування кількості активних абонементів для занять спортом)

Підсумкові результати експериментального дослідження прогнозування кількості активних абонементів для занять спортом

Вид спортивних активностей / відсоток відвідуваності	Random Forest		LSTM за запропонованим методом	
	Середньо-квадратична похибка	Зважена середньо-квадратична похибка	Середньо-квадратична похибка	Зважена середньо-квадратична похибка
Легка атлетика	0,03496	0,01931	0,00183	0,00152
Водні види спорту (басейн)	0,01322	0,01199	0,00187	0,00162
Фітнес	0,0189	0,01401	0,00067	0,00076
Командні види спорту	0,00479	0,00412	0,00098	0,00082
Види спорту з ракетками	0,00044	9,16E-04	3,1573E-05	2,81715E-05
Бойові мистецтва	0,06122	0,01719	0,00134	0,00099
Спорт для людей з обмеженими можливостями	0,02608	0,04043	0,00146	0,00199
Середнє значення	0,0228	0,01542	0,00117	0,0011

17

Рисунок В.17 – Слайд № 17 (Підсумкові результати експериментального дослідження прогнозування кількості активних абонементів для занять спортом)

Висновки

- розроблено програмне забезпечення підтримки роботи клубу для занять спортом з клієнтами для прийняття рішень щодо подальшої діяльності на основі результатів прогнозування кількості активних абонементів для занять спортом;
- наукова новизна роботи полягає в методі прогнозування кількості активних абонементів для занять спортом, який дозволяє прогнозувати кількість абонементів у майбутні дні за групою відвідуваності та спортивною активністю за кількістю абонементів у минулі дні, застосовуючи для побудови нейронних мереж архітектуру LSTM та застосовуючи правила нормування даних;
- виконано експериментальне дослідження на основі даних, зібраних у спортивному центрі в Лісабоні з 1 червня 2014 року до 31 жовтня 2019 року. Було побудовано моделі на основі використання методу Random Forest і методу, запропонованого в дипломній кваліфікаційній роботі. Результати вказують на ефективність застосування моделей LSTM, створених у відповідності з методом, який запропоновано.

18

Рисунок В.18 – Слайд № 18 (Висновки)