НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»
УДК _004.4_

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2022 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Інтегровані інформаційні системи»
зі спеціальності 126 «Інформаційні системи та технології»
на тему: «Інформаційна система для підтримки діяльності тренажерного залу»

Виконав:
студент VI курсу, групи ІА-12мп
Івлєв Андрій Володимирович _____

Керівник:
Доцент кафедри ІСТ, к.т.н., доц.
Новацький Анатолій Олександрович _____

Рецензент:
Доцент кафедри ІПІ, к.т.н., доц.
Лісовиченко Олег Іванович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2022 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Івлєву Андрію Володимировичу

1. Тема дисертації «Інформаційна система для підтримки діяльності тренажерного залу», науковий керівник дисертації Новацький Анатолій Олександрович, к.т.н., доцент, затверджені наказом по університету від «09» 11 2022 р. № 4115-с

2. Термін подання студентом дисертації 12.12.2022 р.

3. Об'єкт дослідження покращення протікання бізнес-процесів всередині тренажерного залу чи мережі залів та взаємодії з клієнтами

4. Вихідні дані автоматизована інформаційна система для адміністрування та надання послуг спортивним залом.

5. Перелік завдань, які потрібно розробити дослідити предметну область та проаналізувати існуючі рішень, визначити сильні та слабкі сторони кожного, сформувати вимоги до системи, обрати технології для розробки, розробити архітектуру системи, розробити структурну схему системи, розробити базу даних, розробити інтерфейс користувача, розробити керуючу програму, розробити стартап-проект.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу діаграма варіантів використання, загальна структурна схема системи, структурна система серверної частини, інфологічна модель бази даних, ER-діаграма, структурна схема хмарних сервісів, діаграма послідовності процесу.

7. Орієнтовний перелік публікацій

8. Дата видачі завдання 05.09.2022 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Видача завдання	05.09.2022 р.	
2	Аналіз предметної області та формування вимог	26.09.2022 р.	
3	Розроблення діаграми використання	07.10.2022 р.	
4	Вибір технологій	14.10.2022 р.	
5	Розроблення структурної схеми системи	25.10.2022 р.	
6	Розроблення ER-діаграм та бази даних	05.11.2022 р.	
7	Розроблення інтерфейсу користувача	12.11.2022 р.	
8	Розроблення керуючої програми	23.11.2022 р.	
9	Оформлення дисертації	01.12.2022 р.	
10	Подання до захисту	12.12.2022 р.	

Студент Андрій ІВЛЄВ

Науковий керівник Анатолій НОВАЦЬКИЙ

АНОТАЦІЯ

Івлєв А.В. Інформаційна система для підтримки діяльності тренажерного залу. «КПІ ім. Ігоря Сікорського», Київ, 2022

Дисертація складається з 130 сторінок, 32 рисунків, 63 таблиць, 21 літературного джерела та 8 додатків.

Метою даної магістерської дисертації є розробка сучасної інформаційної системи зі здатністю до розширення, яка підвищить ефективність ведення справ спортивних залів, а також забезпечить зв’язок з клієнтами. Вона дозволить скоротити операційні витрати бізнесу, підвищити лояльність клієнтів та залучити нових, а отже збільшити прибутки компанії.

Актуальність розробки зумовлена постійним ростом інтересу до сфери спортивних занять та здорового способу життя та як наслідок зростанням попиту на спортивні послуги, а отже збільшенням кількості спортивних залів. Більшість таких установ на ринку не мають власної інформаційної системи, що означає малий рівень конкурентності та потенційний попит.

Об’єктом дослідження є покращення протікання бізнес-процесів всередині спортивного залу чи мережі та взаємодії з клієнтами.

Предметом дослідження є автоматизована інформаційна система для адміністрування та надання послуг спортивним залом.

В результаті виконання магістерської роботи було розроблено інформаційну систему для підтримки діяльності тренажерного залу. Для цього було детально проаналізовано предметну область, розроблено діаграму варіантів використання, обрано тип та архітектуру рішення, обрано технології для реалізації, розроблено структурну схему, схему бази даних, інтерфейс програми та реалізовано систему. Також було досліджено можливість впровадження даного продукту у вигляді стартап-проекту, що визначило її як цілком можливу.

Ключові слова: спортивний зал, веб-застосунок, адміністрування, сервер, Blazor, .NET, клієнт, покращення.

SUMMARY

Ivliev A.V. Information system to support gym activities. "Igor Sikorsky KPI", Kyiv, 2022

The Master`s thesis consists of 130 pages, 32 figures, 63 tables, 21 literary sources and 8 appendixes.

The goal of this master's thesis is to develop a modern information system with the ability to expand, which will increase the efficiency of running the affairs of sports halls, as well as provide communication with customers. It will reduce business operating costs, increase customer loyalty and attract new ones, and thus increase the company's profits.

The relevance of the development is due to the constant growth of interest in the field of sports activities and a healthy lifestyle and, as a result, the growth in demand for sports services, and therefore the increase in the number of sports halls. Most of such institutions on the market do not have their own information system, which means a low level of competition and potential demand.

The object of the study is to improve the flow of business processes inside the gym or network and interaction with customers.

The subject of research is an automated information system for the administration and provision of services in a sports hall.

As a result of the completion of the master's work, an information system was developed to support the activity of the gym. For this, the subject area was analyzed in detail, a diagram of use cases was developed, the type and architecture of the solution was chosen, technologies were chosen for implementation, a structure diagram, a database diagram, a program interface were developed, and the system was implemented. The possibility of implementing this product in the form of a startup project was also investigated, which determined it to be quite possible.

Keywords: gym, web-application, administration, server, Blazor, .NET, client, improvement.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

CRM – Customer Relationship Management або управління відносинами з клієнтами.

UML – Unified Modeling Language або уніфікована мова моделювання.

AWS (англ. Amazon Web Services) – веб сервіси від компанії Amazon.

GCP (англ. Google Cloud Platform) – хмарне середовище від компанії Google.

FAQ (англ. Frequently Asked Question(s)) – найбільш поширені запитання.

IOS (англ. iPhone Operating System) – операційна система смартфонів від компанії Apple.

ОС – операційна система.

БД – база даних.

MPA (англ. Multi Page Applications) – багатосторінкові застосунки.

SPA (англ. Single Page Applications) – односторінкові застосунки.

HTML (англ. Hyper Text Markup Language) – мова гіпертекстової розмітки.

CSS (англ. Cascading Style Sheets) – каскадні таблиці стилів.

CLR (англ. Common Language Runtime) – загальномовне виконуюче середовище.

CLI (англ. Common Line Interface) – загальномовна інфраструктура.

API (англ. Application Programming Interface) – програмований інтерфейс програми.

UI (англ. User Interface) – інтерфейс користувача.

DOM (англ. Document Object Model) – об’єктна модель документу.

JSON (англ. JavaScript Object Notation) – формат обміну даними.

DI (англ. Dependency Injection) – впровадження залежностей.

СУБД – система управління базою даних.

GUI (англ. Graphical User Interface) – графічний інтерфейс користувача.

SQL (англ. Structured Query Language) – мова структурованих запитів до бази даних.

ACID (англ. Atomicity Consistency Isolation Durability) – головні вимоги до системи трансакцій: атомарність, узгодженість, ізольованість, стійкість.

MVCC (англ. Multiversion concurrency control) – механізм СУБД для забезпечення паралельного доступу до БД.

ORM (англ. Object-Relational Mapping) – технологія, що пов’язує концепції даних з базою даних.

DAL (англ. Data Access Layer) – шар доступу до даних.

LINQ (англ.Language-Integrated Query) – .NET мова запитів до сховища даних.

CRUD (англ. Create Read Update Delete) – набір базових функцій для роботи з таблицею бази даних: створення, читання, оновлення, видалення.

IDE (англ. Integrated Development Environmet) – інтегроване середовище розробки.

ER-діаграма (англ. Entity Relationship) – діаграма сутностей та їх зв’язків.

UX (англ. User Experience) – зручність використання.

KISS (англ.Keep It Stupid Simple) – один з принципів написання коду.

DRY (англ. Don’t Repeat Yourself) – один з принципів написання коду.

URL (англ. Uniform Resource Locator) – загально-визначена веб-адреса.

ВСТУП

Двадцять перше століття вже називають віком інформації, що характеризує вектор його подальшого зростання ще на початку. Сучасний світ – світ активного розвитку, та цифрових технологій. Останні ж надають безмежні можливості для оптимізації різного роду завдань. Тому не дивно, що щороку зростає кількість різноманітних сервісів, що покликані вирішувати певні задачі, чи спрощувати або покращувати певні існуючі процеси.

З появою і розвитком Інтернету та розробки програмного забезпечення, надавати різноманітні послуги клієнтам стало дедалі простіше та ефективніше, а самі послуги стали більш якісними. Сьогодні, щоб залишатися конкурентоспроможним, бізнесу варто мати інформаційну систему для взаємодії з користувачами та для контролю роботи керівного персоналу.

У нинішній час, коли людина здебільшого веде сидячий та малорухливий спосіб життя, фізичні навантаження допомагають впоратися з його негативними наслідками. Також не варто забувати, що це позитивно впливає на психологічний та емоційний стан людини, що особливо актуально в період війни.

Тому завданням даної роботи є розробка автоматизованої інформаційної системи для покращення якості послуг, що надаються спортивним залом. З її допомогою клієнт матиме змогу зареєструватися без участі адміністратора та увійти в систему, переглянути власні дані, історію відвідувань, інформацію про стан абонементу, переглянути базу знань, в якій буде міститися корисна інформація щодо процесу тренувань. Адміністратор в свою чергу буде мати можливість зареєструвати користувача, відмічати присутність клієнтів, призначати тренерів, слідкувати за станом оплати абонементу, переглядати зведену статистику продажів. Тренер зможе переглядати та вводити інформацію щодо технічного обслуговування тренажерів, розробляти та призначати програми тренувань клієнтам. Тобто система буде використовуватись для:

- введення обліку;
- контролю роботи керуючого персоналу;

- швидкого та зручного отримання аналітичних даних;
- отримання актуальної інформації про клієнтів;
- надання інформаційних послуг клієнту, без необхідності звертатися до адміністратора чи тренера.

Галуззю використання даного рішення є заклади з надання спортивних послуг. Воно допоможе оптимізувати важливі процеси у спортивних залах, де ще не введено жодних інформаційних систем, та де для більшої частини описаних операцій необхідне втручання людини.

Дана розробка вирішить наступні бізнес-задачі:

- значно зменшить навантаження на керуючий персонал;
- зменшить операційні витрати, за рахунок скорочення штату працівників(адже для адміністрування достатньо буде однієї людини), та зменшення витрат на матеріали(папір, ручки);
- покращить досвід користування послугами клієнтами, що збільшить їх лояльність;
- створить єдину цілодобову інформаційну базу, що покращить швидкодію надання послуг.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

Сьогодні дедалі більше людей задумуються над станом свого фізичного здоров’я, адже він напряму впливає на психологічний стан. Теперішні виклики потребують неабиякої емоційної стійкості та міцного здоров’я.

Одним з можливих рішень є відвідування спортивного залу. Але, прийшовши туди, необхідно спочатку витратити чимало часу на те, аби тебе зареєстрували в системі, надали інформацію щодо вартості різних абонементів, записали на інструктаж, допомогли підібрати тренера, а він у свою чергу проконсультував щодо вартості занять та типу тренувань, після чого підібрав таку, що підходитиме саме тобі. Для адміністратора залу, в якому немає подібної системи, також витрачається час на те, аби перевірити статус абонементу відвідувача, подовжити його чи анулювати та занести дані у таблицю та багато інших дрібних операцій.

Саме для того, щоб уникнути зайвих витрат часу клієнта, адміністратора та тренера, було створено автоматизовану інформаційну систему з підтримки діяльності тренажерного залу.

1.1 Об’єкт та предмет дослідження

Існує багато різних спортивних споруд, що надають послуги для клієнтів. Їх можна поділити на:

- спеціальні;
- комплексні.

Перші є спеціалізованими установами для одного конкретного виду спорту (басейни, велотреки, поля, тощо). Другі ж містять в собі всі необхідні елементи для занять декількома різними видами спорту (водні комплекси, комплексні майданчики, тощо).

Тренажерні зали ж у свою чергу можуть відноситись як до першого типу, в якому наявний інвентар лише для занять важкою атлетикою (гирі, штанги, гантелі),

так і до другого, з наявними біговими доріжками, басейнами, боксерськими грушами, тощо. Також, за розповсюдженістю, їх можна умовно поділити на дві групи, а саме:

- мережеві(до складу яких входить декілька спортивних залів віддалених один від одного);
- одиничні;

Першому типу характерно знаходитись у великих містах, та мати бодай якісь інформаційні системи, що їх обслуговують, але вони здебільшого орієнтовані на адміністрування і майже не допомагають клієнту та тренеру.

Другі ж є більш розповсюдженими в нашій країні, та знаходяться у містах та селах різних розмірів, де достатньо часто контроль відвідуваності та інші подібні операції фіксуються лише на папері, або стандартними програмами на комп'ютері.

Отже предметом дослідження даної роботи є розробка такої інформаційної системи, яка підходитиме як для мережевого, так і для одиничного типу тренажерного залу, а також для спеціальних та комплексних спортивних установ, і матиме змогу покращити вже існуючі процеси в них та додати нових.

Для цього необхідно провести аналіз вже існуючих рішень та сформувати вимоги до системи.

1.2 Огляд існуючих рішень

Аналіз відомих рішень допоможе більш точно сформувати вимоги до системи, та врахувати всі важливі переваги та недоліки кожного з них. Під час пошуку подібних існуючих розробок, було знайдено та розглянуто декілька варіантів систем, що в тій чи іншій мірі реалізовують необхідний функціонал.

1.2.1 Програмний комплекс «GraFit»

GraFit – мережа спортивних фітнес-клубів, що розташовані в Києві, та ближніх селах. Дана мережа має для своїх клієнтів веб-сайт та мобільний додаток.

З наявних функцій веб-сайту (рис. 1.1), можна сказати, що він був створений в ознайомчих цілях, та носить суто інформативний характер.

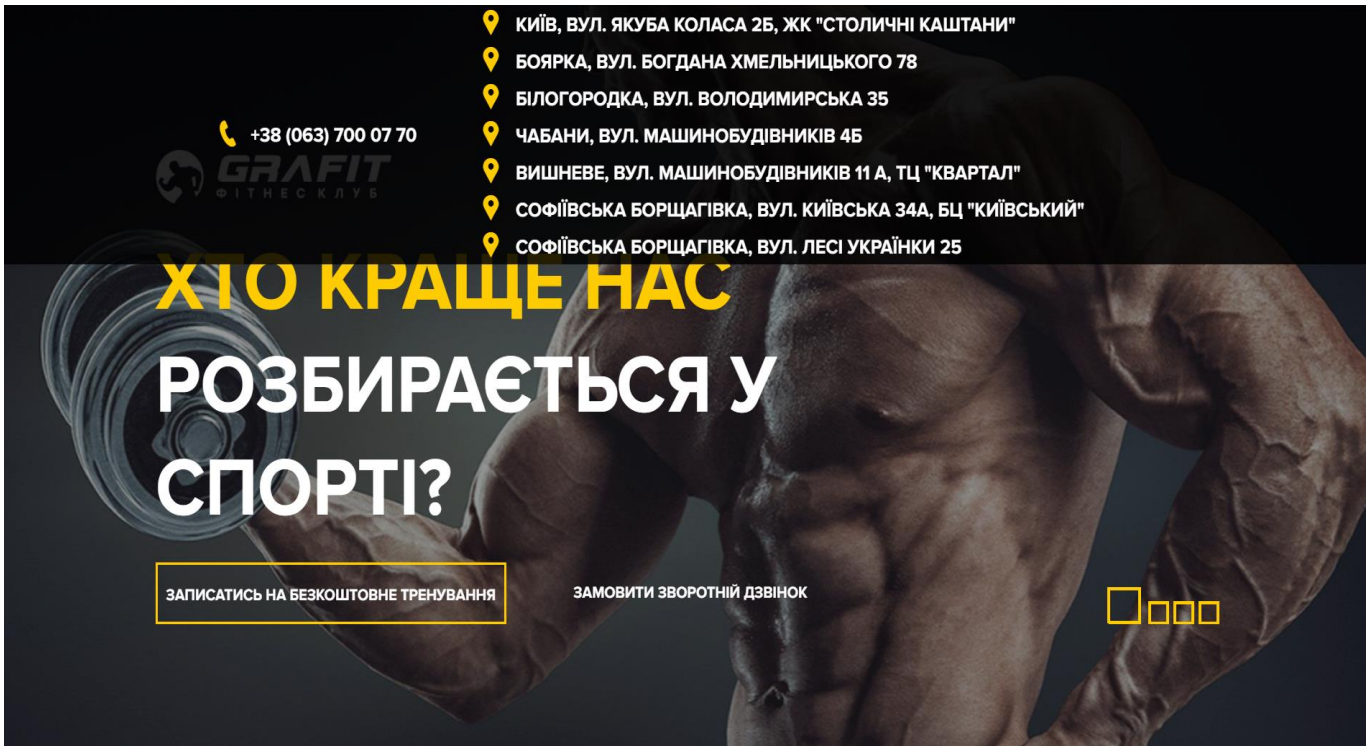

Рисунок 1.1 – Сторінка веб-сайту GraFit [1]

З доступних функцій:

- замовлення зворотного дзвінку;
- запис на безкоштовне перше тренування, що також є форматом замовлення зворотного зв’язку.

Даний сайт має сучасний та стильний дизайн, але замало функціоналу, що зовсім не відповідає необхідним вимогам, тому перейдемо до аналізу мобільного додатку.

Мобільний додаток «GraFit», екран якого зображений на рисунку 1.2, розроблений компанією «ListOk.biz», призначений для користування лише клієнтами, та має такі доступні опції:

- авторизація та реєстрування у системі;
- перегляд всіх залів;
- запис на тренування або його відміна;

- перегляд розкладу;
- перегляд новин;
- перегляд історії платежів та відвідувань.

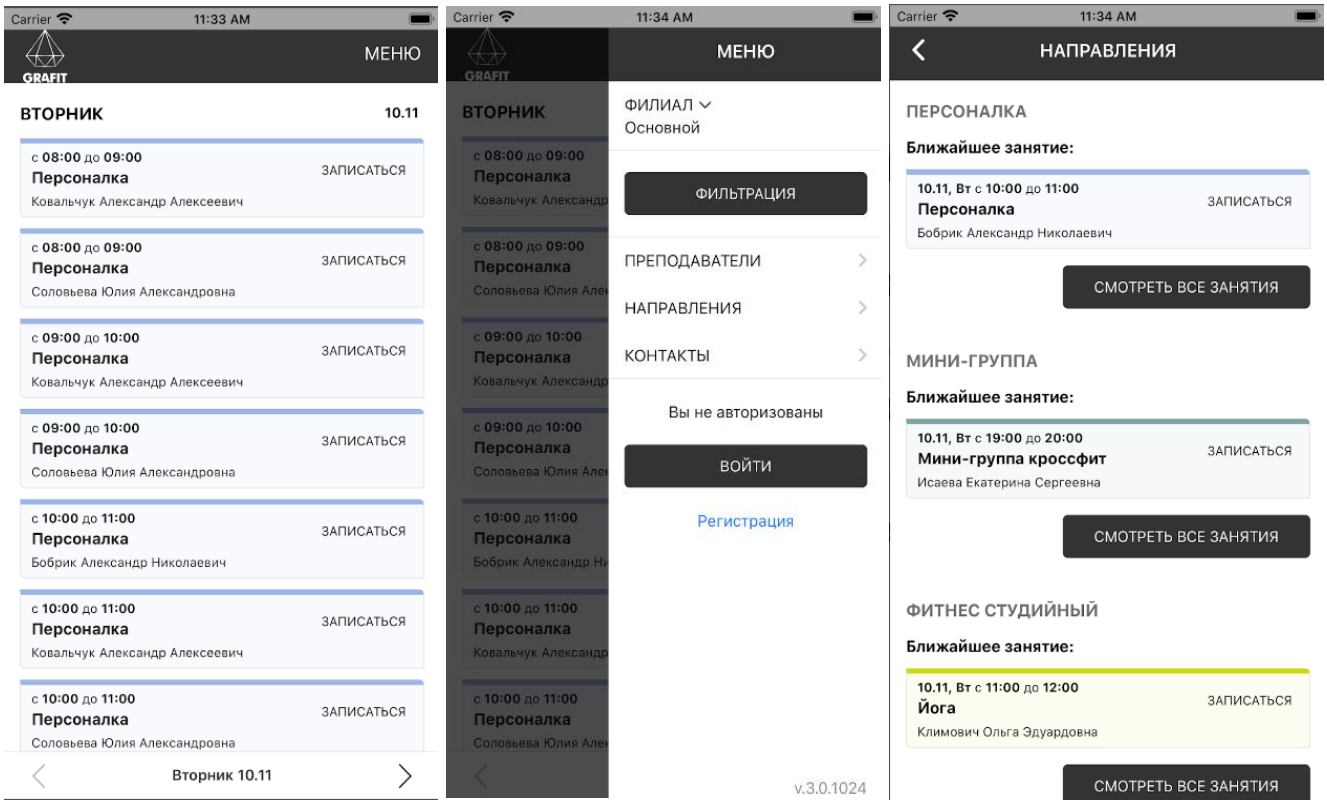

Рисунок 1.2 – Мобільний додаток «GraFit» [2]

Підсумовуючи, можна виділити наступні плюси даного програмного комплексу:

- стильний дизайн(але не зовсім зручний у використанні);
- наявність базового функціоналу, необхідного клієнтові;
- наявність мобільного додатку під платформи: Android, IOS

Серед недоліків можна виділити:

- відсутність ролей адміністратора та тренера;
- відсутність інформаційної бази з вправами для клієнтів;
- відсутність можливості переглянути стан свого абонементу;
- відсутність можливості переглянути програму персонального тренування.

1.2.2 Програмний комплекс «Sport Life»

Sport Life – мережа спортивних клубів, що розташовані в різних містах України. Варто зазначити, що дана мереже є одним з лідерів на ринку спортивних послуг в Україні. Вона, як і попередній програмний комплекс, має свої веб-сайт(рис. 1.3) та додаток для клієнтів.

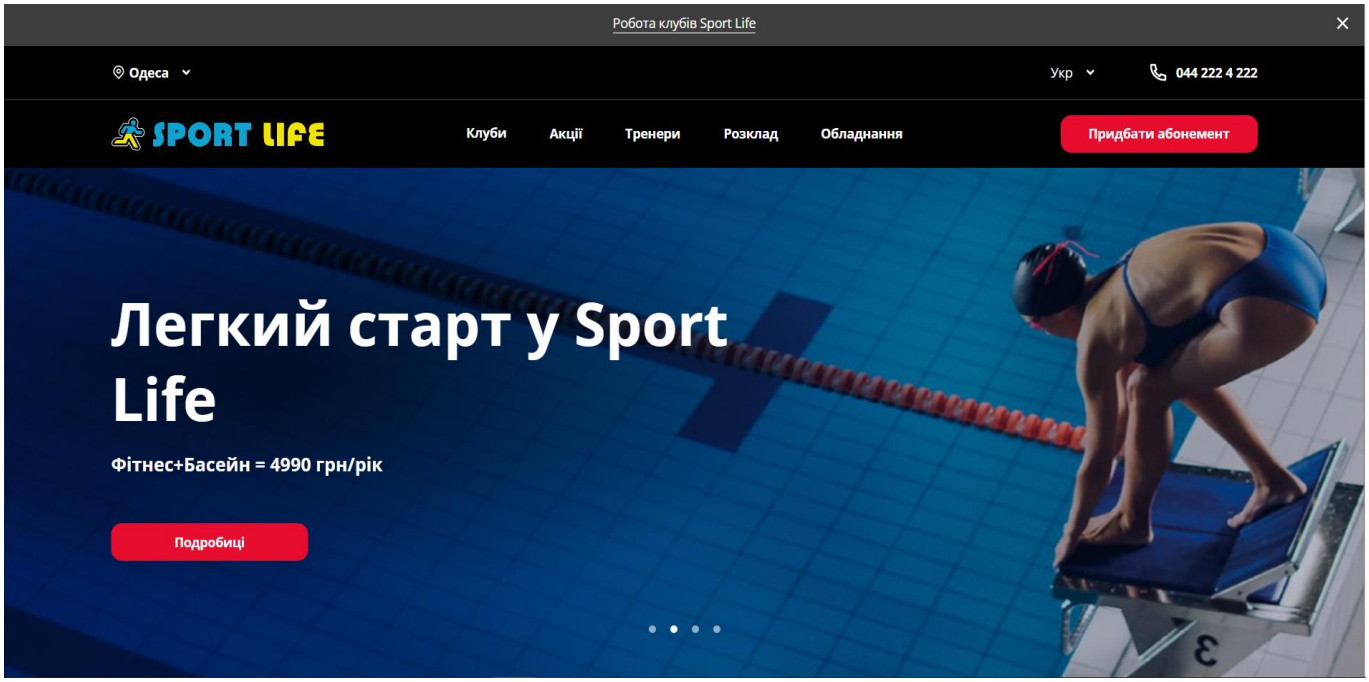

Рисунок 1.3 – Веб-сайт програмного комплексу SportLife [3]

Даний веб-сайт був створений для клієнтів, в ознайомчих цілях, та надає наступні функціональні можливості:

- FAQ – питання та відповіді;
- можливість переглянути адреси клубів;
- ознайомитись з акційними пропозиціями;
- можливість ознайомитись з тренерським складом кожного клубу;
- переглянути розклад роботи залу;
- переглянути виробників тренажерів та обладнання, що використовуються в мережі.

Функціоналу залу повністю достатньо як для відвідувача так і для нового клієнта.

Мобільний додаток «Sport Life Fitness», що зображено на рисунку 1.4, розробений українською компанією «SPORT LIFE INTERNATIONAL», та призначений для використання відвідувачами залу.

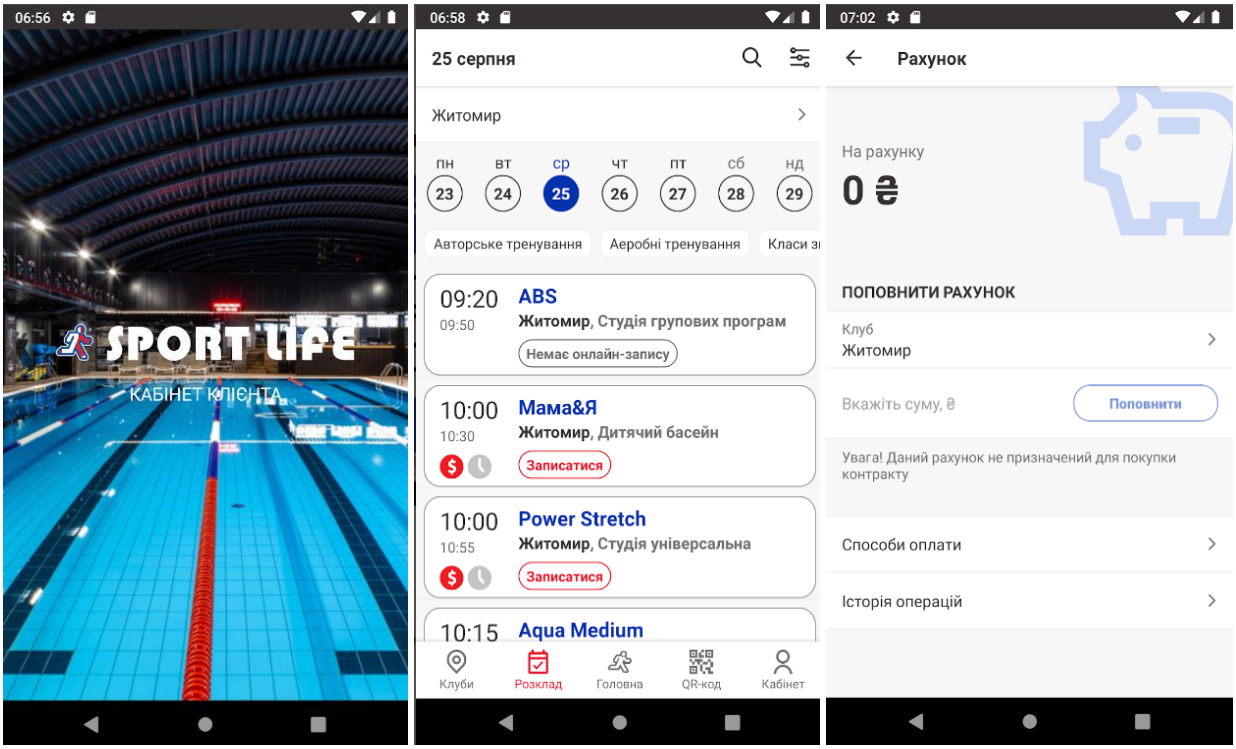

Рисунок 1.4 – Мобільний додаток «Sport Life Fitness» [4]

Серед доступних функцій:

- перегляд розкладу тренувань;
- запис на тренування;
- оплата абонементу;
- перегляд особистої інформації;
- перегляд переліку клубів;
- інформація про поточні акції та пропозиції;
- покупка додаткових послуг.

В якості CRM-системи та для користування адміністраторами та бухгалтерами, дана мережа спортивних клубів використовує послуги програмного комплексу, що розглянуто в підпункті 1.2.3 даного підрозділу.

Провівши аналіз програмного комплексу «Sport Life», можна виділити наступні переваги:

- сучасний дизайн сайту та додатку;
- досить широкий функціонал для клієнтів;
- мобільний додаток для клієнтів під платформи Android, IOS;

До недоліків можна віднести:

- відсутність функціоналу та мобільного додатку для адміністраторів та тренерів;
- відсутність баз знань з вправами для клієнтів;
- відсутність функцій, що доступні на веб-сайті, у мобільному додатку.

Серед існуючих рішень також було оглянуто CRM-системи, що стрімко набувають популярності у наш час. CRM-система – це така, що використовує різні стратегії, методи, інструменти та технології для розвитку бізнесу, утримання та залучення нових клієнтів. Основною метою її впровадження є створення єдиної екосистеми по роботі з клієнтами. Саме через частковий збіг цілей було розглянуто наступний програмний продукт.

1.2.3 Програмний комплекс «LuckyFit»

LuckyFit – CRM-система для спортивних клубів, що розроблена компанією «LuckySoft OU», яка зареєстрована в Естонії. Вона є чудовим інструментом для обліку та взаємодії з клієнтами, а також для контролю бізнес-процесів всередині клубу. Даний продукт має власну веб-сторінку (рис. 1.5) для того, аби потенційні клієнти (спортивні клуби чи їх мережі), могли ознайомитись з переліком послуг, що надаються та з їх ціною.

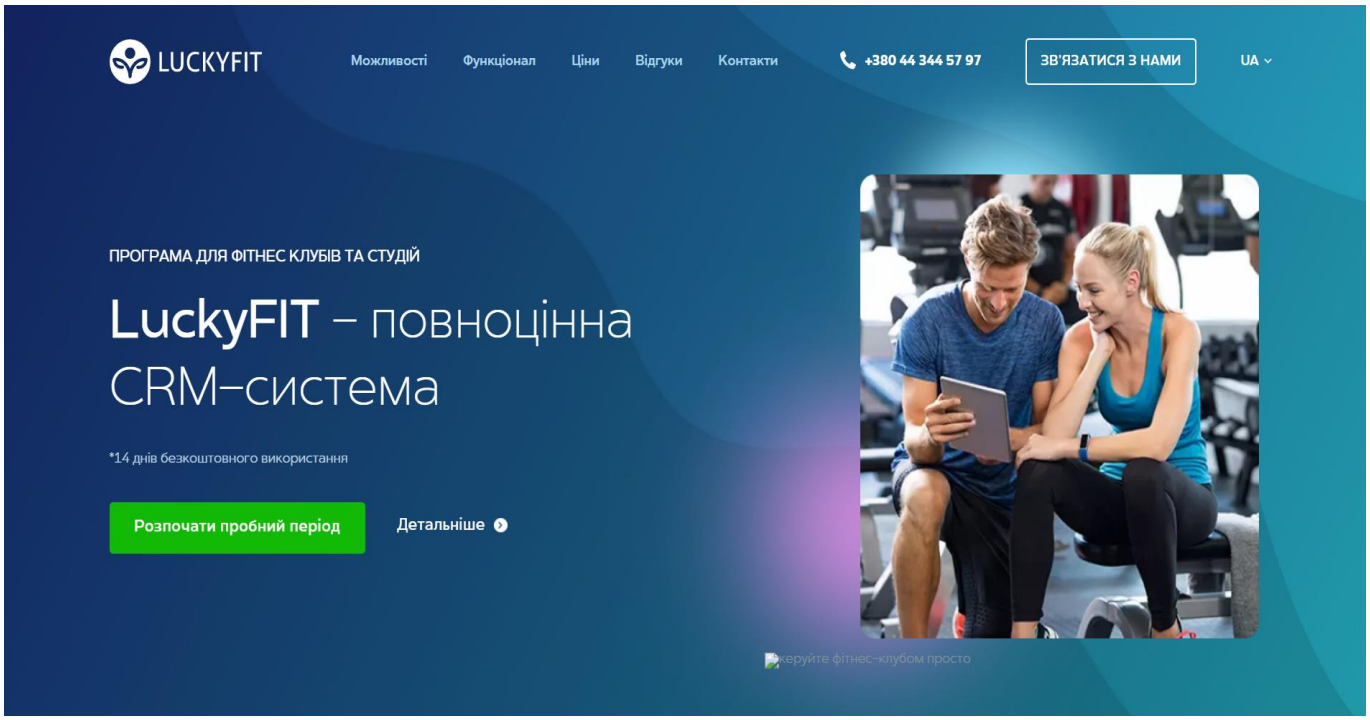

Рисунок 1.5 – Веб-сайт програмного комплексу LuckyFit [5]

Серед іншого наявний такий функціонал:

- абонементна система та налаштування;
- інтеграція з різними турнікетами та POS терміналами;
- власне CRM-система для збільшення кількості продажів;
- облік товарів та послуг;
- розклад, як для клієнтів, так і для адміністраторів;
- реєстрація відвідувань;
- база даних клієнтів;
- мобільний додаток для клієнтів (Android, IOS).

Даний функціонал покриває більшість необхідних процедур та запитів для різного роду спортивних залів. Зручність використання – було пріоритетом у розробників, тому було створено мобільні додатки для клієнтів та для персоналу. З його допомогою клієнт може:

- придбати абонемент;
- записатись на тренування;
- переглянути особисту інформацію (історію платежів, відвідувань, тощо);

- отримувати сповіщення;
- оцінювати тренерів та клуб.

Тренери ж можуть:

- керувати своїми змінами;
- переглядати календар персональних та групових тренувань;
- переглядати інформацію щодо розрахунку заробітної плати.

Бухгалтера:

- вести фінансовий облік у системі;
- контролювати всю документацію товарів.

Адміністратори:

- отримувати нагадування;
- переглядати поточні задачі та історію взаємодії з клієнтами.

Власники бізнесу матимуть простий доступ до системи з будь-якої точки земної кулі, щоб контролювати процеси та переглядати всю аналітику клубу.

До основних переваг даної системи можна віднести:

- простоту використання;
- наявність досить широкого функціоналу;
- наявність адміністративних ролей (адміністратор, бухгалтер, тренер, власник)
- наявність мобільного додатку;

Основним недоліком даного програмного комплексу є складність його підтримки, адже специфіка його налаштування дозволяє здійснювати дану процедуру здебільшого розробникам системи. Тому, купуючи дану систему, компанія вимушена співпрацювати з розробниками впродовж усього часу, задля підтримки її у потрібному стані. Також недоліком є чималі затрати часу на те аби розробити та впровадити рішення. В основному причиною цього виступає те, що зазвичай подібні системи впроваджують компанії, які вже мають якесь початкове рішення. В такому випадку, після оформлення замовлення починається тісне співробітництво компанії-розробника з компанією-клієнтом, що узгоджують проблеми, які можуть бути покращені. Після всіх необхідних узгоджень компанія-розробник повинна зробити

деяку переробки в архітектурі, аби підігнати свій базовий продукт під нові вимоги. Даний процес є недоліком через те, що, як правило, перероблювати існуючу систему набагато довше та важче, аніж написати код з нуля.

Ще одним недоліком є те, що система є важкою для розширення, що вимагає затрат часу на розробку. Також присутні недоліки в мобільному додатку, який є не пристосований до конкретного клієнта своїм дизайном, та має обмеження в функціоналі, так як розробка додаткового приведе до збільшення витрат на дану систему. Таким чином даний продукт є не достатньо гнучким, аби задовольнити різні потреби клієнтів. Також основним недоліком є те, що взаємодія клієнта з системою прив’язана до пристрою, яким він користується та на якому встановлений даний програмний комплекс.

До недоліків також варто віднести вартість послуг даної системи, адже вона є достатньо вагомою, та явно некомфортною для невеликих клубів. Даний сервіс використовується як підписка, за яку потрібно постійно платити, аби продовжити термін використання. Дані про тарифи наведені у таблиці 1.1.

Таблиця 1.1 – Порівняння тарифних планів «LuckyFit»

Вартість, євро за місяць	45	95	195	495
Кількість клієнтів	До 200	До 1000	До 2500	Не обмежено
Наявність мобільного додатку для клієнтів	Ні	Так	Так	Так
Наявність мобільного додатку для персоналу	Ні	Так	Так	Так
Наявність CRM-системи	Ні	Так	Так	Так

Виходячи з даних цієї таблиці можна побачити, що цільовою групою клієнтів даного сервісу є великі мережі спортивних клубів, які маю необмежену кількість клієнтів та здатні платити близько 500 євро за місяць використання сервісу. Найдешевший пакет послуг здається взагалі нікому не потрібною опцією, адже надто мала кількість клієнтів навіть для невеликого спортивного залу, та відсутні основні функціональні можливості даного програмного комплексу.

Тож в даному розділі, було розглянуто актуальність розробки, та визначено об’єкт та предмет дослідження. Після цього було проведено аналіз існуючих рішень, в ході якого було виділено основних аналогів та конкурентів, розглянуто їхні системи, проаналізовано функціональні можливості даних систем та їх складові частини, з яких вони складаються. Наступим кроком було зважження всіх переваг та недоліків кожної.

Результатами виконання даного розділу є чітко зазначені об’єкт та предмет дослідження, а також уявлення про стан ринку. Дана інформація показує, що в сфері застосувань існують прогалини, які зможе заповнити розроблювана система.

Наступним кроком є формування вимог до системи, що розробляється.

2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

Вимога – поняття, що описує характеристику, якій повинний відповідати об’єкт, до якого ця вимога ставиться. Складання списку вимог до системи є чи не найважливішим етапом її розробки. Завдяки цьому спрощується процес контролю розробки, адже всі необхідні вимоги чітко прописані та визначені. Також від успіху на даному етапі залежить наскільки розроблена система відповідатиме очікуванням сторін, що отримають вигоду від даної розробки.

Першим кроком цього етапу є визначення зацікавлених сторін. Проведений аналіз відомих рішень дав змогу однозначно визначити, що такими сторонами є спортивний клуб або їх мережа та відвідувач.

Джерелами вимог виступають три типи користувачів даної системи:

- відвідувач – клієнт спортивного клубу;
- тренер – працівник спортивного клубу;
- адміністратор – працівник спортивного клубу.
- власник тренажерного залу;

Вимоги до системи бувають наступних двох типів:

- функціональні;
- нефункціональні.

Наступним кроком розробки є визначення переліків таких вимог.

2.1 Функціональні вимоги

Функціональні вимоги дають формулювання того, як повинна себе поводити система. Вони визначають той функціонал системи, який вона повинна мати, щоб задовольнити очікування та потреби зацікавлених сторін. Іншими словами це функції, якими може скористатися користувач. Користувачем системи може бути будь-хто з усіх зацікавлених сторін.

Після ретельного аналізу існуючих систем було визначено, що система повинна:

- авторизувати користувачів;
- реєструвати користувачів;
- надавати актуальну інформацію відвідувачу про стан його абонементу;
- надавати інформацію відвідувачу щодо тренерів;
- надавати інформацію відвідувачу стосовно існуючих вправ (база знань з вправ);
- надавати можливість переглянути плани своїх тренувань для відвідувачів;
- надавати можливість переглянути історію відвідувань та поповнень абонементу відвідувачу;
- надавати актуальний графік роботи тренажерного залу відвідувачу;
- надавати можливість обрати зал для відображення інформації(у випадку мережі);
- надавати можливість запису на тренування відвідувачу;
- надавати тренеру можливість створювати та призначати тренування відвідувачу;
- надавати можливість тренеру додати або відмовитись від клієнта;
- надавати можливість тренеру переглянути розклад своїх тренувань;
- надавати можливість тренеру переглядати список своїх клієнтів;
- надавати можливість тренеру переглянути розмір своєї заробітної плати;
- надавати можливість тренеру переглядати/додавати/видаляти вправи з бази знань;
- надавати тренеру можливість переглянути список тренажерів;
- надавати можливість тренеру переглядати графік обслуговування тренажерів та відмічати такі події.
- надавати адміністратору зведену таблицю з інформацією про тренерів з наступними даними: розмір заробітної плати, кількість підлеглих відвідувачів, вартість тренування, кількість тренувань за місяць;
- надавати адміністратору можливість видаляти або додавати нових відвідувачів чи тренерів у систему;

- надавати адміністратору можливість встановлювати/змінювати заробітну плату тренера;
- надавати адміністратору змогу змінювати/встановлювати вартість заняття для тренера;
- надавати адміністратору змогу змінювати розклад роботи залу;
- надавати адміністратору змогу змінювати статус абонемента клієнта;
- надавати адміністратору змогу переглядати зведену інформацію про відвідувачів тренажерного залу;
- надавати адміністратору можливість реєструвати відвідування клієнта в залі;
- надавати можливість адміністратору створювати «магазин» для залу, в якому буде зберігатись перелік товару, що наявний для продажу;
- надавати адміністратору можливість додавати/видаляти/вносити зміни в даний облік;
- надавати можливість адміністратору переглядати історію обслуговування тренажерів;
- надавати можливість власнику додавати/видаляти адміністраторів та тренерів;
- надавати можливість власнику додавати/видаляти зали (у випадку мережі);
- надавати можливість змінювати та переглядати заробітну плату адміністраторів та тренерів;
- надавати можливість власнику отримувати зведену статистику по конкретному спортивному клубу.

Наступним кроком є створення переліку другого типу вимог.

2.2 Нефункціональні вимоги

Нефункціональні вимоги дають характеристику системі, що розроблюється, а також пояснюють обмеження системи. Тобто вони визначають якою система має бути. Існує класифікація даних вимог на різні категорії, як от [6]:

- інтерфейс користувача;
- надійність;
- безпека;
- продуктивність;
- стандарти;

Отже, враховуючи переваги та недоліки розглянутих систем, було визначено наступний перелік:

- система повинна забезпечувати приватність даних при реєстрації та будь-яких інших операціях з їх пересиланням;
- система повинна забезпечувати можливість створювати до 5000 клієнтів в один спортивний зал;
- система повинна надавати змогу додавати до 1000 одиниць товару;
- система повинна надавати змогу відновити аккаунт користувача;
- система повинна зберігати всі дані у хмарному середовищі;
- система повинна бути реалізована без прив’язки до конкретної операційної системи;
- система повинна надавати змогу здійснювати вхід без прив’язки до конкретного пристрою;
- система повинна бути доступна при з’єднанні до інтернету;
- користувацький інтерфейс повинний бути зручним та інтуїтивно зрозумілим у використанні;
- система не повинна потребувати спеціальних навичок від користувача для роботи з нею;

3 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

Після того, як було сформовано функціональні та нефункціональні вимоги, було розроблено UML-діаграму варіантів використання системи, що дозволить більш детально описати реалізацію визначених вимог на концептуальному рівні.

Сценарій використання є чітко визначеною послідовністю дій, що виконує система. Даний етап аналізу допомагає наочно продемонструвати замовнику дії системи зрозумілою для нього мовою, та значною мірою спрощує роботу розробнику, адже детально описує всі функції та результат їх виконання.

Перед розробкою діаграми сценаріїв використання необхідно виділити основні ролі, що будуть взаємодіяти з системою, для того, аби розділити, доступний для кожної, функціонал. Відповідно до, сформованих у попередньому розділі, вимог були визначено таких акторів:

- «Користувач» – неавторизована в системі людина;
- «Відвідувач» – авторизований користувач;
- «Тренер» – авторизований тренер тренажерного залу;
- «Адміністратор» – авторизований адміністратор залу;
- «Власник» – авторизований власник залу;

Після визначення основних дійових осіб у системі, було створено діаграму прецедентів, що наведена у Додатку А. Далі, у наступних підрозділах, описано детально сценарії використання для кожної ролі.

3.1 Опис спільних прецедентів

Так, як система, що розроблюється, призначена для використання авторизованими користувачами, то актору «Користувач» доступні лише два прецедента: авторизація в системі та реєстрація в системі. Вони детально описані в таблицях 3.1-3.2.

Також слід зазначити, що в якості користувача може бути як і раніше зареєстрована в системі особа, так і нова.

Таблиця 3.1 – Прецедент авторизації в системі

Назва	Авторизуватися
Актори	Користувач
Опис	Користувач вводить свої дані для входу в систему
Передумова	Система доступна онлайн, користувач не авторизований
Успішний сценарій	1. Запустити систему 2. Ввести логін та пароль у відповідні поля 3. Система перевіряє введені дані 4. Система авторизує користувача
Результат	Користувач авторизований в системі, та знаходиться на головній сторінці, відповідно до його ролі
Виключення	Система не знайшла запису з введеними даними в базі даних 1. Відобразити повідомлення про помилку 2. Повернення до кроку 2

Таблиця 3.2 – Прецедент реєстрації в системі

Назва	Зареєструватися
Актори	Користувач
Опис	Користувач вводить свої дані для реєстрації в системі
Передумова	Користувач не зареєстрований в системі
Успішний сценарій	1. Запустити систему 2. Ввести логін, пароль та особисті дані у відповідні поля 3. Система перевіряє введені дані 4. Система реєструє нового користувача
Результат	Користувач зареєстрований в системі, та знаходиться на сценарії «Авторизуватися»
Виключення	Система знайшла запис з введеним логіном в базі даних 1. Відобразити повідомлення про помилку 2. Повернення до кроку 2

3.2 Опис прецедентів для відвідувача

В даному підрозділі розглянуто основні варіанти використання системи для відвідувача, що наведені в таблицях 3.3-3.7.

Таблиця 3.3 – Прецедент перегляду доступних залів

Назва	Переглянути перелік залів
Актори	Відвідувач
Опис	Відвідувач переглядає список тренажерних залів мережі для ознайомлення
Передумова	Відвідувач авторизований в системі, наявність залів в базі даних
Успішний сценарій	1. Відвідувач обирає пункт «Зали» в навігаційній панелі 2. Система виконує пошук в базі даних 3. Відображення сторінки з переліком залів
Результат	Відвідувач відкрив сторінку з залами мережі
Виключення	При виникненні помилки при спробі отримати дані, відвідувач буде направлений на сторінку з відображенням тексту помилки

Таблиця 3.4 – Прецедент перегляду інформації про зал

Назва	Переглянути інформацію про зал
Актори	Відвідувач
Опис	Відвідувач переглядає інформацію про обраний зал
Передумова	Відвідувач авторизований в системі
Успішний сценарій	1. Відвідувач використовує сценарій «Переглянути перелік залів» 2. Відвідувач обирає бажаний зал 3. Система відображає сторінку з інформацією про зал
Результат	Відвідувач відкрив сторінку з інформацією про зал
Виключення	При виникненні помилки при спробі отримати дані, відвідувач буде направлений на сторінку з відображенням тексту помилки

Таблиця 3.5 – Прецедент запису на тренування

Назва	Записатися на тренування
Актори	Відвідувач
Опис	Відвідувач записується на заплановане тренування
Передумова	Відвідувач авторизований в системі
Успішний сценарій	1. Обрати пункт «Запис на тренування» в навігаційній панелі 2. Обирати дату та час 3. Обирати тренера 4. Відправити запит на тренування
Результат	Відвідувач записався на тренування
Виключення	При виникненні помилки при спробі надіслати дані, відвідувач буде направлений на сторінку з відображенням тексту помилки

Таблиця 3.6 – Прецедент перегляду вправи

Назва	Переглянути техніку виконання вправи
Актори	Відвідувач
Опис	Відвідувач переглядає базу знань з вправи для вивчення техніки
Передумова	Відвідувач авторизований в системі
Успішний сценарій	1. Обрати пункт «База знань» в навігаційній панелі 2. Обирати вправу з наявного переліку
Успішний сценарій	3. Натиснути на кнопку «Інформація» напроти потрібної вправи
Результат	Відвідувач на сторінці з детальною інформацією про вправу
Виключення	При виникненні помилки при спробі отримати дані, відвідувач буде направлений на сторінку з відображенням тексту помилки

Таблиця 3.7 – Прецедент перегляду особистої інформації

Назва	Переглянути особисту інформацію
Актори	Відвідувач

Продовження таблиці 3.7

Опис	Відвідувач переглядає особисту інформацію для ознайомлення
Передумова	Відвідувач авторизований в системі
Успішний сценарій	1. Відвідувач обирає пункт «Особистий кабінет» в навігаційній панелі 2. Система відкриває сторінку з особистими даними відвідувача
Результат	Відвідувач відкрив свій особистий кабінет
Виключення	При виникненні помилки при завантажені сторінки, відвідувач буде направлений на сторінку з відображенням тексту помилки

3.3 Опис прецедентів для тренера

В даному підрозділі розглянуто основні варіанти використання системи для тренера залу, що наведені в таблицях 3.8-3.10.

Таблиця 3.8 – Прецедент перегляду особистої інформації

Назва	Переглянути особисту інформацію
Актори	Тренер
Опис	Тренер переглядає інформацію про свій розклад тренувань та заробітну плату
Передумова	Тренер авторизований в системі
Успішний сценарій	1. Тренер обирає пункт «Особистий кабінет» в навігаційній панелі 2. Система відкриває сторінку з особистими даними тренера
Результат	Тренер відкрив свій особистий кабінет
Виключення	При виникненні помилки при завантажені сторінки, тренер буде направлений на сторінку з відображенням тексту помилки

Таблиця 3.9 – Прецедент додавання факту обслуговування тренажеру

Назва	Додати факт обслуговування тренажеру
Актори	Тренер

Продовження таблиці 3.9

Опис	Тренер вносить інформацію про обслуговування тренажеру до системи
Передумова	Тренер авторизований в системі
Успішний сценарій	1. Тренер обирає пункт «Тренажери» в навігаційній панелі 2. Система відображає сторінку з переліком тренажерів 3. Тренер натискає на кнопку обслуговування поряд з тренажером 4. Тренер обирає дату, тип обслуговування 5. Тренер вказує коментар до запису, при необхідності 6. Тренер натискає на кнопку «Внести»
Результат	Система отримала та зберегла інформацію про обслуговування тренажеру
Виключення	При виникненні помилки при завантажені сторінки, тренер буде направлений на сторінку з відображенням тексту помилки

В системі також буде можливість надати даний привілей будь-якій ролі.

Таблиця 3.10 – Прецедент складання тренування підлеглому

Назва	Створити програму тренувань
Актори	Тренер
Опис	Тренер створює програму тренувань для відвідувача
Передумова	Тренер авторизований в системі, та має власних клієнтів
Успішний сценарій	1. Тренер обирає пункт «Клієнти» в навігаційній панелі 2. Система відкриває сторінку з переліком клієнтів тренера 3. Тренер обирає клієнта та натискає на кнопку «Створити тренування» 4. Система відкриває сторінку створення тренування 5. Тренер натискає «Створити вправу» 6. Тренер заповнює поля: назва, кількість повторів та підходів, опис

Продовження таблиці 3.10

Успішний сценарій	7. Тренер натискає на кнопку «Додати в список» 8. Тренер формує список та натискає кнопку «Зберегти»
Результат	Система зберегла програму тренування для користувача
Виключення	При виникненні помилки при відправці даних, тренер буде направлений на сторінку з відображенням тексту помилки

3.3 Опис прецедентів для адміністратора

В даному підрозділі розглянуто основні варіанти використання системи для адміністратора спортивного залу, що наведені в таблицях 3.11-3.14.

Таблиця 3.11 – Прецедент внесення присутності відвідувача

Назва	Відмітити присутність відвідувача в залі
Актори	Адміністратор
Опис	Адміністратор, при зустрічі відвідувача, відмічає його присутність в залі
Передумова	Адміністратор авторизований в системі; відвідувач зареєстрований в системі
Успішний сценарій	1. Адміністратор обирає пункт «Присутність» в навігаційній панелі 2. Система відкриває сторінку з переліком всіх клієнтів залу 3. Адміністратор находить відвідувача по номеру абонемента 4. Адміністратор натискає на кнопку «Відмітити» 5. Система фіксує дату та час
Результат	Адміністратор відмітив відвідування залу клієнтом в системі
Виключення	При виникненні помилки на сторінці, адміністратор буде направлений на сторінку з відображенням тексту помилки

Даний прецедент використовується Адміністратором залу, але також може бути використаний Власником, або ж іншою роллю, що можна додати в систему.

Таблиця 3.12 – Прецедент додавання тренера

Назва	Додати тренера
Актори	Адміністратор
Опис	Адміністратор реєструє в системі нового тренера
Передумова	Адміністратор авторизований в системі
Успішний сценарій	1. Адміністратор обирає пункт «Тренери» в навігаційній панелі 2. Система відкриває сторінку з переліком всіх тренерів залу 3. Адміністратор натискає кнопку «Створити» 4. Система відкриває сторінку створення тренера 5. Адміністратор заповнює дані: логін, пароль, особисті дані 6. Адміністратор встановлює заробітну плату тренеру 7. Адміністратор натискає кнопку «Зберегти»
Результат	Новий тренер успішно зареєстрований в системі
Виключення	При виникненні помилки на сторінці, адміністратор буде направлений на сторінку з відображенням тексту помилки

Таблиця 3.13 – Прецедент редагування інформації тренера

Назва	Редагувати інформацію тренера
Актори	Адміністратор
Опис	Адміністратор змінює інформацію тренеру
Передумова	Адміністратор авторизований в системі
Успішний сценарій	1. Адміністратор обирає пункт «Тренери» в навігаційній панелі 2. Система відкриває сторінку з переліком всіх тренерів залу 3. Адміністратор обирає тренера та натискає на кнопку «Інформація» 4. Система відкриває сторінку з інформацією про тренера 5. Адміністратор обирає опцію «Редагувати» 6. Адміністратор редагує необхідні дані 7. Адміністратор натискає на кнопку «Зберегти»

Продовження таблиці 3.13

Результат	Адміністратор змінив необхідну інформацію тренеру
Виключення	При виникненні помилки на сторінці, адміністратор буде направлений на сторінку з відображенням тексту помилки

Таблиця 3.14 – Прецедент додавання товару

Назва	Додати товар для продажу
Актори	Адміністратор
Опис	Адміністратор реєструє нову одиницю товару, що можна придбати в спортивному залі
Передумова	Адміністратор авторизований в системі
Успішний сценарій	1. Адміністратор обирає пункт «Товари» в навігаційній панелі 2. Система відкриває сторінку з переліком всіх товарів залу 3. Адміністратор натискає на кнопку «Додати» 4. Система відкриває сторінку створення товару 5. Адміністратор заповнює поля: назва, код, вартість 6. Адміністратор натискає кнопку «Створити»
Результат	Адміністратор створив нову одиницю товару в системі
Виключення	При виникненні помилки на сторінці, адміністратор буде направлений на сторінку з відображенням тексту помилки

3.4 Опис прецедентів для власника

В даному підрозділі розглянуто основні варіанти використання системи для власника спортивного клубу.

Як видно з графічного зображення діаграми варіантів використання (Додаток А), актор «Власник» може повністю брати участь у сценаріях актора «Адміністратор», завдяки наслідуванню, та дещо доповнює свій функціонал. Основні прецеденти, що відрізняються від ролі «Адміністратор», наведені в таблицях 3.15-3.18.

Таблиця 3.15 – Прецедент редагування заробітної плати адміністратора

Назва	Редагувати заробітну плату адміністратора
Актори	Власник
Опис	Власник змінює зарплатню обраному адміністратору
Передумова	Власник авторизований в системі
Успішний сценарій	1. Власник обирає пункт «Адміністратори» в навігаційній панелі 2. Система відкриває сторінку з переліком всіх адміністраторів 3. Власник обирає адміністратора та натискає на кнопку «Інформація» 4. Система відкриває сторінку з інформацією про адміністратора 5. Власник обирає опцію «Редагувати» 6. Власник редагує заробітну плату 7. Власник натискає на кнопку «Зберегти»
Результат	Власник змінив заробітну плату адміністратору
Виключення	При виникненні помилки на сторінці, власник буде направлений на сторінку з відображенням тексту помилки

Таблиця 3.16 – Прецедент додавання адміністратора

Назва	Додати адміністратора
Актори	Власник
Опис	Власник реєструє нового адміністратора в системі для обраного клубу
Передумова	Власник авторизований в системі
Успішний сценарій	1. Власник обирає пункт «Адміністратори» в навігаційній панелі 2. Система відкриває сторінку з переліком всіх адміністраторів 3. Власник натискає кнопку «Створити» 4. Система відкриває сторінку для створення нового адміністратора 5. Власник заповнює поля: логін, пароль, зал

Продовження таблиці 3.16

Успішний сценарій	6. Власник натискає кнопку «Зберегти»
Результат	Створений новий адміністратор в системі
Виключення	При виникненні помилки на сторінці, власник буде направлений на сторінку з відображенням тексту помилки

Таблиця 3.17 – Прецедент перегляду інформації про зал

Назва	Переглянути інформацію про зал
Актори	Власник
Опис	Власник переглядає статистичну інформацію про зал для звітності
Передумова	Власник авторизований в системі
Успішний сценарій	1. Власник обирає пункт «Зали» в навігаційній панелі 2. Система відкриває сторінку з переліком всіх залів в мережі 3. Власник натискає на кнопку «Інформація» напроти обраного залу 4. Система відкриває сторінку з інформацією про зал
Результат	Система відкрила сторінку з повною інформацією про обраний зал
Виключення	При виникненні помилки на сторінці, власник буде направлений на сторінку з відображенням тексту помилки

Таблиця 3.18 – Прецедент створення нового залу в системі

Назва	Створити зал
Актори	Власник
Опис	Власник відчинив новий тренажерний зал своєї мережі та додає його у систему
Передумова	Власник авторизований в системі

Продовження таблиці 3.18

Успішний сценарій	1. Власник обирає пункт «Зали» в навігаційній панелі 2. Система відкриває сторінку з переліком всіх залів в мережі 3. Власник натискає на кнопку «Створити Зал» 4. Система відкриває сторінку з формою для створення залу 5. Власник заповнює поля назва, адреса, додаткова інформація. 6. Власник обирає адміністратора залу. 7. Власник натискає на кнопку «Створити»
Результат	Система створила запис про новий зал в мережі
Виключення	Система знайшла запис з введеною адресою 1. Відобразити повідомлення про помилку 2. Повернення до кроку 4

В даному розділі, після повторного ретельного аналізу вимог та існуючих аналогів, було визначено основних користувачів, що будуть взаємодіяти з системою, дії, які вони можуть виконувати та за яких умов. Розроблено UML-діаграму сценаріїв використання, що повністю розкриває наявний функціонал системи та дає більш чітке розуміння про послуги, які буде надавати система. Після цього, було описано основні прецеденти для кожного користувача, з урахуванням функціональних вимог з попереднього розділу. В описі також було визначено кроки для їх виконання та обробку помилок для кожного варіанту використання.

В результаті виконання даного розділу синтезовано основні функції, що будуть доступні в системі, а також визначено перелік її користувачів.

Наступним кроком був вибір технологій для реалізації системи.

4 ВИБІР ТА ОБҐРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ

Вибір технологій є важливим кроком в процесі розробки програмного забезпечення, адже від нього буде залежати оптимальність розробленої системи, здатність до розширення, швидкість та вартість розробки.

Базуючись на розроблених функціональних та нефункціональних вимогах необхідно визначити стек технологій та архітектуру системи. Стеком технологій називають набір засобів, інструментів, мов програмування та фреймворків, що використовуються розробниками для написання веб та мобільних додатків.

4.1 Тип та архітектура додатку

Говорячи про тип додатку, мається на увазі яким чином користувач повинен буде мати змогу використовувати функціонал системи. Всі додатки умовно можна поділити на:

- комп’ютерні;
- мобільні;
- веб-додатки.

Кожен з цих типів має свої переваги та недоліки, тому іноді розробники навіть поєднують декілька типів при розробці одного програмного продукту.

Провівши аналіз в попередніх розділах, було вирішено розробляти веб-застосунок, адже це забезпечить виконання всіх, поставлених до розробки, вимог. Дане рішення має ряд суттєвих переваг для розробників та користувачів.

Перше і найважливіше – веб-додаток не потрібно нікуди встановлювати, ні на комп’ютер, ні на телефон. Дана перевага значно підвищує доступність системи, що позитивно вплине на відношення користувачів до неї. Додаток не буде прив’язаним ні до якого пристрою, на якому він встановлений, а отже користувач зможе отримати доступ до функціоналу системи з будь-якої точки світу, маючи будь-який гаджет та

доступ до мережі Інтернет. Також це спростить процес розробки, адже не потрібно розробляти окремо додатки для різних комп'ютерних ОС та для ОС телефонів.

Другим важливим плюсом є те, що даний тип додатків не потребує високої продуктивності від пристрою, на якому він використовується. Іншими словами дана система зможе надавати свій функціонал в повній мірі навіть користувачам з малопотужними телефонами чи комп'ютерами.

Також перевагою є те, що веб-застосунки мають підвищену безпеку, адже вони розгортаються на спеціалізованих серверах, які під наглядом досвідчених адміністраторів.

Наступним кроком є вибір архітектури системи. Враховуючи всі вимоги було вирішено використовувати клієнт-серверну архітектуру. В загальному вигляді вона має такий вигляд (рис.4.1).

Рисунок 4.1 – Схема клієнт-серверної архітектури

Дане рішення допоможе оптимізувати роботу системи, адже на відміну від випадку, коли все зберігається на клієнті, воно має такі переваги:

- потужний сервер зможе оброблювати значно більше запитів;
- відсутнє дублювання коду;
- персональні дані в безпеці, зайва інформація прихована від звичайного користувача.

Присутність БД-бази даних забере зайве навантаження з сервера, та допоможе швидко робити вибірки інформації і зберігати дані в разі перезапуску системи.

Також варто приділити увагу вибору типу клієнтського додатку. Веб-застосунки бувають:

- MPA – багатосторінкові додатки;

- SPA – односторінкові додатки.

Перший тип – це традиційні веб-сайти, в якого при кожному обміні даними з клієнтом, відбувається перезавантаження сторінки, тобто сервер щоразу надсилає нову сторінку. Схематично даний вид взаємодії показано на рисунку 4.2.

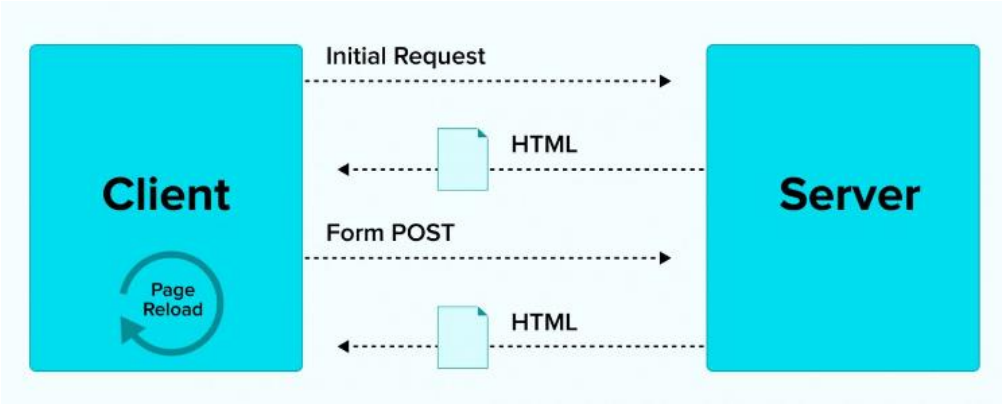

Рисунок 4.2 – Схема обігу даних в багатосторінкових веб-сайтах[7]

Односторінкові веб-додатки – це перш за все сучасний підхід до написання сайту. Це в буквальному сенсі одна сторінка сайту, яка постійно взаємодіє з користувачем, та динамічно перемальовує поточну сторінку. Схема даного типу зображена на рисунку 4.3.

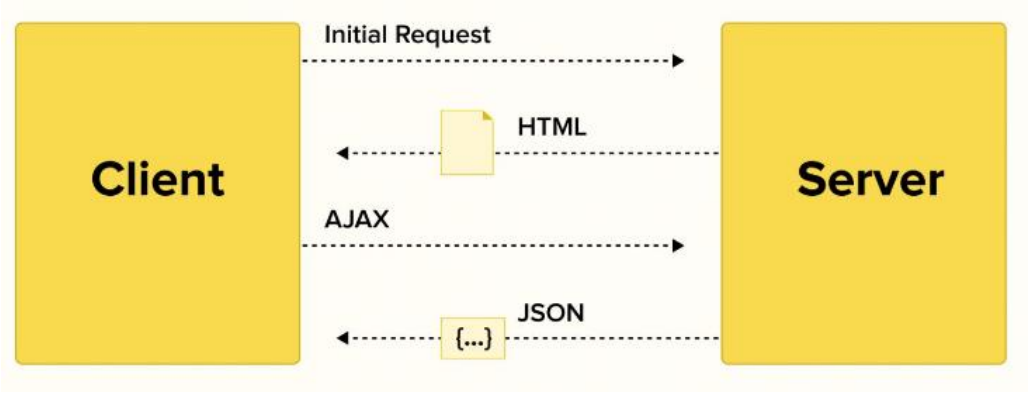

Рисунок 4.3 – Схема обігу даних в односторінкових веб-сайтах[7]

Оскільки однією з вимог є зручний користувацький інтерфейс, то вибір було зроблено на користь SPA. Дані додатки мають кращу швидкодію, можуть ефективно хешувати будь-які локальні дані, та можуть бути використані навіть при втраті

зв’язку, а потім успішно синхронізувати дані при його повторному встановленні. Також односторінкові сайти надають доступ до значно ширшого спектру функціональності, аніж типові форми HTML. Вони використовують HTML5 та AJAX, а більшість ресурсів таких як: HTML, CSS, Scripts, завантажуються лише один раз протягом усього циклу життя програми.

4.2 Платформа та мова програмування

На даних момент серед розробників найбільш популярними є наступні мови програмування:

- Java;
- JavaScript;
- C#.

Мова програмування Java є об’єктно орієнтованою мовою, розробленою компанією Sun Microsystems в 1995 році. Розробники цієї мови черпали натхнення з мови програмування C++, тому синтаксис у них дуже схожий. Однією з основних її переваг є те, що вона не залежить від платформи, на якій виконується додаток. Тобто один й той самий код можна запускати з різних операційних систем. [8]

JavaScript – високорівнева мова програмування, що підтримує функціональний та подіє-орієнтований підхід. Сьогодні є невід’ємною частиною при написані будь-якого сучасного веб-сайту. Особливістю даної мови є те, що вона має динамічну типізацію, на відміну від інших двох, що порівнюються в цьому розділі. Дана мова створювалась, як провідник між Java та користувачами, тому має значно менший поріг входу, тому є легкою у вивченні та використанні. Цьому сприяє також велика кількість існуючих бібліотек та фреймворків, що полегшують процес розробки. Серед недолік варто вказати, що JavaScript не підтримує зчитування файлів та віддалений доступ до системи. Також може по-різному бути інтерпретована різними браузерами, а деякі, такі як Internet Explorer 9, взагалі її не підтримують.

C# – сучасна об’єктно-орієнтована мова програмування, що розроблена та підтримується компанією Microsoft та виконується на платформі .NET. Вона є

прямою спадкоємицею C++, використовує статичну типізацію, та дозволяє розробляти різні типи безпечних та надійних додатків.

.NET – платформа для створення додатків, що існує з 2002 року, та основною ціллю якої є створення застосунків різних типів, які можуть виконуватися на різних платформах. Основою цієї платформи є:

- середовище CLR, що є реалізацією загальномовної інфраструктури мови (CLI), що є міжнародним стандартом Microsoft. Це дозволяє абстрагуватися від мови програмування та виконувати код однаково на всіх платформах;
- платформо-незалежність – з 2016 року, з версії .NET 5, платформа набуває можливості запускатися як на Windows так і на Unix-подібних системах.

Проаналізувавши ці мови програмування, для розробки було обрано платформу .NET та мову програмування C#, адже вона є однією з найпотужніших та розповсюджених мов програмування . Даний вибір також забезпечить, поставленні до системи, вимоги.

4.3 Вибір фреймворку

Фреймворк – це набір заздалегідь написаних програмних компонентів, що значно пришвидшують та покращують процес розробки.

Платформа .NET містить в собі велике різноманіття фреймворків, за допомогою яких можна розробляти як desktop-додатки, так і веб-застосунки. Для розробки останніх доступні такі:

- ASP .NET MVC;
- ASP .NET Core;
- Blazor.

Зазвичай вибір фреймворку для розробки зводиться до власних переваг кожного розробника. Але при розробці даної інформаційної системи було поставлено вимогу, що веб-сайт повинен бути динамічно оновлюваним, тобто SPA. А отже, враховуючи цю та інші вимоги до системи було обрано Blazor.

Blazor – сучасний фреймворк з відкритим кодом, що дозволяє створювати інтерактивні веб-сайти використовуючи C# замість JavaScript. Сайті, написані з використанням даного фреймворку, працюють у всіх сучасних браузерах.

Також Blazor дає можливість розділити логіку програми на серверну та клієнтську частини. Не мало важливим є також той факт, що він дає змогу викликати JavaScript API, що в свою чергу дозволяє використовувати велику екосистему бібліотек JavaScript для інтерфейсу клієнта в той час, як логіка буде написана на C#.

Виходячи з того, що даний фреймворк створювався під впливом таких відомих сучасних фреймворків, як: Angular, React, Vue.js, він отримав таку особливість, як компоненти. Вони – по суті є окремими UI-елементами, що мають власний стиль та логіку роботи, та можуть бути повторно використати в будь-якому місці сайту. Дана особливість допомагає уникнути повторень коду, та значим чином оптимізує його написання. Також компоненти можна використовувати в інших проектах та переносити як бібліотеку класів Razor. Компонентом може бути як окрема форма, так і ціла сторінка.

Існує два типи Blazor додатків:

- Blazor WebAssembly;
- Blazor Server.

Перші – дозволяють створювати інтерактивні веб-додатки клієнтського боку, що запускаються в браузері клієнта та працює за допомогою технології WebAssembly. Завдяки взаємодії з JavaScript, він може звертатися до DOM та API браузера.

Другі – дозволяють створювати серверні додатки, що підтримують ASP .NET Core. Його особливість полягає у тому, що вся обробка подій, оновлення користувацького інтерфейсу, виклики скриптів зі сторони клієнта відбуваються через взаємодію клієнта і сервера через SignalR.

SignalR Core – бібліотека від компанії Microsoft, що призначена для використання при розробці додатків, що працюють в режимі реального часу. Дана технологія використовує двонаправленний зв’язок, щоб підтримувати комунікацію між клієнтом та сервером, що допомагає створити зв’язок типу «видавець-підписник». SignalR використовується в першу чергу там, де дані отримуються в

режимі реального часу. Для забезпечення такого рівня комунікації, вона використовує різні механізми, такі як:

- Web Sockets;
- Server-Side Events;
- Long Polling.

Дані механізми використовуються саме в такій послідовності, в якій вони наведені вище. Тобто спочатку використовується найоптимальніший варіант з них – Web Sockets, якщо ж один з пристроїв не підтримує їх використання, то тоді застосовується Server-Side Events і так далі. Дана бібліотека надає можливості використання таких важливих технологій, як веб-сокети, без необхідності написання коду зайвого коду для їх регулювання, такого як: керування підключеннями, відправка серверу

Також Blazor має інші, базові та необхідні функції, які присутні в звичайному ASP .NET Core, а саме:

- можливсіть конфігурації застосунку за допомогою JSON файлу та доступ до конфігурації в будь-якому місці коду, використовуючи IConfiguration;
- вбудовану підтримку впровадження залежностей – DI, за допомогою якої провайдер сервісів застосунку надає елементу, яку це необхідно, потрібний сервіс;
- логування важливих подій системи;
- робота з сесійним сховищем браузера, що допомагає зберігати необхідні дані, зокрема кукі-файли.

Також до переваг Blazor Server можна віднести:

- менший розмір файлів, що закачуються клієнтом, порівнянно з WebAssembly;
- додаток не обмежений здатностями браузера, та може використовувати серверні потужності для обробки даних;
- підтримка в більш старих версіях браузерів, аніж WebAssemlby.

Водночас з цим, застосунку, що використовує даний фреймворк потрібне постійне підключення до мережі Інтернет.

Отож, провівши детальний аналіз існуючих фреймворків для роботи з .NET платформою, було вирішено, що Blazor Server зможе задовольнити поставленим вимогам.

4.4 Вибір СУБД

Враховуючи сформовані вимоги, розроблювана інформаційна система, повинна зберігати дані програми. Для того, аби з цими даними виконувати базові операції необхідно обрати СУБД.

СУБД – набір засобів (як апаратних, так і програмних), що надає змогу маніпулювати сховищем даних (створювати, налаштовувати, проектувати). Першим кроком є саме вибір типу СУБД.

Типів СУБД існує чимала кількість (ієрархічні, мережеві, об’єктно-орієнтовані, тощо), але найважливішим є вибір серед двох головних категорій:

- нереляційні;
- реляційні.

Особливістю першої категорії є те, що дані зберігаються не у вигляді таблиці, а базовані на структурах даних, таких, як документ. Однією з основних переваг є те, що такі бази даних здебільшого швидше обробляють запити, адже не потребують проходу по декільком таблицям, для формування результату. Також такі бази здатні зберігати велику кількість інформації з малою структурою. До недоліків можна віднести відсутність GUI, великий розмір створюваних файлів, та слід зазначити, що відновлення бази даних є також слабким місцем даного типу.

Реляційні бази даних, на відміну від попередніх, зберігають всю інформацію в таблицях, з зв’язками між сутностями, та є повністю структурованими. Дана особливість гарантує високий рівень цілісності даних. Основними перевагами є відсутність ліміту індексування, що дозволяє швидше виконувати запити та використання мови структурованих запитів – SQL, що є надзвичайно потужним інструментом для роботи з даними. Також серед переваг варто виділити наступні:

- дотримання правил Кодда;

- дотримання принципів ACID;
- забезпечення цілісності даних;
- забезпечення контролю доступу до даних;
- використання потужної мови SQL.

Отже, виходячи з аналізу, можна сказати, що реляційні СУБД мають на меті дотримання узгодженості та доступності даних, жертвуючи стійкістю до розподілу в той час, як нереляційні навпаки: незважаючи на узгодженість прагнуть до високого рівня доступності та підтримки розподіленості даних.

Проаналізувавши дані типи, було вирішено використовувати саме реляційні бази даних, що дадуть змогу працювати з об’єктами програми, та забезпечать високу надійність транзакцій даними.

Наступним кроком є вибір програмного засобу, для взаємодії з сховищем. Серед існуючих аналогів було обрано наступі два для аналізу:

- MySQL;
- PostgreSQL.

Необхідно зазначити, що обидва варіанта є безкоштовними програмами з відкритим кодом.

MySQL – найвідоміший представник безкоштовних СУБД. Існує з 1995 року, та з 2010 року належить програмному гіганту – Oralce[9]. Основними її перевагами є:

- швидкість та простота;
- легке виправлення проблем;
- перевірка на стороні сервера;

До недоліків слід віднести те, що після придбання MySQL компанією Oracle, спільнота розробників її перестала вважати безкоштовною та такою, що з відкритим кодом, що не дає можливості користувачам виправляти помилки. Також через це MySQL має повільніші оновлення. Ще одним недоліком є те, що дана СУБД не забезпечує підтримку наслідування таблиць та матеріалізовані представлення.

Для розробки було обрано PostgreSQL, що є найбільш розвиненим програмним засобом, серед існуючих аналогів. До основних переваг можна віднести:

- повна ACID-сумісність;

- підтримка багатьох типів індексів;
- повна підтримка MVCC, що забезпечує можливість паралельного доступу до бази даних;
- повністю безкоштовна.

Серед недоліків варто виділити, що через свою комплексність дана СУБД є повільнішою та більш складною в виправлені проблем.

Також варто зазначити, що даний програмний комплекс містить інтерфейс для взаємодії з користувачем, є об’єктно-базованим та підтримує більш складні типи даних, як от масиви чи типи даних, створені користувачем.

4.5 Технології для взаємодії з базою даних

Наступним кроком, після вибору СУБД, є вибір інструментарію, що буде використовуватися для взаємодії бази даних безпосередньо з розроблюваною інформаційною системою.

Виходячи з обраної платформи .NET, необхідно визначити готову бібліотеку саме для неї, що дозволить полегшити взаємодію з СУБД. Серед існуючих варіантів, для аналізу було обрано наступні:

- ADO.NET;
- Entity Framework Core.

ADO.NET – вбудована технологія в платформу .NET Framework, що дозволяє працювати з даними. Дана технологія містить в собі уніфікований інтерфейс для роботи з різними СУБД, в яких можуть різнитися мови для створення запитів, наприклад для MS SQL Server це мова T-SQL, а для PostgreSQL це PL-SQL. Для цього, в ній присутні наступні базові класи:

- Connection, що відповідає за встановлення та підтримку підключення до бази даних;
- Command, що надає змогу виконувати різні операції (SELECT, INSERT, UPDATE, DELETE);
- DataReader – зчитує дані, що отримані в результаті запиту;

- DataSet – для зберігання та маніпулювання даними з БД;
- DataAdapter, що призначений для перетворення даних з DataSet, в необхідний для БД формат, та навпаки.

Саме завдяки цим класам відбувається основна робота з базою даних. До основних переваг даної технології варто віднести:

- легка масштабованість;
- висока продуктивність;
- пряме підключення до БД;
- висока гнучкість у розробці.

Серед недоліків:

- великі затрати часу на розробку;
- складність взаємодії;
- труднощі при налагодженні.

Entity Framework Core – це бібліотека з відкритим кодом, що підтримується компанією Microsoft. Даний фреймворк використовує технологію ORM, що дозволяє розробникам працювати на вищому рівні абстракції. Це означає, що вони оперують даними, у вигляді .NET об’єктів, та можуть створювати додатки, без необхідності писати багато рядків коду для шару доступу до даних (DAL), як це зазвичай відбувається при використанні інших технологій, таких, як ADO.NET. Головною концепцією EF є поняття «сутність» (англ. «Entity»), що є набором даних, що відносяться до певного об’єкта. Завдяки цьому взаємодія з базою даних відбувається через роботу з цими «сутностями» та їх колекціями, а не з таблицями.

Entity Framework Core – надбудова над чистим ADO.NET, що спрощує взаємодію з БД. Найбільш відмінною рисою даної технології є те, що для вибірки даних з БД не обов’язково потрібно знати мову SQL, адже її можна замінити на LINQ, від Microsoft. Такий підхід додає уніфікованості технології, адже перед самим запитом, власне EF Core, трансформує даний код в вирази, зрозумілі для кожної конкретної СУБД. Таким чим розробнику немає необхідності вивчати мови кожної СУБД, для взаємодії з ними. Отже EF Core має наступні переваги:

- об’єктно-реляційний фреймворк;

- автоматична генерація моделей та БД, через підхід Code first;
- прямий зв’язок між концептуальними моделями та сутностями БД;
- доступ до CRUD операцій напряму з проекту;
- можливість писати, як на чистому SQL так і на LINQ;
- легкість налагодження, що відбувається засобами в середовища розробки.

Незважаючи на те, що ADO.NET має більшу гнучкість та продуктивність, після проведеного аналізу, для розробки було обрано Entity Framework Core. Продуктивності даної технології цілком вистачить для поставлених вимог, але вона дасть змогу спростити та значно покращити процес розробки та підвищить надійність шару доступу до даних.

4.6 Вибір постачальника хмарних послуг

Важливим кроком в розробці є вибір місця, де буде зберігатися сам додаток. Оскільки не кожний спортивний зал має свою серверну, чи взагалі комп’ютер з обчислювальною потужністю, необхідною для серверу, а також постійне з’єднання з мережею, було вирішено скористатися послугами хмарних сховищ. Для вибору було обрано трьох найякісніших та найвідоміших представників:

- Microsoft Azure;
- Google Cloud Platform (GCP);
- Amazon Web Services (AWS).

Microsoft Azure – продукт компанії Microsoft, що був представлений світу у 2010 році. Даний хмарна платформа має велику популярність та надає доступ до більш ніж 200 різних сервісів своїм клієнтам.

Google Cloud Platform – розроблений компанією Google в 2011 році, хмарний сервіс. Початково створювався для підтримки власних потреб компанії для таких продуктів, як Google Search та YouTube, але згодом став доступним для кожного.

Amazon Web Services – власність компанії Amazon, що є лідером на ринку електронних продаж. Даний продукт був створений у 2006 році, та зазнав багатьох покращень в процесі свого розвитку.

Кожен, з представлених постачальників хмарних послуг, має свої переваги та недоліки та кожен задовольнить вимоги, що були поставлені до системи. З огляду на це, для розробки було вирішено використовувати саме останній, з розглянутих вище, варіантів через такий ряд факторів:

- найбільша кількість зон покриття – 66;
- найбільша доля на ринку;
- найбільша кількість сервісів, що надаються користувачам;
- високий рівень безпеки.

Також важливим показником якості надаваних послуг AWS є те, що дана компанія серед клієнтів має великі компанії з світовим ім’ям. Дана особливість підвищує довіру до Amazon.

Основним недоліком є ціна, яку необхідно сплачувати щомісячно, але зважаючи на всі переваги та на те, що компанія ввела можливість оплати за хвилини використання їхніх послуг, даним недоліком можна знехтувати. Також вони пропонують безкоштовні рішення з невеликими потужностями серверів, що цілком зможуть задовольнити потреби невеликих спортивних залів.

4.7 Вибір середовища розробки

В якості середовища розробки було обрано Visual Studio – потужна IDE, розроблена на мовах програмування C++ та C#, компанією Microsoft. Дана програма повністю підтримує обраний стек технологій, та має ряд переваг:

- зручний інтерфейс;
- вбудована підтримка публікації веб-додатку на AWS;
- штучний інтелект для допомоги написання коду;
- можливість додавати та створювати необмежену кількість бібліотек (нугет пакетів);
- зручні засоби налагодження;
- автоматичне виявлення помилок на етапі написання коду;

- велика кількість підказок по оптимізації та виправленню помилок, що підвищує ефективність написання коду;
- пошук визначення методів та змінних по проекту;
- постійна підтримка виробником свого продукту, та випуски оновлень.

Отже, в результаті виконання даного розділу спершу було вирішено, що інформаційна система повинна бути представлена у вигляді веб-додатку, що зробить її доступною з будь-яких пристроїв, з будь-якою операційною системою, в якого є доступ до мережі Інтернет.

Після цього було проведено аналіз існуючих архітектурних рішень даного типу додатків, та обрано клієнт-серверну архітектуру, що буде реалізована у SPA-додатку. Дане рішення покращить швидкодію системи та зробить зручнішим використання додатку.

Наступним кроком було обрано мову програмування C#, в якості основної мови розробки, та платформу .NET. Після цього обрано фреймворк Blazor для написання веб-додатків, а саме Blazor Server.

Після вибору технології розробки було обрано PostgreSQL для розробки СУБД, та Entity Framework Core для взаємодії останньої із додатком. Також було прийнято рішення про необхідність зберігання серверної частини у хмарному середовищі, значно покращить безпеку системи та її ефективність. В якості постачальника хмарних послуг було обрано Amazon Web Services. Середовищем розробки визначено Microsoft Visual Studio.

5 РОЗРОБКА СТРУКТУРНОЇ СХЕМИ СИСТЕМИ

Розробка структурної схеми є досить вагомим етапом розробки програмного засобу. Вона дає уявлення про загальний вигляд системи, її будову, основні елементи системи, взаємозв’язки між ними та їх призначення[10]. Також структурна схема дозволяє більш наочно проаналізувати стек обраних технологій та їх сумісність.

Отже, зважаючи на обрані технології розробки та вимоги, що були поставлені до системи, було розроблено загальну структурну схему системи, що зображена в Додатку Б.

5.1 Загальна структурна схема

На представленій структурній схемі можна виокремити такі основні компоненти:

- клієнтська частина додатку;
- серверна частина додатку;
- сховище даних.

Клієнтська частина додатку є графічним інтерфейсом для взаємодії користувачів різних типів з системою. За його допомогою буде відбуватись обмін даними між користувачем та системою. На структурній схемі виділено три елементи графічного інтерфейсу:

- інтерфейс користувача;
- інтерфейс адміністративної ролі;
- інтерфейс гостя.

Користувацький інтерфейс призначений для взаємодії з відвідувачами спортивного залу, що вже зареєстровані в системі.

Адміністративний інтерфейс буде доступний для всіх інших ролей в системі, та надаватиме кожній ролі необхідні інструменти відповідно до їх прав.

Інтерфейс гостя буде представлений вікном авторизації для всіх типів користувачів.

Взаємодія з цими інтерфейсами буде відбуватися через інтернет, з будь-якого браузеру, будь-якого гаджета.

Далі за допомогою технології WebSocket, що реалізована бібліотекою SignalR, дані будуть надходити до серверної частини додатку, яку було поділено на наступні елементи:

- логіка побудови сторінок;
- логіка доменної області;
- рівень доступу до даних.

Перший елемент з списку відповідає за побудову сторінок користувача, та містить в собі Razor компоненти з логікою, написаною за допомогою технології .NET.

Для їх побудови він використовує логіку доменної області, що містить в собі:

- сутності, тобто C# класи;
- інтерфейси;
- сервіси

Дані елементи необхідні для функціонування програми та взаємодії двох її інших компонентів. Даний рівень відповідає за логіку, що необхідна для обробки різних даних програми.

Останній, рівень доступу до даних, відповідає за зв’язок програми з базою даних для обміну інформацією. Реалізований за допомогою EF Core, що застосовує об’єктно-реляційне відображення для «спілкування» з БД.

Також варто відзначити DI контейнер, що містить в собі всі сервіси, та видає їх за потреби. Такий підхід значно підвищує можливість тестування програми та повторного використання, а також її обслуговування. Основної цілю є зменшення зв’язку між класами та їхніми залежностями.

Сховище даних реалізоване за допомогою СУБД PostgreSQL, та зберігає в собі всі дані, що необхідні додатку.

Враховуючи вимоги, поставлені в попередніх розділах, на схемі також присутній такий елемент, як хмарне середовище. Він буде зберігати в собі повністю всю програму та базу даних, та надавати необхідні сервіси, які будуть розглянуті,

більш детально, в подальших розділах. В якості постачальника хмарних послуг було обрано Amazon Web Services.

5.2 Структурна схема серверної частини

Якщо загальна структурна схема дає можливість познайомитись з принципом роботи та будовою системи в цілому, то структурна схема серверної частини призначена для більш детального опису архітектури та взаємодії елементів саме серверу. Дана структурна схема зображена у Додатку В.

На ній виокремлено чотири основних елементи, що будуть представлені у вигляді проектів всередині одного рішення:

- додаток для клієнтів;
- додаток для адміністративних ролей;
- бібліотека компонентів;
- бібліотека класів.

Бібліотека класів – проект, що призначено для зберігання усіх необхідних сутностей системи, сервісів та для взаємодії з базою даних. Було прийнято рішення винести ці елементи в окремий проект, оскільки це зменшить об'єм коду, через відсутність повторень для двох додатків, а також пришвидшить процес розробки. Він містить в собі:

- сервіси для роботи з БД;
- сервіси для роботи з даними програми;
- сутності для роботи з сервісами;
- сутності програми;
- інтерфейси;
- розширення.

Сервіси для роботи з БД будуть представленням рівня доступу до даних та будуть використовувати EF Core для взаємодії з СУБД. В сервісах будуть використовуватися сутності та інтерфейси, а також розширення. Також важливо

відзначити, що саме проекті бібліотеки класів буде встановлене підключення до бази даних.

Бібліотека компонентів – проект, що призначений для зберігання Razor компонентів, що можуть використовуватися одночасно в двох додатках. Також вона містить в собі:

- CSS файли;
- JS файли.

Дані файли можуть бути використані Razor компонентами, а також можуть просто зберігатися самі по собі, аби застосувати їх в клієнтських додатках. Такий підхід забезпечить уніфікований стиль інтерфейсу для двох додатків, а також зменшить об’єм коду.

Додаток для клієнтів та для адміністративних ролей містять в собі ряд однакових елементів, але унікальних для кожного проекту. Серед них:

- модуль авторизації;
- логіка побудови сторінок;
- DI контейнер;
- обробники подій;
- прив’язки даних;
- користувацькі/адміністративні сторінки;
- файли зображень.

Модуль авторизації буде представлено сторінкою авторизації та логікою обробки введених даних, для кожного проекту це буде своя сторінка.

Логіка побудови сторінок буде представлена Razor компонентами, що містять в собі обробники подій, що виникають при взаємодії користувача з системою, а також прив’язки даних.

Адміністративні/користувацькі сторінки – Blazor компоненти, або їх сукупність, що будуть відображуватись в браузері користувача, та будуть побудовані відповідно до згаданого вище елементу.

DI контейнери будуть містити в собі сервіси, що необхідні кожному проекту, та видаватимуть їх за потреби.

Файли зображень будуть представлені файлами різних форматів, що будуть необхідні для побудови інтерфейсу користувача.

Відмінністю двох додатків є наявність модуля логіки надання прав ролі в адміністративному додатку. Він буде відповідати за відображення доступного функціоналу для кожної адміністративної ролі відповідно до їх прав.

5.3 Схема взаємодії DAL

Також було приділено увагу розробці схеми взаємодії рівня доступу до даних, адже операції зчитування/запису/редагування інформації в базі даних будуть найбільш розповсюдженими у розроблюваній інформаційній системі.

Дана схема взаємодії представлена у Додатку Г. На ній присутні такі елементи:

- DI контейнер;
- сервіс для роботи з даними конкретної сутності;
- шаблон Repository для конкретної сутності;
- об’єктні сервіси;
- клієнтський провайдер даних;
- провайдер даних ADO.NET;
- база даних.

Призначення DI контейнера вже відоме – зберігати в собі сервіси, а ось інші розглянемо детальніше.

Сервіс для роботи з даними призначений насамперед для виконання базових операцій читання/запису/редагування інформації якоїсь конкретної сутності програми (CRUD операції). Дані операції будуть виконуватися завдяки сервісам, що будуть реалізовані за допомогою патерну «Repository», що дозволить:

- розділити бізнес-логіку, від деталей механізму збереження даних;
- зменшити об’єм та повторюваність коду;
- покращити можливість тестування коду.

Більш детально реалізацію даного патерну наведено у розділі 8.

Далі за допомогою вищеописаних сервісів надсилається LINQ-запит до сховища даних, який потрапляє в об’єктні сервіси. Дані сервіси забезпечують взаємодію з об’єктами клієнтської частини, де останні перетворюються в дерево команд. Також цей елемент відповідає за контроль поточного стану об’єктів, що дозволяє зберігати зроблені в них зміни. Окрім цього, дані сервіси відповідають за перетворення даних, що надходять від БД до сервісу. Так дані у табличному вигляді будуть перетворені в екземпляри класів.

Наступним є елемент: клієнтський провайдер даних. Він відповідає за взаємодію з базою даних, але не звертається до неї напряму. Задля спрощення архітектури це відбувається за допомогою провайдера даних ADO.NET. При отриманні відповіді від БД, шар клієнтського провайдеру даних перетворює його з звичайної табличної форми в спеціальні об’єкти, та передає їх об’єктним сервісам для остаточної обробки[11].

Провайдер даних ADO.NET використовується саме для безпосереднього звернення до СУБД. Даних шар перетворює отримане дерево команд в звичайний SQL запит, та звертається з ним до бази даних. Після цього, отримавши відповідь, він передає її в табличному вигляді до клієнтського провайдеру даних.

В даному розділі було розроблено загальну структурну схему, структурну схему серверної частини, та структурну схема рівня доступу до даних. Ці схеми наочно демонструють основні елементи системи та їх взаємодію. В процесі розробки було проведено повторний аналіз сумісності обраних технологій, що підтвердив правильність вибору.

6 РОЗРОБКА БАЗИ ДАНИХ

База даних – це сховище даних, що призначені для колективного та багаторазового використання. Також її можна охарактеризувати як динамічно-оновлювану інформаційну модель предметної області.[12]

Зважаючи на сформовані вимоги до системи, в якості СУБД було обрано ProstgreSQL, що відноситься до реляційних баз даних.

Розробка бази даних ділиться на кілька етапів, а саме:

- огляд та аналіз предметної області;
- інфологічне проектування;
- даталогічне проектування;
- фізичне проектування.

Огляд та детальний аналіз предметної області було проведено в розділах 2 та 3, що дало представлення про базові сутності системи.

Фізичне проектування, тобто реалізація БД за допомогою DDL обраної СУБД, буде здійснене за допомогою EF Core, що пришвидшить процес розробки.

Отже в даному розділі буде розглянуто інфологічне, даталогічне проектування та опис бази даних.

6.1 Інфологічне проектування

Першим етапом проектування бази даних є концептуальне проектування. Воно передбачає собою створення інформаційно-логічної моделі БД. Дана модель дозволяє описати концептуальну модель за допомогою узагальнених блоків.[13]

Для побудови інфологічної моделі існує багато різних нотацій та методологій. При розробці даної інформаційної системи було вирішено використовувати нотацію Пітера Чена. Він є одним з основоположників реляційних баз даних. Модель у даному вигляді досить довго використовувалась для графічного представлення предметної області у вигляді сутностей та зв’язків між ними, задля її абстрактного представлення на логічному та концептуальному рівнях.[14]

В процесі розробки даної моделі будемо оперувати наступними трьома термінами:

- сутність – структурний елемент, деяка абстракція об'єкту реального світу, явища або процесу, що містить атрибути;
- атрибут – характеристика сутності;
- зв'язок – елемент взаємодії сутностей.

Відповідно до визначених вимог та діаграми варіантів використання було визначено перелік необхідних сутностей та їх атрибутів для належного функціонування системи (табл. 6.1), та побудовано інфологічну модель бази даних, що наведено у Додатку Д.

Таблиця 6.1 – Перелік основних сутностей та їх атрибутів

Сутності	Атрибути
Клієнти	ПІБ, Дата народження, Абонемент, Дата реєстрації, Логін, Пароль, Відвідування, Номер телефону, Номер паспорта.
Спортивні зали	Назва залу, Адреса, Відвідувачі, Власник, Персонал, Графік роботи, Тренажери, Товари.
Графіки роботи	День, Час початку роботи, Час кінця роботи.
Товари	Назва, Кількість, Вартість.
Тренажери	Назва, Кількість, Виробник, Періодичність обслуговування.
Співробітники	Логін, Пароль, ПІБ, Номер паспорту, Номер телефону, Адреса, Роль(Посада), Заробітна плата, Дата влаштування на роботу, Стать, Вартість тренування (у випадку тренера).
Власники	ПІБ, Зали.
Тренування	Клієнт, Тренер, Дата, Час, Вартість.
Ролі	Назва, Привілеї.
Привілеї	Назва.
Програми тренувань	Клієнт, Тренер, Вартість, Вправи.
Вправи	Назва, Опис техніки виконання, Цільові групи м’язів.

Продовження таблиці 6.1

Сутності	Атрибути
Групи м’язів	Назва, Опис функцій.
Поповнення	Абонемент, Дата поповнення, Сума
Відвідування	Клієнт, Зал, Дата та час візиту.
Обслуговування	Тренажер, Співробітник, Дата та час обслуговування.

Окрім сутностей системи, було визначено також зв’язки між ними, з урахуванням наступних особливостей:

- у співробітника є можливість мати декілька ролей, що є необхідністю у випадку, коли у зала мало співробітників і одна й та сама людина виконує декілька функцій (особливо в невеликих залах);
- у зала/мережі може бути декілька власників, що враховує випадок володіння спортивним клубом декількома людьми та забезпечить рівні права доступу до системи;
- зал може не мати товарів для продажі;
- зал може мати тренажери, що не потрібно обслуговувати;
- клієнтові не обов’язково купувати програму тренувань;
- клієнтові не обов’язково потрібно записуватись на тренування з тренером;

6.2 Даталогічне проектування

Наступним, після розробки інфологічної моделі, кроком розробки є даталогічне проектування. В результаті розробки було створено даталогічну модель (ER-діаграму), що наведено у Додатку Е.

Розроблена схема дає більш чітке уявлення про сутності системи, формалізує зв’язки між ними та дає змогу ввести додаткові сутності, для досягнення останнього та надання необхідних функцій системою. До таких функцій належать:

- користувач повинен мати змогу переглянути базу знань з вправами для обраної групи м’язів;

- створена роль може змінювати набір наданих прав;
- можливість змінювати роль (комбінацію ролей) у адміністративного користувача.

Отже для забезпечення приведених вище функцій системи, та для формалізації зв’язків між вже визначеними сутностями було додано ще такі допоміжні сутності-відношення:

- вправи-групи м’язів, де кожна вправа повинна мати одну чи декілька цільових груп м’язів;
- співробітник-роль, що визначає, що жоден співробітник системи не може бути без ролі;
- роль-привілей, де кожній ролі повинен бути присвоєний щонайменше один привілей;
- вправи-програми, де в кожній програмі тренувань присутня щонайменше одна вправа;
- зали-графіки, що визначає графік роботи кожного залу.

Згідно з розробленою ER-діаграмою було обрано первинні та зовнішні ключі для всіх сутностей, що дає чітке уявлення про фізичну реалізацію зв’язків, що визначені у інфологічній моделі. Дані інформація наведена у таблиці 6.2.

Таблиця 6.2 – Належність первинних та зовнішніх ключів до сутностей

Сутність	Первинний ключ	Зовнішній ключ
Клієнти	Id.	Id Залу.
Спортивні зали	Id.	Не містить зовнішніх ключів
Графіки роботи	Id	Не містить зовнішніх ключів
Товари	Id.	Id Залу.
Тренажери	Id.	Id Залу.
Співробітники	Id.	Id Залу.
Власники	Id.	Id Залу.
Тренування	Id.	Id Співробітника, Id Клієнта
Ролі	Id.	Не містить зовнішніх ключів

Продовження таблиці 6.2

Сутність	Первинний ключ	Зовнішній ключ
Привілеї	Id.	Не містить зовнішніх ключів
Програми тренувань	Id.	Id Співробітника, Id Клієнта
Вправи	Id.	Не містить зовнішніх ключів
Групи м’язів	Id.	Не містить зовнішніх ключів
Поповнення	Id.	Id Клієнта.
Відвідування	Id.	Id Клієнта.
Обслуговування	Id.	Id Тренажера, Id Співробітника.
Вправи-групи м’язів	Id.	Id Вправи, Id Групи м’язів
Співробітник-роль	Id.	Id Співробітника, Id Ролі
Роль-привілей	Id.	Id Ролі, Id Привілею
Вправи-програми	Id	Id Вправи, Id Програми тренувань.
Зали-графіки	Id	Id Залу, Id Графіку

Визначених ключів буде достатньо для забезпечення необхідних зв’язків сутностей системи. Наступним кроком була розробка таблиць бази даних.

6.3 Опис схеми бази даних

Після розробки даної ER-діаграми було розпочато фізичне моделювання БД. Для цього було описано схему бази даних, з зазначенням всіх необхідних атрибутів визначених сутностей та з застосуванням обмежень цілісності для реляційних БД.

У таблицях 6.3-6.22 наведені всі атрибути відповідної сутності з зазначенням назви, типу даних та обмежень цілісності. В якості типів даних було використано стандартні типи для СУБД PostgreSQL.

В таблицях також присутні наступні позначення: PK – первинний ключ, FK – зовнішній ключ, NN – не порожнє значення, UN – унікальність поля, CHECK – перевірка значення на відповідність певній умові, DFLT – значення за змовчуванням. Даний набір дозволяє дотримуватись обмежень цілісності даних.

Таблиця 6.3 – Clients (Клієнти)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid.	PK, NN
GymId	Ідентифікатор залу.	Uuid	FK, NN
Login	Логін.	Varchar(50)	UN
Password	Пароль.	Varchar(50)	NN
FirstName	Ім’я	Varchar(100)	NN, CHECK(‘ ’)
LastName	Прізвище	Varchar(100)	NN, CHECK(‘ ’)
SubscriptionNumber	Номер абонементу.	Int	NN, UN
SubscriptionType	Тип абонементу.	Varchar(30)	NN
IsActive	Статус абонементу	Boolean	DFLT False
BirthdayDate	Дата народження.	Date	-
RegistrationDate	Дата реєстрації	Timestamp	NN
Phone	Номер телефону.	Char(13)	UN
Passport	Номер паспорту.	Varchar(9)	UN, NN

Таблиця 6.4 – Refills (Поповнення)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
SubscriptionNumber	Номер абонементу.	Int	NN
Date	Дата поповнення.	Timestamp	NN
Total	Сума поповнення.	Numeric(8,3)	NN

Таблиця 6.5 – Visitings (Відвідування)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
ClientId	Ідентифікатор клієнта.	Uuid	FK, NN
GymId	Ідентифікатор залу.	Uuid	FK, NN
Date	Дата та час.	Timestamp	NN

Таблиця 6.6 – Gyms (Спортивні зали)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
Title	Ім’я	Varchar(100)	NN, CHECK(‘ ’)
Address	Адреса залу.	Text	NN
Description	Короткий опис.	Text	-

Таблиця 6.7 – Schedules (Графіки роботи)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
Day	День неділі.	Varchar(20)	NN, CHECK(‘ ’)
StartTime	Час початку роботи.	Time	NN
EndTime	Час кінця роботи.	Time	NN

Зв’язок типу «багато до багатьох», між таблицями Зали ти Графіки роботи, відповідно до розробленої ER-діаграми реалізується допоміжною таблицею Зали-Графіки, що наведена у таблиці 6.8.

Таблиця 6.8 – GymsSchedules (Зали-графіки)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
ScheduleId	Ідентифікатор графіку.	Uuid	FK, NN
GymId	Ідентифікатор залу.	Uuid	FK, NN

Таблиця 6.9 – Machines (Тренажери)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
GymId	Ідентифікатор залу.	Uuid	FK, NN
Title	Назва тренажеру.	Varchar(100)	NN
Manufacturer	Виробник.	Varchar(100)	
Quantity	Кількість одиниць в залі	Smallint	NN, CHECK(>0)
Period	Періодичність обслуговуваня, місяців	Smallint	NN, CHECK(>0)

Таблиця 6.10 – Employees (Співробітники)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid.	PK, NN
GymId	Ідентифікатор залу.	Uuid	FK, NN
Login	Логін.	Varchar(50)	UN
Password	Пароль.	Varchar(50)	NN
FirstName	Ім’я	Varchar(100)	NN, CHECK(‘ ’)
LastName	Прізвище	Varchar(100)	NN, CHECK(‘ ’)
Address	Адреса проживання.	Text	-
IsMale	Стать.	Boolean	DFLT False
Sallary	Заробітна плата.	Money	NN, CHECK(>0)
BirthdayDate	Дата народження.	Date	-
HireDate	Дата влаштування на роботу	Date	NN
Phone	Номер телефону.	Char(13)	UN
Passport	Номер паспорту.	Varchar(9)	UN, NN
Training Price	Вартість одного тренування	Money	-

Варто відмітити, що необхідний атрибут Гендер, що був визначений на етапі інфологічного проектування, представлений у даній таблиці у вигляді булевого

значення полем IsMale. Дане поле при значенні True буде означати, що це особа чоловічої статі, при False – жіночої. Значення за змовчуванням – True.

Таблиця 6.11 – Roles (Ролі)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
Title	Назва	Varchar(50)	NN, CHECK(‘ ’)

В якості розв’язуючої таблиці зв’язку «багато до багатьох», останніх двох описаних сутностей було створено таблицю Співробітник-Роль, що наведена у таблиці 6.12. Такий підхід дасть змогу призначати одному співробітнику декілька ролей.

Таблиця 6.12 – EmployeesRoles (Співробітник-Роль)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
RoleId	Ідентифікатор тренажеру.	Uuid	FK, NN
EmployeeId	Ідентифікатор співробітника.	Uuid	FK, NN

Таблиця 6.13 – Privileges (Привілеї)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
Title	Назва	Varchar(50)	NN, CHECK(‘ ’)

Також в якості розв’язуючої таблиці зв’язку «багато-до-багатьох» між сутностями ролей та привілеїв, було створено таблицю RolesPrivileges.

Таблиця 6.14 – RolesPrivileges (Роль-Привілеї)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
RoleId	Ідентифікатор тренажеру.	Uuid	FK, NN
PrivilegeId	Ідентифікатор привілею.	Uuid	FK, NN

Таблиця 6.15 – Services (Обслуговування)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
MachineId	Ідентифікатор тренажеру.	Uuid	FK, NN
EmployeeId	Ідентифікатор співробітника.	Uuid	FK, NN
Description	Короткий опис виконаних робіт	Text	-
Date	Дата обслуговування тренажеру.	Timestamp	NN

Таблиця 6.16 – Goods (Товари)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
GymId	Ідентифікатор залу.	Uuid	FK, NN
Title	Назва товару.	Varchar(100)	NN
Manufacturer	Виробник	Varchar(100)	-
Quantity	Кількість одиниць товару	Smallint	NN
Price	Ціна	Money	NN, CHECK(>0)

Таблиця 6.17 – Trainings (Тренування)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
ClientId	Ідентифікатор клієнта.	Uuid	FK, NN
EmployeeId	Ідентифікатор тренера.	Uuid	FK, NN
Date	Дата тренування.	Timestamp	NN
Price	Вартість тренування	Money	NN

Таблиця 6.18 – TrainingPrograms (Програми тренувань)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
ClientId	Ідентифікатор клієнта.	Uuid	FK, NN

Продовження таблиці 6.18

Назва поля	Опис	Тип даних	Обмеження
EmployeeId	Ідентифікатор тренера.	Uuid	FK, NN
Price	Ціна	Money	NN, CHECK(>0)
Date	Дата складання	Date	NN

Таблиця 6.19 – Exercises (Вправи)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
Title	Назва.	Varchar(100)	NN
Technique	Опис техніки.	Text	NN
Description	Короткий опис	Text	-

Варто зазначити, що в даній таблиці також присутнє поле для запису короткого опису вправи для зручності подальшого використання функцій користувачем.

Таблиця 6.20 – ExercisesPrograms (Вправи-програми)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
Назва поля	Опис	Тип даних	Обмеження
ExerciseId	Ідентифікатор вправи.	Uuid	FK, NN
Назва поля	Опис	Тип даних	Обмеження
PragramId	Ідентифікатор програми.	Uuid	FK, NN
Approach	Підходи	Smallint	-
Repetitions	Повторення	Smallint	-

Дана таблиця призначена для розв’язання зв’язку типу «багато до багатьох» між сутностями Програми Тренувань та Вправи. Також в ній можна побачити, що в системі присутня можливість створювати програми тренувань без точної кількості

підходів та повторень у вправах, що дасть можливість варіювати деталізацію розробленої програми, та відповідно вартість.

Таблиця 6.21 – MuscleGroups (Групи м’язів)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
Title	Назва групи.	Varchar(70)	NN
Description	Опис функцій.	Text	-

Поле Description призначене для зберігання додаткової інформації про виконуючі функції м’язів. Дане поле дозволить клієнтові краще зрозуміти біомеханіку виконання вправ.

Таблиця 6.22 – MuscleGroupsExercises (Вправи-Групи м’язів)

Назва поля	Опис	Тип даних	Обмеження
Id	Унікальний ідентифікатор.	Uuid	PK, NN
ExerciseId	Ідентифікатор вправи.	Uuid	FK, NN
MuscleGroupId	Ідентифікатор групи м’язів.	Uuid	FK, NN

В даному розділі було розроблено базу даних для інформаційної системи для підтримки тренажерного залу.

Для цього спочатку було проведено інфологічне проектування, що включало в собі виділення основних сутностей системи, та перелік їх атрибутів, а також розробку інфологічної моделі. Після цього було проведено даталогічне проектування, результатом якого була розроблена даталогічна моделі бази даних, з визначеними всіма необхідними сутностями, а також формалізованими зв’язками між ними. Останнім етапом було фізичне проектувань, в ході якого було описано всі сутності та визначено відповідні типи даних для їх атрибутів.

Розроблена БД повністю відповідає принципам ACID, принципам цілісності даних та задовольняє вимогам, поставленим до системи.

7 РОЗРОБКА ІНТЕРФЕЙСУ СИСТЕМИ

Важливим кроком в розробці інформаційної системи є саме розробка її інтерфейсу для взаємодії з користувачем, адже він є єдиним варіантом взаємодії користувача з системою. Саме цей фактор сильно впливає на привабливість продукту та можливість його подальшого впровадження на ринок.

Оскільки першим, що бачить та відчуває користувач, при взаємодії з системою – її інтерфейс, то він повинен бути максимально привабливим та зручним у використанні. Адже перше враження буде сильно впливати на вибір користувача з приводу подальшого використання системи.

Як вже було сказано, хороший дизайн повинен поєднувати дві важливі складові:

- привабливість (UI);
- зручність (UX).

UI складова дизайну відповідає за сучасний вигляд, поєднання кольорів та шрифтів, тощо. Саме вона відповідає за перше враження користувача.

UX складова, в свою чергу, має на меті зробити сайт максимально функціонально наповненим, та водночас зручним у використанні. Сюди також відноситься інтуїтивне розуміння користувачем, як саме досягти необхідного йому результату, не приходячи до зв’язку з підтримкою чи інструкціям.

Отже, виходячи з обраної архітектури та структури системи, необхідно розробити дизайн для клієнтського сайту та сайту, що буде використовуватися персоналом спортивного залу. Проаналізувавши ринок та існуючі аналоги, було вирішено, що найоптимальнішим варіантом буде розробка мобільної версії сайту для клієнтського додатку, та десктопної версії сайту для адміністративного. Така комбінація зробить продукт найбільш конкурентоспроможним.

Враховуючи поставлені функціональні вимоги до системи, було розроблено графічні інтерфейси, з дотриманням стандартів UI/UX дизайну. Більш детально кожен з них буде розглянуто в підрозділах даного розділу.

7.1 Клієнтський інтерфейс

При розробці інтерфейсу клієнтського сайту, важливим було надати необхідний функціонал з мінімальним набором різного виду кнопок, аби ефективно розподілити простір екрану смартфону.

7.1.1 Авторизація та реєстрація

Отже першим етапом була розробка дизайну вікон авторизації (рисунок 7.1) та реєстрації (рисунок 7.2) користувача в системі. Дані вікна зображені на рисунку 7.1

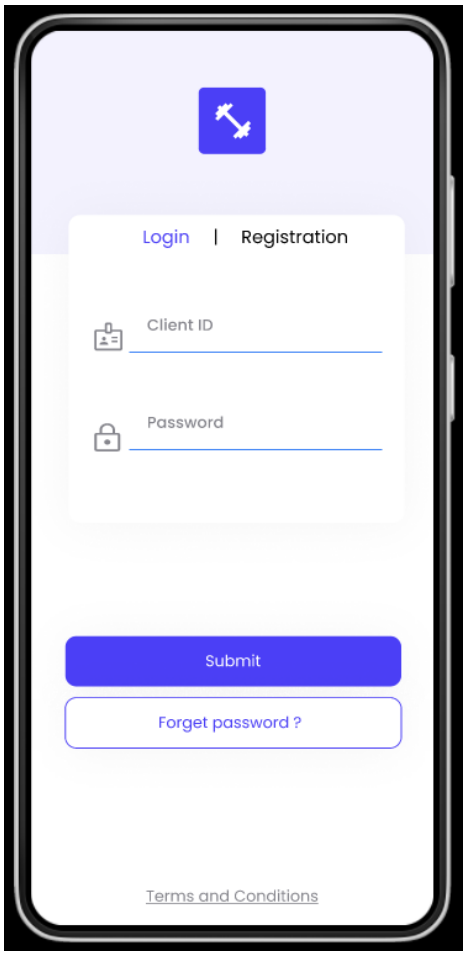

Рисунок 7.1 – Вікно авторизації в системі

За допомогою цього вікна користувач зможе увійти в систему, аби отримати доступ до всіх функцій сайту.

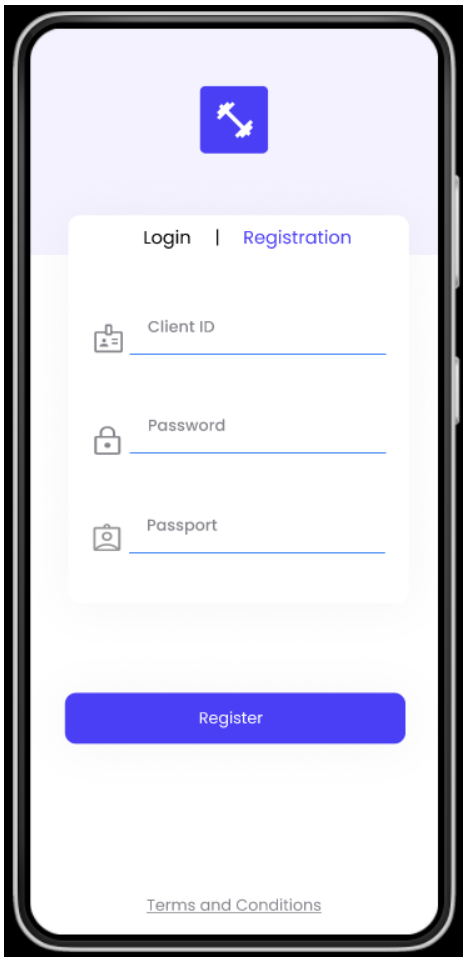

Рисунок 7.2 – Вікно реєстрації в системі

Вікно реєстрації в системі дасть змогу користувачеві самому зареєструватися в системі, без допомоги адміністратора. На ньому присутні такі обов'язкові поля для заповнення:

- номер абонементу, виданого адміністратором;
- ім'я та прізвище;
- номер паспорту.

Наступим кроком було розроблено шаблон головного вікна, що буде загальним для всіх подальших вікон системи. Основною його задачею є навігація по додатку. Було прийнято рішення про створення панелі внизу екрану на всю його ширину з виділенням наступних кнопок:

- профіль;
- база знань;
- зали;

- створити.

7.1.2 Профіль користувача

Вікно пункту меню «Профіль», що зображене на рисунку 7.3 надає клієнту наступний функціонал:

- перегляд особистої інформації;
- редагування особистої інформації;
- перегляд історії відвідувань;
- перегляд історії поповнень абонементу;
- перегляд поточного стану абонементу;
- перегляд придбаних програм тренувань;
- перегляд розкладу назначених тренувань.

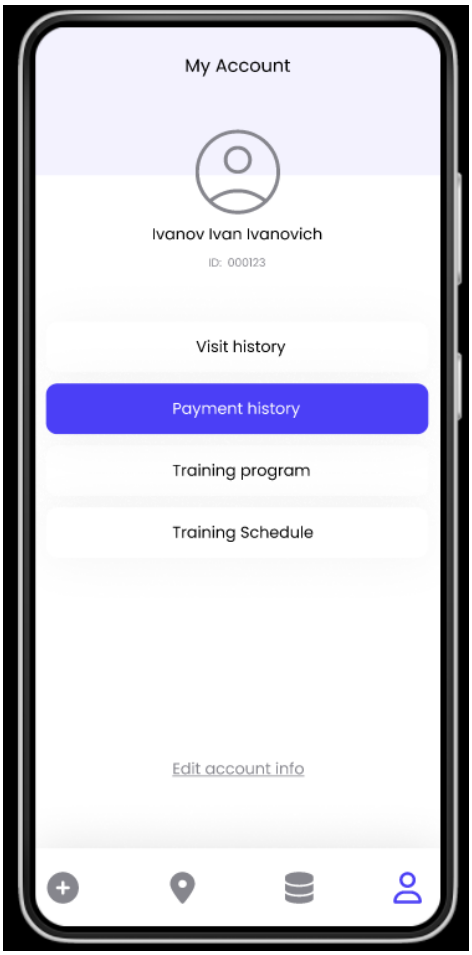

Рисунок 7.3 – Вікно пункту меню «Профіль»

7.1.3 База знань

Даний пункт меню, вигляд інтерфейсу якого зображено на рисунку 7.4, дозволяє користувачеві:

- переглянути перелік існуючих вправ;
- обрати будь-яку для перегляду правильної техніки виконання;
- переглянути короткий опис;
- переглянути які групи м’язів задіяні при виконанні вправи.

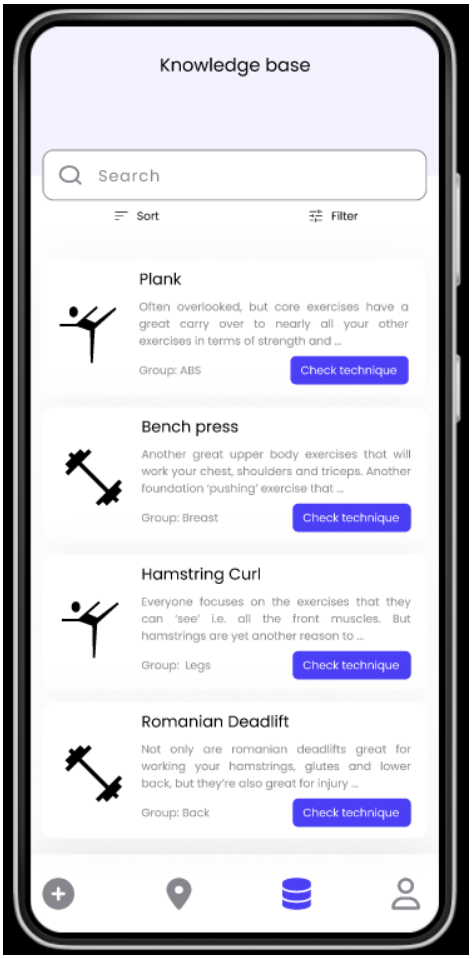

Рисунок 7.4 – Вікно пункту меню «База знань»

Також користувачеві доступна можливість сортування вправ за групами м’язів та пошук вправи за назвою, що значно збільшить зручність використання даного функціоналу.

7.1.4 Зали

Вікно пункту меню «Зали» (рис. 7.5) дозволяє користувачеві переглянути наступну інформацію:

- назву та адресу свого залу;
- назви та адреси інших залів мережі;
- кількість тренерів всіх залів мережі.

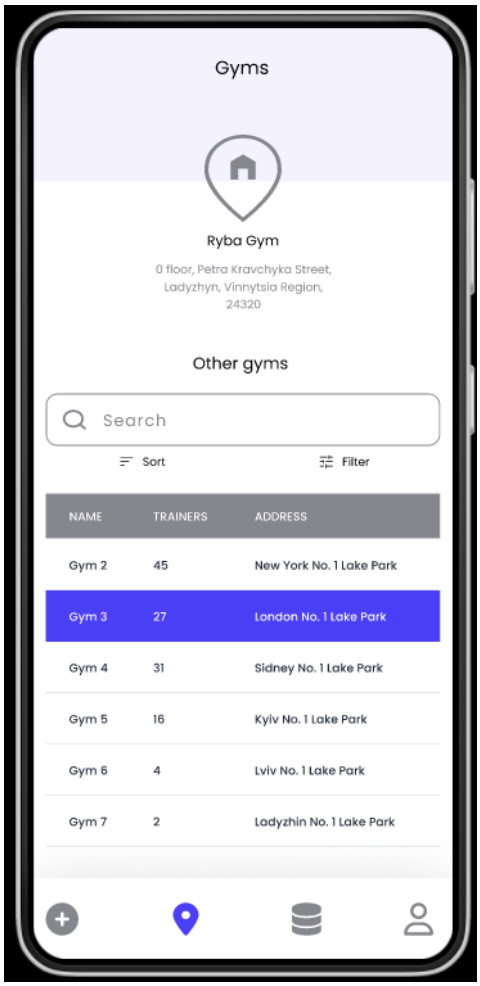

Рисунок 7.5 – Вікно пункту меню «Зали»

7.1.5 Створити запис

Даний пункт меню, що зображено на рисунку 7.6, призначений для створення запису на тренування. Для того, аби записатися, необхідно обрати тренера з

випадаючого списку у верхній частині екрану, обрати бажану дату та час тренування на календарі, після чого натиснути на кнопку внизу екрану.

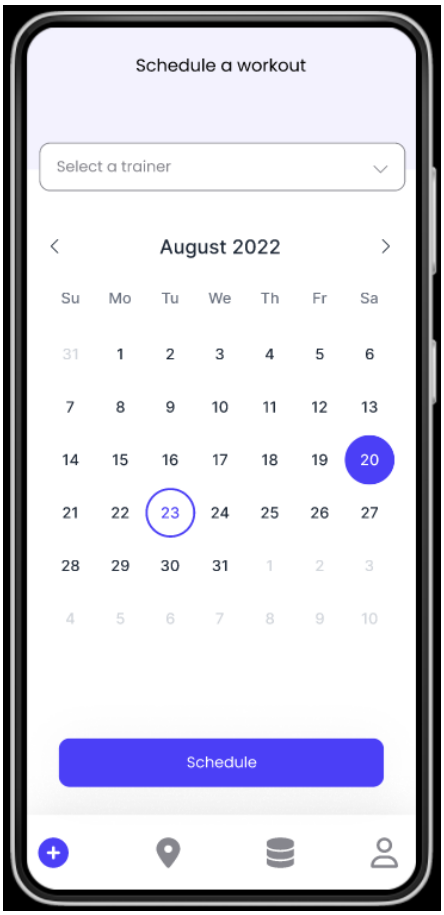

Рисунок 7.6 – Вікно пункту меню «Створити запис»

Після розробки інтерфейсу клієнтського додатку, було розроблено інтерфейс веб-додатку для адміністративних користувачів.

7.2 Адміністративний інтерфейс

Оскільки розроблювана інформаційна система є комплексним рішенням, тобто немає можливості придбати лише один додаток, то й стиль у обох повинен бути дотриманий однаковий.

Зважаючи на те, що в системі будуть тренери, адміністратори та власники, і кожна з них в повній чи частковій мірі повторює якийсь функціонал іншої було

прийнято рішення створити одне спільне вікно авторизації для всіх ролей, а також спільне головне вікно програми з необхідним набором функцій. Даний підхід зменшить час, необхідний на розробку, та ресурси. Також таке рішення дозволить уніфікувати систему, для легшого її розуміння для різних користувачів та дасть змогу з легкістю впроваджувати нові ролі в системі.

Отже, виходячи з описаного вище, було розроблено головне вікно адміністративного додатку, що має збоку панель з пунктами меню, що відображаються для кожної ролі відповідно. Далі в підрозділі було розглянуто основні вікна та зазначено для яких користувачів вони призначені.

7.2.1 Авторизація

Першим кроком була розробка вікна авторизації, що зображене на рисунку 7.7.

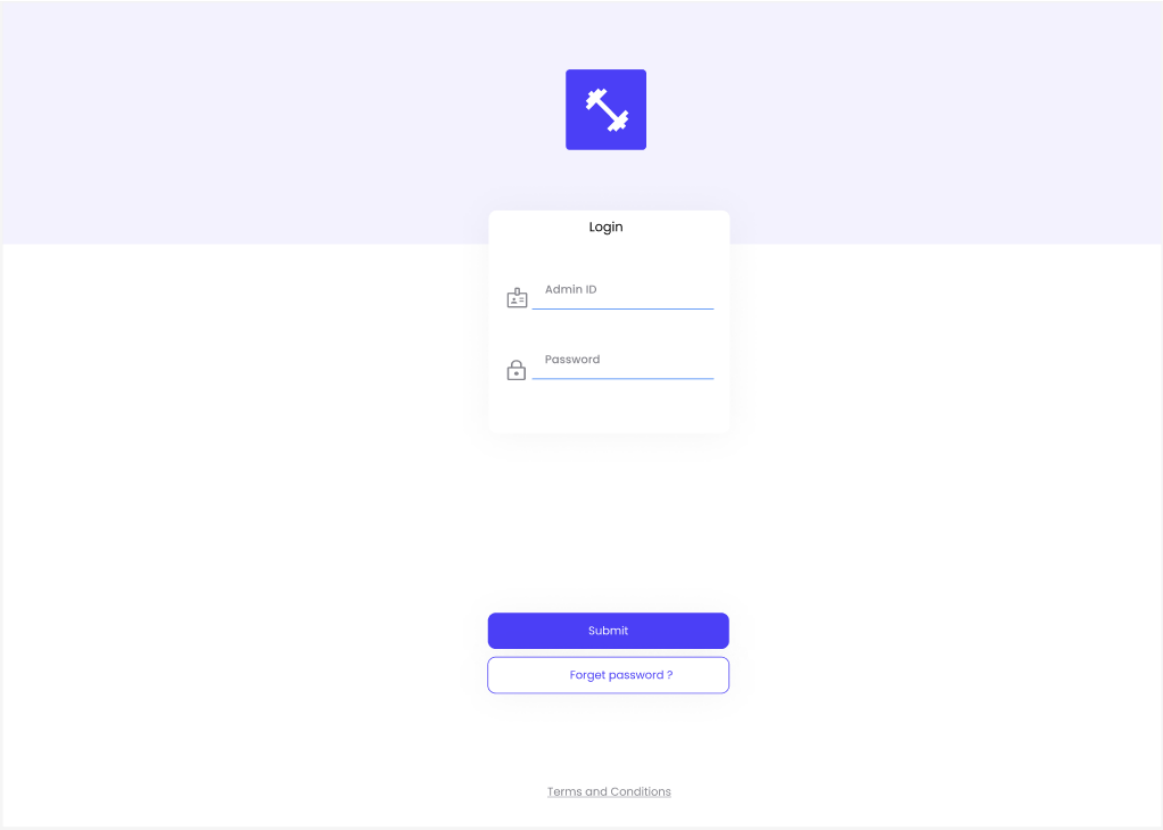

Рисунок 7.7 – Вікно авторизації співробітника

Дане вікно є спільним для всіх ролей і виконує функцію допуску користувачів до системи, а також передає данні про роль користувача головному вікну для побудови бічного меню.

7.2.2 Профіль

Даний пункт меню (рис. 7.8) є також спільним для всіх користувачів, та містить наступні функції:

- перегляд особистих даних;
- перегляд окладу;
- розклад запланованих тренувань з необхідною інформацією.

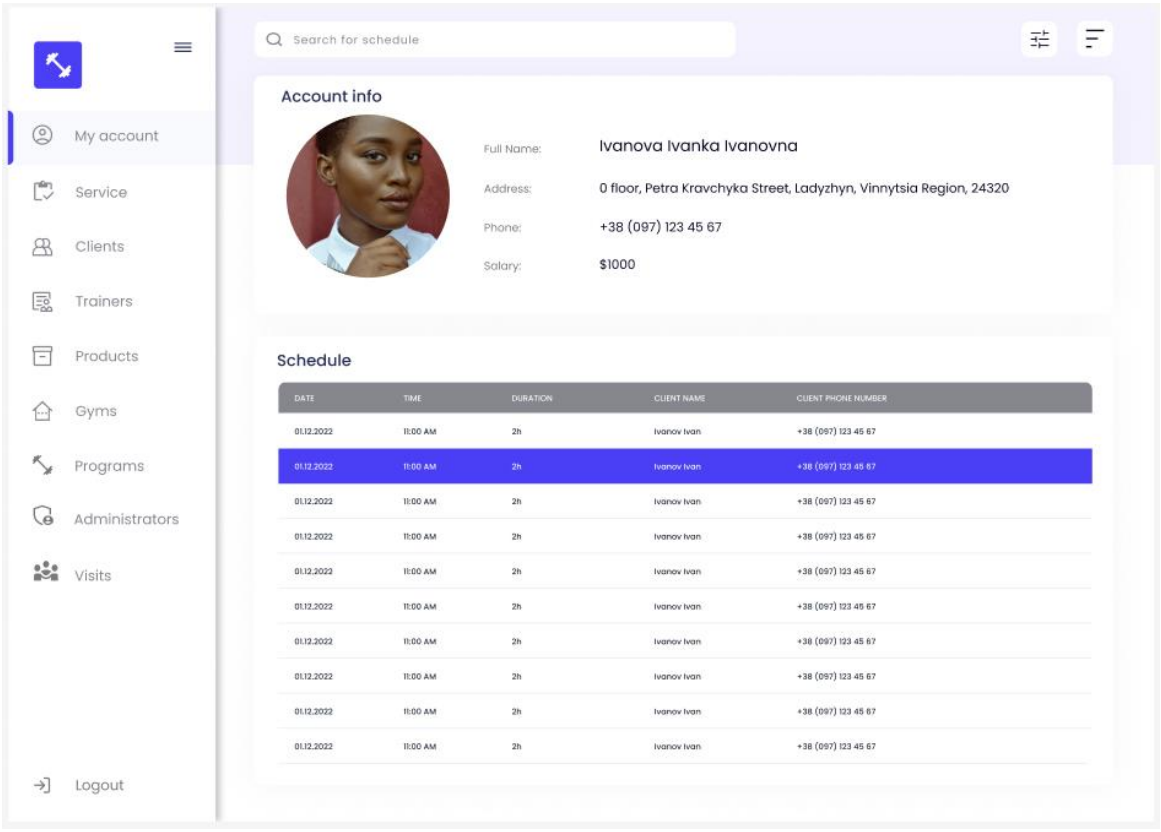

Рисунок 7.8 – Вікно пункту меню «Профіль»

Варто зазначити, що розклад запланованих тренувань відображається лише для користувачів, що мають роль тренера.

7.2.3 Обслуговування тренажерів

Вікно з інформацією про обслуговування тренажерів (рис. 7.9) також доступне для всіх, існуючих на момент розробки, ролей. Воно надає доступ до наступної інформації:

- історія обслуговувань тренажеру;
- створення факту обслуговування.

Для відображення історії обслуговувань, користувач повинен спочатку обрати тренажер з випадаючого списку. Також для створення факту обслуговування в користувача повинен бути відповідний привілей.

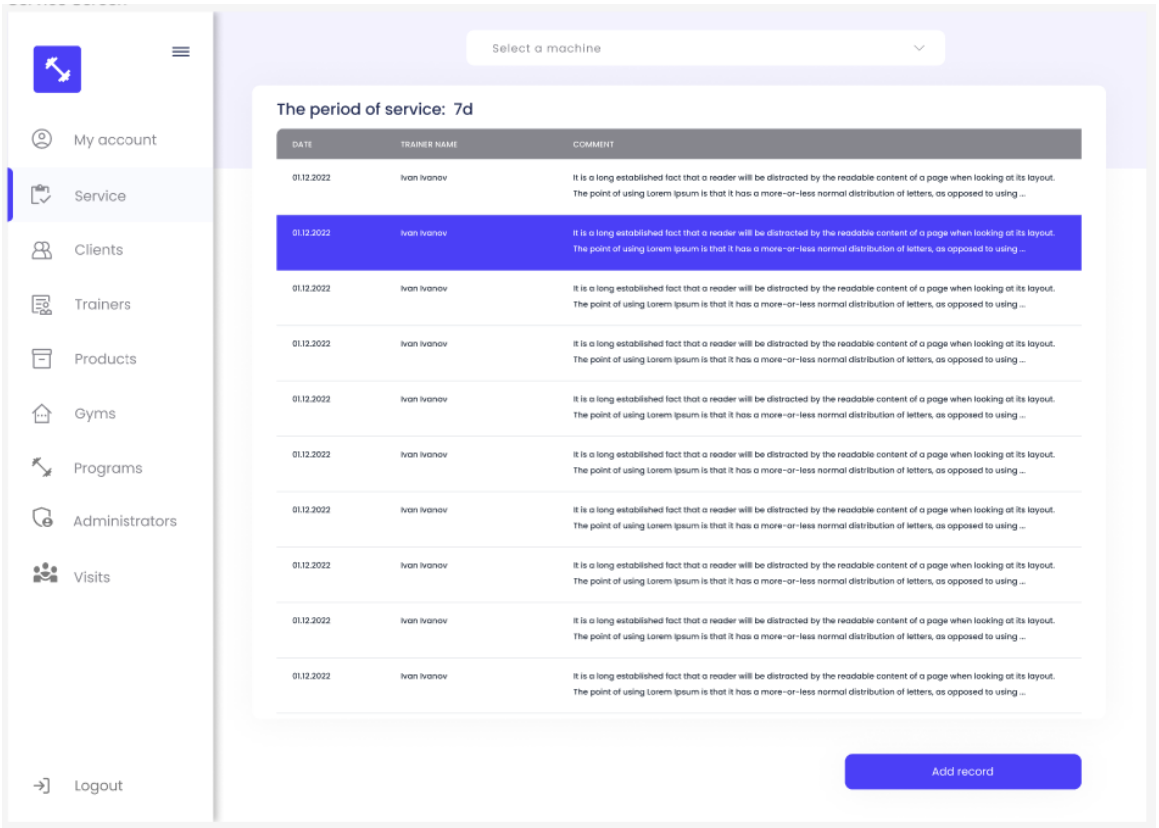

Рисунок 7.9 – Вікно пункту меню «Обслуговування тренажерів»

7.2.4 Клієнти

Даний пункт меню, інтерфейс якого зображено на рисунку 7.10, є спільним для цих трьох груп користувачів, та надає доступ до таких функцій:

- перегляд інформації про клієнтів залу;
- редагування клієнтів залу.

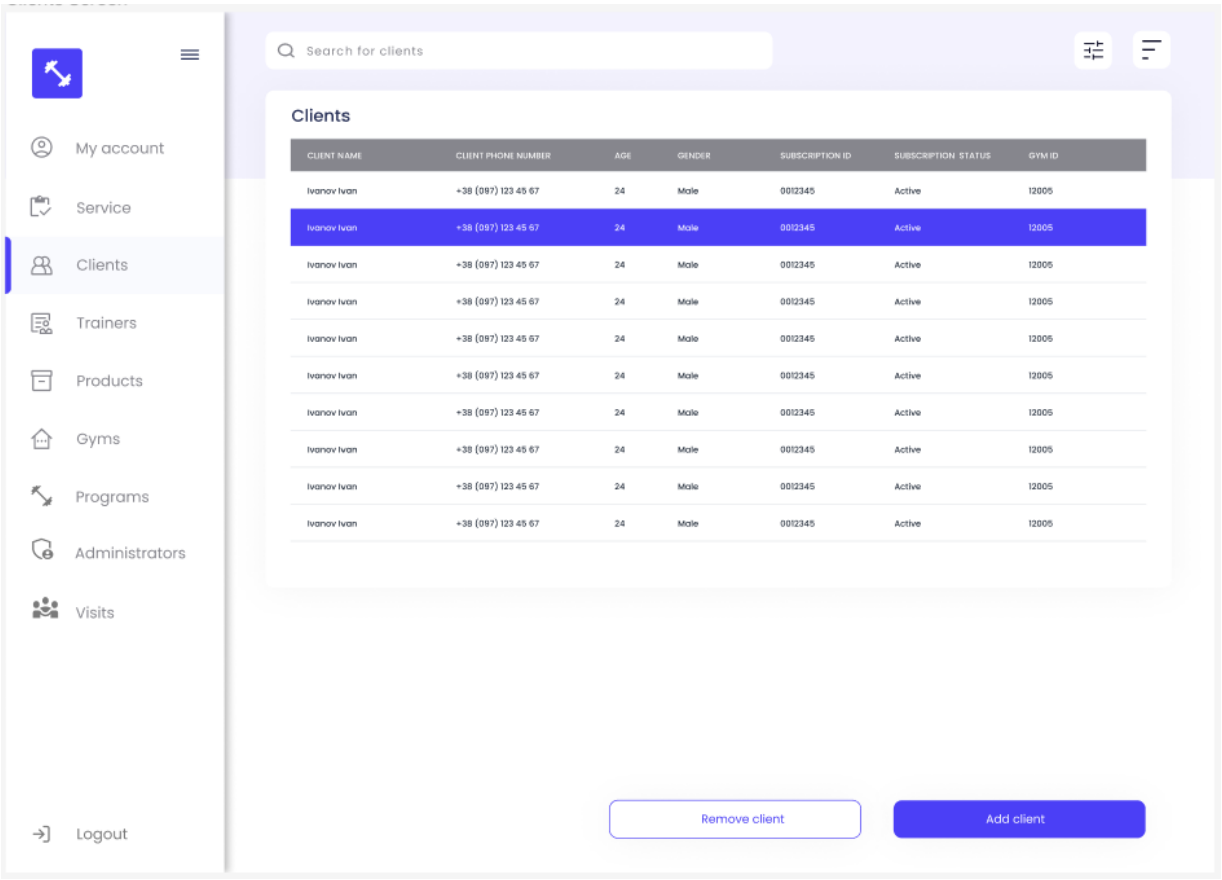

Рисунок 7.10 – Вікно пункту меню «Клієнти»

У випадку використання цього пункту меню власником буде відображено всіх клієнтів всієї мережі. Також варто вказати, що в тренерів немає привілеїв на редагування клієнтів, тому для цієї групи дані функції будуть недоступними.

7.2.5 Програми тренувань

Цей пункт меню (рис. 7.11) призначено для користувачів з роллю тренера, та надає наступний функціонал:

- перегляд власних створених тренувань;
- створення програм тренувань;
- редагування існуючих програм тренувань;

- видалення програм тренувань;
- сортування тренувань за ціною та датою створення;
- пошук програми по назві.

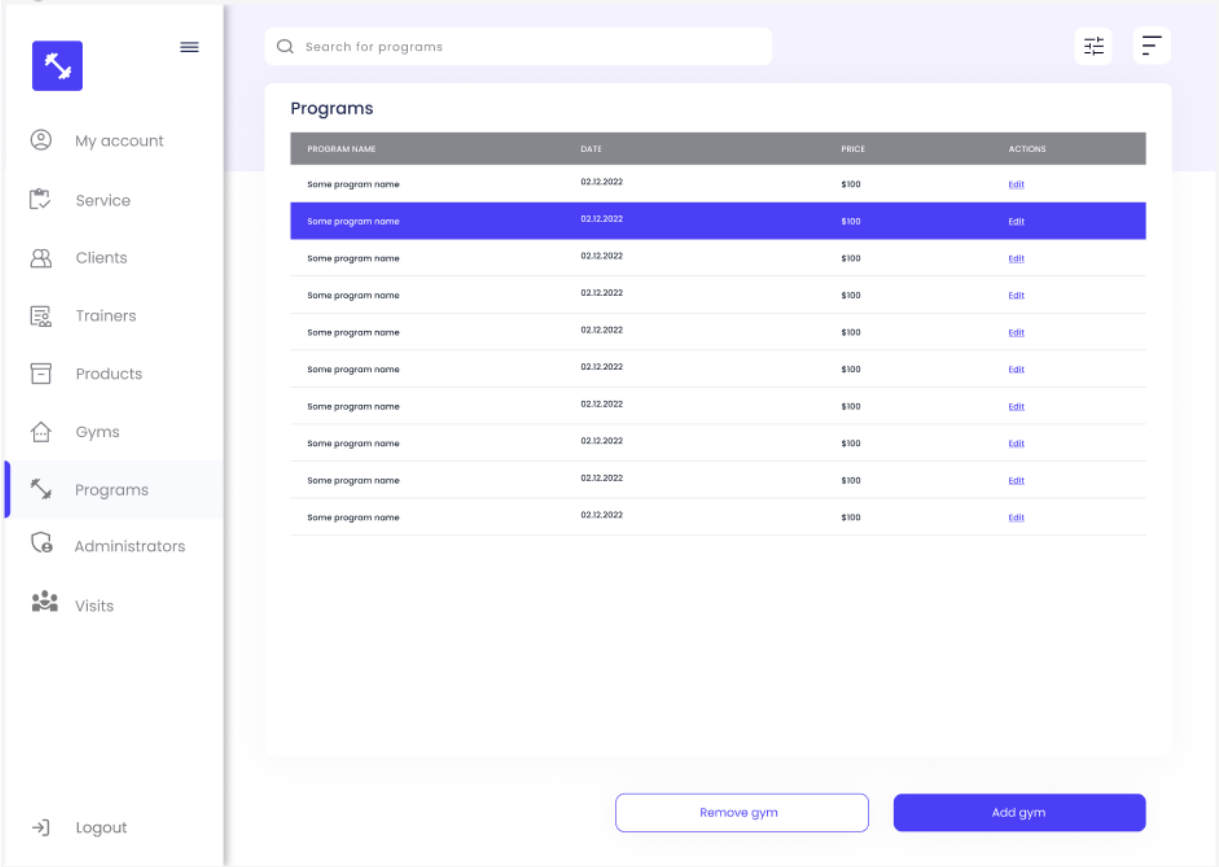

Рисунок 7.11 – Вікно пункту меню «Програми тренувань»

7.2.6 Тренери

Вікно пункту меню «Тренери» (рис. 7.12) призначене для користування адміністраторам та власникам. Це вікно дозволяє виконувати такі операції:

- перегляд зведеної інформації про тренерів залу;
- зміна окладу тренера;
- створення нового тренера;
- видалення тренера.

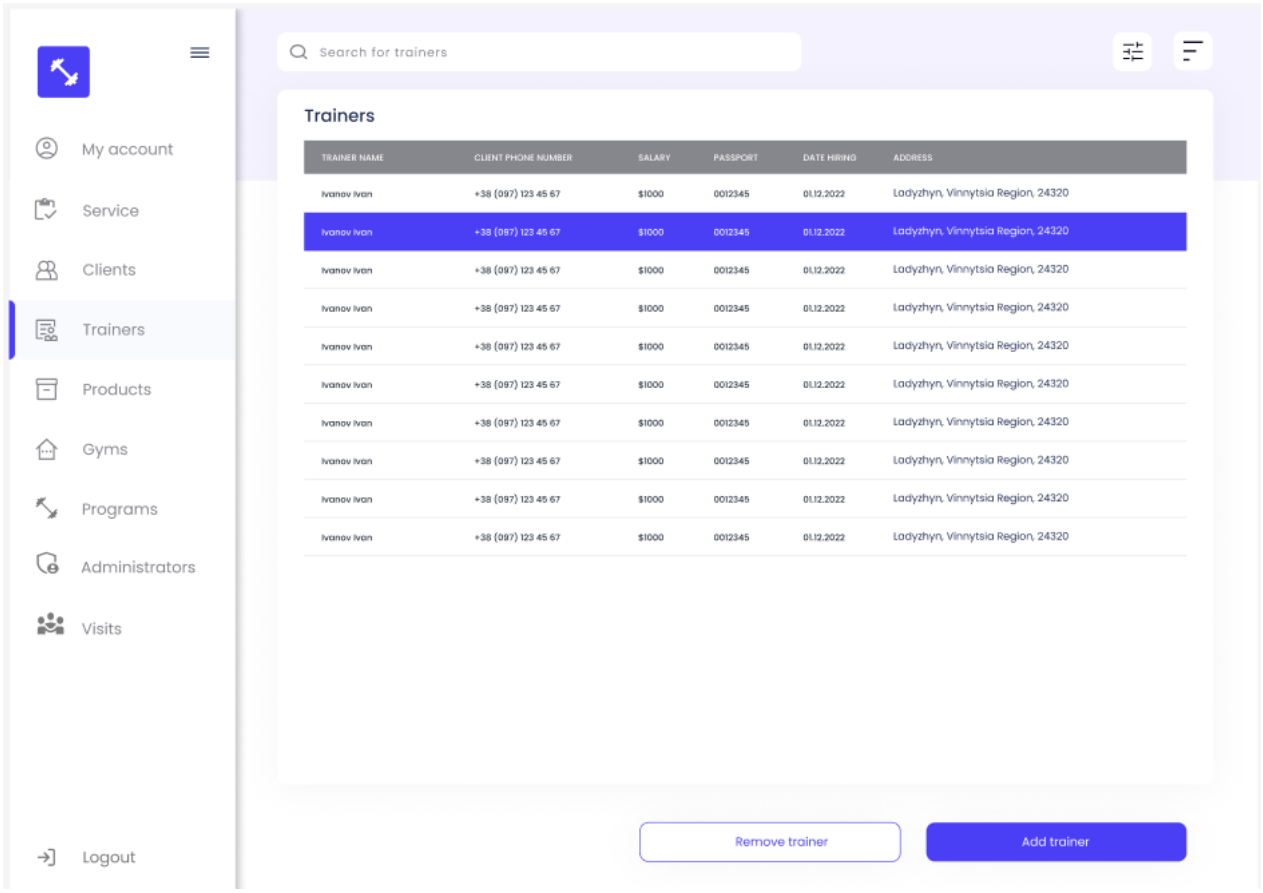

Рисунок 7.12 – Вікно пункту меню «Тренери»

При використанні даного пункту меню власником мережі тренажерних залів, буде відображено інформацію про тренерів всіх залів мережі з можливістю фільтрації по обраному залу.

7.2.7 Товари

Даний пункт меню також призначений лише для власників та адміністраторів тренажерного залу. Вікно товарів, що зображене на рисунку 7.13, дозволяє:

- переглядати наявні позиції товарів для продажу у поточному залі;
- редагувати інформацію про товар;
- додавати новий товар;
- видаляти товар;
- пошук за назвою;
- пошук за виробником.

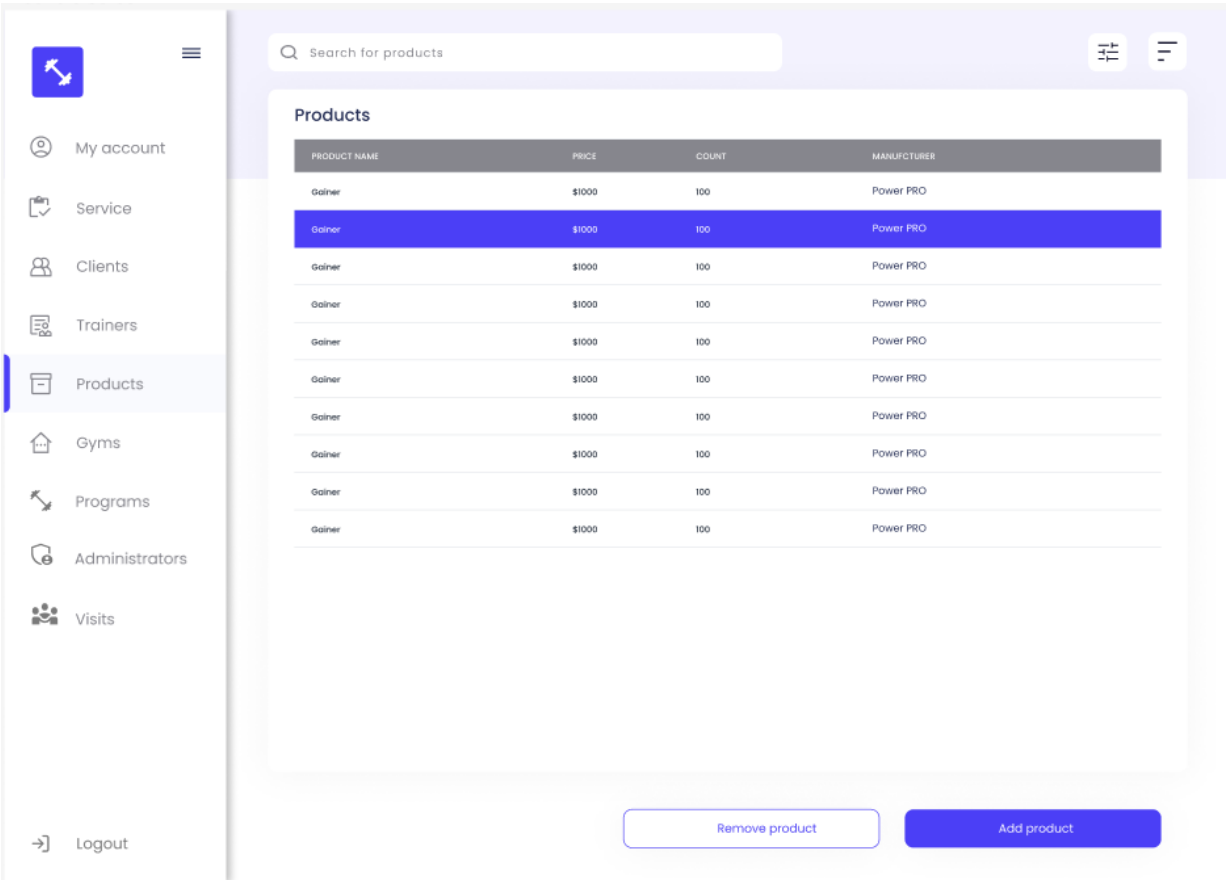

Рисунок 7.13 – Вікно пункту меню «Товари»

При використанні пункту меню власником мережі тренажерних залів, буде відображатися інформація про всі товари мережі з можливістю фільтрації по обраному залу.

7.2.8 Відвідування

Пункт меню відвідування, зображений на рисунку 7.14, призначений для використання адміністраторами та власниками тренажерних залів, та надає наступний функціонал:

- перегляд історії всіх відвідувань;
- додавання факту відвідування;
- видалення відвідування;
- фільтрація по клієнтам;
- фільтрація по датам відвідувань.

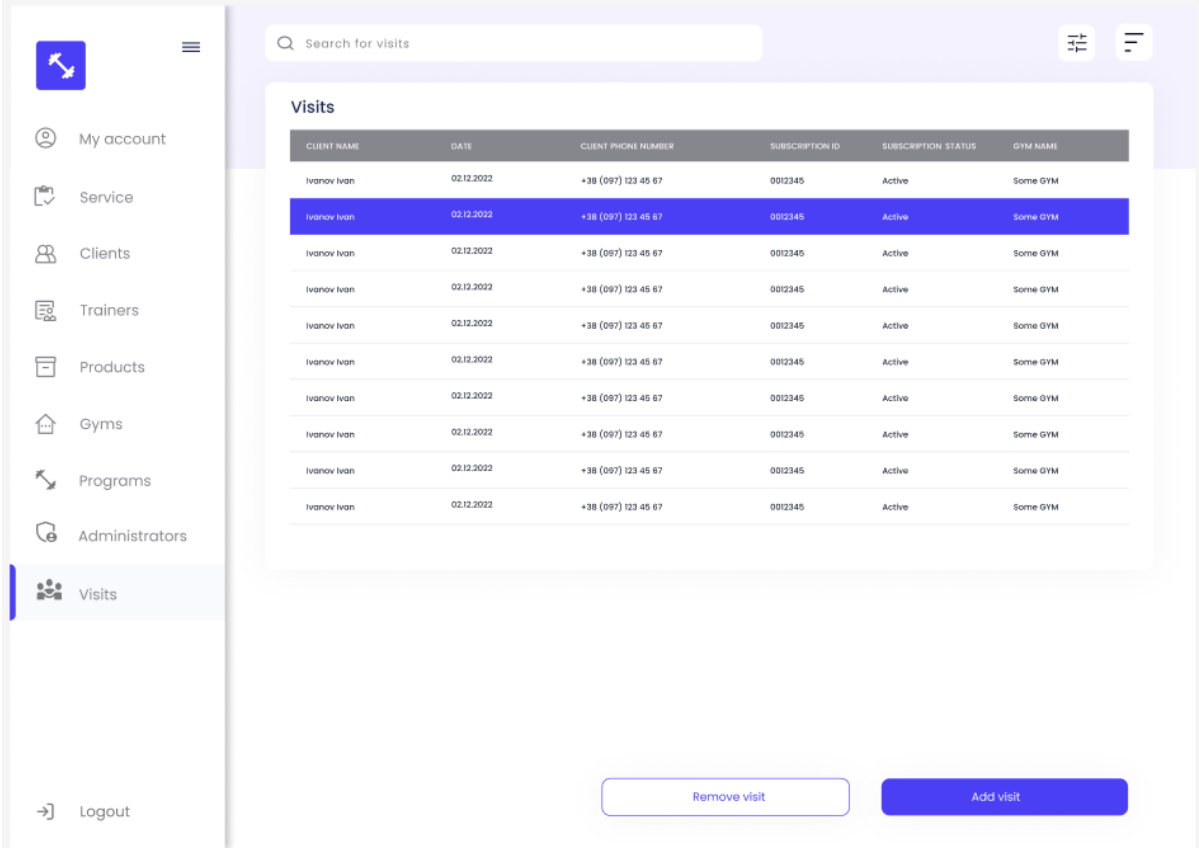

Рисунок 7.14 – Вікно пункту меню «Відвідування»

При використанні даного пункту меню власником мережі тренажерних залів, будуть відображення всі відвідування по всім залам мережі з можливістю їх сортування за останніми.

7.2.9 Адміністратори

Вікно «Адміністратори», що зображено на рисунку 7.15, призначено для користування лише власниками спортивних залів. Воно надає доступ до наступного функціоналу:

- зведена інформація про адміністраторів всіх клубів;
- редагування особистої інформації адміністратора;
- редагування окладу адміністратора;
- додавання адміністратора;
- видалення адміністратора.

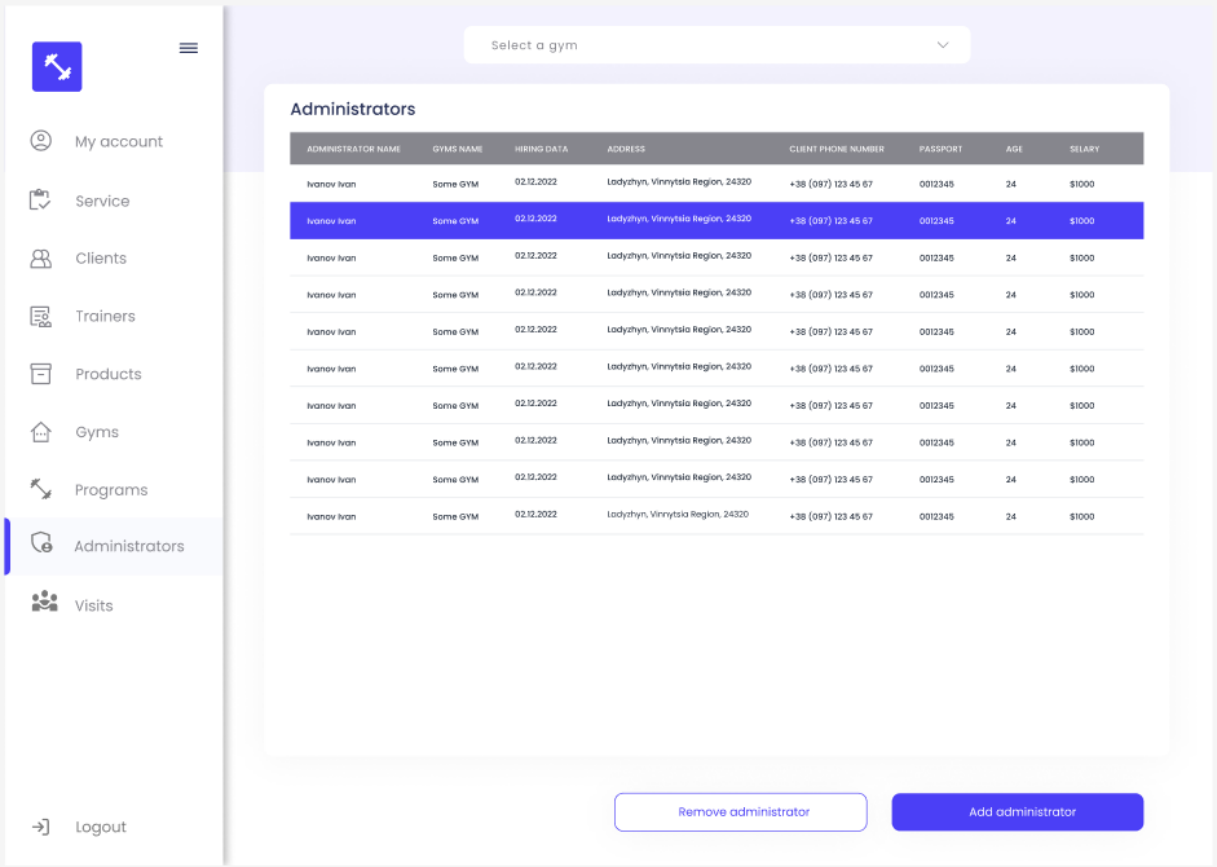

Рисунок 7.15 – Вікно пункту меню «Адміністратори»

Також на сторінці присутня можливість фільтрації адміністраторів по залу, де вони влаштовані.

7.2.10 Зали

Даний пункт меню (рис.7.16) доступний лише для власників та призначений для:

- перегляду зведеної інформації про зали;
- додавання залу;
- видалення залу;
- редагування назви;
- пошук по назві;
- пошук по адресі;
- редагування адреси залу.

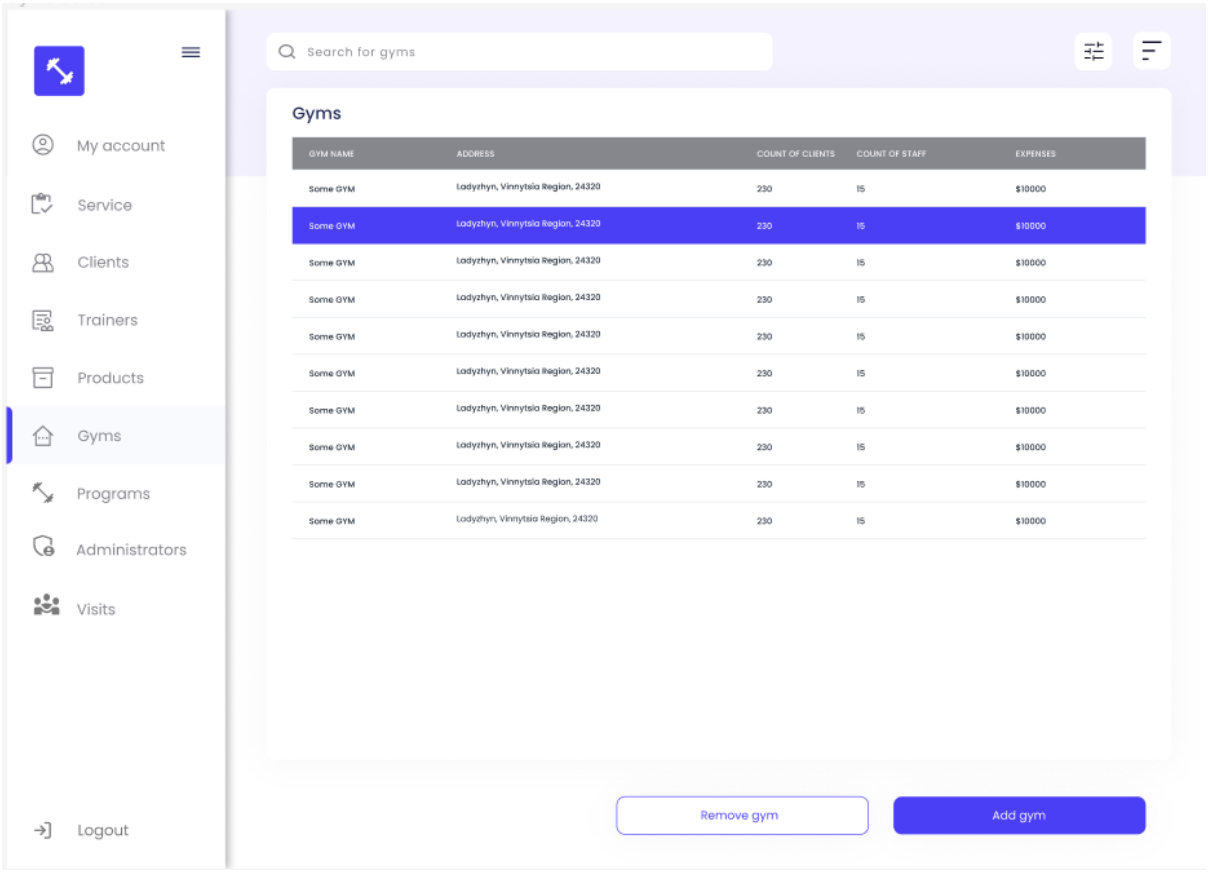

Рисунок 7.16 – Вікно пункту меню «Зали»

Також варто відмітити, що серед інформації про зал присутнє поле витрат, що буде відображати місячні витрати на зал, що будуть включати в себе зарплату всіх співробітників.

7.2.11 Вправи

Пункт меню «Вправи», що зображено на рисунку 7.17, призначений для користування тренерами, але може бути доступним також для власників та адміністраторів тренажерних залів, за потреби. Він призначений для виконання наступних операцій:

- перегляд списку вправ бази знань;
- додавання вправи в базу;
- видалення вправи;
- редагування вправи.

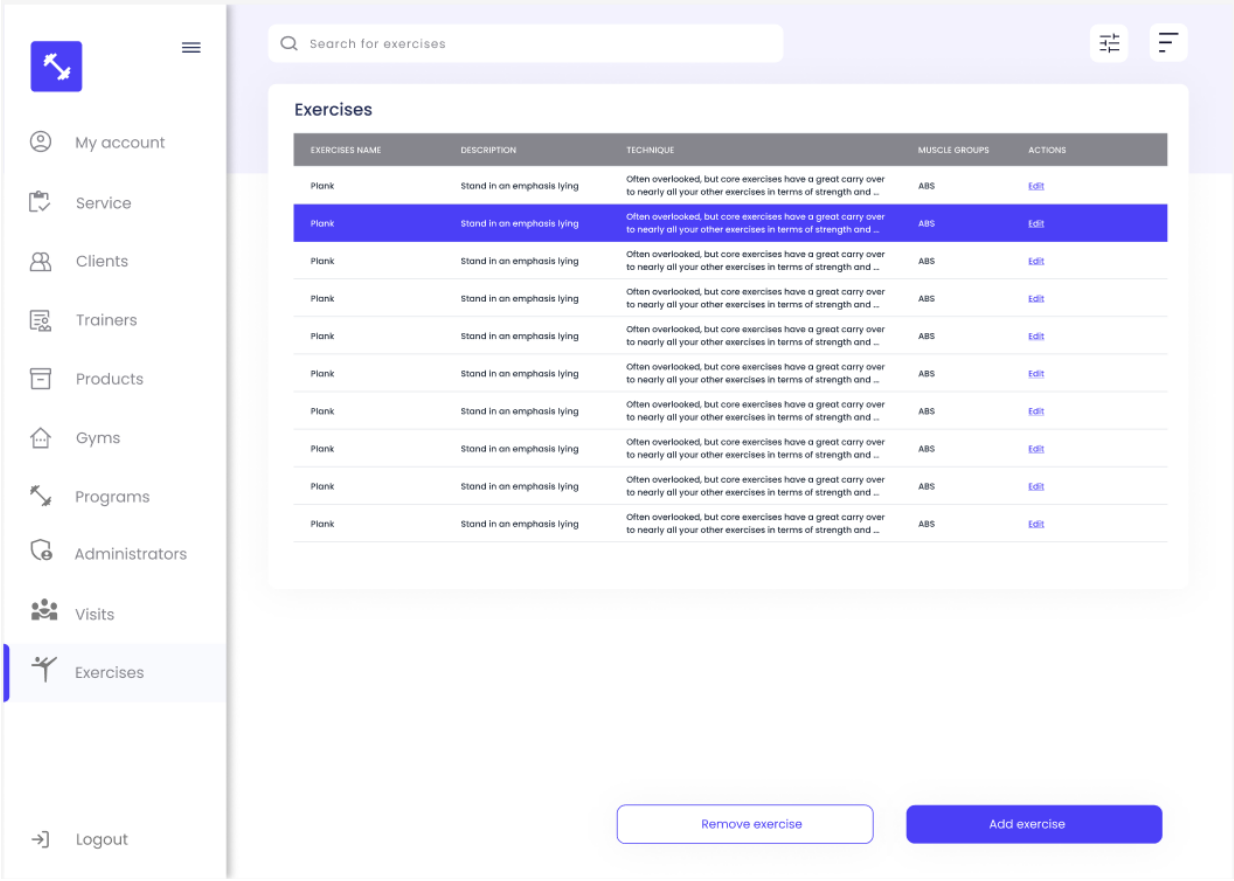

Рисунок 7.17 – Вікно пункту меню «Вправи»

Також присутня можливість фільтрування вправ за групами м'язів, що задіянні при виконанні вправи.

7.2.12 Створення запису

Вікно створення запису, що зображено на рисунку 7.18, є загальним для всіх ролей. Дане вікно є спливаючим та загальним для різних випадків. Основною його задачою є створення записів у таблиці.

Для додавання елементів, дане вікно містить різний набір полів для введення, що відображається відповідно до сутності, що необхідно створити. Після того, як всі необхідні поля були заповнені, необхідно натиснути на кнопку підтвердження внизу вікна, після чого інформація запишеться у відповідну таблицю та вікно автоматично зникне.

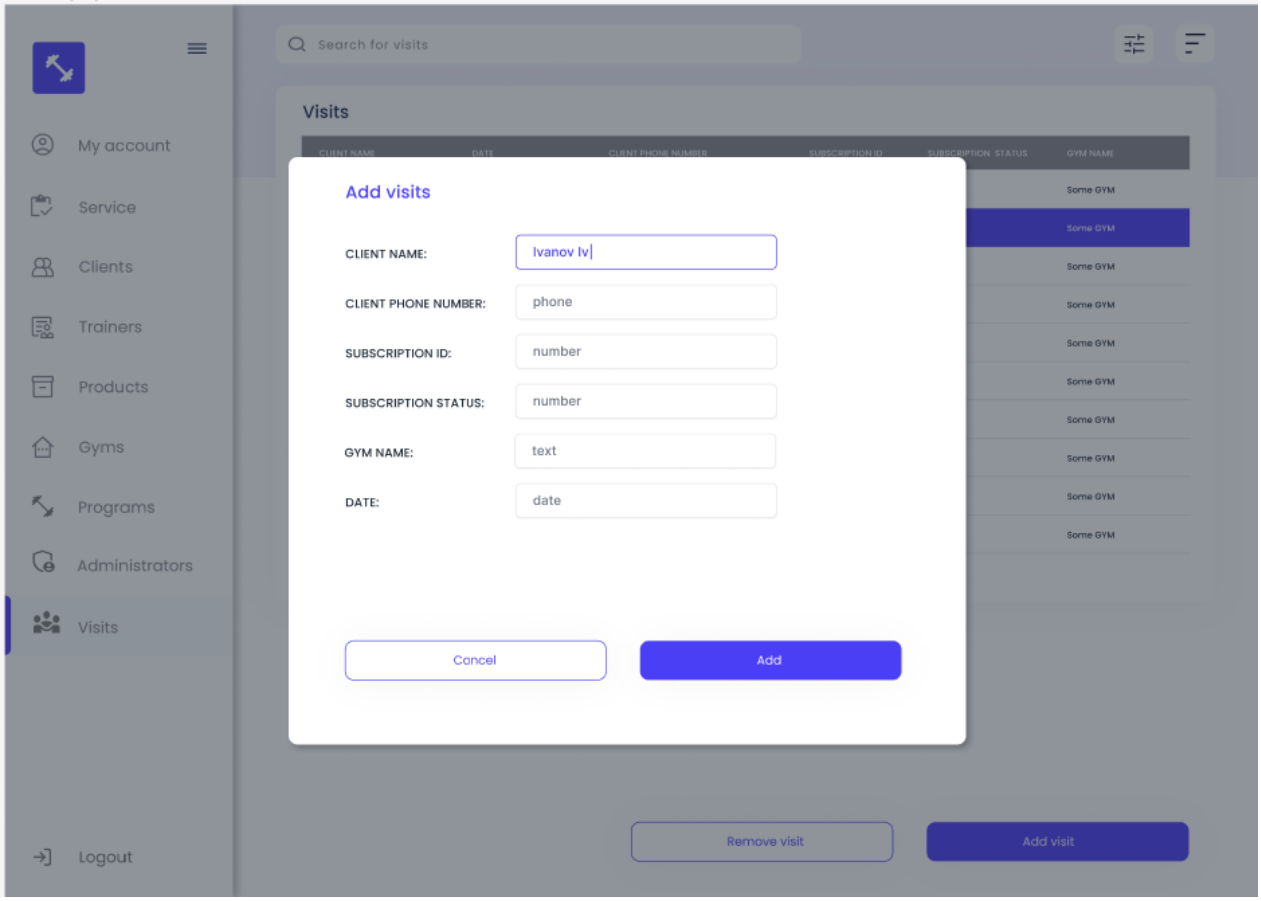

Рисунок 7.18 – Вікно створення запису

Також варто відзначити, що для відображення кожного пункту меню буде створений привілей, що дасть змогу в майбутньому з легкістю впроваджувати нові ролі в систему з будь-яким набором привілей.

В даному розділі було розроблено два типи інтерфейсів мобільний та комп’ютерний. Комп’ютерний інтерфейс спроектовано таким чином, щоб максимально ефективно використати ресурс на розробку, збільшити повторне використання елементів та забезпечити можливість до розширення. Розроблені інтерфейси забезпечують високі стандарти UI/UX дизайну та повністю надають функціонал, визначений вимогами.

8 ПРОГРАМНА РЕАЛІЗАЦІЯ

8.1 Загальні принципи та архітектура

Зважаючи на обрані технології, структуру схему, аналіз вимог до системи, розроблений інтерфейс користувача та спроектовану базу даних, необхідно обрати загальну архітектуру системи.

Розрізняють два основні типи архітектури:

- мікросервісна;
- монолітна.

Незважаючи на те, що мікросервісна архітектура є більш сучасною та дедалі більше набуває популярності через свою легкість до масштабування, для розробки було обрано саме монолітну архітектуру.[15] Таке рішення зумовлено меншою вартістю за розміщення на серверах, зменшенням часу та витрат на розробку та покращенням швидкодії системи.

Оскільки мова C# є об’єктно-орієнтованою, то при написані коду були дотримані такі основні принципи:

- перевага композиції перед наслідуванням;
- мінімальна зв’язність коду;
- мінімум повторюваності коду;
- повторне використання коду;
- KISS;
- DRY;
- SOLID.

Дотримання даних принципів допоможе зберегти чистоту коду, а також дозволить з легкістю масштабувати, додавати нові можливості в майбутньому, навіть при зміні розробників. Також дані принципи підвищать ефективність розробки, адже застережуть від багатьох помилок.

Окрім вище перерахованих принципів та правил було також використано паттерн «Repository», власну реалізацію якого було описано в наступному підрозділі даного розділу.

8.2 Шаблон «Repository»

Даний паттерн було застосовано на рівні доступу до даних, що відповідає за зв’язок керуючої програми з СУБ. Суть даного шаблону полягає в тому, щоб виділити загальну логіку для обробки даних та зробити її незалежною від способу зберігання.[16] На рисунку 8.2 наведена UML-діаграма використання паттерну «Repository».

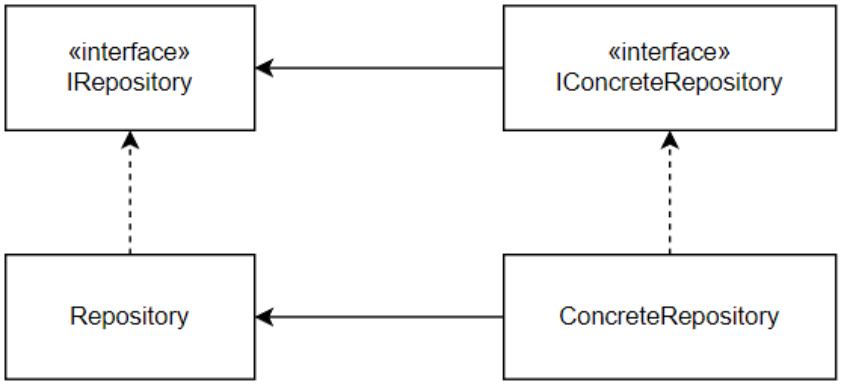

Рисунок 8.1 – Загальний варіант реалізації шаблону «Repository»

Така реалізація включає в себе виділення певного інтерфейсу «IRepository», наслідування від нього інтерфейсу «IConcreteRepository» і вже від них для кожної таблиці в бд створюється класс наслідник з реалізацією необхідної логіки.

При написанні програми було використано власну реалізацію даного шаблону, що зображено на рисунку 8.2.

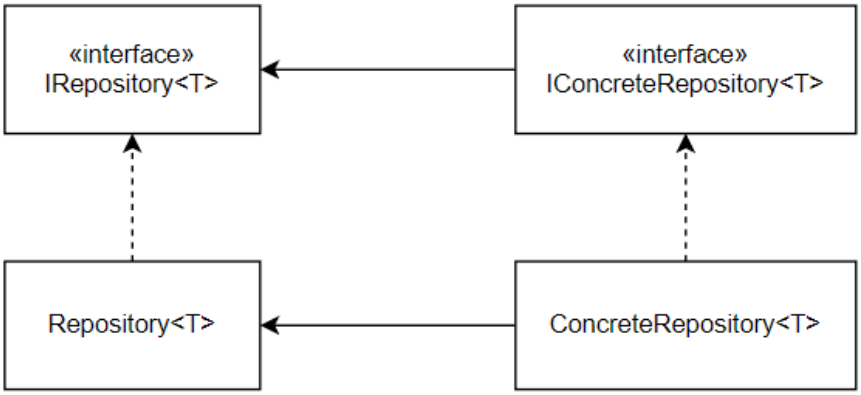

Рисунок 8.2 – Власна реалізація шаблону «Repository»

В даній реалізації використано одну з особливостей мови C#, а саме Generic Type. Даний тип даних є узагальненням та може набувати значення будь-якого іншого типу. Така реалізація даного шаблону допоможе скорити кількість написаного коду та збільшить ефективність розробки. В якості узагальнень в даному шаблоні були використані класи, що відображають конкретні таблиці бази даних.

Для того, аби така реалізація була можливою було:

- визначено загальні поля для всіх сутностей;
- приведено до одного типу первинні ключі сутностей;
- розроблено інтерфейси для загальних полів сутностей.

На рисунку 8.3 приведена реалізація інтерфейсу IRepository<T>.

```
public interface IRepository<T> where T : class, IEntity, new()
{
    Ссылок: 5
    IQueryable<T> Items { get; }

    Ссылок: 2
    T GetById(Guid id);
    Ссылок: 1
    Task<T> GetByIdAsync(Guid id, CancellationToken cancel = default);

    Ссылок: 1
    T Add(T item);
    Ссылок: 1
    Task<T> AddAsync(T item, CancellationToken cancel = default);

    Ссылок: 1
    void Update(T item);
    Ссылок: 1
    Task UpdateAsync(T item, CancellationToken cancel = default);

    Ссылок: 1
    void RemoveById(Guid Id);
    Ссылок: 1
    Task RemoveByIdAsyn(Guid Id, CancellationToken cancel = default);
}
```

Рисунок 8.3 – Реалізація інтерфейсу IRepository<T>

На рис. 8.3 видно, що узагальнений тип даних представлено інтерфейсом IEntity, що містить в собі загальне поле для всіх сутностей – id.

Також дана реалізація дозволяє активно та з легкістю використовувати вбудований контейнер впровадження залежностей (DI) та методи розширення.

8.1 DI контейнер

З огляду на те, що було визначено, що під час розробки керуючої програми необхідно забезпечувати мінімальну зв’язність коду, було прийнято рішення про використання принципу IoC. IoC (англ. Inversion of Control) – абстрактний принцип, що містить в собі набір рекомендацій для написання слабо-зв’язного коду. Його суть полягає в тому, що кожен компонент системи, будь то сервіс чи blazor компонент, необхідно максимально ізолювати від інших, не беручи до уваги деталі конкретної реалізації інших компонентів.[17]

В якості реалізації даного принципу було застосовано DI – впровадження залежностей. Використання такого підходу надає ряд наступних переваг [18]:

- полегшення процесу рефакторингу;
- можливість повторного використання коду;
- легкість тестування.

Реалізувати такий підхід можна за допомогою декількох способів, а саме:

- constructor injection;
- field injection.

При розробці даної інформаційної системи було вирішено використовувати саме варіант з впровадженням залежності через конструктор, оскільки даний підхід вирішує проблему виявлення замкнених залежностей та підтримується вже вбудованим інструментом в ASP.NET Core.

Тож обраний DI Container має на меті створення певного контейнеру, який є частиною системи та зберігає в собі всі необхідні сервіси та їх залежності, та надає їх у ті місця, де вони необхідні.

Для такої реалізації, в ASP.NET Core існує вбудований контейнер сервісів, що знаходиться в Microsoft.AspNetCore.Builder.WebApplicationBuilder у вигляді інтерфейсу IServiceCollection, який і є набором об’єктів що містять опис сервісів. За допомогою метода Add..(), у даної колекції, до неї може бути додано будь-який сервіс.

Серед доступних методів:

- AddSingleton();
- AddTransient();
- AddScoped().

Кожен з даних методів додає сервіс до колекції, але з певним налаштуванням часу на існування даного сервісу під час виконання програми, так звана «область застосування».

Так AddSingleton() забезпечить створення одного екземпляру класу необхідного сервісу, під час його виклику, та продовжить його використання у інших місцях коду, допоки програма не закінчить свою роботу. Такий спосіб дає можливість зменшити кількість пам’яті, що необхідна для роботи програми, але в той же час необхідно бути певним, що в усіх інших частинах програми, де використовується даний сервіс, інформація повинна бути однаковою, що є не дуже гарною практикою при розробці багатокористувацьких додатків.[19]

AddTransient() створює новий екземпляр об’єкту сервісу кожен раз, коли він необхідний у програмі. Даний значно збільшує навантаження на серверну частину за рахунок збільшення оперативної пам’яті та частих створень та видалень об’єктів з неї. Через це його доцільно використовувати лише у випадку сервісів, що мають короткий життєвий цикл доступу до зовнішніх ресурсів.

AddScoped() передбачає створення деякої «області застосування», яка може бути як вбудованою так і визначеною розробником, впродовж якою контейнер буде видавати одну й ту саму сутність об’єкту, але для іншої такої області, це вже будуть інші об’єкти. Даний підхід чудово підходить для веб-додатків з багатокористувацьким доступом.

Під час розробки, як було сказано в попередньому підрозділі, було обрано паттерн «Repository» в якості інструменту для роботи з базою даних. Було створено

класи репозиторіїв для необхідних таблиць даних, для роботи з ними. Після чого, для їх подальшого використання, було додано їх у DI контейнер у класі Program.cs. Для того, аби ще більше зменшити зв'язність коду, кожен репозиторій було винесено у інтерфейс, та додано до контейнеру в парі з ним. Це дозволить змінювати реалізацію сервісу, коли це буде необхідним без зміни реалізації додавання сервісу до контейнеру, тому забезпечить гнучкість системи та її легку масштабованість.

При додаванні сервісів репозиторіїв до контейнеру залежностей в клієнтському та адміністративному додатку було використано саме методи AddScoped у колекції сервісів, адже при такій реалізації в Blazor Server проекті, «область застосування» створюється для так званого «циклу». Циклом вважається момент від відкриття користувачем сторінки до закриття її вкладки в браузері, що по суті є сесією користувача.

Також для реєстрації всіх необхідних додатку сервісів у кожному з них було створено метод RegisterRepositories(), що приймає в якості аргументу колекцію сервісів, та потім додає до неї всі необхідні. Даний метод викликається в файлі Program.cs. Приклад такого додавання наведено на рисунку 8.4.

```
public static IServiceCollection RegisterRepositories(this IServiceCollection services) => services
    .AddScoped<IRepository<Client>, ClientsRepository>()
    .AddScoped<IRepository<Employee>, DbRepository<Employee>>()
```

Рисунок 8.4 – Додавання репозиторіїв до колекції сервісів

Після того, як всі сервіси були додані до DI контейнеру додатку, в будь-якому компоненті чи сторінці програми, де це необхідно, вони можуть бути викликані за допомогою директиви @inject на початку сторінки (рис. 8.5).

```
@page "/customer-list"
@inject IDataAccess DataRepository
```

Рисунок 8.5 – Приклад використання директиви @inject

Таким чином даний підхід зменшив кількість повторюваного коду у двох додатках, та забезпечив кожен додаток необхідним переліком сервісів.

8.3 Компоненти

Після розробки рівня доступу до даних, було розроблено користувацький інтерфейс, згідно до визначеного дизайну. Оскільки в якості фреймворку для розробки було обрано Blazor, а основним будівельним його елементом є компонент, з яких складаються сторінки, то першим кроком при його розробці потрібно визначитись з переліком таких компонентів.

Компонент – самодостатній елемент користувацького інтерфейсу, що описує певну візуальну частину та пов’язану з нею логіку та має розширення файлу .razor.[21]

Для функціонування він містить наступні основні елементи:

- набір параметрів, що визначають можливість передачі даних від батьківського елементу до дочірнього;
- набір атрибутів, що передаються через словник за допомогою директиви @attributes, та потім використовуються під час рендеренгу елементу;
- обробники подій, що мають такий ж набір подій, як в звичайному HTML елементі, але визначаються за допомогою @on… та дозволяють прикріпити до елемента метод компонента;
- директива @page, що надає компоненту шаблон маршруту, що потім використовується при запиті по визначеній URL-адресі.

Саме завдяки подіям в додатку відбувається вся навігація та в цілому взаємодія користувача з системою.

Тож компонент складається з HTML коду, CSS та C#, що дозволяє йому виконувати всі необхідні функції. На рисунку 8.6 зображено вміст файлу такого razor компоненту, що визначає поведінку підсвітлюваного тексту та містить в собі всі вищеописані елементи.

```
@inherits ComponentBase

<div class="breadcrumb-group">
    <ol class="flex">
        @foreach (var breakcrumb in BreakCrumbs)
        {
            var isLast = breakcrumb == BreakCrumbs.Last();

            <li class="breadcrumb-item flex">
                <div class="breadcrumb flex center-vertical">
                    @if (breakcrumb.Event is not null)
                    {
                        <a @onclick="breakcrumb.Event" class="breadcrumb-link@(isLast ? " unactive" : string.Empty)">@breakcrumb.Title</a>
                    }
                    else
                    {
                        <a href="@breakcrumb.Url" class="breadcrumb-link@(isLast ? " unactive" : string.Empty)">@breakcrumb.Title</a>
                    }

                    @if (!isLast)
                    {
                        <i class="material-icons">@Constants.Icons.BreakCrumbDelimiter</i>
                    }
                </div>
            </li>
        }
    </ol>
</div>

<style>
    .breadcrumb-group ol {
        margin: 0;
        padding: 0;
    }
    .breadcrumb-item i {
        margin: 0 0.5rem;
    }
    .breadcrumb-item a.unactive,
    .breadcrumb-item i {
        color: var(--color-text-form-description);
        pointer-events: none;
    }
</style>

@code
{
    [Parameter]
    public List<BreakCrumb> BreakCrumbs { get; set; } = new();
}
```

Рисунок 8.6 – Код .blazor компоненту

Тож, з огляду на розроблений інтерфейс користувача та обрані технології, було визначено переліки та розроблено основні компоненти для клієнтського та адміністративного додатків.

Для клієнтського додатку це:

- шаблон головної сторінки;
- шаблон сторінки авторизації/реєстрації;
- кнопки навігаційного меню;
- кнопки навігації сторінки «Профіль»;
- текстове поле для вводу для пошуку.

Для адміністративного додатку:

- шаблон вікна авторизації;
- шаблон головного вікна;
- шаблон вікна додавання/правки елементу;
- бічне меню;

- шапка кожного вікна з пошуковим полем для введення та сортування;
- таблиця для відображення даних;
- кнопки дій.

Окрім цих було також розроблено різні допоміжні компоненти, які не є ключовими і тому не були описані. Таким чином було максимально зменшено повторюваність коду та підвищено ефективність розробки.

8.4 Хмарні сервіси

При розробці архітектури та виборі технологій було визначено, що дана інформаційна система буде знаходитись у хмарному середовищі, а в якості постачальника таких послуг було обрано компанію Amazon з її продуктом – AWS. Даний продукт містить в собі багато різноманітних сервісів, що значно полегшують розміщення рішення на сервері та покращують його роботу. Розробляючи дану інформаційну систему, було використано наступний перелік сервісів:

- Amazon RDS;
- Amazon EC2;
- Amazon ELB;
- Amazon Elastic Beanstalk;
- Amazon SES;
- Amazon Route53.

За налаштування, публікацію чи підтримку кожного з цих сервісів немає необхідності сплачувати, оскільки оплата нараховується лише за використовувані ресурси. Далі в підрозділі було розглянуто та описано кожен з них.

8.4.1 Amazon RDS

Amazon RDS (англ. Relational Database Service) – веб-сервіс, що дозволяє користувачам створювати, керувати, запускати та масштабувати реляційні бази даних, водночас з цим вона бере на себе керування задачами адміністрування БД. [20]

Станом на час написання цієї дисертації, даний сервіс підтримує СУБД всіх відомих постачальників, в тому числі PostgreSQL, що цілком задовольняє вимогам.

Також серед переваг, які надає використання Amazon RDS варто виділити те, що він здатен автоматично створювати резервну копію БД та підтримувати стан її ПО в актуальному стані. [20] Окрім цього він також спрощує реплікацію, що підвищує доступність та надійність БД, а також забезпечує масштабування ресурсів у випадку великої кількості операцій читання.

8.4.2 Amazon EC2

Amazon EC2 (англ. Elastic Compute) – сервіс, що надає обчислювальні потужності в хмарі. Відбувається це шляхом оренди користувачем віртуальної машини, що називається екземпляром, та запускаються вони за допомогою завчасно сконфігурованих образів AMI (англ. Amazon Machine Image), забезпечуючи таким чином менший час завантаження нового серверу.

Amazon EC2 надає змогу розміщувати екземпляри в декількох місцях розташування, що звуться локаціями. Дані локації складаються із зон доступності та регіонів, де перші – місця розташування, що спроектовані таким чином, щоб бути захищеними від збоїв в інших зонах та забезпечувати мінімальні затримки. Таким чином така система надає змогу розміщувати екземпляри в різних зонах доступності, захищаючи їх від збоїв в одній локації.

Також даний сервіс містить в собі великий спектр доступних функцій, але варто звернути уваги на наступні:

- безмежне масштабування ресурсів, як в більшу так і в меншу сторону;
- оптимізація продуктивності та вартості обчислювальних ресурсів;
- налаштування з оптимізацією центрального процесора;
- перехід екземплярів у сплячий режим;
- захист інформації, що необхідна для входу в екземпляри, за допомогою пари ключів.

8.4.3 Amazon ELB

Amazon ELB (англ. Elastic Load Balancing) – веб-сервіс балансування навантаження. Він відповідає за автоматичне розподілення вхідного трафіку між запущеними екземплярами, про які було згадано в попередньому пункті.

Існує чотири можливі варіанти використання даного сервісу:

- балансувальник навантаження програм (Application Load Balancer);
- балансувальник мережевого навантаження (Network Load Balancer);
- балансувальник навантаження шлюзу (Gateway Load Balancer);
- класичний балансувальник навантаження (Classic Load Balancer).

Перший тип балансувальника використовується, коли необхідно мати гнучкий перелік функцій для додатків із трафіком HTTP та HTTPS. Як правило, вони націлені на програми з мікросервісною архітектурою та контейнерами.

Балансер мережевого навантаження призначений для випадків, коли небхідна висока продуктивність, підтримка UDP та статичні IPv4-адреси для додатків.

Балансер навантаження шлюзу зазвичай використовують, коли необхідно розгорнути набір віртуальних машин від сторонніх розробників та керувати ними.

В якості балансера навантаження розроблюваної інформаційної системи було обрано саме перший варіант – балансер навантаження програми, адже він підтримує технологію веб-сокетів, та працює з необхідними протоколами HTTP/S. В майбутньому, за потреби, може бути використано й інші типи.

8.4.4 Amazon Elastic Beanstalk

Amazon Elastic Beanstalk – сервіс, що призначений для розгортання та масштабування веб-додатків та сервісів. Він дозволяє розгортати додатки без вивчення інфраструктури, що необхідна для їх запуску.

Завдяки даному сервісу код буде просто завантажено на сервер, після чого Elastic Beanstalk виконає всю роботу по виділенню необхідних ресурсів, наданню

сховища даних, балансуванню навантаження, моніторингу та масштабуванню. Також доступна можливість безпечного оновлення версії свого продукту.

Для розгортання, він використовує вище описані сервіси, та підтримує технологію розробки .NET. Також після розгортання, в CLI AWS буде доступною вся інформація про програму та її показники.

Зважаючи на описані переваги використання Elastic Beanstalk, його було обрано в якості інструменту для зручного та швидкого розгортанню клієнтського та адміністративного додатків на сервері.

8.4.5 Amazon SES

Amazon SES (англ. Simple Email Service) – веб-сервіс, що дозволяє спілкуватись з клієнтами конфіденційно та без використання системи Simple Mail Transfer Protocol. Він може бути впроваджений в будь-який додаток, щоб використовувати можливості масового розсилання листів електронною поштою. Даний сервіс підтримує різні варіанти розгортання, звіти по статистиці відправників та панель інформації про кількість саме доставлених листів.

Використовуючи даний сервіс, було уникнуто проблем з управління поштовим сервером та конфігурацією мережі електронної пошти. Окрім цього, перевагою даного рішення є те, що вартість використання даного сервісу не змінюється в залежності від того чи є надіслані листи трансакційними чи маркетинговими.

Amazon SES було використано в клієнтських додатках для надсилання електронних листів з заявкою на запис на тренування тренерам та в процесі реєстрації новим клієнтам.

8.4.6 Amazon Route 53

Amazon Route53 – це масштабований та високошвидкісний сервіс з надання послуг DNS (англ. Domain Name System), що надає такі основні функції:

- реєстрація домену;

- маршрутизація;
- перевірка справності з’єднання.

Також серед важливих можливостей Amazon Route 53 слід виділити:

- захист від рекурсивних запитів DNS;
- перевірка готовності;
- керування маршрутизацією;
- керування потоками трафіку;
- перекидання сервісу DNS, що для уникнення перенавантаження сайту направить користувачів до альтернативного розташування.

Таким чином даний сервіс є одним з найважливіших, що були використані при розробці, адже саме завдяки йому було створено глобальний домен інформаційної системи для адміністративного та клієнтського додатків.

У Додатку Ж наведено схему взаємодії обраних хмарних сервісів.

8.5 Реєстрація клієнта

Оскільки у вимогах до системи було вказано, що користувач повинен мати змогу самостійно зареєструватися, то було розроблено власну логіку реєстрації. Адміністратор, при видачі клієнту придбаного абонементу вносить в систему дані про його паспорт, номер абонементу та пошту, після чого на неї надсилається повідомлення засобами сервісу Amazon SES, з посиланням на клієнтський додаток. Після відкриття клієнтського сайту, користувачеві необхідно ввести номер свого паспорту та абонементу, щоб система перевірила дану комбінацію на наявність. Після успішної перевірки клієнта допущено до системи та зареєстровано з вказаним паролем. Детальніше даний операцію зображено у вигляді діаграми послідовності у Додатку И.

В результаті виконання даного розділу було розроблено адміністративний та клієнтський додатки інформаційної системи, з дотриманням поставлених функціональних та нефункціональних вимог. Також було розміщено дані додатки у хмарному середовищі з використанням різних сервісів хмарного постачальника.

9 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

Стартап-проект – це нові ідеї, які можуть опиратися на вже існуючі, але тим чи іншим чином їх покращувати, або ж зовсім принести щось нове на ринок. Основною задачею такого проекту є збір коштів, для його реалізації. В цьому допомагає залучання інвесторів, або ж кредити в банківських установах, але що в першому, що в другому випадках необхідно спиратись на ринковий аналіз конкурентів та комерційну вигоду проекту.

Метою даного розділу є маркетинговий аналіз стартап-проекту, що буде характеризувати можливість впровадження його на ринку, а також напрямки можливих реалізацій цього впровадження.

9.1 Опис ідеї проекту

Першим кроком був опис ідеї стартап-проекту, що дасть цілісне уявлення про її зміст та потенційні ринки, на яких слід шукати потенційних клієнтів наведені у таблиці 9.1.

Таблиця 9.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Програмний комплекс для підтримки діяльності тренажерного залу це проста та зручна у використанні система для контролю основних параметрів спортивних залів або їх мереж(тренери, адміністратори, клієнти). Також дозволяє вести облік товарів та тренажерів.	1. Малі тренажерні зали.	Покращення зручності надання існуючих послуг клієнтам, та розширення їх кількості.
	2. Великі тренажерні зали.	Підвищення якості надання послуг та зручності спілкування з клієнтами

Продовження таблиці 9.1

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Також їх обслуговування. Клієнтська частина системи дозволить покращити взаємодію між тренером та клієнтом для досягнення цілей, а також надасть останньому інформацію щодо відвідувань та статусу абонементу. Також передбачена база знань з переліком вправ, що доступна для будь-якого клієнта.	3. Мережі тренажерних залів	Збільшення ефективності надання послуг та зручності ведення адміністративних функцій

Даний опис розкрив зміст ідеї, надав чітке уявлення про потенційні ринки збуту інформаційної системи, та сформулював вигоди для користувачів.

Після опису ідеї стартап-проекту, було проведено аналіз потенційних техніко-економічних переваг порівняно із існуючими пропозиціями на ринку, дані якого наведені у таблиці 9.2.

Таблиця 9.2 – Визначення характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Gra Fit	Sport Life	Lucky Fit			
1.	Ролі користувачів в системі	+	-	+	+			+
2.	Профіль клієнта	+	+	+	+		-	

Продовження таблиці 9.2

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Gra Fit	Sport Life	Lucky Fit			
3.	Профіль співробітника	+	-	+	+			+
4.	База співробітників	+	-	+	+		-	
5.	Використання для декількох залів	+	+	+	+			+
6.	Можливість доступу з телефону	+	+	+	+			+
7.	Можливість доступу з ПК	+	-	+	+			+
8.	Можливість доступу без встановлення	+	-	-	-			+
9.	Наявність товарів	+	-	-	+			+
10.	Облік тренажерів	+	-	+	-		-	
11.	База знань з вправами	+	-	-	-			+
12.	Розклад тренувань	+	+	+	+	-		
14.	Зручність інтерфейсу	+	-	+	-		+	

9.2 Технологічний аудит ідеї проекту

В даному підрозділі було розглянуто технологічну здійсненність ідеї проекту. Дані цього аудиту наведені у таблиці 9.3.

Таблиця 9.3 – Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Авторизація користувача	Власна реалізація	Наявна	Доступна безкоштовно
2.	Реалізація логіки та інтерфейсу	Blazor, .NET, HTML, CSS, JS	Наявна	Доступна безкоштовно
3.	Зберігання даних	СУБД PostgreSQL	Наявна	Доступна безкоштовно
4.	Взаємодія з базою даних	Entity Framework	Наявна	Доступна безкоштовно
5.	Розміщення у хмарному середовищі	Amazon Web Services	Наявна	Доступна безкоштовно для невеликих потужностей, інакше - платно
Обрана технологія реалізації ідеї проекту: є цілком можливою				

В результаті проведеного дослідження було визначено, що кожна з необхідних технологій є доступною, а отже реалізація ідеї проекту є можливою.

9.3 Аналіз ринкових можливостей запуску стартап-проекту

Наступним кроком було визначення можливостей, що наявні на ринку, аби використати їх під час впровадження проекту.

Також було визначено ринкові загрози, що можуть завадити реалізації ідеї проекту.

Першим кроком було визначено чи присутній попит на ринку, його обсяг та динаміка розвитку. Ці дані наведено у таблиці 9.4.

Таблиця 9.4 – Попередня характеристика потенційного ринку

№ п/п	Показники стану ринку (найменування)	Характеристика
№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	2 (розробники, мережі спортивних залів)
2	Загальний обсяг продаж, грн/ум.од	Вимірюється мільйонами гривень на рік, але точну суму визначити неможливо через закритість даних.
3	Динаміка ринку (якісна оцінка)	Зростає, з поширенням тенденції автоматизації всіх процесів та тренду здорового способу життя.
4	Наявність обмежень для входу (вказати характер обмежень)	Для повної реалізації проекту необхідно юридично зареєструвати підприємницьку діяльність, а також укласти договір з залом-клієнтом.
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні.
6	Середня норма рентабельності в галузі (або по ринку), %	60%

Середній відсоток вкладень в банки у гривні станом на сьогодні складає близько 7% на рік. Також на ринку присутня чимала інфляція через війну в країні.

Рентабельність даного проекту в рази вища за відсоток за вкладення, тому очевидно, що є сенс вкладати гроші в дану розробку. Зважаючи на невелику кількість конкурентів, та незайнятість ринку, за попереднім оцінюванням ринок є привабливим для входу.

Наступним етапом було визначення характеристик потенційних клієнтів проекту, що наведена у таблиці 9.5.

Таблиця 9.5 – Характеристика потенційних клієнтів

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Покращення ефективності роботи адміністраторів та тренерів клубів	Власники спортивних клубів та співробітники	Для власників важлива ціна та функціональність, а для адміністраторів та тренерів лише останнє. Потреба в різній функціональності. Повне придбання або формат підписки на використання.	Зручність використання. Легке обслуговування. Масштабованість. Широкий функціонал. Стабільна підтримка.
2	Потреба у повноцінній єдиній системі для клієнтів та співробітників	Власники спортивних клубів та співробітники	Для власників важлива ціна та функціональність, а для адміністраторів лише останнє. Потреба в різній функціональності.	Зручність використання. Легке обслуговування. Масштабованість. Широкий функціонал.

Продовження таблиці 9.5

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
2			Повне придбання або формат підписки на використання.	Стабільна підтримка.
3	Потреба у наявності актуальної інформації про зал та комунікації	Клієнти спортивних клубів	Безкоштовне користування без встановлення для підвищення зручності та якості обслуговування, при умові відвідання спортивного клубу.	Інтуїтивно зрозумілий інтерфейс. Безкоштовність. Набір необхідних функцій.

Після цього було проведено аналіз ринку на предмет факторів ризику та сприяння розвитку проекту (табл. 9.6-9.7).

Таблиця 9.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Законна реєстрація	Будь-яку економічну діяльність необхідно зареєструвати юридично.	Офіційно оформити стартап-проект задля співпраці з юридичними особами.

Продовження таблиці 9.6

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
2	Конкуренти	Поява конкурентів, що надають схожі можливості.	Розробка унікальних функцій. Програма лояльності.
3	Кошти на підтримку	Недостатнє фінансування довготривалої підтримки.	Залучення нових інвесторів. Постійне покращення системи. Збільшення ефективності роботи компанії
4	Стрімке розширення кількості користувачів	Різке велике збільшення клієнтів системи.	Впровадити тимчасове рішення, задля пристосування існуючого. Розробка нового проекту. Постійне покращення існуючого продукту.

Таблиця 9.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Збільшення лояльності клієнтів	При покращенні якості надаваних послуг, клієнти з більшою вірогідністю продовжуватимуть використання продукту.	Покращення існуючих функцій та впровадження існуючих бізнес-процесів в систему. Знаходження нових клієнтів.
2	Новий продукт	Збільшення кількості надаваних послуг. Нове рішення сфері.	Розробка нового функціоналу. Надання різних типів ліцензій замовникам.

Продовження таблиці 9.7

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
3	Покращення роботи штату компанії	З наявною системою штат зможе підвищити ефективність роботи, а отже зменшити витрати.	Скорочення зайвого персоналу, підвищення заробітних плат.
4	Реклама	При достатній популярності можливо ввести рекламну інтеграцію в систему задля комерціалізації системи.	Виділення додаткових коштів на розробку рекламної інтеграції. Реклама продукту.

Зважаючи на проведений аналіз, можна сказати, що основини загрозами є поява конкурентів на ринку. Компанія повинна бути готовою до всіх загроз та відповідно реагувати, що дозволить знаходитись на ринку. Можливості ринку переважають ризики, та вказують на наявність тенденції розвитку проекту.

Наступним кроком був аналіз пропозиції, дані якого що наведений у таблиці 9.8.

Таблиця 9.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції: олігополія	Існують недосконалі аналоги на ринку, без підтримки повного функціоналу	Залучення фахівців для проведення рекламних компаній. Створення унікальних функцій.

Продовження таблиці 9.8

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
2. Рівень конкурентної боротьби: міжнародний	Існуючі аналоги розроблялися різними країнами	Додавання різних мов в інтерфейс системи.
3. Галузева ознака: внутрішньогалузева	Даний ПЗ може використовуватися лише при побудові ІТ-систем.	Розвивати проект обраним шляхом.
4. Конкуренція за видами товарів: товарно-видова	Конкурують товари одного виду	Постійне покращення ПЗ. Впровадження підтримки. Розширення функціоналу.
5. Характер конкурентних переваг: не цінова	Доступні унікальні функції в системі. Більш широкий функціонал.	Впровадження нових, більш клієнто-орієнтованих, бізнес-процесів в систему.
6. За інтенсивністю: марочна	Наявність унікальних ознак, що відрізнятиме продукт.	Розробка комерційного дизайну та марки продукту.

Ступеневий аналіз конкуренції показав, що наявні на ринку аналоги мають схожий функціонал, схожі недоліки, та не сильно відрізняються один від одного, а отже вихід нового продукту може відібрати в них частину аудиторії.

Наступним кроком було проведено детальний аналіз умов конкуренції, що дозволить глибше зрозуміти стан конкурентного середовища. Даний аналіз наведений у таблиці 9.9.

Таблиця 9.9 – Аналіз конкуренції в галузі за М.Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Lucky Fit Appointer	Розробники програмного забезпечення. CRM-системи	Хмарні середовища	Спортивні зали різних розмірів	Існуючі системи.
Висновки:	Невелика кількість конкурентів, з частковою реалізацією функцій.	Потенційних конкурентів небагато, та в основному вони фокусуються на інших аспектах системи.	Є невелика залежність від постачальників, але тенденція на ринку показує, що вартість хмарних послуг буде знижуватись	Існують певні умови: швидкодія, доступність, стабільність роботи	Певні клієнти вже користуються певними системами, що може завадити переходу на нову.

З даного аналізу видно, що на ринку вже існують певні рішення, але попит повністю не забезпечений через зростання кількості спортивних залів.

Також було наведено обґрунтування факторів конкурентоспроможності, що викладено у таблиці 9.10.

Таблиця 9.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)

Продовження таблиці 9.10

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Зручність у використанні	Спроектований інтерфейс вміщує в собі більше функцій, та при цьому залишається зрозумілим.
2	Комплексне рішення	Поєднання двох різних типів додатків та різних функцій в одному рішенні роблять ПЗ більш ефективним.
3	Доступність	Система спроектована таким чином, що до неї можливо отримати доступ з будь-якої точки світу з доступом до мережі Інтернет.
4	Незалежність від платформи	Відсутність вимог до операційної системи пристрою дають змогу охопити більше користувачів.
5	Здатність до розширення	Системна архітектура розроблена таким чином, що можна з легкістю додавати нові типи користувачі та функції до системи.
6	Унікальні функції	Наявність бази знань в системі значно підвищують її унікальність.

Виходячи з даних цього аналізу можна дійти висновку, що розроблювана система матиме досить сильну конкурентоспроможність. За визначеними факторами було проведено оцінку сильних та слабких сторін стартап-проекту , що наведені у таблиці 9.11.

Таблиця 9.11 – Порівняльний аналіз сильних та слабких сторін стартап-проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів порівняно з проектом						
			–3	–2	–1	0	+1	+2	+3
1	Зручність у використанні	15	Sport Life		Appoi nter	Luck y Fit			
2	Комплексне рішення	18			Appoi nter, Sport Life	Luck y Fit			
3	Доступність	20	Luck y Fit		Appoi nter	Sport Life			
4	Незалежність від платформи	20				Appo inter	Luck y Fit, Sport Life		
5	Здатність до розширення	12	Luck y Fit, Appo inter		Sport Life				
6	Унікальні функції	10			Lucky Fit, Appoi nter	Sport Life			

Проведений порівняльний аналіз показав, що проект має перевагу перед конкурентами в ряді факторів, що може свідчити про успіх на ринку.

Для повноти оцінки можливостей впровадження проекту було проведено SWOT-аналіз, що дозволить детально оцінити основні чотири складові. Ці дані наведено у таблиці 9.12.

Таблиця 9.12 – SWOT-аналіз

Сильні сторони:	Слабкі сторони:
- широкий функціонал; - простота підтримки; - кроссплатформеність; - можливість розширення; - комплексність рішення; - зручність використання.	- недосконалість інтерфейсу; - час на розробку; - відсутність інтеграції з іншими CRM; - відсутність аналітики.
Можливості: - розширення клієнтської бази; - покращення графічного інтерфейсу; - додавання нових функцій; - додавання нових ролей;	Загрози: - нестача ресурсів на розвиток та покращення системи; - відсутність деяких функцій; - поява на ринку сильного конкурента; - нестабільна політична та економічна ситуація в країні.

На основі SWOT-аналізу було побудовано альтернативи ринкової поведінки для виведення проекту на ринок, що наведені у таблиці 9.13.

Таблиця 9.13 – Альтернативи ринкового впровадження

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Перевести ПЗ у формат підписки.	Велика	5-6 місяців
2	Надання знижок при купівлі з довгим терміном підтримки.	Вище середнього	1 місяць

Продовження таблиці 9.13

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
3	Безкоштовний пробний період користування.	Велика	1-2 місяці
4	Презентація проекту на різних подіях.	Середня	3-4 місяці

Отже, з огляду на проведений аналіз, найбільш оптимальною альтернативою обрано безкоштовний пробний період користування.

9.4 Розроблення ринкової стратегії проекту

Першим кроком в розробленні ринкової стратегії є створення стратегії охоплення ринку, що передбачає спочатку вибір та опис цільових груп споживачів, яким буде пропонуватись розроблена система, а потім вибір базової стратегії розвитку стартап-проекту. У таблиці 9.14 наведено вибір цільових груп потенційних споживачів.

Таблиця 9.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Власники залів будь-яких розмірів	Висока	Високий	Низька	Середня

Продовження таблиці 9.14

2	Власники мереж спортивних залів	Висока	Середній	Висока	Середня
Які цільові групи обрано: власники спортивних залів та мереж.					

Зважаючи на обрану цільову групу, було обрано стратегію диференційованого маркетингу. Після цього було сформовано базову стратегію розвитку (табл. 9.15).

Таблиця 9.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Надання функціоналу, що відсутній у товарів-замінників	Постійне покращення продукту та більше охоплення ринку	Збільшення прихильності клієнтів. Відмінний функціонал та характеристики.	Стратегія диференціації

Наступним кроком було обрано стратегію конкурентної поведінки, що наведена у таблиці 9.16.

Таблиця 9.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки

Продовження таблиці 9.16

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні, система є покращенням існуючих аналогів.	Основним шляхом буде пошук нових клієнтів	Буде копіювання з покращенням таких характеристик: - зручність інтерфейсу; - необхідний функціонал.	Наслідування лідеру.

Наступним кроком було визначення стратегії позиціонування стартап-проекту (табл. 9.17).

Таблиця 9.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Помірна ціна.	Зменшення вартості розробки.	Використання безкоштовних технологій	Дешевизна
2	Зручність використання	Аналіз потреб користувачів.	Інтуїтивно-зрозумілий інтерфейс	Зрозумілість

Продовження таблиці 9.17

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
3	Легкість підтримки та розширення	Купівля довготривалої підтримки	Побудова масштабованої та чистої архітектури.	Перспектива розвитку
4	Наявність необхідного функціоналу для полегшення взаємодії з клієнтами	Формування вимог та розробка	Наявність функцій відміток, запису на тренування та програм тренувань.	Взаємодія з тренером

В результаті було обрано наступну стратегію ринкової поведінки: базова стратегія – стратегія диференціації, стратегія конкурентної поведінки – наслідування лідеру. Дані стратегії визначатимуть в якому напрямку працюватиме стартап-компанія на ринку.

9.5 Розроблення маркетингової програми стартап-проекту

Розробка стратегії маркетингу стартап-проекту є останнім та ключовим етапом підготовки його для виходу на ринок. Першим кроком було формування маркетингової концепції продукту, що отримає споживач. З цією метою необхідно підсумувати результати аналізу конкурентоспроможності. Для цього, у таблиці 9.18,

було визначено основні переваги концепції потенційного товару над товарами конкурентів.

Таблиця 9.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Потреба в автоматизації існуючих процесів обліку клієнтів	Повністю автоматизована інформаційна система.	Швидка та легка інтеграція в спортивний зал будь-якого розміру.
2	Потреба в підвищені лояльності клієнтів	Зручний у використанні клієнтський додаток.	Мобільна версія клієнтського сайту з широким функціоналом.
3	Потреба в контролі власника за бізнесом	Система з доступом для контролю основних бізнес-процесів.	Широкий набір дій та даних, що доступні власнику.
4	Потреба в кроссплатформенності	Система, що не залежить від платформи пристрою.	Веб-застосунок, що не потребує встановлення та не має вимог до ОС.
5	Потреба в комплексному рішенні 2 та 3 потреби	Єдина система, з двома додатками: клієнтським та адміністративним	Веб-застосунок, що виконаний в одному стилі та форматі для двох різних рішень.

До ключових переваг потенційного товару слід віднести те, що даний продукт є новим на ринку, покращує існуючий функціонал аналогів та розширює його. Наступним кроком було розроблено три рівневу маркетингову модель продукту, що наведено у таблиці 9.19 та було наведено цінові межі на продукт, що зображено у таблиці 9.20.

Таблиця 9.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>1. Товар за задумом</td><td colspan="3">Автоматизована інформаційна система для підтримки діяльності спортивного клубу на платформі .NET з використанням технології Blazor.</td></tr>
<tr><td rowspan="11">2. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Автоматизація надання послуг спортивного залу</td><td>Нм</td><td>Тх/Тл/Е</td></tr>
<tr><td>2. Зменшення витрати часу на відмічання присутності</td><td>Нм</td><td>Вр/Тл/Е</td></tr>
<tr><td>3. Зручність використання клієнтами</td><td>Нм</td><td>Ор/Е</td></tr>
<tr><td>4. Зручність використання співробітниками</td><td>Нм</td><td>Ор/Е</td></tr>
<tr><td>5. Рольова модель доступу до системи</td><td>Нм</td><td>Тх/Тл</td></tr>
<tr><td>6. Компонентна розробка</td><td>М</td><td>Вр/Тл</td></tr>
<tr><td>7. Комплексне рішення</td><td>М</td><td>Вр/Тл</td></tr>
<tr><td colspan="3">Якість: дотримання стандартів розробки архітектури, написання коду та захисту інформації. Розробка випадків для тестування.</td></tr>
<tr><td colspan="3">Ліцензія для використання</td></tr>
<tr><td colspan="3">Марка: LaCode «Gym Planet»</td></tr>
<tr><td rowspan="2">3. Товар із підкріпленням</td><td colspan="3">До продажу: наявна документація, різні формати придбання, знижки при купівлі з послугою довгострокової підтримки</td></tr>
<tr><td colspan="3">Після продажу: базова підтримка спеціалістів, можлива додаткова розробка, налаштування</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: реєстрація права на інтелектуальну власність, закритий код продукту.</td></tr>
</table>

Таблиця 9.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	2 000 грн/міс	8 000 грн/міс	>30 000 грн/міс	2 000 грн/міс – 5 000 грн/міс
2	5 000 дол	10 000 дол	>10 000 дол/рік	3 000 дол – 8 000 дол

Наступним кроком було визначено оптимальну систему збуту, що наведено у таблиці 9.21.

Таблиця 9.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Придбання ліцензії на використання	Розгортання проекту на наданих серверах. Підтримка впродовж користування. Впровадження нових функцій.	Глибока	Купівля реклами та пошук клієнтів з бажанням покращити свій бізнес.

Було прийнято рішення про збут власними силами, шляхом розповсюдження інформації про нього по різних інформаційних каналах, в тому числі купівлею рекламних оголошень.

Заключним кроком було створено концепцію маркетингових комунікацій, спираючись на обрану основу для позиціонування та визначену специфіку клієнтської поведінки. Да концепція спрямована на інформування та нагадування споживачам про товар, а також на підтримку його збуту (табл. 9.22).

Таблиця 9.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Бажання збільшити прибуток та зменшити видатки	Знайомства, Інтернет	Набір функціоналу, унікальні можливості, клієнто-орієнтований підхід та доступність продукту	Висвітлити переваги використання системи, її широкий набір функціоналу та доступну ціну	Опис переваг та акцент на унікальних функціях, кросс-платформенності системи, простоті встановлення та зручності використання

В даному розділі було проаналізовано можливості розвитку розробленої інформаційної системи як стартап-проекту. Аналіз показав, що в проекту є можливість ринкової комерціалізації, оскільки наявний попит, зростаюча динаміка ринку та висока рентабельність. Також було визначено потенційні групи клієнтів, конкурентоспроможність та бар’єри входження, після чого було визначено наявність перспектив впровадження. Для ринкової реалізації проекту варто обрати варіант з підписками на користування та безкоштовним пробним періодом. Є доцільною подальша імплементація проекту.

ВИСНОВКИ

Результатом виконання роботи є розроблена інформаційна система для підтримки діяльності тренажерного залу.

Для її розробки, під час виконання магістерської дисертації, було проаналізовано та досліджено предметну область застосування системи. Дане дослідження надало загальну картину сфери застосування.

Наступним кроком був огляд та аналіз існуючих рішень. Було виділено конкурентів та їх основні переваги та недоліки. Проаналізувавши отримані результати було визначено основні функції, що повинна виконувати система.

На основі попереднього аналізу, у розділі 2, було сформовано всі необхідні функціональні та нефункціональні вимоги до системи, що дало змогу приступити до розробки варіантів використання системи.

У розділі 3 було проведено розробку та описано діаграму сценаріїв використання. Спочатку визначено основні ролі (типи користувачів) в системі, а потім, спираючись на сформовані вимоги, було розроблено основні доступні взаємодії користувача з системою. Результатом даного розділу стало чітке розуміння необхідного функціоналу системи.

Наступним кроком був аналіз існуючих архітектурних рішень. Після цього, базуючись на, розробленій у попередньому розділі, діаграмі було обрано тип додатку та його архітектуру. Також у 4 розділі було проаналізовано можливі технології розробки, та обрано найбільш поєднувані між собою. Оскільки було вирішено, що система буде представлена веб-додатком, то розглядалися відповідно технології для побудови веб-сторінок та серверної частини. Також було прийнято рішення про використання хмарного середовища для публікації та підтримки роботи інформаційної системи, і в зв’язку з цим було проаналізовано постачальників послуг хмарних сервісів, та обрано найоптимальніший варіант.

Після вибору стеку технологій було розроблено та описано загальну структурну схему системи, на якій зображено основні її елементи та зв’язки між ними. Вона дає розуміння про загальну будову розроблюваного продукту, та його взаємодію з

користувачами. Також наступним кроком було розроблено та описано структурну схеми серверної частини. На ній більш детально розглянуто елементи саме серверного додатку, та взаємодію обраних технологій. Після чого також було приділено увагу вагомому моменту – розробці структурної схеми рівня доступу до даних, що дає чітке розуміння, як додаток буде комунікувати з обраною СУБД.

Після того, як було розглянуто принцип роботи системи, було спроектовано та розроблено базу даних. Для цього спочатку було виконано інфологічне проектування, з визначенням основних сутностей системи та їх атрибутів. Після чого було розроблено даталогічну модель даних, що зображує всі необхідні сутності, що потім перетворюються в таблиці бази даних, та зв’язки між ними. Останнім кроком був опис схеми бази даних, з зазначенням всіх типів атрибутів та їх обмежень.

Наступним кроком була розробка інтерфейсу користувача для двох додатків: клієнтського та адміністративного. З метою підвищення зручності було вирішено розробити клієнтський додаток для використання за допомогою смартфону, а адміністративний з комп’ютера. Інтерфейс розроблено таким чином, щоб він дозволяв використовувати максимум функцій при наявності мінімуму елементів. Такий підхід забезпечив створення сучасного та зручного дизайну, що є інтуїтивно зрузомілим.

Заключним кроком розробки була програмна реалізація системи. Під час її опису було вказано на основні програмні рішення, що були використані в програмі, та яким чином був реалізований розроблений інтерфейс. Також було обрано та описано хмарні сервіси, що забезпечуватимуть функціонування системи, та приведено схему їх взаємодії.

Після розробки було досліджено можливість реалізації даного продукту у вигляді стартап-проекту. Дослідження встановило, що дана система може цілком бути комерціалізована та має привабливу рентабельність. Також було обрано стратегії базового розвитку та конкурентної поведінки.

Розроблена система повністю відповідає поставленим вимогам, та може бути покращена за потреби.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Спортивний зал GraFit. URL: https://grafitgym.com/.
2. Додаток GraFit в Google Play URL: https://play.google.com/store/apps/details?id=listok.grafit&hl=ru&gl=US.
3. Офіційний сайт SpotLife URL: https://sportlife.ua/uk/discounts/equipment/.
4. Додаток SpotLife в Google Play URL: https://play.google.com/store/apps/details?id=ua.sportlife.customer&hl=ru&gl=US.
5. Офіційний сайт LuckyFit URL: https://lucky-fit.com/?utm_source=google&utm_medium=cpc&utm_campaign=search_fitness_ua&gclid=Cj0KCQjwqc6aBhC4ARIsAN06NmPRk9ADuBNPOfvRYG9r6V_fPuxUx3WFTjSapSas2FYzqtOZt--d-h8aAgfZEALw_wcB#tariffs.
6. Що таке функціональні вимоги URL: https://visuresolutions.com/uk/%D0%B1%D0%BB%D0%BE%D0%B3/%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%96-%D0%B2%D0%B8%D0%BC%D0%BE%D0%B3%D0%B8/
7. Что такое SPA URL: https://wezom.com.ua/blog/chto-takoe-spa-prilozheniya
8. Об'єктно-орієнтоване програмування. Мова Java. / Державний університет телекомунікацій Навчально-науковий інститут телекомунікацій та інформатизації Кафедра комутаційних систем. – 2014. – С. 1–2. URL: https://dut.edu.ua/uploads/l_1216_25218115.pdf.
9. Wikipedia. URL: https://ru.wikipedia.org/wiki/Oracle
10. Структурна схема. URL: https://www.wiki.uk-ua.nina.az/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%B0_%D1%81%D1%85%D0%B5%D0%BC%D0%B0.html
11. Что такое Entity Framework. URL: https://andrey.moveax.ru/post/mvc3-in-depth-entity-framework-01-basics

12. Ульяницька К.О. Загальні поняття. Типи БД. ACID. / НТУУ „КПІ” ім. Ігоря Сікорського. Кафедра Автоматики та керування у технічних системах. Київ. 2019. С. 23.

13. Модель «сутність-зв’язок» URL: https://www.wiki.uk-ua.nina.az/%D0%9D%D0%BE%D1%82%D0%B0%D1%86%D1%96%D1%8F_%D0%A7%D0%B5%D0%BD%D0%B0.html#%D0%9D%D0%BE%D1%82%D0%B0%D1%86%D1%96%D1%8F_%D0%9F%D1%96%D1%82%D0%B5%D1%80%D0%B0_%D0%A7%D0%B5%D0%BD%D0%B0.

14. Нотація Пітера Чена. URL: https://stud.com.ua/77222/informatika/notatsiya_pitera_chena.

15. Мікросервісна архітектура. URL: https://medium.com/@IvanZmerzlyi/microservices-architecture-461687045b3d.

16. Паттерн репозиторий. URL: https://gist.github.com/maestrow/594fd9aee859c809b043.

17. IoC, DI, IoC-контейнер. URL: https://habr.com/ru/post/131993/.

18. Что такое внедрение зависимостей? URL: https://apptractor.ru/info/articles/dependency-injection.html.

19. Dependency Injection Scopes in Blazor. URL: https://www.thinktecture.com/en/blazor/dependency-injection-scopes-in-blazor/

20. Вопросы и ответы по Amazon RDS. URL: https://aws.amazon.com/ru/rds/faqs/.

21. Blazor компоненты. URL: https://metanit.com/sharp/blazor/2.1.php#:~:text=%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%BC%20%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%BC%20%D0%B1%D0%BB%D0%BE%D0%BA%D0%BE%D0%BC%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%20Blazor,%D1%81%D0%B2%D1%8F%D0%B7%D0%B0%D0%BD%D0%BD%D1%83%D1%8E%20%D1%81%20%D0%BD%D0%B5%D0%B9%20%D0%BD%D0%B5%D0%BA%D0%BE%

D1%82%D0%BE%D1%80%D1%83%D1%8E%20%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D1%83.