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

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інтегровані інформаційні технології»
спеціальності 126 «Інформаційні системи та технології»
на тему: «CRM–система для фітнес–центрів»

Виконав:
студент IV курсу, групи ІА-93
Дроздюк Олександр Вікторович _____

Керівник:
асистент кафедри ІСТ,
Цимбал Святослав Ігорович _____

Рецензент:
Доцент кафедри ОТ, к.т.н. доцент,
Волокита Артем Михайлович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року

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

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 126 «Інформаційні системи та технології»
Освітньо–професійна програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проєкт студенту
Дроздюку Олександру Вікторовичу

1. Тема проєкту «CRM–система для фітнес–цетрів», керівник проєкту Цимбал Святослав Ігорович, асистент, затверджені наказом по університету від «31» травня 2023 р. №2101-с

2. Термін подання студентом проєкту 12 червня 2023 року

3. Вихідні дані до проєкту: існуючі рішення CRM–систем для фітнес–центрів, мова програмування C# та TypeScript, середовище програмування Visual Studio, платформа .Net та Angular, хмарне середовище баз даних SQL Server Management Studio.

4. Зміст пояснювальної записки: аналіз предметної області, гляд існуючих засобів реалізації, програмна реалізація,огляд застосунку, тестування застосунку, висновки до роботи.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо): діаграма прецедентів, діаграма пакетів, діаграма бази даних, діаграма активності.

6. Дата видачі завдання 1 березня 2023р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Аналіз предметної області	15.04.22 – 24.04.22	
2	Огляд існуючих рішень	24.04.22 – 03.05.22	
3	Встановлення вимог до продукту	03.05.22 – 10.05.22	
4	Вибір методів та засобів реалізації	10.05.22 – 17.05.22	
5	Створення графічних документів	17.05.22 – 21.05.22	
6	Реалізація програмної частини системи	21.05.22 – 30.05.22	
7	Тестування системи	30.05.22 – 31.05.22	
8	Оформлення пояснювальної записки	31.05.22 – 09.05.22	
9	Подання проєкту	12.06.2023	

Студент Олександр ДРОЗДЮК

Керівник Святослав ЦИМБАЛ

АНОТАЦІЯ

Дроздюк О.В. CRM–система для фітнес–центрів КПІ ім. Ігоря Сікорського, Київ, 2023

Проєкт містить 68 с. тексту, 20 рисунків, 11 таблиць, покликання на 22 літературні джерела та 4 конструкторські документи.

Ключові CRM–система, вебзастосунок, управління фітнес–центрами.

Об’єктом розробки є вебзастосунок для управління фітнес–цетрами.

Предметом розробки є методи ефективного управління фітнес–цетрами та їх користувачами.

Мета розробки – створення вебзастосунку, що допоможе менеджменту фітнес-центрів легко та плідно вести свою діяльність. У свою чергу користувач може лекго і ефективно взаємодіяти з фітнес- цетром .

Було проаналізовано існуючі рішення для управління фітнес- цетрами. Розроблено систему, яка відповідає вимогам надійного клієнт–серверного спілкування. У результаті створено вебзастосунок з використанням платформи .Net та мови програмування C# на сервреній стороні та Angular на клієнській. Для збереження даних була використана система управління базами даних SQL Server Management Studio

Система буде актуальною для використання у всіх фітнес–цетрах, а інтуїтивно зрозумілий інтерфейс доможе користувачам легко та швидко її засвоїти.

SUMMARY

Drozdiuk O.V. CRM-system for fitness centers of Igor Sikorsky Kyiv Polytechnic Institute, Kyiv, 2023

The project contains 68 pages of text, 20 figures, 11 tables, references to 22 literary sources and 4 design documents.

Key words: CRM system, web application, fitness center management.

The object of development is a web application for managing fitness centers.

The subject of development is methods of effective management of fitness centers and their users.

The purpose of the development is to create a web application that will help the management of fitness centers to conduct their activities easily and fruitfully. In turn, the user can easily and effectively interact with the fitness center.

Existing solutions for managing fitness centers were analyzed. We developed a system that meets the requirements of reliable client-server communication. As a result, a web application was created using the .Net platform and the C# programming language on the server side and Angular on the client side. The SQL Server Management Studio database management system was used to store data.

The system will be relevant for use in all fitness centers, and the intuitive interface will help users to learn it easily and quickly..

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІА93.100БАК.004 ПЗ	CRM–система для	68		
6			фітнес–центрів.			
7			Пояснювальна записка			
8						
9	А3	ІА93.100БАК.004 Д1	CRM–система для	1		
10			фітнес–центрів.			
11			Діаграма прецедентів			
12						
13	А3	ІА93.100БАК.004 Д2	CRM–система для	1		
14			фітнес–центрів.			
15			Діаграма бази даних			
16						
17	А3	ІА93.100БАК.004 Д3	CRM–система для	1		
18			фітнес–центрів.			
19			Діаграма активності			
20						
21	А3	ІА93.100БАК.004 Д4	CRM–система для	1		
22			фітнес–центрів.			
23			Діаграма пакетів			
24						
25						
26						
27						
28						

					ІА93.100БАК.004 ТП		
Змн.	Арк.	№ докум.	Підпис	Дат.			
Розробив		Дроздюк О.В.			CRM система для фітнес–центрів. Відомість проєкту	Літ. Т	Арк. 1 / Аркушів 68
Перевірив		Цимбал С.І.				КПІ ім. Ігоря Сікорського Група ІА–93	
Затв.							

Пояснювальна записка
до дипломного проєкту
на тему: «CRM–система для фітнес–центрів»

Київ – 2023 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CRM – Customer Relationship Management;

HTTP – Hyper Text Transfer Protocol;

CSS – Cascading Style Sheets;

JS – JavaScript;

REST – Representational State Transfer;

SQL – Structured Query Language;

API – Application Programming Interface;

JSON – JavaScript Object Notation;

XSS – Cross–Site Scripting;

ACID – Atomicity, Consistency, Isolation, Durability;

LINQ – Language–Integrated Query;

MVC – Model–View–Controller;

ORM – Object–Relational Mapping;

DOM – Document Object Model;

ВСТУП

У сучасному світі фітнес–центри стають все популярнішими місцями для занять спортом та підтримки здорового способу життя. Зростаюча увага до фізичної форми, здоров'я та естетичного вигляду стимулює попит на послуги цих закладів. Проте, в умовах збільшеної конкуренції серед фітнес–центрів стає важливим надавати якісний клієнтський сервіс та підтримку для залучення нових клієнтів і утримання існуючої бази клієнтів.

У цьому контексті Customer Relationship Management (CRM) системи набувають особливого значення. Вони забезпечують комплексний підхід до управління клієнтськими взаєминами, починаючи від залучення нових клієнтів до підтримки та розширення відносин із вже існуючими (рис. 1.1). Однак, загальні CRM–системи, які використовуються в бізнесі, можуть не відповідати унікальним потребам фітнес–центрів, вимагаючи спеціального підходу та адаптації до особливостей цієї галузі.

Рисунок 1.1 – Переваги CRM–системи

Тому метою дипломного проєкту є спрощення та покращення спеціалізованих CRM–систем, призначених саме для фітнес–центрів. Ця система буде враховувати особливості галузі, такі як управління абонементами, розкладами занять, контроль доступу до тренажерного залу, а також надавати інструменти для аналізу інформації про клієнтів та ефективну комунікацію з ними.

Упровадження ефективної CRM–системи в сучасні фітнес–центри є актуальним завданням, що стоїть перед індустрією спорту та здоров'я. У змінних умовах ринку та зростаючої конкуренції важливо мати інструмент, що допомагатиме підтримувати високий рівень клієнтського сервісу, залучати нових клієнтів і зберігати існуючу базу відвідувачів. Метою цього дипломного проєкту є розробка і впровадження CRM–системи, спеціально адаптованої для потреб фітнес–центрів. Це дозволить забезпечити зручне управління клієнтськими взаєминами, автоматизувати процеси продажу, підтримки клієнтів , а також збільшити ефективність маркетингових заходів. У результаті реалізації проєкту очікується поліпшення конкурентоспроможності фітнес–центрів, зростання їхніх доходів, збільшення задоволенних клієнтів. Завдання фітнес–центрів не обмежується лише наданням тренувальних приміщень та обладнання. Вони мають стати місцем, де клієнти знайдуть не лише можливість займатися фізичною активністю, але й отримають особистий підхід, мотивацію та підтримку на шляху до своїх спортивних цілей.

Для клієнтів, які вибирають фітнес–центр, ключовою стає якість наданих послуг та рівень задоволеності. Вони очікують особистого підходу, зручності розкладу занять, доступності тренерів та інших співробітників центру. Клієнти хочуть відчути, що їхні потреби та бажання враховуються і центр готовий пропонувати персоналізовані рішення.

CRM–система, спеціально розроблена для фітнес–центрів, відіграє важливу роль у забезпеченні високої якості обслуговування з боку клієнтів. Вона дозволяє зберігати та оновлювати інформацію про кожного клієнта, його потреб, визначає пріоритети та історію взаємодії. Завдяки цьому фітнес–центр

може використовувати дані для персоналізованої комунікації з клієнтами, надавати індивідуальні пропозиції та програми тренувань.

Крім того, CRM–система допомагає впорядковувати процеси реєстрації та контролю доступу, що спрощує взаємодію клієнта з фітнес–центром. Вона надає зручні онлайн–інструменти для бронювання занять, ведення тренувального журналу та отримання інформації про розклад занять.

У результаті використання CRM–системи фітнес–центри зможуть покращити рівень задоволеності клієнтів, підвищити їхню лояльність та зберегти довгострокові взаємостосунки. Це створить конкурентну перевагу для центру, сприятиме збільшенню прибутків та допоможе зберегти відмінну репутацію в спортивній галузі.

Для цього в ході роботи був створений вебзастосунок, який поєднує в собі функціональні можливості CRM–системи та легкість використання через інтернет. Цей вебзастосунок надає фітнес–центрам зручний інтерфейс для управління клієнтськими даними, моніторингу активності відвідувачів та зв'язку з ними.

Вебзастосунок дозволяє фітнес–центрам зареєструвати нових клієнтів, зберігати та оновлювати їхні дані : контактну інформацію, пріоритети, медичну історію та спортивні цілі. Крім того, центр може вести записи про тренування, прогрес клієнта, досягнення та результати. Все це допомагає тренерам та персоналу центру надавати індивідуальні рекомендації, розробити персоналізовану програму тренувань та допомогти клієнтам досягти спортивних цілей.

Крім того, застосунок включає модуль для автоматизації процесів продажу та підтримки. Це дозволяє фітнес–центрам ефективно керувати абонементами, розраховувати платежі, відстежувати фінансову інформацію та надавати зручні способи оплати для клієнтів.

Завдяки застосунку фітнес–центри отримують цінний інструмент, що сприяє підвищенню ефективності роботи, поліпшенню комунікації з клієнтами та збільшенню конкурентоспроможності. Він допомагає

забезпечити індивідуальний підхід до кожного клієнта, підтримати мобільність та доступ до інформації з будь–якого пристрою з Інтернетом, що робить спілкування з фітнес–центром ще більш зручним та легким.

Для досягнення поставленої мети були сформовані такі завдання:

– проаналізувати предметну область та існуючі рішення;

– обрати технології, які оптимально відповідають вимогам та дозволяють досягти найкращих результатів;

– розробити клієнтську та серверну частину з урахуванням потреб і вимог фінального користувача;

– виконати тестування розробленого продукту/рішення/системи з метою перевірки його функціональності та якості.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1 Означення CRM–систем

CRM–система – це програмне забезпечення, яке допомагає організаціям ефективно керувати стосунками з клієнтами. Вона включає в себе набір інструментів і функціональностей, спрямованих на збір, зберігання та аналіз даних про клієнтів, а також на взаємодію з ними.

CRM–модель – це система управління стосунками з клієнтами, що допомагає компаніям збирати, аналізувати та використовувати дані про клієнтів для покращення взаємодії. Вона включає такі аспекти, як збереження даних про клієнтів, комунікація з ними та розробка стратегій залучення та утримання клієнтів. CRM–модель дозволяє компаніям вести персоналізоване обслуговування, адаптуватися до потреб та бажань клієнтів і розвивати клієнтську базу. Наприклад, дослідження показують, що компанії, які використовують CRM–системи, мають кращі показники утримання клієнтів і збільшення продажів. CRM–модель також надає можливості для автоматизації процесів та аналізу даних. Вона допомагає компаніям ефективніше використовувати свої ресурси, збільшувати ефективність комунікації з клієнтами та приймати обґрунтовані рішення на основі зібраних даних [1].

Основна функціональність CRM–систем включає наступні аспекти:

– зберігання даних клієнтів. CRM–системи дозволяють зберігати повну інформацію про клієнтів, включаючи контактні дані, історію взаємодії, покупки, запити та інші важливі деталі. Це дозволяє підприємствам мати централізований доступ до всієї інформації про клієнтів;

– маркетинг та продажі. CRM–системи надають засоби для планування, виконання та відстеження маркетингових кампаній та продажів. Вони можуть включати функції автоматичного розсилання електронних листів, управління контактами, прогнозування продажів, аналізу потенційних клієнтів та багато іншого;

– сервісна підтримка. CRM–системи допомагають підтримувати якісну сервісну підтримку клієнтів. Вони можуть включати функції для відстеження запитів клієнтів, забезпечення ефективного розподілу завдань та моніторингу часу відгуку на запити;

– аналітика та звітність. CRM–системи надають засоби для аналізу даних клієнтів та створення звітів. Вони можуть включати графіки, діаграми, таблиці, які допомагають оцінювати ефективність маркетингових кампаній, продажів, рівень задоволеності клієнтів та інші параметри;

– автоматизація бізнес–процесів. CRM–системи можуть допомагати автоматизувати рутинні бізнес–процеси, такі як обробка замовлень, управління контрактами, планування зустрічей тощо. Це зменшує потребу в ручній роботі та забезпечує більшу ефективність.

CRM–системи можна поділити на наступні типи:

– операційні CRM–системи, цей тип CRM–систем спрямований на автоматизацію та управління процесами у сфері продажів, маркетингу та обслуговування клієнтів. Вони забезпечують функції, такі як управління контактами, обробка замовлень, відстеження продажів, управління розкладами зустрічей та інше;

– аналітичні CRM–системи. Ці системи зосереджені на аналізі даних клієнтів та створенні звітів. Вони надають засоби для збору, обробки та виведення аналітичної інформації, що допомагає приймати стратегічні рішення, планувати маркетингові активності та вдосконалювати взаємодію з клієнтами;

– стратегічні CRM–системи. Цей тип CRM–систем спрямований на довгострокове стратегічне управління взаєминами з клієнтами. Вони допомагають визначити стратегічні цілі компанії щодо клієнтів, встановити стратегії взаємодії, а також реалізувати програми лояльності та управління клієнтськими сегментами;

– соціальні CRM–системи. Ці системи орієнтовані на використання соціальних медіа для взаємодії з клієнтами. Вони дозволяють відстежувати та

відповідати на повідомлення, коментарі, відгуки клієнтів у соціальних мережах, що сприяє покращенню комунікації та репутації бренду.

Крім того, існують спеціалізовані CRM–системи, які адаптовані для конкретних галузей діяльності, наприклад для фінансових установ, готелів, інтернет–магазинів та інших.

1.2 Походженя CRM–систем

Історія створення CRM–систем починається в 1980–х роках, коли підприємства стали усвідомлювати важливість ефективного управління взаєминами з клієнтами. У цей період підприємства почали шукати способи зберігання, організації та аналізу даних про клієнтів для поліпшення комунікації та задоволення їх потреб. Перші CRM–системи були в основному базовими базами даних, що зберігали інформацію про клієнтів, контакти та історію взаємодії. Однак, ці системи мали обмежену функціональність та не могли забезпечити комплексний підхід до управління взаєминами з клієнтами.

У 1990–х роках з’явилися перші CRM програмні рішення, які поєднували в собі бази даних та функції автоматизації продажів, маркетингу та обслуговування клієнтів. Ці системи надавали можливість планувати та відстежувати маркетингові кампанії, управляти контактами, здійснювати продажі та забезпечувати сервісну підтримку.

У 2000–х роках зростання популярності Інтернету та соціальних медіа привело до появи соціально орієнтованих CRM–систем. Ці системи інтегрували можливості соціального моніторингу, відстежування думок та відгуків клієнтів у соціальних мережах, що дозволяло підприємствам активно взаємодіяти зі своїми клієнтами в онлайн–середовищі [2].

Сьогодні CRM–системи стали невід’ємною частиною багатьох підприємств у різних галузях. Прогресивні технології, такі як штучний інтелект, аналітика даних, машинне навчання, постійно вдосконалюють можливості CRM–систем. Вони дозволяють компаніям збирати, аналізувати

та використовувати великі обсяги даних для зростання продажів, покращення обслуговування клієнтів та прийняття стратегічних рішень.

Причини створення CRM–систем були спрямовані на покращення взаємин з клієнтами і досягнення певних переваг:

– поліпшення комунікації з клієнтами. CRM–системи допомагають зберігати та організовувати інформацію про клієнтів, що дозволяє підприємствам краще розуміти їх потреби та надавати персоналізовану обслуговування;

– збільшення ефективності продажів. CRM–системи надають інструменти для керування продажами, прогнозування продажів та підвищення конверсії. Вони сприяють оптимізації продажних процесів та забезпечують кращу координацію між продажними командами;

– покращення взаємодії з клієнтами. CRM–системи дозволяють забезпечувати якісну сервісну підтримку, відстежувати запити клієнтів, вирішувати проблеми та надавати інформацію вчасно та ефективно;

– аналіз та прийняття рішень. CRM–системи надають аналітичні засоби для вивчення даних клієнтів, їх поведінки та попиту на продукти. Це допомагає підприємствам приймати обґрунтовані стратегічні рішення, планувати маркетингові активності та визначати кращі способи взаємодії з клієнтами.

Із часом CRM–системи стали необхідним інструментом для будь–якої компанії, яка цінує своїх клієнтів та прагне забезпечити їм якісне обслуговування, підвищити лояльність та досягнути успіху на ринку.

1.3 Оцінка наявних програмних рішень

1.3.1 Mindbody

Mindbody (рис. 1.2) є однією з найпопулярніших CRM–систем для фітнес–індустрії [3]. Вона пропонує широкий спектр функцій, таких як управління розкладом занять, облік клієнтів, онлайн–бронювання, оплата та

звітність. Mindbody також інтегрується з іншими популярними платформами, що дозволяє спростити процеси управління.

Переваги:

– широкий функціонал, охоплює всі аспекти управління фітнес–залом;

– інтеграція з іншими платформами та сервісами;

– зручний інтерфейс для клієнтів та співробітників.

Недоліки:

– висока вартість для невеликих підприємств;

– деякі користувачі вказують на складність в освоєнні системи.

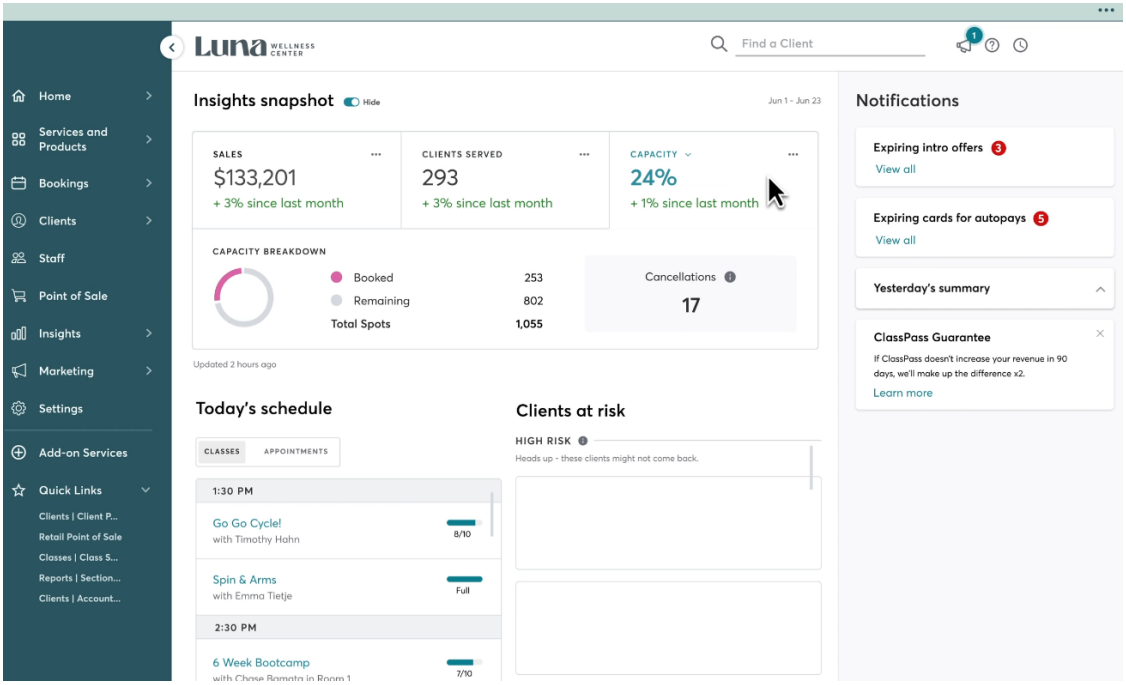


Рисунок 1.2 – Інтерфейс вебз–астосунку Mindbody

1.3.2 Zen Planner

Zen Planner є ще однією популярною CRM–системою, спеціально розробленою для фітнес–центрів [4]. Вона пропонує функції для керування клієнтами, розкладом занять, онлайн–реєстрації, оплати та аналітики.

Переваги:

– легке використання та інтуїтивний інтерфейс (рис. 1.3);

– добре підходить для невеликих фітнес–центрів;

– підтримка для планування тренувань та ресурсів.

Недоліки:

– обмежений функціонал порівняно з іншими CRM–системами;

– відсутність деяких інтеграцій із популярними сервісами.

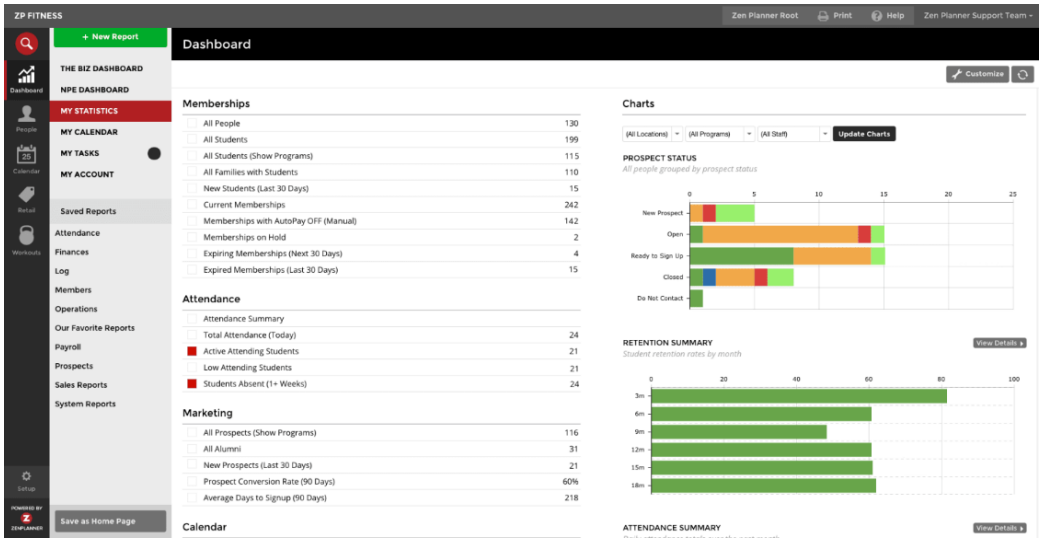


Рисунок 1.3 – Інтерфейс вебзастосунку Zen Planner

1.3.3 Perfect Gym

Perfect Gym є комплексною CRM–системою для фітнес–центрів та спортивних комплексів [5]. Вона надає функції для управління клієнтами, розкладом занять, оплатою, членством та аналітикою (рис. 1.4).

Переваги:

– широкий функціонал та можливість налаштування системи під конкретні потреби;

– інтеграція зі сторонніми сервісами та обладнанням;

– розширені можливості для звітності та аналітики.

Недоліки:

– висока вартість, особливо для невеликих підприємств;

– деякі користувачі зазначають складність в освоєнні системи.

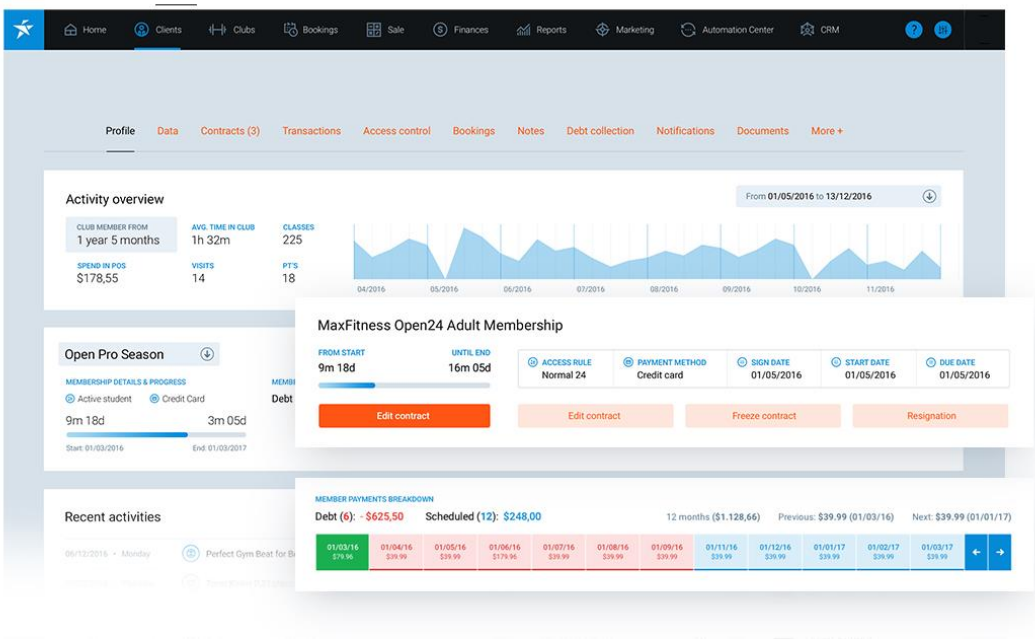


Рисунок 1.4 – Інтерфейс вебзастосунку Perfect Gym

Висновки до розділу 1

У цьому розділі було розглянуто CRM – системи, їх походження та наявні рішення. Отримані дані дозволили зрозуміти, що CRM системи виникли як відповідь на необхідність підтримки та оптимізації взаємодії підприємств зі своїми клієнтами.

У давні часи взаємодія з клієнтами здійснювалася за допомогою простих засобів, таких як записні книжки, картотеки або ексель–таблиці. Однак, з розвитком технологій та збільшенням кількості клієнтів ці методи виявилися недостатніми і недоцільними.

Аналізуючи предметну область, можна зробити наступні висновки:

– походження CRM–системи для фітнес–центрів розвинулися від загальних CRM рішень, спеціалізуючись на унікальних потребах цієї галузі. Вони виникли від необхідності ефективного управління клієнтами, продажами та обслуговуванням у специфічних умовах фітнес–індустрії;

– сьогодні на ринку існує декілька CRM–систем, спеціально призначених для фітнес–центрів. Кожна з них має свої переваги та недоліки. Mindbody пропонує широкий функціонал, але може бути складним у налаштуванні. Zen Planner надає зручний інтерфейс, але має обмежені можливості звітності. Perfect Gym має розширений функціонал та аналітичні можливості, але може бути складним у використанні.

На основі проведеного аналізу існуючих рішень було сформовано задачу дипломного проєкту, яка має на меті вдосконалити існуючі підходи, запропонувати інноваційні рішення та вирішити конкретні проблеми, що виникають у CRM–системах.

2 ОГЛЯД ІСНУЮЧИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Загальний огляд потреб вебзастосунку

Розробка вебзастосунків вимагає використання різних мов програмування залежно від конкретних потреб і завдань проєкту. Вони складаються з двох основних частин: front–end та back–end. Front–end відповідає за відображення і взаємодію з користувачем, тоді як back–end відповідає за обробку даних та логіку додатку на серверному рівні. Крім того, бази даних використовуються для збереження та керування інформацією.

У світі розробки вебзастосунків є кілька популярних мов програмування для кожної з цих областей. У front–end розробці найбільш використовуваною мовою є JavaScript, яка забезпечує взаємодію з користувачем на стороні клієнта. Додатково, HTML і CSS використовуються для структуризації та стилізації вебсторінок відповідно.

У back–end розробці існує багато мов програмування для вибору. Наприклад, мова C# є однією з найпопулярніших, оскільки вона добре підходить для розробки вебзастосунків. Інші популярні мови для back–end розробки включають Python, Ruby, Java та PHP. Кожна з цих мов має свої переваги, такі як швидкодія, розширюваність або легкість використання.

Бази даних грають важливу роль у збереженні та керуванні інформацією, використовуваною в вебзастосунках. Популярні бази даних, які використовуються у веброзробці, включають MySQL, PostgreSQL, MongoDB та SQLite. Кожна з них має свої унікальні особливості та призначена для різних типів проєктів.

Вибір мови програмування та бази даних для розробки вебзастосунків залежить від потреб, вмінь, технічних вимог та можливостей проєкту. Важливо аналізувати переваги та обмеження кожної мови та бази даних, щоб зробити правильний вибір і створити надійний та ефективний вебзастосунок.

2.2 Технології розробки клієнтської частини

Клієнтська частина програмного забезпечення (Front–end) – це та частина додатку, яка відповідає за взаємодію з користувачем. Вона відображається на екрані пристрою, з якого користувач взаємодіє з додатком. Клієнтська частина забезпечує візуальне представлення додатку, обробку користувацьких дій та передачу даних на сервер для обробки.

Основна мета клієнтської частини – забезпечити зручний та ефективний інтерфейс для користувачів, забезпечити їхню взаємодію з додатком, валідацію введених даних, анімацію, реакцію на події, зручну навігацію та багато іншого. Для досягнення цих цілей, розробники використовують різні технології.

HTML є основною мовою розмітки для вебсторінок. Вона використовується для структуризації та організації контенту на сторінці. HTML визначає різні елементи, такі як заголовки, параграфи, списки, таблиці, посилання та інші і дозволяє розміщувати на них різні типи контенту.

CSS використовується для стилізації і вигляду вебсторінок. Він дозволяє змінювати кольори, розміри, шрифти, розташування елементів та багато іншого. CSS працює в парі з HTML, де стилізація задається за допомогою відповідних селекторів і правил.

JavaScript є мовою програмування, яка використовується для створення динамічних та інтерактивних елементів на вебсторінках. Він додає можливість реагувати на події, виконувати асинхронні запити до сервера, маніпулювати вмістом сторінки, створювати анімацію та багато іншого. JavaScript є однією з ключових технологій для розробки вебзастосунків [6].

На сьогодні технології клієнтської частини стали дуже маштабними, що породило набори інструментів, бібліотек та компонентів, які допомагають розробникам будувати вебзастосунки швидше та ефективніше. Вони надають структуру, методології та готові рішення для розробки вебзастосунків і мають назву веб–фреймворк.

Основні причини використання вебфреймворків:

– вебфреймворки надають шаблони, готові компоненти та інструменти, що допомагають зменшити час розробки. Вони спрощують процес створення рутинних завдань, таких як маршрутизація, обробка запитів, валідація форм тощо;

– вебфреймворки надають структуру для організації коду та логіки додатка. Вони визначають правила та патерни для побудови додатків, що допомагає підтримувати код чистим, легким для розуміння та розширення;

– вебфреймворки зазвичай мають вбудовані механізми безпеки, що допомагають уникнути загрозам, таким як SQL–ін'єкції, міжсайтовий скриптинг XSS та інші. Вони надають шаблони для безпечного зберігання, обробки та передачі даних;

– вебфреймворки активно підтримуються спільнотою розробників, що означає наявність документації, оновлень, плагінів та інструментів. Вони також забезпечують сумісність з різними браузерами та пристроями, забезпечуючи кросс–платформену роботу додатків.

Загалом, вебфреймворки спрощують процес розробки вебзастосунків, забезпечують структуру, безпеку та підтримку, що допомагає розробникам ефективно працювати та швидше досягати бажаних результатів.

Angular, React та Vue.js є три з найпопулярніших вебфреймворків, які використовуються для розробки сучасних вебзастосунків. Кожен з них має свої унікальні особливості та підходи до розробки, але спрямовані на полегшення створення динамічних та ефективних інтерфейсів користувача.

Angular є повноцінним фреймворком для розробки вебзастосунків, створений компанією Google. Він базується на мові програмування TypeScript і пропонує широкі можливості для розробки односторінкових додатків. Angular використовує компонентний підхід, де застосунок розбивається на невеликі компоненти, що полегшує управління та поновлення інтерфейсу. Він надає потужні можливості для зв'язку даних, маршрутизації, валідації форм та інших аспектів розробки.

React є JavaScript бібліотекою для створення інтерфейсів користувача, розробленою Facebook. Він використовує віртуальний DOM, що дозволяє ефективно оновлювати та маніпулювати вебсторінкою. React зосереджений на компонентах, де інтерфейс розбивається на незалежні компоненти, які можуть бути повторно використані. Він має велику спільноту розробників і багато сторонніх бібліотек, що робить його дуже гнучким і розширюваним.

Vue.js є прогресивним JavaScript фреймворком для створення інтерфейсів користувача. Він пропонує простий та елегантний синтаксис, що робить його легким для вивчення та використання. Vue.js зосереджений на реактивності даних, де зміни в даних автоматично оновлюють відповідні частини інтерфейсу. Він надає модульність та можливості компонентного підходу, що сприяє повторному використанню коду та швидкій розробці [7].

Кожен з цих фреймворків має свої переваги та використовується в широкому спектрі проєктів, а їх порівняння наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння React Angular та Vue.js

Критерії	React	Angular	Vue.js
Мова програмування	JavaScript	TypeScript	JavaScript
Види компонентів	Функціональні та класові компоненти	Компоненти з шаблонами	Функціональні та класові компоненти
Віртуальний DOM	Так	Так	Так
Розмір	Невеликий	Великий	Невеликий
Екосистема	Широкий вибір сторонніх бібліотек та інструментів	Вбудована платформа Angular з великою кількістю функціоналу	Розширювана екосистема з плагінами

Навчальна крива	Помірна	Складна	Помірна
Підтримка	Велика та активна спільнота	Велика та активна спільнота, офіційна підтримка від Google	Активна спільнота, підтримка Vue.js Core Team
Використання	Широко застосовується в індустрії	Широко використовується в корпоративному середовищі	Використовується в невеликих та середніх проєктах
Продуктивність	Швидкий рендеринг та оновлення	Швидкий рендеринг, але вимагає більше ресурсів	Швидкий рендеринг та висока продуктивність
Компонентна архітектура	Так	Так	Так
Використання великих проєктів	Так	Так	Так

2.3 Технології розробки серверної частини

Серверна частина вебзастосунка є однією з двох ключових складових разом з клієнтською частиною (Front–end) для розробки повнофункціональних вебзастосунків. Серверна частина відповідає за обробку запитів, виконання бізнес–логіки та зберігання даних.

Сервер, себто комп'ютер або програма, надає послуги та ресурси іншим комп'ютерам (клієнтам) через мережу, таку як Інтернет. У контексті веброзробки сервер відповідає за обробку запитів від клієнтів та надсилання відповідей.

Серверна частина (Back–end), яка є складовою частиною вебзастосунка, виконується на сервері та відповідає за обробку логіки та функціональності,

що забезпечує роботу всього застосунку. Вона включає в себе бізнес–логіку, яка визначає, які операції та функції повинні бути виконані для задоволення вимог користувачів та досягнення бажаних цілей. Крім того, серверна частина включає роботу з базою даних, де зберігається інформація про користувачів, вміст та інші дані, необхідні для роботи застосунку [8].

Причини, які обумовлюють необхідність використання серверних мов програмування:

– серверні мови програмування дозволяють розробникам обробляти запити, що надходять від клієнтської частини вебзастосунку. Вони забезпечують можливість створювати логіку обробки запитів, перевіряти вхідні дані, взаємодіяти з базою даних та іншими зовнішніми сервісами;

– серверні мови дозволяють реалізовувати бізнес–логіку вебзастосунків. Це означає, що користувач можете визначити правила, обмеження та процеси, які пов'язані зі специфікою конкретного додатка. Наприклад, встановити логіку аутентифікації, авторизації, обробки платежів тощо;

– більшість вебзастосунків потребують доступу до бази даних для зберігання та отримання інформації. Серверні мови програмування надають засоби для підключення до баз даних, виконання запитів та маніпуляції даними;

– деякі задачі, такі як обробка файлів, взаємодія з вебслужбами, розрахунки та інші складні операції, краще виконувати на сервері. Серверні мови програмування надають засоби для реалізації таких операцій та оптимізації їх виконання;

– використання серверних мов програмування дозволяє розширити можливості вебзастосунку. Вони надають можливість підключати сторонні бібліотеки, фреймворки та інструменти, які розширюють функціональність додатка та спрощують розробку.

Основна мета серверних мов програмування – це надати розробникам потужний інструментарій для реалізації Back–end функціональності

вебзастосунків. Вони дозволяють обробляти запити, валідувати та зберігати дані, здійснювати підключення до зовнішніх сервісів та взаємодіяти з клієнтською частиною [9].

Існують різні підходи до розробки серверних мов програмування:

– мови загального призначення: деякі мови програмування, такі як Python, Ruby, Java та C#, є популярними виборами для розробки серверних додатків. Вони надають розширені можливості для розробки бек–енд функціональності, включаючи роботу з базами даних, мережевими протоколами, обробку запитів та багато іншого;

– спеціалізовані фреймворки: деякі мови програмування, такі як JavaScript (за допомогою Node.js), PHP (за допомогою фреймворка Laravel) та Ruby (за допомогою фреймворка Ruby on Rails), мають спеціалізовані фреймворки, які спрощують розробку серверних додатків. Ці фреймворки надають готові шаблони, бібліотеки та інструменти для розробки серверного коду;

– функціональні мови програмування: функціональні мови програмування, такі як Erlang, Haskell та Clojure, також можуть бути використані для розробки серверної частини. Вони мають унікальні функціональні властивості, такі як нелокальна обробка помилок, багатопоточність та висока масштабованість.

Щодо архітектури мережевих протоколів, то існують різні підходи, але основними є такі:

– клієнт–серверна архітектура. Це найпоширеніший підхід, в якому клієнтська частина (клієнт) взаємодіє з серверною частиною (сервер) за допомогою мережевих запитів. Клієнтська частина ініціює запити, а серверна частина обробляє ці запити та надсилає відповіді;

– RESTful архітектура – це стиль архітектури, який заснований на принципах простоти та широкого використання вебстандартів, таких як HTTP і URI. У цій архітектурі, сервер надає набір ресурсів, а клієнтська частина

здійснює взаємодію з цими ресурсами за допомогою HTTP–методів, таких як GET, POST, PUT і DELETE;

– Event–driven архітектура. У цій архітектурі серверна частина реагує на події, такі як запити, повідомлення або зміни стану, і виконує відповідні дії. Це особливо корисно для вебзастосунків, які вимагають високої швидкодії та підтримки багатопоточності;

– пакетна архітектура. У цій архітектурі дані розбиваються на пакети і передаються окремо. Кожен пакет містить адресу призначення, що дозволяє різним пакетам використовувати спільні шляхи для передачі даних. Цей підхід використовується в Інтернеті, де пакети рухаються окремо та можуть використовувати різні шляхи до призначення;

– комутація повідомлення. У цій архітектурі кожне повідомлення вважається окремим, незалежним від інших повідомлень. Кожне повідомлення має свою адресу призначення та може користуватися будь–яким доступним шляхом до призначення. Цей підхід застосовується, наприклад, в системах електронної пошти.

У цілому для розробки потужної та функціональної серверної частини веб–застосунків є необхідними серверні мови програмування та архітектурні підходи. Вони дозволяють керувати логікою програми, взаємодіяти з базою даних, обробляти запити та впроваджувати бізнес–логіку у застосунку. Вибір конкретної мови програмування для серверної частини залежить від різноманітних факторів та критеріїв, зокрема можливості мови, її екосистеми, продуктивності, підтримки спільнотою розробників та особистіих вподобань. Для порівняння різних відомих мов програмування можна звернутися до таблиці 2.2.

Таблиця 2.2 – Порівняння мов програмування

Критерій	C#	Python	PHP
Синтаксис	C–подібний	Простий та елегантний	Синтаксис, схожий на C та Perl

Використання	Розробка додатків під Windows та .NET	Загальне призначення, веброзробка, наукові дослідження, штучний інтелект	Веброзробка, серверна сторона, широко використовується у WordPress
Популярність	Популярна мова серед корпоративних розробників	Одна з найпопулярніших мов, активна спільнота	Широко використовується, особливо у веброзробці
Розширюваність	Можливість розробки на платформі .NET	Велика кількість модулів та бібліотек, легкість інтеграції	Велика кількість розширень та бібліотек, зріла екосистема WordPress
Продуктивність	Висока швидкодія, оптимізована для Windows–платформи	Помірна швидкодія, інтерпретована мова	Помірна швидкодія, інтерпретована мова
Використання у веброзробці	Так, зокрема з використанням фреймворку ASP.NET	Так, зокрема з використанням фреймворків Django та Flask	Так, основна мова у багатьох вебпроєктах, особливо у WordPress

2.4 Технології зберігання даних

Технології зберігання даних – це набір інструментів, методів та систем, які використовуються для зберігання, керування, організації та доступу до

даних в комп'ютерних системах. Вони забезпечують зручний та ефективний спосіб зберігання та обробки великих обсягів інформації.

Технології зберігання даних є однією з основних складових сучасного програмного забезпечення з різних причин:

– технології зберігання даних надають механізми для ефективного зберігання і організації даних. Вони визначають структуру даних, дозволяють встановлювати взаємозв'язки між ними та забезпечують ефективні алгоритми для доступу до цих даних;

– оптимізація для швидкого доступу до інформації. Вони включають ефективні алгоритми індексування, кешування та оптимізації запитів, що дозволяють швидко витягувати потрібні дані з бази даних;

– вони дозволяють встановлювати правила та обмеження для збереження коректності даних, запобігають втраті інформації, забезпечують безпеку та конфіденційність даних;

– технології зберігання даних надають механізми для масштабування системи зберігання даних. Вони дозволяють розширювати обсяги даних, забезпечують реплікацію, шарування та інші механізми, що дозволяють працювати з великими обсягами інформації.

Існує кілька підходів до зберігання даних, включаючи:

– реляційні бази даних. Це традиційний підхід до зберігання даних, який використовує структуровані таблиці зі зв'язками між ними;

– нереляційні бази даних. Цей підхід використовує неструктуровані формати для зберігання даних, такі як ключ–значення, стовпці, документи або графи;

– кеш–системи використовуються для тимчасового зберігання часто використовуваних даних, що дозволяє покращити швидкодію і зменшити навантаження на основну систему зберігання;

– системи файлового зберігання. Ці системи дозволяють зберігати файли та керувати ними, такими як зображення, відео, документи тощо;

– хмарні сервіси зберігання – це вебпослуги, які надають масштабований та доступний спосіб зберігання даних у хмарних середовищах [10].

ACID використовується у системах баз даних для забезпечення надійності та цілісності даних. Вони гарантують, що транзакції виконуються безпечно та надійно, незалежно від навантаження на базу даних чи конкуренції між транзакціями. ACID є основою для забезпечення надійності в багатьох додатках та системах, включаючи банківські операції, системи управління запасами та системи електронної комерції. ACID став стандартом у багатьох системах баз даних та забезпечує надійну та цілісну обробку даних. Він застосовується в багатьох областях, включаючи фінансові установи, системи управління запасами, електронну комерцію та будь–яку систему, де надійність та цілісність даних є важливими факторами [11].

Основні фактори, які спричинили появу ACID, включають:

– у багатьох сценаріях, таких як фінансові транзакції або системи управління запасами, важливо, щоб дані були завжди в правильному та консистентному стані. ACID забезпечує механізми для запобігання порушенню цілісності даних під час операцій з базою даних;

– системи баз даних повинні забезпечувати постійну доступність та надійність даних навіть у випадку аварій або відмови обладнання. ACID забезпечує механізми для тривалого зберігання та відновлення даних після відмови системи;

– у ситуаціях, коли багато користувачів одночасно працюють з базою даних та виконують транзакції, можуть виникати проблеми конкуренції, які можуть призвести до некоректного стану даних. ACID забезпечує ізоляцію транзакцій, щоб уникнути конфліктів та зберегти правильність даних;

– транзакції повинні бути атомарними, тобто вони повинні бути виконані повністю або не виконуватися зовсім. ACID забезпечує механізми для контролю та керування виконанням транзакцій, включаючи можливість відкату (відміни) змін у разі невдалого завершення транзакції.

Вибір технологій зберігання даних залежить від конкретних потреб проєкту. Потрібно звертати увагу на вимоги проєкту щодо обсягу даних, швидкодії, масштабованості, доступності, безпеки та інших факторів. Врахувати типи даних, їх структуру і взаємозв'язки.

2.5 Обрані технології для розробки системи

Були обрані наступні технології для розробки мого проєкту:

– C# відзначається високою продуктивністю, забезпечує надійність та безпеку в якості мови програмування;

– ASP.NET Web API як методологію для створення Back–end частини вебзастосунку. ASP.NET Web API надає потужний фреймворк для розробки API, що використовується для взаємодії з Front–end та обміну даними. Цей фреймворк дозволяє легко створювати RESTful API, що підтримує стандарти вебархітектури та добру масштабованість;

– RESTful API для взаємодії між Front–end та Back–end. REST є архітектурним стилем, який дозволяє створювати легкі, масштабовані та зручні для використання API. RESTful API дозволяє передавати дані між Front–end та Back–end у стандартному форматі, такому як JSON, і використовує HTTP–протокол для комунікації;

– Angular для розробки Front–end частини вебзастосунку. Angular є потужним фреймворком для створення динамічних та масштабованих вебзастосунків. Він надає широкі можливості для створення інтерактивного інтерфейсу користувача, керування станом додатку, маршрутизації та багато іншого;

– SQL як систему управління базами даних для зберігання даних. SQL є та популярною мовою запитів для управління реляційними базами даних. Використання SQL дозволяє ефективно зберігати, організовувати та взаємодіяти з даними.

Вибір таких технологій дозволить створити якісний, масштабований та ефективний вебзастосунок, шо забезпечує зручний інтерфейс для користувачів та коректну взаємодію усіх її комнонентів між собою.

Висновки до розділу 2

У цьому розділі було розглянуто загальну характеристику вимог до вебзастосунку і технологій, що використовуються для створення клієнтської та серверної частини, а також для зберігання даних. За допомогою клієнтської частини, яка використовує HTML, CSS і JS, ми можемо створювати зручний та інтерактивний інтерфейс користувача. Серверна частина, розроблена з використанням мови програмування, такої як C#, та вебфреймворків, наприклад як Angular і ASP.NET Web API, дозволяє керувати бізнес–логікою, забезпечувати взаємодію з базою даних і надавати API для зв'язку з клієнтом.

Вибір технологій зберігання даних, такої як SQL, дає можливість ефективно зберігати та управляти даними нашого вебзастосунку.

Використання RESTful API для взаємодії між Front–end та Back–end компонентами забезпечує стандартизований та ефективний обмін даними.

Архітектура проєкту дозволить розділити функціональність застосунку між клієнтською та серверною частинами, що надасть гнучкість, швидкодію та масштабованість системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Архітектура вебзастосунку

Розробка архітектури програми – це процес створення високорівневого плану, який визначає структуру, компоненти, взаємодію та організацію програмного продукту. Цей план визначає, як програма буде розбита на модулі, які взаємодіють між собою, та які рішення технологій будуть використовуватися. Була розроблена архітектура, що зображена на рисунку 3.1, компоненти системи представлені у вигляді модулів, які виконують певні функції та взаємодіють між собою через обмін пакетами даних. Взаємодія пакетів продемонстрована на краслинику ІА93.100БАК.004 Д4. Цей обмін пакетів дозволяє передавати необхідну інформацію між компонентами системи, забезпечуючи їхню спільну роботу і взаємодію.

У процесі розробки архітектури були враховані основні принципи коректної архітектури програмного забезпечення, такі як модульність, розширюваність, зручність тестування та підтримки. Крім того, вибір технологій і фреймворків для реалізації окремих компонентів був здійснений з урахуванням їхньої сумісності, швидкодії та забезпечення необхідної функціональності [12].

Під час розробки архітектури застосунку були використані як окремі технології, так їх компоненти. Головними у даній архітектурі були:

– Angular 2+, а саме Angular 1.16.0 як Front–end частина;

– Ng2–charts є бібліотекою, яка надає можливість використовувати графіки в Angular додатках. Вона базується на бібліотеці Chart.js, яка є потужним інструментом для візуалізації даних у вебзастосунках;

– RestApi, що є архітектурою мережевих протоколів;

– Angular Material – це набір готових компонентів та розширень для Angular, розроблений компанією Google. Він надає розробникам готові інтерфейсні елементи, такі як кнопки, картки, таблиці, діалогові вікна, форми

і багато іншого, щоб швидко та просто створювати сучасний та естетичний інтерфейс користувача;

– .Net Web Api фреймворк для створення вебслужб, які надають доступ до даних та функцій застосунку через HTTP протокол. Він базується на платформі .NET і надає розширені можливості для створення масштабованих та надійних вебзастосунків;

– Entity Framework, що є ORM фреймворком, розробленим компанією Microsoft. Він надає можливість взаємодіяти з базою даних з використанням об'єктно–орієнтованого підходу, приховуючи деталі роботи з базою даних, і дозволяє працювати з об'єктами та класами замість SQL запитів;

– SQL Server Management Studio – це програмне забезпечення, призначене для управління та роботи з базами даних, використовуючи мову SQL. Вона надає засоби для створення, збереження, оновлення та видалення даних в базі даних.

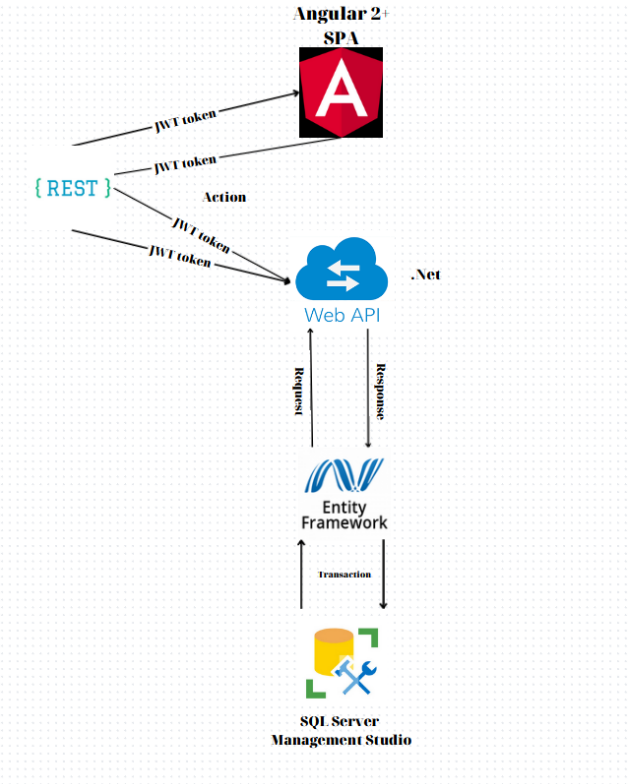


Рисунок 3.1 – Архітектура CRM–системи

3.2 Моделювання системи

Моделювання системи управління фітнес–центрами є одним із етапів у розробці програмного продукту. Це процес визначення структури, функціональності та взаємодії компонентів системи з метою забезпечення ефективного управління фітнес–центром та задоволення потреб клієнтів.

При моделюванні системи управління фітнес–центрами використовуються діаграми, які допомагають уявити структуру та функціональність системи, зокрема діаграма прецедентів, яка зображена на кресленику ІА93.100БАК.004 Д1.

Діаграма прецедентів відображає основні функції та можливості системи з точки зору її користувачів. Вона ілюструє, як різні типи користувачів взаємодіють з системою та які дії вони можуть виконувати. На діаграмі прецедентів представлені такі елементи як актори (користувачі системи) та прецеденти (функції або можливості системи) [13].

Моделювання системи управління фітнес–центрами допомагає розробникам та зацікавленим сторонам отримати чітке уявлення про функціональність та взаємодію компонентів системи, що дозволяє ефективно розробляти та підтримувати програмний продукт, задовольняючи потреби клієнтів та забезпечуючи ефективне управління фітнес–центром.

Прецедент – це поняття, яке використовується в контексті моделювання систем для опису функціональних можливостей, поведінки та взаємодії користувачів з системою. Він відображається на діаграмах прецедентів, що допомагають уявити, як система буде використовуватись та які функції вона має виконувати. Прецедент описує конкретну дію або можливість, яку користувач (актор) може виконати в системі. Він фокусується на зовнішньому поведінці системи з погляду користувача і визначає, як система реагує на його запити та дії. Кожен прецедент має назву, яка чітко вказує на його суть, а також опис, який розкриває його деталі та контекст використання. Прецеденти

можуть мати параметри, що передаються системі, а також можуть викликати інші прецеденти або бути частиною більшої послідовності дій.

Деякі приклади прецедентів які присутні у системі:

– реєстрація нового користувача. Цей прецедент описує процес реєстрації нового користувача у системі. Адміністратор може заповнити реєстраційну форму, вводячи особисті дані клієнта, його контактну інформацію та обрати певний план членства;

– запис на тренування. Цей прецедент дозволяє користувачеві записатися на певне тренування або заняття у фітнес–центрі. Користувач може переглянути розклад тренувань, вибрати зручний час і здійснити бронювання місця;

– управління тренуваннями. Цей прецедент описує можливості для адміністратора фітнес–центру керувати розкладом тренувань. Адміністратор може додавати нові тренування, видаляти або змінювати існуючі, налаштовувати кількість доступних місць і оновлювати інформацію про тренера.

3.3 Структура бази даних

При створенні бази даних і побудові її структури важливо враховувати декілька загальних положень та принципів, що допоможуть створити добре організовану та ефективну базу даних [14]. Основні принципи побудови хорошої структури бази даних включають:

– нормалізація даних. Нормалізація даних є процесом розбиття бази даних на набір таблиць і встановлення залежностей між ними. Це допомагає уникнути повторення даних і забезпечує консистентність даних. Зазвичай використовуються нормальні форми, такі як перша, друга і третя нормальні форми, а також більш високі форми;

– визначення правильних зв'язків. Зв'язки між таблицями визначаються за допомогою первинних та зовнішніх ключів. Правильне

визначення зв'язків дозволяє встановити залежності між даними і забезпечити цілісність бази даних;

– ефективність запитів. Структура бази даних повинна бути спроєктована таким чином, щоб дозволяти ефективне виконання запитів. Це включає вибір правильних індексів, оптимізацію запитів і використання оптимальних алгоритмів для обробки даних;

– безпека та доступ до даних. Забезпечення безпеки бази даних є важливим аспектом. Необхідно встановити відповідні рівні доступу до даних для користувачів та ролей, щоб забезпечити конфіденційність і цілісність даних;

– резервне копіювання та відновлення. Необхідно розробити стратегію резервного копіювання і відновлення бази даних, щоб уникнути втрати даних у разі непередбачуваних ситуацій, таких як відмови обладнання або помилкові операції;

– документація та опис схеми: Важливо документувати структуру бази даних, описуючи таблиці, їх поля, зв'язки, права доступу та інші важливі аспекти. Це полегшує розуміння бази даних і полегшує подальше розширення та підтримку.

Урахування цих загальних положень та принципів допоможе створити добре структуровану, ефективну та безпечну базу даних [15].

Структура бази даних, яка відображена у таблиці 3.1, забезпечує збереження різноманітної інформації про клієнтів, підписки на послуги, розклад занять, тренерів та їх посади, а також ролі користувачів. Зв'язки між таблицями реалізовані за допомогою використання зовнішніх ключів, що дозволяє пов'язувати дані між собою та виконувати операції об'єднання даних з різних таблиць, як показано на діаграмі, зображеній у кресленні ІА93.100БАК.004 Д2. Наприклад, ця структура дозволяє отримати докладну інформацію про клієнта та його підписки, а також проводити різноманітні запити, що використовують дані з різних таблиць.

Таблиця 3.1 – Структура бази даних

Назва таблиці	Опис
Client	Основна інформація про клієнта
Subscription	Абонемент та інформація про нього
Classes	Групові тренування
Coach	Інформація про тренера
Position	Посада
Role	Роль
User	Користувач

Таблиця Client містить інформацію про клієнтів. Основна мета цієї таблиці – зберігати дані про особисті дані клієнтів, такі як ім'я, електронна пошта та телефонний номер, що дозволяє ефективно керувати клієнтською базою та надавати персоналізовані послуги (таблиця 3.2).

Таблиця 3.2 – Опис таблиці Client

Назва поля	Опис	Тип
client_id	ID клієнта	INT
first_name	Ім'я	NVARCHAR
last_name	Прізвище	NVARCHAR
address	Адреса	NVARCHAR
email	Електронна пошта	NVARCHAR
phone	Телефон	NVARCHAR
registration_date	Дата реєстрації	DATETIME
birth_date	Дата народження	DATETIME
gender	Стать	NVARCHAR
role_id	ID ролі	INT
coach_id	ID тренера	INT

Таблиця Subscription містить інформацію про підписки клієнтів на певні послуги або програми. Ця таблиця дозволяє відстежувати та керувати різними

аспектами підписки клієнтів, такими як дата початку та закінчення підписки. Ця таблиця дозволяє зберігати інформацію про підписки клієнтів, керувати їх термінами дії та надавати звіти або статистику про активність клієнтів у певний період часу (таблиця 3.3).

Таблиця 3.3 – Опис таблиці Subscription

Назва поля	Опис	Тип
subscription_id	ID абонемента	INT
client_id	ID клієнта	INT
subscription_type	Тип абонемента	NVARCHAR
start_date	Дата початку	DATETIME
end_date	Дата закінчення	DATETIME
active	Активний	BIT

Таблиця Classes містить інформацію про різні заняття або тренування, які надаються клієнтам. Ця таблиця дозволяє організувати розклад занять з тренерами керувати ним та іншими деталями, пов'язаними з класами. Таблиця Classes може мати зв'язки з іншими таблицями, наприклад з таблицями Subscription і Coach, що дозволяє зв'язати заняття з підписками клієнтів та відповідним тренером. Вона також допомагає керувати учасниками занять та надає звіти або статистику про активність клієнтів на різних заняттях (таблиця 3.4).

Таблиця 3.4 – Опис таблиці Classes

Назва поля	Опис	Тип
class_id	ID заняття	INT
class_name	Назва заняття	NVARCHAR
trainer_id	ID тренера	INT
start_time	Початок заняття	DATETIME
end_time	Кінець заняття	DATETIME
capacity	Місткість	INT

Таблиця Coach містить інформацію про тренерів або інструкторів, які працюють у спортивному клубі або фітнес–центрі. Ця таблиця дозволяє зберігати особисті дані тренерів та відстежувати їхні посади або ролі в організації. Ця таблиця дозволяє зберігати інформацію про тренерів, включаючи їхні особисті дані та посаду. Вона також допомагає керувати розкладом занять та надає звіти або статистику про активність тренерів в організації (таблиця 3.5).

Таблиця 3.5 – Опис таблиці Coach

Назва поля	Опис	Тип
trainer_id	ID тренера	INT
first_name	Ім'я	NVARCHAR
last_name	Прізвище	NVARCHAR
specialization	Спеціалізація	NVARCHAR
position_id	Позиція	INT

Таблиця Position містить інформацію про різні посади або ролі в спортивному клубі або фітнес–центрі. Ця таблиця дозволяє визначити різні посади та керувати ними або ролями працівників в організації. Вона також допомагає зв'язувати посади з відповідними працівниками або користувачами системи (таблиця 3.6).

Таблиця 3.6 – Опис таблиці Position

Назва поля	Опис	Тип
position_id	ID посади	INT
position_name	Назва посади	NVARCHAR
salary	Зарплата	INT

Таблиця Role містить інформацію про різні ролі доступу користувачів в системі фітнес-центру. Ця таблиця дозволяє визначати та керувати рівнями доступу користувачів до різних функцій та ресурсів системи. Кожна роль визначається унікальним ідентифікатором та назвою, яка описує її

функціональні можливості. Наявність таких ролей дозволяє гнучко налаштовувати доступ до різних модулів та функцій системи залежно від потреб та обов'язків користувачів. Завдяки таблиці Role можливо ефективно керувати привілеями та обмеженням доступу, забезпечуючи безпеку та контроль над інформацією та операціями в системі фітнес-центру (таблиця 3.7).

Таблиця 3.7 – Опис таблиці Role

Назва поля	Опис	Тип
role_id	ID ролі	INT
role_name	Назва ролі	NVARCHAR
client_id	ID клієнта	INT

Ця структура таблиці User дозволяє зв'язати користувача з відповідним клієнтом. Наприклад, вона використовується, коли клієнт реєструється в системі або має обліковий запис користувача для доступу до додаткових функцій та особистої інформації (таблиця 3.8).

Таблиця 3.8 – Опис таблиці User

Назва поля	Опис	Тип
user_id	ID користувача	INT
client_id	ID клієнта	INT

Наступні зв’язки (таблиця 3.8) між сутностями описуються в базі диних:

– зв'язок Client – Role (Один до багатьох): Кожен клієнт може мати лише одну роль, але кожна роль може бути пов'язана з багатьма клієнтами. Це означає, що кожен клієнт може мати доступ до однієї ролі, але кілька клієнтів можуть мати ту саму роль;

– зв'язок User – Client (Один до одного): Кожен користувач системи (User) може бути пов'язаний з одним і тільки одним клієнтом. Це означає, що кожен користувач системи має лише один клієнтський обліковий запис;

– зв'язок Coach – Position (Один до багатьох): Кожен тренер (Coach) може мати лише одну посаду (Position), але кожна посада може бути пов'язана з багатьма тренерами. Це означає, що кожен тренер має конкретну посаду, але декілька тренерів можуть мати ту саму посаду;

– зв'язок Subscription – Client (Один до багатьох): Кожна підписка (Subscription) пов'язана з одним клієнтом, але кожен клієнт може мати багато підписок. Це означає, що кожна підписка належить лише одному клієнту, але клієнт може мати декілька підписок;

– зв'язок Classes – Coach (Один до багатьох): на кожному занятті (Classes) є лише один тренер, але кожен тренер може проводити багато занять. Це означає, що кожне заняття має одного тренера, але тренер може бути пов'язаний з багатьма заняттями;

– зв'язок Client – Coach (Один до багатьох): Кожен клієнт може мати багато тренерів, але кожен тренер може мати багато клієнтів. Це означає, що кожен клієнт може бути пов'язаний з декількома тренерами, а тренер може мати багато клієнтів.

Таблиця 3.8 – Зв’язки між сутностями

№	Сутність	Сутність	Тип зв’язку
1	Client	Role	Один до багатьох
2	User	Client	Один до одного
3	Coach	Position	Один до багатьох
4	Subscription	Client	Один до багатьох
5	Classes	Coach	Один до багатьох
6	Client	Coach	Один до багатьох

3.4 Структура клієнтської частини

Структура клієнтської частини була організована наступним чином:

– клієнтська частина була організована за допомогою модулів. Кожен модуль включає компоненти, директиви, сервіси та інші функціональні блоки, які співпрацюють між собою;

– компоненти є основними будівельними блоками клієнтської частини. Кожен компонент відповідає за відображення та управління певною частиною інтерфейсу користувача. Вони можуть містити шаблони, стилі та логіку, необхідні для взаємодії з користувачем;

– сервіси використовуються для реалізації загальної логіки, яка не прив'язана до конкретного компоненту. Вони можуть надавати функції для отримання або збереження даних, роботи зі зовнішніми сервісами або реалізації додаткової функціональності;

– директиви використовуються для зміни поведінки або вигляду DOM–елементів. Вони можуть додавати додаткові функції, включати анімацію, забезпечувати валідацію форм та інші маніпуляції з DOM [16];

– маршрутизація в Angular дозволяє переходити між різними сторінками або компонентами, забезпечуючи навігацію в додатку. Вона використовується для визначення шляхів URL та відповідних компонентів, які будуть відображені;

– Angular Material – це набір готових компонентів та стилів, розроблений командою Angular. Він надає готові рішення для розмітки, кнопок, форм, таблиць та багатьох інших елементів інтерфейсу користувача. Використання Angular Material спрощує створення сучасного та стилізованого інтерфейсу;

– Ng2–Forms – це модуль Angular, який надає засоби для роботи з формами. Він містить компоненти, директиви та сервіси, які допомагають керувати введенням даних, валідацією та взаємодією з формами на стороні клієнта;

– HTTP Client (HTTP клієнт). Angular надає модуль HttpClient для взаємодії з сервером по протоколу HTTP. Це дозволяє отримувати та відправляти дані з сервера, реалізовувати асинхронні запити та обробляти результати.

У цілому структура клієнтської частини була організована в окремі модулі, які відповідали за конкретну сторінку (рис. 3.2-3.3) [17]. Маршрутизація була побудована вбудованим модулем Angular.

Структура директорій була виконана наступним чином:

– /login – створінка входу користувача;

– /register – сторінка реєстрації користувача;

– /dashboard – основна сторінка у вебзастосунокку, яка надає користувачу зведену інформацію та доступ до ключових функцій;

– /events – інформація щодо групових занять;

– /coachandmentee– іформація про персональні тренування;

– /chart– інформація представлена у графіках;

– /customers– інформація про існуючих користувачів;

– /logout – вихід з додатку.

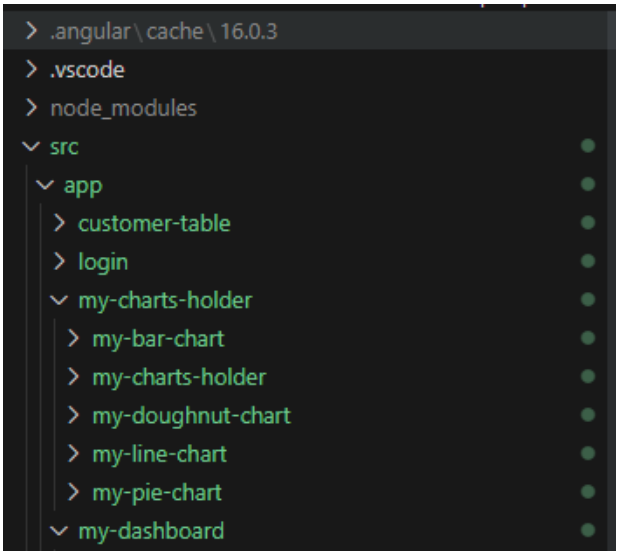


Рисунок 3.2 – Структура клієнтської частини

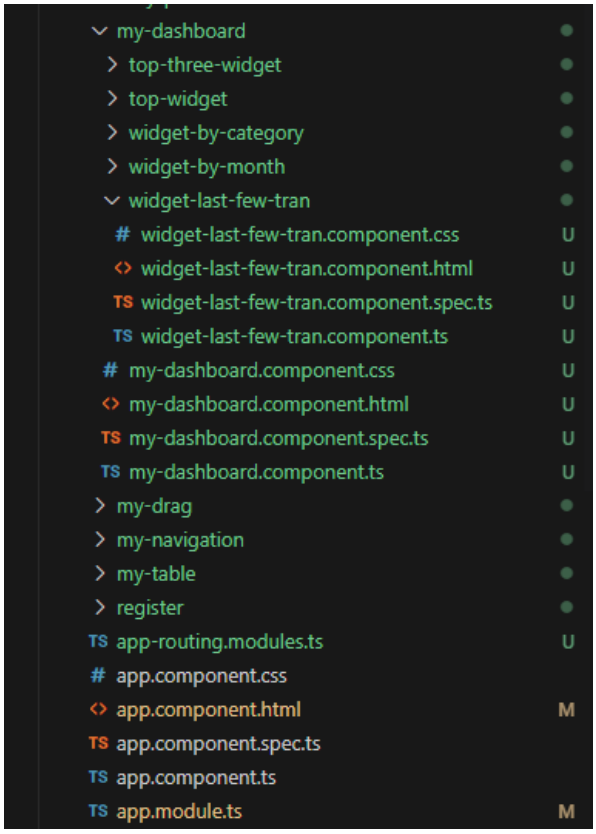


Рисунок 3.3 – Структура клієнтської частини

3.5 Структура серверної частини

Структуру серверної частини програми, заснованої на .NET Web API, організовано відповідно до принципів і шаблонів проектування (рис. 3.4). Вона включає в себе контролери, сервіси, моделі, контекст бази даних і використання принципу впровадження залежностей.

Контролери обробляють HTTP–запити, отримують параметри від клієнта і викликають відповідні сервіси для обробки бізнес–логіки. Використовують атрибути, такі як Http Get, Http Post, Route, для маршрутизації запитів [18].

Сервіси містять логіку бізнес–процесів і обробку даних. Включають методи для взаємодії з базою даних, зовнішніми службами та виконання бізнес–логіки. Можуть залежати від інших сервісів або репозиторіїв для отримання даних.

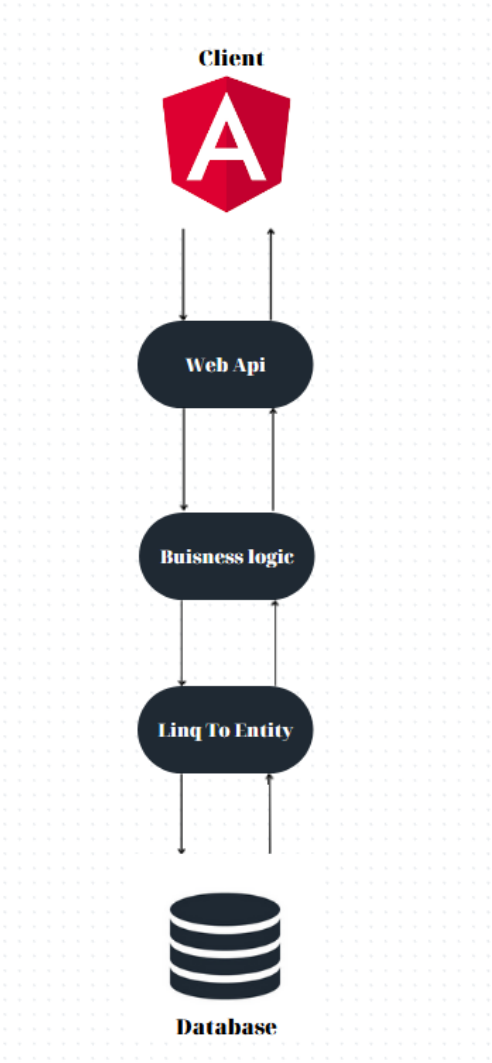


Рисунок 3.4 – Архітектура Web Api застосунку

Моделі визначають структуру даних, що використовуються в додатку. Представляють об'єкти, з якими працює серверна частина, і використовуються для передачі даних між клієнтом і сервером. Можуть містити атрибути для валідації даних і відображення на сутності бази даних [19].

Контекст бази даних (DBContext) є ключовим компонентом в платформі Entity Framework, який визначає з'єднання з базою даних і набори сутностей, з якими можна працювати. DBContext дозволяє розробникам взаємодіяти з базою даних шляхом використання об'єктів–сутностей та виконання різноманітних операцій, таких як додавання, оновлення та видалення даних.

Один з основних компонентів DBContext – це набори сутностей (DbSet). DbSet є проксі–об'єктом, який представляє таблицю бази даних і дозволяє

взаємодіяти з нею за допомогою мови запитів LINQ to Entities. Кожен DbSet відповідає певній таблиці бази даних і надає набір методів, що дозволяють виконувати додавання, видалення, оновлення та запити до цієї таблиці.

DBContext також забезпечує відстеження змін, що дозволяє автоматично виявляти та зберігати зміни в базі даних при виклику методу SaveChanges. Крім того, DBContext надає можливість налаштування параметрів підключення до бази даних, керування транзакціями та вирішення конфліктів.

Наприклад в базі даних існує таблиця "Користувачі", то можна визначити DbSet з сутністю User у DBContext, що представлятиме цю таблицю. Потім цей DbSet буде використовуватися для взаємодії з таблицею "Користувачі" шляхом виконання запитів, таких як отримання всіх користувачів, додавання нового користувача або оновлення існуючого [20].

Використання принципу впровадження залежностей спрощує управління залежностями між компонентами програми. Цей принцип дозволяє ізолювати компоненти від конкретних реалізацій інших компонентів, роблячи їх залежними від абстракцій.

Це дозволяє легко змінювати або підміняти реалізацію залежностей без необхідності вносити зміни в код компонентів, що їх використовують. Це забезпечує гнучкість та розширюваність системи, оскільки можна замінити конкретну реалізацію залежності на іншу, що задовольняє ті самі контракти або інтерфейси.

Крім того, принцип упровадження залежностей дозволяє локалізовану конфігурацію залежностей. Замість того, щоб мати жорстко закодовані залежності всередині компонентів, можна визначити ці залежності в окремому місці, наприклад у контейнері впровадження залежностей (dependency injection container). Це спрощує налаштування та зміну залежностей, не втручаючись у код компонентів.

І, нарешті, використання принципу впровадження залежностей спрощує тестування та обслуговування коду. Залежності можуть бути замінені на підроблені (mock) або підмінні (stub) об'єкти під час модульного тестування,

що дозволяє ізолювати компоненти для їх окремого тестування. Це покращує якість та стабільність коду, оскільки дозволяє легко виявляти та виправляти помилки. Крім того, обслуговування коду стає простішим, оскільки можна легко вносити зміни в реалізацію залежностей без необхідності змінювати код компонентів, що їх використовують [21].

Ці компоненти можуть бути організовані у відповідних папках або проєктах для більш зручної структури та керування кодом. Контролери розташовані в папці «Controllers», сервіси – в папці «Services», а моделі – в папці «Models».

Контролери Web API виконують основну роль у взаємодії з клієнтами та обробці їх запитів. Основні функції контролерів включають:

– контролери приймають HTTP–запити від клієнтів. Вони прослуховують вхідні маршрути та методи запитів (GET, POST, PUT, DELETE тощо) та ініціюють відповідні дії;

– контролери отримують дані, передані клієнтом, як параметри запиту або частини тіла запиту. Вони можуть виконувати перевірку та валідацію даних перед їх подальшою обробкою;

– виконання базової бізнес–логіки. Контролери можуть містити певну бізнес–логіку, пов'язану з обробкою запитів. Це може включати перевірку прав доступу, обробку даних, виклик сервісів для виконання специфічних операцій та інші взаємодії з компонентами системи;

– формування відповідей. Вони генерують відповіді, які повертаються клієнтам у вигляді HTTP–статусу, заголовків та тіла відповіді. Дані повертаються у форматі JSON після чого обробляються клієнтською частиною застосунку.

Розуміння розділення логіки та виносу її в сервіси є важливим аспектом в розробці веб–додатків. Основні причини виносу логіки в сервіси включають:

– розділення відповідальностей. Шлях до добре структурованого та легко керованого коду полягає в розділенні відповідальностей. Контролери веб–API відповідають за обробку HTTP–запитів та координацію взаємодії з

клієнтами, тоді як сервіси беруть на себе більш складну бізнес–логіку та операції з даними;

– перевикористання логіки. Виносити логіку в сервіси дозволяє перевикористовувати її в інших частинах додатку або навіть в інших додатках. Це сприяє полегшенню та швидкості розробки нових функціональностей і полегшує тестування;

– розділення логіки у сервіси спрощує їх тестування, оскільки окремі сервіси можна тестувати незалежно від контролерів та інших компонентів системи. Це дозволяє легко створювати автоматичні тести для перевірки правильності роботи сервісів;

– розділення логіки у сервіси дозволяє масштабувати окремі компоненти системи незалежно один від одного. Це дає можливість горизонтально масштабувати лише потрібні сервіси, що сприяє покращенню продуктивності та забезпечує ефективне використання ресурсів сервера.

Виносячи логіку в сервіси, ми створюємо чіткий поділ відповідальностей між контролерами та сервісами, полегшуючи розробку, тестування та підтримку вебзастосунків.

Шар бізнес–логіки в програмному застосунку відповідає за обробку бізнес–правил, логіки та обчислень, пов'язаних з доменною моделлю або конкретними функціональними вимогами. Він виконується між шаром презентації (наприклад, користувацьким інтерфейсом) і шаром доступу до даних (наприклад, базою даних).

Основна мета шару бізнес–логіки – це забезпечити правильність та цілісність обробки даних та бізнес–правил в рамках програмного застосунку. До його обов'язків входять:

– шар бізнес–логіки визначає та виконує правила, які пов'язані з бізнес–логікою програми. Це можуть бути перевірки на валідність даних, розрахунки, перетворення даних, обчислення метрик та інші операції, які відповідають за коректну обробку бізнес–правил;

– управління доменною моделлю. Шар бізнес–логіки виконує операції з доменною моделлю, яка представляє основні сутності та відношення в програмному застосунку. Він забезпечує додавання, видалення та оновлення сутностей, виконує запити до даних, валідує та забезпечує цілісність даних;

– шар бізнес–логіки відповідає за координацію дій та взаємодію між різними компонентами програмного застосунку. Він може включати в себе виклики до інших сервісів або компонентів, обробку подій та взаємодію зі зовнішніми системами;

– управління транзакціями. Шар бізнес–логіки може включати в себе управління транзакціями, щоб забезпечити консистентність та надійність операцій над даними. Він визначає межі транзакцій, реагує на помилки та виконує відміну або фіксацію змін в базі даних.

Використання шару бізнес–логіки дозволяє відокремити бізнес–правила та логіку від інших компонентів програмного застосунку. Це забезпечує більшу модульність, повторне використання коду та полегшує тестування та обслуговування програми.

Серверна части поділяється на натсупні (рис. 3.5) дирикторії:

– Api – сутність що містить у собі контролери;

– BLL – сутності що містять бізнес–логіку;

– Models – стуності згрупованих даних;

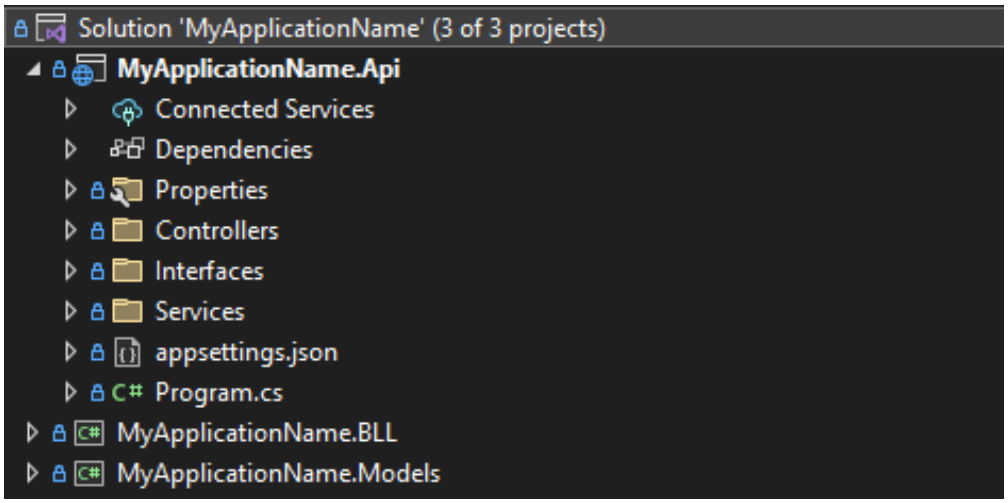


Рисунок 3.5 – Структура директорії серверної частини

Така структура дає змогу розділити функціональність, спростити розробку, підтримку і тестування коду, а також зробити його більш організованим і розширюваним [22].

Висновки до розділу 3

У розділі було виконано розробку застосунку, який передбачав перетворення концепцій та проектних рішень у реальний функціональний продукт. Процес реалізації включав створення серверної частини вебзастосунку, розробку архітектури програми, імплементацію бізнес–логіки, роботу з базою даних, інтеграцію зі сторонніми сервісами та інші технічні аспекти. У ході розробки дотримані принципи модульності, розширюваності та читабельності коду, а також використано ряд патернів проектування.

Структура клієнтської частини дозволяє розбити функціональність на незалежні компоненти, що сприяє повторному використанню коду та покращує розширюваність. Використання патерну MVC дозволяє відокремити логіку бізнес–логіки від представлення даних та взаємодії з ними.

У серверній частині була використана структура, що базується на принципах RESTful архітектури, що дозволяє розробляти гнучкі та масштабовані вебсервіси. При проектуванні використовувалися патерни, такі як Singleton та Repository, що сприяють ефективному управлінню об'єктами та доступу до даних.

4 ОГЛЯД ЗАСТОСУНКУ

4.1 Адміністратор системи

Реєстрація користувача є важливим етапом у більшості вебзастосунків, оскільки вона дозволяє створити особистий обліковий запис і отримати доступ до функціональності додатку. Незалежно від типу застосунку : чи це соціальна мережа, електронна комерція, фітнес–застосунок або будь–який інший вид програмного забезпечення - реєстрація користувачів є необхідним кроком.

У системі існує концепція "нульового адміністратора", який має повноваження реєструвати нових користувачів. Цей нульовий адміністратор використовує діаграму активності, що зображена на креслинику ІА93.100БАК.004 Д3, для керування процесом реєстрації. На діаграмі відображаються кроки, які необхідно виконати під час реєстрації, такі як введення особистих даних, вибір логіна та пароля, підтвердження електронної пошти та інші дії, що забезпечують успішне створення облікового запису.

Цей нульовий адміністратор має спеціальні привілеї, які дозволяють йому виконувати ці дії. Завдяки цим привілеям він може створювати облікові записи як для адміністраторів системи, так і для звичайних користувачів, надаючи їм необхідні права та обмеження. Діаграма активності допомагає візуалізувати послідовність дій, які виконує нульовий адміністратор, спрощує процес реєстрації та забезпечує однорідність та точність процесу.

Реєстраційний процес зазвичай включає заповнення форми з основною інформацією про користувача (рис. 4.1), такою як ім'я, електронна пошта, пароль і, можливо, додаткові дані. Деякі додатки також можуть запитувати підтвердження електронної пошти або номеру телефону для підтвердження ідентичності користувача.

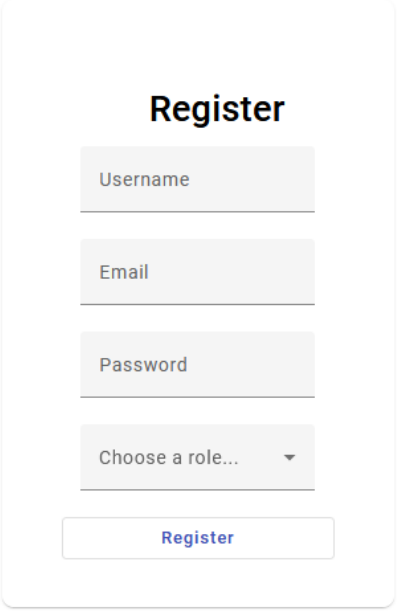


Рисунок 4.1 – Форма реєстрації користувача

Після введення необхідних даних і натискання кнопки "зареєструватися", система перевіряє валідність даних та можливість створення облікового запису. Якщо всі дані валідні, обліковий запис користувача створюється, і він отримує доступ до функціональності додатку.

Реєстрація користувача є важливим етапом, що дозволяє створити ідентифікаційні дані та забезпечити безпеку доступу до системи. Цей процес допомагає ідентифікувати та аутентифікувати користувачів, щоб вони могли користуватись усіма можливостями додатку.

У CRM–системі для фітнес–залів зазвичай можливість реєстрації нового користувача є привілеєм адміністратора. Адміністратор володіє повноваженнями для управління системою і має можливість створювати облікові записи для нових користувачів.

Наступним етапом після реєстрації є процес входу. Вхід користувача є необхідним етапом для доступу до функціональності багатьох вебзастосунків, включаючи CRM–системи для фітнес–залів. Вхід дозволяє користувачеві ідентифікувати себе і отримати доступ до особистого облікового запису з належними правами та привілеями.

Процес входу користувача зазвичай включає введення ідентифікаційних даних, таких як електронна пошта або логін та пароль у форму входу (рис. 4.2). Після введення цих даних, система перевіряє їх на правильність і здійснює автентифікацію користувача. Якщо введені дані є валідними і співпадають з даними, збереженими в системі, користувач успішно увійде до свого облікового запису.

Вхід користувача важливий для забезпечення безпеки даних та контролю доступу до функцій та інформації, які пов'язані з обліковим записом користувача. Він дозволяє персоналізувати досвід користувача, забезпечує можливість взаємодії зі своїми даними, змінювати налаштування та виконувати різноманітні дії, які передбачені функціональністю CRM–системи для фітнес–залів.

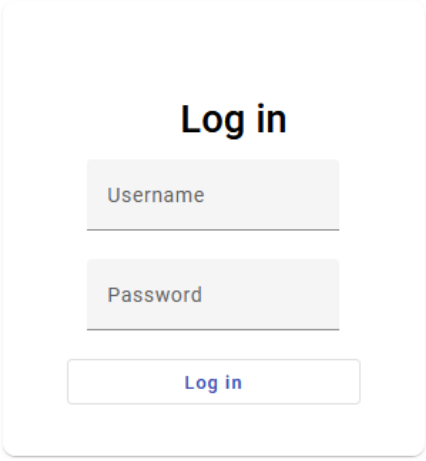


Рисунок 4.2 – Форма входу користувача

Після успішного входу у систему адміністратор отримує доступ до низки функцій, представлених у додатку. Першою сторінкою, на яку він потрапляє, є сторінка Dashboard (рис. 4.3)

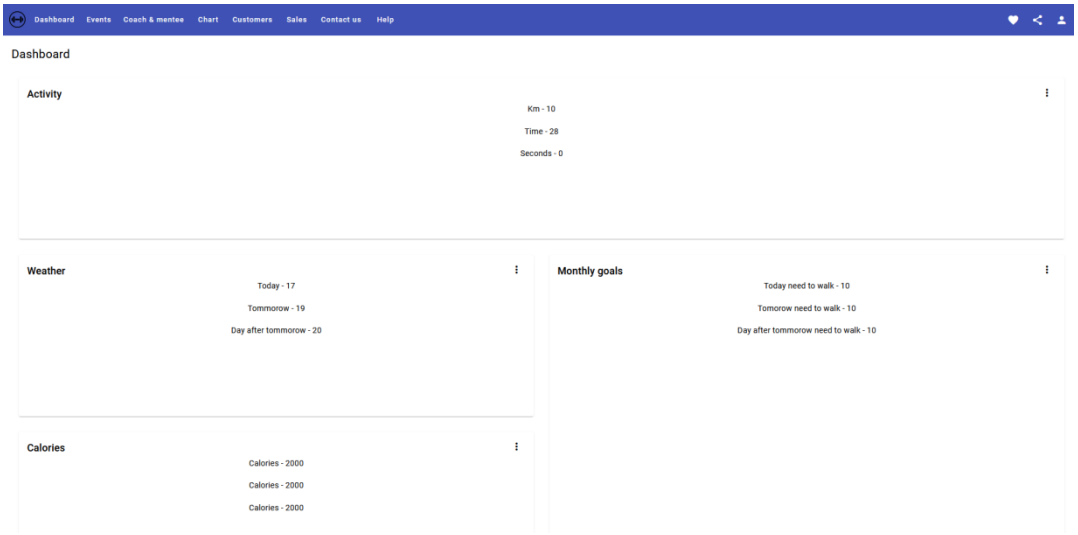


Рисунок 4.3 – Dashboard сторінка

Сторінка Dashboard є центральним місцем інформації в CRM–системі для фітнес–залів. На цій сторінці користувач може переглядати основні дані, які відображаються у вигляді категорій, забезпечуючи швидкий огляд стану та досягнень.

Однією з категорій на сторінці Dashboard є активність. Вона відображає інформацію про фізичну активність користувача, таку як кількість кроків, пройдених упродовж дня, тривалість тренувань або інші метрики, пов'язані з фітнесом. Це дозволяє користувачу відстежувати свою активність та спостерігати за її змінами з часом.

Ще однією категорією є погода. Вона надає користувачеві інформацію про погодні умови на даний момент або передбачення на найближчі дні. Це може бути корисним для планування тренувань або інших активностей, які можуть залежати від погоди.

Місячні цілі є ще однією важливою категорією на сторінці Dashboard. Вони дозволяють користувачу встановити і відстежувати свої цілі на місяць, такі як кількість тренувань, витрату калорій або досягнення певної метрики. Це стимулює користувача до досягнення своїх цілей і забезпечує мотивацію для тренувань та здорового способу життя.

Остання категорія на сторінці Dashboard – основні витрати калорій та енергії. Вона надає користувачу інформацію про витрату калорій та енергії під

час фізичних активностей або тренувань. Це може включати спалені калорії, кількість витраченої енергії або інші показники, які допомагають користувачеві контролювати свої зусилля та досягнення.

Сторінка Dashboard забезпечує зручний та швидкий огляд основної інформації для користувача CRM–системи для фітнес–залів. Вона допомагає користувачеві відстежувати свою активність, планувати тренування, контролювати свої цілі та витрату калорій, а також отримувати актуальну інформацію про погоду. Ця сторінка сприяє збереженню мотивації та покращенню результатів користувачів у фітнес–залі.

Наступною сторінкою є сторінка групових занять з назвою Enents. Сторінка Events є ключовим компонентом CRM–системи для фітнес–залів, яка присвячена груповим заняттям. На цій сторінці користувачі можуть переглядати список доступних занять, їх ціни та кількість людей, які вже доєдналися до конкретного заняття. Інформація про тренера, який буде проводити це заняття, теж вказується.

Центральним елементом сторінки Events є таблиця, яка відображає різні заняття, доступні для запису (рис. 4.4). Кожен рядок таблиці включає деталі про певне заняття, такі як його назва, тривалість, розклад, ціна та кількість доступних місць. Це надає користувачам зручний спосіб ознайомитися з різноманітними заняттями та їх характеристиками.

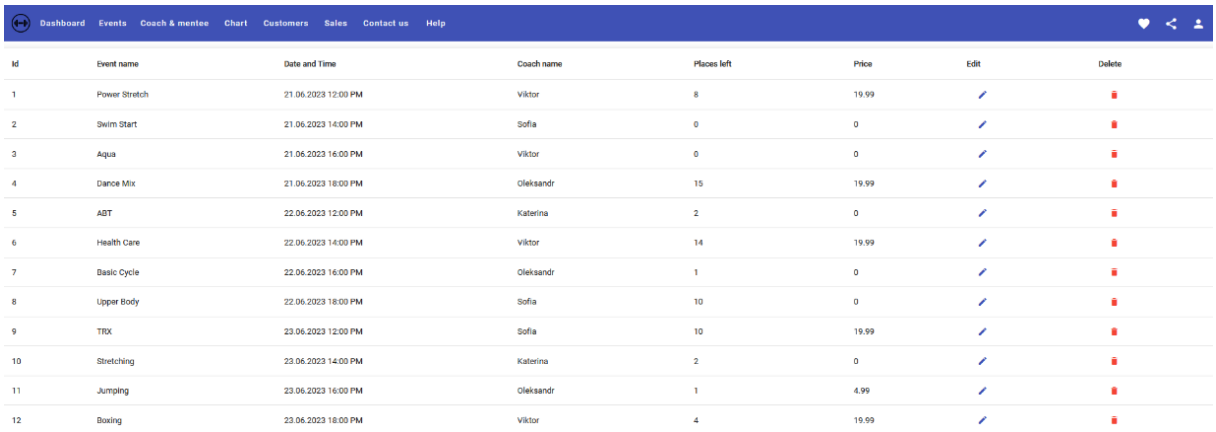


Рисунок 4.4 – Огляд сторінки групових занять

Крім того, в таблиці можуть бути вказані дані про тренера, який проводить заняття. Це дозволяє користувачам отримати інформацію про професійний рівень тренера, його досвід та спеціалізацію. Це важливо для користувачів, які хочуть бути впевненими, що заняття буде проводитися кваліфікованим фахівцем.

Сторінка Events дозволяє користувачам легко знайти та вибрати бажане заняття, ознайомитися з його вартістю, кількістю доступних місць та інформацією про тренера. Ця сторінка сприяє зручному плануванню групових занять і допомагає користувачам знайти оптимальний варіант для своїх потреб та вподобань.

Сторінка, яка іде слід за сторінкою групових занять, є сторінка персональних занять. Сторінка персональних занять в CRM–системі для фітнес–залів є канбан дошкою, яка містить інформацію про тренера, клієнта та час проведення заняття (рис. 4.5). Ця сторінка є доступною тільки для адміністратора системи, що дозволяє йому керувати персональними тренуваннями клієнтів. Крім канбан дошки, на сторінці присутні графіки, які надають адміністратору статистичну інформацію щодо проведених тренувань.

Адміністратор може переглянути графіки, які відображають кількість пропущених тренувань та проведених тренувань за останній тиждень. Це дозволяє адміністратору відстежувати активність клієнтів та забезпечувати їм відповідну підтримку.

Рисунок 4.5 – Огляд сторінки персональних тренувань

Сторінка, яка слідує за сторінкою персональних занять, є сторінкою діаграм, де надається графічне представлення інформації. Сторінка діаграм є однією з ключових сторінок CRM–системи фітнес–залу, яка доступна лише адміністратору. На цій сторінці представлені чотири різні діаграми, що відображають прогрес та зростання нових клієнтів за останні чотири місяці.

Перша діаграма (рис. 4.6) є графіком, який надає інформацію щодо прогресу нових клієнтів протягом останніх дванадцяти місяців. Цей графік відображає зміни кількості нових клієнтів на осі X та кількість нових клієнтів на осі Y, дозволяючи адміністратору аналізувати тенденції та періоди зростання або спаду нових клієнтів

.

Рисунок 4.6 – Прогрес нових клієнтів протягом останніх дванадцяти місяців

Друга діаграма (рис. 4.7) є стовпчиковою діаграмою, яка відображає розподіл нових клієнтів за останні дванадцять місяців. Кожен стовпчик представляє окремий місяць, а висота стовпчика відображає кількість нових клієнтів. Це дозволяє адміністратору порівняти кількість нових клієнтів між різними місяцями та визначити періоди з більшим або меншим потоком нових клієнтів.

Рисунок 4.7 – Розподіл нових клієнтів за останні дванадцять місяців

Третя діаграма (рис. 4.8) є круговою діаграмою, яка показує розподіл нових клієнтів за джерелами привернення. Кожний сектор кругової діаграми представляє окреме джерело привернення клієнтів, а розмір сектора відображає відсоткове співвідношення нових клієнтів, які прийшли з цього джерела. Це дозволяє адміністратору оцінити ефективність різних маркетингових стратегій та сконцентруватися на найбільш результативних джерелах привернення клієнтів.

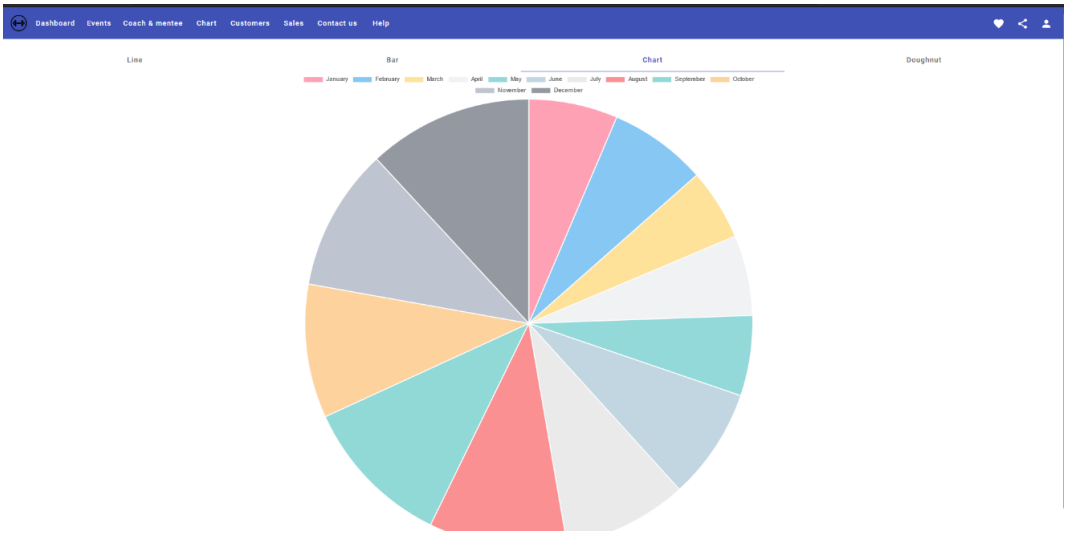


Рисунок 4.8 – Розподіл нових клієнтів за джерелами привернення

Четверта діаграма (рис. 4.9) є круговою діаграмою без середини,яка відображає розподіл нових клієнтів за категоріями. Кожний сектор кругової діаграми представляє окрему категорію клієнтів, а розмір сектора відображає відсоткове співвідношення клієнтів у кожній категорії. Це дозволяє адміністратору отримати уявлення про розподіл клієнтів за певними параметрами, наприклад за типом абонементу або за віковими групами.

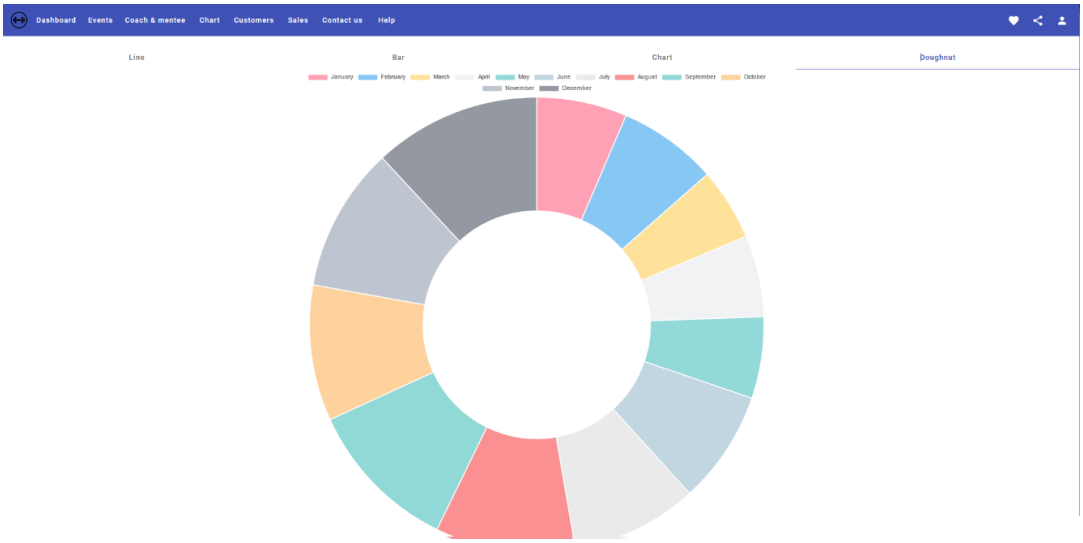


Рисунок 4.9 – Розподіл нових клієнтів за категоріями

Сторінка діаграм у цілому надає адміністратору зручну можливість візуального аналізу даних щодо прогресу та зростання нових клієнтів. Це допомагає ухвалювати обґрунтовані рішення, спираючись на числові та графічні представлення, та покращує управління фітнес–залом.

Наступна сторінка є сторінка перегляду існуючих користувачів. Сторінка Customers є важливою складовою CRM–системи фітнес–залу, яка надає адміністратору зручний спосіб управління та перегляду даних про користувачів. На цій сторінці розміщена таблиця, яка дозволяє адміністратору переглядати список користувачів та змінювати їх дані за потребою (рис. 4.10).

Customers містяться стовпці, що відображають різні аспекти інформації про користувачів. Серед цих стовпців зазвичай присутні такі дані як ім'я, прізвище, вік, дата народження, номер телефону та стать користувача. Кожен

рядок таблиці відповідає окремому користувачу, а кожен стовпець містить відповідні дані.

Адміністратор може виконувати різні дії на сторінці Customers, зокрема переглядати, редагувати та видаляти інформацію про користувачів. Це дає можливість змінювати дані, наприклад оновлювати контактні номери телефонів або вносити зміни до особистої інформації користувачів. Такий інтерактивний підхід сприяє ефективному управлінню базою даних користувачів та забезпечує актуальність і достовірність інформації.

У цілому сторінка Customers є інструментом для керування та обробки даних про користувачів. Вона дозволяє адміністратору зручно переглядати та змінювати інформацію, що стосується клієнтів, саме це сприяє покращенню процесами керування клієнтською базою та забезпечує високий рівень обслуговування.

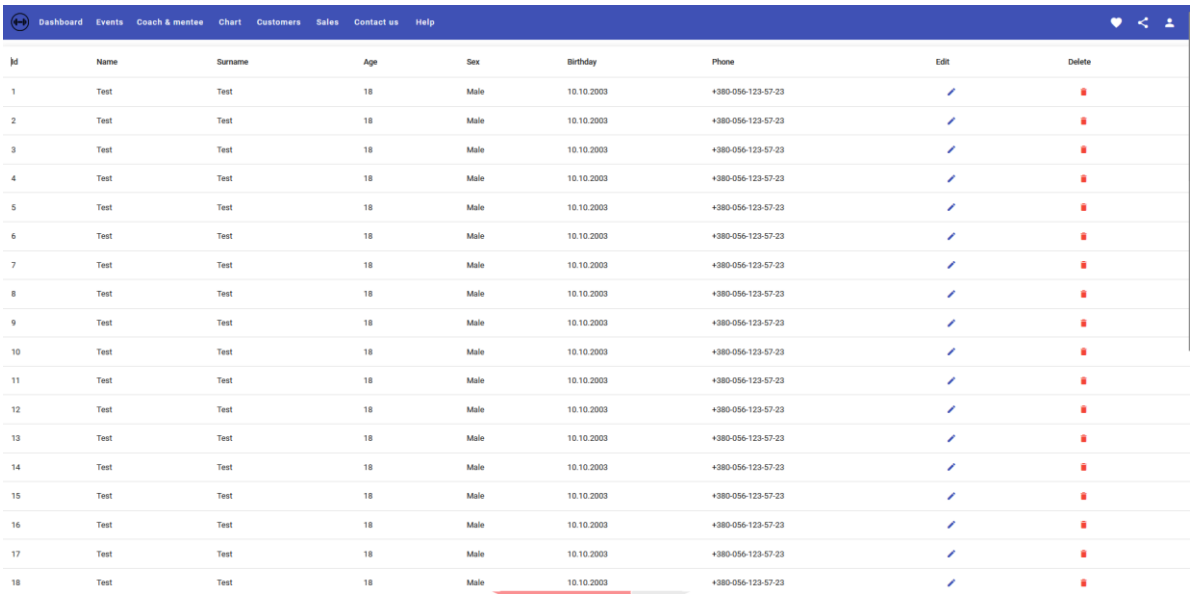


Рисунок 4.10 – Огляд сторінки користувачів

4.2 Користувач системи

Користувачі CRM–системи для фітнес–центрів відкривають перед собою нові можливості для зручного та ефективного взаємодії зі своїм улюбленим фітнес–центром. Вони стають активними учасниками процесу

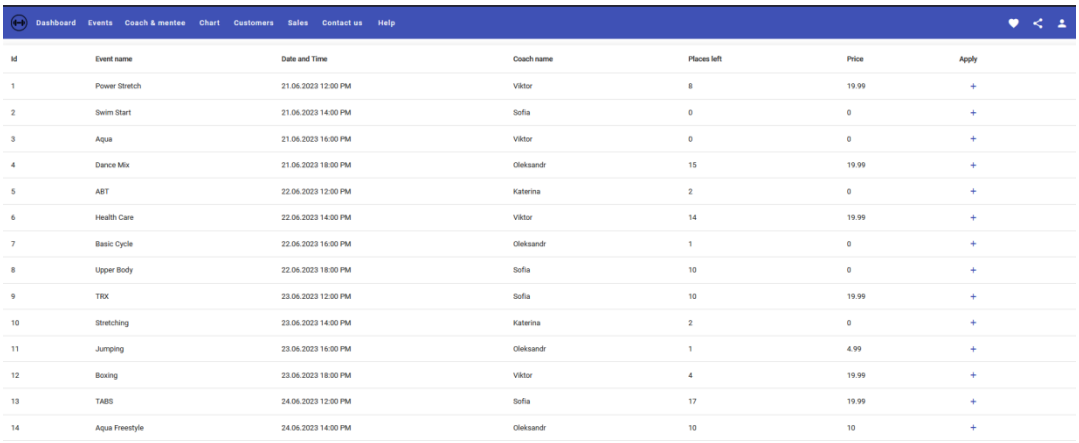
керування своїм спортивним життям, мають доступ до персоналізованої інформації та можуть контролювати свої тренування та прогрес.

Користувачі системи отримують можливість реєструватися на заняття, змінювати свій розклад та переглядати доступні тренування. Вони можуть вибрати оптимальний час та тип тренування відповідно до своїх потреб та цілей. Крім того, користувачі мають можливість відстежувати свій прогрес, фіксувати досягнення та результати, що дозволяє ставити пеед собою цілі та відстежувати досягнення, отримувати індивідуальні рекомендації та підказки.

На сторінці групових занять (рис.. 4.11) клієнти отримують доступ до широкого спектру інформації та функціональних можливостей, що робить їх участь у групових заняттях ще більш зручною та ефективною. Ця сторінка створена з метою забезпечення максимального комфорту та контролю за участю в групових тренуваннях.

На першому етапі клієнти можуть ознайомитися з розкладом групових занять та переглянути доступні типи тренувань. Вони можуть вибрати певний клас, який найкраще відповідає їхнім потребам і цілям. Додаткова інформація допомагає клієнтам зробити вибір.

Крім того, на сторінці групових занять клієнти можуть реєструватися на конкретні тренування.



Id	Event name	Date and Time	Coach name	Places left	Price	Apply
1	Power Stretch	21.06.2023 12:00 PM	Viktor	8	19.99	+
2	Swim Start	21.06.2023 14:00 PM	Sofia	0	0	+
3	Aqua	21.06.2023 16:00 PM	Viktor	0	0	+
4	Dance Mix	21.06.2023 18:00 PM	Oleksandr	15	19.99	+
5	ABT	22.06.2023 12:00 PM	Katerina	2	0	+
6	Health Care	22.06.2023 14:00 PM	Viktor	14	19.99	+
7	Basic Cycle	22.06.2023 16:00 PM	Oleksandr	1	0	+
8	Upper Body	22.06.2023 18:00 PM	Sofia	10	0	+
9	TRX	23.06.2023 12:00 PM	Sofia	10	19.99	+
10	Stretching	23.06.2023 14:00 PM	Katerina	2	0	+
11	Jumping	23.06.2023 16:00 PM	Oleksandr	1	4.99	+
12	Boxing	23.06.2023 18:00 PM	Viktor	4	19.99	+
13	TABS	24.06.2023 12:00 PM	Sofia	17	19.99	+
14	Aqua Freestyle	24.06.2023 14:00 PM	Oleksandr	10	10	+

Рисунок 4.11 – Огляд сторінки групових занять зі сторони клієнта

Користувачі системи мають доступ до сторінки входу, головної сторінки та сторінки персональних тренувань, які мають такий самий дизайн і інтерфейс, як у адміністратора. Це створює єдність в зовнішньому вигляді та допомагає користувачам легко орієнтуватися в системі.

На сторінці входу користувачі можуть увести свої облікові дані для авторизації в системі. Ця сторінка має відповідний дизайн, який включає поля для введення імені користувача та паролю, а також кнопку входу. Візуальна єдність з адміністративною стороною системи допомагає користувачам швидко знайти необхідні елементи та спрощує розробку системи.

Головна сторінка також має спільний дизайн для користувачів і адміністраторів. Вона містить інформацію про останні новини, події, акції та оновлення в фітнес–центрі. Користувачі можуть бачити важливі повідомлення, актуальний розклад тренувань, можливість зареєструватися на групові або персональні тренування, а також відкривати доступ до сторінок зі своїми особистими налаштуваннями та іншими функціями.

Сторінка персональних тренувань також має подібний дизайн для обох типів користувачів. Тут клієнти можуть бачити свій особистий графік тренувань, переглядати деталі занять, вносити зміни до свого графіку або скасовувати реєстрацію. Ця сторінка також надає користувачам можливість відстежувати свій прогрес, переглядати свою історію тренувань та отримувати повідомлення про зміни або нові пропозиції.

Такий спільний дизайн сторінок для користувачів та адміністратора створює єдність інтерфейсу та спрощує навігацію по системі. Він дозволяє користувачам швидко засвоїти основні функції, забезпечує зручність використання та створює позитивний досвід взаємодії з системою

Висновки до розділу 4

У цьому розділі було оглянуто основні сторінки та їх функціонал у системі.

– домашня сторінка має організований інтерфейс зі зручним доступом до основних функцій та інформації;

– сторінка розкладу групових занять надає можливість перегляду доступних занять та запису на них;

– сторінка персональних тренувань дозволяє клієнтам відстежувати графік тренувань, спілкуватися з тренером та змінювати розклад;

– сторінка статистики та прогресу дозволяє відстежувати досягнення через графіки та діаграми;

– сторінка управління клієнтами що дозволяє переглядати та змінювати клієнтів.

Були враховані вимоги та очікування користувачів. На основі цієї інформації було розроблено інтуїтивно зрозумілий та інформативний інтерфейс, який забезпечує зручну взаємодію з системою та ефективне управління тренуваннями та відстеженням прогресу. Виконання сторінок відповідає принципам та шаблонам програмування, що гарантує якість та надійність функціонування системи.

5 ТЕСТУВАННЯ ЗАСТОСУНКУ

5.1 Тестування сторінок входу та реєстрації

Для перевірки правильності роботи функцій логіну та реєстрації вебзастосунку було проведено тестування з використанням наступних кроків:

Тестування функції реєстрації:

- введення даних в реєстраційну форму, включаючи ім'я, електронну пошту та пароль;
- надсилання форми та перевірка відповідного повідомлення про успішну реєстрацію;
- перевірка, чи відображається новий користувач в базі даних або списку зареєстрованих користувачів.

Тестування функції логіну:

- введення валідних облікових даних (електронна пошта та пароль) в форму логіну.
- натискання кнопки "Увійти" та перевірка відповідного повідомлення про успішний вхід;
- перевірка, чи користувач має доступ до облікового кабінету після входу.

Тестування обробки помилок:

- введення невалідних даних в реєстраційну форму (наприклад, неправильний формат електронної пошти);
- перевірка, чи відображається відповідне повідомлення про помилку та пропонується виправити некоректні дані;
- введення неправильних облікових даних в форму логіну та перевірка відповідного повідомлення про помилку.

Тестування безпеки:

- спроба зареєструватися з використанням уже існуючого електронного адресу, який вже зареєстрований в системі. Перевірка наявності відповідного повідомлення про помилку та заборону реєстрації;

– спроба використовувати неправильні облікові дані під час логіну. Перевірка наявності відповідного повідомлення про помилку та заборону входу.

Ці кроки допомогли перевірити правильність роботи функцій логіну та реєстрації вебзастосунку, виявити можливі помилки та забезпечити безпеку та надійність системи.

5.2 Тестування основного функціоналу

Для перевірки правильності роботи сторінок Dashboard, Events, сторінок з канбан–дошкою та сторінки Customer вебзастосунку було проведено тестування, включаючи наступні етапи:

Тестування сторінки Dashboard:

– перевірка наявності основних елементів на сторінці, таких як заголовок, графіки, інформаційні блоки;

– перевірка коректності відображення даних, які повинні бути присутніми на Dashboard, а саме загальна статистика, огляд подій тощо;

– упевненість в тому, що кнопки та посилання на цій сторінці працюють належним чином.

Тестування сторінки Events:

– перевірка відображення списку подій та їх деталей на сторінці;

– упевненість в правильності сортування подій за датою або іншими параметрами;

– перевірка реакції на натискання кнопок або посилань, пов'язаних з подіями.

Тестування сторінок з канбан дошкою:

– перевірка правильності відображення колонок та завдань на канбан дошці;

– впевненість у можливості додавання нових завдань, переміщення їх між колонками та видалення;

– перевірка коректності збереження змін та оновлення канбан дошки.

Тестування сторінки Customer:

– перевірка відображення персональних даних користувача, таких як ім'я, контактна інформація тощо;

– перевірка можливості оновлення та збереження змін у профілі користувача;

– впевненість у функціональності кнопок або посилань, пов'язаних з користувачем.

Ці етапи допомогли перевірити правильність роботи сторінок Dashboard, Events, сторінок з канбан дошкою та сторінки Customer вебзастосунку, виявити можливі помилки та забезпечити якість та надійність системи.

Висновки до розділу 5

Під час тестування вебзастосунку були перевірені різні сторінки: Dashboard, Events, канбан–дошка та сторінка Customer. Отримані результати свідчать про їх правильну роботу та відповідність вимогам:

– сторінка Dashboard надає необхідну інформацію та графіки, працює без помилок;

– сторінка Events коректно відображає події та їх деталі, має функціональне сортування;

– сторінки канбан дошки правильно відображають колонки та завдання, функції додавання, переміщення та видалення працюють належним чином;

– сторінка Customer відображає персональні дані користувача, функціональність пов'язаних з користувачем елементів перевірена.

В результаті тестування були виявлені та виправлені помилки, що позитивно вплинуло на якість та надійність вебзастосунку.

ВИСНОВКИ

У процесі виконання дипломного проєкту було розроблено CRM–систему для фітнес–залів у вигляді вебзастосунку.

Було проведено аналіз існуючих рішень на ринку, що дозволило визначити оптимальний напрям розробки вебзастосунку. Було обрано мову програмування, що найкраще відповідає вимогам проєкту і забезпечує ефективну розробку.

У процесі розробки була дотримана систематична методологія, що сприяла ефективному управлінню процесом розробки, а також дозволила вчасно виявити та вирішити можливі проблеми.

У результаті огляду застосунку було встановлено, що він відповідає вимогам та очікуванням користувачів, має зручний інтерфейс та функціональність, що сприяє ефективній роботі з системою.

Розроблений вебзастосунок був детально протестований, що дало можливість виявити та виправити помилки та недоліки, забезпечуючи високу якість та надійність продукту.

Також застоснок має потенціал для масштабування і його послуги можуть використовувати багато користувачів. Його інфраструктура може бути легко розширена для забезпечення високої продуктивності і швидкодії при збільшенні навантаження.

Крім того, існують певні покращення, які можна внести у вебзастосунок для його подальшого удосконалення і задоволення потреб користувачів:

– удосконалення інтерфейсу користувача. Можна провести аналіз ергономіки та взаємодії з користувачем, з метою поліпшення інтуїтивного сприйняття додатку. Оптимізація розміщення елементів, використання більш зрозумілих символів та значків, а також покращення навігаційної структури можуть покращити користувацький досвід;

– вивчення потреб користувачів та ринкових тенденцій може допомогти виявити нові можливості для розширення функціональності

додатку. Наприклад, додавання додаткових модулів або інтеграція з іншими популярними сервісами може зробити застосунок більш привабливим для користувачів;

– оптимізація роботи додатку і покращення його продуктивності можуть забезпечити більш швидку та ефективну роботу. Це можна досягти шляхом удосконалення алгоритмів обробки даних, кешування, розподіленого обчислення та інших методів оптимізації;

– удосконалення системи безпеки є критичним аспектом додатку. Аудит безпеки, застосування найкращих практик шифрування даних, регулярні оновлення та моніторинг можуть забезпечити захист від потенційних загроз і зламів;

– упевненість у тому, що застосунок працює на різних платформах, браузерах та пристроях, є важливою. Варто забезпечити тестування додатку на різних конфігураціях та платформах для виявлення та виправлення можливих проблем із сумісністю.

Ці покращення допоможуть зробити вебзастосунок ще більш привабливим, функціональним та надійним для користувачів, відповідаючи їх потребам і вимогам.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Customer-centric business management system (CRM – system) / A. O. Ongdash та ін. European Journal of Humanities and Social Sciences. 2016. С. 58–61. URL: https://doi.org/10.20534/ejhss-16-1-58-61 (дата звернення: 02.06.2023).

2. Kuchumov A. V., Testina Y. S. CRM SYSTEMS: HISTORY, DEFINITIONS, CLASSIFICATION. ECONOMIC VECTOR. 2022. Т. 1, № 28. С. 41–46. URL: https://doi.org/10.36807/2411-7269-2022-1-28-41-46 (дата звернення: 01.06.2023).

3. PerfectGym. URL: https://www.perfectgym.com/ (дата звернення: 24.05.2023).

4. MindBody. URL: https://www.mindbodyonline.com/ (дата звернення: 26.05.2023).

5. Zenplanner. URL: https://zenplanner.com/ (дата звернення: 20.05.2023).

6. Kurata D. Doing Web Development: Client Side Techniques. Apress, 2001. 300 с.

7. Strazzullo F. Let’s Talk About Frameworks. Frameworkless Front-End Development. Berkeley, CA, 2019. С. 1–21. URL: https://doi.org/10.1007/978-1-4842-4967-3_1 (дата звернення: 13.06.2023).

8. Sherry P. Server-Side Language Environments. Foundation Mac OS X Web Development. Berkeley, CA, 2004. P. 223–252. URL: https://doi.org/10.1007/978-1-4302-5133-0_9 (date of access: 13.06.2023).

9. Kaluža M., Kalanj M., Vukelić B. A comparison of back-end frameworks for web application development. Zbornik Veleučilišta u Rijeci. 2019. Т. 7, № 1. С. 317–332. URL: https://doi.org/10.31784/zvr.7.1.10 (дата звернення: 13.06.2023).

10. Storage System. Encyclopedia of Big Data. Cham, 2022. С. 889. URL: https://doi.org/10.1007/978-3-319-32010-6_300191 (дата звернення: 13.06.2023).

11. Vossen G. ACID Properties. Encyclopedia of Database Systems. Boston, MA, 2009. С. 19–21. URL: https://doi.org/10.1007/978-0-387-39940-9_831 (дата звернення: 13.06.2023).

12. Sharan K. Model-View-Controller Pattern. Learn JavaFX 8. Berkeley, CA, 2015. С. 419–434. URL: https://doi.org/10.1007/978-1-4842-1142-7_11 (дата звернення: 13.06.2023).

13. The Use Case Diagram / M. Seidl та ін. UML @ Classroom. Cham, 2015. С. 23–47. URL: https://doi.org/10.1007/978-3-319-12742-2_3 (дата звернення: 13.06.2023).

14. Koymen K., Cai Q. SQL∗: A recursive SQL. Information Systems. 1993. Т. 18, № 2. С. 121–128. URL: https://doi.org/10.1016/0306–4379(93)90009–p (дата звернення: 01.06.2023).

15. Dewson R. SQL Server Management Studio. Beginning SQL Server for Developers. Berkeley, CA, 2014. С. 25–42. URL: https://doi.org/10.1007/978–1–4842–0280–7_2 (дата звернення: 02.06.2023).

16. Frisbie M. Angular 2 Cookbook. Packt Publishing – ebooks Account, 2017. 464 с.

17. Freeman A. Creating an Angular Project. Pro Angular. Berkeley, CA, 2017. С. 197–225. URL: https://doi.org/10.1007/978–1–4842–2307–9_11 (дата звернення: 17.05.2023).

18. Stephens R. C# 5.0 programmer’s reference. Indianapolis, Indiana : Wiley, 2014. URL: https://archive.org/details/c50programmersre0000step

19. Richter J. CLR via C#. 4–те вид. Redmond : Microsoft Press, 2012. 863 с.

20. Skeet J. C# in depth. 2–ге вид. Stamford, CT : Manning, 2011. 554 с.

21. Villela R. .NET Platform. Pro .NET 5 Custom Libraries. Berkeley, CA, 2020. С. 1–27. URL: https://doi.org/10.1007/978–1–4842–6391–4_1 (дата звернення: 28.05.2023).

22. Martin R. Clean Code: A Handbook of Agile Software Craftsmanship. Pearson Education, Limited.

ДОДАТОК А

Посилання на репозиторій з вихідним кодом

https://github.com/FeltMe/Client