

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

«Система управління фітнес-клубом»

Виконала студентка групи ІПЗ-21-1
Лукашевич Влада Вадимівна

Керівник роботи:
ВАКАЛЮК Тетяна Анатоліївна

Рецензент:
ЧИЖМОТРИЯ Олена Геннадіївна

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна Вакалюк

«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЛУКАШЕВИЧ Влада Вадимівна**

Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**

Тема роботи: **«Система управління фітнес-клубом»,**

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: процеси автоматизації управління фітнес-клубом, що включають реєстрацію користувачів із різними ролями, бронювання та облік тренувальних сесій, обробку абонементів і платежів і формування аналітичних звітів.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т.А.	27.02.2025р.	27.02.2025р.
2	Вакалюк Т.А.	27.02.2025р.	27.02.2025р.
3	Вакалюк Т.А.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з веб-додатку системи управління фітнес-клубом та пояснювальної записки. Пояснювальна записка до випускної роботи містить 104 сторінки, 62 ілюстрацій та 33 таблиці.

Метою роботи є розробка та впровадження веб-додатку для автоматизації процесів управління фітнес-клубом із застосуванням сучасних технологій.

Архітектура рішення базується на використанні контролерів ASP.NET MVC, які обробляють запити та взаємодіють із Microsoft Azure SQL Database. Така побудова забезпечує високу продуктивність, надійне зберігання даних і можливість подальшого масштабування.

Інтерфейс має адаптивний дизайн і інтуїтивну навігацію, що дозволяє легко планувати тренування, керувати підписками та переглядати аналітичні звіти.

КЛЮЧОВІ СЛОВА: СИСТЕМА УПРАВЛІННЯ ФІТНЕС-КЛУБОМ; ASP.NET MVC; MICROSOFT AZURE SQL DATABASE; АВТОМАТИЗАЦІЯ.

					ІПЗ.КР.Б –121 – 25 – ІЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Система управління фітнес-клубом			
Розроб.		Лукашевич В.В.						
Перевір.		Вакалюк Т.А.						
Керівник								
Н. контр.		Гичина О.П.						
Зав. каф.		Вакалюк Т.А.			Житомирська політехніка, група ІПЗ-21-1			
					Літ.	Арк.	Аркуші	
						3	104	

ABSTRACT

The final bachelor's thesis consists of a web application of a fitness club management system and an explanatory note. The explanatory note to the final thesis contains 104 pages, 62 illustrations and 33 tables.

The aim of the work is to develop and implement a web application for automating the management processes of a fitness club using modern technologies.

The solution architecture is based on the use of ASP.NET MVC controllers that process requests and interact with Microsoft Azure SQL Database. This structure ensures high performance, reliable data storage and the possibility of further scaling.

The interface has an adaptive design and intuitive navigation, which makes it easy to plan workouts, manage subscriptions, and view analytical reports.

KEYWORDS: FITNESS CLUB MANAGEMENT SYSTEM; ASP.NET MVC; MICROSOFT AZURE SQL DATABASE; AUTOMATION.

					ИПЗ.КР.Б –121 – 25 – ПЗ	
						4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ЗМІСТ	5
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ СИСТЕМИ УПРАВЛІННЯ ФІТНЕС-КЛУБОМ.....	9
1.1. Постановка задачі.....	9
1.2. Аналіз аналогів програмного продукту	12
1.3. Вибір архітектури веб-орієнтованої системи управління фітнес-клубом	18
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	20
Висновки до розділу 1	25
РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ФІТНЕС-КЛУБОМ.....	26
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	26
2.2. Розробка бази даних системи.....	34
2.3. Проектування та реалізація алгоритмів роботи системи	41
2.4. Реалізація функціоналу системи управління фітнес-клубом.....	47
Висновки до розділу 2	55
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІЗ СИСТЕМОЮ УПРАВЛІННЯ ФІТНЕС-КЛУБОМ.....	57
3.1. Порядок встановлення та налаштування параметрів системи	57
3.2. Структура інтерфейсу системи управління фітнес-клубом	59
3.3. Тестування роботи веб-додатку.....	87
Висновки до розділу 3	90
ВИСНОВКИ.....	92
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	93
ДОДАТКИ.....	96
Додаток А.....	97
Додаток Б	98

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

SDK – набір для розробки програмного забезпечення (software development kit)

MVC – схема розділення даних додатків і керуючої логіки на три окремі компоненти Model-View-Controller

CRUD – основні операції, що використовуються для управління даними (Create , Read , Update , Delete)

ВСТУП

Актуальність теми зумовлена стрімким зростанням потреби у цифровій трансформації бізнес-процесів у сфері фітнес-послуг. Із розширенням клієнтської бази, урізноманітненням видів тренувань та збільшенням обсягів даних про відвідування, абонементи й платежі виникає необхідність в ефективному, швидкому та точному управлінні. Традиційні методи ручного обліку вже не забезпечують належного рівня сервісу й ускладнюють оперативне прийняття рішень. Сучасні веб-технології й серверні фреймворки відкривають можливість створення єдиної інтегрованої системи, яка поєднує адміністрування, комунікацію з клієнтами та аналітичні інструменти в одному цифровому середовищі, що суттєво підвищує якість керування фітнес-клубом.

Мета цієї випускної кваліфікаційної роботи є розробка веб-орієнтованої системи управління фітнес-клубом, що забезпечить автоматизацію процесів реєстрації та авторизації користувачів із різними ролями (клієнт, тренер, менеджер, адміністратор), керування абонементом та розкладом тренувань, збереження даних про клієнтів та персонал, а також облік платежів. Система має полегшити роботу адміністрації клубу, мінімізувати ризики помилок при ручному обліку та забезпечити клієнтам зручну взаємодію з розкладом, оплатами та історією відвідувань.

Для реалізації визначеної мети необхідно виконати наступні **завдання**: провести аналіз предметної області та ролей користувачів, спроектувати архітектуру з чітким розподілом відповідальностей між її логічними рівнями, розробити надійну й оптимізовану базу даних з урахуванням специфіки фітнес-індустрії, реалізувати ключові алгоритми бізнес-логіки, створити адаптивний інтерфейс для різних типів пристроїв із акцентом на мобільне використання та провести тестування для перевірки відповідності вимогам до функціоналу й якості. Усі компоненти системи повинні забезпечувати стабільну роботу, швидкий відгук, надійне збереження даних і зручність взаємодії для користувачів із різним рівнем технічної підготовки.

Об’єктом дослідження є інформаційні процеси автоматизації функцій фітнес-клубу: реєстрація клієнтів, управління тренувальним розкладом, облік відвідувань та платежів.

Предметом дослідження виступають методи та засоби побудови веб-орієнтованої системи, зокрема архітектурні підходи ASP.NET MVC, механізми збереження даних у Microsoft Azure SQL Database та реалізація бізнес-логіки через Entity Framework Core.

Для досягнення поставленої мети ми застосовували **методи** структурного і функціонального аналізу предметної області, об’єктно-орієнтоване моделювання з побудовою UML-діаграм, а також практичні підходи прототипування інтерфейсів із подальшим тестуванням на реальних користувачах. Інформаційною базою дослідження стали нормативні документи з організації фітнес-послуг, внутрішні регламенти клубу «Fitness Club», а також результати опитувань і аналітичні звіти щодо поведінки клієнтів у цифрових системах запису на тренування. Для валідації технічних рішень ми провели порівняльний аналіз існуючих систем управління у сфері фітнесу й обрали ті архітектурні патерни, які найкраще відповідають вимогам нашого проєкту.

Практична цінність одержаних результатів полягає в готовності розробленого програмного комплексу до впровадження в реальному бізнес-середовищі: система вже забезпечує повний цикл обробки даних від створення облікових записів до генерації фінансових та аналітичних звітів.

Структура роботи побудована логічно: у вступі обґрунтовано актуальність і визначено мету, завдання, об’єкт і предмет дослідження, методи та інформаційну базу. У першому розділі наведено аналіз вимог і вибір архітектури, другий розділ присвячено проєктуванню даних, алгоритмів і реалізації функціоналу, третій – опису деталей імплементації та тестування, а в завершальних висновках підбито підсумки й окреслено напрями подальшого розвитку системи.

За темою випускної кваліфікаційної роботи бакалавра було взято участь у 1 конференції та опубліковано тези [29], [30].

					ІПЗ.КР.Б –121 – 25 – ПЗ	
						8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ СИСТЕМИ УПРАВЛІННЯ ФІТНЕС-КЛУБОМ

1.1. Постановка задачі

Сучасна індустрія фітнесу стрімко розвивається, дедалі більше людей стають прихильниками здорового способу життя, що спричиняє підвищення попиту на якісні фітнес-послуги. Водночас, збільшення кількості клієнтів створює додаткове навантаження на адміністративний персонал фітнес-клубів, що зумовлює необхідність впровадження ефективних автоматизованих інформаційних систем управління. Автоматизація процесів управління дозволяє покращити якість надання послуг, зменшити кількість помилок та підвищити рівень задоволеності клієнтів.

Однією з основних проблем сучасних фітнес-клубів є використання застарілих методів адміністрування: ручне ведення обліку абонементів, клієнтів, розкладів тренувань та взаємодії з клієнтами. Це призводить до великої кількості помилок, неефективного використання робочого часу персоналу, і як наслідок, до зниження конкурентоспроможності клубу. Ручне адміністрування не дозволяє швидко і гнучко адаптуватися до змін у роботі клубу, створює незручності для клієнтів через затримки з обслуговуванням та погіршує загальний імідж клубу.

Використання веб-орієнтованих інформаційних систем для управління діяльністю фітнес-клубів є ефективним рішенням зазначених проблем. Такі системи дозволяють автоматизувати ведення клієнтської бази, управління абонементом, планування та контроль тренувань, автоматизувати розсилку сповіщень та повідомлень клієнтам, а також забезпечувати ефективну комунікацію між клієнтами, тренерами та адміністративним персоналом. Це значно спрощує та прискорює процеси управління, дозволяє мінімізувати ризик помилок та підвищує рівень задоволеності клієнтів якістю послуг.

З огляду на ці переваги, впровадження автоматизованої системи управління фітнес-клубом є доцільним кроком у розвитку фітнес-клубу. Вибір сучасних інформаційних технологій, забезпечує високу продуктивність,

надійність і масштабованість системи. Також важливим є використання об'єктно-орієнтованого програмування та методології UML для проектування та розробки системи, що дозволяє створити зрозумілу та гнучку структуру програмного забезпечення.

Таким чином, актуальність розробки системи управління фітнес-клубом визначається необхідністю автоматизації управлінських процесів та покращення якості обслуговування клієнтів. Використання сучасних технологій розробки веб-додатків та баз даних дозволить створити ефективну і гнучку систему, що задовольнить потреби як персоналу, так і клієнтів фітнес-клубу.

Одним із ключових напрямів проектування стало впровадження багаторівневої авторизації з чітким розмежуванням ролей користувачів: клієнт, тренер і адміністратор. Такий підхід забезпечує контрольований доступ до функціональних можливостей системи, дозволяє підтримувати безпеку персональних даних та сприяє ефективному використанню кожним користувачем саме тих інструментів, які відповідають його ролі. Клієнти можуть переглядати та редагувати особисту інформацію, знайомитися з доступними абонементом, оформлювати їх придбання, а також самостійно записуватися на тренування. Тренерам надано можливість створювати групові та індивідуальні тренування, переглядати списки записаних учасників і керувати актуальним розкладом. Адміністративна частина включає повне управління обліковими записами, налаштування розкладу, створення та редагування типів підписок, а також перегляд статистичних і фінансових даних, необхідних для аналітики та стратегічного управління діяльністю клубу.

Особливу увагу приділено збереженню, узгодженості й цілісності даних. Для реалізації цих вимог було обрано Microsoft Azure SQL Database — потужну систему управління базами даних, що забезпечує високу продуктивність, підтримку транзакцій, резервне копіювання, а також горизонтальне й вертикальне масштабування. Це дозволяє обробляти як типові, так і складні запити з гарантованою стабільністю, що є критично важливим у системах із

великою кількістю взаємопов'язаних сутностей, таких як абонементи, користувачі, відвідування та тренування.

До функціональних компонентів системи включено модулі автоматичного відстеження статусів абонементів, які забезпечують контроль за їхнім поточним станом. Це дає змогу фіксувати, чи активований абонемент, скільки відвідувань залишилося, чи перебуває він у статусі «заморожений», або чи завершився його термін дії. Вся інформація оновлюється у режимі реального часу і відображається як клієнтам, так і адміністраторам. Такий механізм дозволяє уникнути непорозумінь щодо доступу до тренувань, автоматично обмежувати або розширювати можливості користувачів залежно від поточного статусу підписки, а також формувати базу для подальшого аналізу використання абонементів у розрізі типів, періодів та ефективності.

Для зручності взаємодії з системою було реалізовано адаптивний веб-інтерфейс на основі ASP.NET MVC. Його структура передбачає коректне відображення на різних пристроях — від настільних комп'ютерів до мобільних телефонів. Завдяки зрозумілій навігації, логічній організації контенту та мінімалістичному дизайну інтерфейс є простим у використанні, що знижує потребу в додатковому навчанні персоналу та мінімізує кількість звернень до технічної підтримки.

Окрім вищезгаданого, система дозволяє забезпечити цілісне представлення усіх взаємозв'язків між сутностями, такими як клієнти, тренування, абонементи та відвідування, що створює єдину керовану інформаційну модель. Завдяки цьому адміністрація отримує повний контроль над усіма бізнес-процесами в режимі реального часу, що сприяє підвищенню рівня прозорості та ефективності управління фітнес-клубом.

Сформована архітектура системи та реалізовані функціональні рішення дозволяють забезпечити надійну, безпечну та масштабовану основу для управління фітнес-клубом. Впровадження цієї платформи створює передумови для ефективної автоматизації щоденних бізнес-процесів, підвищення якості

обслуговування клієнтів і досягнення стабільних конкурентних переваг на ринку фітнес-послуг.

1.2. Аналіз аналогів програмного продукту

У рамках даної дипломної роботи було проведено детальний аналіз вже існуючих аналогів програмного продукту для фітнес-клубів. Розглянуто три найбільш показових приклади з точки зору популярності, доступного функціоналу та реалізації інтерфейсів: мобільний додаток мережі клубів Sport Life, платформа Beauty Pro для фітнес-клубу Wings Fitness, а також комплексне рішення LuckyFit (CRM-система та мобільний додаток).

Першим розглянутим продуктом є мобільний додаток Sport Life [1]. Цей додаток позиціонується як зручний інструмент для відвідувачів клубу, надаючи можливості бронювання тренувань з персональними тренерами або запису на групові заняття (рис.1.1).

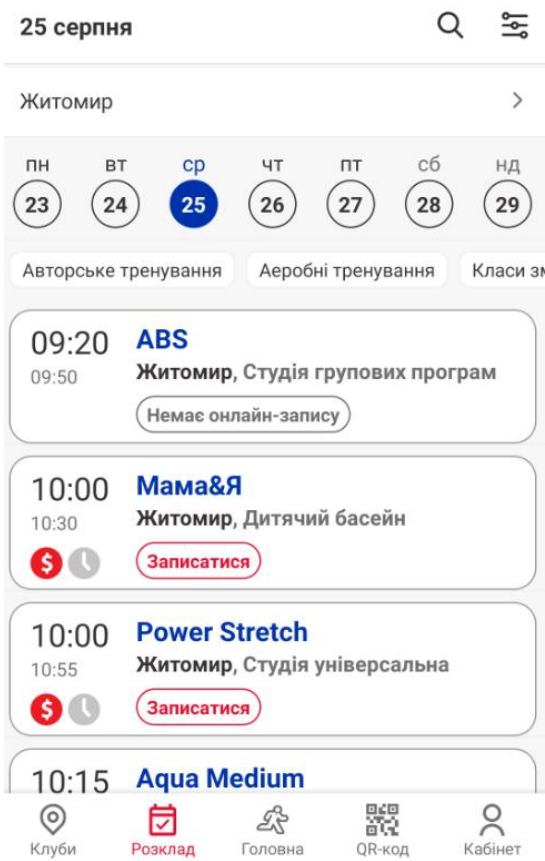


Рис. 1.1 – Сторінка розклад у додатку Sport Life

Також користувачам доступна функція оплати послуг клубу безпосередньо через інтегрований електронний гаманець, отримання різноманітних промо-пропозицій і знижок, а також перегляд поточного розкладу клубу (рис. 1.2).

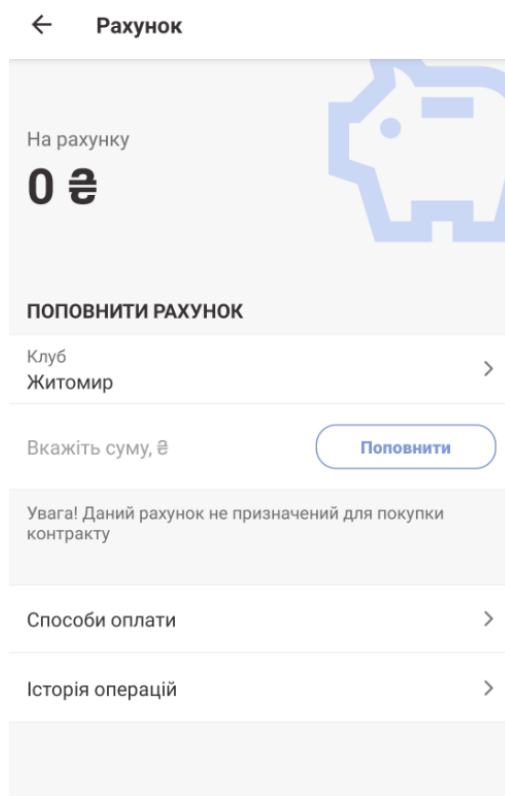


Рис. 1.2 – Сторінка рахунок у додатку Sport Life

Однак у межах проведеного аналізу, було виявлено серйозні обмеження. Найбільш очевидною вадою є відсутність можливості придбання абонементів через додаток, що суттєво обмежує зручність клієнтів та збільшує адміністративне навантаження на менеджерів клубу. Крім того, важливим функціоналом для зручності користувачів є історія записів на тренування, яка також не представлена належним чином у цьому додатку. Інформація щодо відвіданих занять або їх скасування не зберігається і не доступна клієнтам для перегляду в подальшому. Це створює певні незручності, адже клієнтам доводиться звертатися безпосередньо до адміністрації клубу для уточнення подібної інформації. Ще одним суттєвим недоліком є той факт, що інтерфейс клієнта реалізований виключно у вигляді мобільного додатку. Відсутність можливості використання веб-браузера змушує всіх користувачів встановлювати

додаток, що може бути неприйнятним для певної аудиторії, яка бажає зберігати місце на пристрої або не хоче витрачати час на додаткову авторизацію.

Другим програмним продуктом для порівняння є рішення від компанії Beauty Pro, яке було інтегровано у фітнес-клуб Wings Fitness [2]. В першу чергу, варто зазначити, що початково ця платформа розроблялася для салонів краси і мала функціонал, орієнтований на запис клієнтів на процедури та ведення внутрішньої бази клієнтів з історією їхніх відвідувань. У процесі адаптації для фітнес-клубу Wings Fitness додаток не був суттєво модифікований, що призвело до низки серйозних обмежень. Зокрема, відсутній модуль продажу та управління абонементом, що є ключовим елементом будь-якого професійного рішення для фітнес-клубів. Окрім цього, клієнти не мають можливості зручно сортувати заняття за тренерами або категоріями, що значно ускладнює пошук необхідних тренувань та навантажує користувачів додатковими діями (рис.1.3).

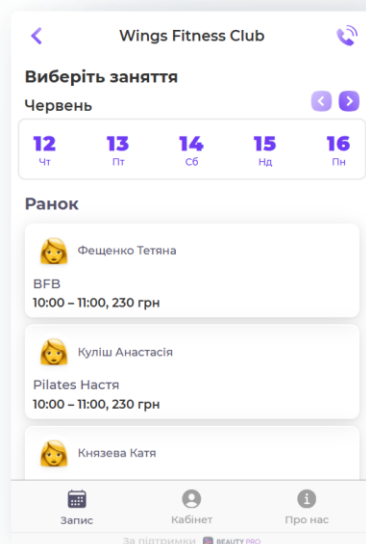


Рис. 1.3 – Сторінка запису у веб-додатку Wings Fitness Club

Ще одним значним мінусом є те, що додаток не відображає інформацію про кількість доступних місць на заняттях, тому клієнти не можуть оперативно оцінити зайнятість конкретних групових занять. Крім того, в системі не представлена публічно доступна CRM-частина для менеджерів, що вказує на ймовірність суттєвих обмежень і в менеджерському функціоналі. На відміну від Sport Life, взаємодія клієнтів із системою Wings Fitness через Beauty Pro

здійснюється виключно у веб-форматі. Проте веб-версія розроблена виключно під мобільні пристрої, і не має повноцінної адаптації під десктопні екрани. Це означає, що при відкритті веб-версії на комп'ютері користувач бачить сторінку у формі, яка відповідає розмірам і пропорціям мобільного телефону. Таке рішення є незручним для користувачів, які надають перевагу роботі з персональних комп'ютерів чи ноутбуків, оскільки вони змушені працювати з обмеженим по ширині інтерфейсом, який не використовує переваг великих екранів. Через таку недостатню адаптивність користувачі позбавлені зручності одночасного перегляду значної кількості інформації або швидкого доступу до додаткових функцій, що значно обмежує загальну зручність використання веб-продукту та негативно впливає на користувацький досвід.

Третім, і найбільш комплексним рішенням серед розглянутих аналогів є CRM-система LuckyFit [3], яка також доповнюється власним мобільним додатком (рис 1.4).

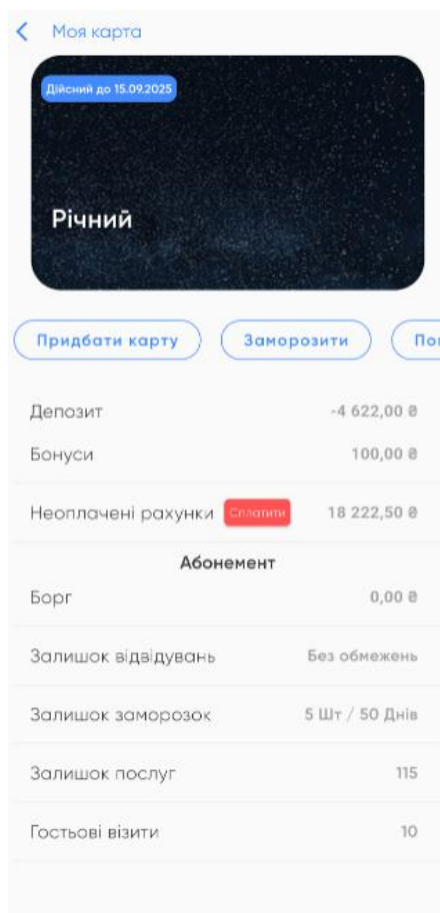


Рис. 1.4 – Сторінка «Моя карта» у додатку LuckyFit

Система LuckyFit орієнтована на повний цикл управління фітнес-клубом і забезпечує не тільки зручний клієнтський інтерфейс, а й потужний інструментарій для адміністраторів та менеджерів (рис. 1.5).

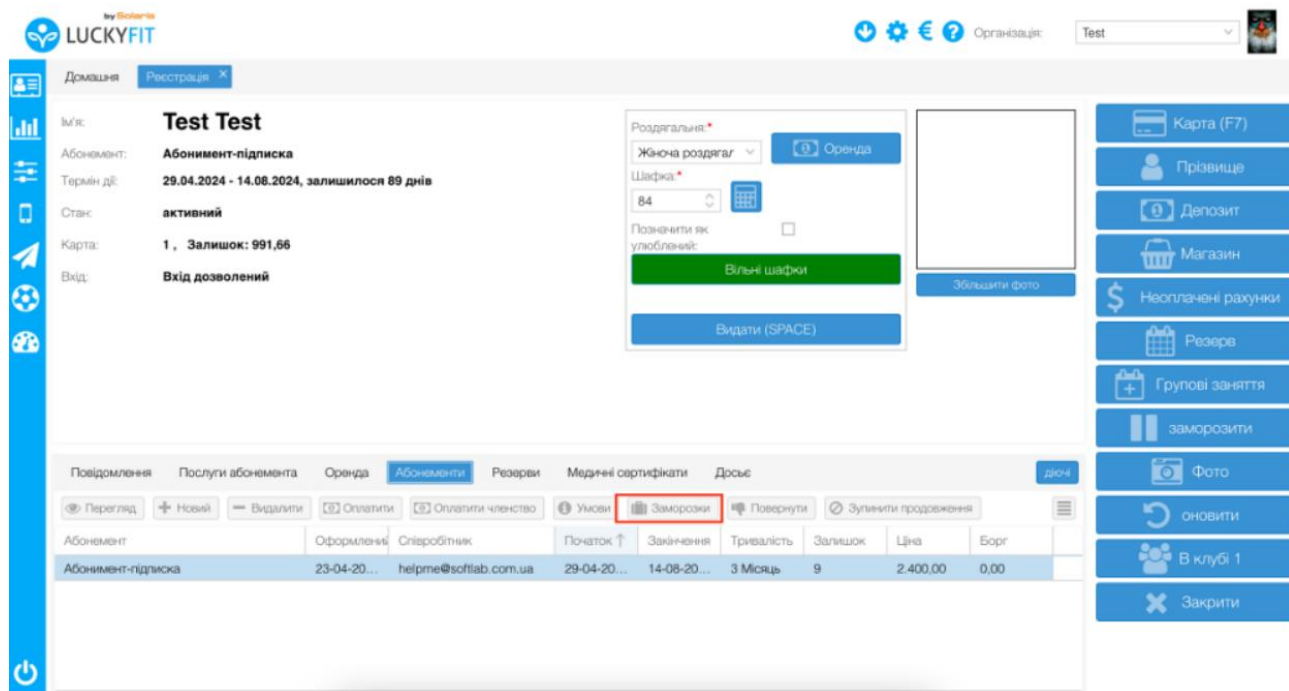


Рис. 1.5 – Сторінка створення абонементу у системі LuckyFit

Серед ключового функціоналу LuckyFit варто відзначити можливість продажу та управління абонементами, гнучку систему бронювання занять і ресурсів клубу, а також інтеграцію з фізичними пристроями, такими як турнікети для контролю доступу.

Ще однією сильною стороною LuckyFit є наявність повноцінного бек-офісу у вигляді CRM-платформи для адміністраторів, яка надає доступ до звітів, аналітики та детального управління роботою клубу. Незважаючи на широкий функціонал, LuckyFit також має певні обмеження. Зокрема, для клієнтів єдиним доступним каналом взаємодії залишається мобільний додаток, тоді адміністративному персоналу доступна лише десктоп-версія, що може створювати додаткові незручності для окремих категорій користувачів.

Для підсумку порівняння вище описаних застосунків було створено порівняльну таблицю (табл. 1.1).

Порівняння аналогів застосунку

Платформа	Sport Life	Wings Fitness / Beauty Pro	LuckyFit
Функціонал для клієнта	Бронювання занять, платежі, промо-акції	Запис на групові та індивідуальні послуги	Купівля абонементів та послуг, бронювання, нагадування.
Продаж абонементів	Ні	Ні	Так
Історія записів, доступна клієнту	Ні	Так	Так
Інформація про вільні місця на тренуванні	Ні	Ні	Так
Платформи взаємодії для клієнта	Мобільний додаток	Веб-застосунок у мобільній версії	Мобільний додаток
Переваги	Широкі маркетингові інструменти, е-грошовий гаманець	Інтеграція з салонною CRM Beauty Pro	Повний цикл: продаж, записи, аналітика, інтеграції (турнікети)
Недоліки	Відсутні абонементи та історія, залежність від моб. додатку	Обмежений функціонал, немає сортування та інформації, додаток єдиний канал	Клієнт обмежений мобільним додатком

Отже, проведений аналіз виявив як суттєві переваги, так і серйозні обмеження в розглянутих аналогах. Для майбутньої розробки програмного продукту важливо врахувати ці фактори і створити рішення, яке буде максимально зручним як для клієнтів, так і для адміністративного персоналу, пропонуючи належний спектр функціоналу без необхідності обов'язкового встановлення мобільного додатку чи десктоп програми, а з можливістю зручної роботи через веб-інтерфейс.

1.3. Вибір архітектури веб-орієнтованої системи управління фітнес-клубом

При розробці системи управління фітнес-клубом «Fitness Club» ключовими критеріями стали необхідність забезпечити високу продуктивність, надійність і гнучкість масштабування. У результаті аналізу предметної області було визначено, що система має складатися із трьох основних логічних рівнів: клієнтського інтерфейсу, серверної бізнес-логіки, а також підсистеми збереження та обробки даних. Трирівнева архітектура надає можливість чітко розмежувати функціональні обов'язки між компонентами, спростити підтримку, майбутній розвиток системи, а також забезпечити незалежне масштабування кожного шару залежно від навантаження та потреб клубу.

Клієнтська частина реалізована за допомогою Razor Pages [4] в рамках платформи ASP.NET MVC [5]. Вона відповідає за відображення даних, взаємодію з користувачами та первинну валідацію введених даних. Використання архітектурного патерну MVC забезпечує чітке розмежування представлення від бізнес-логіки й значно полегшує внесення змін до користувацького інтерфейсу. Крім того, цей підхід дозволяє зручно масштабувати систему, включаючи можливість у майбутньому додавати мобільні або односторінкові (SPA) версії застосунку без серйозних змін у серверній частині.

Серверна бізнес-логіка побудована на основі контролерів ASP.NET MVC, які обробляють HTTP-запити, виконують авторизацію та маршрутизацію, координують виклики сервісів та при потребі формують відповіді у форматі JSON [6]. Вся бізнес-логіка інкапсульована у виділених сервісних класах, що дозволяє підтримувати принцип єдиної відповідальності — Single Responsibility Principle [7].

Для рівня збереження даних обрано Microsoft Azure SQL Database [8], що обумовлено високими вимогами до цілісності, швидкодії та надійності. Azure SQL забезпечує виконання складних запитів, гарантує транзакційну безпеку та резервне копіювання, що особливо важливо для коректного збереження історії

записів, розкладів тренувань і даних клієнтів. Використання цієї бази даних дозволяє легко масштабувати систему як вертикально (збільшуючи потужність існуючих серверів), так і горизонтально (додаючи нові вузли до кластеру), що забезпечує стабільну роботу при зростанні навантаження.

Для розгортання веб-застосунку було обрано сервіс Microsoft Azure App Service (WebApp), який забезпечує швидке та масштабоване розміщення додатків без необхідності самотійного налаштування серверів [9]. Цей хмарний сервіс дозволяє створити надійне середовище виконання з високим рівнем доступності, автоматичним управлінням оновленнями інфраструктури та підтримкою SSL, що важливо для забезпечення захисту персональних даних користувачів системи. Azure App Service підтримує безперервну інтеграцію та доставку (CI/CD) безпосередньо з репозиторіїв GitLab, що дозволяє реалізувати автоматичне розгортання застосунку після кожного оновлення коду [10]. Такий підхід значно скорочує час на доставку нових версій і знижує ризик помилок, пов'язаних з ручним оновленням. Крім того, розробник має змогу гнучко керувати середовищем, переглядати журнали подій, використовувати інструменти моніторингу продуктивності та оперативно реагувати на можливі збої в роботі системи.

При виборі архітектурних патернів було розглянуто кілька варіантів: монолітна архітектура, мікросервіси та сервісно-орієнтована архітектура (SOA). У результаті було прийнято рішення реалізувати систему за монолітною трирівневою архітектурою MVC із чітко виділеними модулями: контролерами, сервісами та репозиторіями. Це рішення оптимально підходить для етапу розробки проєкту, оскільки спрощує розгортання, налаштування та тестування системи. Водночас обрана архітектура передбачає можливість поступової трансформації до мікросервісної структури у майбутньому за необхідності.

З метою підвищення модульності, зручності підтримки коду та розділення доступу до даних, у проєкті реалізовано шаблон проектування «Репозиторій» (Repository Pattern) [11]. Цей патерн забезпечує абстракцію над джерелом даних,

що дозволяє гнучко взаємодіяти з базою та легко замінювати або розширювати логіку доступу до даних без порушення загальної структури сервісів.

Таким чином, обрана архітектура забезпечує:

- чітке розмежування відповідальностей між шарами;
- високу масштабованість та продуктивність системи;
- гнучкість завдяки застосуванню архітектурного підходу MVC та патерну «Репозиторій»;
- надійність та безпечність зберігання і обробки даних;
- простоту подальшого розвитку та інтеграції нових компонентів і функцій.

Цей підхід дозволить у майбутньому швидко адаптувати систему до нових вимог і забезпечить стабільну роботу та високу якість сервісу для всіх клієнтів і співробітників «Fitness Club».

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Сучасна веб-розробка вимагає використання ефективних, надійних і гнучких інструментальних засобів, що забезпечують створення структурованих, розширюваних і підтримуваних програмних продуктів. При розробці системи управління фітнес-клубом «Fitness Club» було здійснено ретельний аналіз наявних технологій і платформ, в результаті чого було сформовано оптимальний технологічний стек.

Під час проєктування архітектури системи управління фітнес-клубом важливим кроком стало обґрунтування вибору інструментів для створення веб-інтерфейсу. Сучасна веб-розробка вимагає використання ефективних технологій, що забезпечують швидке створення якісних програмних продуктів із чіткою структурою, високою продуктивністю та можливістю подальшої підтримки. У межах платформи .NET одним із найбільш поширених підходів є ASP.NET MVC, який реалізує архітектурний шаблон Model-View-Controller. Цей патерн дозволяє чітко розділити застосунок на незалежні логічні компоненти, що значно полегшує розробку, модифікацію, тестування та супровід системи.

					ІПЗ.КР.Б –121 – 25 – ПЗ	
						20

Реалізація на основі ASP.NET MVC передбачає структурування коду на три чітко визначені шари: модель, представлення та контролер. Такий підхід забезпечує організованість коду, зручність його підтримки, спрощує масштабування й адаптацію додатку до нових бізнес-вимог. Однією з ключових переваг цієї архітектури є її висока тестованість — оскільки компоненти ізольовані, можна створювати юніт-тести для логіки, не залежачи від інтерфейсу чи бази даних. Окрім цього, ASP.NET MVC підтримує гнучку маршрутизацію, що дозволяє створювати логічні, зручні для користувачів та оптимізовані для пошукових систем URL-адреси. Повний контроль над HTML та JavaScript забезпечує гнучкість в побудові користувацького інтерфейсу, а можливість кешування даних і оптимізація запитів позитивно впливають на продуктивність.

Однак, як і будь-яка складна технологія, ASP.NET MVC має і свої недоліки. Перш за все, вона може виявитися складною для новачків, які не мають досвіду з MVC-патернами, адже вимагає розуміння взаємодії між контролерами, моделями та представленням. У порівнянні з такими технологіями як Razor Pages чи Web Forms, розробка на MVC потребує більше «ручної» роботи, зокрема при конфігурації маршрутів, розбитті логіки на компоненти та підтримці чистої архітектури. Також відсутнє автоматичне управління станом сторінки (ViewState), що змушує розробника самостійно передбачати механізми збереження контексту. І нарешті, складні системи на основі MVC потребують ретельного проєктування структури на ранніх етапах, оскільки помилки на рівні архітектури можуть ускладнити подальший розвиток системи [21].

Альтернативою ASP.NET MVC є Razor Pages — спрощена модель розробки веб-додатків у межах тієї ж платформи ASP.NET Core. Вона орієнтована на сторінково-центричний підхід, де вся логіка пов'язана безпосередньо з конкретною сторінкою, а не із загальними контролерами. Razor Pages добре підходить для простих CRUD-систем, де кожна сторінка відповідає окремому сценарію взаємодії. Її основною перевагою є швидкий старт, простота структури та зменшена кількість конфігурацій. Проте така архітектура має меншу гнучкість при масштабуванні та не так зручно підходить для складних

багаторівневих систем із розділенням ролей і контролем доступу до різних зон застосунку.

Для обґрунтування вибору було проведено системне порівняння між ASP.NET Core MVC та Razor Pages (табл. 1.2).

Таблиця 1.2

Порівняння ASP.NET Core MVC та Razor Pages

Критерій	ASP.NET Core MVC	Razor Pages
Архітектура	Побудована навколо контролерів і маршрутів	Побудована навколо сторінок із логікою в page-моделях
Структура проєкту	Чітке розділення на контролери, моделі й уявлення	Менш виражене розділення, код ближче до шаблону сторінки
Придатність для масштабування	Висока, зручна для великих багатомодульних систем	Обмежена, зручна для простих CRUD-застосунків
Гнучкість маршрутизації	Повна, підтримка атрибутів, кастомних маршрутів	Статична структура маршруту за шаблоном сторінки
Підтримка шаблонів	Легко реалізовує MVC-патерни, DI, фільтри, middleware	Також підтримує DI, але з меншим акцентом на масштабітність

З огляду на функціональні та архітектурні вимоги системи «Fitness Club», доцільним є вибір саме ASP.NET Core MVC як основної архітектурної моделі. Такий вибір зумовлений потребою у високій модульності, підтримці багатьох ролей із різними правами доступу, складній маршрутизації, можливості організувати масштабовану структуру з ізольованими компонентами. Крім того, ASP.NET MVC дає змогу краще інтегрувати механізми авторизації, кастомних фільтрів, middleware-компонентів та сервісних шарів — що є основою для ефективного контролю поведінки системи під навантаженням.

Для зберігання всієї інформації, яка обробляється в межах системи управління фітнес-клубом — включаючи дані про користувачів, абонементи, тренування, розклад, фінансові транзакції та історію відвідувань — необхідно використовувати надійну, структуровану систему керування базами даних. Оскільки всі ці об'єкти мають чіткі зв'язки між собою, мають бути забезпечені

цілісність, узгодженість і підтримка транзакцій, найкращим варіантом для таких вимог є реляційна база даних.

Реляційні СУБД забезпечують високий рівень структурованості, дозволяють встановлювати складні зв'язки між таблицями, підтримують нормалізацію, обмеження цілісності, транзакції з гарантією ACID-властивостей і, як правило, добре масштабуються у вертикальному напрямку. Враховуючи це, було прийнято рішення провести порівняння трьох найпопулярніших реляційних систем: Microsoft Azure SQL Database, PostgreSQL [12] та MySQL [13], аби визначити оптимальний варіант для обраної архітектури.

Таблиця 1.3

Порівняння реляційних СУБД

Характеристика	Microsoft Azure SQL Database	PostgreSQL	MySQL
Транзакції (ACID)	Повна підтримка	Повна підтримка	Повна підтримка
Масштабованість	Горизонтальна та вертикальна	Горизонтальна	Обмежена вертикальна
Інструменти резервування	Вбудовані, автоматичні	Через розширення	Частково зовнішні засоби
Підтримка Windows	Відмінна	Хороша	Хороша
Ліцензія	Пропрієтарна	Відкрита	Відкрита
Доступність (SLA)	99.99% з вбудованим failover	Залежить від хостингу	Залежить від хостингу
Безпека	Вбудований шифрування, RBAC, MFA	Залежить від конфігурації	Залежить від конфігурації
Продуктивність	Автоматичне масштабування, кешування, індекси	Висока при налаштуванні	Залежить від конфігурації

На основі порівняльного аналізу трьох провідних реляційних систем керування базами даних — Microsoft Azure SQL Database, PostgreSQL та MySQL — було зроблено висновок, що саме Microsoft Azure SQL Database найкраще відповідає вимогам, поставленим перед системою «Fitness Club». Ця керована хмарна платформа забезпечує повну підтримку ACID-транзакцій, високу

продуктивність, автоматичне резервне копіювання, гнучке масштабування та гарантовану доступність на рівні 99.99%. Крім того, вона має глибоку інтеграцію з .NET-платформою, що спрощує розробку та обслуговування. Завдяки цьому забезпечується надійна та безпечна обробка фінансових і персональних даних — критично важливий аспект для безперервної роботи системи управління фітнес-клубом.

У якості хостингового середовища для розгортання веб-застосунку було обрано Microsoft Azure App Service. Це рішення забезпечує зручне та масштабоване розміщення веб-додатків у хмарі з високим рівнем доступності та безпеки. Платформа дозволяє швидко налаштувати середовище виконання, автоматично обробляє оновлення інфраструктури та підтримує гнучке керування ресурсами без необхідності адміністрування серверів.

Однією з ключових переваг використання Azure є підтримка автоматичного розгортання з Git. Це дає змогу безпосередньо підключити репозиторій до Azure і налаштувати процес безперервної інтеграції та доставки (CI/CD). Усі зміни в коді можуть автоматично публікуватися на сервері, що значно скорочує час на оновлення застосунку та усуває ризики помилок при ручному розгортанні.

У якості основного інструментального середовища розробки було обрано Visual Studio 2022 [14]. Ця інтегроване середовище є офіційним інструментом для розробки під платформу .NET, надає повноцінну підтримку ASP.NET Core, T-SQL, а також включає в себе зручні засоби для налагодження, профілювання, тестування, керування пакетами NuGet, підключення до бази даних і візуального дизайну інтерфейсу [15]. Visual Studio також підтримує інтеграцію з системами контролю версій, зокрема GitLab, та CI/CD-сервісами, що дозволяє організувати повноцінний цикл розробки, тестування й розгортання без використання додаткових засобів.

Важливою перевагою є також наявність підтримки хмарних сервісів Azure, можливості запуску та налагодження багатопроєктних рішень одночасно, що значно спрощує процес розробки складних розподілених систем.

Висновки до розділу 1

У першому розділі дипломної роботи було проведено комплексне дослідження предметної області автоматизованих систем управління фітнес-клубами, визначено основні проблеми ручного адміністрування та сформульовано постановку задачі, що лягла в основу подальшого проектування системи «Fitness Club». Зокрема, проаналізовано ключові аспекти взаємодії між персоналом та клієнтами, механізми ведення розкладу тренувань, облік абонементів і платежів. Сформульовані цілі та завдання розробки передбачають впровадження гнучкої багаторольової системи з чітко визначеними повноваженнями користувачів, автоматизованого обліку підписок, тренувань і відвідувань, а також реалізацію адаптивного веб-інтерфейсу з фокусом на мобільну взаємодію.

У межах аналізу існуючих рішень було детально розглянуто функціональні можливості популярних програмних продуктів, що використовуються у фітнес-індустрії. Вивчення інтерфейсів, підходів до ведення обліку клієнтів і роботи з фінансовими даними дало змогу чітко сформулювати недоліки цих систем. Це дозволило виділити ключові вимоги до майбутньої системи та побудувати бачення конкурентних переваг.

Також у розділі було обґрунтовано вибір архітектурного підходу до реалізації системи. З урахуванням необхідності масштабованості, стабільності, безпеки, підтримки ролей, багатофункціональності та реалізації складних бізнес-процесів, було обрано трирівневу архітектуру на базі ASP.NET Core MVC.

Окрім цього, було обґрунтовано вибір технологічного стеку. Для зберігання структурованих даних і забезпечення транзакційної цілісності було обрано Microsoft Azure SQL Database — хмарну СУБД, яка дозволяє працювати з масштабованими структурами та має високу надійність. У якості середовища розробки використано Visual Studio 2022, що забезпечує зручне налагодження, тестування й інтеграцію з GitLab. Усі ці фактори створили надійне підґрунтя для реалізації сучасної, безпечної, стабільної та масштабованої системи управління фітнес-клубом.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ФІТНЕС-КЛУБОМ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Розробка системи управління фітнес-клубом передбачає використання різними категоріями користувачів. Основними варіантами використання є авторизація та управління обліковими записами, запис на тренування, проведення тренувань, управління абонементом, фінансові операції та автоматизація комунікацій.

Для кращого розуміння взаємодії користувачів із системою було створено діаграму варіантів використання (рис. 2.1), яка відображає основні функціональні можливості кожної ролі: клієнта, тренера, менеджера та адміністратора.

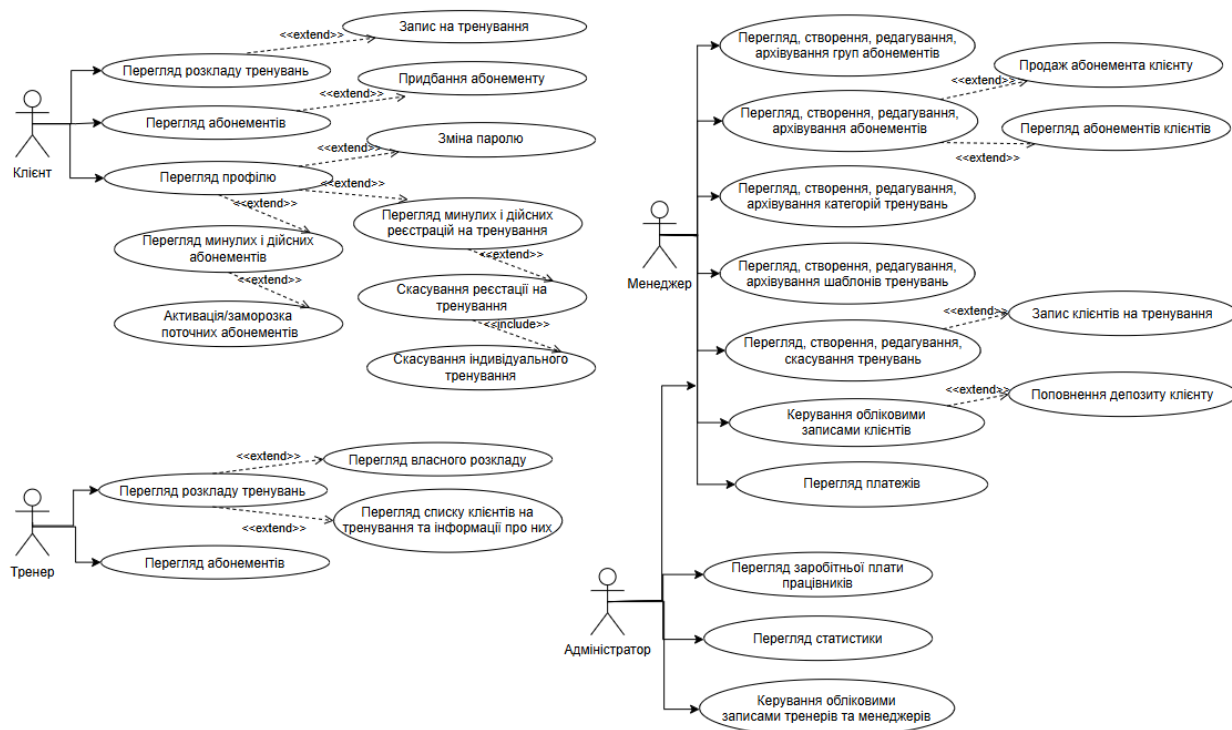


Рис. 2.1 – Діаграма варіантів використання системи управління
фітнес-клубом

Для глибокого розуміння функціональних можливостей системи та взаємодії користувачів із нею, було наведено опис окремих прецедентів згідно з діаграмою варіантів використання (табл. 2.1 – 2.11).

Таблиця 2.1

Опис прецеденту “Запис на тренування”

Назва варіанту використання	Запис на тренування
Основні актори	Клієнт
Короткий опис	Клієнт може переглядати розклад тренувань та здійснити запис на доступне групове або індивідуальне тренування. Запис можливий лише при активному абонементі.

Таблиця 2.2

Опис прецеденту “Купівля абонемента”

Назва варіанту використання	Купівля абонемента
Основні актори	Клієнт
Короткий опис	Клієнт може обрати та оплатити абонемент із доступних у системі. Система автоматично активує підписку згідно з умовами.

Таблиця 2.3

Опис прецеденту “Скасування реєстрації на тренування”

Назва варіанту використання	Скасування реєстрації на тренування
Основні актори	Клієнт
Короткий опис	Клієнт має можливість самостійно скасувати запис на тренування зі свого кабінету. У разі персонального тренування система автоматично скасовує тренування повністю, якщо клієнт був єдиним записаним учасником.

Таблиця 2.4

Опис прецеденту “Активація / заморозка поточного абонемента”

Назва варіанту використання	Активація / заморозка поточного абонемента
Основні актори	Клієнт, Менеджер, Адміністратор

Продовження таблиці 2.4

Короткий опис	Клієнт може активувати або тимчасово заморозити поточний абонемент у межах дозволених умов. Менеджер та Адміністратор мають змогу переглядати статуси і примусово змінювати їх за необхідності.
---------------	---

Таблиця 2.5

Опис прецеденту “Перегляд абонементів клієнтів”

Назва варіанту використання	Перегляд абонементів клієнтів
Основні актори	Менеджер, Адміністратор
Короткий опис	Користувачі можуть переглядати активні, завершені, заморожені або заблоковані абонементи, що пов’язані з конкретним клієнтом. Також можуть змінювати їх статус.

Таблиця 2.6

Опис прецеденту “Поповнення депозиту клієнту”

Назва варіанту використання	Поповнення депозиту клієнту
Основні актори	Менеджер, Адміністратор
Короткий опис	Менеджер та Адміністратор можуть вручну поповнити баланс клієнтського депозиту, наприклад, при оплаті готівкою чи картою в клубі. Інформація відображається у фінансовій історії.

Таблиця 2.7

Опис прецеденту “Продаж абонементу клієнту”

Назва варіанту використання	Продаж абонементу клієнту
Основні актори	Менеджер, Адміністратор
Короткий опис	Менеджер та Адміністратор мають можливість здійснити продаж абонементу клієнту через внутрішній інтерфейс системи, обравши тип абонементу, клієнта, розмір знижки та спосіб оплати.

Таблиця 2.8

Опис прецеденту “Перегляд списку клієнтів на тренування”

Назва варіанту використання	Перегляд списку клієнтів на тренування
Основні актори	Тренер, Менеджер, Адміністратор
Короткий опис	Менеджер та Адміністратор можуть переглядати перелік клієнтів, які записані на кожне з тренувань, а Тренер - тільки на власні.

Таблиця 2.9

Опис прецеденту “Запис клієнтів на тренування”

Назва варіанту використання	Запис клієнтів на тренування
Основні актори	Менеджер, Адміністратор
Короткий опис	Менеджер та Адміністратор мають змогу вручну записати клієнта на тренування з розкладу, міняючи стандартну онлайн-реєстрацію, що корисно при реєстрації через рецепцію або за попереднім запитом. Якщо тренування персональне – запис клієнта відбувається автоматично під час створення тренування.

Таблиця 2.10

Опис прецеденту “Перегляд статистики”

Назва варіанту використання	Перегляд статистики
Основні актори	Адміністратор
Короткий опис	Адміністратор може переглядати узагальнену статистику щодо кількості проведених тренувань різними тренерами, кількості записів і скасування відвідувань, а також скасування самих тренувань. Також інформація сортується за шаблонами тренувань та групами шаблонів.

Опис прецеденту “Керування обліковими записами”

Назва варіанту використання	Керування обліковими записами
Основні актори	Адміністратор
Короткий опис	Адміністратор може створювати, редагувати, переглядати та блокувати облікові записи користувачів всіх ролей.

Основна ціль розробки системи управління фітнес-клубом «Fitness Club» полягає у створенні комплексного веб-застосунку, що забезпечить автоматизацію ключових бізнес-процесів фітнес-центру та підвищить ефективність взаємодії між адміністрацією, тренерами та клієнтами. Система має на меті усунути недоліки ручного адміністрування, зменшити кількість помилок, оптимізувати щоденну роботу персоналу та забезпечити сучасний рівень сервісу для відвідувачів.

Основна мета проєкту — надати інструмент, який дозволить ефективно керувати абонементами, розкладом тренувань, записом клієнтів, фінансовими операціями та комунікацією між усіма сторонами. Завдяки впровадженню багаторівневої авторизації, гнучкого розподілу прав доступу, автоматизованого обліку та адаптивного інтерфейсу, система забезпечує прозоре керування клубом будь-якого масштабу — від невеликих студій до великих мережевих фітнес-центрів.

Функціональні вимоги до системи управління фітнес-клубом «Fitness Club» охоплюють повний спектр дій, які мають бути доступними для різних типів користувачів відповідно до їхніх ролей. Основна функціональність починається з процесу реєстрації та авторизації, що дозволяє кожному користувачеві створити персональний обліковий запис або увійти до системи з уже існуючими даними. Після входу система має автоматично визначати роль користувача (клієнт, тренер, менеджер чи адміністратор) та надавати доступ лише до відповідної функціональності. Кожен користувач повинен мати

можливість редагувати власний профіль, змінювати особисту інформацію та керувати налаштуваннями свого облікового запису.

Клієнтам надається можливість переглядати доступні абонементи, вивчати їх характеристики та оформляти купівлю онлайн з депозитних коштів. Система повинна забезпечити збереження історії всіх придбаних абонементів, відображення поточного статусу, кількості тренувань, що залишились, та терміну дії. У межах послуг клієнт має можливість переглядати розклад доступних тренувань, здійснювати запис на групові або персональні заняття, а також відмовлятися від участі у вже запланованих тренуваннях. У разі персонального тренування система повинна автоматично скасувати його, якщо клієнт скасував єдиний запис.

Менеджери повинні мати змогу керувати абонементами, формувати нові, змінювати існуючі параметри, визначати тривалість, кількість занять та доступні типи тренувань. Вони також мають адмініструвати продаж абонементів безпосередньо через панель керування, фіксуючи дані про покупця, застосовану знижку, дату та спосіб оплати. Крім того, менеджер повинен мати змогу поповнювати депозит клієнта, переглядати історію платежів і змінювати статус абонементів (активувати, заморожувати, завершувати достроково). Він має доступ до формування шаблонів тренувань, які використовуються для створення розкладу, а також керування групами — наприклад, за категоріями або рівнем складності.

Тренери мають працювати з власним розкладом, переглядати створені для них тренування, бачити списки записаних клієнтів. Додатково тренеру надається доступ до інформації про клієнтів, які відвідують його заняття — зокрема, їхнє ім'я, контактні данні та інформація про рівень підготовки та стан здоров'я клієнта.

Адміністратор отримує найширший спектр можливостей — він може створювати нові облікові записи, змінювати ролі користувачів, блокувати або розблоковувати акаунти. Йому доступна система глобальної аналітики: перегляд

статистики, даних про кількість активних підписок, відвідування занять, популярність окремих тренувань та ефективність роботи тренерів.

Крім цього, система має реалізовувати механізми логування дій з елементами системи, фіксації змін у базі даних, а також забезпечувати відповідну реакцію інтерфейсу на дії користувача. Кожна дія, що виконується у веб-інтерфейсі, повинна мати чіткий зворотний зв'язок у вигляді повідомлення про успішність чи помилку операції. Це дозволяє підтримувати високий рівень зручності, інтуїтивності та надійності використання системи для всіх учасників процесу.

В свою чергу система управління фітнес-клубом повинна відповідати переліку нефункціональних вимог, а саме: забезпечувати високу швидкодію, стабільність і зручність використання. Інтерфейс має бути інтуїтивно зрозумілим: нові користувачі повинні освоїти основні функції приблизно за 10 хвилин, а досвідчені — за 3–5 хвилин. Типові операції (перегляд розкладу, запис на тренування, оформлення абонементу) виконуються не довше ніж за 5 секунд, складні запити — до 10 секунд.

Нефункціональні вимоги системи передбачають стабільну роботу з гарантією середнього часу безвідмовного функціонування не менше 90 днів. Допустимий час недоступності — не більше 5% від загального часу, а рівень критичних помилок — не більше 3 на кожні 10 000 рядків коду. Платформа повинна стабільно функціонувати при одночасній роботі щонайменше 100 активних користувачів, забезпечуючи обробку запитів без зниження продуктивності.

Системні вимоги до розгортання та стабільної роботи системи управління фітнес-клубом визначають мінімальні та рекомендовані характеристики серверного та клієнтського обладнання, а також програмного забезпечення, необхідного для підтримки функціональності платформи.

Для серверної частини рекомендовано використовувати сервер із процесором не нижче Intel Xeon або AMD Ryzen з тактовою частотою від 3.0 GHz, оперативною пам'яттю обсягом не менше 16 GB і 64-розрядною

					ІПЗ.КР.Б –121 – 25 – ПЗ	
						32

операційною системою Windows 10 або новішою. Також важливим є стабільне Інтернет підключення, що забезпечить безперервну передачу даних і коректне функціонування всіх сервісів

Для клієнтської частини система має підтримувати роботу в сучасних браузерях без необхідності встановлення додаткового програмного забезпечення. Мінімальні вимоги для користувача — ПК або мобільний пристрій із доступом до інтернету, екраном від 5 дюймів, оперативною пам'яттю від 2 GB і стабільним браузером, оновленим не пізніше ніж за останні 6 місяців.

Один із ключових елементів структури додатку є побудова доступу до даних на основі патерна «Репозиторій». Усі сервіси системи, які працюють із базою даних, реалізовані через узагальнений інтерфейс репозиторію, що забезпечує уніфікований підхід до виконання CRUD-операцій, фільтрації, сортування та пагінації. На прикладі UserService продемонстровано, як відбувається взаємодія сервісів із репозиторіями в системі (рис. А.1).

У верхній частині діаграми зображено інтерфейс IRepository<TEntity>, який визначає набір універсальних CRUD-операцій (Create, Read, Update, Delete) та методів для отримання вибірок із пагінацією, сортуванням, фільтрацією тощо. Методи реалізовані з використанням узагальнень (<TEntity>) і функціональних виразів (Expression<Func<TEntity, ...>>), що дозволяє будувати запити до БД у гнучкий та типобезпечний спосіб.

Клас Repository<TEntity> реалізує інтерфейс IRepository<TEntity> та виконує фактичну роботу з контекстом бази даних через Entity Framework. Така реалізація дозволяє будь-якому сервісу, який потребує доступу до сутностей, використовувати єдиний репозиторій, налаштований для відповідного типу, без дублювання логіки запитів.

Нижній фрагмент діаграми демонструє інтерфейс IUserService та його реалізацію — клас UserService. Цей сервіс працює з користувачами системи та реалізує перевірку email і номера телефону, реєстрацію, блокування, розблокування та зміну пароля. Він використовує репозиторій IRepository<User> для взаємодії з таблицею користувачів у базі.

Усі сервіси доступу до даних у системі, включаючи сервіси для роботи з тренуваннями, клієнтами, оплатами, абонементами, тощо, реалізовані за цим принципом. Для кожного типу сутності створюється відповідний репозиторій через впровадження `IRepository<T>`, який у свою чергу додається у сервіс, що містить бізнес-логіку.

Таким чином, діаграма ілюструє не лише окрему реалізацію `UserService`, а й загальну архітектурну модель, яка застосовується до всієї системи. Патерн `Repository` разом з інтерфейсно-орієнтованим підходом формує надійний фундамент для реалізації розширюваних і підтримуваних сервісів у рамках проєкту.

2.2. Розробка бази даних системи

У рамках проєкту розробки системи управління фітнес-клубом було попередньо обрано використання хмарної бази даних `Azure SQL`, що забезпечує високу надійність, масштабованість і доступність даних у будь-який час. Це рішення обумовлене потребою в централізованому зберіганні інформації, доступній як для адміністративного персоналу, так і для клієнтів у режимі реального часу. Використання `Azure SQL` дозволяє легко інтегрувати систему з іншими сервісами екосистеми `Microsoft` та гарантує відповідність сучасним вимогам до безпеки й продуктивності.

Розробка структури бази даних виконується з використанням технології `Entity Framework Core` із підходом `Code First` [16]. Це означає, що усі сутності, зв'язки між ними та основні правила логіки визначаються безпосередньо в `C#`-моделях на етапі створення доменної моделі [17]. Такий підхід дозволяє максимально зосередитися на бізнес-логіці застосунку, автоматично синхронізуючи її з відповідною структурою бази даних за допомогою механізмів міграцій. `Code First` забезпечує високу гнучкість у модифікації структури БД під час еволюції проєкту, спрощує тестування та пришвидшує ітеративну розробку.

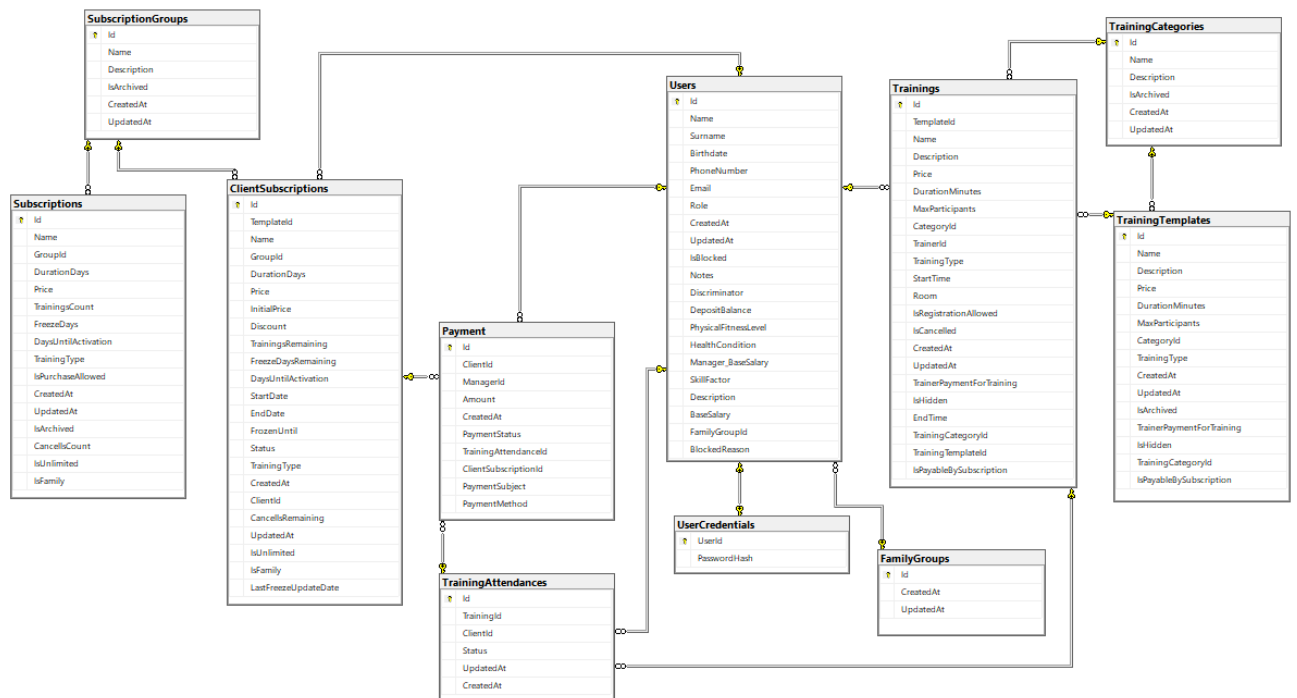


Рис. 2.2 – Діаграма бази даних системи управління фітнес-клубом

Таблиця Users є центральною сутністю, яка містить дані про всіх користувачів системи, включаючи клієнтів, тренерів, менеджерів і адміністраторів. У ній зберігається загальна інформація (ім'я, прізвище, контактні дані, роль) та додаткові поля, специфічні для окремих типів користувачів, такі як фізична підготовка, базова зарплата чи причина блокування. Зважаючи на ієрархію сутностей користувачів системи та наявність спільних полів, для зберігання було обрано підхід Table-per-Hierarchy (TPH), при якому всі типи користувачів зберігаються в одній таблиці Users, а їхня спеціалізація визначається за допомогою поля Discriminator [18]. Це дозволяє спростити структуру бази та забезпечити зручний доступ до загальних і специфічних атрибутів кожної ролі.

Таблиця UserCredentials відповідає за зберігання хешів паролів користувачів і реалізує механізм авторизації. Вона прив'язана до Users через зовнішній ключ і забезпечує безпечне зберігання облікових даних.

FamilyGroups використовується для об'єднання користувачів у сімейну групу. Кожен користувач може мати посилання на певну сімейну групу, що дає змогу реалізувати спільні абонементи або знижки для родини.

Таблиця Subscriptions містить шаблони абонементів, які можуть бути доступні для придбання клієнтами. Вона визначає ключові параметри такі, як: кількість занять, тривалість, ціна, тип тренування та можливість заморожування або безлімітного використання.

ClientSubscriptions зберігає персоналізовані абонементи, які були придбані конкретними клієнтами. Тут фіксуються дати початку та завершення, статус, залишок занять, а також інформація про застосовані знижки або заморозку.

Для структурування асортименту абонементів використовується таблиця SubscriptionGroups, яка дозволяє об'єднувати абонементи в групи за типом або цільовим призначенням.

Таблиця TrainingTemplates містить шаблони тренувань, на основі яких формуються конкретні події у розкладі. Вона включає опис, тривалість, кількість місць, ціну, тип тренування і визначає категорію, до якої належить тренування.

На основі шаблонів створюються записи в таблиці Trainings, яка описує реальні тренування з точною датою, часом, закріпленим тренером та інформацією про дозволену реєстрацію, скасування, оплату, тощо.

Категоризація тренувань реалізується через таблицю TrainingCategories, яка дозволяє розділяти тренування за напрямками. Вона використовується як у шаблонах, так і у конкретних тренуваннях.

Таблиця TrainingAttendances реєструє записи клієнтів на тренування. У ній фіксується статус відвідування, а також посилання на тренування, клієнта та історію платежів, що дозволяє відстежувати історію участі.

Для обліку фінансових операцій використовується таблиця Payment, яка зберігає інформацію про суми оплат, спосіб оплати, статус транзакції, пов'язані абонементи або тренування, а також за наявності менеджера, який здійснив операцію. Це дозволяє вести фінансовий облік діяльності фітнес-клубу.

Для кращого розуміння значення полів у вище згаданих таблицях, наведено їх опис (табл. 2.12 – 2.23)

Таблиця 2.12

Опис таблиці User

Поле	Тип даних	Опис
Id	uniqueidentifier	Унікальний ідентифікатор користувача
Name	nvarchar	Ім'я користувача
Surname	nvarchar	Прізвище користувача
Birthdate	date	Дата народження
PhoneNumber	nvarchar	Номер телефону
Email	nvarchar	Електронна пошта
Role	int	Роль користувача в системі
CreatedAt	datetime	Дата створення облікового запису
UpdatedAt	datetime	Дата останнього оновлення запису
IsBlocked	bit	Статус блокування користувача
Notes	nvarchar	Примітки адміністратора
Discriminator	nvarchar	Тип користувача в ієрархії (ТРН)
DepositBalance	decimal	Поточний залишок на депозиті (Клієнт)
PhysicalFitnessLevel	int	Рівень фізичної підготовки (Клієнт)
HealthCondition	nvarchar	Інформація про стан здоров'я (Клієнт)
Manager_BaseSalary	decimal	Базова зарплата (Менеджер)
SkillFactor	int	Оцінка навичок (Тренер)
Description	nvarchar	Опис (Тренер)
BaseSalary	decimal	Базова зарплата (Тренер)
FamilyGroupId	uniqueidentifier	Ідентифікатор сімейної групи (Клієнт)
BlockedReason	nvarchar	Причина блокування користувача

Таблиця 2.13

Опис таблиці UserCredentials

Поле	Тип даних	Опис
UserId	uniqueidentifier	Ідентифікатор користувача (зовнішній ключ до Users)
PasswordHash	nvarchar	Хешований пароль користувача

Таблиця 2.14

Опис таблиці FamilyGroups

Поле	Тип даних	Опис
Id	uniqueidentifier	Унікальний ідентифікатор сімейної групи
CreatedAt	datetime	Дата створення групи

UpdatedAt	datetime	Дата останнього оновлення
-----------	----------	---------------------------

Таблиця 2.15

Опис таблиці SubscriptionGroups

Поле	Тип даних	Опис
Id	uniqueidentifier	Унікальний ідентифікатор групи підписок
Name	nvarchar	Назва групи підписок
Description	nvarchar	Опис призначення групи
IsArchived	bit	Ознака архівування
CreatedAt	datetime	Дата створення
UpdatedAt	datetime	Дата останнього оновлення

Таблиця 2.16

Опис таблиці Subscriptions

Поле	Тип даних	Опис
Id	uniqueidentifier	Унікальний ідентифікатор шаблону абонементу
Name	nvarchar	Назва абонементу
GroupId	uniqueidentifier	Посилання на групу підписок
DurationDays	int	Тривалість у днях
Price	decimal	Ціна абонементу
TrainingsCount	int	Кількість тренувань
FreezeDays	int	Доступна кількість днів заморозки
DaysUntilActivation	int	Кількість днів до активації
TrainingType	int	Тип тренування, що підтримується
IsPurchaseAllowed	bit	Чи дозволена покупка клієнтом
IsUnlimited	bit	Ознака безлімітного абонементу
IsFamily	bit	Ознака сімейного абонементу
CreatedAt	datetime	Дата створення
UpdatedAt	datetime	Дата оновлення
CancelledCount	int	Максимальна кількість скасувань
IsArchived	bit	Ознака архівування

Таблиця 2.17

Опис таблиці TrainingCategories

Поле	Тип даних	Опис
Id	uniqueidentifier	Ідентифікатор категорії
Name	nvarchar	Назва категорії

Description	nvarchar	Опис
IsArchived	bit	Ознака архівованого стану
CreatedAt	datetime	Дата створення
UpdatedAt	datetime	Дата оновлення

Таблиця 2.18

Опис таблиці ClientSubscriptions

Поле	Тип даних	Опис
Id	uniqueidentifier	Унікальний ідентифікатор абонементу клієнта
TemplateId	uniqueidentifier	Посилання на шаблон абонементу
GroupId	uniqueidentifier	Посилання на групу підписок
Name	nvarchar	Назва підписки
DurationDays	int	Тривалість дії в днях
Price	decimal	Початкова ціна
Discount	decimal	Знижка
TrainingsRemaining	int	Залишок тренувань
FreezeDaysRemaining	int	Залишок днів заморозки
DaysUntilActivation	int	Залишок днів до активації
StartDate	datetime	Дата початку дії
EndDate	datetime	Дата завершення
FrozenUntil	datetime	Період заморозки
Status	int	Поточний статус підписки
TrainingType	int	Тип тренування
CancelledRemaining	int	Доступні скасування
ClientId	uniqueidentifier	Посилання на клієнта
CreatedAt	datetime	Дата створення
UpdatedAt	datetime	Дата оновлення
IsUnlimited	bit	Ознака безлімітності
IsFamily	bit	Чи є підписка сімейною
LastFreezeUpdateDate	datetime	Дата останньої зміни заморозки для відстеження в сервісі оновлення статусу абонементу

Таблиця 2.19

Опис таблиці TrainingTemplates

Поле	Тип даних	Опис
Id	uniqueidentifier	Ідентифікатор шаблону тренування
Name	nvarchar	Назва тренування
Description	nvarchar	Опис тренування
Price	decimal	Ціна

Продовження таблиці 2.19

DurationMinutes	int	Тривалість у хвиликах
MaxParticipants	int	Максимальна кількість учасників
CategoryId	uniqueidentifier	Категорія тренування
TrainingType	int	Тип тренування
CreatedAt	datetime	Дата створення
UpdatedAt	datetime	Дата оновлення
TrainerPaymentForTraining	decimal	Оплата тренеру
IsHidden	bit	Чи приховане для клієнтів
IsPayableBySubscription	bit	Чи доступне по абонементу

Таблиця 2.20

Опис таблиці Trainings

Поле	Тип даних	Опис
Id	uniqueidentifier	Ідентифікатор тренування
TemplateId	uniqueidentifier	Посилання на шаблон
Name	nvarchar	Назва тренування
Description	nvarchar	Опис
Price	decimal	Ціна
DurationMinutes	int	Тривалість
MaxParticipants	int	Макс. кількість учасників
TrainerId	uniqueidentifier	Тренер
StartTime	datetime	Початок
Room	nvarchar	Зал/приміщення
IsRegistrationAllowed	bit	Дозвіл на запис
IsCancelled	bit	Скасоване чи ні
CreatedAt	datetime	Дата створення
UpdatedAt	datetime	Дата оновлення
TrainerPaymentForTraining	decimal	Оплата тренеру
IsHidden	bit	Приховане для клієнтів
EndTime	datetime	Кінець
TrainingCategoryId	uniqueidentifier	Категорія
TrainingTemplateId	uniqueidentifier	Шаблон
IsPayableBySubscription	bit	Можна оплатити абонементом

Таблиця 2.21

Опис таблиці TrainingAttendances

Поле	Тип даних	Опис
Id	uniqueidentifier	Ідентифікатор відвідування
TrainingId	uniqueidentifier	Посилання на тренування

ClientId	uniqueidentifier	Посилання на клієнта
Status	int	Статус участі
CreatedAt	datetime	Дата створення
UpdatedAt	datetime	Дата оновлення

Таблиця 2.22

Опис таблиці Payments

Поле	Тип даних	Опис
Id	uniqueidentifier	Ідентифікатор платежу
ClientId	uniqueidentifier	Клієнт
ManagerId	uniqueidentifier	Менеджер
Amount	decimal	Сума платежу
CreatedAt	datetime	Дата створення
PaymentStatus	int	Статус платежу
TrainingAttendanceId	uniqueidentifier	Прив'язка до відвідування
ClientSubscriptionId	uniqueidentifier	Прив'язка до абонементу
PaymentSubject	int	Призначення платежу
PaymentMethod	int	Метод оплати

Таким чином, структура БД відповідає вимогам цілісності, продуктивності й гнучкості, закладаючи надійну основу для подальших операцій та аналітики.

2.3. Проектування та реалізація алгоритмів роботи системи

Система управління фітнес-клубом «Fitness Club» побудована за принципами багаторівневої архітектури, що включає клієнтський інтерфейс, серверну логіку (контролери та сервіси), а також рівень доступу до бази даних через репозиторії. Така структура забезпечує чітке розділення відповідальностей між компонентами, що дозволяє легко масштабувати, підтримувати й розширювати систему.

Взаємодія користувача із системою починається зі сторінок інтерфейсу, створених за допомогою ASP.NET MVC, де кожна дія користувача ініціює запити до відповідних контролерів. Контролери обробляють запити, здійснюють

авторизацію, валідацію вхідних даних і передають керування відповідним сервісам, які інкапсулюють бізнес-логіку системи.

Сервіси реалізують основну поведінку системи — наприклад, перевірку можливості запису на тренування, розрахунок залишку занять, автоматичне застосування заморозки, генерацію платіжних записів, зміну статусу підписки тощо. Усі зміни, що стосуються збереження або читання даних, відбуваються через репозиторії, які забезпечують взаємодію з базою даних у вигляді транзакцій, гарантуючи цілісність і узгодженість інформації.

Інформація в системі зберігається в реляційній базі даних Microsoft Azure SQL, структура якої сформована за допомогою Entity Framework Core за підходом Code First. Усі сутності мають чітко визначені зв'язки, які враховують бізнес-логіку клубу, зокрема: співвідношення клієнтів і підписок, тренерів і тренувань, платежів і відповідних послуг. Зміни у структурі бази відстежуються через механізм міграцій, що дозволяє синхронізувати модель домену з фактичним станом бази даних без втрати даних.

Доступ до функцій системи обмежується відповідно до ролі користувача. Роль визначається на етапі авторизації та зберігається у контексті сесії. Це дозволяє, наприклад, клієнту переглядати лише власні підписки та розклад, а менеджеру — керувати абонементом інших користувачів та продажами. Адміністратор має повний доступ до аналітики, списків користувачів і загальних налаштувань системи.

Загальна логіка системи передбачає гнучку обробку дій у режимі реального часу, а також автоматичне виконання певних сценаріїв (наприклад, автоматичне скасування тренування, якщо клієнт відмовився від персонального заняття, або автоматичне оновлення статусу підписки при завершенні її терміну дії). Така структура забезпечує стабільну роботу системи та високу швидкодію навіть при значному навантаженні.

Розглянемо декілька алгоритмів роботи детальніше.



Рис. 2.3 – Діаграма активності «Створення тренування»

Алгоритм створення тренування починається з авторизованого доступу користувача, який має роль менеджера або адміна. Після входу в систему менеджер переходить до відповідного розділу, де обирає шаблон тренування, який містить попередньо задані параметри — категорію, тривалість, назву, тип і максимальну кількість учасників.

Менеджер визначає дату та час тренування, при цьому інтерфейс автоматично блокує можливість вибору часу в минулому. Після встановлення бажаного часу клієнтська частина через AJAX-запит надсилає запит до контролера, який верифікує введені дані та передає запит у відповідний сервіс [19]. Сервіс звертається до бази даних і формує список доступних тренерів та вільних залів на вказаний період часу.

Результати повертаються назад у вигляді списку варіантів, з якого менеджер обирає тренера та зал. Після цього підтверджується створення

тренування, і сервіс зберігає новий запис у таблиці Trainings, встановлюючи всі відповідні зв'язки з шаблоном, тренером, залом і категорією.

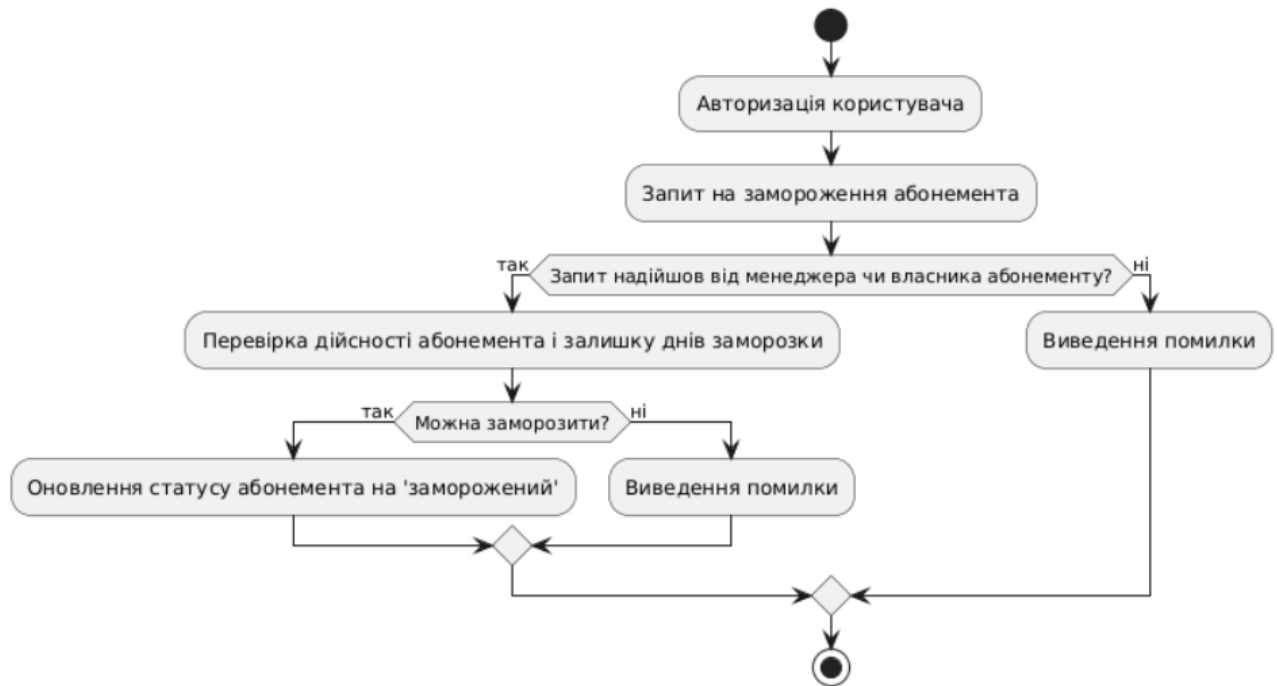


Рис. 2.4 – Діаграма активності «Заморозка абонементу»

Після входу в систему авторизований клієнт або менеджер ініціює заморожування активного абонементу. Контролер отримує запит і перевіряє, хто саме виконує дію — власник підписки чи менеджер клубу. Це важливо для перевірки прав доступу, оскільки лише менеджер має право заморожувати будь-яку підписку, тоді як клієнт — лише власну.

Далі контролер передає обробку в сервіс, який перевіряє, чи є абонемент дійсним, чи ще не завершився строк дії, і чи не перебуває він у стані заморозки. Також перевіряється, чи залишились доступні дні заморозки (FreezeDaysRemaining). У разі позитивного результату сервіс переводить підписку у стан заморозки, фіксує дату FrozenUntil та зменшує кількість доступних днів заморозки.

Цей процес супроводжується оновленням відповідного запису в ClientSubscriptions та поверненням підтвердження клієнту або менеджеру у вигляді сповіщення.

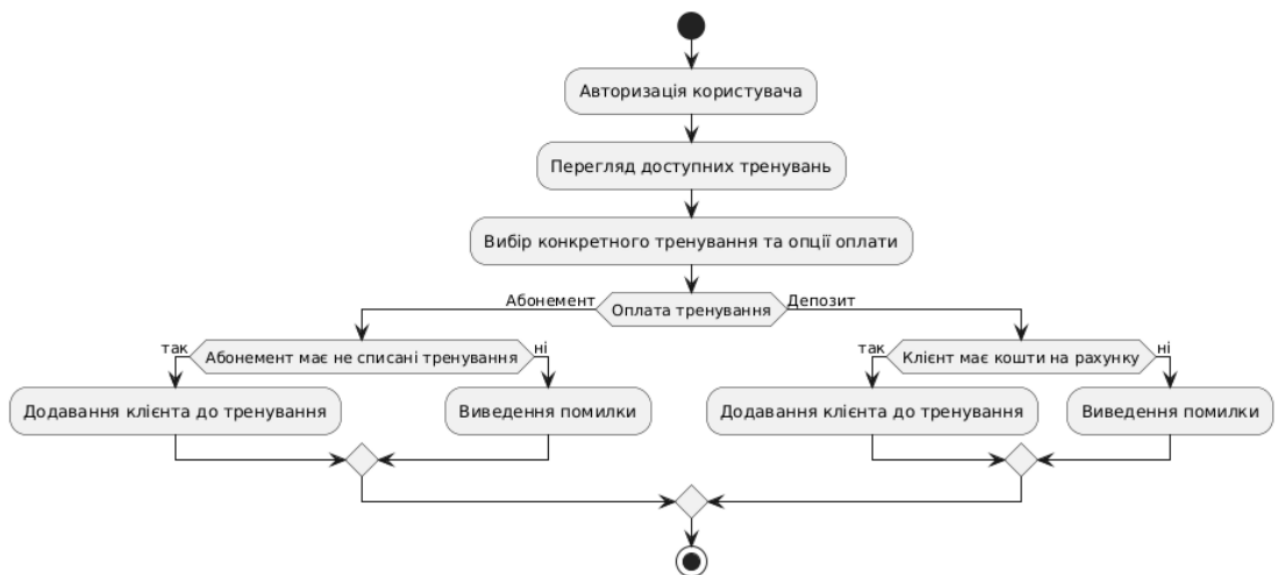


Рис. 2.5 – Діаграма активності «Запис на тренування»

Після авторизації в системі клієнт переглядає доступний розклад тренувань і обирає бажане. При спробі запису формується запит до контролера, який перевіряє, чи авторизований користувач має дійсний активний абонемент, що дозволяє відвідування обраного типу тренування.

Контролер викликає сервіс, який додатково перевіряє:

- чи не скасоване тренування;
- чи дозволена реєстрація (IsRegistrationAllowed);
- чи не перевищено максимальну кількість учасників;
- чи не має клієнт уже запису на це тренування.

У разі виконання всіх умов створюється запис у таблиці TrainingAttendances зі статусом Записаний. Кількість доступних тренувань у підписці зменшується (за винятком випадків з безлімітним абонементом), і клієнт отримує підтвердження про успішну реєстрацію.

Після входу в особистий кабінет клієнт обирає бажаний абонемент з доступного списку. Під час підтвердження купівлі система ініціює перевірку на наявність активних підписок цього ж типу. Якщо така вже існує, покупка блокується або користувач отримує попередження (рис.2.6).

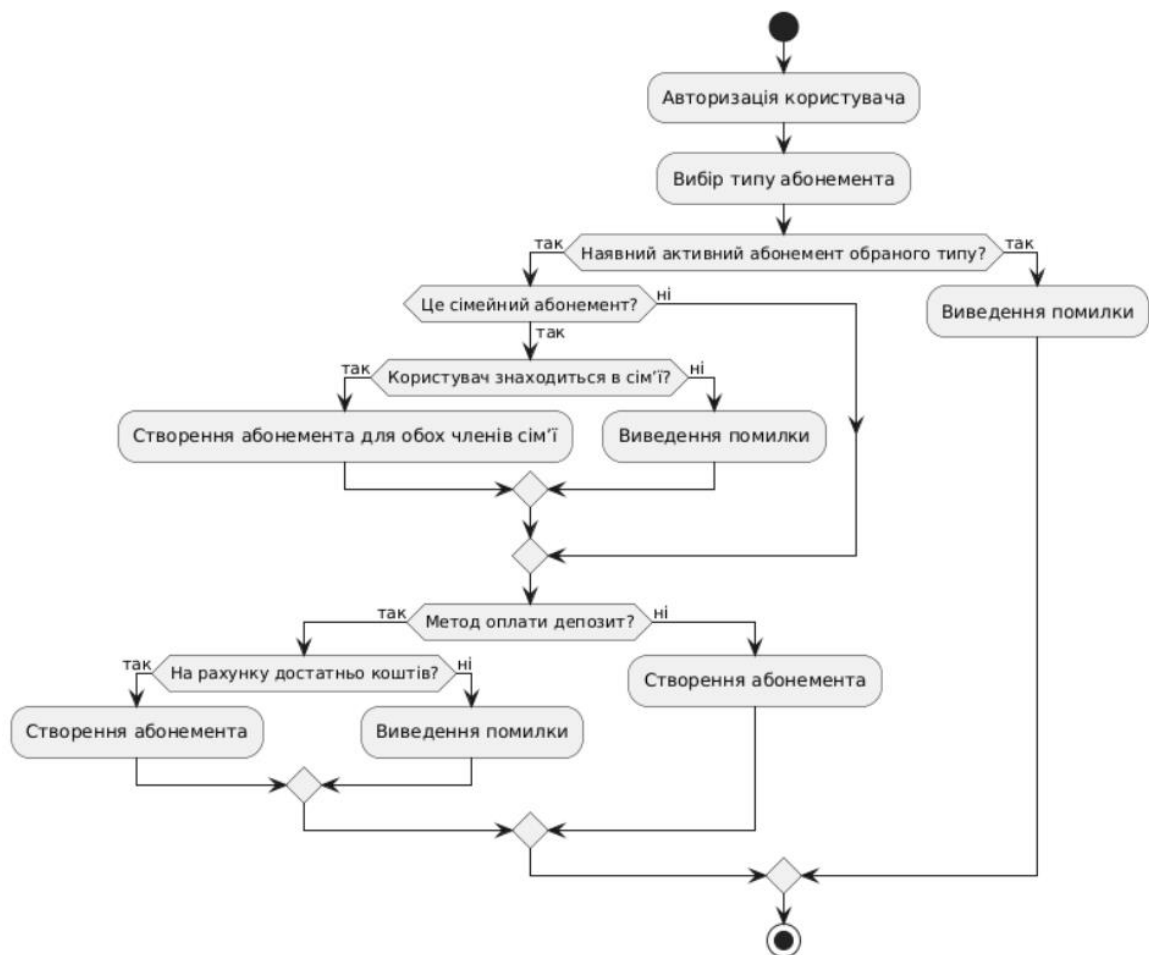


Рис. 2.6 – Діаграма активності «Придбання абонементу»

У разі покупки сімейного абонементу, сервіс перевіряє, чи належить клієнт до сімейної групи (FamilyGroupId). Якщо група існує, система автоматично створює другу копію абонементу для іншого учасника сім'ї з однаковими параметрами — ціна, тривалість, дата початку. Обидва абонементи мають між собою логічну прив'язку через тип і статус.

Далі створюється запис у таблиці ClientSubscriptions, фіксується платіж у таблиці Payment, і абонемент набуває статусу активного. Система автоматично враховує знижку, якщо вона застосовується, і за потреби оновлює фінансовий баланс клієнта.

2.4. Реалізація функціоналу системи управління фітнес-клубом

Розроблений програмний комплекс реалізований у вигляді веб-додатку на платформі ASP.NET Core з використанням MVC. Реалізація охоплює клієнтську, серверну логіку, механізми обробки даних та взаємодію з базою даних через Entity Framework.

Під час авторизації користувача в системі «Fitness Club» застосовується механізм аутентифікації на основі cookies, із використанням об'єктів Claims, які містять усю необхідну інформацію про користувача [20]. Це дозволяє не лише підтвердити автентичність користувача, а й визначити його роль і статус у системі після входу.

Після успішного введення даних електронної пошти та пароля, що проходять перевірку у відповідному сервісі, формується колекція claims — заяв про користувача, які надалі слугуватимуть його цифровим паспортом у межах сесії. Серед них фіксується унікальний ідентифікатор, email, роль у системі (Client, Trainer, Manager або Admin), ім'я та прізвище, а також статус блокування облікового запису:

```
var claims = new List<Claim>
{
    new Claim(ClaimTypes.NameIdentifier, user.Id.ToString()),
    new Claim(ClaimTypes.Email, user.Email),
    new Claim(ClaimTypes.Role, user.Role.ToString()),
    new Claim(ClaimTypes.Name, $"{user.Name} {user.Surname}"),
    new Claim("IsBlocked", user.IsBlocked.ToString()),
};
```

На основі цих claims створюється об'єкт ClaimsIdentity, який додається до ClaimsPrincipal, що ідентифікує користувача в усій системі. Цей об'єкт зберігається в cookies за допомогою HttpContext.SignInAsync [21]. Встановлюється час сесії: у 14 днів у разі вибору опції "Запам'ятати мене", або одна година при звичайному вході.

```
await HttpContext.SignInAsync("MyCookieAuth", principal, new
AuthenticationProperties
{
    IsPersistent = model.RememberMe,
    ExpiresUtc = model.RememberMe ? DateTime.UtcNow.AddDays(14) :
DateTime.UtcNow.AddHours(1)
});
```

В свою чергу перевірка даних входу в сервісі виглядає наступним чином:

```
var credential = await _userCredentialRepository.GetAsync(
    uc => uc.User.Email.ToLower() == email.ToLower(),
    [uc => uc.User]);

if (credential == null)
    return null;

var result = _passwordHasher.VerifyHashedPassword(
    credential.User, credential.PasswordHash, plainPassword);

return result == PasswordVerificationResult.Success ? credential.User : null;
```

Цей фрагмент коду відповідає за підготовку даних для відображення сторінки запису клієнта на тренування. Спочатку визначається кількість вже зареєстрованих учасників через сервіс відвідувань. Далі, якщо користувач є клієнтом, перевіряється: чи він має активний абонемент відповідного типу, чи може оплатити тренування з нього або через депозит (перевіряється баланс), а також чи вже зареєстрований на це тренування. Усі ці значення передаються у ViewBag, щоб адаптувати інтерфейс: показати доступні варіанти оплати, статус реєстрації та доступність запису [22].

```
var registeredCount = await
_trainingAttendanceService.CountRegisteredByTrainingIdAsync(id);
ViewBag.RegisteredCount = registeredCount;

var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);

bool isRegistered = false;
bool hasSubscription = false;
bool canPayFromSubscription = false;
bool canPayFromDeposit = false;

if (User.IsInRole("Client") && Guid.TryParse(userId, out var clientId))
{
    hasSubscription = await
_clientSubscriptionService.ClientHasCurrentSubscriptionAsync(
    clientId, training.TrainingType);
    if (hasSubscription)
    {
        canPayFromSubscription = (await
_clientSubscriptionService.GetCurrentSubscriptionByClientIdAsync(
        clientId, training.TrainingType)).Status == SubscriptionStatus.Active;
    }

    var client = await _clientService.GetByIdAsync(clientId);
    canPayFromDeposit = client?.DepositBalance >= training.Price;

    isRegistered = await _trainingAttendanceService.IsClientRegisteredAsync(id,
    clientId);
}
```

Цей код виконує пошук усіх тренерів, які не мають конфліктів у розкладі в заданий проміжок часу. Спочатку завантажуються всі не заблоковані тренери з бази даних. Потім для кожного з них виконується перевірка на наявність перетинаючих тренувань у вказаний період: якщо тренування не скасоване, його час перетинається з новим, і (за потреби) його ID не збігається з виключенням. Якщо конфліктів немає — тренер вважається доступним і додається до результату. У підсумку метод повертає список тренерів, які можуть провести нове тренування в обраному часовому інтервалі.

```
var trainers = await _trainerRepository.GetAllAsync(c => !c.IsBlocked);
var availableTrainers = new List<Trainer>();
foreach (var trainer in trainers)
{
    var hasConflictingTraining = await _trainingRepository.ExistsAsync(t =>
        t.TrainerId == trainer.Id &&
        !t.IsCancelled &&
        t.StartTime < endTimeUtc &&
        t.EndTime > startTimeUtc &&
        (!excludeTrainingId.HasValue || t.Id != excludeTrainingId.Value));
    if (!hasConflictingTraining)
    {
        availableTrainers.Add(trainer);
    }
}
```

Цей код реалізує логіку автоматичного оновлення статусів клієнтських абонементів у системі. На початку створюється новий скоуп для доступу до необхідного сервісу IClientSubscriptionService. Потім завантажуються всі активні або поточні абонементи клієнтів.

Далі цикл проходить по кожному абонементу й виконує кілька ключових перевірок. Якщо абонемент наразі заморожений (FrozenUntil > today), система щодня зменшує кількість доступних днів заморозки та подовжує кінцеву дату абонементу на один день, щоб компенсувати заморожений період. Коли дні заморозки вичерпано, поле FrozenUntil обнуляється, що дозволяє автоматично повернути абонемент у звичайний стан.

Після цього визначається актуальний статус абонементу. Якщо ще не настала дата початку — статус ToBeActivated. Якщо дата закінчення вже минула — статус Expired. Якщо триває період заморозки — статус Frozen. В інших випадках — абонемент активний (Active).

Якщо в процесі виконання відбулись зміни в статусі або був оброблений день заморозки, абонемент оновлюється в базі. Такий підхід дозволяє системі щоденно актуалізувати статуси всіх абонементів без втручання користувача.

```
private async Task UpdateStatusesAsync()
{
    using var scope = _scopeFactory.CreateScope();
    var clientSubscriptionService =
scope.ServiceProvider.GetRequiredService<IClientSubscriptionService>();

    var clientSubscriptions = await clientSubscriptionService.GetAllCurrentAsync();
    var now = TimeZoneInfo.ConvertTimeFromUtc(DateTime.UtcNow, UkrainianTimeZone);
    var today = now.Date;

    foreach (var s in clientSubscriptions)
    {
        var originalStatus = s.Status;
        var freezeWasProcessed = false;

        if (s.FrozenUntil.HasValue && s.FrozenUntil > today)
        {
            if (!s.LastFreezeUpdateDate.HasValue ||
s.LastFreezeUpdateDate.Value.Date < today)
            {
                s.FreezeDaysRemaining = Math.Max(0, (s.FreezeDaysRemaining ?? 0) -
1);

                s.LastFreezeUpdateDate = today;
                s.EndDate = s.EndDate.AddDays(1);
                freezeWasProcessed = true;

                if (s.FreezeDaysRemaining == 0)
                {
                    s.FrozenUntil = today;
                }
            }
        }
        if (today < s.StartDate)
        {
            s.Status = SubscriptionStatus.ToBeActivated;
        }
        else if (today > s.EndDate)
        {
            s.Status = SubscriptionStatus.Expired;
        }
        else if (s.FrozenUntil.HasValue && s.FrozenUntil > today)
        {
            s.Status = SubscriptionStatus.Frozen;
        }
        else
        {
            s.Status = SubscriptionStatus.Active;
        }

        if (s.Status != originalStatus || freezeWasProcessed)
        {
            await clientSubscriptionService.UpdateAsync(s);
        }
    }

    _logger.LogInformation("Client subscription statuses updated at {Time}", now);
}
```

Цей фрагмент реалізує middleware, який перевіряє, чи автентифікований користувач є заблокованим. Якщо користувач залогінений (`IsAuthenticated == true`), система зчитує значення клейму "IsBlocked" з колекції `claims`. Якщо це значення дорівнює `true`, відбувається автоматичне завершення сесії (`SignOutAsync`) і перенаправлення користувача, наприклад, на головну сторінку. Якщо користувач не заблокований, виконання передається наступному компоненту в конвеєрі.

```
var user = context.User;

if (user.Identity?.IsAuthenticated == true)
{
    var isBlocked = user.Claims.FirstOrDefault(c => c.Type == "IsBlocked")?.Value;

    if (bool.TryParse(isBlocked, out var blocked) && blocked)
    {
        await context.SignOutAsync("MyCookieAuth");

        context.Response.Redirect("");
        return;
    }
}

await _next(context);
```

Метод `GetAllPagedAsync` є універсальною функцією для отримання сторінкових вибірок (pagination) з бази даних з використанням Entity Framework. Він забезпечує гнучку фільтрацію, сортування, включення пов'язаних сутностей і пагінацію.

На початку створюється запит до таблиці відповідного типу сутності `TEntity`, який розширюється з використанням `AsExpandable()` для підтримки динамічного розгортання виразів. Далі, якщо задані `includes`, система додає в запит пов'язані таблиці. Після цього за потреби застосовуються умови фільтрації — основний фільтр (`predicate`) і додатковий фільтр пошуку (`searchFilter`).

Для сортування можна вказати один або два критерії (`orderBy`, `thenBy`) з напрямком (`descending`). Якщо не вказано сортування, застосовується заглушка `OrderBy(e => true)` — щоб гарантувати консистентність виконання `ToPagedListAsync` [23].

Наприкінці запит передається до методу `ToPagedListAsync`, який розбиває результати на сторінки згідно з параметрами `page` і `pageSize`. Таким чином, цей

метод дозволяє ефективно обробляти великі обсяги даних із гнучкою конфігурацією.

```
public async Task<IPagedList<TEntity>> GetAllPagedAsync<TSort1, TSort2>(  
    Expression<Func<TEntity, bool>>? predicate = null,  
    Expression<Func<TEntity, bool>>? searchFilter = null,  
    Expression<Func<TEntity, TSort1>>? orderBy = null,  
    Expression<Func<TEntity, TSort2>>? thenBy = null,  
    bool descending = false,  
    int page = 1,  
    int pageSize = PaginationDefaults.PageSize,  
    Expression<Func<TEntity, object>>[]? includes = null)  
{  
    IQueryable<TEntity> query = _context.Set<TEntity>().AsExpandable();  
  
    if (includes != null)  
        foreach (var include in includes)  
            query = query.Include(include);  
  
    if (predicate != null)  
        query = query.Where(predicate);  
  
    if (searchFilter != null)  
        query = query.Where(searchFilter);  
  
    if (orderBy != null)  
    {  
        query = descending  
            ? query.OrderByDescending(orderBy)  
            : query.OrderBy(orderBy);  
  
        if (thenBy != null)  
        {  
            query = descending  
                ? ((IOrderedQueryable<TEntity>)query).ThenByDescending(thenBy)  
                : ((IOrderedQueryable<TEntity>)query).ThenBy(thenBy);  
        }  
    }  
    else  
        query = query.OrderBy(e => true);  
  
    return await query.ToPagedListAsync(page, pageSize);  
}
```

Даний HTML-код описує Bootstrap-модальне вікно з ідентифікатором clientSearchModal, яке відкривається для здійснення пошуку клієнта за введеними параметрами [24]. Вміст модального вікна включає заголовок, кнопку закриття та поле вводу для введення пошукового запиту. Нижче поля розташовано контейнер clientSearchResults, у якому динамічно виводяться результати пошуку.

JavaScript-код слухає подію keyup у текстовому полі #clientSearchBox. Коли користувач вводить хоча б два символи, відправляється GET-запит до методу SearchClients у відповідному контролері. Запит передає параметр query,

який містить поточне значення введеного тексту. У відповідь очікується HTML-контент (наприклад, список знайдених клієнтів), який вставляється в контейнер #clientSearchResults. Якщо запит коротший за два символи, результати очищуються.

Таким чином реалізується інтерактивний пошук клієнтів без перезавантаження сторінки.

```
<div class="modal fade" id="clientSearchModal" tabindex="-1" aria-  
labelledby="clientSearchModalLabel" aria-hidden="true">  
  <div class="modal-dialog modal-lg">  
    <div class="modal-content">  
      <div class="modal-header">  
        <h5 class="modal-title" id="clientSearchModalLabel">Пошук  
клієнта</h5>  
        <button type="button" class="btn-close" data-bs-dismiss="modal"   
aria-label="Закрити"></button>  
      </div>  
      <div class="modal-body">  
        <input type="text" id="clientSearchBox" class="form-control mb-2"   
placeholder="Ім'я, Прізвище, Телефон, Email" />  
        <div id="clientSearchResults"></div>  
      </div>  
    </div>  
  </div>  
</div>  
$('#clientSearchBox').on('keyup', function () {  
  var query = $(this).val();  
  if (query.length > 1) {  
    $.get('@Url.Action("SearchClients")', { query: query }, function (html) {  
      $('#clientSearchResults').html(html);  
    });  
  } else {  
    $('#clientSearchResults').empty();  
  }  
});
```

Цей фрагмент коду відповідає за повернення коштів клієнту, яке здійснюється залежно від методу оплати, вказаного в об'єкті payment.

Спочатку перевіряється, чи був платіж здійснений за допомогою абонементу (PaymentMethod.Subscription). У такому випадку виконується пошук активного абонементу користувача за допомогою _clientSubscriptionService.GetCurrentSubscriptionByClientIdAsync. Якщо абонемент існує і його статус відмінний від ToBeActivated, система вважає, що повернення тренування можливе, і збільшує кількість доступних тренувань на один. Оновлена інформація про абонемент зберігається в базі.

Якщо ж такий абонемент не знайдено або його ще не активовано (статус ToBeActivated), система не може повернути тренування, і виконується повернення коштів на депозит. Для цього змінюється спосіб оплати в об'єкті payment на Deposit, і викликається метод AddToDepositAsync, який поповнює внутрішній рахунок клієнта на суму повернення.

У разі, якщо початково платіж був здійснений не через абонемент, система просто додає відповідну суму на депозит клієнта.

Нарешті, об'єкту payment присвоюється новий Id, фіксується поточний час як CreatedAt, встановлюється статус Refunded, і цей платіж зберігається в базі даних через метод _paymentRepository.AddAsync. Таким чином, фрагмент забезпечує гнучку обробку повернення коштів або тренувань залежно від початкового методу оплати.

```
if (payment.PaymentMethod == PaymentMethod.Subscription)
{
    var clientSubscription = await
_clientSubscriptionService.GetCurrentSubscriptionByClientIdAsync(
payment.ClientId, trainingType);
    if (clientSubscription != null &&
        clientSubscription.Status != SubscriptionStatus.ToBeActivated)
    {
        clientSubscription.TrainingsRemaining++;
        await _clientSubscriptionService.UpdateAsync(clientSubscription);
    }
    else
    {
        payment.PaymentMethod = PaymentMethod.Deposit;
        await _clientService.AddToDepositAsync(payment.ClientId, payment.Amount);
    }
}
else
    await _clientService.AddToDepositAsync(payment.ClientId, payment.Amount);

payment.Id = Guid.NewGuid();
payment.CreatedAt = DateTime.UtcNow;
payment.PaymentStatus = PaymentStatus.Refunded;

var created = await _paymentRepository.AddAsync(payment);
```

Висновки до розділу 2

У межах другого розділу було проведено комплексне проєктування системи управління фітнес-клубом, що охоплює всі ключові етапи побудови сучасного веб-орієнтованого програмного продукту. Перш за все, було здійснено формалізацію вимог до системи з урахуванням потреб різних категорій користувачів — клієнтів, тренерів та адміністраторів. Було визначено як функціональні, так і нефункціональні вимоги, а також специфіковано часові характеристики реакції системи на дії користувача. Особливу увагу приділено сценаріям взаємодії користувачів із системою, які відображено через варіанти використання. На основі цих сценаріїв було створено детальний опис для кожного ключового варіанту, що дозволяє чітко уявити очікувану поведінку системи в різних умовах.

У процесі проєктування об'єктно-орієнтованої структури було обґрунтовано використання підходу table-per-hierarchy для моделі користувача, що дало змогу ефективно об'єднати зберігання інформації про клієнтів, тренерів та менеджерів в єдиній таблиці бази даних. Далі було розроблено логічну модель бази даних, яка включає одинадцять таблиць, описано їхнє призначення, зв'язки та структуру.

Окремо було розглянуто етап проєктування та реалізації ключових алгоритмів, які забезпечують автоматизовану обробку бізнес-логіки. До таких алгоритмів належать купівля та активація абонементів, створення тренувань на основі шаблонів, заморожування та облік днів абонементу, а також логіка запису на тренування з урахуванням обмежень. Для кожного з них було побудовано діаграми активностей та послідовностей, що ілюструють детальний процес взаємодії між компонентами системи. Реалізація цих алгоритмів здійснювалася з урахуванням вимог до безпеки, перевірок прав доступу, відповідності статусів та даних, що гарантує стабільну й коректну поведінку системи.

На завершення було представлено практичну реалізацію окремих фрагментів програмного коду, що охоплюють функціонал авторизації, купівлі абонементу, запису на тренування, обробки повернень, фільтрації та

динамічного завантаження користувачів. Кожен із цих фрагментів супроводжувався поясненнями логіки їхньої роботи в контексті всієї системи.

Таким чином, другий розділ сформував цілісне бачення архітектури, логіки та реалізації основного функціоналу системи управління фітнес-клубом, що надалі стало основою для завершення повної реалізації програмного комплексу.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІЗ СИСТЕМОЮ УПРАВЛІННЯ ФІТНЕС-КЛУБОМ

3.1. Порядок встановлення та налаштування параметрів системи

Налаштування та запуск системи управління фітнес-клубом передбачає роботу з кількома ключовими компонентами, кожен із яких виконує окрему функцію та потребує відповідного розгортання. Оскільки ці компоненти взаємодіють між собою, важливо мати уявлення про їхню структурну організацію та принципи обміну даними. З цією метою було побудовано діаграму розгортання, яка відображає загальну архітектуру системи та ілюструє спосіб взаємодії клієнтської частини з веб-додатком і базою даних (рис. 3.1).

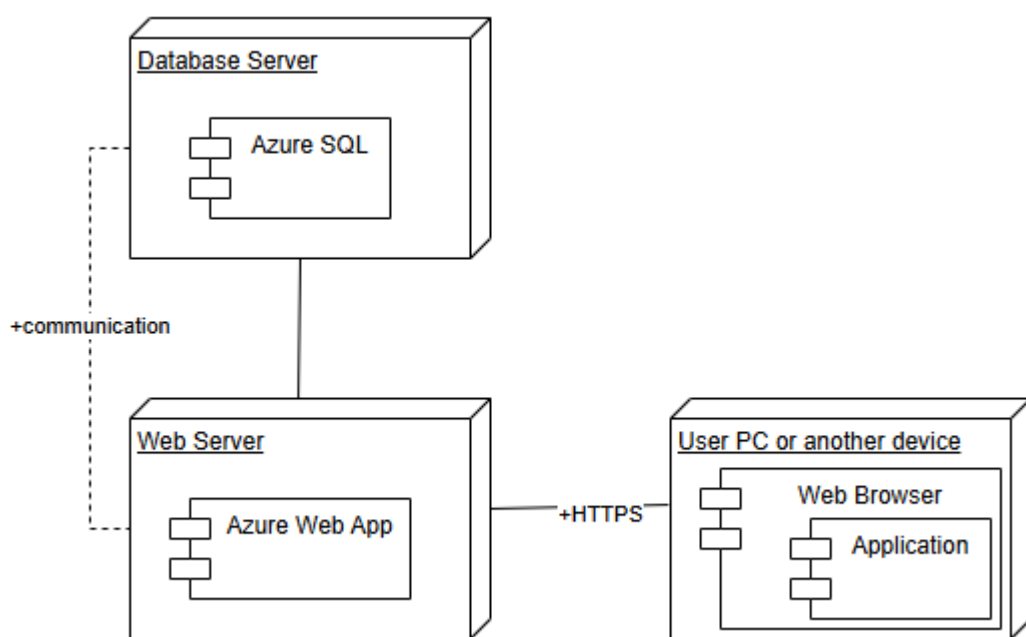


Рис. 3.1 – Діаграма розгортання системи управління фітнес-клубом

У лівій частині діаграми зображено Database Server, що представлений у вигляді служби Azure SQL Database. Саме в цій базі зберігаються всі структуровані дані системи: інформація про користувачів, абонементи, відвідування тренувань, платежі, розклад та інші сутності. Azure SQL — це керована реляційна СУБД, що забезпечує автоматичне масштабування, резервне копіювання, контроль доступу та високу доступність. Сервер бази даних

ізолюваний від клієнтів і взаємодіє лише з веб-додатком через безпечне внутрішнє з'єднання в рамках хмарної інфраструктури.

Нижче розміщено Web Server, який у нашому випадку реалізовано на базі Azure Web App — це повністю керований сервіс для хостингу веб-застосунків, написаних на .NET. Саме тут розгорнута серверна частина системи — логіка обробки запитів, маршрутизація, доступ до бази даних через Entity Framework Core, автентифікація та бізнес-функції. Перевага Azure Web App полягає у швидкому деплої, автоматичному масштабуванні та інтеграції з системами CI/CD, такі як GitLab, що спрощує оновлення програмного коду без зупинки сервісу.

Праворуч на діаграмі показано клієнтські пристрої — комп'ютери, планшети, смартфони, які взаємодіють із системою через веб-браузер. Увесь інтерфейс користувача реалізовано як адаптивний веб-додаток, доступний через звичайний браузер. З'єднання між клієнтом і веб-сервером здійснюється за допомогою протоколу HTTPS, що забезпечує шифрування всього трафіку між користувачем і сервером [25]. Це критично важливо для захисту персональних даних, облікових записів, а також для відповідності вимогам безпеки.

Усі запити, які надсилає користувач через браузер (наприклад, авторизація, перегляд розкладу, запис на тренування), передаються через HTTPS до Azure Web App, де обробляються контролерами MVC. У разі потреби — дані витягуються з Azure SQL Database, обробляються, та результат повертається у браузер у вигляді HTML-сторінки або JSON-відповіді. Така схема забезпечує централізоване управління логікою застосунку й є оптимальною для веб-орієнтованих систем з великою кількістю користувачів.

У підсумку, розміщення системи у хмарі забезпечує високу доступність, автоматичне резервне копіювання, масштабованість при збільшенні навантаження та зниження витрат на інфраструктуру. Вибір Microsoft Azure як платформи розгортання дозволяє інтегрувати систему з іншими хмарними сервісами, а також оперативно впроваджувати оновлення без простоїв, що особливо важливо для стабільної роботи додатку фітнес-клубу.

3.2. Структура інтерфейсу системи управління фітнес-клубом

Інтерфейс системи управління фітнес-клубом побудований з урахуванням сучасних вимог до зручності, логіки взаємодії та адаптивності для різних типів пристроїв. Завдяки використанню ASP.NET MVC та шаблонів представлення .cshtml, інтерфейс є функціональним, структуровано організованим і зручним для користувачів із різними ролями. Усі елементи системи розроблено таким чином, щоб забезпечити легкий доступ до основних функцій платформи — від реєстрації та запису на тренування до керування клієнтами й перегляду статистики.

Головна сторінка є початковою точкою взаємодії користувача з платформою. Вона містить базову інформацію про фітнес-клуб, включаючи опис послуг, переваги та візуальні елементи для ознайомлення. У верхній частині розміщена навігаційна панель із кнопками входу та реєстрації, доступними для неавторизованих користувачів (рис.3.2-3.4).

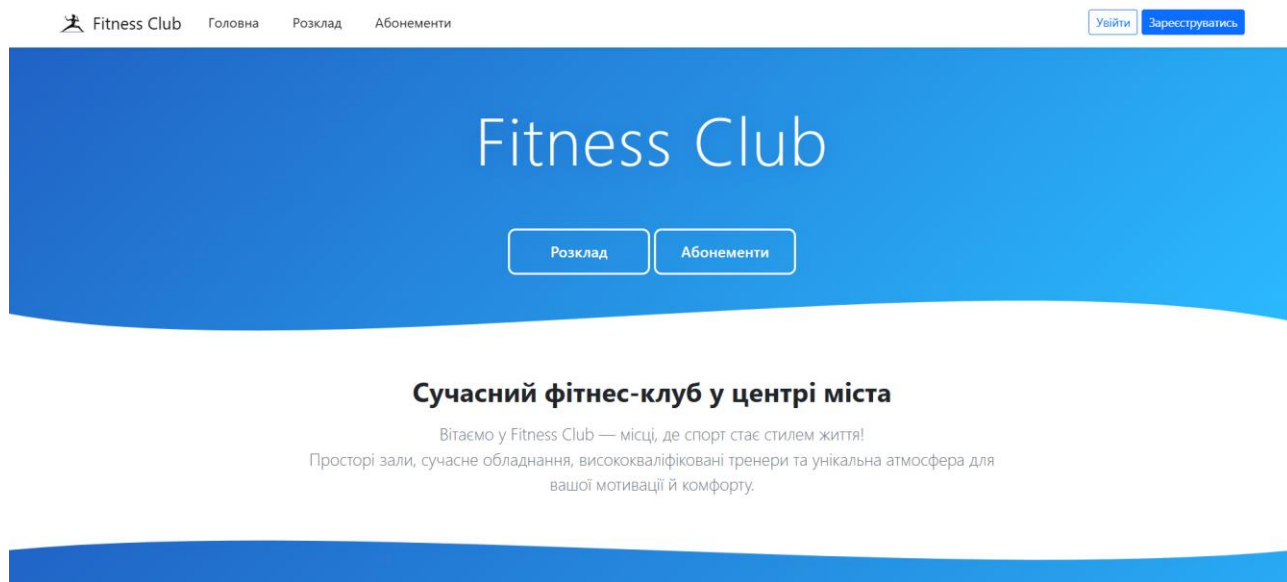


Рис. 3.2 – Головна сторінка веб-додатку

Ввійдемо як клієнт.

Для клієнта є доступною сторінка абонементів, де представлено перелік абонементів у відповідних групах. При натисненні на певне віконце з назвою, розгортається картка з короткою інформацією і кнопкою купівлі (рис. 3.5), що в свою чергу відкриває сторінку з повною інформацією і кнопкою купівлі з депозиту, якщо на рахунку є кошти (рис. 3.6).

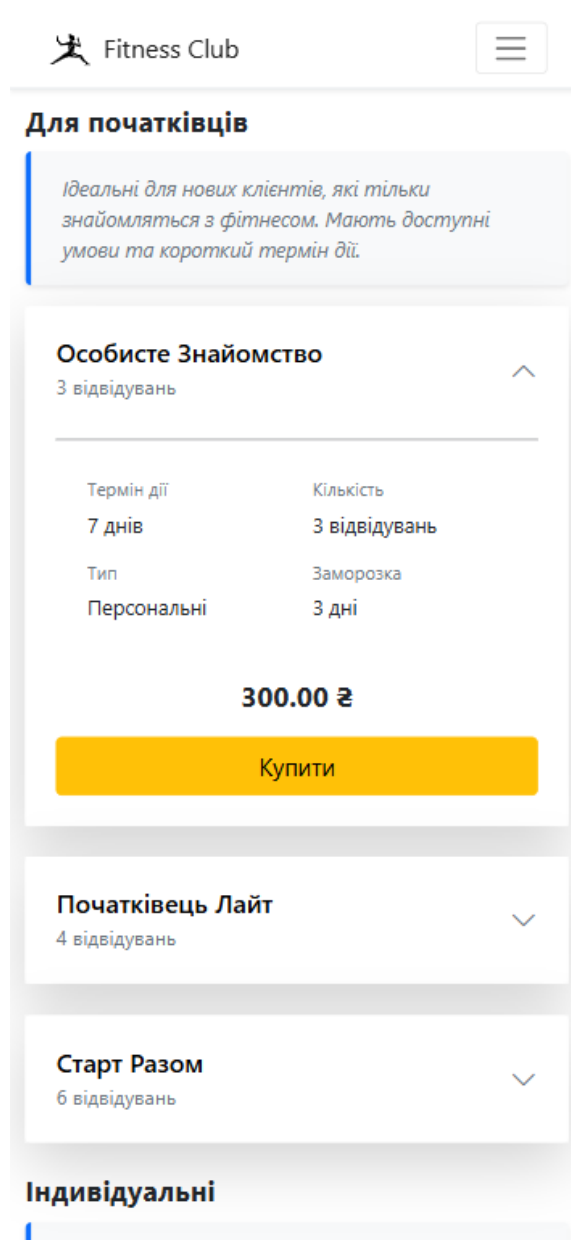


Рис. 3.5 – Сторінка абонементів

Особисте Знайомство
Для початківців

Тривалість:	7 днів
Тренувань у абонементі:	3
Тип тренувань:	Персональні
Заморозка:	до 3 днів
Скасування:	1
Тип:	Звичайний
До активації:	10 днів
Ціна:	300.00 ₴

[Купити з депозиту](#)

[Назад](#)

Політика конфіденційності
Договір про надання послуг
Правила клубу
© 2025 – Fitness Club

Рис. 3.6 – Сторінка купівлі абонементу

Клієнт має можливість перейти на сторінку розкладу тренувань, де для зручності реалізовано два режими перегляду – у вигляді списку або по днях тижня. Це дозволяє користувачам обирати найбільш зручний формат взаємодії. Для швидкого пошуку потрібного тренування передбачено фільтри за категорією тренування (наприклад, кардіо, силові, йога тощо) та за конкретним тренером. Також доступна кнопка «Мої тренування», яка при активації відображає лише ті сесії, на які користувач уже зареєстрований, що значно спрощує контроль за особистим розкладом (рис. 3.7).

Після натискання на конкретне тренування, клієнт потрапляє на його детальну сторінку. Тут відображається основна інформація: назва, опис, тривалість, зал, вартість, і кількість доступних місць. Якщо тренування ще не розпочалось і залишаються вільні місця, користувач може здійснити запис одним кліком (рис. 3.8). У разі успішної реєстрації система оновлює статус тренування, відображаючи його у списку «Мої тренування».

Список

День

Тренер

▼

Категорія

▼

☐

Мої тренування

<

ПН

16 червня

ВТ

17 червня

СР

18 червня

>

Callanetics / Йога, пілатес, стрейчинг

Зала 2

📅

вівторок, 17 червня, 09:25

Місце 10

Максим Дударенко

HIIT / Кардіо та інтервали

Зала 3

📅

вівторок, 17 червня, 09:40

Місце 16

Валерій Марченко

Power Individual / Силові тренування

Зала 1

📅

вівторок, 17 червня, 16:30

Олена Ковальчук

Персональне

DanceFit / Танці та ритм

Зала 4

Рис. 3.7 – Сторінка розкладу тренувань

Callanetics

Йога, пілатес, стрейчинг

Статичне глибоке тренування на основні м'язи та поставу.

Тренер:

Максим Дударенко

Тип:

Групові

Тривалість:

55 хв

Дата й час:

17 червня 2025, 09:25

Зал:

2

Вартість:

190.00 ₪

Учасники:

0 / 10

Оплатити з депозиту

← Назад до розкладу

Політика конфіденційності

Договір про надання послуг

Правила клубу

© 2025 – Fitness Club

Рис. 3.8 – Сторінка реєстрації на тренування

На особистій сторінці клієнта (рис. 3.9) зібрана вся важлива інформація: ім'я, контактні дані, залишок на депозиті, а також активні та минулі абонементи (рис. 3.10). Тут користувач може переглядати статуси абонементів, кількість доступних тренувань, дні заморозки та інші параметри. Також клієнту доступна історія його тренувань (рис. 3.11), з можливістю скасування майбутніх записів заздалегідь.

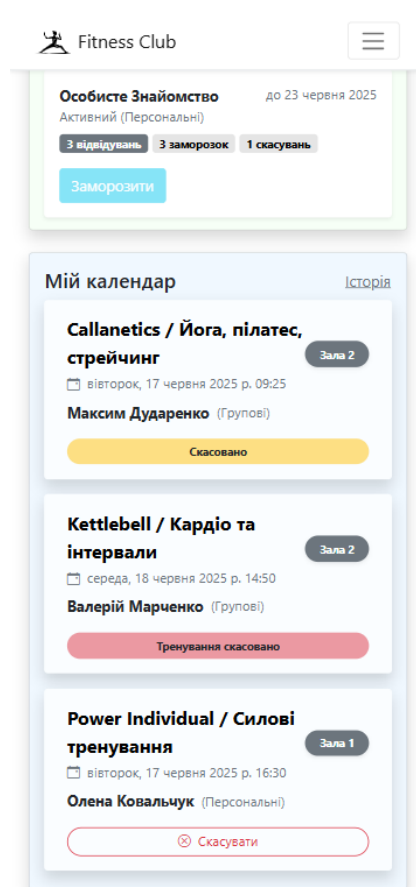


Рис. 3.9 – Особиста сторінка клієнта

У ролі тренера інтерфейс змінюється відповідно до його функціональних обов'язків. Перейшовши на сторінку розкладу, тренер бачить лише ті тренування, які закріплені за ним. Опція «Мої тренування» дозволяє фільтрувати як групові, так і персональні сесії, що він веде (рис. 3.12). При відкритті конкретного тренування, система відображає повний список зареєстрованих клієнтів із зазначенням їх контактної інформації та статусу участі (рис. 3.13), що дозволяє тренеру ефективно організувати проведення заняття.

Історія абонементів

×

Family3/Family

Статус

Заблокований

Ціна

2.00 ₴

Оплата

Депозит

Відвідувань

2

Переносів

0

Заморожок

2

Закінчення

11.06.2025

Maya/group12

Статус

Заблокований

Ціна

2005.40 ₴

Оплата

Депозит

Відвідувань

121

Переносів

1

Заморожок

4

Закінчення

09.07.2025

Expensive4 fam ind/Expensive

Статус

Закінчився

Ціна

1.00 ₴

Оплата

Депозит

Відвідувань

1

Переносів

0

Заморожок

0

Закінчення

09.06.2025

Рис. 3.10 – Сторінка історії придбаних абонементів

Історія відвідувань

×

2 реєстрації / Танці

02:38 | п'ятниця, 13 червня 2025

Групові

Скасовано

Тренер: Андрій Павлусенко

Оплата: Абонемент

2 реєстрації / Танці

02:38 | п'ятниця, 13 червня 2025

Групові

Скасовано

Тренер: Андрій Павлусенко

Оплата: Абонемент

7324601872 / Танці

01:09 | субота, 14 червня 2025

Персональні

Тренер: Валерій Марченко

Оплата: Готівка

2 реєстрації / Танці

02:38 | п'ятниця, 13 червня 2025

Групові

Скасовано

Тренер: Андрій Павлусенко

Оплата: Абонемент

300 хвилин задоволення / я люблю маму

21:49 | п'ятниця, 13 червня 2025

Персональні

Тренер: Андрій Павлусенко

Оплата: Карта

персональне оплачується абонементом

22:52 | середа, 11 червня 2025

Персональні

Тренер: Андрій Павлусенко

Оплата: Депозит

2 реєстрації / Танці

Рис. 3.11 – Сторінка історії відвідувань клієнта

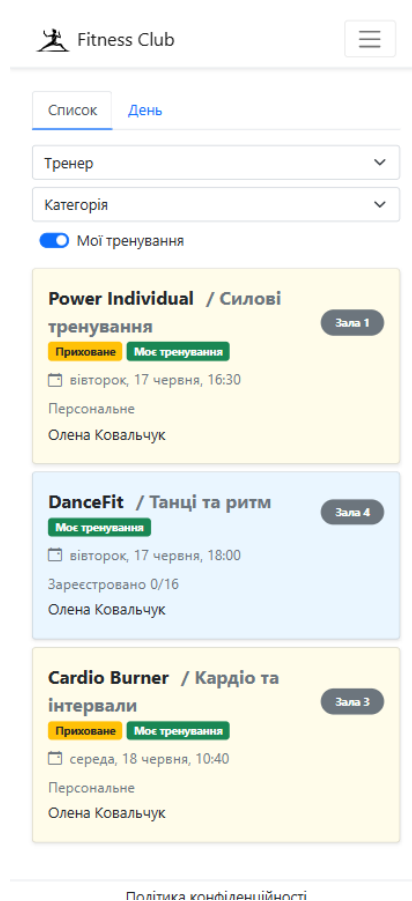


Рис. 3.12 – Сторінка розкладу тренувань з відміткою «Мої тренування»

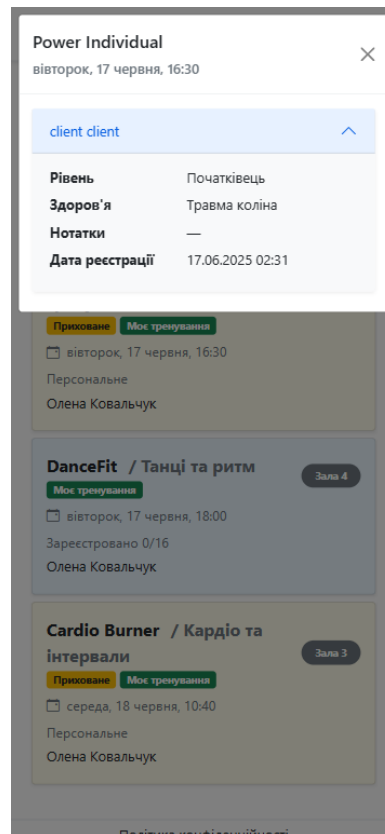


Рис. 3.13 – Вікно перегляду списку клієнтів, які записані на тренування

Після входу в обліковий запис із роллю «Менеджер» користувача зустрічає зручна та функціональна панель, що містить усі необхідні інструменти для керування даними клієнтів, абонементами та тренуваннями.

При переході на сторінку списку клієнтів (рис. 3.14) менеджеру надається доступ до повного переліку зареєстрованих користувачів. Зверху розташоване поле пошуку, яке дозволяє швидко знайти потрібну особу за іменем, прізвищем, номером телефону або електронною адресою. Кожен запис супроводжується кнопкою «Деталі», що відкриває розширену інформацію про клієнта.

Fitness Club						Абонементи ▾	Тренування ▾	Клієнти ▾	Платіжі	manager manager ▾
Клієнти										
Пошук за іменем, телефоном, email						Пошук				
Ім'я	Прізвище	Баланс	Телефон	Email	У клубі з					
client	client	5169.14 ₴	+380972541234	client@test.com	травень 2025	Деталі				
Софія	Будник	0.00 ₴	+380974512774	sof.bud24@icloud.com	червень 2025	Деталі				
Анна	Васильківська	8185.60 ₴	+380692453647	anna@test.com	травень 2025	Деталі				
Артем	Віщенко	0.00 ₴	+380967846122	artem@gmail.com	червень 2025	Деталі				
Обожнююю	Владу	0.00 ₴	+3809611111070	po2ukr.net80@gmail.com	червень 2025	Деталі				
Віктор	Гаврилов	0.00 ₴	+380632344197	viktor153.ua@gmail.com	червень 2025	Деталі				
Олена	Ліщук	0.00 ₴	+380964732443	olena@gmail.com	червень 2025	Деталі				
Олександра	Мацюбинська	3999.00 ₴		olexandra@test.com	травень 2025	Деталі				
Євгенія	Михайлюк	1199.00 ₴	12333212322	euger=nia@gmail.com	травень 2025	Деталі				
Катерина	Паномарчук	0.00 ₴		kate_cool_girl@icloud.com	червень 2025	Деталі				
Анастасія	Розклад	43671.60 ₴	+380876625533	anastasiz@gmail.com	травень 2025	Деталі				
Наталія	Шевчук	123.00 ₴	+380972253344	shev12@gmail.com	травень 2025	Деталі				

Рис. 3.14 – Сторінка списку всіх клієнтів

Сторінка детальної інформації про клієнта (рис. 3.15–3.16) відображає його особисті дані, поточний стан рахунку, активні абонементи та заплановані тренування. Звідси менеджер може легко керувати взаємодією з клієнтом.

Серед доступних дій — поповнення депозиту (рис. 3.17), що відкриває модальне вікно з введенням суми та підтвердженням операції. Також реалізовано функцію продажу абонементу (рис. 3.18), де вибирається клієнт, шаблон абонементу та спосіб оплати. Якщо клієнт ще не належить до жодної сімейної групи, йому можна запропонувати приєднатися до наявної або створити нову (рис. 3.19). У такому випадку відкривається список інших клієнтів для додавання до сім'ї.

Абонементи

Тренування

Клієнти

Плажети

manager manager

Клієнт

client client

Баланс

5169.14 ₴

Телефон

+380972541234

Email

client@test.com

Дата народження

01.01.2001 (24 р.)

Нотатки

—

Фізична підготовка

Початківець

Стан здоров'я

Травма коліна

Сім'я

Євгенія Михайлюк

Створено

5/22/2025 1:47:16 AM

Оновлено

6/17/2025 6:40:34 PM

← Назад до списку

Заблокувати

Видалити з сім'ї

Редагувати

Поповнити депозит

Продати абонемент

Абонементи

Відвідування

Активні абонементи

Відобразити всі

Назва	Ціна	Тип	Тренувань	Заморозок	Скасувань	Дата закінчення	Статус	
Особисте Знайомство	300 ₴	Персональні	3	3	1	23/06/2025	Активний	Деталі
Старт Разом	1000 ₴	Групові	6	—	3	18/07/2025	Чекат на активацію	Деталі

Рис. 3.15 – Сторінка детальної інформації про клієнта

Абонементи

Відвідування

Відвідування

Відобразити всі

Тренування	Дата	Тип	Тренер	Статус	
Cardio Burner	18/06/2025 10:40	Індивідуальне	Олена Ковальчук	Зареєстровано	Деталі Скасувати
Kettlebell	18/06/2025 14:50	Групове	Валерій Марченко	Тренування скасовано	Деталі

Рис. 3.16 – Вкладка «Відвідування» на сторінці інформації про клієнта

Крім того, менеджер може редагувати особисті дані клієнта (рис. 3.20), зокрема змінювати ім'я, дату народження, телефон або email. У випадку потреби клієнт може бути заблокований — у такому разі його профіль буде позначено відповідною міткою, а також з'явиться поле для введення причини блокування (рис. 3.21).

Окремим інструментом є форма створення нового клієнта (рис. 3.22). Вона містить поля для введення особистої інформації з валідацією формату дати народження, номера телефону (з урахуванням українського формату) та email-адреси. Це дозволяє забезпечити цілісність даних і уникнути помилок при реєстрації.

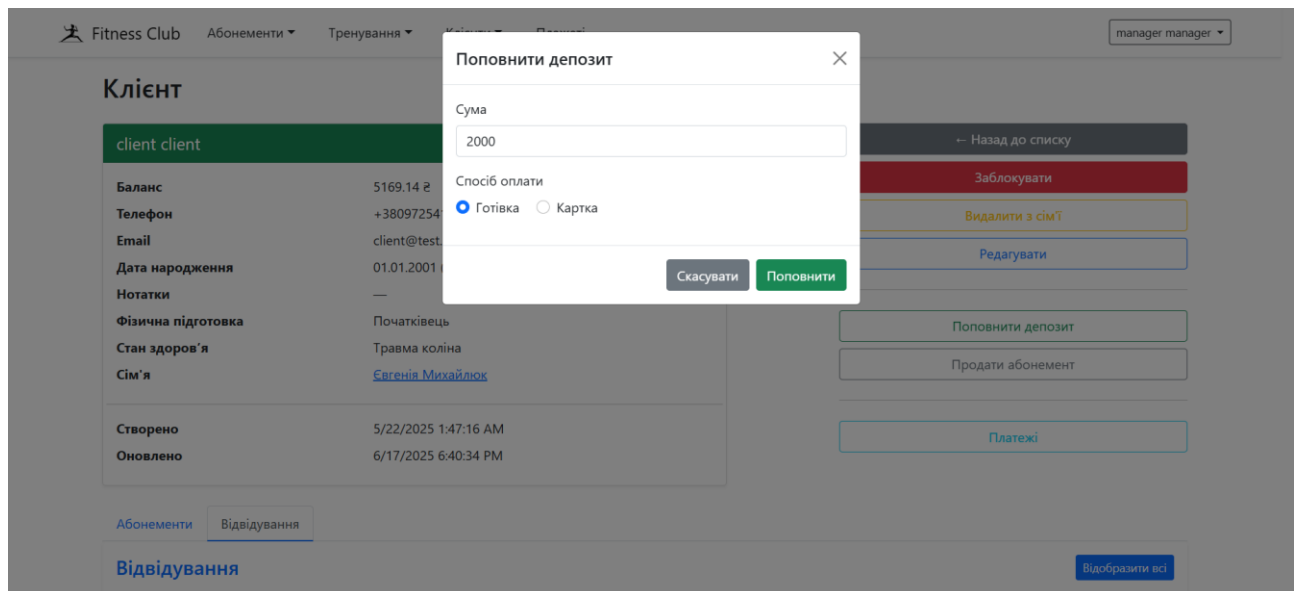


Рис. 3.17 – Вікно поповнення депозиту клієнта

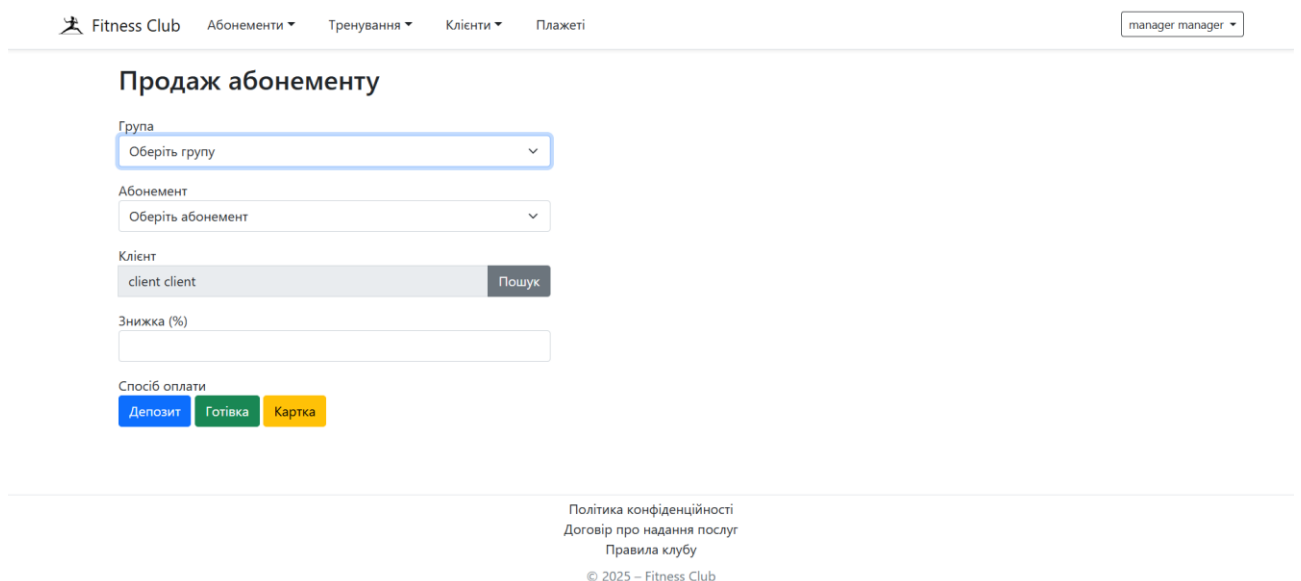


Рис. 3.18 – Сторінка продажу абонементу вибраному клієнту

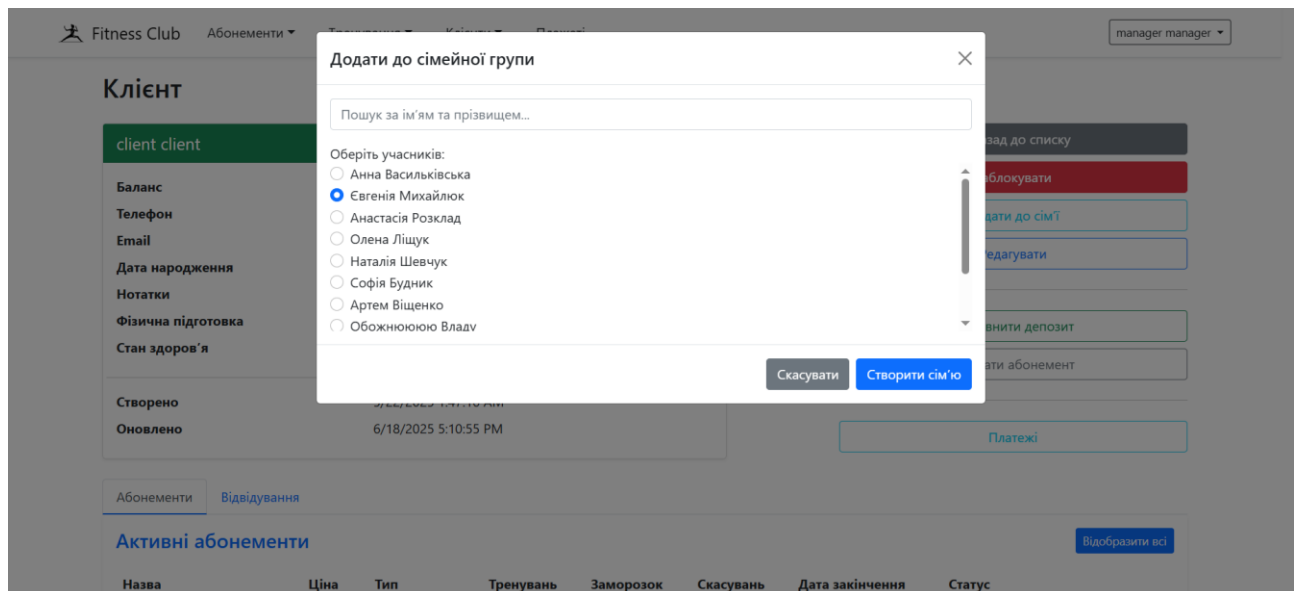


Рис. 3.19 – Вікно з пошуком клієнта для створення сімейної групи

Fitness Club Абонементи Тренування Клієнти Платежі manager manager

Редагувати клієнта

Ім'я
client

Прізвище
client

Дата народження
01.01.2001

Телефон
+380972541234

Email
client@test.com

Рівень фізичної підготовки
Початківець

Стан здоров'я
Травма коліна

Нотатки

Зберегти

← Назад

Політика конфіденційності
Договір про надання послуг
Правила клубу
© 2025 – Fitness Club

Рис. 3.20 – Сторінка редагування інформації про клієнта

Клієнт

Наталія Шевчук		Заблокований
Баланс	123.00 ₪	
Телефон	+380972253344	
Email	shev12@gmail.com	
Дата народження	18.07.2001 (23 р.)	
Нотатки	довге волосся	
Фізична підготовка	—	
Стан здоров'я	—	
Причина блокування	Невиконання умов договору	
Створено	5/27/2025 5:00:38 PM	
Оновлено	6/18/2025 5:12:27 PM	

[← Назад до списку](#)

Розблокувати

Платежі

Абонементи

Відвідування

Активні абонементи

Відобразити всі



Немає активних абонементів

Політика конфіденційності
Договір про надання послуг
Правила клубу

© 2025 – Fitness Club

Рис. 3.21 – Сторінка детальної інформації заблокованого клієнта

Створити клієнта

Ім'я

Прізвище

Дата народження

дд.мм.рррр

Телефон

Email

Рівень фізичної підготовки

Стан здоров'я

Нотатки

Створити

← Назад до списку

Політика конфіденційності
Договір про надання послуг
Правила клубу

© 2025 – Fitness Club

Рис. 3.22 – Сторінка створення нового клієнта

Після переходу на сторінку всіх груп абонементів (рис. 3.23) користувач отримує перелік усіх створених груп, у якому для кожної групи вказується кількість шаблонів абонементів, що до неї належать. Додатково доступні посилання для перегляду цих абонементів, а також кнопка для переходу до абонементів клієнтів, придбаних у межах відповідної групи.

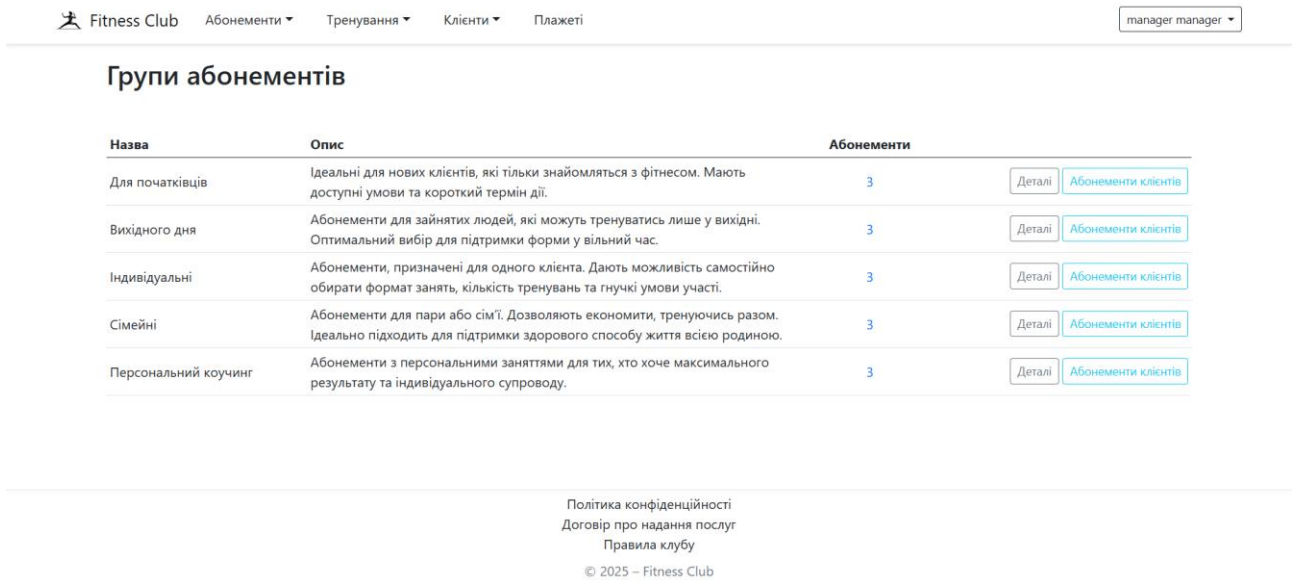


Рис. 3.23 – Сторінка з переліком груп абонементів

Натиснувши кнопку «Деталі» біля назви групи, відкривається сторінка перегляду групи абонементів (рис. 3.24). Тут є можливість архівувати або редагувати групу — ці дії виконуються за аналогією з редагуванням або архівуванням клієнта, що є однаковим для всіх сторінок редагування в системі.

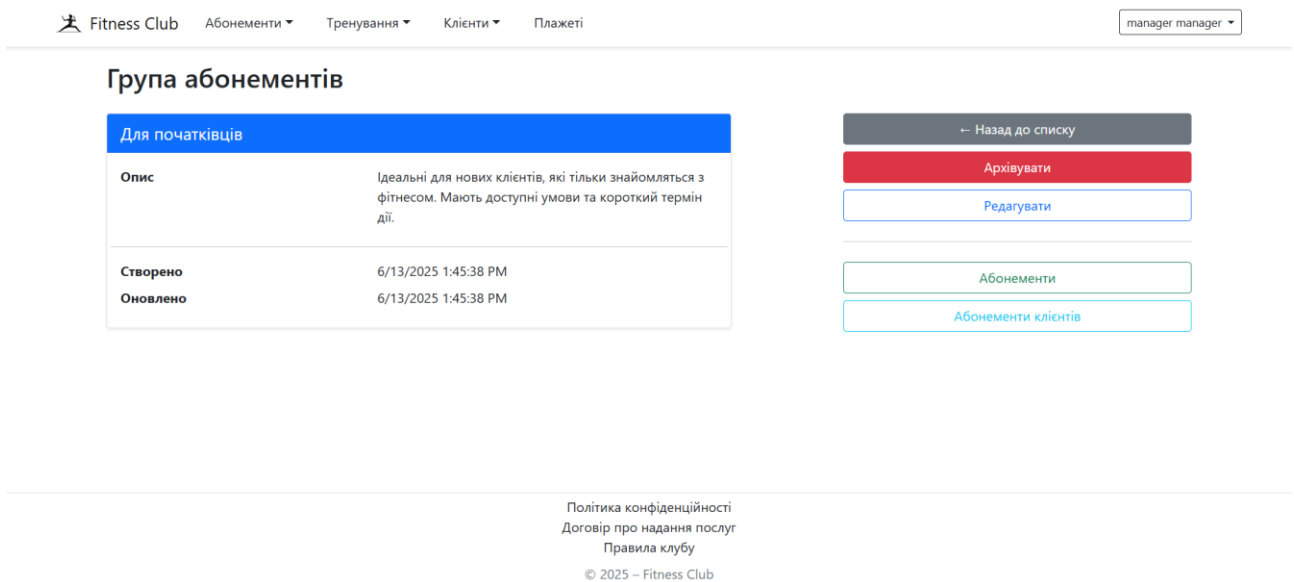


Рис. 3.24 – Сторінка з інформацією про групу абонементів

На сторінці шаблонів абонементів (рис. 3.25) відображаються всі наявні абонементи, згруповані відповідно до своїх груп. Для кожного з шаблонів доступна кнопка продажу, яка відкриває відповідну форму, а також кнопка перегляду деталей.

Рис. 3.25 – Сторінка зі списком всіх абонементів

Також реалізовано окрему сторінку перегляду архівованих шаблонів абонементів, у якій доступний пошук за назвою або групою. При натисканні кнопки «Відновити» відкривається модальне вікно підтвердження. Якщо шаблон належить до архівованої групи, з'являється вибір: відновити шаблон разом із групою або перенести його до іншої активної групи (рис. 3.26).

Fitness Club

Абонементи

Тренування

Клієнти

Платіжні

manager manager

Архівовані абонементи

Введіть назву абонементу чи групи

Пошук

Назва	Група	Тривалість	Тип	Тренування	Тип тренувань	Дата архівування	Деталі	Відновити
тест активація 1 день, закінчення 2 дні	Сімейні					6/13/2025 1:47:15 PM	Деталі	Відновити
фітфлпфцкп	Family					6/13/2025 1:28:33 PM	Деталі	Відновити
Family1	Family					6/13/2025 1:28:33 PM	Деталі	Відновити
Family3	Family	2 дні	Звичайний	2	Групові	6/13/2025 1:28:33 PM	Деталі	Відновити
Fitness	Family	4 місяці	Звичайний	12	Групові	6/13/2025 1:28:33 PM	Деталі	Відновити
Family2	Family	2 дні	Звичайний	2	Групові	6/13/2025 1:28:33 PM	Деталі	Відновити
Expensive5	Expensive	1 день	Звичайний	1	Персональні	6/13/2025 1:28:27 PM	Деталі	Відновити
Expensive4 fam ind	Expensive	1 день	Сімейний	1	Персональні	6/13/2025 1:28:27 PM	Деталі	Відновити
Expensive2	Expensive	1 день	Звичайний	1	Групові	6/13/2025 1:28:27 PM	Деталі	Відновити
Balet	Personal	4 місяці	Звичайний	Безлім	Групові	6/13/2025 1:27:50 PM	Деталі	Відновити
Personal	Personal	2 дні	Звичайний	2	Персональні	6/13/2025 1:27:50 PM	Деталі	Відновити

Абонемент у архівованій групі

Абонемент належить до архівованої групи "Family".

Оберіть дію:

Відновити абонемент разом з групою

або виберіть нову групу:

-- Оберіть групу --

Призначити нову групу та відновити

Рис. 3.26 – Вікно з уточненням при відновленні абонементу

У деталях шаблону абонементу (рис. 3.27) користувач може виконати дії: архівувати, зупинити продаж, редагувати або продати шаблон. Нижче розміщується перелік клієнтських абонементів, що були придбані на основі цього шаблону. Вони поділені на п'ять вкладок: активні, неактивовані, заморожені, без відвідувань та завершені. Для зручності реалізовано пошук за ім'ям або прізвищем клієнта.

Fitness Club
Абонементи
Тренування
Клієнти
Плажети

manager manager

Абонемент

Особисте Знайомство / Для початківців

В продажі

Тип	Звичайний
Ціна	300.00 ₪
Тривалість	7 днів
Тип тренувань	Персональні
Кількість тренувань	3
Заморозка	3 дні
Кількість скасування	1
Активация через	5 днів

Створено

6/13/2025 8:41:31 PM

Оновлено

6/16/2025 2:52:10 AM

← Назад до списку

Архівувати

Зупинити продаж

Редагувати

Продати

Пошук по імені та прізвищу

Пошук

Придбані

Активовані

Не активовані

Заморожені

Без відвідувань

Придбано	Період дії	Користувач	Тренувань	Заморозок	Скасування
6/11/2025	6/16/2025 - 6/23/2025	client client	3	3 дні	1

Деталі

Політика конфіденційності

Договір про надання послуг

Правила клубу

© 2025 – Fitness Club

Рис. 3.27 – Сторінка детальної інформації про абонемент

Крім цього, у системі є сторінка для перегляду всіх придбаних клієнтами абонементів (рис. 3.28). Вона також поділена на вкладки за статусом: усі, чекає на активацію, активні, заморожені, закінчені та заблоковані. Для швидкого пошуку реалізовано великий набір фільтрів, зокрема за типом, клієнтом, групою, статусом, терміном дії, знижкою тощо.

Абонементи клієнтів

Пошук за клієнтом/назвою

ID клієнта

ID шаблону

ID групи

Тривалість днів

Тренувань залишилось

Кількість тренувань

Вид

Тип

Знижка

дд.мм.рррр

дд.мм.рррр

Дата завершення

Нові

Фільтрувати

Очистити

Усі Чекає на активацію Активний Заморожений Закінчився Заблокований

Клієнт	Назва	Ціна	Тривалість	Тренувань	Вид	Тип	Знижка	Дата завершення	
client client	Особисте Знайомство	300.00 ₪	7 днів	2	Звичайний	Персональні	Hi	23/06/2025	Деталі
Євгенія Михайлюк	Expensive2	1.00 ₪	1 день	1	Звичайний	Групові	Hi	19/06/2025	Деталі
Євгенія Михайлюк	Старт Разом	1000.00 ₪	21 день	6	Сімейний	Групові	Hi	18/07/2025	Деталі
client client	Старт Разом	1000.00 ₪	21 день	6	Сімейний	Групові	Hi	18/07/2025	Деталі
client client	Персонал Безліміт	3.00 ₪	1 місяць	Безліміт	Звичайний	Персональні	Hi	15/07/2025	Деталі
client client	Maya	2005.40 ₪	1 місяць	121	Звичайний	Групові	Hi	09/07/2025	Деталі
Анастасія Розклад	Family1	0.88 ₪	1 день	13	Сімейний	Групові	Так	15/06/2025	Деталі
Анна Васильківська	Expensive5	1.00 ₪	1 день	1	Звичайний	Персональні	Hi	15/06/2025	Деталі

Рис. 3.28 – Сторінка зі списком всіх придбаних абонементів

На сторінці перегляду конкретного абонементу клієнта (рис. 3.29) менеджеру доступні основні керуючі дії — активувати, заморозити, розморозити або заблокувати абонемент. Перелік доступних дій динамічно залежить від поточного статусу абонементу. Крім цього, можна переглянути всі оплати, пов’язані з даним абонементом, натиснувши на відповідну кнопку.

Абонемент клієнта

Особисте Знайомство / Для початківців		Активний
Клієнт	client client	
Ціна	300.00 ₪	
Період дії	6/16/2025 – 6/23/2025	
Тривалість	7 днів	
Вид	Звичайний	
Тип тренувань	Персональні	
Тренувань залишилось	2	
Заморозок залишилось	3	
Скасувань залишилось	1	
Створено	6/17/2025 2:20:55 AM	
Оновлено	6/18/2025 5:30:12 PM	

[← Назад до списку](#)

Заблокувати

Заморозити

Платежі

[Політика конфіденційності](#)
[Договір про надання послуг](#)

Рис. 3.29 – Сторінка деталей абонементу клієнта

У системі реалізовано сторінку з переліком усіх категорій тренувань (рис. 3.30), де користувач може виконувати пошук за назвою категорії, а також натискати відповідні кнопки для переходу до шаблонів тренувань або безпосередньо до тренувань, що належать цій категорії. Сторінки створення,

редагування, деталей та списку архівованих виглядають аналогічно інших в додатку.

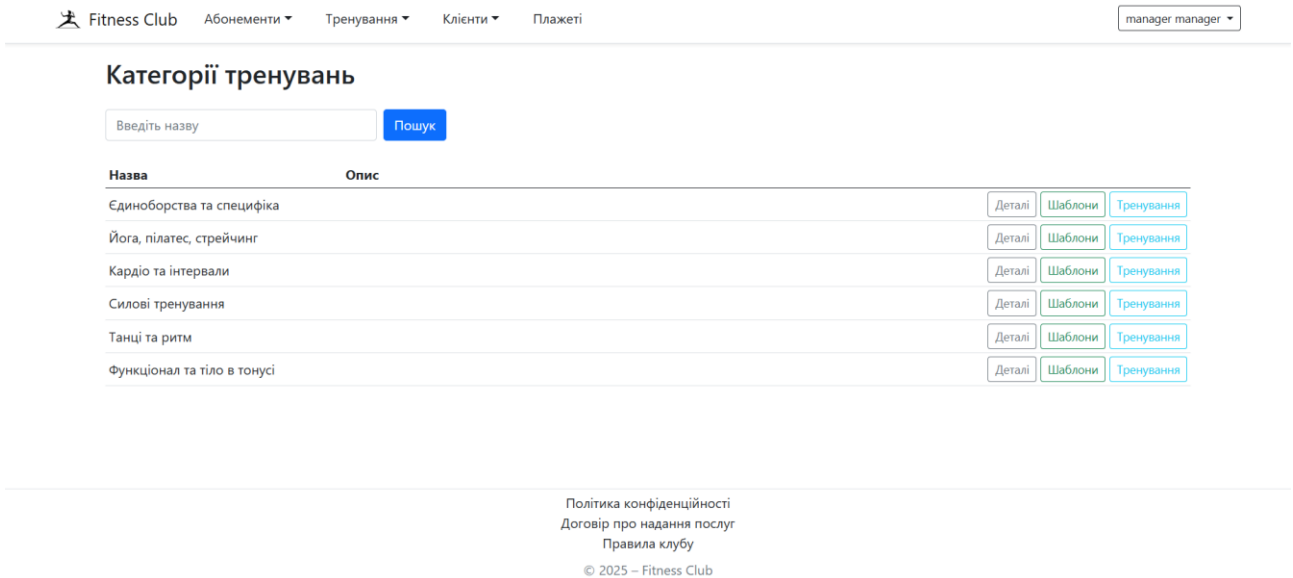


Рис. 3.30 – Сторінка з переліком категорій тренувань

Сторінка шаблонів тренувань (рис. 3.31) містить вкладки, кожна з яких відповідає окремій категорії. Завдяки такій побудові користувачеві зручно орієнтуватися серед великої кількості шаблонів. Додатково передбачено текстовий пошук для швидкого знаходження потрібного шаблону.

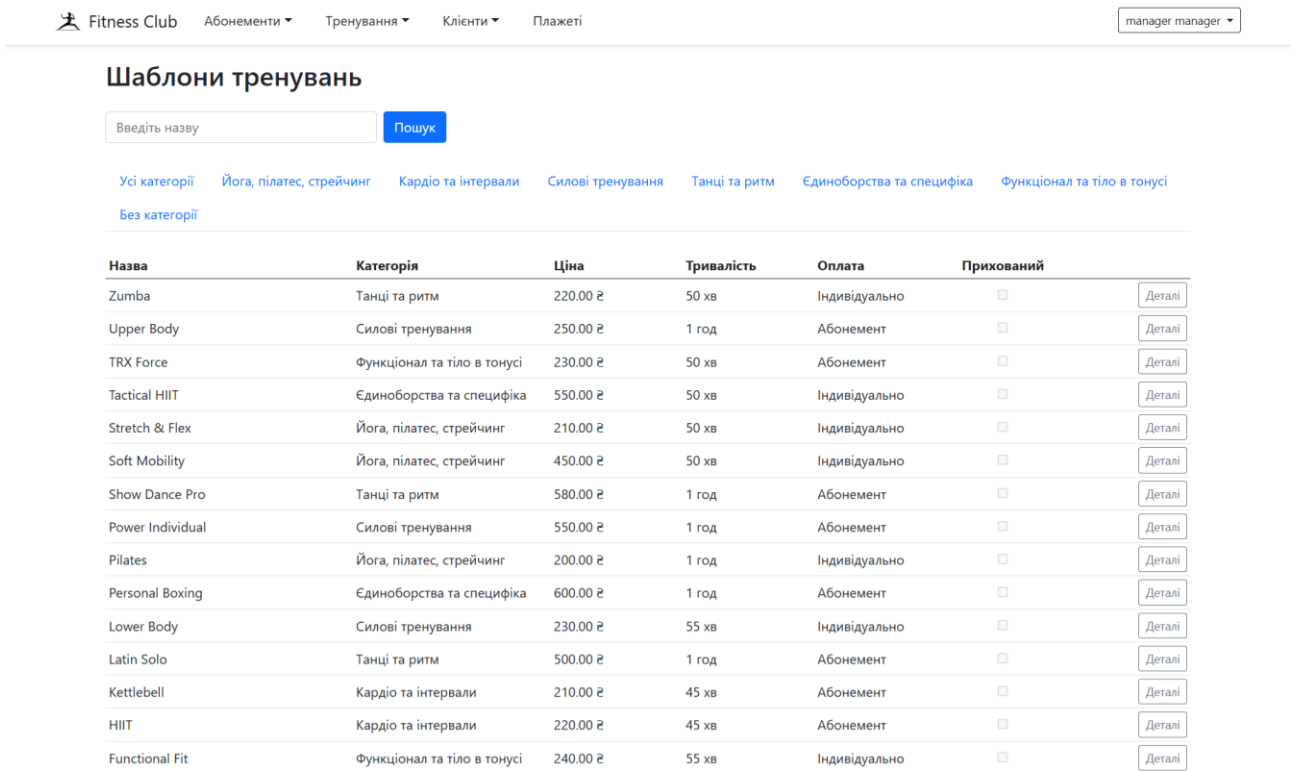


Рис. 3.31 – Сторінка усіх шаблонів тренувань

На сторінці перегляду шаблону тренування (рис. 3.32) доступні дії: архівація шаблону, редагування, приховування (що виключає шаблон із переліку під час створення нового тренування). Також передбачено можливість перегляду всіх тренувань, створених на основі цього шаблону, через відповідну кнопку.

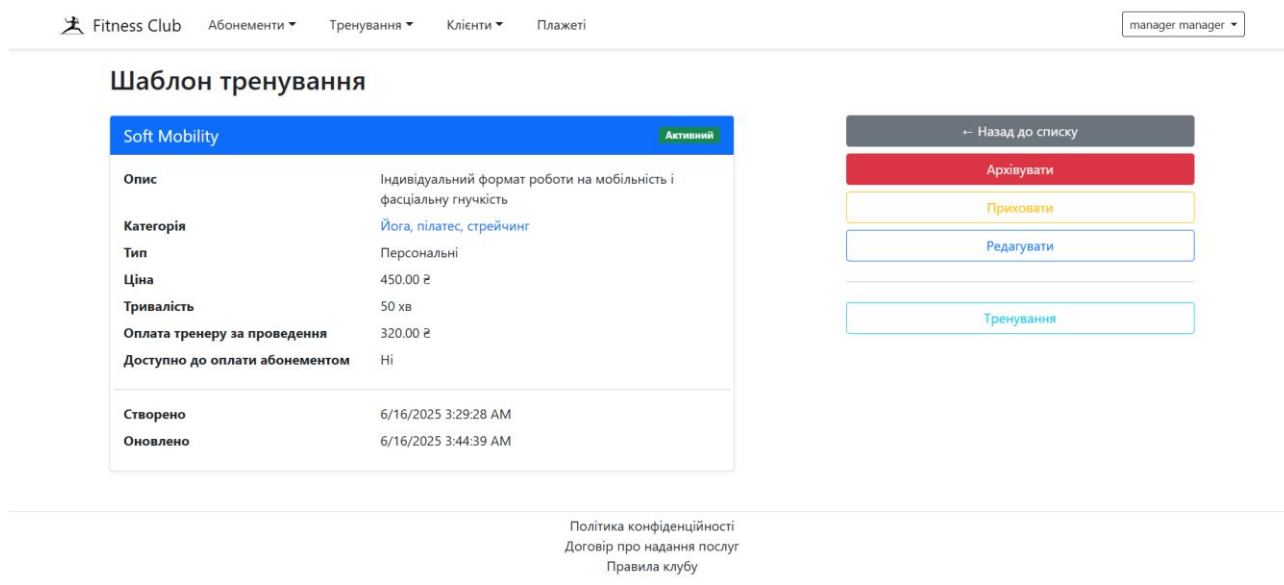


Рис. 3.32 – Сторінка деталей шаблону тренування

У випадку, якщо шаблон був архівований, його можна відновити на відповідній сторінці (рис. 3.33). При цьому, якщо шаблон не належав до жодної категорії, система дозволяє обрати нову категорію або залишити без категорії. Цей вибір відображається у вигляді модального вікна після натискання кнопки «Відновити».

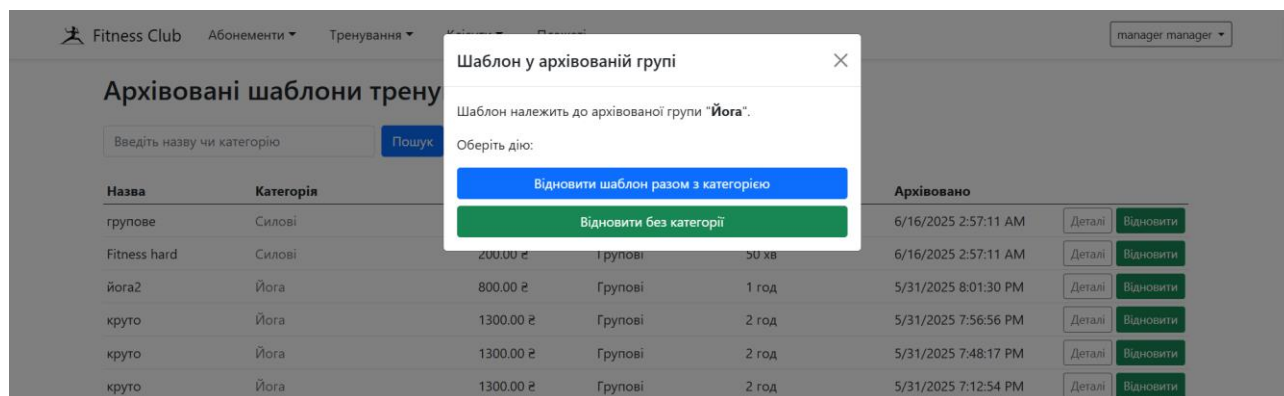


Рис. 3.33 – Вікно з опціями дій при відновленні шаблону тренування

Сторінка зі списком усіх тренувань (рис. 3.34) надає доступ до широкого спектра фільтрів: за категорією, тренером, статусом, типом тренування та датою.

[Абонементи](#)
[Тренування](#)
[Клієнти](#)
[Плажети](#)
manager manager

Рис. 3.34 – Сторінка зі списком усіх тренувань

Форма створення нового тренування (рис. 3.38) динамічно змінюється залежно від вибраного типу тренування. Якщо вибрано групове тренування, з'являються параметри «Дозволити реєстрацію» та «Приховати з розкладу». Якщо ж обрано персональне, додається поле пошуку клієнта та вибору способу оплати.

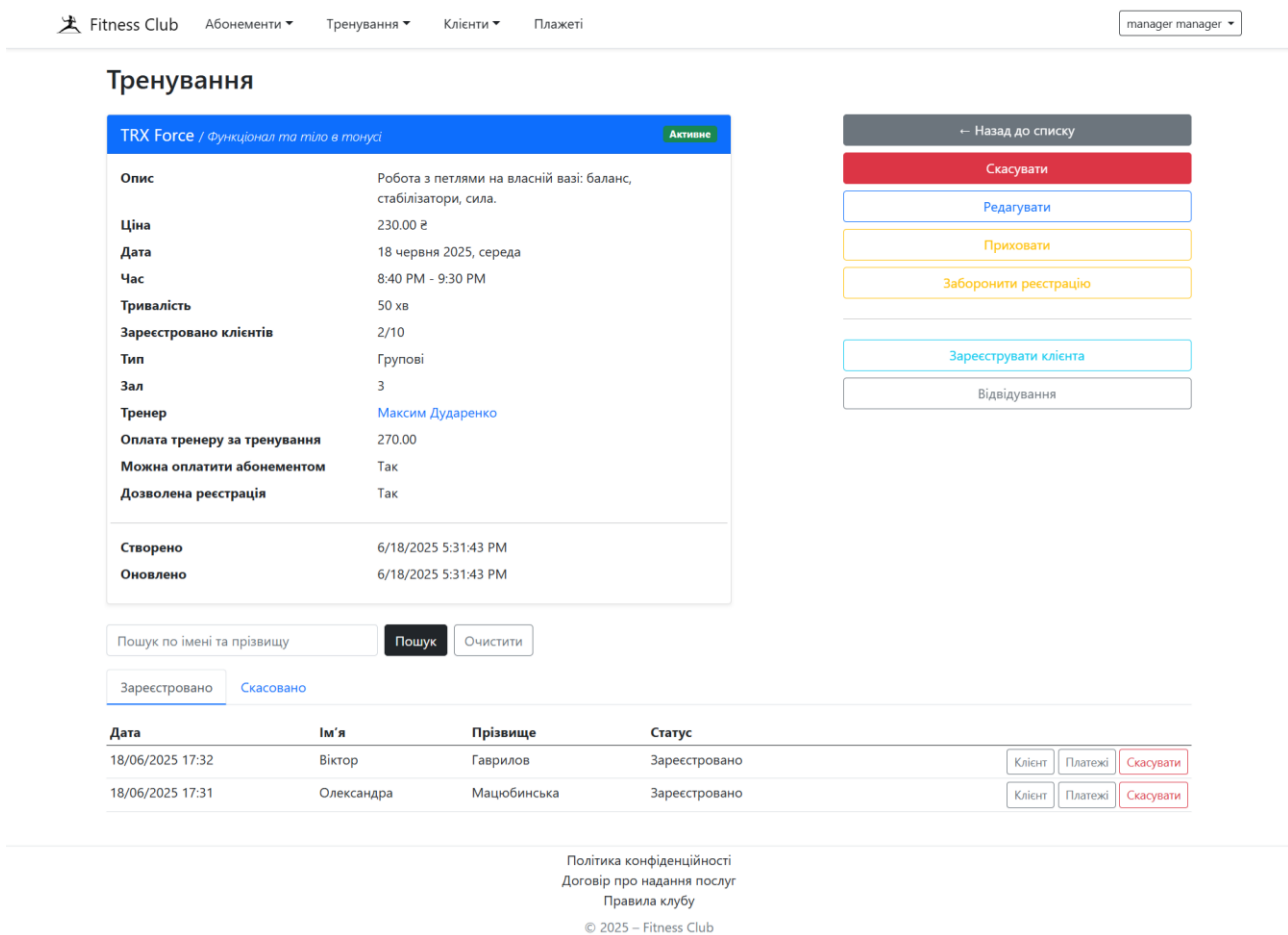


Рис. 3.35 – Сторінка детальної інформації тренування

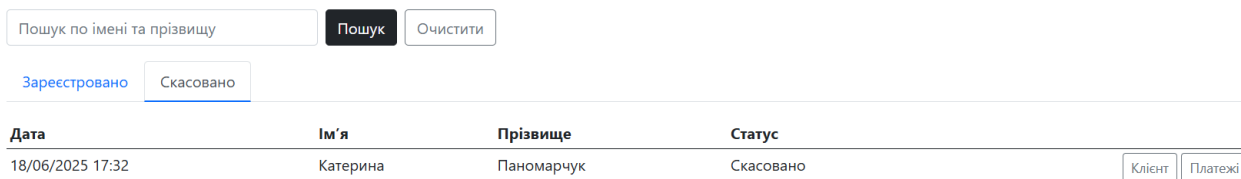


Рис. 3.36 – Вкладка скасованих відвідувань на сторінці тренування

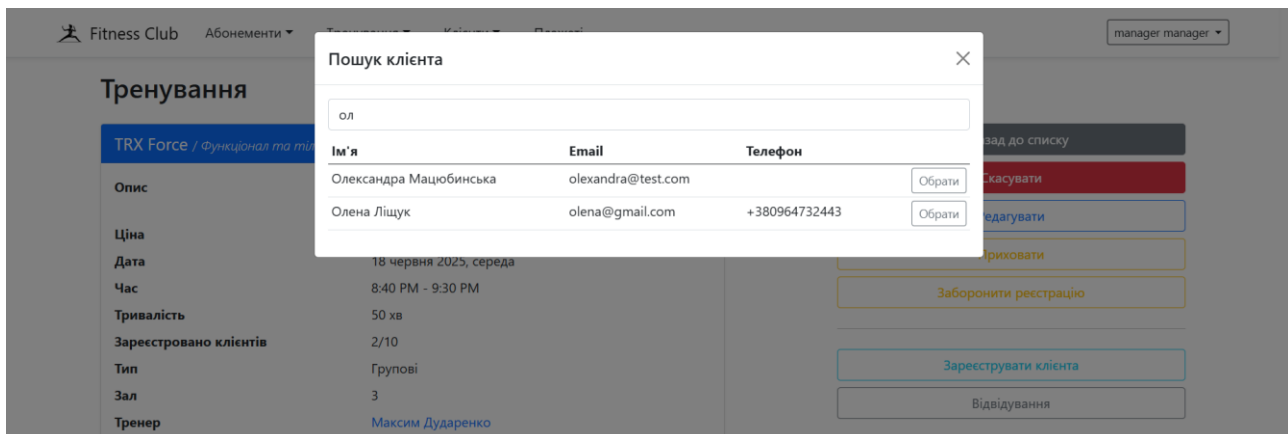


Рис. 3.37 – Вікно з пошуком клієнта для реєстрації на тренування

Абонементи

Тренування

Клієнти

Платежі

admin admin

Створення тренування

Категорія

Функціонал та тіло в тонусі

Шаблон тренування

Core Control

Дата та час початку

18.06.2025 21:41

Тренер

Оберіть тренера

Оберіть тренера

Максим Дударенко

Андрій Павлусенко

Валерій Марченко

Виберіть клієнта...

Пошук

Спосіб оплати:

☐ Картка
☐ Готівка
☐ Депозит
☐ Абонемент

Створити

← Назад до списку

Рис. 3.38 – Сторінка створення тренування

У випадку редагування тренування (рис. 3.39), якщо на ньому вже зареєстровані клієнти, дозволено змінювати лише дату, час, тренера та зал. Якщо ж реєстрацій не було — доступне повне редагування всіх параметрів.

Абонементи

Тренування

Клієнти

Платежі

manager manager

Редагувати тренування

Категорія

Йога, пілатес, стрейчинг

Шаблон тренування

Soft Mobility

Дата та час початку

19.06.2025 08:20

Тренер

Андрій Павлусенко

Зал

2

Клієнт (для персонального тренування)

Олена Ліщук

Зберегти

← Назад

Рис. 3.39 – Сторінка редагування тренування

Також у системі доступна сторінка перегляду всіх відвідувань тренувань (рис. 3.40), де вказано статус кожного запису (присутній, відсутній, скасований тощо), а також розміщено кнопку перегляду пов’язаних платежів.

Окрема сторінка присвячена перегляду всіх фінансових операцій, що здійснювались у системі (рис. 3.41). Тут реалізовано потужну систему фільтрації за клієнтом, менеджером, способом оплати (депозит, готівка, абонемент),

предметом оплати (абонемент, тренування) і статусом транзакції. Це дозволяє гнучко аналізувати історію фінансових дій.

Fitness Club

Абонементи

Тренування

Клієнти

Платежі

manager

manager

Відвідування тренувань

Пошук за тренуванням/клієнтом

Статус

ID відвідування

ID тренування

ID клієнта

дд.мм.рррр

18.06.2025

Нові

Фільтрувати

Очистити

ID	Клієнт	Тренування	Тренер	Оновлено	Статус	
d4b74214-d862-44f3-bfa5-33f314014ba2	client client	Cardio Burner	Олена Ковальчук	6/17/2025 2:40:01 AM	Зареєстровано	Платежі
dfb4ca72-b696-4b83-8de7-785b00aba612	client client	Callanetics	Максим Дударенко	6/17/2025 2:35:44 AM	Скасовано	Платежі
d1c2181a-1a36-4a03-ad98-1ed8d6592dcf	client client	Kettlebell	Валерій Марченко	6/17/2025 2:35:22 AM	Тренування скасовано	Платежі
c0b3175b-2632-4a52-98ae-f5b8276f7e69	client client	Power Individual	Олена Ковальчук	6/17/2025 2:31:15 AM	Зареєстровано	Платежі
1b2502d0-cc3f-4ab3-9660-e5a7aba100b0	client client	2 реєстрації	Андрій Павлусенко	6/13/2025 1:40:01 AM	Скасовано	Платежі
f9dec643-0aee-45b3-abe8-b998db75a748	client client	2 реєстрації	Андрій Павлусенко	6/13/2025 1:39:18 AM	Скасовано	Платежі
0e0cf27b-1e3f-4583-bd94-7add71f8c657	client client	7324601872	Валерій Марченко	6/13/2025 1:09:24 AM	Зареєстровано	Платежі
57076d99-3da2-49c9-bec9-b35a6b06cf25	client client	2 реєстрації	Андрій Павлусенко	6/13/2025 12:48:17 AM	Скасовано	Платежі
021444ed-f7c1-4729-8421-2371d781946e	client client	300 хвилин задоволення	Андрій Павлусенко	6/13/2025 12:34:08 AM	Скасовано	Платежі
8c51d016-e42d-4f6f-aecf-9f7ee4ff60af	Анна Васильківська	7324601872	Андрій Павлусенко	6/12/2025 11:50:00 PM	Тренування скасовано	Платежі
4b1349de-9076-447b-b86c-577a8ab49636	Анна Васильківська	2 реєстрації	Валерій Марченко	6/12/2025 9:10:02 PM	Тренування скасовано	Платежі

Рис. 3.40 – Сторінка зі списком відвідувань тренувань

Fitness Club

Абонементи

Тренування

Клієнти

Платежі

manager manager

Платежі

Пошук за клієнтом

Менеджер

Спосіб

Предмет

Статус

дд.мм.рррр

18.06.2025

ID абоненту

ID тренування/відвідуван

Нові

Фільтрувати

Очистити

Клієнт	Менеджер	Сума	Спосіб	Предмет	Статус	Дата
Артем Віщенко	manager manager	470.00 ₪	Готівка	Тренування	Оплачено	6/18/2025 6:26:41 PM
Олена Ліщук	manager manager	450.00 ₪	Карта	Тренування	Оплачено	6/18/2025 6:11:20 PM
Артем Віщенко	admin admin	480.00 ₪	Готівка	Тренування	Оплачено	6/18/2025 5:59:32 PM
Олена Ліщук	manager manager	230.00 ₪	Депозит	Тренування	Оплачено	6/18/2025 5:38:27 PM
Віктор Гаврилов	manager manager	230.00 ₪	Депозит	Тренування	Оплачено	6/18/2025 5:32:10 PM
Олександра Мацюбинська	manager manager	230.00 ₪	Карта	Тренування	Оплачено	6/18/2025 5:31:58 PM
Катерина Паномарчук	manager manager	230.00 ₪	Готівка	Тренування	Оплачено	6/18/2025 5:31:49 PM
client client	manager manager	600.00 ₪	Абонемент	Тренування	Оплачено	6/18/2025 5:30:12 PM
client client	manager manager	2000.00 ₪	Готівка	Депозит	Оплачено	6/18/2025 5:10:11 PM
client client		1000.00 ₪	Депозит	Абонемент	Оплачено	6/17/2025 6:40:34 PM
client client	admin admin	480.00 ₪	Депозит	Тренування	Оплачено	6/17/2025 2:40:01 AM
client client		210.00 ₪	Депозит	Тренування	Повернено	6/17/2025 2:35:22 AM
client client		210.00 ₪	Депозит	Тренування	Оплачено	6/17/2025 2:34:19 AM
client client		190.00 ₪	Депозит	Тренування	Оплачено	6/17/2025 2:33:34 AM
client client	admin admin	550.00 ₪	Карта	Тренування	Оплачено	6/17/2025 2:31:15 AM
client client		300.00 ₪	Депозит	Абонемент	Оплачено	6/17/2025 2:20:55 AM
Катерина Паномарчук	admin admin	1.80 ₪	Карта	Абонемент	Оплачено	6/13/2025 9:18:29 PM
client client		3.00 ₪	Депозит	Абонемент	Оплачено	6/13/2025 9:14:14 PM
client client		12.00 ₪	Абонемент	Тренування	Повернено	6/13/2025 1:40:01 AM
client client		12.00 ₪	Депозит	Тренування	Оплачено	6/13/2025 1:39:23 AM

1

2

3

4

5

>

Рис. 3.41 – Сторінка всіх фінансових операцій системи

Після входу до облікового запису адміністратора користувач потрапляє до головної панелі керування, яка відкриває доступ до всіх функцій системи. Оскільки адміністратор має найвищий рівень прав, йому доступні як власні інструменти, так і можливість перемикання до панелі менеджера для зручної навігації та роботи (рис. 3.42).

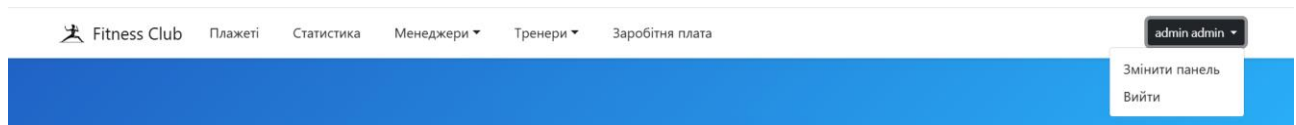


Рис. 3.42 – Панель навігації адміністратора

Однією з ключових вкладок є розділ «Платежі», яка перенаправляє користувача до вже раніше розглянутої сторінки з усіма транзакціями — дублювати її опис немає потреби. Далі адміністратор може перейти до сторінки перегляду всіх зареєстрованих у системі тренерів (рис. 3.43). Натиснувши на конкретного тренера, відкривається детальна сторінка з інформацією про нього (рис. 3.44), де відображається найближче тренування тренера, а також доступні дії: блокування, скидання паролю, редагування профілю та перегляд персонального розкладу.

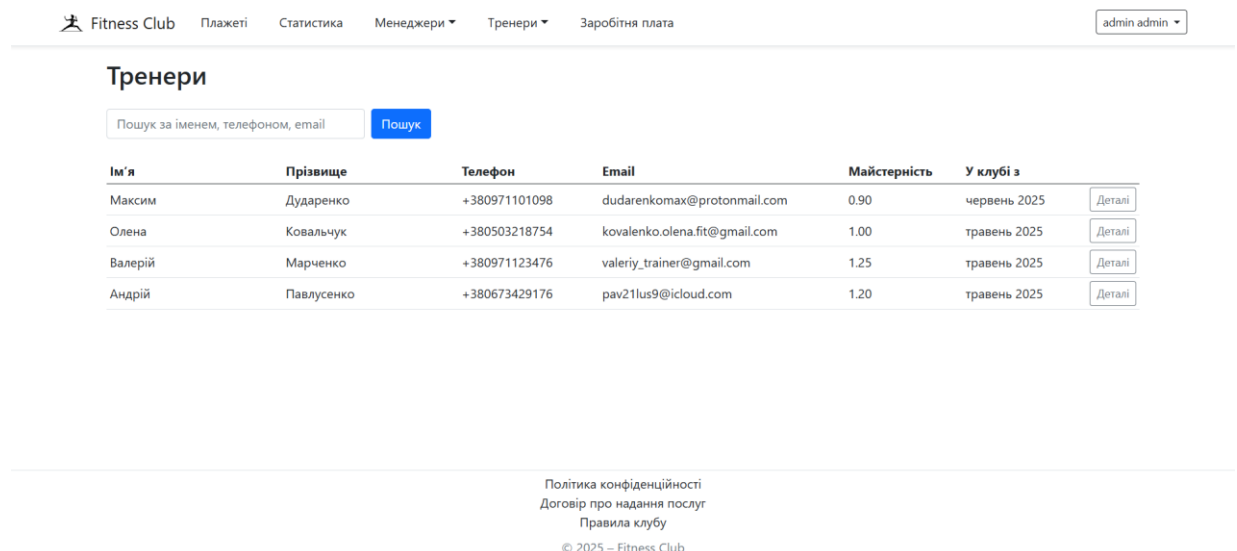


Рис. 3.43 – Сторінка усіх тренерів

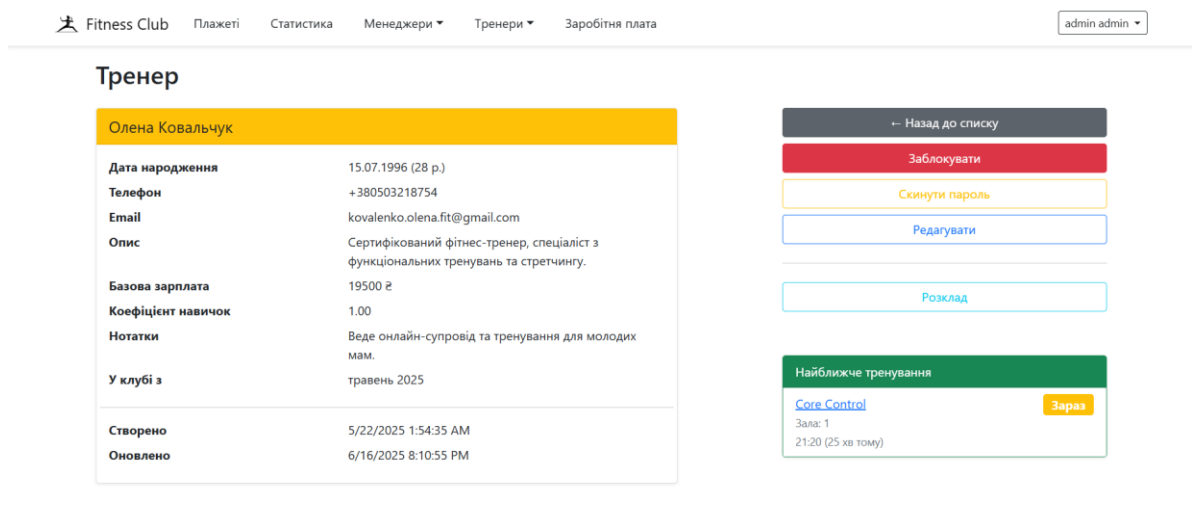


Рис. 3.44 – Сторінка детальної інформації тренера

При спробі заблокувати тренера відкривається модальне вікно з підтвердженням і вимогою вказати причину блокування, що забезпечує контроль над процесом (рис. 3.45).

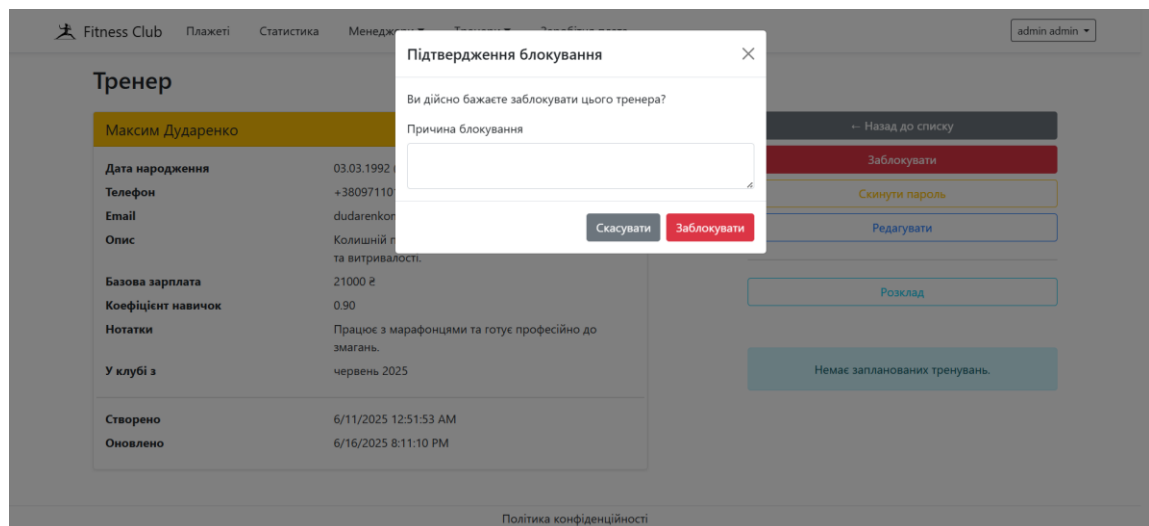


Рис. 3.45 – Вікно для підтвердження блокування

Сторінка перегляду всіх менеджерів реалізована в єдиному візуальному стилі з іншими списками користувачів (рис. 3.46). На сторінці детальної інформації про менеджера (рис. 3.47) адміністратор має змогу редагувати дані, блокувати обліковий запис, або переглянути список фінансових транзакцій, які були здійснені цим менеджером. Функція скидання паролю також доступна — при її активації з'являється вікно з полем для введення нового паролю (рис. 3.48).

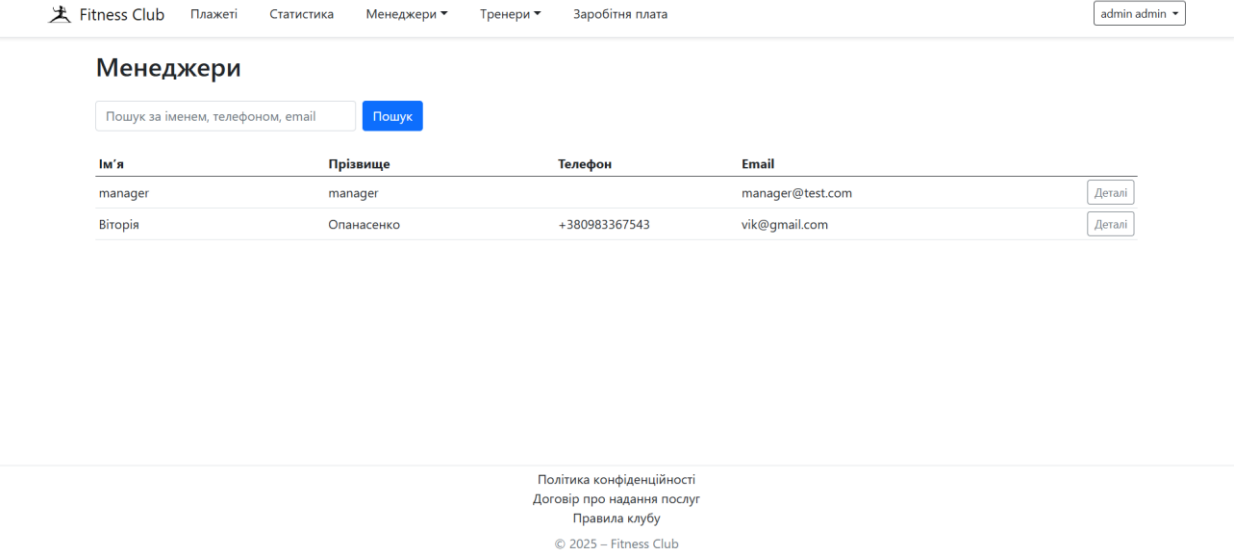


Рис. 3.46 – Сторінка зі списком менеджерів

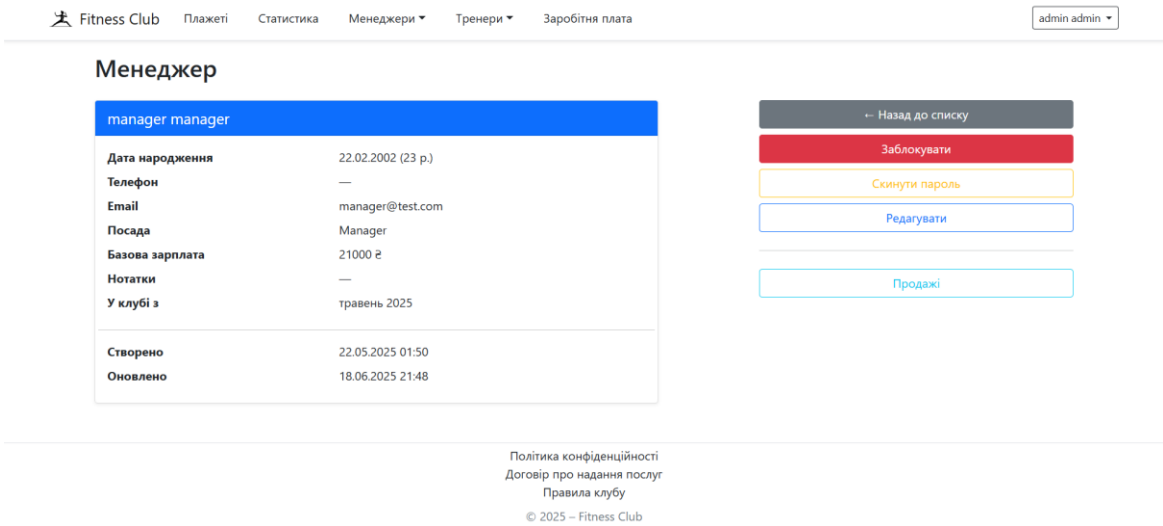


Рис. 3.47 – Сторінка деталей менеджера

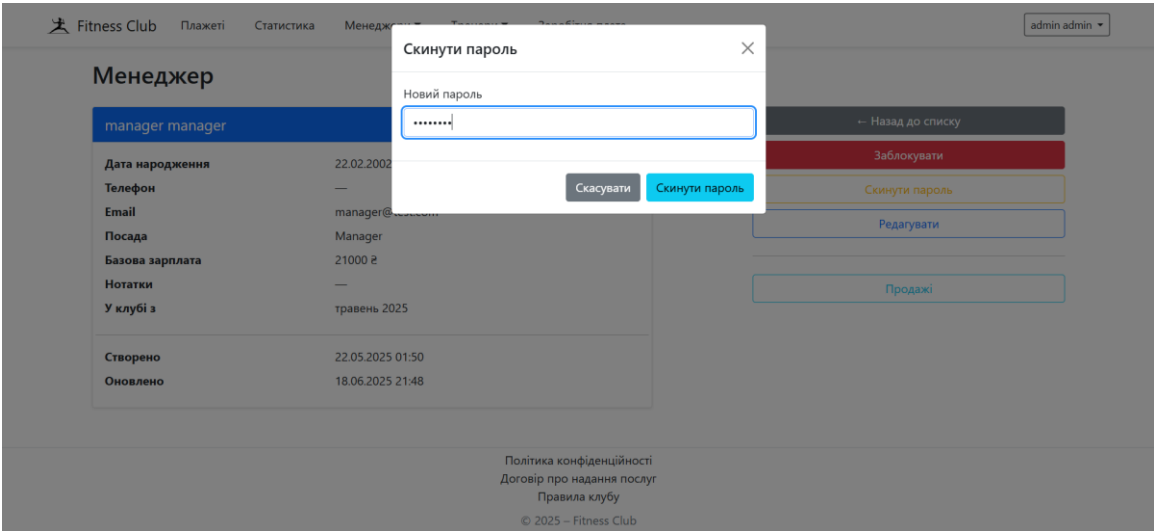


Рис. 3.48 – Вікно для скидання паролю

Окремий модуль системи присвячений перегляду статистики тренувань. Він дозволяє обрати будь-який місяць або рік і переглянути відповідні підсумки — загальну кількість тренувань, кількість клієнтів, популярні категорії та інші показники (рис. 3.49). Такий інструмент є надзвичайно корисним для оцінки завантаженості персоналу та планування майбутньої активності.

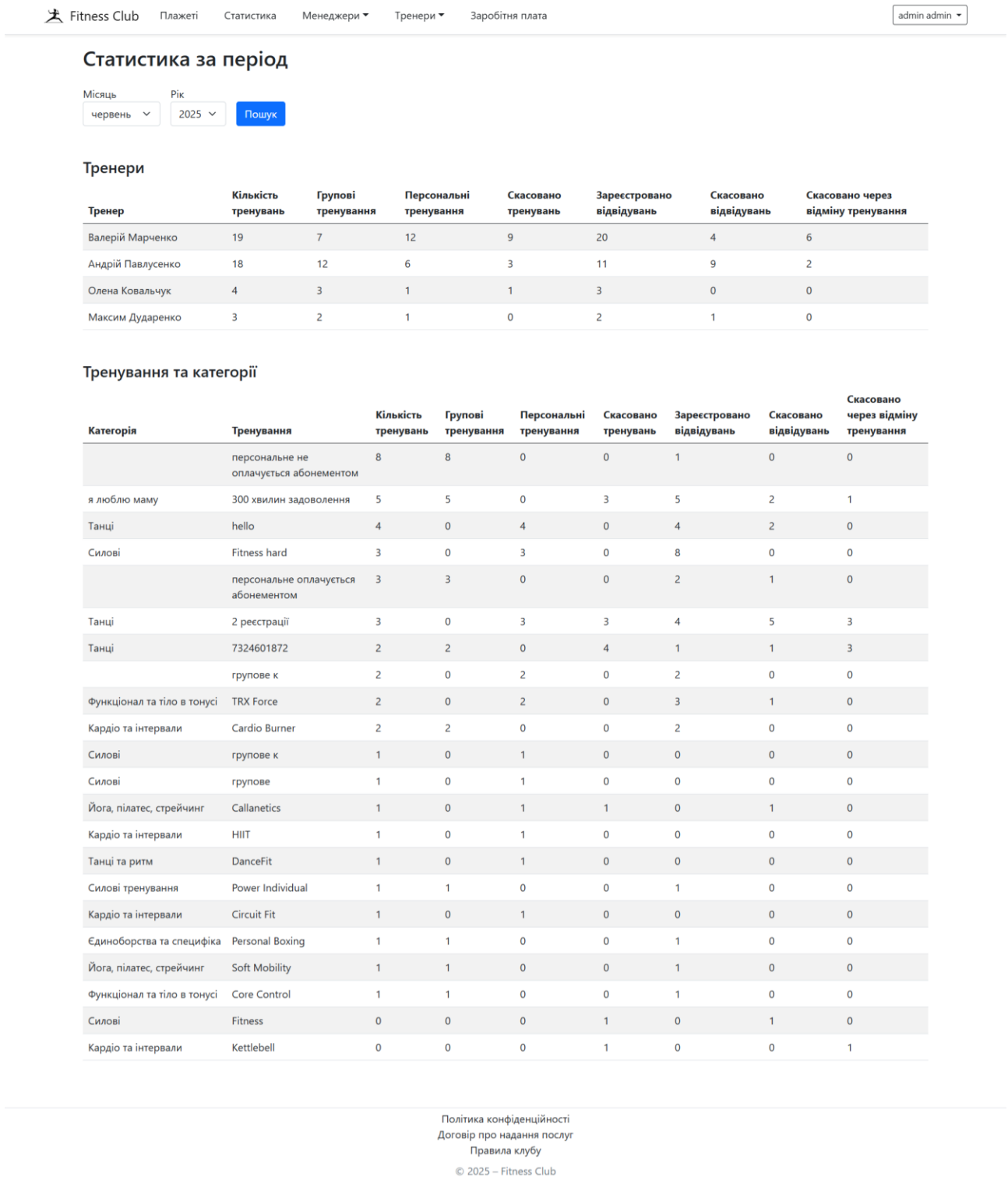


Рис. 3.49 – Сторінка з статистикою по тренуванням

Крім того, у системі реалізовано автоматичне обчислення заробітної плати для всього персоналу (рис. 3.50). Користувач може вибрати місяць і рік або довільний часовий інтервал. Заробітна плата менеджерів розраховується відповідно до фіксованої ставки, а для тренерів — на основі базової ставки з додатковою оплатою за кожне проведене тренування, що множиться на індивідуальний коефіцієнт навичок. Такий підхід дозволяє забезпечити прозору і гнучку систему мотивації персоналу.

Fitness Club

Платіжі

Статистика

Менеджери ▾

Тренери ▾

Заробітна плата

admin admin ▾

Заробітна плата

Рік

Місяць

2025 ▾

червень ▾

З

По

01.06.2025

30.06.2025

☐ Довільний період

Показати

Менеджери

manager manager

Основна ставка: 21000.00 ₴

21000.00 ₴

Вікторія Опанасенко

Основна ставка: 20000.00 ₴

20000.00 ₴

Тренери

Максим Дударенко

Основна ставка: 21000.00 ₴ | Коефіцієнт навичок: 0.90 | Групові: 1 | Персональні: 2 | За тренування: 918.00 ₴

Тренування

21918.00 ₴

Андрій Павлусенко

Основна ставка: 18000.00 ₴ | Коефіцієнт навичок: 1.20 | Групові: 6 | Персональні: 12 | За тренування: 3925.44 ₴

Тренування

21925.44 ₴

Валерій Марченко

Основна ставка: 20000.00 ₴ | Коефіцієнт навичок: 1.25 | Групові: 12 | Персональні: 7 | За тренування: 6450.38 ₴

Тренування

26450.38 ₴

Олена Ковальчук

Основна ставка: 19500.00 ₴ | Коефіцієнт навичок: 1.00 | Групові: 1 | Персональні: 3 | За тренування: 1370.00 ₴

Тренування

20870.00 ₴

Політика конфіденційності

Договір про надання послуг

Правила клубу

© 2025 – Fitness Club

Рис. 3.50 – Сторінка розрахунку заробітніх плат персоналу

ІПЗ.КР.Б –121 – 25 – ПЗ

86

3.3. Тестування роботи веб-додатку

Тестування роботи веб-додатку «Fitness Club» здійснювалося з метою перевірки стабільності функціонування системи, коректності реалізації основних бізнес-функцій, а також відповідності застосунку очікуваним вимогам користувачів. У процесі розробки застосовувалися переважно ручні методи тестування, а також елементи модульного та інтеграційного тестування в межах локального середовища розробника та після розгортання на сервері Azure Web App. Особливу увагу приділяли перевірки функціональності, що є критично важливою для повсякденного використання системи в умовах реального фітнес-клубу.

На першому етапі було проведено функціональне тестування, яке полягало у перевірці роботи основних можливостей системи: реєстрації та входу користувачів, запису на тренування, придбання абонементів, роботи зі сторінкою розкладу та перевірки ролей доступу. Функції перевірялись у різних комбінаціях вхідних даних, зокрема з некоректними або порожніми значеннями, з метою переконатися в наявності обробки помилок. Окремо тестувалися елементи авторизації з перевіркою на блокування, а також дії, які виконує користувач із роллю менеджера або адміністратора. Цей етап дозволив переконатися, що логіка ролей працює коректно, а користувач не має змоги виконувати заборонені дії.

На другому етапі було проведено інтеграційне тестування, у межах якого перевірялась взаємодія між різними компонентами: сервісами, контролерами, репозиторіями та інтерфейсом. Тестувалися запити до бази даних, відповідність даних, що відображаються на інтерфейсі, запис та оновлення інформації, облік залишку тренувань і статусів абонементів. Під час цього етапу виявлялись дрібні логічні помилки — зокрема, неправильне оновлення статусу при закінченні абонементу або неочікувана поведінка при покупці сімейного абонементу без наявної сім'ї. Ці недоліки були оперативно виправлені в рамках сервісного рівня.

Заключним етапом стало тестування після деплою на Azure, яке дозволило переконатися в коректності розгортання, стабільності роботи з базою даних та

підтримці HTTPS-з'єднання. Здійснювалась перевірка на різних пристроях і в популярних браузерях, щоб переконатися в адаптивності інтерфейсу та зручності навігації для кінцевих користувачів. Під час тестування були виявлені незначні технічні помилки, зокрема у вигляді некоректного відображення окремих елементів або неконсистентної поведінки при навігації, які були швидко усунені.

Для підтвердження працездатності основних функцій веб-додатку було проведено ручне тестування ключових сценаріїв взаємодії користувача із системою. Тестування охоплювало як дії клієнтів, так і менеджерів та адміністраторів, що дозволило перевірити логіку доступу, коректність обробки даних і відображення інформації в інтерфейсі. Нижче наведено результати тестування окремих функціональних блоків, що дозволяють оцінити відповідність фактичної поведінки системи очікуваній (табл. 3.1 – 3.8).

Таблиця 3.1

Тест «Реєстрація нового клієнта»

Кроки для перевірки	Заповнити форму реєстрації та натиснути 'Зареєструватись'
Вхідні дані	Ім'я, прізвище, email, пароль
Очікуваний результат	Користувача зареєстровано та перенаправлено до профілю
Фактичний результат	Відповідає

Таблиця 3.2

Тест «Авторизація клієнта»

Кроки для перевірки	Увести email і пароль на сторінці входу
Вхідні дані	Зареєстровані email і пароль
Очікуваний результат	Вхід до особистого кабінету
Фактичний результат	Відповідає

Таблиця 3.3

Тест «Купівля абонементу»

Кроки для перевірки	Вибрати абонемент і підтвердити покупку
Вхідні дані	Абонемент типу 'Місячний'
Очікуваний результат	Абонемент збережено в обліковці клієнта
Фактичний результат	Відповідає

Таблиця 3.4

Тест «Запис на тренування»

Кроки для перевірки	Натиснути 'Записатись' на доступне тренування
Вхідні дані	Активний абонемент і тренування
Очікуваний результат	Клієнта додано до списку учасників
Фактичний результат	Відповідає

Таблиця 3.5

Тест «Фільтрація тренувань за категорією»

Кроки для перевірки	Вибрати категорію у фільтрі на сторінці розкладу
Вхідні дані	Категорія 'Силові'
Очікуваний результат	Відображаються лише силові тренування
Фактичний результат	Відповідає

Таблиця 3.6

Тест «Заморожування абонементу менеджером»

Кроки для перевірки	Увійти як менеджер і заморозити активний абонемент
Вхідні дані	Абонемент клієнта
Очікуваний результат	Статус абонементу змінено на 'Заморожено'
Фактичний результат	Відповідає

Таблиця 3.7

Тест «Блокування клієнта адміністратором»

Кроки для перевірки	Увійти як адміністратор і заблокувати клієнта
Вхідні дані	Ідентифікатор клієнта
Очікуваний результат	Клієнт позначений як заблокований, доступ обмежено
Фактичний результат	Відповідає

Таблиця 3.8

Тест «Блокування клієнта адміністратором»

Кроки для перевірки	Увійти як менеджер, перейти на сторінку тренування та натиснути кнопку «Скасувати»
Вхідні дані	Ідентифікатор тренування
Очікуваний результат	Статус тренування змінено на «Скасовано», клієнтам повернуто оплату на депозит або абонемент
Фактичний результат	Відповідає

У результаті проведеного тестування веб-додатку «Fitness Club» було підтверджено його стабільну роботу та відповідність основним функціональним вимогам. Перевірка ключових сценаріїв взаємодії з системою продемонструвала правильність реалізації логіки доступу, обробки абонементів, запису на тренування та адміністрування. Виявлені незначні помилки були оперативно виправлені в процесі розробки, що дозволило підвищити загальну якість застосунку.

Висновки до розділу 3

У третьому розділі було детально розглянуто практичні аспекти роботи з розробленою системою управління фітнес-клубом. Було описано процес встановлення і налаштування програми на платформі Azure, що включає використання веб-сервісу Azure Web App для розгортання серверної частини та бази даних Azure SQL для зберігання інформації. Такий вибір інфраструктури забезпечив стабільну роботу додатку, його доступність з будь-якого пристрою через захищені HTTPS-з'єднання та можливість масштабування, що є критично важливим для зростаючої кількості користувачів та обсягів даних.

Особливу увагу було приділено аналізу структури інтерфейсу користувача. Було продемонстровано, як реалізовано навігацію, адаптивність до мобільних пристроїв і розмежування функціоналу для різних ролей у системі: клієнтів, тренерів, менеджерів та адміністраторів. Завдяки інтуїтивно зрозумілому інтерфейсу користувачі можуть легко взаємодіяти з усіма основними функціями, такими як перегляд розкладу, керування абонементом, здійснення платежів, адміністрування персоналу та аналітика, що підвищує ефективність роботи персоналу та зручність для клієнтів.

Окремим етапом стало тестування веб-додатку, в ході якого було проведено функціональні, інтеграційні та модульні перевірки. Тестування охоплювало як роботу окремих компонентів системи, так і їх взаємодію. Було виявлено незначні помилки, які були оперативно усунені в процесі

налагодження. Проведені тести підтвердили працездатність додатку відповідно до поставлених вимог і продемонстрували його надійність у типових сценаріях використання.

Таким чином, третій розділ засвідчує готовність розробленого програмного забезпечення до практичного використання, демонструє його зручність, функціональну повноту та відповідність сучасним вимогам до веб-орієнтованих систем управління. Це підтверджує ефективність прийнятих архітектурних рішень та їх доцільність у контексті реального використання.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було всебічно розроблено, спроектовано та реалізовано веб-орієнтовану систему управління фітнес-клубом, яка охоплює ключові аспекти автоматизації внутрішніх процесів закладу, забезпечує зручну взаємодію з клієнтами та персоналом і дозволяє ефективно адмініструвати розклад, абонементи та фінансові операції.

У першому розділі було проведено аналіз сучасних підходів до цифровізації фітнес-сфери, розглянуто існуючі програмні рішення, їх функціональні можливості та недоліки. Особливу увагу було приділено вивченню вимог до подібних систем з боку різних категорій користувачів: клієнтів, тренерів, менеджерів та адміністраторів. Це дозволило чітко сформулювати цілі проекту, обґрунтувати його актуальність та визначити ключові функціональні компоненти майбутньої системи.

Другий розділ присвячено безпосередньо проектуванню веб-додатку. Було створено UML-діаграми, які описують варіанти використання, структуру даних і логіку роботи системи. Розроблено модель бази даних, реалізовано ключові бізнес-алгоритми, а також обґрунтовано використання патернів проектування для забезпечення масштабованості та підтримуваності коду.

У третьому розділі було розглянуто практичні аспекти розгортання, тестування й використання додатку. Здійснено опис процесу налаштування середовища, проаналізовано структуру користувацького інтерфейсу та надано детальний огляд функціоналу для кожної ролі. Завдяки адаптивному дизайну, система зручна для мобільного використання клієнтами й тренерами, а також забезпечує повноцінну роботу з функціоналом адміністратора на десктопних пристроях. Тестування охопило як функціональну, так і інтеграційну перевірку, що підтвердило відповідність реалізації технічному завданню.

Загалом, розроблена система задовольняє вимоги до сучасних інформаційних рішень у галузі фітнес-послуг, є функціонально повною, зручною у використанні та відкритою для подальшого вдосконалення й масштабування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sport Life. URL: <https://sportlife.ua/uk/>;
2. Wings Fitness Club. URL: <https://beautyprosoftware.com/b/543958/order/type>;
3. LuckyFit. URL: <https://lucky-fit.com/>;
4. Introduction to Razor Pages in ASP.NET Core. URL: <https://learn.microsoft.com/en-us/aspnet/core/razor-pages/>;
5. ASP.NET documentation. URL: <https://learn.microsoft.com/en-us/aspnet/core/>;
6. JSON. URL: <https://www.json.org/json-en.html>;
7. Martin R. C. Agile Software Development, Principles, Patterns, and Practices. 2nd ed. Prentice Hall, 2002. 529 p. ;
8. Azure SQL documentation - Azure SQL. URL: <https://learn.microsoft.com/en-us/azure/azure-sql/>;
9. Azure App Service documentation - Azure App Service. URL: <https://learn.microsoft.com/uk-ua/azure/app-service/>;
10. Get started with GitLab CI/CD. *GitLab Docs*. URL: <https://docs.gitlab.com/ci/>;
11. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2002. 560 p.;
12. PostgreSQL. URL: <https://www.postgresql.org/>;
13. MySQL. URL: <https://www.mysql.com/>;
14. Visual Studio 2022 IDE. URL: <https://visualstudio.microsoft.com/vs/>;
15. Transact-SQL Reference (Database Engine) - SQL Server. URL: <https://learn.microsoft.com/en-us/sql/t-sql/>;
16. Getting Started - EF Core. URL: <https://learn.microsoft.com/en-us/ef/core/get-started/>;
17. C# Guide - .NET managed language. URL: <https://learn.microsoft.com/en-us/dotnet/csharp/>;
18. Inheritance - EF Core. URL: <https://learn.microsoft.com/en-us/ef/core/modeling/inheritance>;

19. Ajax - Glossary | MDN. *MDN Web Docs*. URL: <https://developer.mozilla.org/en-US/docs/Glossary/AJAX>;
20. Using HTTP cookies - HTTP | MDN. *MDN Web Docs*. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Cookies>;
21. AuthenticationHttpContextExtensions.SignInAsync Method (Microsoft.AspNetCore.Authentication). URL: <https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.authentication.authenticationhttpcontextextensions.signinasync?view=aspnetcore-8.0>;
22. ViewUserControl.ViewBag Property (System.Web.Mvc). URL: <https://learn.microsoft.com/en-us/dotnet/api/system.web.mvc.viewusercontrol.viewbag?view=aspnet-mvc-5.2>;
23. X.PagedList. *GitHub*. URL: <https://github.com/dncuug/X.PagedList/wiki>;
24. Modal. *Bootstrap*. URL: <https://getbootstrap.com/docs/4.0/components/modal/>;
25. Pedersen T. HTTPS, HTTP over TLS. *Encyclopedia of Cryptography and Security*. Boston, MA, 2011. P. 569–570. URL: https://doi.org/10.1007/978-1-4419-5906-5_288;
26. Introduction. *Bootstrap*. URL: <https://getbootstrap.com/docs/4.1/getting-started/introduction/>;
27. Fowler M. UML distilled: A brief guide to the standard modeling language. 3rd ed. Boston, Mass : Addison-Wesley, 2004. 175 p;
28. Мартін Р. Чиста архітектура: мистецтво розробки програмного забезпечення. Фабула, 2019. 416 с. ;
29. Лукашевич В.В., Вакалюк Т.А. Актуальність теми розробки системи управління фітнес-клубом. Інформаційні технології – 2025: зб. тез XII Всеукраїнської науково-практичної конференції молодих учених, 15 трав. 2025 р., м. Київ / Київський столичний університет імені Бориса Грінченка; Відповід. за вип.: М.М. Астаф'єва, Д.М. Бодненко, О.М. Глушак, І.В. Машкіна, О.В. Локазюк, В.В. Прошкін, С.М. Шевченко. Київ : Київський столичний університет імені Бориса Грінченка, 2025. С. 159-160. <https://zcit.kubg.edu.ua/index.php/journal/issue/view/13/23>;

30. Лукашевич В.В., Вакалюк Т.А. Переваги та недоліки ASP.NET MVC. Інформаційні технології – 2025: зб. тез XII Всеукраїнської науково-практичної конференції молодих учених, 15 трав. 2025 р., м. Київ / Київський столичний університет імені Бориса Грінченка; Відповід. за вип.: М.М. Астаф'єва, Д.М. Бодненко, О.М. Глушак, І.В. Машкіна, О.В. Локазюк, В.В. Прошкін, С.М. Шевченко. Київ : Київський столичний університет імені Бориса Грінченка, 2025. С. 158-159.
<https://zcit.kubg.edu.ua/index.php/journal/issue/view/13/23>.

ДОДАТКИ

					ІПЗ.КР.Б –121 – 25 – ІЗ	
						96

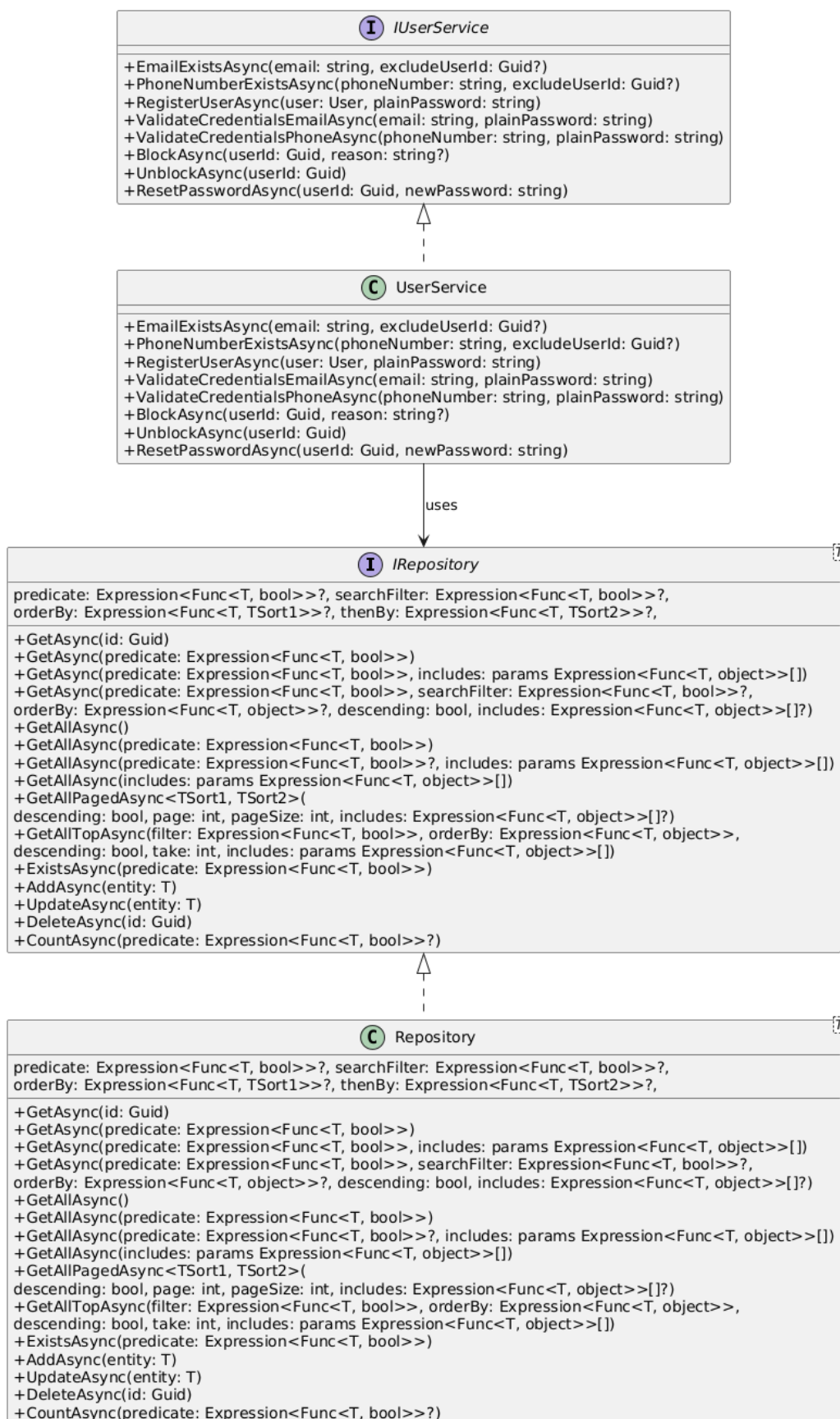


Рис. А.1 – Діаграма класів реалізованого паттерну Репозиторій

IRepository.cs

```

public class Repository<TEntity> : IRepository<TEntity> where TEntity : class
{
    protected readonly ApplicationDbContext _context;

    public Repository(ApplicationDbContext context)
    {
        _context = context;
    }

    public async Task<IPagedList<TEntity>> GetAllPagedAsync<TSort1, TSort2>(
        Expression<Func<TEntity, bool>>? predicate = null,
        Expression<Func<TEntity, bool>>? searchFilter = null,
        Expression<Func<TEntity, TSort1>>? orderBy = null,
        Expression<Func<TEntity, TSort2>>? thenBy = null,
        bool descending = false,
        int page = 1,
        int pageSize = PaginationDefaults.PageSize,
        Expression<Func<TEntity, object>>[]? includes = null)
    {
        IQueryable<TEntity> query = _context.Set<TEntity>().AsExpandable();

        if (includes != null)
            foreach (var include in includes)
                query = query.Include(include);

        if (predicate != null)
            query = query.Where(predicate);

        if (searchFilter != null)
            query = query.Where(searchFilter);

        if (orderBy != null)
        {
            query = descending
                ? query.OrderByDescending(orderBy)
                : query.OrderBy(orderBy);

            if (thenBy != null)
            {
                query = descending
                    ? ((IOrderedQueryable<TEntity>)query).ThenByDescending(thenBy)
                    : ((IOrderedQueryable<TEntity>)query).ThenBy(thenBy);
            }
        }
        else
            query = query.OrderBy(e => true);

        return await query.ToPagedListAsync(page, pageSize);
    }

    public async Task<List<TEntity>> GetAllTopAsync(
        Expression<Func<TEntity, bool>> filter,
        Expression<Func<TEntity, object>> orderBy,
        bool descending,
        int take,
        params Expression<Func<TEntity, object>>[] includes)
    {
        IQueryable<TEntity> query = _context.Set<TEntity>();

        foreach (var include in includes)
        {

```

```

        query = query.Include(include);
    }

    query = query.Where(filter);

    query = descending
        ? query.OrderByDescending(orderBy)
        : query.OrderBy(orderBy);

    return await query.Take(take).ToListAsync();
}

public async Task<List<TEntity>> GetAllAsync()
{
    return await _context.Set<TEntity>().ToListAsync();
}

public async Task<List<TEntity>> GetAllAsync(Expression<Func<TEntity, bool>>?
predicate = null,
                                params Expression<Func<TEntity,
object>>[] includes)
{
    IQueryable<TEntity> query = _context.Set<TEntity>();

    foreach (var include in includes)
        query = query.Include(include);

    if (predicate != null)
        query = query.Where(predicate);

    return await query.ToListAsync();
}

public async Task<List<TEntity>> GetAllAsync(params Expression<Func<TEntity,
object>>[] includes)
{
    IQueryable<TEntity> query = _context.Set<TEntity>();

    foreach (var include in includes)
        query = query.Include(include);

    return await query.ToListAsync();
}

public async Task<List<TEntity>> GetAllAsync(Expression<Func<TEntity, bool>>
predicate)
{
    return await _context.Set<TEntity>().Where(predicate).ToListAsync();
}

public async Task<TEntity?> GetAsync(Guid id)
{
    var entity = await _context.Set<TEntity>().FindAsync(id);

    return entity;
}

public async Task<TEntity?> GetAsync(Expression<Func<TEntity, bool>> predicate,
                                params Expression<Func<TEntity, object>>[]
includes)
{
    IQueryable<TEntity> query = _context.Set<TEntity>();

```

```

        foreach (var include in includes)
            query = query.Include(include);

        var result = await query.FirstOrDefaultAsync(predicate);

        return result;
    }

    public async Task<TEntity?> GetAsync(Expression<Func<TEntity, bool>> predicate)
    {
        var result = await _context.Set<TEntity>().FirstOrDefaultAsync(predicate);

        return result;
    }

    public async Task<TEntity?> GetAsync(
        Expression<Func<TEntity, bool>> predicate,
        Expression<Func<TEntity, bool>>? searchFilter = null,
        Expression<Func<TEntity, object>>? orderBy = null,
        bool descending = false,
        Expression<Func<TEntity, object>>[]? includes = null)
    {
        IQueryable<TEntity> query = _context.Set<TEntity>();

        query = query.Where(predicate);

        if (searchFilter != null)
            query = query.Where(searchFilter);

        if (orderBy != null)
        {
            query = descending
                ? query.OrderByDescending(orderBy)
                : query.OrderBy(orderBy);
        }

        if (includes != null)
            foreach (var include in includes)
                query = query.Include(include);

        return await query.FirstOrDefaultAsync();
    }

    public async Task<TEntity> AddAsync(TEntity entity)
    {
        if (entity is IAuditable audit)
        {
            var now = DateTime.UtcNow;
            audit.CreatedAt = now;
            audit.UpdatedAt = now;
        }

        TrimEntityStrings(entity);

        _context.Set<TEntity>().Add(entity);
        var affected = await _context.SaveChangesAsync();

        if (affected == 0)
            throw new SaveFailedException(typeof(TEntity).Name);
    }

```

```

        return entity;
    }

    public async Task DeleteAsync(Guid id)
    {
        var entity = await GetAsync(id);

        _context.Set<TEntity>().Remove(entity);
        var affected = await _context.SaveChangesAsync();

        if (affected == 0)
            throw new DeleteFailedException(typeof(TEntity).Name);
    }

    public async Task<TEntity> UpdateAsync(TEntity entity)
    {
        if (entity is IAuditable audit)
            audit.UpdatedAt = DateTime.UtcNow;

        TrimEntityStrings(entity);

        _context.Set<TEntity>().Update(entity);
        var affected = await _context.SaveChangesAsync();

        if (affected == 0)
            throw new SaveFailedException(typeof(TEntity).Name);

        return entity;
    }

    public async Task<bool> ExistsAsync(Expression<Func<TEntity, bool>> predicate)
    {
        return await _context.Set<TEntity>().AnyAsync(predicate);
    }

    public async Task<int> CountAsync(Expression<Func<TEntity, bool>>? predicate =
null)
    {
        return predicate == null
            ? await _context.Set<TEntity>().CountAsync()
            : await _context.Set<TEntity>().CountAsync(predicate);
    }

    private void TrimEntityStrings(TEntity entity)
    {
        var stringProperties = typeof(TEntity)
            .GetProperties(BindingFlags.Public | BindingFlags.Instance)
            .Where(p => p.CanRead && p.CanWrite && p.PropertyType == typeof(string));

        foreach (var prop in stringProperties)
        {
            var value = (string)prop.GetValue(entity);
            if (!string.IsNullOrEmpty(value))
            {
                prop.SetValue(entity, value.Trim());
            }
        }
    }
}

```

_Layout.cshtml

```
@{
    var forceDesktop = User?.IsInRole("Admin") == true || User?.IsInRole("Manager") ==
true;
}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    @if (forceDesktop)
    {
        <meta name="viewport" content="width=1200">
    }
    else
    {
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    }
    <title>@ViewData["Title"] - Fitness Club</title>
    <link rel="icon" type="image/svg+xml" href="~/images/logo.svg" />

    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css"/>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.10.5/font/bootstrap-icons.css">
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/Web.styles.css" asp-append-version="true" />
</head>
<body>
    <header class="header-sticky">
        <nav class="navbar navbar-expand-sm navbar-light bg-white border-bottom
shadow-sm px-3 px-lg-5">
            <div class="container-fluid">
                <a class="navbar-brand d-flex align-items-center gap-2" asp-area=""
asp-controller="Home" asp-action="Index">
                    
                    Fitness Club
                </a>

                <button class="navbar-toggler" type="button" data-bs-
toggle="offcanvas" data-bs-target="#mobileMenu">
                    <span class="navbar-toggler-icon"></span>
                </button>

                <div class="collapse navbar-collapse d-none d-sm-flex justify-content-
between">
                    @await Html.PartialAsync("_NavigationContent")
                </div>
            </div>
        </nav>

        <!-- Mobile Offcanvas -->
        <div class="offcanvas offcanvas-end tabindex="-1" id="mobileMenu">
            <div class="offcanvas-header justify-content-end align-items-center">
                <button type="button" class="btn-close ms-auto p-1 mt-2 me-2" data-bs-
dismiss="offcanvas" aria-label="Close"></button>
            </div>
            <div class="offcanvas-body">
                @await Html.PartialAsync("_NavigationContent")
            </div>
        </div>
    </header>
```

```

<main role="main" class="pb-4 pt-3 container">
    @RenderBody()
</main>

<footer class="border-top w-100 text-muted">
    <div class="container text-center py-2">
        <div class="d-flex flex-column mb-3 align-items-center gap-2">
            <a class="text-decoration-none text-dark py-0 my-0" style="line-
height: 1;" asp-area="" asp-controller="Home" asp-action="Index">Політика
конфіденційності</a>
            <a class="text-decoration-none text-dark py-0 my-0" style="line-
height: 1;" asp-area="" asp-controller="Home" asp-action="Index">Договір про надання
послуг</a>
            <a class="text-decoration-none text-dark py-0 my-0" style="line-
height: 1;" asp-area="" asp-controller="Home" asp-action="Index">Правила клубу</a>
        </div>

        <div class="text-muted py-0 my-0" style="line-height: 1;">
            &copy; 2025 - Fitness Club
        </div>

    </div>
</footer>

<div class="toast-container position-fixed top-0 end-0 mt-5 p-3" style="z-index:
1100;">
    @if (TempData["Error"] != null)
    {
        <div class="toast align-items-center text-bg-danger border-0" role="alert"
aria-live="assertive" aria-atomic="true" data-bs-delay="5000" data-bs-autohide="true">
            <div class="d-flex">
                <div class="toast-body">
                    @TempData["Error"]
                </div>
                <button type="button" class="btn-close btn-close-white me-2 m-
auto" data-bs-dismiss="toast" aria-label="Закрити"></button>
            </div>
        </div>
    }
    @if (TempData["Success"] != null)
    {
        <div class="toast align-items-center text-bg-success border-0"
role="alert" aria-live="assertive" aria-atomic="true" data-bs-delay="5000" data-bs-
autohide="true">
            <div class="d-flex">
                <div class="toast-body">
                    @TempData["Success"]
                </div>
                <button type="button" class="btn-close btn-close-white me-2 m-
auto" data-bs-dismiss="toast" aria-label="Закрити"></button>
            </div>
        </div>
    }
</div>

<script>
    document.addEventListener('DOMContentLoaded', function () {
        var toastElList = [].slice.call(document.querySelectorAll('.toast'));
        toastElList.forEach(function (toastEl) {
            var toast = new bootstrap.Toast(toastEl);
            toast.show();
        });
    });
</script>

```

```
<script src=~\lib\jquery\dist\jquery.min.js"></script>
<script src=~\lib\bootstrap\dist\js\bootstrap.bundle.min.js"></script>
<script src=~\js\site.js" asp-append-version="true"></script>
@await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```