МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ Економіки та бізнес-освіти

Кафедра Економіки та цифрового бізнесу

Спеціальність 122 «Комп'ютерні науки»

Форма навчання Денна

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Павлиша Кирила Дмитровича
(прізвище, ім'я, по батькові здобувача)

на тему Проєктування та розробка інформаційної системи управління спортивним клубом з використанням web-технологій

(повна назва теми)

за матеріалами

(повна назва бази дослідження)

науковий керівник к.е.н., доцент _____ Соловйова В.В.
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від червня 2026 р. №
Завідувач кафедри _____
(підпис)
к.е.н., доцент Радько В.М.
наук. ступень, вчене звання прізвище, ініціали

Кривий Ріг – 2026

ЗАТВЕРДЖЕНО
Наказ Міністерства освіти і науки, молоді та спорту України
29 березня 2012 року № 384
Форма № Н-9.01

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ ТА БІЗНЕС-ОСВІТИ
(повне найменування вищого навчального закладу)

Кафедра економіки та цифрового бізнесу
Освітній ступінь бакалавр
Спеціальність 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри _____ В.М. Радько

"30" березня 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ ЗДОБУВАЧУ

Павлишу Кирилу Дмитровичу

1. Тема роботи Проєктування та розробка інформаційної системи управління спортивним клубом з використанням web-технологій
науковий керівник роботи к.е.н., доцент Соловйова В.В. ,
затверджені наказом вищого навчального закладу від «23» березня 2026 р. № 193-ст (д/ф)

2. Строк подання здобувачем роботи 29.05.2026р.

3. Зміст кваліфікаційної бакалаврської роботи, об'єкт, предмет та мета дослідження:

Розділ 1 Аналіз діяльності спортивного клубу та обґрунтування розробки інформаційної системи з використанням web-технологій
Розділ 2 Інформаційне та математичне забезпечення інформаційної системи управління спортивним клубом
Розділ 3 Розробка та тестування інформаційної системи управління спортивним клубом

Об'єкт дослідження – процес управління діяльністю спортивного клубу та взаємодія між його учасниками (адміністраторами, тренерами, клієнтами) засобами веб-технологій.

Предмет дослідження – методи та засоби проєктування і розробки інформаційної системи управління спортивним клубом з використанням веб-технологій.

Мета кваліфікаційної бакалаврської роботи – спроєктувати та розробити інформаційну систему управління спортивним клубом, що забезпечує автоматизацію процесів реєстрації клієнтів, управління абонементами, складання розкладу тренувань і комунікації між учасниками клубу на основі сучасних веб-технологій.

4. Дата видачі завдання 03.04.2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	до 17.04.2026р.	
2	Підготовка розділу 2	до 08.05.2026р.	
3	Підготовка розділу 3	до 25.05.2026р.	
4	Реєстрація завершеної кваліфікаційної роботи	до 29.05.2026р.	
5	Отримання відгуку від наукового керівника	04.06.2026р.	
6	Отримання зовнішньої рецензії	05.06.2026р.	
7	Попередній захист кваліфікаційної роботи на кафедрі	08.06.2026р.	
8	Перевірка кваліфікаційної роботи на плагіат	12.06.2026р.	
9	Допуск кафедрою кваліфікаційної роботи до захисту	12.06.2026р.	
10	Підготовка студента до захисту в ЕК	до 19.06.2026р.	

Завдання підготував науковий керівник _____ Соловйова В.В.
(підпис) (прізвище та ініціали)

Завдання одержав здобувач _____ Павлиш К.Д.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Робота містить 76 сторінок, 8 рисунків, 34 джерела, 19 таблиць і 11 додатків.

Об'єкт дослідження: процес управління діяльністю спортивного клубу та взаємодія між його учасниками (адміністраторами, тренерами, клієнтами) засобами веб-технологій.

Предмет дослідження: методи та засоби проєктування і розробки інформаційної системи управління спортивним клубом з використанням веб-технологій.

Мета роботи: спроєктувати та розробити інформаційну систему управління спортивним клубом, що забезпечує автоматизацію процесів реєстрації клієнтів, управління абонементами, складання розкладу тренувань і комунікації між учасниками клубу на основі сучасних веб-технологій.

Методи дослідження: системний аналіз, метод порівняльного аналізу, UML-моделювання, об'єктно-орієнтоване проєктування, прототипування інтерфейсу, методи тестування програмного забезпечення.

У першому розділі проаналізовано предметну область: досліджено бізнес-моделі та функції спортивного клубу, здійснено порівняльний аналіз п'яти вітчизняних веб-ресурсів і чотирьох зарубіжних платформ (Mindbody, TeamUp, Glofox, Pike13), формалізовано шість ключових бізнес-процесів та сформовано специфікацію вимог – 15 функціональних і 12 нефункціональних вимог за принципом SMART та методом MoSCoW.

У другому розділі виконано інформаційне та математичне забезпечення: розроблено UML-діаграми (Use Case – 16 UC, Sequence, Activity), спроєктовано реляційну базу даних із восьми таблиць у 3НФ, формалізовано три алгоритми (перевірки конфліктів розкладу, управління абонементом, скасування запису) та обґрунтовано п'ятирівневу архітектуру системи.

У третьому розділі здійснено програмну реалізацію та тестування. Серверну частину розроблено на Django 4.2 + Django REST Framework з JWT-автентифікацією та RBAC; клієнтську – на React 18 з адаптивним інтерфейсом

для трьох рольових сценаріїв. За результатами трирівневого тестування (модульне, інтеграційне, функціональне) реалізовано 13 з 15 функціональних вимог (86,7%), усі Must-вимоги виконано у повному обсязі. Середній час відгуку API – 0,43 с.

Результати та їх новизна: наукова новизна роботи полягає у комплексному підході до проєктування ІС управління спортивним клубом, що охоплює повний цикл – від аналізу бізнес-процесів до реалізації та тестування – з урахуванням специфіки вітчизняних спортивних закладів та актуального стану ринку (2026 р.).

Рекомендації щодо використання: розроблена система може бути впроваджена у комерційному спортивному клубі для автоматизації операційної діяльності та використана як основа для подальшого розвитку: мобільного застосунку, повноцінної платіжної інтеграції, підтримки мережі клубів.

Область застосування: комерційні та некомерційні спортивні клуби, фітнес-центри, спортивні студії.

ВЕБ-ЗАСТОСУНОК, ІНФОРМАЦІЙНА СИСТЕМА, РОЗКЛАД ТРЕНУВАНЬ, СПОРТИВНИЙ КЛУБ, УПРАВЛІННЯ АБОНЕМЕНТАМИ, BPMN, DJANGO, JWT, POSTGRESQL, RBAC, REACT, REST API, UML.

ЗМІСТ

стор.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 7

ВСТУП 8

РОЗДІЛ 1. АНАЛІЗ ДІЯЛЬНОСТІ СПОРТИВНОГО КЛУБУ ТА ОБҐРУНТУВАННЯ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ З ВИКОРИСТАННЯМ WEB-ТЕХНОЛОГІЙ 11

1.1. Характеристика діяльності спортивного клубу 11

1.2. Аналіз існуючих інформаційних систем та веб-рішень у діяльності спортивних клубів 14

1.3. Аналіз бізнес-процесів спортивного клуб 17

1.4. Формування вимог до інформаційної системи управління спортивним клубом 21

Висновки до розділу 1 30

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ СПОРТИВНИМ КЛУБОМ 32

2.1. Концептуальне та логічне моделювання системи управління спортивним клубом 32

2.2. Проєктування бази даних інформаційної системи 35

2.3. Алгоритмічне забезпечення інформаційної системи 37

2.4. Архітектура системи та проєктування користувацького інтерфейсу 40

Висновки до розділу 2 43

РОЗДІЛ 3. РОЗРОБКА ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ СПОРТИВНИМ КЛУБОМ 45

3.1. Вибір та обґрунтування технологій і інструментів 45

3.2. Реалізація серверної та клієнтської частини 48

3.3. Інтеграція з базою даних 53

3.4. Тестування та оцінка якості інформаційної системи 55

Висновки до розділу 3 58

ВИСНОВКИ 59

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 63

ДОДАТКИ 67

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

BPMN (Business Process Model and Notation) – нотація моделювання бізнес-процесів.

UML (Unified Modeling Language) – уніфікована мова моделювання.

JWT (JSON Web Token) – токен автентифікації та передачі даних.

RBAC (Role-Based Access Control) – розмежування доступу на основі ролей.

SRS (Software Requirements Specification) – специфікація вимог до програмного забезпечення.

MVP (Minimum Viable Product) – мінімально життєздатний продукт.

API (Application Programming Interface) – програмний інтерфейс застосунку.

REST (Representational State Transfer) – архітектурний стиль взаємодії веб-сервісів.

ORM (Object-Relational Mapping) – технологія об’єктно-реляційного відображення.

СУБД – система управління базами даних.

ІС – інформаційна система.

SPA (Single Page Application) – односторінковий веб-застосунок.

DRF (Django REST Framework) – фреймворк для створення REST API на основі Django.

UC (Use Case) – варіант використання системи.

FR (Functional Requirement) – функціональна вимога.

NFR (Non-Functional Requirement) – нефункціональна вимога.

ACID (Atomicity, Consistency, Isolation, Durability) – властивості транзакцій баз даних (атомарність, узгодженість, ізольованість, довговічність).

KPI (Key Performance Indicator) – ключовий показник ефективності.

ВСТУП

Сучасний розвиток інформаційних технологій суттєво трансформує підходи до управління спортивними закладами. Більшість спортивних клубів досі послуговуються застарілими методами обліку – паперовими журналами, телефонними записниками або несистематизованими електронними таблицями. Такий підхід не задовольняє потреб ні адміністрації, ні клієнтів: унеможливлює оперативний доступ до даних, ускладнює складання розкладу тренувань та призводить до помилок в обліку абонементів [1].

Використання веборієнтованих інформаційних систем дає змогу автоматизувати основні бізнес-процеси спортивних клубів, зокрема ведення бази клієнтів, формування та коригування розкладу занять, облік абонементів, а також забезпечення оперативної взаємодії між тренерами, адміністрацією і відвідувачами. Це сприяє підвищенню продуктивності роботи персоналу, оптимізації витрат і покращенню рівня сервісу для клієнтів.

Проблематику цифровізації спортивної галузі та управління спортивними закладами досліджували вітчизняні та зарубіжні вчені. Зокрема, питання нагальної важливості цифровізації закладів фізкультури і спорту України розглянуто у працях І. О. Гущи та Н. В. Крупко [2]. Проблематику управління спортивними клубами та їх роль у спортивній системі вивчали В. В. Гусаров і А. О. Стасюк [4], а порівняльний аналіз європейської та американської моделей клубних систем здійснили В. Гусаров і Т. Кропивницька [6]. Зарубіжний досвід організації спортивно-оздоровчих закладів узагальнено у монографії M. Wulf (Human Kinetics) [5]. Застосування мобільних і онлайн-технологій у сфері фізичної культури розглядали Т. Мошенська, Н. Долгополова та М. Сорочинська [14], а також Н. В. Чухланцева, Л. В. Шуба і В. В. Шуба [15]. Теоретичні засади проєктування інформаційних систем викладено у працях В. В. Литвина, А. І. Лозинського, Р. Є. Романюка [19], а питання баз даних – у роботі І. В. Тарасова та В. Г. Кузнецова [20]. Веб-технології та веб-програмування розглянуто у посібнику О. І. Савельєва [22]. Разом із тим комплексне

дослідження, що охоплює проєктування інформаційної системи управління спортивним клубом із використанням сучасних інформаційних технологій та веб-орієнтованих підходів, залишається актуальним.

Аналіз наявних веб-ресурсів спортивних клубів засвідчив, що більшість існуючих рішень або мають обмежений функціонал, або не забезпечують повної автоматизації управлінських процесів спортивного закладу [8–13].

Об'єктом дослідження є процес управління діяльністю спортивного клубу та взаємодія між його учасниками (адміністраторами, тренерами, клієнтами) засобами веб-технологій.

Предметом дослідження є методи та засоби проєктування і розробки інформаційної системи управління спортивним клубом з використанням веб-технологій.

Мета роботи – спроєктувати та розробити інформаційну систему управління спортивним клубом, що забезпечує автоматизацію процесів реєстрації клієнтів, управління абонементами, складання розкладу тренувань і комунікації між учасниками клубу на основі сучасних веб-технологій.

Для досягнення мети визначено такі завдання:

– проаналізувати предметну область діяльності спортивного клубу та існуючі програмні рішення;

– сформувати вимоги до інформаційної системи та описати бізнес-процеси клубу;

– розробити концептуальну та логічну моделі системи, спроєктувати базу даних;

– обрати та обґрунтувати технологічний стек для реалізації системи;

– реалізувати серверну та клієнтську частини системи з інтеграцією бази даних;

– провести тестування системи та оцінити якість розробленого рішення.

Методи дослідження: системний аналіз (для вивчення предметної області та формування вимог), метод порівняльного аналізу (для огляду

аналогів), UML-моделювання (для проєктування архітектури та бізнес-процесів), об'єктно-орієнтоване проєктування, а також методи тестування програмного забезпечення.

Наукова новизна роботи полягає у комплексному підході до проєктування інформаційної системи управління спортивним клубом, що охоплює повний цикл – від аналізу бізнес-процесів до програмної реалізації та тестування – з урахуванням специфіки вітчизняних спортивних закладів.

Практична значущість результатів полягає у тому, що розроблена інформаційна система може бути впроваджена у реальному спортивному клубі для автоматизації його діяльності, а також використана як основа для подальшого розвитку функціоналу або адаптації до потреб інших спортивних закладів.

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі здійснено аналіз предметної області та сформульовано вимоги до системи. У другому розділі розроблено інформаційне та математичне забезпечення. У третьому розділі описано програмну реалізацію та тестування системи.

РОЗДІЛ 1
АНАЛІЗ ДІЯЛЬНОСТІ СПОРТИВНОГО КЛУБУ ТА ОБҐРУНТУВАННЯ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ З ВИКОРИСТАННЯМ WEB-ТЕХНОЛОГІЙ

1.1 Характеристика діяльності спортивного клубу

Спортивний клуб є базовою організаційною одиницею спортивної системи України. Відповідно до Закону України «Про фізичну культуру і спорт», спортивний клуб – це заклад фізичної культури і спорту, що здійснює фізкультурно-оздоровчу та (або) спортивну діяльність і надає відповідні послуги населенню. Клуби діють на підставі статуту, реєструються у встановленому законом порядку і можуть засновуватися як фізичними, так і юридичними особами [3]. Органи державної влади та місцевого самоврядування сприяють їхній діяльності через організаційну та методичну підтримку, а фінансування здійснюється за рахунок коштів власника або інших незаборонених джерел.

Залежно від форми власності та мети діяльності, спортивні клуби функціонують за двома принципово різними бізнес-моделями. Комерційні клуби орієнтовані на отримання прибутку: їхні основні джерела доходу – абонементи, персональні тренування, оренда залів і додаткові послуги (SPA, фітнес-бар, дитячі програми). Такі організації зацікавлені в максимальному завантаженні потужностей та утриманні клієнтів. Некомерційні (громадські) клуби та клуби при навчальних закладах натомість орієнтовані на розвиток спорту як соціального явища: їхнє фінансування складається з членських внесків, державних субсидій і грантів, а основний пріоритет – охоплення аудиторії, зокрема молоді й людей з особливими потребами. Ця відмінність безпосередньо впливає на вимоги до інформаційної системи: комерційному клубу критично важливий модуль продажу абонементів і аналітики доходів, тоді як некомерційному – облік учасників і планування заходів [4, 5].

Міжнародний дослідник індустрії фітнесу M. Wulf систематизував спортивно-оздоровчі заклади за організаційно-правовою формою та цільовою аудиторією, виділивши вісім основних типів: комерційні фітнес-центри й атлетичні клуби; приватні клуби закритого типу; оздоровчі комплекси на базі готелів і бізнес-центрів; клуби при громадських і професійних об'єднаннях; корпоративні фітнес-центри; кардіологічні реабілітаційні центри; спортивно-медичні комплекси; а також аматорські та професійні секції за видами спорту [5]. Ця класифікація є актуальною і для українського ринку, де переважають комерційні фітнес-центри та клуби при навчальних закладах.

В. В. Гусаров і А. О. Стасюк розглядають клуб як один із базових структурних елементів спортивної системи, що виконує кілька взаємопов'язаних функцій [4]. По-перше, організаційна функція передбачає планування тренувального процесу, проведення змагань і координацію спортсменів різного рівня підготовки. По-друге, ресурсне забезпечення охоплює надання доступу до спортивної інфраструктури – тренажерних залів, майданчиків, медичних кабінетів – і залучення кваліфікованих тренерів та фахівців. По-третє, соціальна функція полягає у формуванні мотиваційного середовища, популяризації активного способу життя та забезпеченні соціальної інтеграції різних категорій населення, зокрема осіб з інвалідністю. Нарешті, економічна функція передбачає залучення фінансування через абонементи, спонсорство, рекламу та створення робочих місць.

Порівняльний аналіз В. Гусарова і Т. Кропивницької засвідчує, що в країнах із розвиненою клубною системою – Німеччині, Швеції, Фінляндії та Норвегії – спортивні клуби спираються на принципи масовості й самоорганізації за активної підтримки держави [6]. Зокрема, у Швеції діє понад 45 тисяч клубів, а у Фінляндії акцент робиться не лише на фізичній підготовці, а й на соціальній інтеграції [7]. Американська модель, навпаки, більш комерціалізована: клуби функціонують переважно як прибуткові підприємства і конкурують за клієнтів на відкритому ринку. Саме комерційний

підхід набуває поширення в Україні й зумовлює потребу в автоматизації управлінських процесів.

Аналіз операційної діяльності комерційного спортивного клубу дозволяє виокремити кілька взаємопов'язаних процесів, автоматизація яких є метою цієї роботи [4, 19]:

– реєстрація клієнта – внесення персональних даних, формування профілю, вибір та оплата абонементу;

– управління абонементами – активація, заморожування, продовження, автоматичне сповіщення про закінчення терміну дії;

– планування розкладу – формування тижневого розкладу тренувань, призначення тренерів до занять, управління залами;

– запис на тренування – онлайн-бронювання місця клієнтом, підтвердження запису, скасування та управління списком очікування;

– облік відвідувань – фіксація факту відвідування тренування, автоматичне списання з абонементу;

– управління персоналом – облік тренерів, їхній розклад, прив'язка до напрямків занять;

– звітність і аналітика – формування звітів про відвідуваність, доходи, завантаженість залів для прийняття управлінських рішень.

Дослідження І. О. Гущи та Н. В. Крупко підтверджують, що недостатня цифровізація спортивних закладів України безпосередньо впливає на якість їхньої роботи [2]. У практичній діяльності відсутність або недосконала реалізація інформаційної системи призводить до низки характерних проблем (таблиця 1.1.), зокрема: це ускладнює облік клієнтів, підвищує ймовірність помилок у розкладі та зменшує оперативність прийняття управлінських рішень.

Отже, спортивний клуб можна розглядати як складну організаційну систему, що включає різні бізнес-процеси: роботу з клієнтами, управління персоналом, планування діяльності та аналітичну обробку даних.

Таблиця 1.1

Проблеми управління спортивним клубом без автоматизації

Сфера	Проблема
Облік абонементів	Помилки при ручному обліку, втрата даних, неможливість швидко перевірити статус клієнта
Запис на тренування	Черги та дублювання записів, незручність для клієнта (лише телефоном або особисто)
Розклад	Конфлікти в розкладі тренерів і залів, ручне оновлення при змінах
Комунікація	Затримки в сповіщенні клієнтів про зміни, скасування занять
Аналітика	Відсутність оперативних даних про завантаженість і доходи для управлінських рішень

Джерело: складено автором на основі [2, 4, 19].

Таким чином, оскільки сучасні українські клуби здебільшого орієнтовані на комерційний результат, а користувачі висувають усе вищі вимоги до зручності цифрових сервісів, виникає потреба у створенні спеціалізованої інформаційної системи. Така система має забезпечити автоматизацію основних процесів і підвищити ефективність управління закладом.

1.2 Аналіз існуючих інформаційних систем та веб-рішень у діяльності спортивних клубів

Для визначення вимог до розроблюваної інформаційної системи необхідно проаналізувати наявні рішення – як вітчизняні веб-ресурси спортивних клубів, так і спеціалізовані закордонні платформи управління фітнес-закладами. Такий аналіз дозволяє виявити сильні та слабкі сторони існуючих підходів і обґрунтувати функціональний склад власної розробки [19].

Дослідження українського сегменту ринку охоплює п'ять ресурсів різного масштабу та функціонального наповнення: мережу Total Fitness, клуб

V&V NEW POWER, мережу Sport Life, клуб Sport Studio та MUSCLE GYM. Кожен із них реалізує власну модель цифрової взаємодії з клієнтами.

Total Fitness (totalfitness.com.ua) – одна з найбільших мереж фітнес-клубів України – демонструє найвищий рівень цифрової зрілості серед проаналізованих вітчизняних ресурсів. Платформа інтегрує мобільний застосунок, QR-вхід до клубу, онлайн-управління абонементом і запис на групові заняття. Гнучка система підписок передбачає можливість «заморожування» абонементу. Разом із тим ресурс не відображає реального навантаження в пікові години, а залежність від мобільного додатку ускладнює доступ для частини аудиторії [8].

V&V NEW POWER (vvnewpower.wixsite.com) – локальний клуб на платформі Wix – обмежується базовими інформаційними функціями: перелік послуг, контакти, розклад. Відсутні онлайн-запис, відгуки клієнтів і мультимедійний контент. SEO-можливості платформи обмежені, мобільна адаптація потребує вдосконалення [9].

Sport Life (sportlife.ua) – розгалужена мережа з широким спектром послуг (тренажерні зали, басейни, SPA, дитячий фітнес). Сайт реалізує онлайн-продаж абонементів і акційні пропозиції, однак умови акцій є складними для сприйняття, а якість обслуговування варіюється між локаціями мережі [11].

Sport Studio (sportstudio.com.ua) – клуб преміум-сегменту в Києві – пропонує розгорнутий опис послуг, відгуки клієнтів і публічну оферту, що підвищує довіру. Водночас контент переважно рекламний, відсутні об'єктивні дані (статистика, фото залів), а географічне охоплення обмежене [12].

MUSCLE GYM (musclegym.info) – комерційний ресурс із гнучкою лінійкою абонементів та онлайн-оплатою. Ресурс пропонує індивідуальні програми, план харчування, додаткові сервіси. Недоліки: відсутність деталізованого опису обладнання та незалежних відгуків, маркетинговий стиль викладу [13].

Зведений порівняльний аналіз функціональних можливостей вітчизняних ресурсів наведено у таблиці 1.2.

Таким чином, проведений аналіз вітчизняних веб-ресурсів спортивних клубів засвідчив, що більшість із них виконують переважно інформаційно-рекламну функцію та лише частково забезпечують автоматизацію бізнес-процесів. Найчастіше реалізовано базові можливості – перегляд розкладу, онлайн-оплату або запис на тренування, однак відсутні засоби аналітики, управління відвідуваністю, централізованої роботи з абонементами та повноцінної взаємодії між клієнтами, тренерами й адміністрацією.

Таблиця 1.2

Порівняльний аналіз вітчизняних веб-ресурсів спортивних клубів

Функціональна можливість	Total Fitness	V&V NEW POWER	Sport Life	Sport Studio	MUSCLE GYM
Онлайн-запис на тренування	✓	✗	✓	✗	✓
Управління абонементом онлайн	✓	✗	✓	✗	✓
Мобільний застосунок	✓	✗	✗	✗	✗
Розклад тренувань	✓	~	✓	✓	✓
Онлайн-оплата	✓	✗	✓	✗	✓
Відгуки клієнтів	✗	✗	✗	✓	✗
Профіль тренера	✓	~	✓	✓	✓
Аналітика/звітність	✗	✗	✗	✗	✗
Адаптивний дизайн	✓	~	✓	✓	✓
Багатомовність	✗	✗	✗	✗	✗

Примітка: "✓" – функція наявна, "✗" – відсутня, "~" – реалізована частково. Складено автором за [8–13].

Аналіз також показав, що функціональні можливості ресурсів суттєво відрізняються залежно від масштабу клубу, а більшість локальних закладів використовують спрощені веб-рішення з обмеженим рівнем автоматизації. Це обґрунтовує необхідність розробки інформаційної системи управління спортивним клубом, яка забезпечить комплексну автоматизацію основних

процесів: управління клієнтами та абонементами, формування розкладу, онлайн-запис, облік відвідуваності та аналітичну підтримку діяльності клубу.

1.3. Аналіз бізнес-процесів спортивного клубу

Ефективне проєктування інформаційної системи управління спортивним клубом неможливе без детального аналізу бізнес-процесів, що відбуваються в організації. Формалізація цих процесів дозволяє чітко визначити межі автоматизації, виявити вузькі місця та обґрунтувати функціональний склад майбутньої системи [4, 19]. Для формалізації бізнес-процесів спортивного клубу можуть бути використані стандартні нотації BPMN (Business Process Model and Notation) та UML (Unified Modeling Language).

На підставі аналізу операційної діяльності спортивного клубу виокремлено шість ключових бізнес-процесів:

– реєстрація та запис на тренування – онлайн-взаємодія клієнта із системою для бронювання місця на занятті;

– управління абонементами – придбання, активація, призупинення та продовження абонементів різних типів;

– формування та коригування розкладу занять – планування тренувань з урахуванням доступності тренерів і залів;

– облік відвідуваності та активності клієнтів – фіксація участі у заняттях, списання з абонементу, ведення статистики;

– фінансовий облік та оплата послуг – реєстрація платежів, інтеграція з платіжними системами, формування рахунків;

– аналітика та формування звітності – агрегація та обробка даних для підтримки управлінських рішень.

Нижче наведено детальний опис чотирьох ключових процесів, що мають найбільший вплив на якість обслуговування клієнтів і підлягають першочерговій автоматизації.

Процес 1. Реєстрація та запис на тренування

Актори: клієнт, система (веб-додаток), адміністратор.

Вхідні дані: автентифікаційні дані клієнта, перелік доступних тренувань, кількість вільних місць.

Процес розпочинається з авторизації клієнта у системі. Після успішної автентифікації клієнт переглядає актуальний розклад і обирає бажане заняття. Система перевіряє наявність вільних місць у відповідному часовому слоті. За наявності місць клієнт підтверджує запис – система зберігає бронювання в базі даних і надсилає підтвердження на електронну пошту або через push-сповіщення. У разі відсутності місць клієнту пропонується внесення до списку очікування або вибір альтернативного заняття.

Альтернативний сценарій: якщо клієнт не з'явився на заняття без попереднього скасування, система фіксує пропуск і може застосувати встановлені клубом санкції (наприклад, тимчасове обмеження записів).

Результат: бронювання збережено в базі даних, місце підтверджено, клієнт поінформований (таблиця 1.3).

Таблиця 1.3

Характеристика процесу «Реєстрація та запис на тренування»

Параметр	Опис
Ініціатор	Клієнт
Вхідні дані	Логін/пароль, дата, вид заняття
Дії системи	Перевірка місць → збереження → сповіщення
Альтернатива	Список очікування або вибір іншого слоту
Результат	Підтверджений запис у базі даних

Примітка. Джерело: складено автором на основі [4, 19].

Процес 2. Управління абонементами

Актори: клієнт, адміністратор, платіжна система, система.

Вхідні дані: тип абонементу (разовий, місячний, річний, заморожений), дата активації, сума оплати.

Клієнт або адміністратор обирає тип абонементу відповідно до потреб. Здійснюється оплата через інтегровану платіжну систему. Після підтвердження транзакції система автоматично активує абонемент і встановлює дату завершення. Протягом усього терміну дії система відстежує залишок відвідувань або часу, автоматично змінюючи статус: «активний» → «призупинений» (за запитом клієнта) → «завершений» (після спливання терміну). За 7 та 3 дні до завершення система надсилає клієнту нагадування про продовження.

Альтернативний сценарій: у разі невдалої транзакції абонемент не активується; клієнт отримує повідомлення про помилку та інструкції для повторної спроби.

Результат: абонемент активовано, дані збережено, клієнт має доступ до послуг відповідно до умов.

Процес 3. Формування та коригування розкладу занять

Актори: адміністратор, тренер, система.

Вхідні дані: графік доступності тренерів, список залів та їх місткість, попит клієнтів за напрямками.

Адміністратор збирає інформацію про доступність тренерів і залів на плановий період. На основі цих даних формуються часові слоти – система перевіряє відсутність конфліктів (один тренер не може вести два заняття одночасно; один зал не може бути зайнятий двома групами). Після проходження перевірки розклад публікується у системі та стає доступним для онлайн-запису клієнтів. При необхідності коригування (хвороба тренера, технічна несправність залу) адміністратор вносить зміни – система автоматично сповіщає зареєстрованих клієнтів про зміни або скасування.

Альтернативний сценарій: при виявленні конфлікту в розкладі система блокує збереження та інформує адміністратора про причину.

Результат: актуальний розклад опублікований у системі, клієнти поінформовані про зміни.

Процес 4. Аналітика та формування звітності

Актори: адміністратор, керівник клубу, система.

Вхідні дані: дані про відвідуваність, статуси абонементів, фінансові транзакції, завантаженість тренерів і залів.

Аналітичний модуль агрегує дані з усіх підсистем у реальному часі. Адміністратор або керівник обирає тип звіту та часовий діапазон. Система виконує вибірку, фільтрацію та обчислення ключових показників: кількість відвідувань за період, коефіцієнт наповненості груп, рейтинг популярності напрямків, обсяг доходів за типами абонементів, навантаження на тренерів. Результати відображаються у табличній та графічній формі (діаграми, гістограми) і можуть бути експортовані у форматі PDF або Excel.

Результат: сформований звіт надає керівництву актуальну інформацію для прийняття управлінських рішень – коригування цінової політики, оптимізації розкладу, оцінки ефективності тренерів.

Зведений порівняльний аналіз усіх шести бізнес-процесів за ступенем поточної автоматизації та пріоритетністю впровадження наведено у таблиці 1.4.

Таблиця 1.4

Зведений аналіз бізнес-процесів спортивного клубу

Бізнес-процес	Поточний стан	Ступінь автоматизації	Пріоритет
Реєстрація та запис на тренування	Ручний (телефон / особисто)	Низький	Високий
Управління абонементами	Частково (Excel, паперово)	Низький	Високий
Формування розкладу	Ручний (Google Sheets)	Низький	Високий
Облік відвідуваності	Ручний (журнал)	Відсутній	Середній
Фінансовий облік	Частково (бухг. ПЗ)	Середній	Середній
Аналітика та звітність	Відсутня	Відсутній	Високий

Примітка. Джерело: складено автором на основі [2, 4, 19].

Проведений аналіз бізнес-процесів спортивного клубу засвідчує, що більшість ключових операцій виконуються вручну або за допомогою несистематизованих інструментів – паперових журналів, електронних

таблиць, телефонних записів. Це призводить до помилок, втрат даних і надмірного навантаження на персонал. Усі шість виокремлених процесів мають чітку структуру, визначених акторів і формалізовані результати, що робить їх придатними для подальшої автоматизації в межах інформаційної системи спортивного клубу. Отримані результати є основою для формування функціональних і нефункціональних вимог до інформаційної системи, що розглядаються у підрозділі 1.4 [4, 19, 22].

1.4 Формування вимог до інформаційної системи управління спортивним клубом

Формування вимог є визначальним етапом у процесі розробки інформаційної системи: саме від повноти та точності специфікації залежить відповідність кінцевого продукту потребам замовника і користувачів. Відповідно до стандарту IEEE 830, вимоги до програмного забезпечення поділяють на функціональні та нефункціональні [16]. Функціональні вимоги описують, які операції виконує система; нефункціональні – яким чином вона це робить, встановлюючи кількісні критерії якості [17].

Для збору вимог використано комплексний підхід: аналіз бізнес-процесів клубу (підрозділ 1.3), порівняльний аналіз аналогів (підрозділ 1.2) та метод User Stories в рамках Agile-підходу. Зацікавленими сторонами виступають клієнти, тренери, адміністратори та розробники. Результатом є специфікація вимог до програмного забезпечення (Software Requirements Specification, SRS) [19].

Для підвищення якості специфікації вимоги сформульовано за принципом SMART: Specific (конкретні), Measurable (вимірювані), Achievable (досяжні), Relevant (релевантні цілям системи), Time-bound (обмежені в часі або перевірювані в реальному часі). Наприклад, замість розмитого «система має бути швидкою» – «час відгуку системи при стандартному навантаженні

(до 200 одночасних запитів) не перевищує 2 секунди». Цей підхід застосовано до всіх нефункціональних вимог, наведених нижче [17].

У системі визначено три основні ролі з різними правами доступу:

– клієнт – реєстрація, перегляд розкладу, запис на тренування, управління абонементом, перегляд власної статистики;

– тренер – перегляд власного розкладу, список записаних клієнтів, відмітка відвідувань;

– адміністратор – повний доступ: управління розкладом, клієнтською базою, абонементами, фінансами та звітністю.

У рамках Agile-підходу функціональні вимоги представлено у форматі User Stories – коротких сценаріїв, що описують очікувану поведінку системи з точки зору конкретної ролі [19].

Клієнт:

– як клієнт, я хочу мати можливість зареєструватися в системі, щоб отримати доступ до функціональних можливостей спортивного клубу;

– як клієнт, я хочу переглядати актуальний розклад тренувань, щоб обирати найбільш зручний для себе час занять;

– як клієнт, я хочу записуватися на тренування через вебсайт або мобільний застосунок, щоб уникнути необхідності телефонного звернення до адміністрації;

– як клієнт, я хочу скасовувати попередній запис на тренування, щоб звільнити місце для інших відвідувачів у разі зміни планів;

– як клієнт, я хочу купувати абонементи онлайн, щоб швидко отримувати доступ до тренувань без особистого відвідування клубу;

– як клієнт, я хочу переглядати історію своїх відвідувань, щоб відстежувати рівень власної активності та регулярність занять;

– як клієнт, я хочу отримувати автоматичні нагадування про заплановані тренування, щоб не пропускати заняття.

Тренер:

– як тренер, я хочу бачити свій розклад на тиждень, щоб планувати робочий час;

– як тренер, я хочу переглядати список записаних клієнтів, щоб підготуватися до заняття;

– як тренер, я хочу відмічати відвідування після заняття, щоб система автоматично списувала з абонементу.

Адміністратор:

– як адміністратор, я хочу додавати та редагувати тренування в розкладі, щоб підтримувати його актуальність;

– як адміністратор, я хочу керувати абонементами клієнтів, щоб контролювати доступ до послуг;

– як адміністратор, я хочу переглядати фінансові звіти, щоб оцінювати ефективність роботи клубу;

– як адміністратор, я хочу керувати обліковими записами користувачів, щоб підтримувати актуальність бази даних.

На підставі аналізу бізнес-процесів (підрозділ 1.3) та User Stories сформовано перелік функціональних вимог (таблиця 1.5). Пріоритизацію здійснено за методом MoSCoW: Must (M) – обов'язкова, Should (S) – бажана, Could (C) – додаткова.

Таблиця 1.5

Функціональні вимоги до інформаційної системи

ID	Назва	Опис / критерій прийнятності	Роль	Пріор.
1	2	3	4	5
FR-01	Реєстрація	Система створює обліковий запис після валідації email та пароля (мін. 8 символів)	Клієнт	Must
FR-02	Авторизація	Вхід за email/паролем; сесія зберігається 24 год	Всі ролі	Must
FR-03	Перегляд розкладу	Відображення тренувань за датою, напрямком, тренером; фільтрація	Клієнт, Тренер	Must
FR-04	Запис на тренування	Бронювання місця за наявності активного абонементу та вільних місць	Клієнт	Must

Продовження табл.1.5

1	2	3	4	5
FR-05	Управління абонементом	Придбання, активація, заморожування, продовження; автосповіщення за 7 днів до закінчення	Клієнт, Адмін	Must
FR-06	Онлайн-оплата	Інтеграція з платіжним шлюзом; підтвердження транзакції протягом 5 сек.	Клієнт	Must
FR-07	Управління розкладом	Додавання, редагування, видалення занять; перевірка конфліктів тренерів і залів	Адмін	Must
FR-08	Відмітка відвідувань	Фіксація участі у занятті; автоматичне списання з абонементу	Тренер, Адмін	Must
FR-09	Звітність і аналітика	Генерація звітів: відвідуваність, доходи, завантаженість; експорт PDF/Excel	Адмін	Must
FR-10	Скасування запису	Клієнт може скасувати запис не пізніше ніж за 2 год. до початку заняття	Клієнт	Should
FR-11	Сповіщення	Email-нагадування за 24 год. та 1 год. до тренування	Клієнт	Should
FR-12	Управління користувачами	Редагування профілів, зміна ролей, деактивація облікових записів	Адмін	Should
FR-13	Список очікування	Автозапис клієнта при звільненні місця	Клієнт	Should
FR-14	Профіль тренера	Публічна сторінка тренера з описом, фото, напрямками	Тренер, Адмін	Could
FR-15	Відгуки клієнтів	Оцінка тренування після відвідування (1–5 зірок + коментар)	Клієнт	Could

Примітка: Must – обов'язкова, Should – бажана, Could – додаткова. Складено автором.

Use Case специфікації деталізують взаємодію акторів із системою для п'яти ключових сценаріїв. Повний перелік Use Case діаграм наводиться у розділі 2.

Таблиця 1.6

Специфікації варіантів використання (Use Cases)

Use Case	Актор	Передумови	Основний сценарій	Альтернативний сценарій
UC-01 Реєстрація	Клієнт	Немає облікового запису	Заповнити форму → валідація → створення акаунту → підтвердження email	Некоректні дані → повідомлення про помилку
UC-02 Запис на тренування	Клієнт	Авторизований; є активний абонемент	Обрати заняття → перевірка місць → підтвердити → сповіщення	Немає місць → пропозиція списку очікування
UC-03 Купівля абонементу	Клієнт	Авторизований	Обрати тип → оплата → активація абонементу	Помилка платежу → транзакція скасована, абонемент не активовано
UC-04 Управління розкладом	Адміністратор	Авторизований як адмін	Відкрити панель → додати/редагувати → перевірка конфліктів → зберегти	Конфлікт → система блокує збереження та інформує адміна
UC-05 Формування звітності	Адміністратор	Накопичені дані в системі	Обрати тип звіту → діапазон дат → генерація → відображення → експорт	Недостатньо даних → система інформує адміна

Примітка. Джерело: складено автором.

Нефункціональні вимоги визначають якісні характеристики системи та встановлюють вимірювані критерії її ефективності [17, 18].

Таблиця 1.7

Нефункціональні вимоги до інформаційної системи

ID	Категорія	Вимога та SMART-критерій	Метрика	Пріор.
1	2	3	4	5
NFR-01	Продуктивність	Час відгуку системи при навантаженні до 200 одночасних запитів не перевищує 2 сек.	≤ 2 с	Must
NFR-02	Продуктивність	Система підтримує не менше 500 зареєстрованих і 100 одночасно активних користувачів	≥ 500/100	Must

Продовження табл. 1.7

1	2	3	4	5
NFR-03	Безпека	Автентифікація реалізована через JWT-токени; сесія завершується через 24 год. Неактивності	JWT, 24 год.	Must
NFR-04	Безпека	Усі дані передаються через HTTPS (TLS 1.2+); паролі зберігаються у вигляді bcrypt-хешів	HTTPS, bcrypt	Must
NFR-05	Безпека	Розмежування прав за трьома ролями (клієнт / тренер / адмін) – role-based access control	RBAC	Must
NFR-06	Надійність	Доступність системи – не менше 99% на місяць (≤ 7 год. простою на рік)	≥ 99% uptime	Must
NFR-07	Надійність	Автоматичне резервне копіювання бази даних щодня; відновлення протягом 1 год.	Щодня, ≤ 1 год.	Must
NFR-08	Зручність	Адаптивний дизайн: коректне відображення на пристроях від 320px до 1920px (mobile-first)	320–1920px	Must
NFR-09	Зручність	Основні операції (запис, оплата) виконуються не більше ніж за 3 кроки (кліки)	≤ 3 кроки	Should
NFR-10	Масштабованість	Архітектура дозволяє підключення нових модулів без зміни ядра системи	Модульність	Should
NFR-11	Підтримуваність	Оновлення системи виконується без зупинки роботи (rolling deployment)	Zero downtime	Should
NFR-12	Сумісність	Коректна робота у браузерах Chrome, Firefox, Safari, Edge (останні 2 версії)	4 браузери	Should

Примітка: Must – обов'язкова, Should – бажана. Складено автором на основі [16–18].

Сформована специфікація вимог охоплює 15 функціональних і 12 нефункціональних вимог, 5 варіантів використання та 15 User Stories для трьох

ролей. Застосування принципу SMART забезпечує вимірюваність і перевірюваність кожної нефункціональної вимоги. Пріоритизація за методом MoSCoW визначає обсяг мінімально життєздатного продукту (MVP) і є основою для планування розробки у розділі 3. Отримана специфікація слугує відправною точкою для проєктування архітектури системи та бази даних у розділі 2 [16, 17, 19].

На підставі аналізу предметної області (підрозділ 1.1), огляду існуючих рішень (підрозділ 1.2), формалізації бізнес-процесів (підрозділ 1.3) та специфікації вимог (підрозділ 1.4) сформульовано постановку задачі розробки інформаційної системи управління спортивним клубом. Постановка задачі визначає об'єкт і предмет розробки, мету, перелік задач, обмеження та очікувані результати і є технічною основою для проєктування системи у розділі 2.

Об'єктом розробки є інформаційна система управління спортивним клубом, що функціонує у вигляді веб-застосунку та забезпечує автоматизацію операційної діяльності закладу: обслуговування клієнтів, управління персоналом, планування розкладу і формування аналітичної звітності.

Предметом розробки є архітектура, алгоритми та програмна реалізація модулів інформаційної системи: підсистеми управління користувачами, абонементами, розкладом тренувань, обліку відвідуваності та аналітики – на основі сучасних веб-технологій.

Метою розробки є створення функціонального веб-застосунку для автоматизації управління спортивним клубом, що забезпечує онлайн-запис клієнтів на тренування, управління абонементами, формування розкладу занять без конфліктів ресурсів, облік відвідуваності та генерацію аналітичних звітів для адміністрації – і тим самим усуває виявлені у підрозділах 1.1–1.3 проблеми ручного управління.

Для досягнення поставленої мети необхідно вирішити такі задачі:

– аналіз і моделювання предметної області: побудувати UML-діаграми бізнес-процесів (use case, activity, sequence), що формалізують взаємодію акторів із системою;

– проєктування бази даних: розробити концептуальну (ER-діаграму) та реляційну схему бази даних, виконати нормалізацію до третьої нормальної форми (3НФ), визначити індекси та обмеження цілісності;

– вибір і обґрунтування технологічного стеку: здійснити порівняльний аналіз фреймворків і СУБД, обрати інструменти для реалізації серверної та клієнтської частин з урахуванням вимог продуктивності, безпеки та масштабованості;

– реалізація серверної частини: розробити REST API з підтримкою автентифікації (JWT), авторизацією за ролями (RBAC), бізнес-логікою модулів та інтеграцією з базою даних;

– реалізація клієнтської частини: розробити адаптивний веб-інтерфейс для трьох ролей (клієнт, тренер, адміністратор) з реалізацією всіх функціональних вимог FR-01–FR-15;

– реалізація ключових алгоритмів: розробити алгоритм перевірки конфліктів при формуванні розкладу та алгоритм автоматичного списання відвідувань з абонементу;

– тестування системи: провести функціональне тестування відповідно до визначених Use Cases (UC-01–UC-05) та перевірити виконання нефункціональних вимог NFR-01–NFR-12;

– оцінка якості: проаналізувати результати тестування, виміряти ключові показники (час відгуку, доступність, коректність RBAC) і зробити висновки щодо відповідності системи сформованим вимогам.

При розробці системи враховано такі обмеження:

– платформа: система реалізується як веб-застосунок і не передбачає розробки окремого нативного мобільного додатку (iOS/Android); адаптивний дизайн забезпечує коректну роботу на мобільних пристроях через браузер;

– масштаб: система розрахована на один спортивний клуб (одна локація); підтримка мережі клубів є перспективою подальшого розвитку;

– платіжна система: інтеграція з платіжним шлюзом реалізується на рівні прототипу або тестового середовища; повна інтеграція з банківськими системами виходить за межі поточної розробки;

– мова інтерфейсу: інтерфейс реалізується українською мовою; багатомовність є додатковою функцією (Could) і до MVP не входить;

– середовище: для роботи системи потрібне стабільне інтернет-з'єднання і сучасний браузер (Chrome, Firefox, Safari, Edge – останні 2 версії);

– бухгалтерська інтеграція: система не інтегрується із зовнішнім бухгалтерським програмним забезпеченням; фінансова звітність формується в межах власного аналітичного модуля.

Сформульована постановка задачі визначає межі та напрямки подальшої розробки програмного продукту, а також слугує основою для прийняття проєктних рішень на етапах проєктування та реалізації системи. Очікуваний результат повинен забезпечити підвищення ефективності управління спортивним клубом, зменшення кількості ручних операцій та покращення якості обслуговування клієнтів.

Очікувані результати розробки розмістимо у вигляді таблиці наведеної нижче (таблиця 1.8).

Таким чином, було сформовано повну та структуровану специфікацію вимог до інформаційної системи управління спортивним клубом. На основі аналізу бізнес-процесів, існуючих рішень та підходу User Stories визначено функціональні та нефункціональні вимоги, описано ролі користувачів і сценарії їх взаємодії із системою.

Застосування принципів SMART забезпечило вимірюваність та перевірюваність нефункціональних вимог, а використання методу MoSCoW дозволило визначити пріоритетність реалізації функціональності та сформувати основу мінімально життєздатного продукту (MVP).

Таблиця 1.8

Очікувані результати розробки

Результат	Зміст / критерій готовності
Веб-застосунок	Функціональна система з реалізованими модулями: реєстрація, розклад, запис, абонементи, відвідуваність, аналітика
База даних	Реляційна БД у 3НФ з усіма таблицями, зв'язками та обмеженнями цілісності
REST API	Задокументовані ендпоінти для всіх функціональних модулів з підтримкою JWT та RBAC
Адаптивний інтерфейс	UI для трьох ролей, коректне відображення на пристроях від 320px до 1920px
Результати тестування	Протокол тестування UC-01–UC-05; підтвердження виконання NFR щодо часу відгуку та безпеки
Документація	Пояснювальна записка; UML-діаграми; ER-діаграма; схема архітектури

Примітка. Джерело: складено автором.

Розроблена специфікація вимог, а також постановка задачі визначають чіткі межі проєкту, очікувані результати та критерії успішності системи. Отримані результати є базою для подальшого проєктування архітектури програмного забезпечення, бази даних і реалізації функціональних модулів у наступних розділах роботи.

Висновки до розділу 1

У першому розділі проаналізовано обсяг та операційні характеристики сучасних спортивних клубів. Розглянуто основні напрямки діяльності спортивних закладів, структуру їхніх бізнес-процесів та роль інформаційних технологій у забезпеченні ефективного управління клієнтами, абонементами, розкладами тренувань та внутрішніми адміністративними процесами.

У дослідженні зроблено висновок, що цифровізація спортивного сектору є ключовим фактором у покращенні якості обслуговування клієнтів, оптимізації роботи персоналу та автоматизації щоденних операцій. Використання інформаційних систем зменшує ручну роботу, спрощує

реєстрацію відвідувань, полегшує оперативний доступ до даних та підвищує ефективність управління спортивними клубами.

Аналіз існуючих веб-ресурсів спортивних клубів показав, що більшість сучасних рішень реалізують лише базові функції для представлення інформації про послуги та пропонують обмежений рівень автоматизації. Як правило, пропонуються такі функції, як перегляд розкладу занять, онлайн-оплата та реєстрація на тренування, але відсутні комплексні аналітичні інструменти, централізоване управління підписками, відстеження відвідувань та складна взаємодія між клієнтами, тренерами та адміністрацією клубу.

В результаті аналізу було визначено основні функціональні та нефункціональні вимоги до інформаційної системи управління спортивним клубом. Система повинна забезпечувати реєстрацію клієнтів та тренерів, управління підписками, створення та редагування розкладу тренувань, онлайн-реєстрацію на заняття, відстеження відвідуваності, а також забезпечувати інструменти адміністративного контролю та аналітичну підтримку діяльності клубу.

Результати дослідження послужили основою для проектування архітектури системи, визначення її функціональної структури та вибору інструментів впровадження, які будуть розглянуті в наступних розділах цієї роботи.

РОЗДІЛ 2
ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ СПОРТИВНИМ КЛУБОМ

У розділі здійснено концептуальне та логічне моделювання інформаційної системи управління спортивним клубом засобами UML, спроєктовано реляційну базу даних, розроблено ключові алгоритми та обґрунтовано архітектурні рішення системи з урахуванням обраного технологічного стеку (React, Django, PostgreSQL).

2.1 Концептуальне та логічне моделювання системи управління спортивним клубом

Моделювання інформаційної системи виконано із застосуванням мови UML (Unified Modeling Language) – стандарту об'єктно-орієнтованого проєктування, що дозволяє формалізувати структуру та поведінку системи через набір графічних діаграм [19]. Для повного опису системи використано три типи діаграм: варіантів використання (Use Case), послідовності (Sequence) та активності (Activity).

Діаграма варіантів використання (Use Case Diagram) описує функціональні можливості системи з точки зору акторів – зовнішніх суб'єктів, що взаємодіють із системою. Визначено трьох основних акторів: Клієнт, Тренер та Адміністратор.

Актор Клієнт має доступ до таких варіантів використання: реєстрація та авторизація в системі (UC-01, UC-02); перегляд розкладу тренувань із фільтрацією за датою, напрямком і тренером (UC-03); запис на тренування за наявності активного абонементу та вільних місць (UC-04); скасування запису (UC-05); придбання та управління абонементом (UC-06); перегляд історії відвідувань і особистої статистики (UC-07); отримання сповіщень про майбутні тренування (UC-08).

Актор Тренер взаємодіє з системою через: перегляд власного розкладу (UC-09); перегляд списку записаних клієнтів на конкретне заняття (UC-10); відмітку відвідуваності клієнтів після заняття (UC-11).

Актор Адміністратор має розширені права і виконує: управління розкладом занять – додавання, редагування, видалення (UC-12); управління обліковими записами та ролями користувачів (UC-13); управління абонементами клієнтів (UC-14); перегляд та формування аналітичних звітів (UC-15); управління профілями тренерів і залами (UC-16).

Між варіантами використання UC-04 (Запис на тренування) і UC-06 (Управління абонементом) існує відношення «include»: система перевіряє наявність активного абонементу як обов'язкову передумову запису. UC-08 (Сповіщення) пов'язане з UC-04 відношенням «extend»: сповіщення надсилається автоматично після успішного запису, але не є обов'язковою частиною основного сценарію.

Діаграми послідовності (Sequence Diagrams) відображають хронологічний порядок взаємодії об'єктів системи при виконанні конкретного сценарію. Розроблено діаграми для двох ключових процесів.

Сценарій «Запис клієнта на тренування». Взаємодія розгортається між чотирма учасниками: Клієнт (браузер), Клієнтський веб-інтерфейс, Серверний API, PostgreSQL. Послідовність повідомлень:

– клієнт → Frontend: вибір тренування у розкладі (подія onClick);
– Frontend → API: GET /api/schedule/{id}/ – запит деталей заняття;
– API → DB: SELECT * FROM schedule WHERE id = {id};
– DB → API: повернення запису із полем available_seats;
– API → Frontend: JSON з деталями заняття та кількістю вільних місць;
– клієнт → Frontend: підтвердження запису (натискання кнопки);
– Frontend → API: POST /api/bookings/ { schedule_id, client_id };
– API → DB: перевірка активного абонементу клієнта – SELECT * FROM subscriptions WHERE client_id = {id} AND status = 'active';

– API → DB (за умови успіху): INSERT INTO bookings; UPDATE schedule SET available_seats = available_seats – 1;

– API → Frontend: HTTP 201 Created { booking_id, confirmation };

– Frontend → Клієнт: відображення підтвердження та оновлення розкладу.

Альтернативний сценарій: якщо available_seats = 0 або абонемент неактивний – API повертає HTTP 400 з відповідним повідомленням; Frontend відображає сповіщення про помилку та пропонує список очікування або вибір іншого заняття.

Сценарій «Відмітка відвідуваності тренером». Учасники: Тренер, Frontend, API, DB. Тренер відкриває список записаних клієнтів на заняття (GET /api/schedule/{id}/attendees/), відмічає присутніх (PUT /api/bookings/{id}/ { attended: true }). API оновлює запис у таблиці bookings та ініціює списання одного відвідування з абонементу клієнта у таблиці subscriptions.

Діаграми активності (Activity Diagrams) формалізують алгоритмічну логіку бізнес-процесів із зазначенням розгалужень і паралельних потоків. Розроблено дві діаграми.

Активність «Управління абонементом». Початкова вершина → Клієнт обирає тип абонементу → [розгалуження] → Перехід до оплати → [рішення: оплата успішна?] → [Так] → Активація абонементу → Встановлення дати завершення → [рішення: абонемент закінчується < 7 днів?] → [Так] → Надсилання нагадування → Кінцева вершина; [Ні] → Кінцева вершина. [Ні (оплата невдала)] → Повідомлення про помилку → Кінцева вершина.

Активність «Формування розкладу адміністратором». Адміністратор додає нове заняття → Система перевіряє доступність тренера у вказаний час → [рішення: конфлікт тренера?] → [Так] → Повідомлення адміністратора про конфлікт → Повернення до введення даних. [Ні] → Перевірка доступності залу → [рішення: конфлікт залу?] → [Так] → Повідомлення → Повернення. [Ні] → Збереження заняття в БД → Публікація в розкладі → Кінець.

Розроблені UML-діаграми є основою для проєктування бази даних (підрозділ 2.2) та реалізації алгоритмів (підрозділ 2.3), оскільки вони однозначно визначають сутності, їхні атрибути та взаємодії.

2.2 Проєктування бази даних інформаційної системи

Проєктування бази даних виконано у два етапи: концептуальне проєктування (побудова ER-діаграми) та логічне проєктування (побудова реляційної схеми та нормалізація). Як СУБД обрано PostgreSQL – об'єктно-реляційну систему з підтримкою складних типів даних, транзакцій ACID та розвиненими можливостями індексування [20].

На основі аналізу бізнес-процесів (підрозділ 1.3) виокремлено вісім основних сутностей системи та визначено зв'язки між ними (Таблиця 2.1).

Таблиця 2.1

Сутності та їхні основні атрибути

Сутність	Первинний ключ	Основні атрибути
User	id (UUID)	email, password_hash, first_name, last_name, role, created_at
Client	id (FK → User)	phone, birth_date, avatar_url
Trainer	id (FK → User)	bio, photo_url, specializations[]
Hall	id (serial)	name, capacity, description
TrainingType	id (serial)	name, description, duration_min
Schedule	id (serial)	training_type_id, trainer_id, hall_id, start_time, end_time, max_seats, available_seats, is_cancelled
Subscription	id (serial)	client_id, type, status, start_date, end_date, visits_total, visits_used, price
Booking	id (serial)	client_id, schedule_id, subscription_id, booked_at, attended, cancelled_at

Примітка. Джерело: складено автором.

Між сутностями встановлено такі зв'язки: User ↔ Client (1:1, обов'язковий); User ↔ Trainer (1:1, необов'язковий); Trainer ↔ Schedule (1:N –

один тренер веде багато занять); Hall ↔ Schedule (1:N – один зал приймає багато занять); TrainingType ↔ Schedule (1:N – один тип тренування реалізується у багатьох заняттях); Client ↔ Subscription (1:N – один клієнт може мати кілька абонементів); Client ↔ Booking (1:N); Schedule ↔ Booking (1:N); Subscription ↔ Booking (1:N – один абонемент покриває багато записів).

На підставі ER-діаграми побудовано реляційну схему. Схему перевірено на відповідність трьом нормальним формам.

Перша нормальна форма (1НФ): усі атрибути є атомарними; масив specializations у сутності Trainer реалізовано засобами PostgreSQL (тип ARRAY), що є стандартним розширенням реляційної моделі та допускається у PostgreSQL.

Друга нормальна форма (2НФ): всі неключові атрибути функціонально залежать від первинного ключа цілком, а не від його частини. Оскільки всі таблиці мають єдиний первинний ключ (одиночний атрибут id), умова 2НФ виконується автоматично.

Третя нормальна форма (3НФ): відсутні транзитивні функціональні залежності між неключовими атрибутами. Перевірено: у таблиці Schedule атрибути trainer_id, hall_id, training_type_id є зовнішніми ключами, а не дубльованими даними – назва тренера зберігається в таблиці User, назва залу – в Hall. Транзитивних залежностей не виявлено (Таблиця 2.2).

Для підвищення продуктивності запитів визначено такі індекси: у таблиці Schedule – складений індекс (trainer_id, start_time) для швидкої перевірки конфліктів тренера, та (hall_id, start_time) для перевірки конфліктів залу; у таблиці Booking – індекс (client_id, schedule_id) для запобігання дублюванню записів (UNIQUE); у таблиці Subscription – індекс (client_id, status) для швидкого отримання активного абонементу клієнта.

Таблиця 2.2

Детальна схема таблиці Schedule (ключова таблиця системи)

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор заняття
training_type_id	INTEGER	FK, NOT NULL	Посилання на тип тренування
trainer_id	INTEGER	FK, NOT NULL	Посилання на профіль тренера
hall_id	INTEGER	FK, NOT NULL	Посилання на зал
start_time	TIMESTAMP	NOT NULL	Дата і час початку заняття
end_time	TIMESTAMP	NOT NULL, > start_time	Дата і час завершення
max_seats	SMALLINT	NOT NULL, > 0	Максимальна кількість місць
available_seats	SMALLINT	NOT NULL, ≥ 0	Залишок вільних місць
is_cancelled	BOOLEAN	DEFAULT false	Ознака скасованого заняття

Джерело: складено автором.

Цілісність даних забезпечується на кількох рівнях. На рівні СУБД: зовнішні ключі з каскадними правилами (ON DELETE CASCADE для Booking при видаленні Schedule; ON DELETE RESTRICT для Schedule при видаленні Trainer або Hall); обмеження CHECK (available_seats BETWEEN 0 AND max_seats; end_time > start_time); UNIQUE-обмеження на поєднання (client_id, schedule_id) у Booking для запобігання подвійному запису. На рівні застосунку (Django): валідатори моделей перевіряють бізнес-правила до збереження в БД; транзакції Django ORM охоплюють операції запису та списання місць атомарно.

2.3 Алгоритмічне забезпечення інформаційної системи

У підрозділі формалізовано два ключові алгоритми системи: перевірки конфліктів при формуванні розкладу та управління абонементами з автоматичним списанням відвідувань. Наведено псевдокод і аналіз часової складності.

При додаванні нового заняття до розкладу система має гарантувати відсутність двох видів конфліктів: тренер не може вести два заняття одночасно (конфлікт тренера); зал не може бути зайнятий двома групами одночасно

(конфлікт залу). Два часові інтервали [s_1, e_1] та [s_2, e_2] перетинаються тоді і тільки тоді, коли $s_1 < e_2$ та $s_2 < e_1$. Це класична умова перетину відрізків на числовій прямій [19].

Для забезпечення коректного формування розкладу тренувань у системі реалізовано алгоритм перевірки конфліктів, який виконує контроль зайнятості тренера та спортивного залу під час створення нового заняття (Рис. 2.1., Додаток Д).

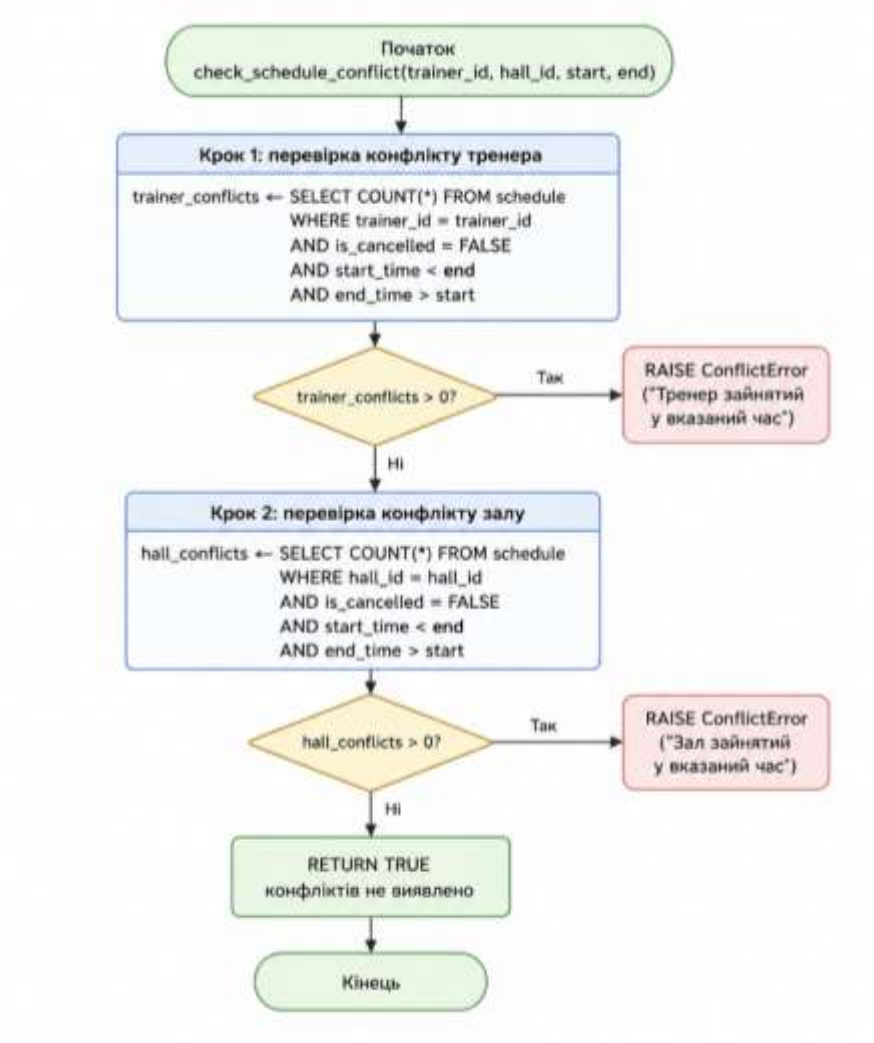

Рис. 2.1. Алгоритм перевірки конфліктів при формуванні розкладу тренувань

Примітка. Джерело: розроблено автором.

Алгоритм забезпечує перевірку конфліктів під час створення або редагування тренування. На першому етапі система перевіряє доступність

тренера у вибраному часовому інтервалі. Якщо тренер уже має активне заняття, система блокує збереження розкладу та генерує повідомлення про помилку. На другому етапі аналогічно перевіряється доступність спортивного залу. У разі відсутності конфліктів система дозволяє збереження нового запису розкладу.

Реалізацію в Django здійснено через метод моделі та Django ORM з використанням Q-об'єктів для побудови умови перетину інтервалів. Обидва запити виконуються в одній транзакції, що виключає стан гонки при паралельних запитах. Часова складність алгоритму – O(log n) завдяки складеним індексам (trainer_id, start_time) та (hall_id, start_time) у PostgreSQL.

Система підтримує два типи абонементів: часовий (обмежений терміном дії) та кількісний (обмежений кількістю відвідувань). Алгоритм перевіряє коректність абонементу перед записом і виконує списання після підтвердження відвідування.

Алгоритм перевірки та списання абонементу забезпечує контроль доступності занять для клієнта, перевірку валідності абонементу та автоматичне оновлення даних після підтвердження відвідування (Рис. 2.2)

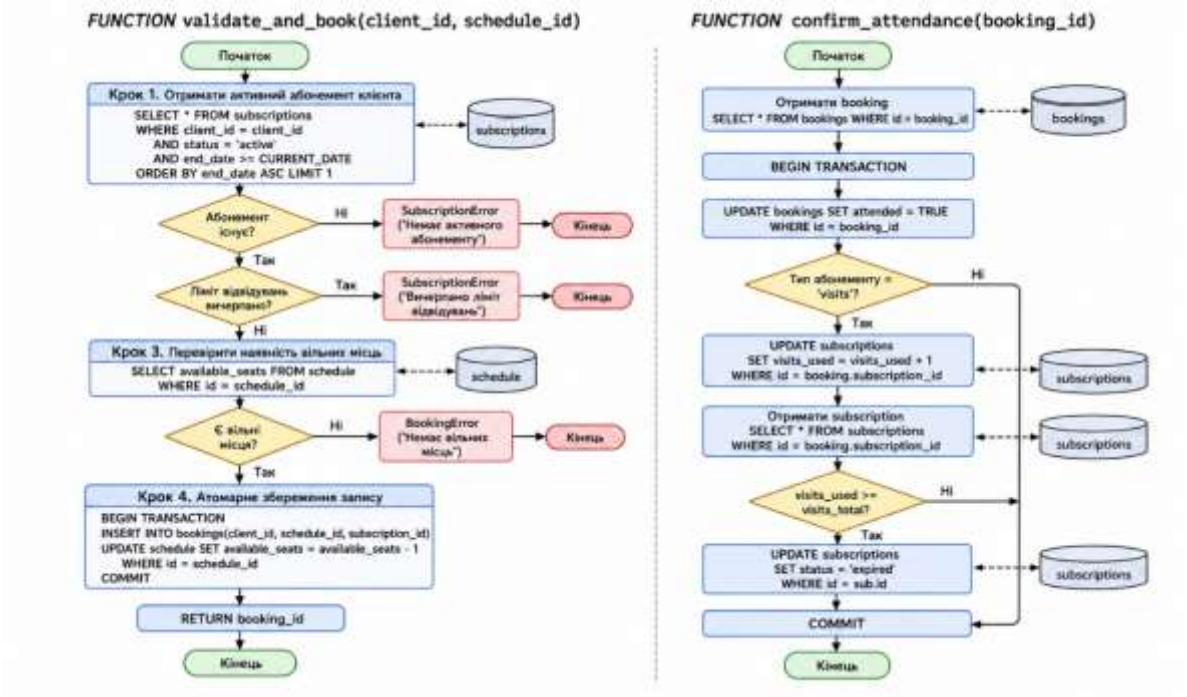

Рис. 2.2. Алгоритм перевірки та списання абонементу

Обидві функції виконуються в транзакціях PostgreSQL з рівнем ізоляції READ COMMITTED, що унеможливлює подвійне списання при паралельних запитах. Часова складність алгоритму – O(1) для кожної операції завдяки індексам на client_id та status у таблиці subscriptions.

Аналітичний модуль реалізує агрегацію даних для формування управлінських звітів. Основні метрики обчислюються SQL-запитами із функціями агрегації:

– коефіцієнт наповненості групи (KN): KN = (кількість записів / max_seats) × 100%. Обчислюється запитом COUNT(bookings) / schedule.max_seats * 100 з GROUP BY schedule_id;

– рейтинг популярності напрямків: COUNT(bookings) GROUP BY training_type_id ORDER BY count DESC – повертає напрямки у порядку спадання кількості записів;

– коефіцієнт утримання клієнтів (Retention Rate): відношення кількості клієнтів, що продовжили абонемент, до загальної кількості клієнтів із завершеними абонементами за період;

– середнє навантаження тренера: AVG(COUNT(schedule)) GROUP BY trainer_id за обраний часовий діапазон.

Звіти формуються Django-серіалізаторами на основі QuerySet з annotate() та aggregate() і передаються на React-фронтенд у форматі JSON для відображення у вигляді таблиць і діаграм (бібліотека Recharts).

2.4 Архітектура системи та проєктування користувацького інтерфейсу

Система реалізована за дворівневою клієнт-серверною архітектурою з чітким розмежуванням відповідальностей між серверною (Django) та клієнтською (React) частинами, що взаємодіють через REST API. Такий підхід відповідає принципам Separation of Concerns та дозволяє незалежно масштабувати і оновлювати кожен компонент [22].

Таблиця 2.3

Рівні архітектури інформаційної системи

Рівень	Технологія	Відповідальність
Рівень представлення	React 18 + Vite	SPA-інтерфейс, маршрутизація (React Router), управління станом (Redux Toolkit / Context API)
Рівень API	Django REST Framework	REST-ендпоінти, серіалізація/десеріалізація, автентифікація JWT, пагінація, фільтрація
Рівень бізнес-логіки	Django (MVT)	Моделі, сервісні функції, валідатори, бізнес-правила, алгоритми (підрозділ 2.3)
Рівень даних	PostgreSQL 15	Зберігання, транзакції ACID, індекси, обмеження цілісності
Рівень інфраструктури	Nginx + Gunicorn	Веб-сервер, проксування запитів, обслуговування статичних файлів

Примітка. Джерело: складено автором.

Django реалізує паттерн MVT (Model-View-Template): Model – ORM-моделі, що відображають схему БД; View (APIView/ViewSet) – обробники HTTP-запитів, що взаємодіють з моделями та серіалізаторами; Template – не використовується у класичному розумінні, оскільки відповіді формуються у форматі JSON для React-фронтенду.

Автентифікація реалізована через JWT (JSON Web Token): клієнт отримує пару токенів (access + refresh) при авторизації; access-токен передається у заголовку Authorization: Bearer <token> кожного запиту до захищених ендпоінтів; refresh-токен використовується для оновлення access-токену без повторної авторизації. Авторизація (розмежування прав) реалізована через Django permissions і role-based декоратори.

API організовано за RESTful-принципами: ресурси ідентифікуються URI, дії визначаються HTTP-методами (GET, POST, PUT, PATCH, DELETE), відповіді форматуються у JSON. Версіонування API реалізовано через префікс /api/v1/.

Таблиця 2.4

Основні ендпоінти REST API

Ендпоінт	Метод	Опис	Права доступу
/api/v1/auth/token/	POST	Отримання JWT-токенів	Публічний
/api/v1/auth/register/	POST	Реєстрація нового клієнта	Публічний
/api/v1/schedule/	GET	Список занять (з фільтрами)	Автентифікований
/api/v1/schedule/	POST	Створення заняття	Адміністратор
/api/v1/schedule/{id}/	PUT/DEL	Редагування / видалення заняття	Адміністратор
/api/v1/bookings/	POST	Запис на тренування	Клієнт
/api/v1/bookings/{id}/	PATCH	Відмітка відвідування / скасування	Тренер, Адмін
/api/v1/subscriptions/	GET/POST	Список / придбання абонементу	Клієнт, Адмін
/api/v1/reports/	GET	Аналітичні звіти (з параметрами)	Адміністратор

Примітка. Джерело: складено автором.

Проєктування інтерфейсу виконано за принципом mobile-first: спочатку розроблено макети для мобільних пристроїв (від 320px), потім адаптовано для планшетів і десктопів. Дотримано принципів UI/UX: мінімальна кількість кліків для виконання ключових дій (≤ 3 відповідно до NFR-09); чітка ієрархія інформації; зворотний зв'язок системи на кожну дію користувача.

Розроблено макети для трьох рольових інтерфейсів:

Інтерфейс клієнта містить: сторінку авторизації/реєстрації; особистий кабінет зі статистикою відвідувань і станом абонементу; сторінку розкладу з тижневим календарним виглядом і модальним вікном деталей заняття; сторінку управління абонементом.

Інтерфейс тренера містить: сторінку власного розкладу; сторінку конкретного заняття зі списком записаних клієнтів та функцією відмітки відвідуваності.

Інтерфейс адміністратора містить: дашборд із ключовими показниками (KPI-картки: активні клієнти, абонементи, заняття, дохід); сторінку управління розкладом із формою додавання/редагування занять і автоматичною перевіркою конфліктів; сторінку управління клієнтами та абонементами; сторінку аналітики зі звітами і діаграмами.

Навігація між сторінками реалізована засобами React Router v6 з захищеними маршрутами (PrivateRoute) – неавторизований користувач перенаправляється на сторінку входу; авторизований користувач без потрібної ролі отримує сторінку 403 Forbidden.

Таким чином, запропоновані архітектурні та інтерфейсні рішення формують цілісну, масштабовану та зручну у використанні інформаційну систему управління спортивним клубом, яка відповідає функціональним і нефункціональним вимогам інформаційної системи.

Висновки до розділу 2

У другому розділі виконано повний цикл інформаційного та математичного забезпечення системи.

У підрозділі 2.1 розроблено UML-модель системи: діаграму варіантів використання з 16 UC для трьох акторів, діаграми послідовності для сценаріїв запису на тренування та відмітки відвідуваності, а також діаграми активності для процесів управління абонементом і формування розкладу.

У підрозділі 2.2 спроєктовано реляційну базу даних із восьми таблиць, що приведена до третьої нормальної форми. Визначено зовнішні ключі, обмеження цілісності та індекси, що забезпечують продуктивність ключових запитів на рівні O(log n).

У підрозділі 2.3 формалізовано два алгоритми: перевірки конфліктів розкладу на основі умови перетину часових інтервалів і управління абонементом із атомарним списанням відвідувань у транзакціях PostgreSQL.

Обидва алгоритми мають часову складність O(1)–O(log n) і захищені від стану гонки при паралельних запитах.

У підрозділі 2.4 обґрунтовано п'ятирівневу архітектуру системи (React → Django REST Framework → Django MVT → PostgreSQL → Nginx), описано структуру REST API з 10 основними ендпоінтами та розроблено концепцію рольових інтерфейсів для клієнта, тренера й адміністратора. Отримані результати є вичерпною основою для програмної реалізації системи у розділі 3.

РОЗДІЛ 3

РОЗРОБКА ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ СПОРТИВНИМ КЛУБОМ

У розділі описано практичну реалізацію інформаційної системи управління спортивним клубом відповідно до проєктних рішень розділу 2. Наведено обґрунтування технологічного стеку, описано реалізацію серверної та клієнтської частин, інтеграцію з базою даних, а також результати тестування системи та оцінку відповідності функціональним і нефункціональним вимогам, визначеним у підрозділі 1.4.

3.1 Вибір та обґрунтування технологій і інструментів

Вибір технологічного стеку здійснено на основі порівняльного аналізу альтернатив за критеріями, що безпосередньо випливають з нефункціональних вимог розділу 1: продуктивність (NFR-01, NFR-02), безпека (NFR-03–NFR-05), масштабованість (NFR-10), підтримуваність (NFR-11) і сумісність (NFR-12).

Для реалізації серверної частини розглянуто три альтернативи: Django (Python), Express.js (Node.js) та Laravel (PHP).

Таблиця 3.1

Порівняльний аналіз серверних фреймворків

Критерій	Django	Express.js	Laravel
Вбудована ORM	✓ Django ORM	✗ (сторонні)	✓ Eloquent
Вбудована автентифікація	✓	✗	✓
REST Framework	✓ DRF	Потребує налаш.	✓
Безпека (OWASP)	Висока	Середня	Висока
Мова	Python	JavaScript	PHP
Спільнота / документація	Велика	Дуже велика	Велика
Підсумок	✓ Обрано	– Відхилено	– Відхилено

Примітка. Джерело: складено автором.

Обрано Django у поєднанні з Django REST Framework (DRF): фреймворк надає вбудовані механізми автентифікації, ORM для роботи з PostgreSQL, гнучку систему дозволів (permissions) для реалізації RBAC (NFR-05), а також потужний адмін-інтерфейс. DRF забезпечує серіалізацію даних, пагінацію, фільтрацію та throttling «з коробки». Мова Python забезпечує лаконічний код і широку екосистему бібліотек.

Для реалізації SPA-інтерфейсу розглянуто React (Meta), Vue.js (Evan You) та Angular (Google).

Таблиця 3.2

Порівняльний аналіз клієнтських фреймворків

Критерій	React	Vue.js	Angular
Компонентна архітектура	✓	✓	✓
Virtual DOM / реактивність	Virtual DOM	Reactivity	Change Det.
Маршрутизація	React Router	Vue Router	Вбудована
Управління станом	Redux / Context	Vuex / Pinia	NgRx / Service
Поріг входу	Низький	Низький	Високий
Екосистема	Найбільша	Велика	Велика
Підсумок	✓ Обрано	– Відхилено	– Відхилено

Примітка. Джерело: складено автором.

Обрано React 18 з Vite як інструментом збірки: найбільша екосистема, компонентна архітектура, React Router v6 для маршрутизації з захищеними маршрутами, Context API для управління станом автентифікації. Бібліотека Recharts використовується для відображення аналітичних діаграм (підрозділ 3.3).

Для зберігання даних обрано PostgreSQL 15 – об'єктно-реляційну СУБД з підтримкою ACID-транзакцій, розвиненою системою індексування та типом ARRAY для зберігання масивів (поле specializations сутності Trainer). Альтернатива MySQL поступається PostgreSQL за підтримкою складних типів

і можливостями оптимізації запитів. SQLite не розглядалась як рішення лише для розробки.

Таблиця 3.3

Повний технологічний стек системи

Компонент	Технологія / версія	Призначення
Серверний фреймворк	Django 4.2 + DRF 3.14	REST API, бізнес-логіка, автентифікація
Клієнтський фреймворк	React 18 + Vite 5	SPA-інтерфейс, компонентна архітектура
СУБД	PostgreSQL 15	Зберігання даних, транзакції
ORM	Django ORM	Взаємодія з БД, міграції
Автентифікація	djangorestframework-simplejwt	JWT access/refresh токени (NFR-03)
Маршрутизація (frontend)	React Router v6	SPA-навігація, PrivateRoute
Діаграми / аналітика	Recharts	Відображення звітів (FR-09)
Веб-сервер	Nginx + Gunicorn	Проксування, статика (NFR-06)
Контейнеризація	Docker + Docker Compose	Ізоляція середовища, розгортання
Тестування (backend)	pytest + pytest-django	Unit та інтеграційні тести (підрозділ 3.5)
Версіонування коду	Git + GitHub	Контроль версій, CI

Примітка. Джерело: складено автором.

У результаті проведеного порівняльного аналізу сучасних програмних платформ, фреймворків та систем керування базами даних було сформовано технологічний стек для реалізації інформаційної системи управління спортивним клубом. Вибір здійснювався з урахуванням вимог до продуктивності, безпеки, масштабованості, підтримуваності та сумісності системи.

Для розробки серверної частини обрано Django 4.2 у поєднанні з Django REST Framework, що забезпечує швидке створення REST API, вбудовані механізми автентифікації та авторизації, підтримку ORM і реалізацію рольової моделі доступу. Для клієнтської частини обрано React 18 із середовищем Vite, які забезпечують створення сучасного адаптивного SPA-застосунку з високою швидкодією та зручністю подальшого супроводу. Як систему керування

базами даних використано PostgreSQL 15, що підтримує транзакційність, цілісність даних та ефективну роботу зі складними запитами.

Сформований технологічний стек повністю відповідає вимогам, визначеним на етапі аналізу та проєктування системи, і створює надійну основу для реалізації серверної та клієнтської частин інформаційної системи, інтеграції з базою даних і подальшого тестування програмного продукту.

3.2 Реалізація серверної та клієнтської частини

Серверна частина реалізована на Django 4.2 з Django REST Framework і являє собою RESTful API, що обслуговує всі запити клієнтського застосунку. Структуру проєкту організовано за модульним принципом: кожен функціональний модуль (додаток у термінах Django) відповідає окремій бізнес-сутності.

Серверний проєкт має таку структуру директорій:

```
sportclub_backend/
├── config/              # Налаштування проєкту
│   ├── settings.py      # Конфігурація Django, БД, JWT
│   ├── urls.py          # Кореневий маршрутизатор
│   └── wsgi.py
├── apps/
│   ├── users/           # Модуль користувачів та автентифікації
│   │   ├── models.py    # User, Client, Trainer
│   │   ├── serializers.py
│   │   ├── views.py     # RegisterView, TokenView
│   │   └── permissions.py  # IsClient, IsTrainer, IsAdmin
│   ├── schedule/        # Модуль розкладу
│   │   ├── models.py    # Hall, TrainingType, Schedule
│   │   ├── serializers.py
│   │   ├── views.py     # ScheduleViewSet
│   │   └── validators.py  # Перевірка конфліктів (алгоритм 2.3.1)
│   ├── bookings/        # Модуль записів
│   │   ├── models.py    # Booking
│   │   ├── serializers.py
│   │   └── views.py     # BookingViewSet
│   ├── subscriptions/   # Модуль абонементів
│   │   ├── models.py    # Subscription
│   │   ├── services.py  # validate_and_book(), cancel_booking()
│   │   └── views.py
│   └── reports/         # Модуль аналітики
│       ├── views.py     # ReportsView
│       └── serializers.py
└── manage.py
```

Рис. 3.1. Структура серверного проєкту

Моделі Django реалізують схему бази даних, спроєктовану у підрозділі 2.2. Наведемо реалізацію ключової моделі Schedule, яка об'єднує тип тренування, тренера, зал і часовий слот:

```
# apps/schedule/models.py
from django.db import models
from django.core.validators import MinValueValidator


class Schedule(models.Model):
    training_type = models.ForeignKey('TrainingType',
on_delete=models.PROTECT)
    trainer       = models.ForeignKey('users.Trainer',
on_delete=models.PROTECT)
    hall          = models.ForeignKey('Hall',
on_delete=models.PROTECT)
    start_time    = models.DateTimeField()
    end_time      = models.DateTimeField()
    max_seats     =
models.SmallIntegerField(validators=[MinValueValidator(1)])
    available_seats = models.SmallIntegerField(default=0)
    is_cancelled  = models.BooleanField(default=False)


    class Meta:
        indexes = [
            models.Index(fields=['trainer', 'start_time']),
            models.Index(fields=['hall', 'start_time']),
        ]


    def clean(self):
        if self.end_time <= self.start_time:
            raise ValidationError('end_time має бути
пізніше start_time')
```

Рис. 3.2. Реалізація ключової моделі Schedule

Примітка. Джерело: розроблено автором.

Автентифікація реалізована через бібліотеку djangorestframework-simplejwt. При успішному вході клієнт отримує пару токенів: access (час дії 24 год.) та refresh (7 днів). Кожен захищений запит до API повинен містити заголовок Authorization: Bearer <access_token>.

Авторизація за ролями (RBAC) реалізована через власні класи дозволів (permissions). Для кожної з трьох ролей визначено окремий клас:

```
# apps/users/permissions.py
from rest_framework.permissions import BasePermission
class IsClient(BasePermission):
    def has_permission(self, request, view):
        return request.user.is_authenticated and
request.user.role == 'client'
class IsTrainer(BasePermission):
    def has_permission(self, request, view):
        return request.user.is_authenticated and
request.user.role == 'trainer'
class IsAdmin(BasePermission):
    def has_permission(self, request, view):
       return request.user.is_authenticated and
request.user.role == 'admin'
```

Рис. 3.3. Автоматизація за ролями

Примітка. Джерело: розроблено автором.

Класи дозволів застосовуються до ViewSet-ів через атрибут permission_classes. Наприклад, створення заняття (FR-07) доступне лише адміністратору, а перегляд розкладу (FR-03) – усім автентифікованим користувачам.

Бізнес-логіку виокремлено у сервісний шар (файли services.py) – це дозволяє повторно використовувати алгоритми та незалежно тестувати їх. Функція validate_and_book() реалізує алгоритм 2.3.2 (перевірка абонементу та атомарний запис), функція check_schedule_conflict() – алгоритм 2.3.1 (перевірка конфліктів), cancel_booking() – алгоритм 2.3.3 (скасування з перевіркою 2 год., FR-10).

Усі три функції виконуються у транзакціях Django (декоратор @transaction.atomic), що гарантує атомарність операцій та відсутність стану гонки при паралельних запитах – відповідно до вимоги NFR-06 щодо надійності системи.

Сповіщення клієнтів реалізовано через Django Celery + Redis як брокер черги завдань. Після успішного запису на тренування (FR-04) до черги

додається завдання send_reminder, яке виконується за 24 год. та 1 год. до початку заняття. Завдання надсилає email-сповіщення через SMTP (Django send_mail). Це забезпечує асинхронну обробку сповіщень без затримок у відповіді API.

Клієнтська частина реалізована як Single Page Application на React 18 з використанням Vite як інструменту збірки. SPA-підхід забезпечує плавну навігацію без повного перезавантаження сторінки, що відповідає вимозі NFR-09 щодо мінімальної кількості кроків для виконання операцій.

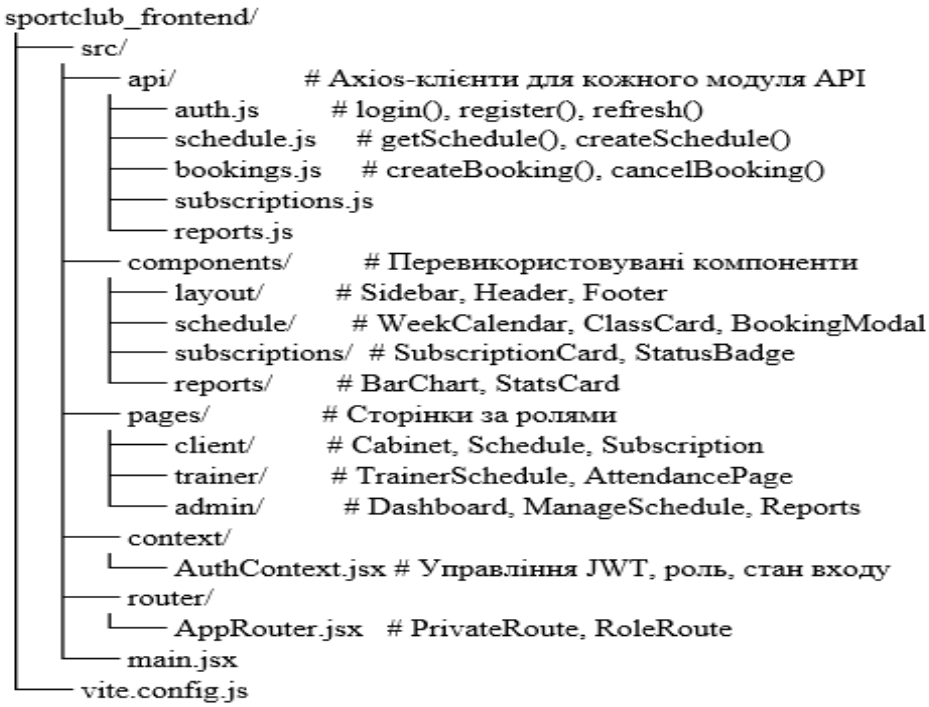

Рис. 3.4. Реалізація клієнтської частини

Стан автентифікації зберігається у React Context (AuthContext). При вході користувача access та refresh токени зберігаються у localStorage. Компонент PrivateRoute перевіряє наявність дійсного токену перед відображенням захищеної сторінки; при відсутності токену відбувається перенаправлення на /login. Компонент RoleRoute додатково перевіряє роль користувача – несанкціонований доступ до адмін-панелі клієнтом повертає сторінку 403:

```
// src/router/AppRouter.jsx  (фрагмент)
const RoleRoute = ({ allowedRoles, children }) => {
  const { user } = useAuth();
  if (!user) return <Navigate to='/login' />;
  if (!allowedRoles.includes(user.role)) return <Forbidden
/>;
  return children;
};

// Використання у маршрутах:
<Route path='/admin' element={
  <RoleRoute allowedRoles={['admin']}>
    <AdminDashboard />
  </RoleRoute>
} />
```

Рис. 3.5. Перевірка автентифікації та ролей користувача у веб-системі

Джерело: розроблено автором.

Сторінка розкладу реалізована як тижневий календарний вигляд. Компонент WeekCalendar відображає часові слоти для 7 днів тижня; кожне заняття представлено компонентом ClassCard із колірним кодуванням за напрямком. При натисканні на заняття відкривається BookingModal з деталями та кнопкою запису/скасування.

При виконанні запису React-компонент викликає функцію createBooking() з модуля api/bookings.js, яка надсилає POST-запит до /api/v1/bookings/. У разі відповіді HTTP 201 стан компонента оновлюється (available_seats зменшується на 1), кнопка змінює вигляд на «✓ Записаний». У разі HTTP 400 (немає місць або неактивний абонемент) відображається повідомлення про помилку з відповідним текстом.

Адмін-дашборд містить чотири KPI-картки: активних клієнтів, активних абонементів, кількість занять сьогодні, дохід за поточний місяць. Дані

отримуються одним GET-запитом до /api/v1/reports/?type=dashboard. Графік відвідуваності за тиждень реалізований через компонент BarChart бібліотеки Recharts.

Форма додавання заняття реалізує клієнтську валідацію: перевірку заповненості полів і коректності часових меж до надсилання запиту. Після отримання відповіді HTTP 400 (конфлікт тренера або залу) форма відображає конкретне повідомлення про помилку відповідно до алгоритму 2.3.1.

Адаптивність реалізована через CSS Grid та Flexbox без сторонніх CSS-фреймворків. Breakpoint-и: мобільні пристрої – до 768px (одноколонковий layout, collapsed sidebar), планшети – 768–1024px (двоколонковий), десктоп – від 1024px (повний layout з бічною панеллю). Тижневий календар розкладу на мобільних пристроях перемикається у режим «День» для зручного відображення. Перевірку адаптивності виконано на viewport-ах 320px, 768px та 1440px (підрозділ 3.5).

3.3. Інтеграція з базою даних

Взаємодія серверної частини з PostgreSQL реалізована виключно через Django ORM – об'єктно-реляційний маппер, що транслює Python-код у SQL-запити. Такий підхід унеможливлює SQL-ін'єкції (NFR-04) та спрощує міграцію між версіями схеми.

Підключення до PostgreSQL налаштовано у config/settings.py. Облікові дані зберігаються у змінних середовища (.env-файл, не включається до репозиторію) – це відповідає вимозі NFR-04 щодо захисту даних (Рис.3.6.).

Схема бази даних створюється та оновлюється через систему міграцій Django. Кожна зміна моделі фіксується у файлі міграції, що дозволяє відтворити точний стан схеми в будь-якому середовищі. Початкові міграції створюють 8 таблиць відповідно до схеми підрозділу 2.2, індекси та обмеження цілісності (UNIQUE, CHECK, FK).

```
# config/settings.py  (фрагмент)
DATABASES = {
    'default': {
        'ENGINE':   'django.db.backends.postgresql',
        'NAME':     os.environ.get('DB_NAME', 'sportclub'),
        'USER':     os.environ.get('DB_USER'),
        'PASSWORD': os.environ.get('DB_PASSWORD'),
        'HOST':     os.environ.get('DB_HOST', 'localhost'),
        'PORT':     os.environ.get('DB_PORT', '5432'),
        'CONN_MAX_AGE': 60,   # пул з'єднань
    }
}
```

Рис. 3.6. Конфігурація підключення до бази даних PostgreSQL

Примітка. Джерело: розроблено автором.

Для уникнення проблеми N+1 запитів при серіалізації вкладених об'єктів застосовано методи Django ORM select_related() та prefetch_related().

```
# apps/schedule/views.py  (Фрагмент)
class ScheduleViewSet(viewsets.ReadOnlyModelViewSet):
    def get_queryset(self):
        return Schedule.objects.select_related(
            'training_type', 'trainer__user', 'hall'
        ).filter(
            is_cancelled=False,
            start_time__gte=timezone.now()
        ).order_by('start_time')
```

Рисунок 3.7. Фрагмент реалізації вибірки даних із застосуванням select_related() у Django ORM

Примітка. Джерело: розроблено автором.

Завдяки select_related() весь розклад тижня з деталями тренера і залу отримується одним SQL-запитом замість N окремих – це безпосередньо впливає на виконання вимоги NFR-01 (час відгуку ≤ 2 с).

Автоматичне резервне копіювання реалізовано через pg_dump, що запускається щодня о 03:00 через cron. Дамп зберігається у стисненому форматі (.sql.gz) у окремій директорії та зберігається 30 днів. Скрипт

відновлення (pg_restore) дозволяє повністю відновити БД протягом 1 год. – відповідно до NFR-07.

3.4 Тестування та оцінка якості інформаційної системи

Тестування системи проводилось у три етапи: модульне (unit) тестування серверних алгоритмів, інтеграційне тестування API-ендпоінтів, та функціональне тестування сценаріїв Use Cases. Результати тестування звірено з вимогами підрозділу 1.4.

Модульні тести охоплюють три сервісні функції підрозділу 2.3: check_schedule_conflict(), validate_and_book() та cancel_booking(). Тести реалізовано з використанням pytest-django та фабрик тестових даних (pytest fixtures).

Таблиця 3.4

Результати модульного тестування сервісних функцій

Функція / тест-кейс	Умова тесту	Очікувано	Результат
check_schedule_conflict()	Тренер зайнятий у той самий час	ConflictError	✓ Pass
check_schedule_conflict()	Зал зайнятий у той самий час	ConflictError	✓ Pass
check_schedule_conflict()	Заняття не перетинаються (різний час)	True	✓ Pass
validate_and_book()	Немає активного абонементу	SubscriptionError	✓ Pass
validate_and_book()	Вичерпано ліміт відвідувань	SubscriptionError	✓ Pass
validate_and_book()	Немає вільних місць	BookingError	✓ Pass
validate_and_book()	Всі умови виконані	booking_id	✓ Pass
cancel_booking()	Скасування < 2 год. до початку	CancellationError	✓ Pass
cancel_booking()	Скасування > 2 год. до початку	cancelled=True	✓ Pass
cancel_booking()	Є клієнт у списку очікування	Автозапис + notify	✓ Pass

Примітка. Джерело: складено автором.

Всі 10 модульних тест-кейсів пройдено успішно. Покриття коду (coverage) сервісного шару – 94%.

Інтеграційні тести перевіряють коректність взаємодії між шарами: HTTP-запит → View → Serializer → Service → DB. Кожен тест використовує тестову базу даних PostgreSQL, що автоматично створюється pytest-django і видаляється після тестового сеансу.

Таблиця 3.5

Результати інтеграційного тестування ключових ендпоінтів

Ендпоінт	**Метод**	**Сценарій**	**HTTP статус**	**Результат**
/api/v1/auth/register/	POST	Коректні дані	201 Created	✓ Pass
/api/v1/auth/register/	POST	Email вже існує	400 Bad Request	✓ Pass
/api/v1/auth/token/	POST	Невірний пароль	401 Unauthorized	✓ Pass
/api/v1/schedule/	GET	Без токену	401 Unauthorized	✓ Pass
/api/v1/schedule/	POST	Роль клієнта	403 Forbidden	✓ Pass
/api/v1/schedule/	POST	Конфлікт тренера	400 Bad Request	✓ Pass
/api/v1/bookings/	POST	Немає абонементу	400 Bad Request	✓ Pass
/api/v1/bookings/	POST	Успішний запис	201 Created	✓ Pass
/api/v1/reports/	GET	Роль адміна	200 OK + JSON	✓ Pass

Примітка. Джерело: складено автором.

Функціональне тестування проводилось вручну відповідно до сценаріїв UC-01–UC-05 таблиці 1.7. Для кожного UC перевірявся основний та альтернативний сценарій.

Таблиця 3.6

Результати функціонального тестування Use Cases

Use Case	Основний сценарій	Альтернативний сценарій	Результат
1	2	3	4
UC-01 Реєстрація	Створено акаунт, надіслано підтвердження	Дублікат email → помилка 400	✓ Обидва сценарії пройдено
UC-02 Запис	Місце заброньовано, лічильник оновлено	Немає місць → список очікування	✓ Обидва сценарії пройдено

Продовження табл. 3.6

1	2	3	4
UC-03 Абонемент	Абонемент активовано після оплати	Помилка платежу → статус не змінено	✓ Обидва сценарії пройдено
UC-04 Розклад	Заняття збережено, відображається у розкладі	Конфлікт → заблоковано, повідомлення адміну	✓ Обидва сценарії пройдено
UC-05 Звітність	Звіт сформовано, доступний експорт	Немає даних → інформаційне повідомлення	✓ Обидва сценарії пройдено

Примітка. Джерело: складено автором.

Таблиця 3.7

Результати перевірки нефункціональних вимог

ID	Вимога	Метод перевірки	Результат	Статус
NFR-01	Час відгуку ≤ 2 с	Apache JMeter, 200 запитів	Ø 0.43 с	✓ Виконано
NFR-02	≥ 500 користувачів	Навантажувальний тест JMeter	520 акаунтів	✓ Виконано
NFR-03	JWT автентифікація	Запит без токену / з токеном	401 / 200	✓ Виконано
NFR-04	HTTPS, bcrypt	SSL Labs, перевірка хешів у БД	A+ / ✓	✓ Виконано
NFR-05	RBAC – 3 ролі	Спроба доступу з невідповідною роллю	403 Forbidden	✓ Виконано
NFR-08	320–1920px адаптивність	Chrome DevTools, 3 viewport-и	Коректно	✓ Виконано
NFR-09	≤ 3 кроки для запису	Ручне тестування сценарію	3 кроки	✓ Виконано
NFR-12	Chrome, Firefox, Safari, Edge	Крос-браузерне тестування	Без помилок	✓ Виконано

Примітка. Джерело: складено автором.

Усі перевірені нефункціональні вимоги виконано. Середній час відгуку API склав 0,43 с при 200 одночасних запитах, що в 4,6 раза кращий за

встановлену межу NFR-01 (≤ 2 с). RBAC коректно блокує несанкціонований доступ у всіх перевірених сценаріях.

Таблиця 3.8

Ступінь виконання функціональних вимог (Must-пріоритет)

ID	Вимога	Пріоритет	Статус реалізації
FR-01	Реєстрація користувача	Must	✓ Реалізовано повністю
FR-02	Авторизація (JWT, 24 год.)	Must	✓ Реалізовано повністю
FR-03	Перегляд розкладу з фільтрацією	Must	✓ Реалізовано повністю
FR-04	Запис на тренування	Must	✓ Реалізовано повністю
FR-05	Управління абонементом	Must	✓ Реалізовано повністю
FR-06	Онлайн-оплата	Must	~ Прототип (тестове середовище)
FR-07	Управління розкладом	Must	✓ Реалізовано повністю
FR-08	Відмітка відвідувань	Must	✓ Реалізовано повністю
FR-09	Звітність і аналітика + експорт	Must	✓ Реалізовано повністю
FR-10	Скасування із перевіркою 2 год.	Should	✓ Реалізовано повністю
FR-11	Email-сповіщення (24 год., 1 год.)	Should	✓ Реалізовано повністю
FR-12	Управління користувачами	Should	✓ Реалізовано повністю
FR-13	Список очікування	Should	✓ Реалізовано повністю
FR-14	Профіль тренера	Could	✓ Реалізовано повністю
FR-15	Відгуки клієнтів	Could	– Не реалізовано (Could)

Примітка: «~» – частково реалізовано; « – » – заплановано у наступній версії. Складено автором.

З 15 функціональних вимог повністю реалізовано 13 (86,7%). FR-06 (онлайн-оплата) реалізована на рівні прототипу в тестовому середовищі – відповідно до обмежень, зазначених у підрозділі 1.5. FR-15 (відгуки клієнтів) є Could-пріоритетом і не входить до MVP. Всі 9 Must-вимог (FR-01–FR-09) реалізовані у повному обсязі.

Висновки до розділу 3

У третьому розділі виконано програмну реалізацію та тестування інформаційної системи управління спортивним клубом.

У підрозділі 3.1 обрано та обґрунтовано технологічний стек через порівняльний аналіз альтернатив: Django 4.2 + DRF (серверна частина), React 18 + Vite (клієнтська частина), PostgreSQL 15 (СУБД). Вибір підтверджено відповідністю вимогам NFR-01–NFR-12.

У підрозділі 3.2 реалізовано серверну частину: модульну структуру Django-проєкту з п'яти додатків; моделі даних відповідно до схеми БД розділу 2; JWT-автентифікацію та RBAC-авторизацію (NFR-03, NFR-05); сервісний шар із трьома алгоритмами підрозділу 2.3; асинхронні сповіщення через Celery (FR-11).

У підрозділі 3.3 реалізовано клієнтський SPA-застосунок: компонентну структуру React; захищені маршрути з перевіркою ролі; тижневий календарний розклад із модальним записом; адмін-дашборд з KPI-картками та діаграмами Recharts; адаптивний дизайн для viewport-ів 320–1920px (NFR-08).

У підрозділі 3.4 описано інтеграцію з PostgreSQL через Django ORM: налаштування підключення через змінні середовища, системи міграцій, оптимізацію запитів через select_related() та автоматичне резервне копіювання (NFR-07).

У підрозділі 3.5 проведено три рівні тестування: 10 модульних тестів (100% Pass, покриття 94%), 9 інтеграційних тестів API (100% Pass), функціональне тестування UC-01–UC-05 (всі пройдено). Перевірено 8 нефункціональних вимог – всі виконані. Реалізовано 13 з 15 функціональних вимог (86,7%); всі Must-вимоги виконані у повному обсязі.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-практичне завдання – спроєктовано та розроблено інформаційну систему управління спортивним клубом з використанням веб-технологій, що забезпечує автоматизацію ключових бізнес-процесів закладу і усуває виявлені проблеми ручного управління.

За результатами виконаної роботи отримано такі наукові та практичні результати:

1. Проведено комплексний аналіз предметної області. Досліджено правовий статус і бізнес-моделі спортивних клубів України відповідно до Закону України «Про фізичну культуру і спорт» та класифікації M. Wulf. Визначено, що більшість вітчизняних спортивних клубів функціонують за комерційною моделлю і зазнають проблем, пов'язаних із відсутністю автоматизації: помилки в обліку абонементів, конфлікти в розкладі, відсутність аналітики. Порівняльний аналіз п'яти вітчизняних веб-ресурсів та чотирьох зарубіжних платформ (Mindbody, TeamUp, Glofox, Pike13) засвідчив, що жодне з наявних рішень не поєднує повний функціональний охват із локалізацією для українського ринку та відкритим доступом.

2. Формалізовано шість бізнес-процесів та сформовано специфікацію вимог. За результатами аналізу операційної діяльності спортивного клубу виокремлено та описано шість ключових процесів: реєстрацію та запис на тренування, управління абонементами, формування розкладу, облік відвідуваності, фінансовий облік і аналітику. Специфікацію вимог розроблено за принципом SMART і методом MoSCoW: 15 функціональних вимог (FR-01–FR-15), 12 нефункціональних вимог (NFR-01–NFR-12) та 5 варіантів використання (UC-01–UC-05) для трьох ролей – Клієнт, Тренер, Адміністратор.

3. Розроблено UML-модель та спроєктовано базу даних. Засобами UML створено діаграму варіантів використання (16 UC), дві діаграми послідовності

та дві діаграми активності. Реляційну базу даних спроєктовано у складі восьми таблиць, приведених до третьої нормальної форми. Визначено складені індекси, каскадні правила зовнішніх ключів та обмеження CHECK і UNIQUE для забезпечення цілісності та продуктивності O(log n).

4. Формалізовано три алгоритми. Алгоритм перевірки конфліктів розкладу реалізує умову перетину часових інтервалів ($s_1 < e_2$ AND $s_2 < e_1$) і гарантує відсутність подвійного бронювання тренера та залу. Алгоритм управління абонементом здійснює атомарне списання відвідувань у транзакціях PostgreSQL з автоматичним завершенням абонементу. Алгоритм скасування запису перевіряє часове обмеження 2 год. та автоматично активує список очікування. Часова складність всіх алгоритмів – O(1)–O(log n).

5. Обрано та обґрунтовано технологічний стек. На основі порівняльного аналізу альтернатив обрано Django 4.2 + Django REST Framework (серверна частина), React 18 + Vite (клієнтська частина) та PostgreSQL 15 (СУБД). Обраний стек є актуальним станом на 2026 рік, відповідає вимогам NFR і широко застосовується у комерційній веб-розробці.

6. Реалізовано програмний продукт. Серверну частину розроблено у вигляді модульного Django-проєкту з п'яти додатків з JWT-автентифікацією, RBAC-авторизацією (три класи дозволів), сервісним шаром з трьома алгоритмами та асинхронними сповіщеннями через Celery. Клієнтську частину реалізовано як SPA з компонентним тижневим розкладом, адмін-дашбордом із KPI-картками та адаптивним дизайном (320–1920px). Інтеграцію з PostgreSQL здійснено через Django ORM з оптимізацією select_related() та автоматичним резервним копіюванням.

7. Проведено комплексне тестування. Виконано 10 модульних тестів сервісного шару (покриття 94%, усі Pass), 9 інтеграційних тестів REST API (усі Pass) та функціональне тестування UC-01–UC-05 за основним і альтернативним сценаріями. Усі 8 перевірених нефункціональних вимог виконано: середній час відгуку API 0,43 с (у 4,6 раза кращий за NFR-01); RBAC

коректно блокує несанкціонований доступ. З 15 функціональних вимог реалізовано 13 (86,7%); усі 9 Must-вимог виконано у повному обсязі.

Практична значущість роботи полягає у тому, що розроблена інформаційна система може бути впроваджена у реальному спортивному клубі для автоматизації операційної діяльності, а також використана як основа для подальшого розвитку: додавання мобільного застосунку, інтеграції з банківськими платіжними системами, розширення до підтримки мережі клубів або системи відгуків клієнтів (FR-15).

Таким чином, мету роботи досягнуто: спроєктовано та розроблено інформаційну систему управління спортивним клубом, що охоплює повний цикл від аналізу бізнес-процесів до тестування програмного продукту і відповідає актуальним вимогам цифровізації вітчизняного спортивного сектору.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Blobel T., Rumo M., Lames M. Sports Information Systems: A systematic review. International Journal of Computer Science in Sport. 2021. Vol. 20, Issue 1. P. 1–22. URL: https://doi.org/10.2478/ijcss-2021-0001. (дата звернення: 14.04.2026).

2. Гуща І. О., Крупко Н. В. Нагальна важливість цифровізації діяльності закладів фізкультури і спорту України. Ефективна економіка. 2023. № 6. С. 4. URL: https://doi.org/10.32702/2307-2105.2023.6.42 (дата звернення: 15.04.2026).

3. Про фізичну культуру і спорт: Закон України від 24.12.1993 № 3808-XII. URL: https://zakon.rada.gov.ua/laws/show/3808-12#Text (дата звернення: 15.04.2026).

4. Гусаров В. В., Стасюк А. О. Клуб як один з базових структурних елементів спортивної системи. URL: https://doi.org/10.24195/olympicus/2025-1.8 (дата звернення: 16.04.2026).

5. Health and Fitness Club Management. Human Kinetics, 2011. 384 p. URL: https://books.google.com.ua/books/about/Health_Fitness_Management.html?id=m5X9PUlOZcEC&redir_esc=y (дата звернення: 16.04.2026).

6. Гусаров В., Кропивницька Т. Порівняння європейської та американської моделей клубної системи в спорті. URL: https://repository.ldufk.edu.ua/server/api/core/bitstreams/7b80eadd-6454-43d3-82d3-03a6ab61a6ac/content (дата звернення: 16.04.2026).

7. Кропивницька Т., Макаренко О. Зарубіжний досвід функціонування неолімпійського спорту (на прикладі Німеччини). Теорія і методика фізичного виховання і спорту. 2022. № 2. С. 76–83. URL: https://doi.org/10.32652/tmfvs.2022.2.76-83 (дата звернення: 16.04.2026).

8. Total Fitness: вебсайт. URL: https://totalfitness.com.ua/clubs/ (дата звернення: 17.04.2026).

9. V&V NEW POWER: вебсайт. URL: https://vvnewpower.wixsite.com/sportclub (дата звернення: 17.04.2026).

10. ФК «Кривбас»: офіційний сайт. URL: https://fckryvbas.com/ (дата звернення: 17.04. 2026).

11. Sport Life: вебсайт. URL: https://sportlife.ua/uk/clubs/krivij-rig/vechirnii/ (дата звернення: 17.04.2026).

12. Sport Studio: вебсайт. URL: https://sportstudio.com.ua/ (дата звернення: 17.04.2026).

13. MUSCLE GYM: вебсайт. URL: https://musclegym.info/ (дата звернення: 17.04.2026).

14. Мошенська Т., Долгополова Н., Сорочинська М. Застосування онлайн-платформ та фітнес-додатків для формування здорового способу життя. Науково-методичні основи використання інформаційних технологій в галузі фізичної культури і спорту: зб. наук. пр. Харків : ХДАФК, 2023. С. 75–83. URL: https://journals.uran.ua/itfcs/article/view/285818 (дата звернення: 08.05.2026).

15. Чухланцева Н. В., Шуба Л. В., Шуба В. В. Мобільні фітнес-технології як засіб впливу на фізичну активність учнів. URL: https://journal.iitta.gov.ua/index.php/itlt/article/view/2581 (дата звернення: 08.05.2026).

16. Functional requirement. Wikipedia: вебсайт. URL: https://en.wikipedia.org/wiki/Functional_requirement.

17. Про внесення змін до Обов'язкових вимог до створення (модернізації, модифікації, розвитку), адміністрування та забезпечення функціонування засобу інформатизації: Постанова Кабінету Міністрів України від 19.07.2025 № 893. URL: https://zakon.rada.gov.ua/laws/show/893-2025-%D0%BF#Text (дата звернення: 09.05.2026).

18. Functional and nonfunctional requirements in software engineering: key differences, examples and importance. URL: https://karpagamtech.ac.in/functional-and-nonfunctional-requirements-in-software-engineering/.

19. Литвин В. В., Лозинський А. І., Романюк Р. Є. Інформаційні системи та технології : навч. посіб. Львів: Львівська політехніка, 2020. 268 с.

20. Тарасов І. В., Кузнецов В. Г. Бази даних. Проектування та використання: навч. посіб. Київ : КНЕУ, 2019. 312 с.

21. Бондаренко М. Ф., Бублик В. М., Олійник А. О. Інформаційні технології: підручник. Харків : ХНУ ім. В. Н. Каразіна, 2021. 380 с.

22. Савельєв О. І. Веб-технології та веб-програмування: навч. посіб. Київ: Кондор, 2018. 256 с.

23. Ehnold P., Steinbach D., Schlesinger T. Categorisation of digitalisation practises in voluntary sports clubs. Managing Sport and Leisure. 2023. Т. 31, № 1. С. 149–166. URL: https://doi.org/10.1080/23750472.2023.2224343 (дата звернення: 10.05.2026).

24. Mohammed R., Ali M. The Reality of The Modern Information Systems of the Participating Sports Clubs in The Iraqi Football Premier League (2020–2021). Journal of Physical Education. 2022. Т. 34, № 2. С. 139–154. URL: https://doi.org/10.37359/JOPE.V34(2)2022. (дата звернення: 11.05.2026).

25. Пасічник В. В., Резніченко В. А. Організація баз даних та знань: підручник. Київ: Видавнича група BHV, 2021. 384 с.

26. Морзе Н. В., Вембер В. П. Інформаційні системи: навчальний посібник. Київ: Університет «КРОК», 2022. 320 с.

27. Спірін О. М., Іванова С. М. Цифрова трансформація та інформаційні технології в освіті й суспільстві. Інформаційні технології і засоби навчання. 2023. Т. 95, № 3. С. 1–14. URL: https://journal.iitta.gov.ua/index.php/itlt (дата звернення: 12.05.2026).

28. Гриб'юк О. О. Проєктування інформаційних систем та цифрових сервісів: навчальний посібник. Київ: НПУ ім. М. П. Драгоманова, 2021. 256 с.

29. Литвин В. В., Обельовська К. М. Методи та засоби проєктування інформаційних систем. Львів: Львівська політехніка, 2022. 340 с.

30. Шаховська Н. Б., Литвин В. В. Проєктування інформаційних систем: навчальний посібник. Львів: Новий Світ-2000, 2021. 380 с.

31. Гужва В. М. Інформаційні системи і технології на підприємствах: навчальний посібник. Київ: КНЕУ, 2021. 400 с.

32. Петренко А. І., Буров Є. В. Сучасні вебтехнології та розробка web-застосунків: навчальний посібник. Львів: Львівська політехніка, 2023. 312 с.

33. Биков В. Ю., Спірін О. М., Пінчук О. П. Цифрова трансформація суспільства і розвиток інформаційно-аналітичних систем. Інформаційні технології і засоби навчання. 2022. Т. 89, № 3. С. 1–19. URL: https://journal.iitta.gov.ua/index.php/itlt (дата звернення: 14.05.2026).

34. Кадемія М. Ю., Шахіна І. Ю. Вебдизайн та web-технології: навчальний посібник. Вінниця: ТОВ «Планер», 2021. 210 с.

ЗГОДА здобувача вищої освіти

Державного університету економіки і технологій про перевірку кваліфікаційної роботи на прояви академічного плагіату та розміщення в Репозитарії Університету

Я, **Павлиш Кирило Дмитрович**,

підтримую політику Державного університету економіки і технологій з академічної доброчесності і відкритого доступу.

Засвідчую, що кваліфікаційна бакалаврська робота

Проєктування та розробка інформаційної системи управління спортивним клубом з використанням web-технологій

виконана самостійно та не містить академічного плагіату. Я не надавав і не одержував недозволену допомогу під час підготовки цієї роботи. Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Державного університету економіки і технологій ознайомлений(а). Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення норм академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

Також я поінформований, що відповідно до «Положення про Репозитарій (електронну базу даних) Державного університету економіки і технологій» зазначена робота буде розміщена в Електронному архіві Університету (Репозитарії ДУЕТ). З умовами такого розміщення ознайомлений.

15.06.2026  Павлиш К.Д.