МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії

програмного забезпечення

_____ Євген ДАВИДЕНКО

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБЗАСТОСУНОК АВТОМАТИЗАЦІЇ ІНФОРМУВАННЯ ТА ВЗАЄМОДІЇ З КЛІЄНТАМИ ТРЕНАЖЕРНОГО ЗАЛУ

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Анна НЕЙ**

«__» _____ 20__ р.

Керівник роботи

PhD, старший

викладач _____ **Ігор КАНДИБА**

«__» _____ 20__ р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___» _____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувача

Ней Анни

1. Тема кваліфікаційної роботи вебзастосунок автоматизації інформування та взаємодії з клієнтами тренажерного залу затверджена наказом ректора ЧНУ ім. Петра Могили № 349 від «26» грудня 2025 р.

2. Строк представлення кваліфікаційної роботи «22» _червня__ 2026 р.

3. Очікуваний результат роботи та початкові дані якщо такі потрібні:

Очікуваний результат роботи – вебзастосунок для тренажерного залу. Початкові дані – інформація про діяльність тренажерного залу, дані про користувачів, розклад занять, а також обрані технології та інструменти розробки.

4. Перелік питань, що підлягають розробці:

- дослідження існуючих аналогів предметної області;
- формування вимог до програмного забезпечення;
- проєктування архітектури програмного забезпечення;
- структурування бази даних;
- основний функціонал вебзастосунку;
- аналіз та тестування результатів.

5. Перелік графічних матеріалів:

Слайди презентації.

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «____» _____ 20___ р.

Календарний план виконання кваліфікаційної роботи

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: **Вебзастосунок автоматизації інформування та взаємодії з клієнтами тренажерного залу.**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Підготовка та погодження завдання на виконання КБР	18.10.2025	27.10.2025	Виконано
2.	Огляд літератури за темою роботи	19.01.2026	24.01.2026	Виконано
3.	Розроблення календарного плану КБР	25.01.2026	31.01.2026	Виконано
4.	Аналіз та дослідження предметної області	01.02.2026	10.02.2026	Виконано
5.	Вибір і розробка проєктних рішень	14.02.2026	15.03.2026	Виконано
6.	Моделювання та конструювання ПЗ	17.03.2026	01.04.2026	Виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	03.04.2026	15.05.2026	Виконано
8.	Відгук керівника КБР	29.05.2026	31.05.2026	Виконано
9.	Оформлення пояснювальної записки КБР	16.05.2026	25.05.2026	Виконано
10.	Попередній захист	25.05.2026	26.05.2026	Виконано
11.	Рецензування			
12.	Завершення оформлення КБР та презентації			
13.	Захист кваліфікаційної роботи			

Здобувач _____ **Анна НЕЙ**

«__» _____ 20__ р.

Керівник роботи

PhD, ст. викладач _____ **Ігор КАНДИБА**

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

«Вебзастосунок автоматизації інформування та взаємодії з клієнтами тренажерного залу»

Здобувачка 408 гр.: Ней Анна

Керівник: PhD, ст.викладач Ігор Кандиба

У бакалаврській роботі представлено розробку системи для управлінням записом, взаємодії користувачів із сервісом бронювання і оплати. Актуальність обраної теми полягає у створені вебзастосунку, що дозволить клієнтам доступно і зручно взаємодіяти через онлайн платформу. Відсутність автоматизованих рішень, показує що значна частина дій виконується вручну. Це призводить до помилок, ускладнення комунікації. Саме тому впровадження сучасних вебтехнологій є доцільним і необхідним у діяльності тренажерного залу.

Об’єктом роботи є процес взаємодії між користувачами та адміністрацією тренажерного залу.

Предметом роботи є методи і засоби розробки вебзастосунку інформування та взаємодії адміністрації з клієнтами тренажерного залу.

Метою роботи є розробка вебзастосунку тренажерного залу для спрощення взаємодії між користувачами та адміністрацією закладу.

Бакалаврська робота складається з вступу, чотирьох розділів, висновків, переліку використаних джерел.

Вступі визначено обґрунтування актуальності, об’єкт, мету, завдання та предмет роботи.

У першому розділі проведено аналіз предметної області, сформовано основні напрямки в яких система може стати конкурентною, описано структурні і функціональні особливості об’єкта.

У другому розділі описує обґрунтовування вибору технологій та специфікації вимог програмного забезпечення.

Третьому розділі виконано моделювання архітектури програмного забезпечення, показано взаємодію між компонентами.

Четвертий розділ присвячено опису реалізації інтерфейсу та серверної частини, інтеграцію з сторонніми сервісами та тестування застосунку.

У висновках узагальнено результати проведеної роботи, аспекти подальшого розвитку.

КРБ викладена на 68 сторінки, вона містить 4 розділи, 34 ілюстрацій, 7 таблиць, 19 джерел в переліку посилань.

Ключові слова: інформаційна система, користувач, розклад, бронювання, оплата, адаптивний інтерфейс.

ABSTRACT

of the Bachelor's Thesis

"Web application for automating communication and interaction with gym clients"

Student of group 408: Nei Anna

Supervisor: PhD, senior Lecturer Kandyba Ihor

This bachelor's thesis presents the development of a system for managing reservations and user interactions with a booking and payment service. The relevance of this topic lies in the creation of a web application that will allow customers to interact easily and conveniently via an online platform. The lack of automated solutions indicates that a significant portion of tasks is performed manually. This leads to errors and complicates communication. That is why the implementation of modern web technologies is both appropriate and necessary for the operation of a gym.

The object of this work is the process of interaction between users and the gym administration.

The subject of this work is the methods and means of developing a web application for informing and facilitating interaction between the administration and gym clients.

Purpose of this work is to develop a web application for a gym to simplify interaction between users and the facility's administration.

The bachelor's thesis consists of an introduction, four chapters, conclusions, and a list of references.

The introduction defines the rationale for the relevance, the object, the goal, the objectives, and the subject of the thesis.

The first chapter analyses the subject area, identifies the main areas in which the system can become competitive, and describes the structural and functional features of the object.

The second chapter describes the rationale for the choice of technologies and the software requirements specification.

The third chapter presents a model of software architecture and illustrates the interaction between components.

The fourth section is devoted to the description of the implementation of the interface and the server part, integration with third-party services and testing of the application.

The conclusions summarize the results of the work carried out, aspects of further development.

The qualification work is presented on 68 pages of typewritten text, consists of an introduction, 4 sections, general conclusions, a list of references with 19 titles. The work contains 7 tables and 35 figures.

Keywords: information system, user, schedule, booking, payment, adaptive interface.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 3

ВСТУП 4

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 5

1.1 Розкриття об’єкту та предмету дослідження 5

1.2 Існуючі аналоги програмних рішень 8

1.3 Опис структурних і функціональних особливостей об’єкта 11

Висновки до розділу 1 14

2 ОПИС ТЕХНОЛОГІЙ ТА СПЕЦИФІКАЦІЇ ВИМОГ 15

2.1 Аналіз інструментарію, методів та моделей 15

2.2 Специфікація вимог до ПЗ 22

Висновки до розділу 2 29

3 АРХІТЕКТУРА ТА МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ 30

3.1 Розробка діаграми варіантів використання 30

3.2 Архітектурні рішення 34

3.3 Структура інтерфейсу застосунку 39

Висновки до розділу 3 42

4 ПРОГАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ 43

4.1 Робота з базою даних 43

4.2 Програмна реалізація 45

4.3 Керівництво користувача 53

4.4 Тестування вебзастосунку 56

Висновки до розділу 4 58

ВИСНОВКИ 59

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 60

ПЕРЕЛІК СКОРОЧЕНЬ

БД – База даних

ПЗ – Програмне забезпечення

ПК – Персональний комп'ютер

API – Application Programming Interface

CRM – Customer Relationship Management (управління взаємовідносинами з клієнтами)

EDA – Event-driven systems

HTTPS – HyperText Transfer Protocol Secure

MVC – Model-View-Controller (Модель-Вигляд-Контролер)

ORM – Object-Relational Mapping (Об'єктно-реляційне відображення)

REST API – Representational State Transfer інтерфейсу програмування додатків.

SMART – Specific (конкретна), Measurable (вимірювана), Achievable (досяжна), Relevant (значуща) та Time-bound (обмежена в часі).

TSL – Transport Layer Security

UML – Unified Modelling Language

ВСТУП

За умов стрімкого розвитку інформаційних технологій, практично в усіх наявних сферах діяльності відбувається жвава диджиталізація різних підприємств, зокрема й індустрія фітнесу та здорового способу життя. Тренажерні зали активно впроваджують онлайн-сервіси для зручного доступу до послуг, підвищення якості обслуговування клієнтів, та корегуванню внутрішніх процесів.

Актуальність обраної теми полягає у створені вебзастосунку, що дозволить клієнтам доступно і зручно взаємодіяти через онлайн платформу. Відсутність автоматизованих рішень, показує що значна частина дій виконується вручну. Це призводить до помилок, ускладнення комунікації. Саме тому впровадження сучасних вебтехнологій є доцільним і необхідним у діяльності тренажерного залу.

Мета роботи – розробка вебзастосунку тренажерного залу для спрощення взаємодії між користувачами та адміністрацією закладу.

Для досягнення поставленої мети необхідно вирішити наступні **завдання**:

– провести аналіз наявних систем, щоб врахувати помилки і зробити платформу більш професійною у роботі;

– сформувати основні вимоги до функціональності та інтерфейсу;

– визначити архітектуру, враховуючи вибір технологій та рішень для всього проєкту;

– створити структуру бази даних та серверної частини;

– розробка frontend частини;

– протестувати розроблений вебзастосунок.

Об’єкт роботи – процес взаємодії між користувачами та адміністрацією тренажерного залу.

Предмет роботи – методи і засоби розробки вебзастосунку інформування та взаємодії адміністрації з клієнтами тренажерного залу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Розкриття об’єкту та предмету дослідження

У ході виконання кваліфікаційної роботи, об’єктом став процес взаємодії між користувачами та адміністрацією тренажерного залу. Коротко характеризуючи інформаційні процеси обслуговування клієнтів, слід зазначити, що вони охоплюють обробку, зберігання та передачу даних, пов’язаних із наданням послуг.

Враховуючи зростаючу роль інформаційних технологій у сучасному світі, потенційно ця розробка може стати важливим інструментом у підвищенні якості обслуговування, зменшення навантаження на персонал, сприяє популяризації занять спортом. Створює передумови для подальшого розвитку і масштабування бізнес.

Сучасне суспільство дедалі більше орієнтується на зручність, швидкість та технологічні рішення. Користувачі очікують, що більшість дій можна виконати у кілька натискань без зайвих витрат часу та фізичних зусиль. Особливо нові покоління, починаючи від зумерів згідно нових досліджень [1], зараз надають увагу більш закритої комунікації, для них це сприймається як більш контрольована та менш стресова форма взаємодії. Додати сюди людей з соціальною фобією, інтровертів, чи інших людей з тривожністю для них такий формат взаємодії є особливо зручним. Якщо брати інші вікові категорії, то багато користувачів, вже адаптувались і звикли до легкого інформаційного середовища, і бажають швидко вирішувати всі питання. Згідно графіку досліджень (див. рис. 1.1) в Україні, найбільше споживачів у фітнес-центрах віком 18-45 років. До 17 років теж доволі велика частка аж 13% від загальної кількості. Основна цільова аудиторія, як чоловіки так і жінки. Водночас споживачі віком понад 45 років від загальної аудиторії, складають 16%. Це свідчить, що орієнтир ринку направлений на переважно активну працездатну аудиторію. Зараз в Україні функціонує понад 1500 фітнес-об’єктів, і спостерігається укрупнення бізнесу [2].

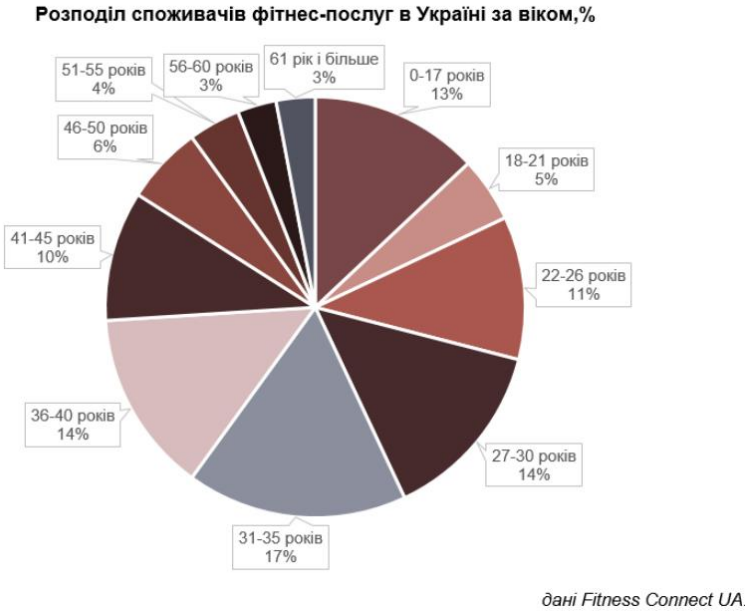

Рисунок 1.1 – Вікова категорія ринку фітнес-послуг [2]

Впровадження технологій стрімко трансформує сферу послуг, знижуючи бар’єри входу та оптимізуючи бізнес-процеси. Основне вирішення проблем підприємства включає автоматизацію рутини, використання хмарним сервісів. У більшості випадків це підвищує конкурентоспроможність, фінансові показники, пришвидшує комунікацію між сторонами.

Розглядаючи застосунок, згідно сучасним емпіричним дослідженням [3], впровадження цифрових технологій, позитивно впливає на продуктивність підприємств, а особливо з кожним роком всі сфери все більше адаптуються під нові технології (див. рис. 1.2).

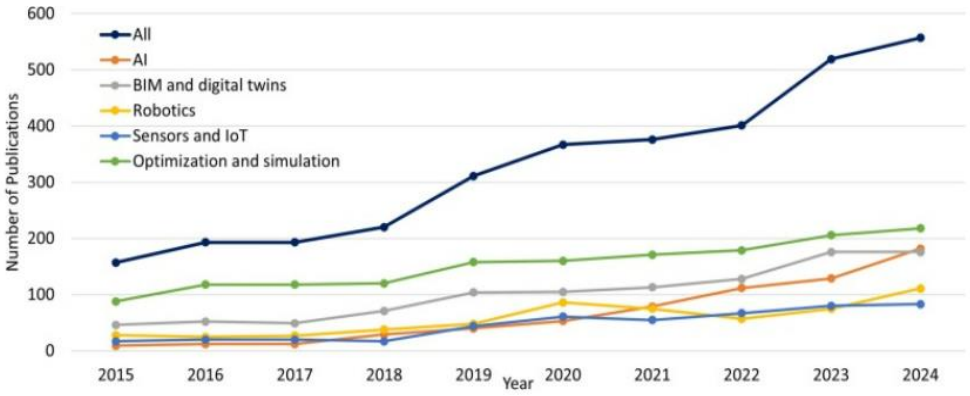

Рисунок 1.2 – Графік впровадження технологічних рішень [3]

Якщо розглядати аспект предметної області, то серед основних підходів виділяється використання методів, і різних систем сповіщень. Серед методів інформування окреме місце займають системи автоматичних сповіщень, які працюють на основі подій (event-driven systems). Вони забезпечують надсилання повідомлень у момент настання певної події, наприклад запису на тренування, зміни часу заняття або підтвердження оплати. Це зменшує навантаження на персонал, зменшує ризик помилок. У роботі використовується підхід, подібний до CRM-системи (див. рис. 1.3), які дозволяють централізовано керувати базою клієнтів, і впорядковувати бізнес-процеси.

Рисунок 1.3 – Процеси, які упорядковуються під час впровадження

На відміну від універсальним CRM, вебзастосунок буде створений під конкретні потреби організації, тому орієнтуватись саме на специфіку тренажерного залу.

Дуже важливо аби головні цілі буди реалістичні та доступні. Цілі мають базуватись не лише на потребах бізнесу, але й на аналітиці ринку наявних продуктів, поведінці аудиторії. Варто визначити які звички має кінцевий користувач. Це допомагає сформувати релевантну пропозицію. Згідно критерія SMART [4] сформовано завдання проєкту, ця абревіатура критеріїв розшифровується як конкретний, вимірюваний, досяжний, релевантний, обмежений у часі. Постановка цілей складається з 5 критеріїв:

1) розробити вебзастосунок з базовим функціоналом, мінімальний рушій (реєстрація, авторизація, перегляд розкладу, запис на тренування);

2) реалізувати не менше 4 ключових функцій системи, що дає змогу користувачам взаємодіяти з адміністрацією спортзалу;

3) досягти стабільної роботи без критичних помилок під час тестування;

4) користувач має швидко отримувати доступ до основних функцій, тому створення відповідного інтерфейсу;

5) завершити розробку, тестування до захисту кваліфікаційної роботи.

Додатково визначено основні сутності, які будуть використовуватись у системі. До них належить користувачі, тренери, розклад занять, абонементи та записи на тренування. Створення ролей у системі є необхідним. Користувач має отримувати доступ до перегляду розкладу та інших послуг застосунку, тоді як менеджер здійснює керування розкладом клієнтами та послугами.

1.2 Існуючі аналоги програмних рішень

На сучасному ринку існують аналоги системи, кожне з яких має свої особливості. У більшості випадків сфера спорту, та мереж фітнес клубів має погані автоматизовані системи обліку клієнтів. Хоча є і конструктори сайтів, що дозволяють швидко створити сторінку з розкладом і формою запису, проте мають обмежену функціональність і слабку адаптацію під конкретні бізнес-процеси. Також існують окремі рішення для онлайн-запису або вибору тренера, але вони не покривають усі потреби в одному продукті. Тобто, більшість аналогів або вузькоспеціалізовані, або не покривають повністю всі потреби, які має користувач.

Аналог №1. Worldgym (табл. 1.1) – американський фітнес-центр, який вже 50 років на світовому ринку. Цей спортзал є франшизою і відкритий у багатьох містах світу. Центральний сайт зручний бо можна глянути де саме поблизу відкритий зал.

Таблиця 1.1 – Характеристика аналогу Worldgym

Ознака	**Характеристики**
Архітектура	Web, iOS, Android, трирівнева архітектура сайту.
Мова реалізації	Php, JavaScript framework Vue.js та Nuxt.js.
Основні функції	– подана основна інформація про спортзал. – можливість переглянути найближчі до користувацького розташування спортзали. – пояснення кожної дисципліни, яка можлива у обраному спортзалі. – якщо користувач хоче відкрити самостійно спортзал цієї мережі надано форму для заповнення. – свій мерч магазину.
Переваги	– швидкий пошук доступних спортзалів будь якій точці світу. – простий та оптимізований інтерфейс – досить велике ком’юніті.
Недоліки	– носить характер інформаційного формату, немає функцій які б клієнт зміг би для себе забронювати – неможливість швидкісної оцінки ціни та рейтингу спортзалу. – не всі вебсайти робочі і ведуть до реальних інформаційних джерел.

Аналог №2. Gym-smile (табл. 1.2) – це мережа фінтес-центрів в Одесі, що пропонує сучасні тренажерні зали, кардіозони, персональні тренування. Вона вміщує в себе цілу мережу залів, розташованих по всьому місту.

Таблиця 1.2 – Характеристика аналогу Gym-smile

Ознаки	**Характеристики**
Архітектура	Ієрархічна структура, Клієнт-сервіс, мобільна версія
Мова реалізації	Vue.js, Next.js
Основні функції	– перегляд розкладу всіх підпорядкованих спортзалів. – можливість поставити менеджеру запитання. – доступне оформлення відвідання басейну. – інформування про знижки та івенти, доступні контакти зв'язку. – перегляд зовнішнього вигляду кожного клубу та його розташування
Переваги	Для кожної тренажерної зали мережі, виділена окрема сторінка. Доступний не тільки похід в тренувальну залу, а також відвідування басейну (в якому теж є свої тренувальні програми).
Недоліки	– можливе оформлення тільки через зв'язок з менеджером по телефону. Відсутність тренерської бази для перегляду. – погана адаптивність до різних розмірів екрану. – неможливість заплатити та вибрати онлайн абонементи чи тренування.

Аналог №3. Junglegym (табл. 1.3) – це сучасний спортивний зал, розташований у центрі Києва. Також, крім їхньої основної сторінки існує однойменний застосунок для фітнес-професіоналів у Google Play. Із списку систем, що аналізуються саме цей аналог є найбільш потужним, і з цікавими розробками. Саме на нього при розробці варто опиратись і врахувати цікаві функції які наявні, а також розширення системних функцій.

Таблиця 1.3 – Характеристики аналогу Junglegym

Ознаки	Характеристики
Архітектура	Клієнт-серверна (веб-додаток). Доступний застосунок для Androis та IOS.
Мова реалізації	Next.js, JavaScript(React, Redux, Emotion).
Основні функції	– можливість онлайн замовити персональні тренування. – перегляд розкладу занять. – оформлення абонементу. – персональний профіль користувача. – перегляд тренерів
Переваги	– достатньо великий функціонал можливих взаємодій, а також динамічний та лаконічний інтерфейс. – багато можливостей оформлення онлайн (наприклад абонементу, реєстрування користувача). – вказано не тільки текстом розташування, а ще й додано карту для зручності.
Недоліки	– відсутня інформація про внутрішній інтерфейс спортзалу. – неповна навігаційна панель, функції які присутні у сайту не відображені для швидкого доступу

1.3 Опис структурних і функціональних особливостей об’єкта

Вебзастосунок призначений для автоматизації інформування та покращення комунікації з клієнтами тренажерного залу.

Основні функції, системи що проєктується:

1) оформлення онлайн абонементів;

2) навігаційна система;

3) реєстрація та автоматизація користувача;

4) редагування профілю;

5) установка ролей користування сайтом;

6) інформаційна платформа;

7) бронювання тренувань, їхнє скасування;

8) перегляд розкладу занять;

9) історія оплати і рахунків;

10) купівля онлайн обраних занять чи підписок;

11) сповіщення на пошту чи телефон, а саме нагадування про тренування

12) профілі тренерів, показаний досвід чи рейтинг;

13) контакти з менеджером для запитань;

14) карта з місцем розташуванням;

15) управління для менеджера профілями користувачів, розкладом чи платежем;

16) особистий кабінет, із можливістю встановлювати цілі;

17) самостійне оформлення членства клубу;

18) картка абонементу;

19) аналітика відвідуваності;

20) адаптивний інтерфейс для різних платформ та гаджетів.

Цей проєкт передбачає розробку вебсайту для спортзалу, який включатиме ключові сторінки: головну, розклад занять, інформацію про тренерів, тарифи та особистий кабінет користувача. Кабінет користувача матиме важливу значимість у розробці, оскільки міститиме не тільки головні сторінки, але і конкуретну розробку функції, це графік відвідуваності.

Проведено аналіз системи, що буде проєктуватись з точки зору основних задач, засобів програмного та апаратного забезпечення, які вихідні дані буде отримано.

Таблиця 1.4 – Аналіз системи, що проєктується

Основні задачі	1)підключення Google maps; 2) оформлення онлайн абонементів; 4) реєстрація та автоматизація користувача; 5) редагування профілю; 6) аналітика відвідуваності.
Користувачі системи	Менеджер, Користувач, Гість
Опис структури	База даних – MySQL. Бекенд частина – Laravel. Фронтенд частина – Next.js
Сценарії роботи системи	1) Користувач обирає спосіб оплати(готівка, картка, онлайн), після оплати стає активним. Система створює запис в БД. 2) Користувач входить на сайт, обирає спосіб авторизації. Успішно вводить свої дані. 3) Клієнт заходить у розділ "Розклад". Може фільтрувати за типом тренування, обирати тренера. 4) Користувач відкриває сторінку персональних цілей. Встановлює ціль, дату завершення. 5) Користувач відкриває власний профіль бачить статистику відвідування.
Засоби програмної та апаратної реалізації	Програмна частина: visual studio, xampp, phpMyAdmin. Апаратна частина:Windows, потужний процесор, стабільне мережеве підключення.
Вихідні дані	Список користувачів(клієнтів, тренерів). Розклад та запис тренувань. Інформація про абонементи. Сповіщення для користувача.

Система також реалізовує обмеження доступу до окремих функцій залежно від статусу користувача, незареєстрований відвідувач, клієнт та менеджер.

Функціонал вебсайту охоплює можливість реєстрації та авторизації користувачів, перегляд актуального розкладу тренувань, онлайн-запис на заняття, а також адміністрування контенту через спеціальну панель управління. Адміністратор може редагувати розклад занять, додавати або змінювати інформацію про тренерів і керувати списком клієнтів.

Висновки до розділу 1

У першому розділі здійснено аналітичний аналіз обраної предметної області, яка стосується фітнес-систем. Досліджено предметну область, що включає актуальні вікові та гендерні особливості.

Здійснено порівняльний аналіз, передових аналогів. Враховано їхній функціонал, архітектуру, переваги та недоліки які можна використати на противагу розробці що проєктується. У ході огляду програмних рішень у даній сфері встановлено, що більшість популярних платформ вузькоспеціалізована. Зазвичай спостерігається обмежена функціональність і слабка адаптація під конкретні бізнес-процеси. Також існують окремі рішення для онлайн-запису або вибору тренера, але вони не покривають усі потреби в одному продукті.

Описано, структурні та функціональні особливості об’єкта. Яка подальша структура вебзастосунку повинна бути, її основні сторони. Функціональні можливості описують процеси, які користувач може виконувати, з урахуванням недоліків конкурентів.

Отже, на основі проведеного аналізу визначено наступні кроки для подальших етапів кваліфікаційної роботи. Вибрано напрям найкращих рішень та поєднання вже існуючих функцій з врахуванням недоліків, щоб створити зручність та функціональність у одному продукті.

2 ОПИС ТЕХНОЛОГІЙ ТА СПЕЦИФІКАЦІЇ ВИМОГ

2.1 Аналіз інструментарію, методів та моделей

У роботі [6] авторами описано переваги використання багаторівневої моделі при створені веборієнтованих застосунків. Розробка вебзастосунку поділяється на етапи, кожен з яких реалізує відповідні методи та підходи реалізації системи. У сучасній веброзробці використовують багаторівневу архітектуру (див. рис. 2.1). Взаємодія між рівнями здійснюється через чітко визначені інтерфейси та протоколи обміну даних.

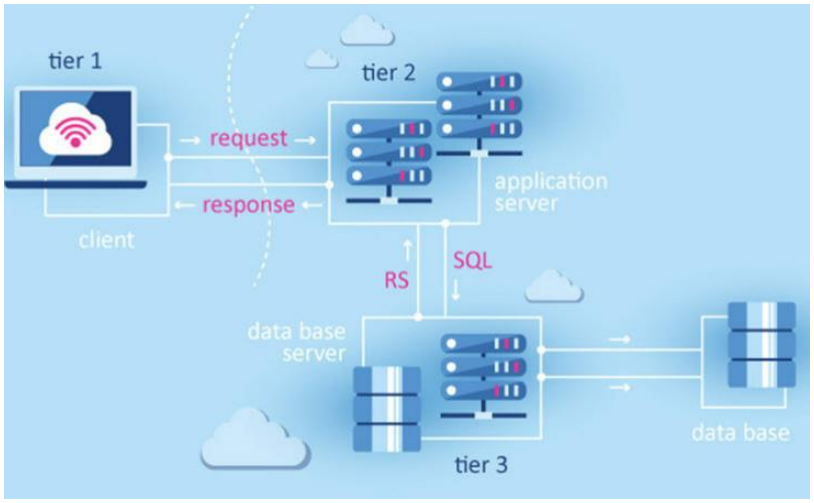

Рисунок 2.1 – Принцип роботи багаторівневої архітектури [7]

Перший шар клієнтська частина, це те що бачить користувач у своєму браузері, тобто кнопки, форми, картинки, інтерфейс. Користувач формує запит, який передається до серверної частини за допомогою мережевого протоколу HTTPS, який забезпечує захищену передачу за допомогою шифрування TSL. Обмін даними відбується у форматі JSON через REST API.

Згідно порівняльного аналізу фронтенд-фреймворків для веб-застосунків [886]. Найпопулярніше технологічне рішення у розробці цього шару, зазвичай обирають React. Це потужний фреймворк, він зручний для динамічних систем, бо у проєктах які не є навчальними, звичайного JavaScript недостатньо, і він зупиняє масштабування, і код стає надто хаотичним. Також

використовують Vue.js та Next.js який є логічним продовженням фреймворку React, він просто додає готову інфраструктуру (табл. 2.1).

Таблиця 2.1 – Порівняння технологій до реалізації frontend частини

Критерії	React	Vue.js	Next.js
Складність розробки	Середня	Низька	Висока
Масштабованість	Висока	Висока	Дуже висока
Робота з динамічними даними	Зручна	Зручна	Дуже зручна
Архітектура	Компонентна	Компонентна	Компонентна і серверна
Підтримка SEO	Середня	Середня	Висока

У результаті аналізу, для розробки сайту для тренажерного залу варто обрати Next.js. Оскільки застосунок несе не просто навчальний характер, а являється реальною масштабованою системою, хоча інші фреймворки є досить потужними щоб реалізувати задумане.

Згідно з тенденціями NPM [9], React, Next.js та Vue.js є найбільш популярними JavaScript-фреймворками, хоча боротьба продовжиться у 2026 році (див. рис. 2.2).

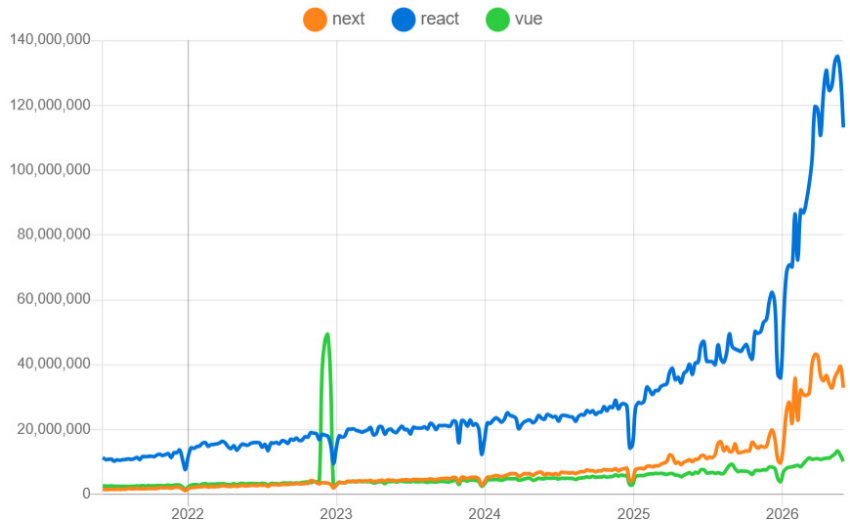

Рисунок 2.2 – Тенденції по завантаженню пакетів

Якщо оцінити завантаження всіх трьох фреймворків веб-розробки за останні 5 років, то, безсумнівно, React лідирує, тоді як завантаження Next.js є частішим, ніж Vue.js.

Шар логіки серверу приймає запити від клієнта, обробляє запити та у процесі взаємодіє з базою даних. Внутрішній обмін відбувається через ORM-рівень, який формує запити до БД. Тобто взаємодія виглядає так що сервер формує запити та передає їх до бази даних через драйвери з використанням відповідного протоколу, після чого отримує результат і повертає назад клієнту уже з сформованою відповіддю. Щодо технологій які обираються на цьому шару, для навчальних задач обирається чистий Node.js. У великих проєктах це швидко може перетворитись хаос, тому тут варто обрати фреймворки. Найпопулярніший мінімалістичний фреймворк з простим API, це Express.js. Його мінуси немає готових шаблонів архітектури, яку треба будувати самостійно. Також для серверної частини є фреймворк NestJS, він потужніший але складніший поріг входу для програміста.

Таблиця 2.2 – Порівняння технологій до реалізації backend частини

Критерії	Express.js	NestJS	Laravel
Складність розробки	Низька	Висока	Середня
Масштабованість	Середня	Дуже висока	Висока
Вбудовані можливості	Мінімальні	Багато	Багато(ORM, auth, міграції)
Тип	Мінімалістичний фреймворк	Повноцінний архітектурний фреймворк	Full-stack фреймфорк
REST API	Легка реалізація	Вбудована підтримка	Підтримується

Laravel дуже популярний у веброзробці, є готова ORM, проста робота з БД і авторизацією. З мінусів його застосування може спостерігатись нижча проективність на деяких сценаріях. У результаті аналізу серверних технологій, видно що фреймворки логічної частини всі дуже потужні. Різниця у їх обранні не є суттєвою, тому варто обрати найбільш зрозумілий і комфортний у написанні, це є Laravel.

Основна відмінність фреймворків від бібліотек полягає в управлінні потоком виконання. Бібліотеки, які важливо використати при розробці FullCalendar. Вона застосовується для відображення розкладу тренувань дозволяє інтерактивно відображувати події у вигляді календаря, підтримує різні режими перегляду, може працювати динамічно.

Що ж стосується БД системи, в яких зберігаються абсолютна більшість даних, вона є ключовим компонентом. У випадку розробки застосунку вебсайту тренажерного залу, в ній буде зберігатись розклад тренувань, тренери, записи на заняття, дані користувачів, тощо.

Для реалізації найчастіше використовується реляційна база даних. Вона забезпечує цілісність даних, підтримку зв’язків між таблицями. Нереляційні бази даних використовуються рідше, і не підходить для досліджуваної системи бо вона має чітку структуру даних. Такий тип даних підходить найкраще на випадок зберігання гнучких або неструктурованих даних. Найпоширенішою і простою, яка добре інтегруються з серверною частиною являється MySQL, також існують інші PostgreSQL, MongoDB. Для надійного зберігання у проєкті найбільш доцільним буде використання реляційної бази даних.

З інструментів для доступу до баз даних, використовується середовище XAMPP Control Panel. Він дозволяє локально розгорнути серверне середовище, яке включає вебсервер Apache та систему керування базами даних MySQL. Це забезпечує можливість тестування без необхідності використання віддаленого хостингу.

Під час розробки, для комунікації сервісів і сховища застосовується написання SQL-запитів. Завдяки потужному інструменту Eloquent ORM

(Object-Relational Mapping) у Laravel, розробники можуть уникати написання складних запитів, це значно прискорює розробку. Тому є надзвичайно важливим інструментом у проєктуванні системи кваліфікаційної роботи.

Архітектура взаємодії компонентів, побудованого з використанням шаблону MVC та ORM-технології (див. рис. 2.3). Моделі, являються основним інструментом взаємодії з даними, та представляють структуру таблиць бази даних у вигляді об’єктів. При виконанні операцій отримання, додавання та інших модель надсилає запит до ORM, яка виступає посередником між логікою програми та реляційною БД. Eloquent ORM автоматично перетворює запити, сформовані у вигляді методів мови програмування, у відповідні SQL-запити. База даних виконує отримані запити та повертає результат назад через підключення, наприклад вибірка записів чи підтвердження змін повертається та обробляється у вигляді об’єктів.

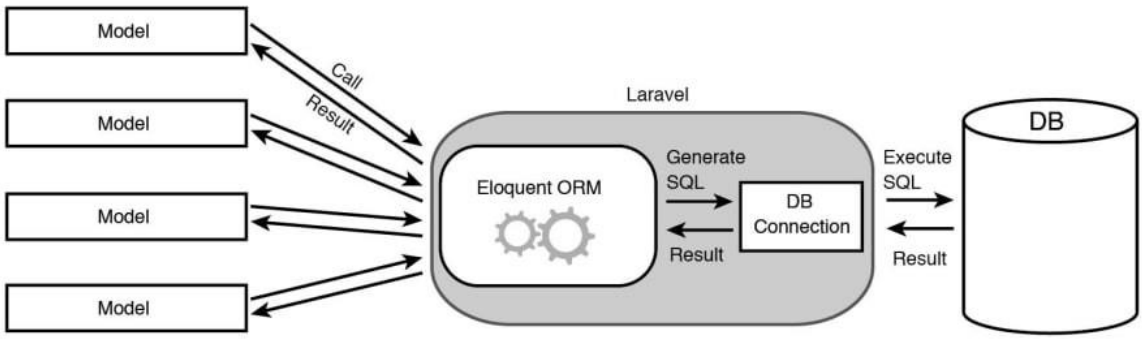

Рисунок 2.3 – Eloquent ORM. Операції з базою даних Laravel [10]

Застосування ORM-технології значно спрощує розробку, та взаємодію з базою даних. Це зменшить кількість помилок при написанні SQL-запитів, а отже підвищить швидкість розробки вебзастосунку.

Безпечна оплата, є основним показником успішності виконання кваліфікаційної роботи. Оскільки передбачає обробку конфіденційних фінансових даних користувачів. Основними технологіями безпеки є TSL шифрування, усі дані передаються за допомогою протоколу HTTPS, це є одним із гарантів унеможливлювано перехоплення третіми особами. Для

роботи з платіжними картками використовується стандарт безпеки PCI DSS, який визначає вимоги до захисту даних власників карток, це стосується саме організацій які зберігають дані клієнтів.

Зовнішнім сервісом, який буде забезпечувати безпеку даних, та оплату абонементів обрано LiqPay. Платіжна платформа, яка забезпечує інтернет-еквайринг та приймання платежів на вебсайтах, у мобільних додатках і інших онлайн-ресурсах [11]. Використання платіжних шлюзів, таких як LiqPay та Stripe дає змогу оброблювати транзакції без збереження платіжної інформації на стороні сайту. Українська платіжна платформа що має відкритий API, дає можливість тестових ключів для налаштування правильності сервісу перед підключенням. Безпека передавання даних, автоматичне списання коштів, переказ на рахунок бізнесу.

У роботі [12] авторами проведено огляд дослідження присвяченого платіжним системам та їх підключення. Розглянуто основні підходи та вимоги до безпеки. Аналіз наведених у роботі підходів став підґрунтям для вибору платіжної системи LiqPay у межах даної розробки. Інтеграція зовнішньої системи реалізується за схемою серверної взаємодії (див. рис. 2.4).

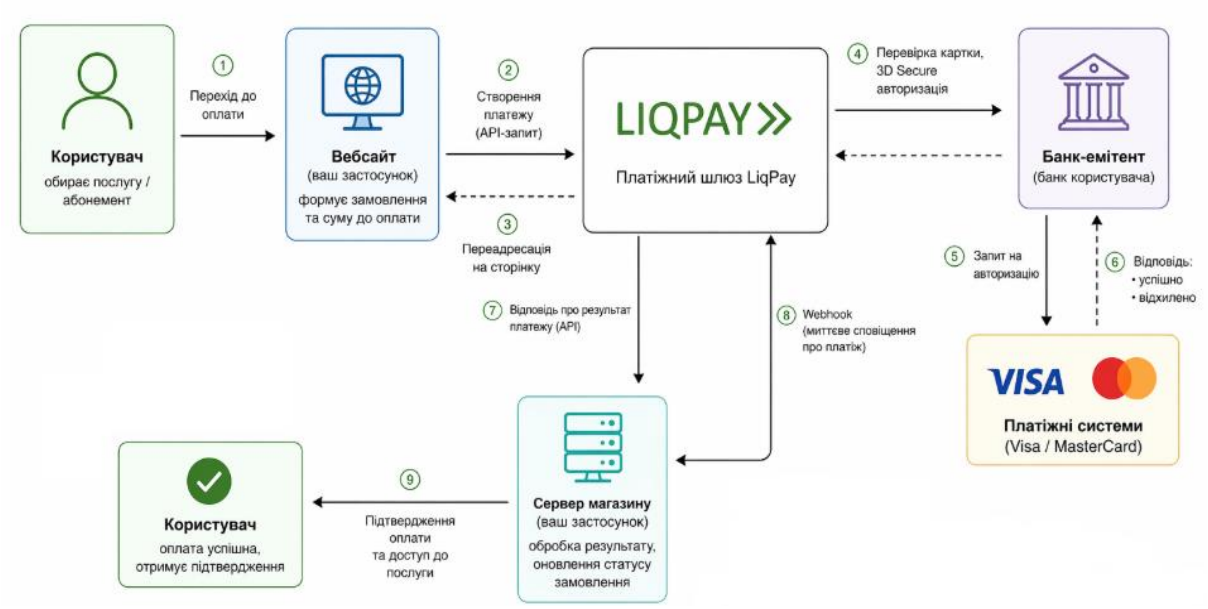

Рисунок 2.4 – Схема обробки платежів через LiqPay

Після ініціалізація платежу, сервер формує платіжний запит та передає його до API LiqPay. Платіжна система надсилає callback-запитна сервер

застосунку, який підтверджує статус операції та оновлює дані у БД. Такий підхід забезпечує безпеку фінансових операцій.

Методи забезпечення безпеки:

– використання унікального ідентифікатора токену, який не містить реальних платіжних даних;

– додатковий рівень перевірки користувача, через підтвердження кодом або банківським застосунком;

– перевірка введених даних на стороні клієнта і сервера, запобіганням помилкам та атакам.

За відображення та швидку розробку стилів відповідає фреймворк Tailwind. Він дозволяє швидше писати та підтримувати код програми. Завдяки тому що фреймворк надає саме службові класи, це надає гнучкість і контроль вигляду [13]. Компілятор збірки уникає попередньої компіляції всіх стилів, і компілює лише за необхідності. Адаптивний дизайн буде швидко створюватись, оскільки у ньому створена вбудована система відступів, шрифтів та кольорів. Важливо щоб застосунок був адаптивний під різний тип пристрою, і зберігав стиль та структуру. Швидка масштабованість у різних проєктах, це дає можливість легкого розширення системи у подальшій розробці без зміни технологій. Тому цей фреймворк є найкращим рішенням для розробки вебзастосунку тренажерного залу.

Для візуалізації аналітичних даних доцільно використовувати бібліотеку Chart.js. Вона дозволяє будувати графіки і діаграми на основі динамічних даних. Дуже популярна, безкоштовна, активно розробляється та підтримується спільнотою, підтримує широкий спектр типів візуалізації. Інтеграцію бібліотеки необхідно здійснювати на стороні клієнта шляхом підготовлених сервером даних.

Щоб реалізувати механізм сповіщень у вебзастосунку варто використовувати вбудовану систему повідомлень фреймворку Laravel Mail. Кожен електронний лист керується за допомогою спеціального класу. А для

керування доступом пакет Sanctum, який використовується для видачі токенів без складнощів авторизації.

2.2 Специфікація вимог до ПЗ

Розробити вебзастосунок для автоматизації інформування та взаємодії з клієнтами тренажерного залу. Система повинна забезпечувати реєстрацію та авторизацію користувачів із розподілом ролей. Клієнти мають мати доступ до особистого кабінету, з можливістю зміни даних свого профілю, перегляду активних абонементів та розкладу занять. Необхідно забезпечити захищений доступ до оплати тренувань та різних типів клубних карт.

Система повинна підтримувати створення різних типів абонементів та контроль терміну їх дій. Тренери повинні мати доступ до списку клієнтів та управлінням розкладу, тарифами. Треба реалізувати систему сповіщень про заплановані тренування чи зміни у розкладі, а також підтвердження реєстрації. Інтерфейс повинен бути інтуїтивно зрозумілим, та адаптивним до різних пристроїв. Користувача має бути власний профіль із можливістю зміни інформації, та встановленням особистих цілей. Відображатись частота відвідувань та успішних занять за весь період користування.

Вебзастосунок повинен легко масштабуватись та передбачити подальше розширення функціоналу. Написання коду треба дотримуватись базових принципів програмування щоб забезпечити легкість розуміння іншим програмістам. Частина серверної та клієнтської частини повинна реалізовуватись на сучасному фреймворку. Головним завданням стає забезпечення безпеки та захист від типових атак, вебзагроз.

1. ***Призначення та межі проєкту:***

1.1 Призначення системи (застосунку), для якої розробляється програмне забезпечення

Вебзастосунок призначений для автоматизації інформування та кращої взаємодії з клієнтами тренажерного залу.

1.2 Погодження, що ухвалені в програмній документації

Для проєктування програмного забезпечення, узгоджено зовнішній фреймворк для написання frontend частини Netx.js, а також для серверної частини обрано фреймворк Laravel .

1.3 Межі проєкту ПЗ

Дата завершення роботи над проєктом – 25.05.2026 р.

2 ***Загальний опис:***

2.1 Сфера застосування

Головною галуззю застосування обраного ПЗ являється сфера спорту, для подання інформування та налагодження комунікації між клієнтами спортивного залу.

2.2 Характеристики користувачів

Користувачу необхідно мати гаджет (смартфон, ноутбук, ПК) зі стабільним доступом до мережі Інтернет.

2.3 Загальна структура і склад системи

Система складається з таких модулів:

– Модуль авторизації та реєстрації користувачів – цей модуль відповідає за авторизацію адміністраторів в адмін-панелі, для внесення необхідних дій на сайті, наприклад додаванні нових облікових записів у якості адміністратора. Також авторизацію самих користувачів зі створенням свого профілю, чи входом уже створений запис.

– Модуль адміністрування – цей модуль відповідає за функціонал адмін-панелі, з якою будуть працювати менеджери застосунку. У менеджерів буде наявний весь доступ до інформації на вебсайті з можливістю її зміни чи додавання.

– Модуль новин та контенту – він відповідає за надання відвідувачам вебсайту можливості переглядати загальну інформацію про тренажерний зал, новинки чи акції. Різні новинки, чи інформацію мають право редагувати менеджери.

– Модуль розкладу – дозволяє користувачу робити запис чи відміну запису, переглядати розклад занять, виступає одним з ключових у системі модулів.

– Модуль абонементів – призначений для автоматичного керуванням контролю термінів дії, можливістю перегляду доступних тарифів чи самостійним оформленням абонементу користувача.

– Модуль тренерів – цей модуль дозволяє переглядати список тренерів та їхніх цінових політик, записуватись та персональні тренування, ознайомитись з їхнім досвідом та результатами робіт.

– Модуль оплати – він буде відповідати за інтеграцію з платіжними системами, легкий доступ оплати онлайн абонементу чи обраного персонального тренування.

Загальна структура системи має наступний вигляд:

1. Модуль авторизації користувачів

– форма логіну та реєстрації для існуючих користувачів;

– форма керуванням нових користувачів менеджером вебсайту в адмін-панелі.

2. Модуль адміністрування

– редактор для додавання нового контенту, сторінок, постів, новин, результатів, активності, акцій;

– вкладка додавання та редагування користувачів системи, створення нових ролей в системі. Туди також будуть входити керування тренерами та заняттями;

– вкладка налаштування теми вебзастосунку.

3. Модуль новин та контенту

– вкладка для публікації новин оновлення інформації про зал;

– ведення контентом;

– редагування та видалення публікацій.

4. Модуль розкладу

– перегляд розкладу занять;

– запис на тренування;

– відміна запису.

5. Модуль тренерів

– перегляд списку тренерів;

– ознайомлення з їх профілями та досвідом;

– можливість записуватися на персональні тренування.

6. Модуль оплати

– інтеграція з платіжними системи;

– онлайн оплата абонементів та занять з тренером;

– історія будь якої оплати у цій системі.

Загальні обмеження

Система орієнтована та конкретний тренажерний зал, частина функцій доступна лише авторизованим користувачам.

3 Функції системи:

3.1 Функція системи оплата абонементу чи обраних тренувань.

3.1.1 Опис функції

Дана функція дозволяє сплачувати абонемент чи обране тренування онлайн.

3.1.2 Вхідна і вихідна інформація

Вхідна інформація – ім’я та прізвище користувача, тип абонементу чи тип тренування, загальна сума.

Вихідна інформація – чек про оплату, активована картка користування.

3.1.3 Функціональні вимоги

– користувач повинен мати можливість переглядати доступні абонементи;

– система повинна отримувати результат транзакції та зберігати всю інформацію про транзакції;

– система повинна повідомляти користувача про етапи оплати;

– менеджер повинен мати можливість керувати статусами, тарифами, переглядати платежі та відміну оплати з поверненням коштів у разі запиту клієнта.

3.2 Функція реєстрація

3.2.1 Опис функції

Створення нового облікового запису користувача в системі, отримання доступу до обмежених функцій вебзастосунку.

3.2.2 Вхідна і вихідна інформація

Вхідна інформація: ім’я та прізвище користувача, його пошта та пароль.

Вихідна інформація: новий обліковий запис та можливість входу до системи.

3.2.3 Функціональні вимоги

– система має перевіряти складність пароля та правильність введення пошти, забезпечувати їхню унікальність;

– користувач повинен заповнити реєстраційну форму;

– користувач отримує підтвердження та можливість входу у систему;

– система має повідомляти про помилки введення та вказувати на причину помилки.

4 ***Вимоги до інформаційного забезпечення:***

4.1 Джерела і зміст вхідної інформації (даних)

Зовнішніми джерелами виступають сервіси повідомлення та платіжних дій. Користувацькі джерела включають реєстраційні дані та дані профілю.

4.2 Нормативно-довідкова інформація (класифікатори, довідники тощо)

В цьому пункті вимоги відсутні.

4.3 Вимоги до способів організації, збереження та ведення інформації

Інформація яка надходить повинна бути достовірною та актуальною, проходити валідацію перед обробкою. Дані зберігаються у реляційній базі даних, забезпечується логічний зв’язок між сутностями.

5 ***Вимоги до технічного забезпечення.***

Для роботи системи, повинна бути стабільна робота програмного продукту, щоб технічний засіб користувача відповідав мінімальним системним вимогам для використання браузеру а також підтримувати обробку, збереження та передачу даних.

6 ***Вимоги до програмного забезпечення.***

6.1 Архітектура програмної системи

Система має сервісну частину управлінням запитів, базу даних а також клієнтська частина.

6.2 Системне програмне забезпечення

Вебзастосунок реалізовується за допомогою фреймворків Laravel та Next.js. В якості бази даних виступає MySQL.

6.3 Мережне програмне забезпечення

Протокол передачі даних HTTPS для захисту і який є сучасним стандартом. З’єднання між клієнтом та сервісом. Використання системи обміну повідомлень та контроль навантаження.

6.4 Програмне забезпечення ведення інформаційної бази

Усі дійства виконуються через MySQL, з використанням інтерфейсу програмованого програмного забезпечення.

6.5 Мова і технологія розробки ПЗ

Використання PHP з фреймворком Laravel та JavaScript з фреймворком Next.js.

7 ***Вимоги до зовнішніх інтерфейсів.***

7.1 Інтерфейс користувача

Інтерфейс має бути максимально комфортним та інтуїтивно зрозумілий для користувача. Кольори підібрані згідно естетики інтерфейсу, та не приносити дискомфорт користувачеві.

7.2 Апаратний інтерфейс

В якості апаратного інтерфейсу має виступати будь який пристрій, який дозволяє робити вихід в мережу Інтернет.

7.3 Програмний інтерфейс

Програмний інтерфейс застосунку забезпечує взаємодію клієнтської частини на Next.js із серверною частиною на Laravel через REST API для обміну даними та керування функціоналом системи.

7.4 Комунікаційний протокол

Методи передачі запитів і відповідей між частинами системи, використання стандартних протоколів HTTPS, формати даних а також механізми обробки помилок.

8 ***Властивості програмного забезпечення.***

8.1 Доступність

Програмне забезпечення повинно забезпечувати цілодобовий доступ користувачів, за винятком періодів проведення технічних робіт або аварій на серверному обладнанні чи у хостинг-провайдера, коректно функціонувати на різних типах пристроїв (ПК, ноутбуки, планшети, смартфони) та надавати доступ до інформації без необхідності встановлення додаткового програмного забезпечення.

8.2 Супроводжуваність

Код та структура застосунку повинна бути інтуїтивно зрозумілим, і дотримуватись стандартів введення, щоб забезпечити подальший супровід іншими розробниками.

8.3Переносимість

Програмне забезпечення має бути переміщуваним та допускати розгортання на іншому сервері або хостингу без істотних змін у кодовій базі. Переносимість досягається завдяки, використанню стандартного серверного стеку (PHP, MySQL), наявності засобів експорту та імпорту бази даних і файлів застосунку, незалежності від конкретної операційної системи сервера.

8.4 Продуктивність

Коректна робота при великій кількості користувачів і їхньому одночасному доступі. Швидка та стабільна робота системи.

8.5 Надійність

Введення регулярного резервного копіювання бази даних та фалів вебзастосунку. Перевірка помилок та захист даних сучасними стандартами. У випадку збоїв система повинна швидко корегувати та відновлювати працездатність.

8.6 Безпека

Регулювання несанкціонованого доступу, використання міжнародних стандартів безпеки сайтів. Захист учасників шляхом розділенням ролей користувачів.

9 *Інші вимоги.*

Простота встановлення та подальшого оновлення системи нових версій. Документування коду та коментарі для подальшого супроводу іншими розробниками. Адаптивний дизайн для різних пристроїв.

Висновки до розділу 2

У другому розділі проведено аналіз сучасних методів і технологій веброзробки, які будуть використовуватись для створення об’єкту дослідження. Розглянуто принцип роботи багаторівневої архітектури, що передбачає розподіл системи на рівні клієнтської взаємодії, сторону сервера тобто логіки проєкту, та робота з даними. Для цього визначено різні інструменти реалізації. Проаналізовано можливості сучасних вебтехнологій для створення інтерфейсу користувача, з ефективною обробкою запитів та взаємодією бази даних.

Розглянуто інтеграцію платіжних сервісів. Розписана та візуально показана схема обробки платежів через LiqPay, яка забезпечуватиме безпеку платіжних даних за допомогою перенаправлення користувача, з отриманням результату транзакції. Це забезпечить безпеку і дозволяє уникнути зберігання конфіденційних платіжних даних на стороні вебзастосунку. Визначено протоколи та різні механізми захисту персональних даних користувачів.

Отримані результати аналізу, стануть основою для подальшого проєктування системи, його основним вибором стеку технологій.

3 АРХІТЕКТУРА ТА МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ

Важливо розуміти майбутній функціонал який буде реалізовано, і як взаємодіють між собою складові проєкту, так і окремі функції. Існує багато різних способів для відображення структури застосунку, вони допомагають уявити структуру, логіку роботи і зв’язок між частинами компонентів системи.

Інструментом програмного забезпечення, для опису роботи є візуальне моделювання нотації UML. Це процес поступового переходу між рівнями моделювання, від абстрактної концепції до її логічного представлення, а потім до фізичної моделі відповідної програмної системи. Спеціальна система графічних позначень, яку використовують для опису певного ПЗ. Завдяки тому що UML складається з різних типів діаграм, які допомагають не просто описати ідеї, а є помічником у проєктуванні системи ще до написання коду. Такий стандарт дозволяє інженерам розуміти концепцію не заглиблюючись у фактичний код продукту, це не мова програмування, а набір правил і стандартів для створення.

3.1 Розробка діаграми варіантів використання

Діаграма прецедентів візуально представляє типи ролей і їх діяльність у системі вебзастосунку спортзалу, без дотримання порядку послідовності виконання кроків. З варіантів опису є не тільки візуальне представлення, а ще і текстове.

Діаграма варіантів використання показує, що робить система і як актори нею користуються, але вона не показує роботи всередині застосунку. Вона показує трьох основних акторів системи: користувача, тренера та адміністратора. Звичайний зв’язок між актором та прецедентом, показаний суцільною лінією і має назву асоціація. Будь-який актор повинен бути пов’язаний принаймні з одним варіантом використання. Найбільшу кількість прецедентів має користувач.

Згідно до ролі, кожен з них має певний рівень доступу. Тренер буде мати ті ж самі функції що і користувач тільки з можливістю редагувати розклад, це

показано зв’язком generalization. Зв’язок generalization – генералізація (успадкування, узагальнення) представляє собою батьківські-дочірні взаємозв’язки. Генералізація позначається суцільною лінією з трикутним не зафарбованим вказівником, що вказує на прецедент-предок [14].

Основний функціонал, а отже і ключовий актор системи це авторизований користувач. Редагування власного профілю, бронювання заняття, покупка онлайн, аналітика відвідуваності, перегляд розкладу. Гість має обмежений доступ до функціоналу вебсайту, але після реєстрації він значно може розширити свої можливості.

Адміністратор має розширені права доступу, він відповідає за коректну підтримку роботи вебсайту. Його функціональні можливості є керування контентом, ведення журналу подій та публікація акція, керування абонементами. Змодельовано базовий потік подій, та показано поведінку системи (див. рис. 3.1) .

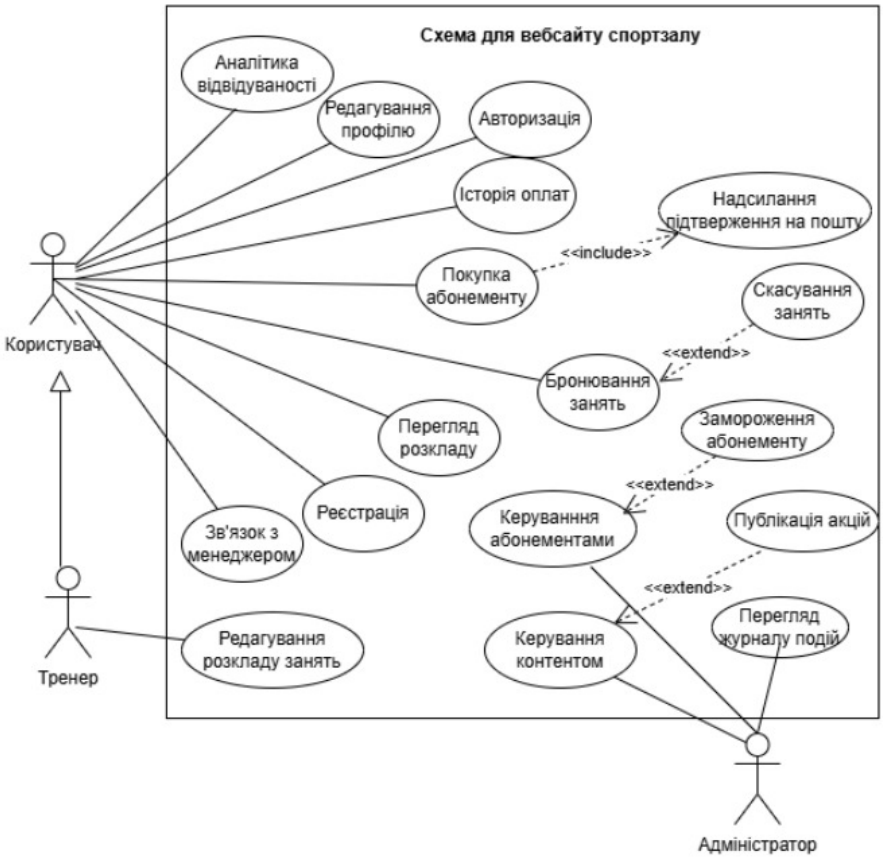

Рисунок 3.1 – Діаграма використання

Дотримано важливу умову, що діаграма використання розроблялась з точки зору користувача. Тобто у сценаріях використовувались назви елементів управління видимі користувачеві, без зображення деталей реалізації незрозумілі замовнику. Зв’язок на діаграмі include застосовується для відображення обов’язкових функцій, тобто поведінка одного прецеденту включається як складовий компонент у послідовності виконання роботи іншого. Без цієї дії головний сценарій такий як наприклад, покупка абонементу не завершиться поки не буде надіслано підтвердження на пошту. Зв’язок extend використовується для демонстрації додаткових або розширених функцій. Без нього система існувала але він дає додаткові можливості використання.

Коротка версія використання системи

User Case 1. Реєстрація користувача

Користувач заходить на сайт спортзалу, відкриває сторінку реєстрації, вводить свої особисті дані (ПІБ, email, пароль). Підтверджує реєстрацію через email, після чого отримує доступ до особистого кабінету.

Поверхнева версія використання системи

User Case 2: Бронювання тренування онлайн

Головний успішний сценарій: Користувач, який вже попередньо зареєстрований у системі виконує авторизацію, відкриває розклад тренувань. Він обирає цікавий йому вид спорту, переглядає доступність місць і бронює участь, натиснувши відповідну кнопку. Система перевіряє доступність бронювання й додає запис до особистого кабінету користувача.

Альтернативні сценарії:

1) Якщо місць немає, система повідомляє про відсутність та пропонує доступні дати чи інші тренування.

2) Користувач намагається забронювати два тренування на один і той самий час, система у свою чергу видає попередження про конфлікт у розкладі.

3) При бронюванні місця система перевіряє статус користувача, якщо абонемент на відвідування тренажерного залу закінчений, видає відповідне

повідомлення, що краще поновити зараз або по приходу, бо заняття не зможе відбутись.

4) Користувач хоче змінити час чи дату бронювання, система дозволяє це зробити .

5) Користувач передумав йти на тренування, яке заброньовано, натискає відмінити система якщо успішно видаляє користувача з бази даних повідомляє про це.

Повна версія використання системи

User Case 3. Оплата абонементу(онлайн)

Таблиця 3.8 – User Case оплата абонементу(онлайн)

Usecase section	Comment
Use Case Name	Оплата абонементу(онлайн)
Scope	Сайт спортзалу
Level	Мета користувача (user-goal)
Primary Actor	Користувач
Stakeholders and interests	1) Користувач: Зацікавлений у швидкій оплаті та отримання своєї абонементної картки; 2) Адміністратор: Зацікавлений тому щоб в користувача не було проблем із доступом або якихось неочікуваних проблем на сайті; 3)Компанія яка надає послуги: Зацікавлена в якісному обслуговувані, та отримані вигоди.
Preconditions	Клієнт авторизований у системі. Користувач обрав спосіб оплата онлайн. У клієнта є банківська картка.
Success guarantee	У користувача з’явилась у обліковому записі абонемент картка.
Main Success Scenario	1) Користувач переходить до розділу "Абонементи" 2) Обирає тип абонементу (місячний, річний, тощо). 3) Обирає тип занять, за які здійснюється оплата (спорзал, фітнес зал). 4) Натискає кнопку "Оплатити" 5) Система перенаправляє на сторінку оплати 6) Користувач вводить дані платіжної картки 7) Платіжна система обробляє запит 8) Успішна оплата – користувач отримує підтвердження на email та в особистому кабінеті може переглядати абонементну картку

Кінець таблиці 3.8

Extensions	1) Якщо платіж не проходить система виводить повідомлення про помилку та пропонує спробувати ще раз 2) Користувач натискає "Скасувати", відбувається повернення на сторінку абонементів без змін. 4) Невірний формат введення платіжних даних, система підказує як виправити помилку 5) Якщо в користувача недостатньо коштів, система виводить повідомлення та пропонує змінити платіжний засіб 6)Коли у користувача ще досі дійсний абонемент, системою виводиться відповідне сповіщення та пропонує продовжити докупити з можливістю на майбутнє.
Special Requirements	Швидкість доступу до мережі Інтернет клієнту має бути більшою ніж 60 кб/сек. Система повинна забезпечувати конфіденційність та безпеку особистих даних користувача під час введення даних платіжної системи. Інтерфейс редагування даних повинен бути адаптований для користування на різних пристроях, включаючи мобільні телефони та планшети.
Technology and Data Variations List	Технологія Stripe, яка захищає дані за допомогою tokenization. Історія оплат в особистому кабінеті. Пропозиція відправляти чек про оплату на пошту.
Frequency ofOccurrence	40%
Miscellaneous	Чи додати функцію оплати частинами, промокодами? Чи робити функцію автоматичної сплати після закінчення абонементу, так як саме її реалізувати?

3.2 Архітектурні рішення

Для документування архітектури програмного забезпечення, використовується **діаграма класів**. Вона є формою структурних діаграм, дає статистичне представлення структури, тобто визначає що включено до модельованої системи [15]. Окремий об’єкт класу, представляє абстрактний опис множини однорідних об’єктів з однаковими атрибутами, операціями й відносини з об’єктами інших класів. Діаграма класів описує, як працює програма з точки зору об’єктів , відображає класи, типи даних їх зміст та відношення (див. рис. 3.2).

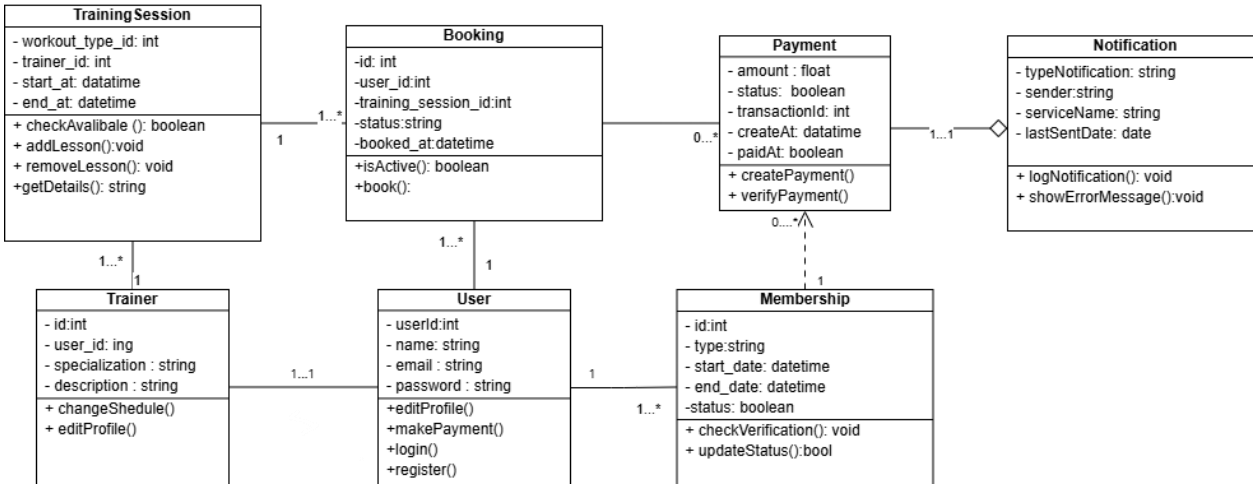

Рисунок 3.2 – Діаграма класів

Компонент користувача показує, що можливо пройти реєстрацію, після чого отримати можливість редагувати свій профіль, оформлювати бронювання та здійснювати оплати. Тренер має функцію управлінням розкладом. Кожна окрема тренувальна сесія прив'язана до одного тренера, однак один тренер може організовувати кілька сесій одночасно.

Користувач має змогу бронювати одну або кілька тренувальних сесій за допомогою сутності Booking. Кожне окреме бронювання завжди пов’язане лише з одним користувачем і однією тренувальною сесією. При створенні бронювання система автоматично генерує платіж, який містить інформацію про суму, статус оплати й дату проведення транзакції. У разі успішної оплати або виникнення помилки система надсилає користувачу повідомлення через сутність Notification. Окрім бронювань, користувачі можуть мати членство Membership, яке визначає період дії їх доступу до самої зали.

Діаграма пакетів являє собою залежності між пакетами, з яких і складається модель. У свою чергу пакет, є елементом моделі, який використовують для групування інших елементів.

Представлено діаграму пакетів для застосунку спортзалу (див. рис. 3.3). Зовнішня система представляє собою рівень, який описує інтеграції з іншими сервісами. У GYM-проєкті, це зв’язок із банком, або взаємодія з поштою, робота з локаціями.

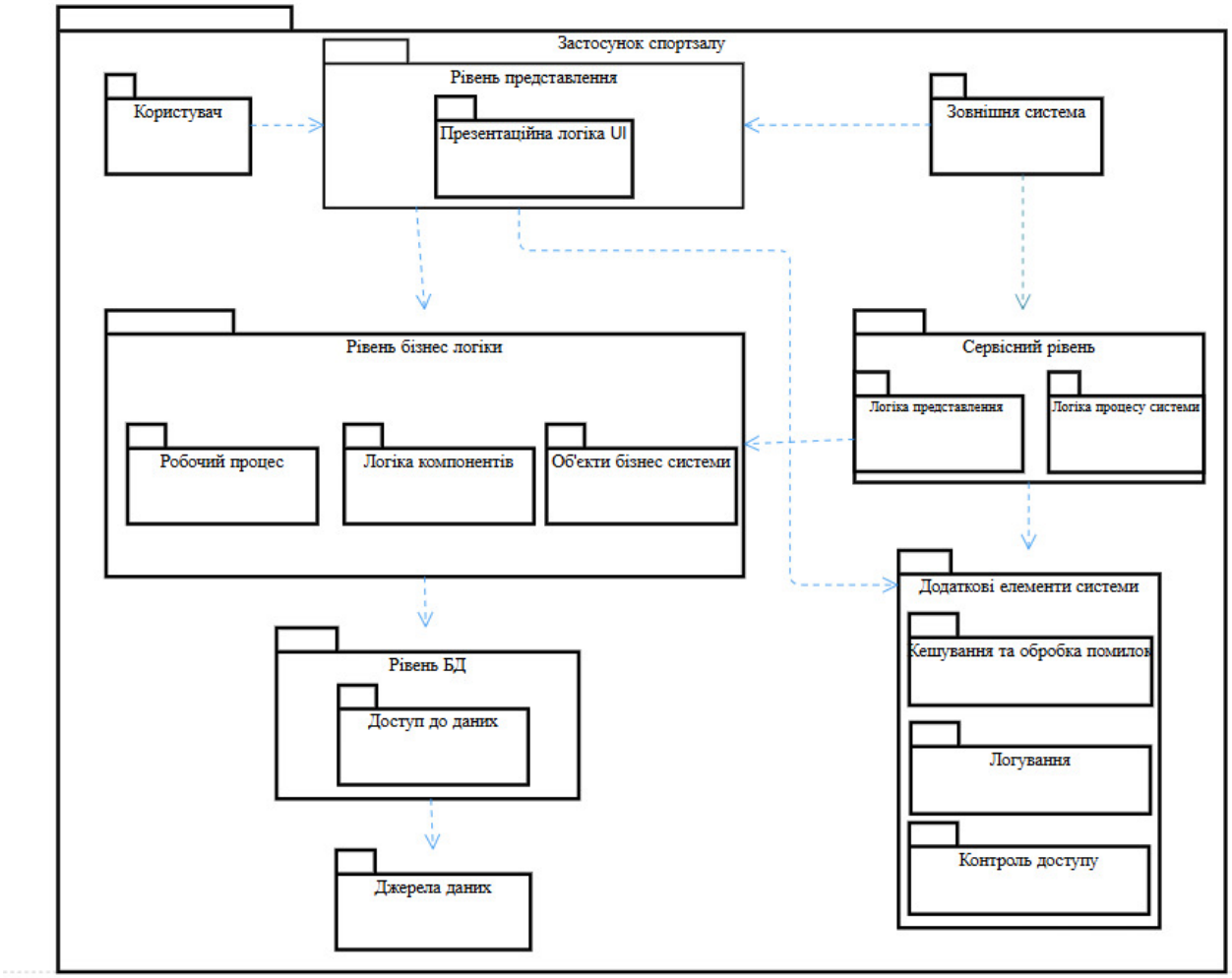

Рисунок 3.3 – Діаграма пакетів

Презентаційна логіка це все, що стосується UI або взаємодії з користувачем. Отже, візуальні елементи, такі як кнопки, меню, шрифти, кольори, іконки та анімації. На рівні бази даних, пакет сервісний агент, він взаємодіє із зовнішніми сервісами або внутрішніми модулями, така собі логіка спілкування. Зовнішній система у застосунку, тут синхронізується API для обробки транзакцій.

На рівні бізнес логіки застосунку об’єкти бізнесу, на яких базується логіка програми, саме сутності бізнесу (абонементи, люди, тренери, ресурси спортзалу). Логікою компонентів сервісів, менеджерів, модулів (перевірити, чи є вільне місце у класі, показати, чи абонемент дійсний, обчислити наступне тренування, визначити вартість знижки). Робочий процес, є реалізація конкретних сценаріїв, які виконує користувач або система, користувач купує абонемент або записується на тренування. Додаткові елементи системи, тут

представленні речі, які проникають через усі шари системи і виконуються в багатьох місцях, але не є бізнес-логікою.

Діаграма переходів (або діаграма станів та переходів), це схема у якій показано множину об’єктів та ймовірні переходи між ними внаслідок певних подій. Така діаграма є корисним інструментом для моделювання поведінки об'єктів у програмному забезпеченні або складних системах, показуючи, як система змінює свій стан, а також які умови спричиняють ці зміни. Зображено діаграму переходів із загальним процесом функціонування вебзастосунку спортзалу (див. рис. 3.4).

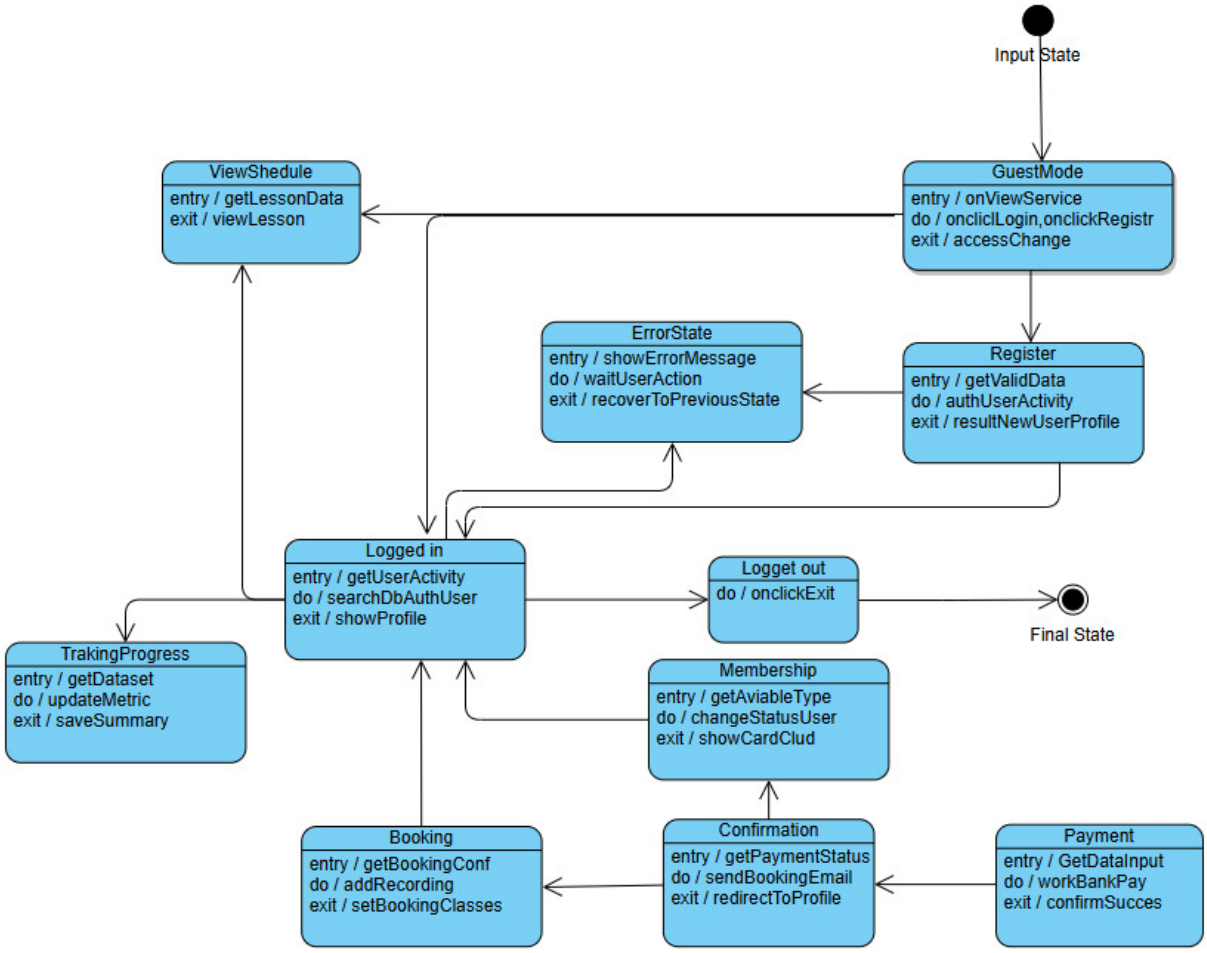

Рисунок 3.4 – Діаграма переходів

Вказано стан введення який починається коли користувач є гостем, GuestMode у цьому режимі користувач може обирати авторизація або реєстрацію, далі процес ділиться на дії які він може виконувати. Коли вже він авторизувався на сайті, в нього виникає дуже великий спектр інших дій

впровадження. З стану Logged in користувач може переглядати розклад, перейти до бронювання, та відстеження прогресу відвідування, можливість отримання картки членства у спортзалі. Модуль з підтвердженням отримує статус оплати від зовнішньої системи LipPay. Основна дія надіслати поштою підтвердження, і у кінці повернути користувача до початкової сторінки, у проєкті це сторінка профілю. Оплата і отримання членства у клубі відбувається за тією ж логікою з надсиланням підтвердження. Також два процеси входу і реєстрації ведуть до блоку помилки.

Діаграма послідовності показує взаємодію між об’єктами під час сценарію виконання, для компонентів інтерфейсу користувача з компонентами програмного забезпечення [16]. На діаграмі (див. рис. 3.12) об’єкти надсилають повідомлення один одному, кожен об’єкт займає стовпчик обмежений вертикальною лінією що тягнеться до низу сторінки, це має назву лінія життя. Дії між об’єктами розташовані у хронологічному порядку, в якому відбувається взаємодія. Повідомлення відображають виклики методів, передачу даних, відповіді системи. Зазвичай це застосування показує деталізацію варіантів використання, для допомоги порядку оброки запитів, передачі керування між модулями та залежності.

Обведена частина діаграми для позначення циклу. Для розділення логіки умовного оператора if, позначено горизонтальною пунктирною лінією. Цикл від авторизації до бронювання заняття складається з трьох компонентів та одного актора. Спочатку користувач виконує авторизацію у системі. Блок loop вказує на те що система буде повторювати процес авторизації поки користувач не введе коректні дані. Блок alt сервіс виконує інтенсифікацію користувача, якщо невдала спроба система поверне помилку і повторно форму входу. Якщо ж правильні дані, база поверне валідні дані, і це дасть можливість до наступного функціоналу. Користувач відкриває форму та обирає тренування, і система має два варіанти це створення запису бронювання, чи все ж видалення запису, залежить від натиснення користувачем відповідного поля і дорівнюватиме умові.

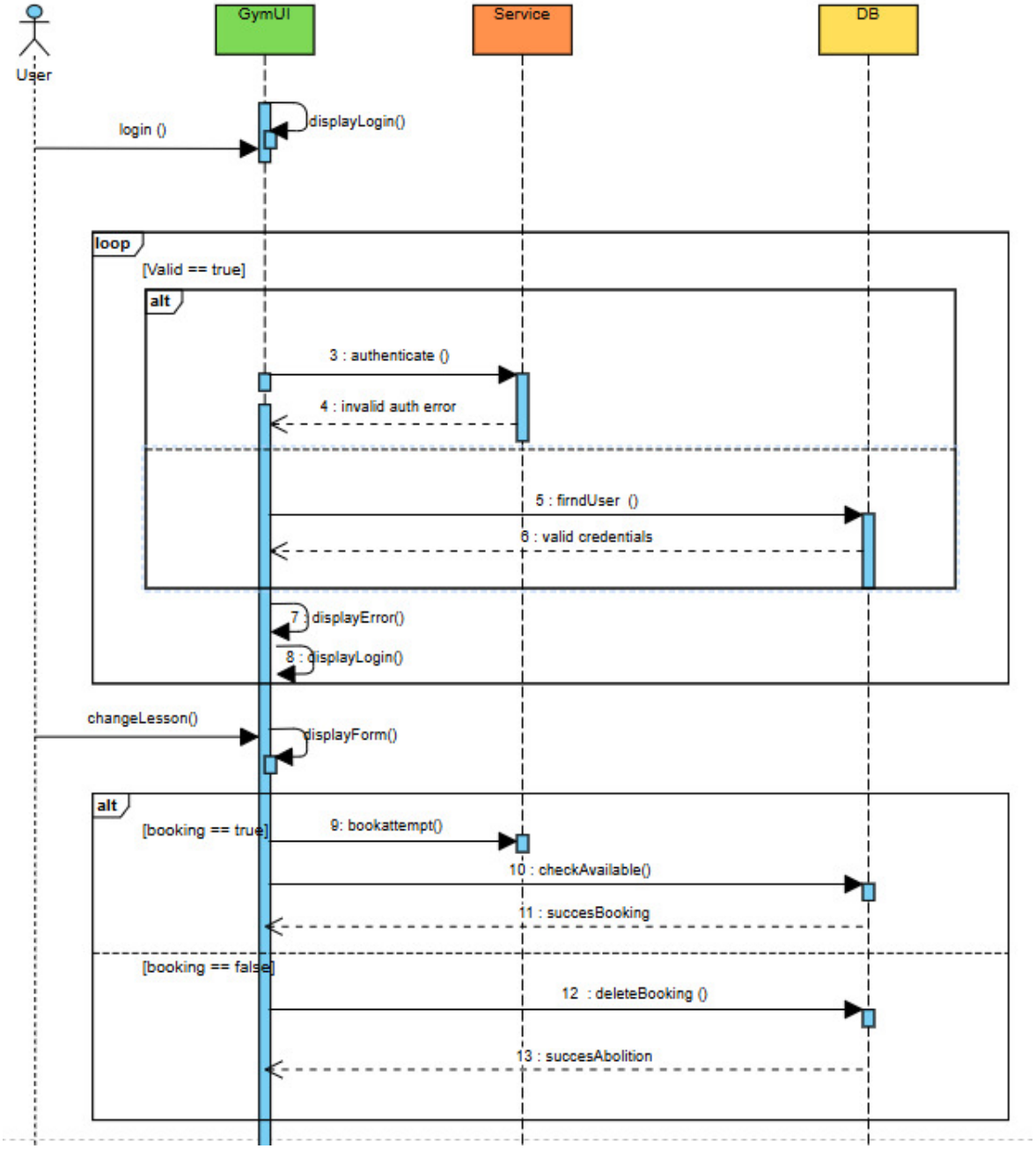

Рисунок 3.5 – Діаграма взаємодії для одного циклу авторизації з бронюванням заняття

Отже, сформована архітектура проєкту, дає підґрунтя для розробки системи та візуально показує всі сторони, визначає взаємозв'язки між компонентами, функціональні можливості.

3.3 Структура інтерфейсу застосунку

Ще до реалізації серверної частини та розробки фронтенд частини, мокапи розробляються для реалістичної візуалізації. Щоб відібрати кольори

та оцінити зручність, пропорції, допомогти виявити помилки ще до того як буде написаний код.

Використання онлайн-сервісу Moqups дозволяє створювати макети інтерфейсу для візуального представлення та розуміння розташування. Головна сторінка представлена (див.рис.3.14) бежевого кольору, зверху розташована шапка сайту, із сторінками. З правої сторони містить фото з логотипом спортзалу. Знизу розташовані маленькі картинки залів які він містить, при натисканні буде автоматичний перехід на сторінку з більшою кількістю картинок. Зліва опис та розташована кнопка запису на тренування.

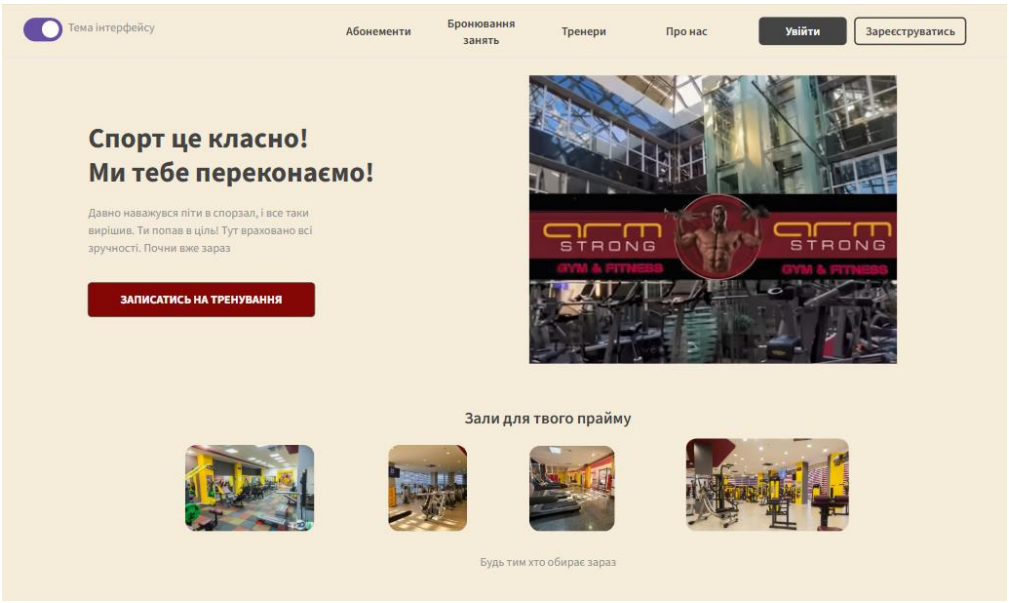

Рисунок 3.6 – Mockup головної сторінки

Основний функціонал, який буде персоналізованим розташований у профілі користувача, звідти можна отримати доступ до всіх функцій які доступні на сайті . Кольори підібрані під логотип спортзалу, і найбільш не різкі та лаконічні щоб зручно для користувацького ока. У верхній частині розташоване привітання з ім’я користувача, його аватар профілю. Справа розташоване навігаційне меню з переходом на основні сторінки, які будуть наявні. Така структура дозволяє забезпечити зручну навігацію. Основна частина, що посередині включає бронювання занять, тренерів, особисті нотатки, історію оплати та графік відвідуваності для аналізу регулярності

занять та відстеження власної активності. Весь інтерфейс виглядає досить сучасно, і спрямований на зручність використання.

Секція про абонементи, яка буде розташована нижче за першу головну інформацію (див. рис. 3.8). Три основних категорії абонементів доступних для оплати. Кожен перелік містить інформацію про те що входить у оплату, ціна та кнопка бронювання.

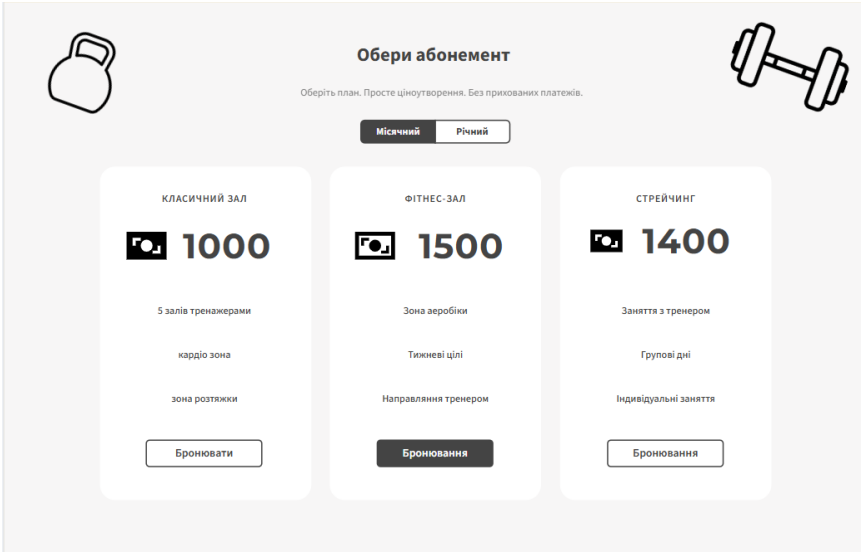

Рисунок 3.8 – Mockup абонементи

Секція з інформацією про вибір тренера, знизу описані причини, чому користувач має обрати тренера, тобто інформаційне поле (див. рис. 3.17).

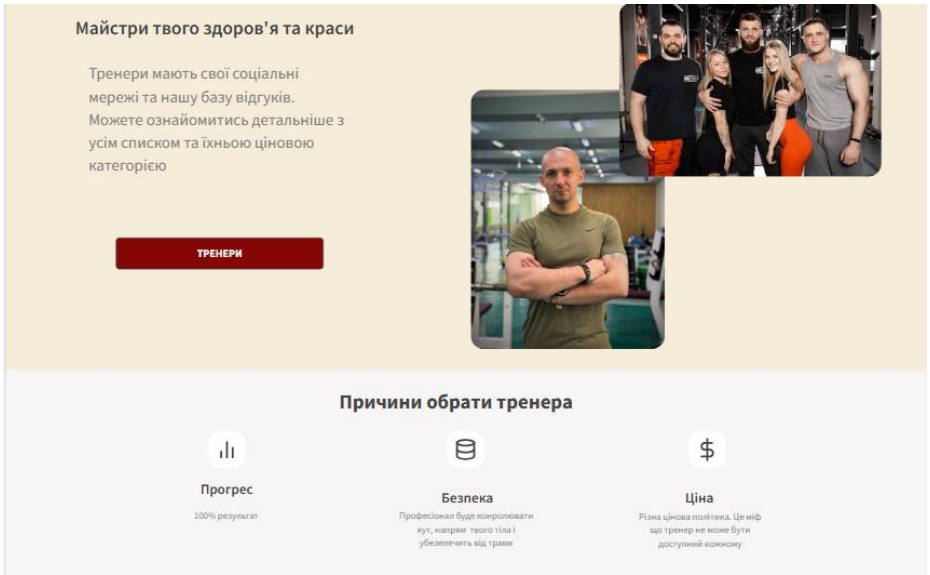

Рисунок 3.9 – Mockup тренери

Якщо натиснути на кнопку щоб переглянути кожного тренера окремо. Справа розташовані дві картинки тренерів. Зліва кнопка щоб дізнатись більше інформації та цінову політику тренерів.

Розроблені макети інтерфейсу демонструють логічну узгодженість, що підвищує зрозумілість та знижує візуальний шум. Застосовано принцип використання вже візуально знайомих користувачам елементів інтерфейсу для забезпечення інтуїтивності й покращення користувацького досвіду.

Висновки до розділу 3

У третьому розділі описано моделювання, та архітектуру застосунку для тренажерного залу. Варіанти використання чітко окреслюють вимоги застосунку. Діаграми забезпечують графічне відображення як функціональної поведінки, так і програмної структури застосунку. Візуальне відображення всіх основних модулів і компонентів, дозволить чітко зрозуміти структуру майбутнього проєкту та визначить, яку логіку мають наслідувати компоненти для комунікації. Клієнт-серверна архітектура поділений на клієнтську частину (інтерфейс користувача) та сервер, який виконує обробку даних, бізнес-логіку та роботу з базою даних, є найкращим рішенням до вимог і потреб системи. За допомогою Moqups створено макети основних екранів. Усі компоненти розташовані з точки зору комфорту та інтуїтивності для користувача. Усі компоненти чітко розмежовані за функціональністю, що забезпечить масштабування системи у майбутніх версіях проєкту. Отже, всі дані для безпосередньої розробки коду готові.

4 ПРОГАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.1 Робота з базою даних

Крім встановлення залежності пакетів необхідних для програмування, найпершим кроком реалізації застосунку стало створення бази даних, яка буде містити основну інформацію про тренерів, користувачів, типи тренування, броньовані місця. База даних MySQL та середовище адміністрування phpMyadmin, розгорнуто на локальному сервері. Підключення до бази даних відбувається у файлі що використовується для зберігання змінних середовища у вигляді пар ключ-значення (див.рис.4.1).

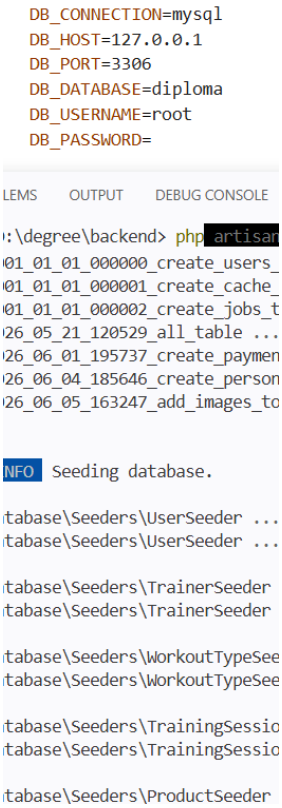

Рисунок 4.1 – Підключення та успішні міграції основних таблиць

У розробці системи використовується саме ORM-модель, вона буде забезпечувати комунікацію між таблицями бази даних та програмним кодом. Спочатку створено моделі з відповідними полями, та взаємозв’язки. Такий підхід дозволяє працювати з базою даних через об’єкти, а не постійно писати SQL-запити. Для заповнення початковими даними використано seeders.

Після міграції головних таблиць, та зв'язків можна легко додавати в подальшій розробці розширення застосунку інші таблиці відповідно до вимог, на момент розробки база даних складається з таблицями які потрібні для коректної роботи Laravel, а також таблиці з даними для застосунку (див. рис. 4.2).

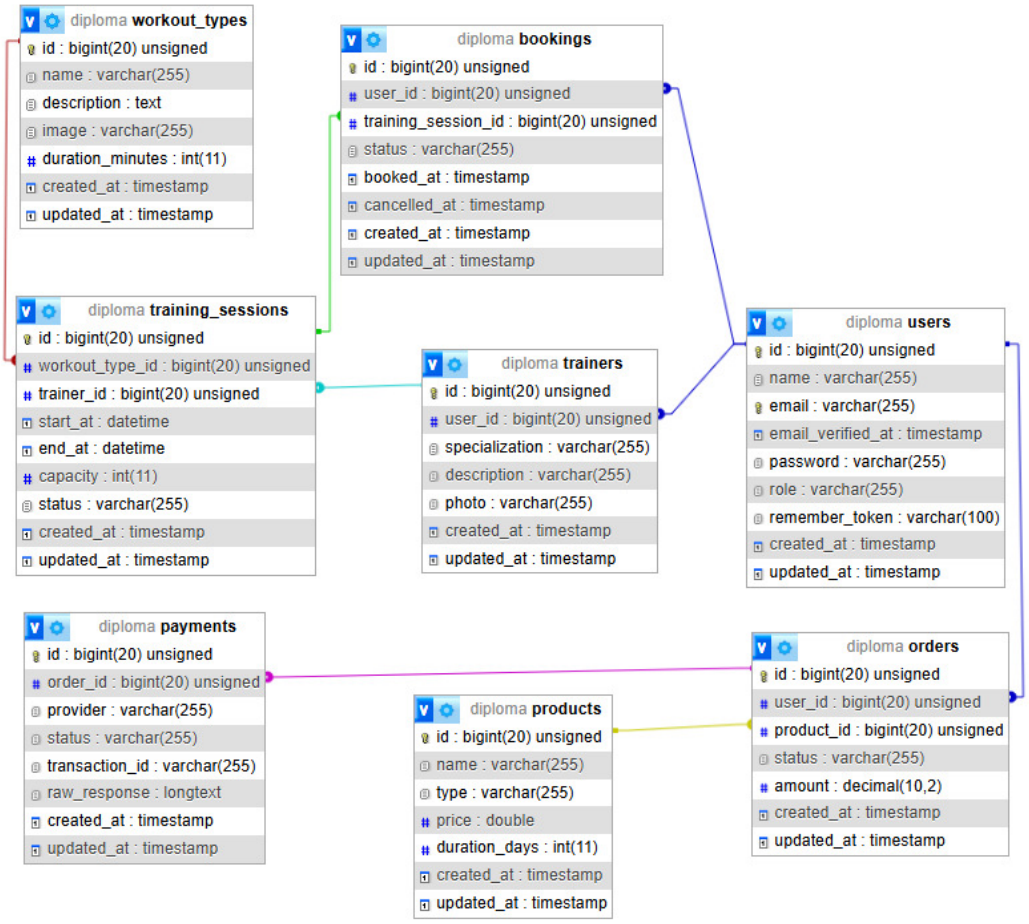

Рисунок 4.2 – Відображення створених таблиць та зв'язку між ними

Для сповіщень використано стандартний Laravel Notifications для надсилання системних сповіщень користувачу з поліморфним зв'язком. Тому тут не відображається взаємозв'язок, хоча ця таблиця буде автоматично створена. Для реальних проєктів це дуже гарна практика, бо фреймворк дає готову інфраструктуру, механізм який протестований, менше коду, легко додати нові канали повідомлень. Таблиця product в ній буде зберігатись саме членство до клубу, місячний річний абонемент та інше. У таблиці training_sessinons вона всі заплановані тренування. У Laravel, важливо вказати правильні назви таблиць міграції, оскільки автоматично очікується що модель

і таблиці мають однакову назву, а якщо у назві два слова виокремити великою буквою у випадку моделі, то в таблиці нижнє підкреслення.

4.2 Програмна реалізація

Глобальні налаштування потрібно підключати безпосередньо у app/bootstrap/app. Тут можна зареєструвати потрібні middleware, авторизації та логування запитів. Після цього всі запити будуть оброблятися відповідно до цих глобальних налаштувань.

```php
return Application::configure(basePath: dirname(__DIR__))
    ->withRouting(
        web: __DIR__.'/../routes/web.php',
        api: __DIR__.'/../routes/api.php',
        commands: __DIR__.'/../routes/console.php',
        health: '/up', )
    ->withMiddleware(function (Middleware $middleware): void {
        $middleware->appendToGroup('api', [ HandleCors::class,]);
        $middleware->validateCsrfTokens(except: [
            'liqpay/callback',
        ]);
    })
    ->withExceptions(function (Exceptions $exceptions): void {
    })->create();
```

Рисунок 4.3 – Налаштування шляхів та доступу

Перевірка токена необхідне для коректної взаємодії між застосунком та платіжною системою LiqPay. Оскільки серверні повідомлення надсилаються автоматично і не містять CSRF-токена, запит буде відхилено системою захисту застосунку, тому додавання дозволить безпечно приймати підтвердження про статус платежу та автоматично оновлювати статус.

Найпершим етапом є реалізація авторизації, реєстрації користувача. Щоб забезпечити цілісність входу використано middleware, після його додавання до будь-якого маршруту буде допуск згідно зазначених ролей. Політики доступу використано для керування окремими діями, у випадку системи ролі тренера відкривається можливість редагування занять (див. рис. 4.3). За доменно-орієнтованим підходом, основна логіка винесена у окремі сервіси, таким чином побудована архітектура системи. Контролери виконують лише функцію прийому запитів та передачі даних.

```php
class TrainingSessionPolicy
{
    public function viewAny(User $user): bool
    {
        return true;
    }
    public function view(User $user, TrainingSession $session): bool
    {
        return true;
    }
    public function create(User $user): bool
    {
        return in_array($user->role, ['admin', 'trainer']);
    }
    public function update(User $user, TrainingSession $session): bool
    {
        return in_array($user->role, ['admin', 'trainer']);
    }
    public function delete(User $user, TrainingSession $session): bool
    {
        return in_array($user->role, ['admin', 'trainer']);
```

Рисунок 4.3 – Політика редагування тренувань

Основні бізнес-процеси такі як бронювання тренувань, перевірка доступності місця, скасовані заняття реалізовані у BookingService. Функція book перевіряє статус тренування, чи доступні місця, чи користувач не записаний. Застосовано Enum для зручності реєстру системи, щоб не перевіряти кожного разу правильність написання слова (див. рис. 4.4).

```php
class BookingService
{ public function book(User $user, int $sessionId): Booking
    {
        return DB::transaction(function () use ($user, $sessionId) {
            $session = TrainingSession::query()
                ->lockForUpdate()
                ->find($sessionId);
            if (!$session) {
                throw ValidationException::withMessages(['session' => 'Тренування не знайдено']);
            }
            // статусу тренування
            if ($session->status !== 'scheduled') {
                throw ValidationException::withMessages(['session' => 'Запис на це тренування неможливий']);
            }
            // чи користувач вже записаний
                $booking = Booking::where('user_id', $user->id)
                ->where('training_session_id', $sessionId)
                ->first();
            if ($booking) {
                if ($booking->status === BookingStatusEnum::BOOKED) {
                    throw ValidationException::withMessages(['booking' => 'Ви вже забронювали це тренування']);
                }
                $booking->update([
                    'status' => BookingStatusEnum::BOOKED,
                    'cancelled_at' => null,]);
                return $booking;
            }
            // кількості місць
            $bookedCount = Booking::where('training_session_id', $sessionId)
                ->where('status', BookingStatusEnum::BOOKED)
                ->count();
            if ($bookedCount >= $session->capacity) {
                throw ValidationException::withMessages(['session' => 'Немає вільних місць']);
            }
            return Booking::create([
                'user_id' => $user->id,
                'training_session_id' => $sessionId,
                'status' => BookingStatusEnum::BOOKED,
                'booked_at' => now(),]);
```

Рисунок 4.4 – Функція бронювання занять

Скасування занять реалізовано через функцію cancel сервісу BookingService (див. рис. 4.5). Тут використовується саме анонімна функція в середині транзакції, без use вона б не бачила зовнішніх змінних.

```php
public function cancel(User $user, int $bookingId): Booking
{
    return DB::transaction(function () use ($user, $bookingId) {
        $booking = Booking::where('id', $bookingId)
            ->where('user_id', $user->id)
            ->first();

        if (!$booking) {
            throw ValidationException::withMessages(['booking' => 'Бронювання не знайдено']);
        }
        if ($booking->status === BookingStatusEnum::CANCELLED) {
            throw ValidationException::withMessages(['booking' => 'Бронювання вже скасоване']);
        }
        $booking->update(['status' => BookingStatusEnum::CANCELLED,'cancelled_at' => now(),]);
        return $booking;
```

Рисунок 4.5 – Функція скасування занять

Ця функція виконує саме змінну статусу бронювання, а не видаляє запис для того щоб зберігати статистику відвідувань окремого користувача, а також щоб не перевантажувати дані кожного разу до бази даних.

Карту товариства користувач може оплатити самостійно, використано зовнішній компонент який буде відповідати за безпеку даних, і дозволить створити потужний застосунок. На стороні сервісу треба потурбуватись про дані та сигнатуру. Щоб автентифікувати запит від сервера LipPay необхідно сформувати підпис на стороні сервера використовуючи дані отримані у відповідь. Підпис обчислюється шляхом хешування об'єднаних значень приватного ключа, закодованих даних та повторного приватного ключа за допомогою алгоритму SHA-1. Остаточний підпис необхідно порівняти з отриманим від зворотного виклику і якщо підпис ідентичний то отримано справжню відповідь не змінену третьою стороною [17]. Такий підхід забезпечує цілість даних системи. Створено контролер який має два методи. Перший метод викликається після натискання користувачем оплати послуги. Створює новий запис замовлення із статусом. Після створення контролер звертається до сервісу (див. рис. 4.6).

```php
class PaymentService
{
    public function getPaymentData(Order $order): array
    {
        $params = [
            'public_key' => config('services.liqpay.public_key'),
            'version' => '3',
            'action' => 'pay',
            'sandbox' => '1',
            'amount' => $order->amount,
            'currency' => 'UAH',
            'description' => 'Оплата: ' . $order->product->name,
            'order_id' => $order->id,
            'result_url' => config('app.url') . '/products',
            'server_url' => config('app.url') . '/liqpay/callback',
        ];
        $data = base64_encode(json_encode($params));
        $privateKey = trim(config('services.liqpay.private_key'));
        $signature = base64_encode(sha1($privateKey . $data . $privateKey, true));
        return [
            'data' => $data,
            'signature' => $signature,
        ];
    }
    public function checkSignature(string $data, string $signature): bool
    {
        $generatedSignature = base64_encode(
        sha1(config('services.liqpay.private_key') . $data . config('services.liqpay.private_key'), true));
        return hash_equals($generatedSignature, $signature);
    }
    public function decodeData(string $data): array
    {
        return json_decode(base64_decode($data), true);
```

Рисунок 4.6 – Сервіс оплати послуги

Масив params місить усю необхідну інформацію про платіж, його ідентифікатор, рекомендовану версію API, тип операції суму платежу, валюту опис послуги, адреси щоб повернути користувача а також відповідь для сервера щоб змінити статус оплати у базі даних. Параметр sandbox дає можливість тестувати роботу без фактичного списання коштів з банківської картки користувача.

Другий метод у контролері призначений для обробки серверного повідомлення, після завершення платіжної операції. У випадку успішної перевірки закодованих даних статус платежу змінюється.

Важливим моментом для виконання оплати, та перевірки отримання даних є те що сайт повинен бути у загальному доступі бо звернення LipPay до адреси, яка розгорнута на локальному сервері, не можливе. У цьому допомогла хмарна мережева платформа ngrok (див. рис. 4.7). Це глобально розподілений зворотній проксі-сервер, що захищає доступ до локально запущених

застосунків через тимчасово згенеровані публічні адреси [18]. Завдяки створенню захищеного тунелю платіжна система отримувала можливість надсилати серверні повідомлення.

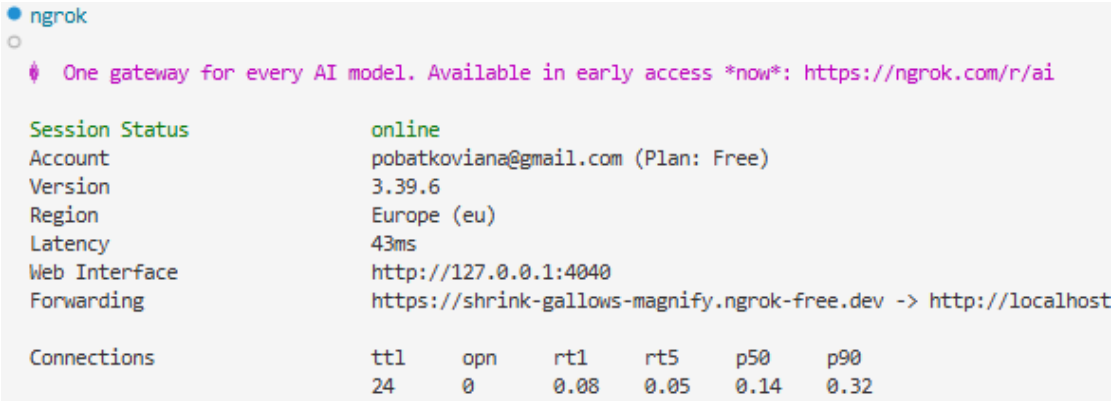

Рисунок 4.7 – Розгортання сайту у публічний доступ

У застосунку буде роздільна програмна логіка частин. API створене на Laravel, буде одночасно використовуватись вебзастосунком, так інші пристрої можуть мати можливість доступу, що дає розширення та масштабування у майбутньому. Щоб frontend і backend могли спілкуватись потрібні cors. Треба щоб браузер дозволив обмін даними між ними, бо робота на різних доменах та портах, забороняється без правильно налаштованих запитів з клієнтської частини, запити блокуються політикою same-origin. У Laravel cors можна налаштувати у файлі config/cors.php, вказавши дозволені джерела, методи та заголовки (див. рис. 4.8). Важливо що публічні адреси не варто додавати у файл cors, оскільки вони працюють тільки з браузерними запитами.

```php
<?php
return [
    'paths' => ['api/*', 'sanctum/csrf-cookie', 'login', 'logout'
    'allowed_methods' => ['*'],
    'allowed_origins' => ['http://localhost:3000'],
    'allowed_origins_patterns' => [],
    'allowed_headers' => ['*'],
    'exposed_headers' => [],
    'max_age' => 0,
    'supports_credentials' => true,
];
```

Рисунок 4.8 – Налаштування cors

Налаштування axios важливо для дозволу надсилати запити до серверів та отримувати дані щоб працювати з відповідями. Асинхронна обробка для одночасного використання кількох завдань, за допомогою promise [19]. У цьому файлі на стороні frontend важливо вказати токен який пред'являється для надання доступу до захищеного ресурсу. Під час входу на сервер ідентифікує облікові дані та повертає захищений токен. Щоб отримати доступ цей токен надсилається у кожному заголовку (див. рис. 4.8). Використання такого розмежування логіки для зручного оброблення результатів. Перехоплювач дозволяє змінювати запити швидко перед їх відправкою чи отриманням.

```
export const api = axios.create({
  baseURL: process.env.NEXT_PUBLIC_API_URL,
});
api.interceptors.request.use((config) => {
  const token = localStorage.getItem("token");
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});
```

Рисунок 4.9 – Дозвіл надсилання запитів

Важливо вказувати великими літерами саме з таким префіксом назву у файлі налаштувань проєкту, бо Next розділяє за назвою сервісні та публічні змінні. Для профілю, бронювання, історії оплати, головної сторінки, статистики відвідувань, тренерів та абонементів створено компоненти.

Компонент використовує стани для збереження та оновлення даних інтерфейсу без необхідності перевантаження сторінки. Змінні оголошені на початку, призначенні для зберігання переліку доступних тренувань, та оновлення цього переліку після отримання нових даних (див. рис. 4.10). Змінна message використовується для відображення інформаційних повідомлень. Хук використовуються для одноразового завантаження списку тренувань під час першого відображення сторінки. Завдяки асинхронній взаємодії з контролем стану, динамічно оновлюється інтерфейс без необхідності повторного завантаження вебсторінки.

```
export default function BookingPage() {
  const [sessions, setSessions] = useState([]);
  const [message, setMessage] = useState("");

  const loadSessions = async () => {
  const data = await getSessions();
  setSessions(data);
};
  useEffect(() => {
    loadSessions();
  }, []);

const book = async (id: number) => {
  try {
    const res = await createBooking(id);
    setMessage(res.message);
    loadSessions();
  } catch (error: any) {
    setMessage(error.response?.data?.message || "Помилка бронювання");
  }
};
  const cancel = async (id: number) => {
    await cancelBooking(id);
    loadSessions();
```

Рисунок 4.10 – Компонент бронювання

У файлі компонента разом із розміткою прописуються стилі Tailwind. Логіка написання така що сама назва класу говорить про суть які він виконує, коли рівень не початковий це зручно для швидкого написання, і легко у підтримці, де одразу видно структуру компонента (див. рис. 4.11).

```
return (
 <div className="p-10 max-w-7xl mx-auto" >
   <h1 className="text-3xl font-bold mb-2 mx-auto text-center">
     Оберіть тренування та забронюйте місце!
   </h1>
   <div className="grid grid-cols-2 gap-4 ">
     {message && (
       <div className="mb-2 rounded bg-gray-100 p-3">
         {message}
       </div>
     )}
     {sessions.map((s: any) => {
     const activeBooking = s.bookings?.find(
         (b: any) => b.status === "booked");
     const availablePlaces = s.capacity - s.booked_count;
       return (
         <div key={s.id} className="bg-[#fff2f2] rounded-2xl shadow p-4 flex gap-4 "
           <div  className="flex flex-col gap-5">
             <h3 className="text-xl font-semibold">
               {s.workout_type.name}
             </h3>
```

Рисунок 4.11 – Частина структури розмітки

Інші компоненти розроблено відповідно до зазначеної та описаної логіки. Єдиний компонент який відрізняється своєю логікою оскільки відображає дані у вигляді графіку, для нього потрібно підключити бібліотеку Chart.js. Встановлення необхідних пакетів реєстрації бібліотеки, отримання статистичних даних і перетворення у формат сумісний для розуміння. Після підготовки даних, передача до компонента графіка, який виконує їх графічне відображення. Також додано шар сервісів доступу, для відправки запиту, повернення даних компонентам. Для того щоб зменшити дублювання коду, коли код буде потрібен в кількох компонентах, а також зручність заміни не доведеться шукати всі місця в коді. Централізація роботи з API на сервері, задає правила роботи системи.

```
import { api } from "@/lib/axios";
export const login = async (
  email: string,
  password: string
) => {const res = await api.post("/login", {
    email,
    password,
  });
  localStorage.setItem("token", res.data.token);
  console.log(localStorage.getItem("token"));
  return res.data;
};
export const register = async (
  name: string,
  email: string,
  password: string,
  password_confirmation: string
) => {
  const res = await api.post("/register", {
    name,
    email,
    password,
    password_confirmation,});
  return res.data;
};
export const getUser = async () => {
  return api.get("/profile");
};
export const logout = async () => {
  return api.post("/logout");
};
```

Рисунок 4.12 – Сервіси на стороні frontend частини

Неавторизованому користувачу будуть недоступні деякі дії. Головна сторінка буде виконувати інформаційну панель, що одну із завдань поставленої роботи. Інтеграція механізмів позитивно впливає на зручність використання системи.

4.3 Керівництво користувача

Щоб зорієнтуватись і мати уявлення про дії на створеному вебзастосунку, детально описано чіткі кроки. Для більшого доступу до функцій крім перегляду інформації про сайт, тренерів і контакти, зали. Користувач має бути авторизованим у системі. Після заповнення полів користувача зрівнюють згідно його ролі. Користувачу тренеру буде доступна платформа редагування запланованих тренувань.

Зліва розташована панель доступу до різних компонентів. У авторизованого користувача вона знаходиться зліва. Бронювання занять відповідає всім перевіркам і користувача з’являється кнопка скасування тільки після реального бронювання (див. рис. 4.13). Зміна кількості місць відповідно також зменшується більше вказаного ліміту в базі даних не буде створено додаткових записів.

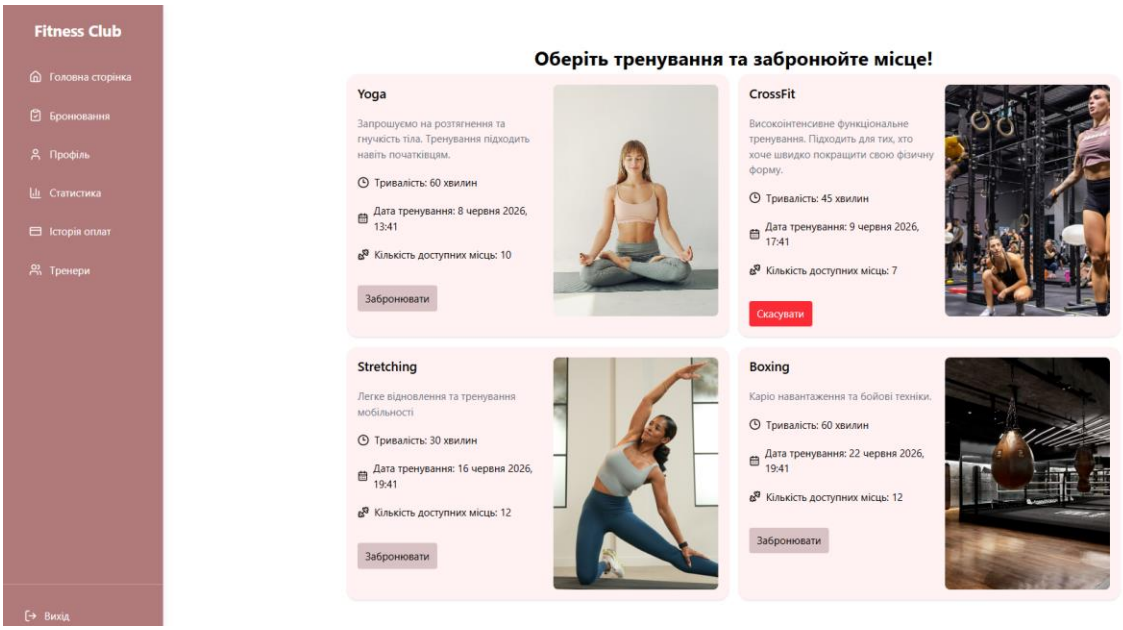

Рисунок 4.13 – Бронювання заняття

У відділі статистики користувач може індивідуально слідкувати за планом своїх відвідувань, яка сформована на основі даних бронювання занять (див. рис. 4.14).

Рисунок 4.14 – Статистика відвідувань

На головній сторінці є інформація представлення абонементів. Відповідно кнопка оплати, яка введе до зовнішнього сервісу LiqPay. Його було кастомізовано щоб користувач мав уявлення за що саме він платить. У відділі історії оплат буде відображатись всі виконані дії.

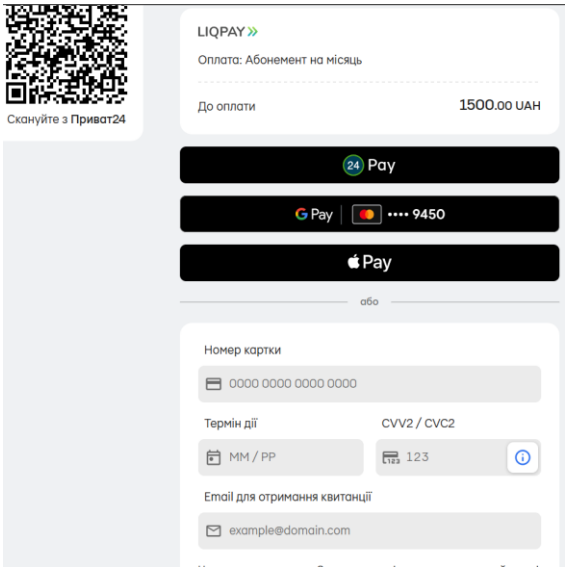

Рисунок 4.15 – Кастомізація LiqPay відповідно до послуги

Користувач здійснює успішну оплату, переходить до власного профілю і бачить дані не тільки про своє ім’я, адресу пошти, роль у системі, але і картку

абонементу доступу до визначеної дати за обраним абонементом відвідування (див. рис. 4.16).

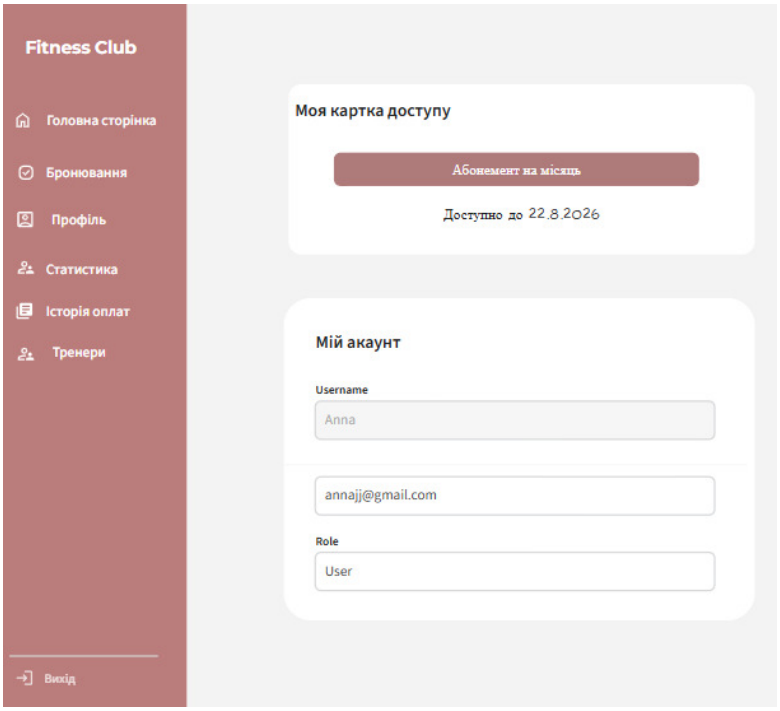

Рисунок 4.16 – Відображення картки у профілі користувача

Користувач також може відвідати сторінку з його історією оплати (. Стиль сторінок має один напрям, статус обробки також видно в залежності відповідної інформації з бази даних.

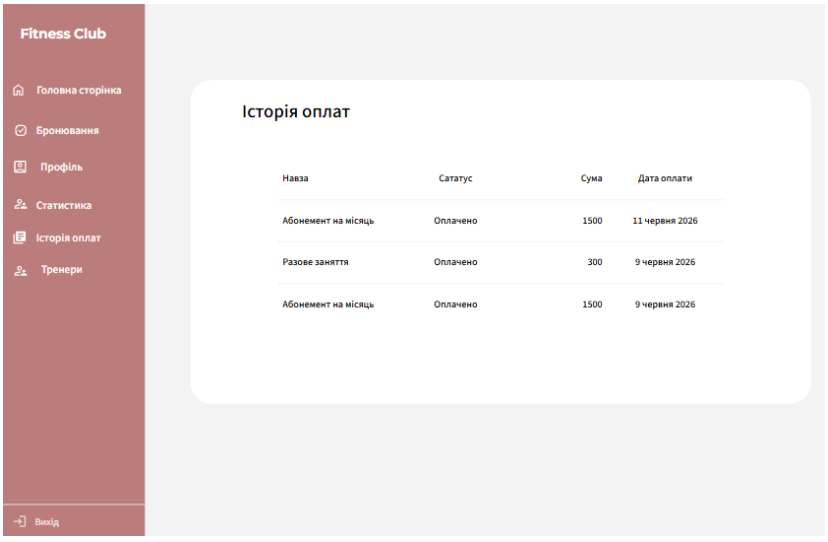

Рисунок 4.17 – Відображення картки у профілі користувача

У результаті виконання зазначених кроків користувач матиме зручне середовище слідкування та комунікації з адміністрацією тренажерного залу. Через зручність самостійної оплати, зменшується навантаження на персонал.

А гарний інтерфейс та інтернактивність процесу відвідування спортзалу мотивуватиме користувача.

4.4 Тестування вебзастосунку

Заключним етапом розробки стає тестування програмного забезпечення. Важливо виявити помилки, та оцінити якість. Верифікація процесу визначає чи правильно реалізовано функціональні вимоги. На підтвердження чи кінцевий продукт задовольняє вимоги користувача. У реальних сценаріях використання, оцінюється коректність роботи. У межах вебзастосунку для керування процесами спортзалу використано автоматизоване тестування засобами фреймворку Laravel.

Тестування білиться на дві категорії модельного тестування та Feature. Використання першого типу перевіряє роботу окремих компонентів системи ізольовано від інших частин. Саме тут тестувались функції генерації параметрів для платежу, формування коректних даних системи бронювання. Це допомогло локалізувати помилку у конкретному модулі системи. Інший тип тестування використано для перевірки цілісних сценаріїв, яка вони працюють із компонентами. У межах цього сценарію створювались замовлення після вибору конкретного абонементу, змінювались значення статусу. Перевірялась авторизація користувача. Тестів виконано 9 за різними головними функціями перевірки системи. Отже, проведено тестування набору файлів які відповідають окремим блокам перевірки різного функціоналу. У таблиці 4.1 подано опис дій які виконує кожен з них та їх результат.

Таблиця 4.1 – Звіт тестових перевірок

Результат	Відповідає за	Час обробки
MembershipTest.php	Список абонементів, відкриття сторінки картки, отримання оновлених коректних дані	1.66 ms
OrderTest.php	Після натискання створюється запис у таблиці із статусом	11 ms

Кінець таблиці 4.1

SignatureTest.php	Перевірка даних які повернуто сервісом	2.1 ms
CallbackTest.php	Створення запису у таблиці після підтвердження	2.5 ms
AuthTest.php	Авторизація коректність модулів	0.78ms
BookingTest.php	Дублювання даних, пошук користувача за минулими записами	1.1 ms
ProfileTest.php	Підтягування даних конкретного користувача, відображення функцій відповідно до ролі	1 ms
CancelBookingTest.php	Пошук за минулими записами, зміна статусу без видалення	2 ms
AccessTest.php	Доступ неавторизованого користувача до закритих функцій	1.5 ms

Наведено функцію перевірки бронювання (див. рис. 4.8), щоб орієнтовно зрозуміти з чого складаються та яка логіка написання інших тестів системи. У консолі подано успішний результат запуску перших п’яти тестів різного типу тестування.

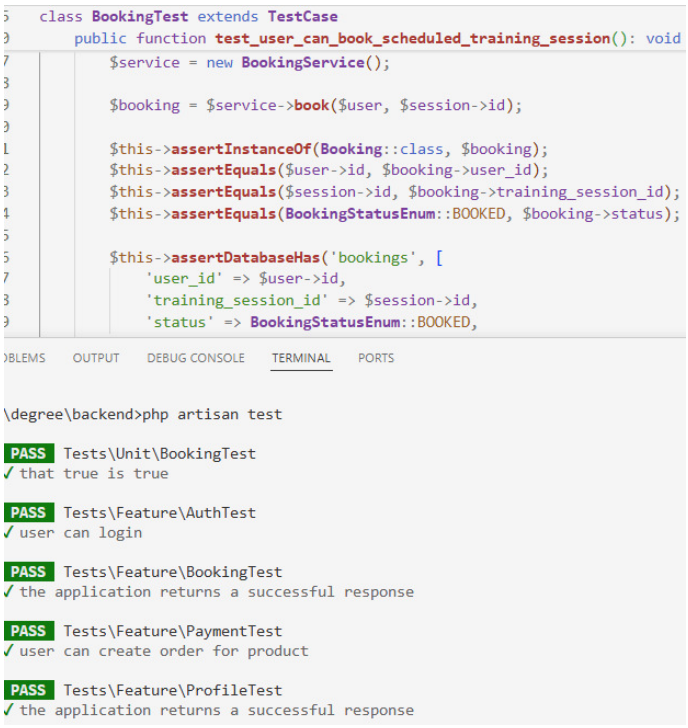

Рисунок 4.18 – Тестування головних функцій

Результатом, на поточному етапі розробки став застосунок що виконує та тестує головні сценарії обробки. Виконані тести підтвердили коректність базових процесів, зокрема авторизації, бронювання, обробки замовлень, формування платежів та оновлення даних користувача. При базовому використані демонструється надійність та відсутність провальних тестів. Подальше тестування має бути спрямоване на перевірку менш типових ситуації, граничних випадків.

Висновки до розділу 4

У четвертому розділі представлено реалізацію на серверної частини, для вебзастосунку тренажерного залу. Детально описано всі процеси починаючи з побудови бази даних, роботу сутностей і зв’язків. Розглянуто механізм бронювання тренувань, управління замовленнями, роботу з профілями користувачів та систему авторизації, інтеграцію із зовнішнім сервісом LiqPay. Описано головні функцій зі сторони коду та логіки знаходження, особливості використання транзакцій.

Процес формування параметрів платежу, генерації цифрового підпису. Для перевірки реальної оплати використано хмарне середовище для розгортання у відкритий доступ. Це забезпечило отримання повідомлень та внесення інформації у базу даних.

Наведено результати тестування основних модулів системи. Саме механізми авторизації, бронювання тренувань, обробки замовлень, роботи платіжного модуля та відображення інформацій відповідно до ролей. Подальше тестування має бути спрямоване на перевірку менш типових ситуації, граничних випадків. Наступні версії застосунку передбачають покращення та розширення функціоналу.

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи став відлагоджений вебзастосунок для автоматизації інформування та взаємодії з клієнтами тренажерного залу. Досягнута поставлена мета розробка вебзастосунку тренажерного залу для спрощення взаємодії між користувачами та адміністрацією закладу. Виконано всі поставлені завдання:

– проведено аналіз наявних систем, щоб врахувати помилки і зробити платформу більш професійною у роботі;

– сформовано основні вимоги до функціональності та інтерфейсу;

– спроєктовано архітектуру, враховуючи вибір технологій та рішень для всього проєкту;

– змодельована структура бази даних та серверної частини;

– розроблено frontend частину;

– протестовано розроблений вебзастосунок.

Проведений аналіз предметної області та сучасних аналогів системи, допоміг зрозуміти потребу і недоліки конкурентів, на чому варто зосередити увагу при розробці. Обрано найкращі і сучасні технології необхідні для реалізації. Змодельовано діаграми для відображення архітектури майбутнього застосунку. Розроблено програмну частину та виконано тестування. Врахування аналогічних рішень конкурентів дозволила створити лаконічний застосунок із інтуїтивним інтерфейсом. Подальший розвиток застосунку може включати розширення функціоналу під індивідуальні потреби користувача, бо аналіз показав що на ринку не так багато потужних і дійсно цікавих рішень для спортивного життя користувачів.

У підсумку створений вебзастосунок повністю відповідає поставленим початковим цілям. Результати тестування підтвердили коректність функціонування вебзастосунку, його готовність до практичного застосування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Dolot, A. (2018). The characteristics of Generation Z. *E-mentor*, *74*(2), P.44-50.

2. Fitness content UA. URL:https://fitnessconnect.com.ua/ua.html (дата звернення 27.04.2026).

3. Wang, Zijian, et al. Construction productivity and digital technologies. Automation in Construction, 2026. DOI: 10.1016/j.autcon.2026.106768

4. Bjerke, May Britt; Renger, Ralph. Being smart about writing SMART objectives. *Evaluation and program planning*, 2017, P. 125-127.

5. Tie, J., Jin, J., & Wang, X. (2011, July). Study on application model of three-tiered architecture. In *2011 Second International Conference on Mechanic Automation and Control Engineering* P. 7715-7718.

6. Sundeuk Kim, Jong Seon Kim, Hoh Peter. Web Programming and Multi-Tier Architecture of Web Applications. TEM Journal. – 2024. – Vol. 13, Issue 4. – P. 3286–3294. (MDPI, in press).

7. Клієнт-серверна архітектура. URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення 27.04.2026).

8. Vyas, Rishi. Comparative analysis on front-end frameworks for web applications. *International Journal for Research in Applied Science and Engineering Technology*, 2022, ISSN: 2321-9653. DOI: 10.22214/ijraset.2022.45260. (Scopus, in press).

9. Тенденції завантажень пакетів npm. URL: https://npmtrends.com/next-vs-react-vs-vue (дата звернення 28.04.2026).

10. ORM (Object-relational mapping). URL: https://snettech.com/orm---anh-xa-quan-he-doi-tuong-objectrelational-mapping-27.html (дата звернення 25.04.2026).

11. LiqPay. URL: https://www.liqpay.ua/information/about/info (дата звернення 28.04.2026)

12. Stupin, A., Hlynchuk, L., & Hryshanovych, T. (2022). Алгоритм підключення сервісів онлайн-оплат Fondy, LiqPay та їх реалізація. *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*, *1*(17), 6575. , DOI:10.28925/2663-4023.2022.17.6575

13. TailwindCSS. URL: https://tailwindcss.com/docs/installation/using-vite (дата звернення 29.04.2026)

14. Спільнота програмістів. UML. URL: https://dou.ua/forums/topic/40575/ (дата звернення 05.05.2026).

15. Діаграма класів. URL: https://www.mindonmap.com/uk/blog/what-is-uml-class-diagram/ (дата звернення 06.05.2026).

16. Діаграма послідовності (Sequence Diagrams). URL: https://www.maxzosim.com/sequence-diagrams/ (дата звернення 08.05.2026).

17. Документація підключення LipPay. URL: https://www.liqpay.ua/en/doc (дата звернення 12.05.2026).

18. Документація ngrok. URL: https://ngrok.com/docs/what-is-ngrok (дата звернення 20.05.2026).

19. Основні можливості Axios. URL: https://www.it-notes.wiki/javascript/axios-library-in-javascript/ (дата звернення 25.06.2026).