МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет Інформаційних технологій
Кафедра Інформатики і прикладного програмного забезпечення
Спеціальність Інженерія програмного забезпечення
Форма навчання Денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Калініна Олексія Миколайовича
(прізвище, ім'я, по батькові здобувача)

на тему «Розробка програмного забезпечення замовлень для клієнтів спортивних закладів»
(повна назва теми)

за матеріалами праць провідних спеціалістів з розробки ПЗ та проектування БД
(повна назва бази дослідження)

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

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 11.06.2025 р. № 12
Завідувач кафедри _____
(підпис)
д.т.н., професор Зеленський О.С.
Наук. ступень, вчене звання *Ініціали, прізвище*

Кривий Ріг – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____ Зеленський О.С.
(підпис) (Прізвище, ініціали)
« 11 » червня 2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ

1. Тема роботи «Розробка програмного забезпечення замовлень для клієнтів спортивних закладів»

Керівник роботи к.е.н., доцент Баран С.В.
затверджені наказом закладу вищої освіти від «04» квітня 2025 р. № 222-ст

2. Строк подання здобувачем роботи до «11» червня 2025 р.

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

Розділ 1. Постановка задачі

Розділ 2. Проектування web-сайту

Розділ 3. Проектування бази даних web-сайту

Розділ 4. Розробка web-сайту

Об'єкт дослідження: web-сайти управління замовленнями в діяльності спортивних закладів

Предмет дослідження: алгоритм автоматизації роботи сервісу

Мета кваліфікаційної роботи: *створення web-сайту*

5. Дата видачі завдання «04» квітня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів МДР	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	04.04.2025-13.04.2025	
2	Підготовка розділу 2	14.04.2025-26.04.2025	
3.	Підготовка розділу 3	27.04.2025-15.05.2025	
4	Підготовка розділу 4	16.05.2025-08.06.2025	
5	Реєстрація завершеної кваліфікаційної роботи	09.06.2025	Реєстраційний №___ «09»червня 2025 р.
6	Отримання відгуку від наукового керівника	10.06.2025	
7	Подання кваліфікаційної роботи на перегляд завідувачу кафедри	11.06.2025	
8	Отримання зовнішньої рецензії	12.06.2025	
9	Попередній захист кваліфікаційної роботи на кафедрі	13.06.2025	
10	Підготовка до захисту в ЕК	16.06.2025-21.06.2025	

Завдання підготував науковий керівник _____ __Зеленський О.С.___
(підпис) (прізвище та ініціали)

Завдання одержав _____ ___Сергієнко С.С.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

на кваліфікаційну бакалаврську роботу

«Розробка програмного забезпечення замовлень для клієнтів спортивних закладів»

Калініна Олексія Миколайовича

Кваліфікаційна бакалаврська робота на здобуття освітньо-кваліфікаційного рівня бакалавра зі спеціальності 121 «Інженерія програмного забезпечення» – Державний університет економіки і технологій – Кривий Ріг, 2025.

У кваліфікаційній роботі розроблене програмне забезпечення для управління замовленнями для клієнтів спортивних закладів.

Для реалізації задачі було створено web-сайт, розроблено серверну частину, призначену для обробки запитів користувачів. В процесі розробки було використано сучасні інструменти: мова програмування PHP, framework Laravel, СУБД MySQL.

Результатом кваліфікаційної роботи стала розробка web-сайту, який надає користувачу можливість управління замовленнями у мобільному додатку.

Розроблений веб-сайт відповдає сучасним вимогам і може бути використаний для підприємств, що займаються продажем відповідних послуг.

Ключові слова PHP, LARAVEL, MYSQL.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД (база даних)	Впорядкований набір логічно взаємопов’язаних даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів
СУБД	Система управління базами даних
ПЗ	Програмне забезпечення
Laravel	Безкоштовний веб-фреймворк з відкритим кодом, призначений для розробки з використанням архітектурної моделі MVC (англ. Model View Controller - модель-уявлення-контролер)
Міграції	Система контролю версій для вашої бази даних
Роутінг	Вхідний URL розбирається спеціальним чином і по його результату виконується певний код

ЗМІСТ

ВСТУП

У сучасних умовах швидкого розвитку інформаційних технологій та збільшення попиту на послуги спортивних закладів ефективне управління замовленнями клієнтів стає одним із ключових факторів конкурентоспроможності.

За даними EFPMG, обсяг ринку мобільних і веб-рішень для керування спортивними закладами у Європі щороку зростає приблизно на 15%. Водночас, згідно з опитуванням Всеукраїнської асоціації фітнес-індустрії, тільки близько 40% українських середніх і малих спортклубів мають власні інформаційні системи для автоматизації записів клієнтів, обробки замовлень і контролю оплат. Це створює суттєвий розрив між потребами ринку і ступенем цифрової трансформації локальних підприємств. У результаті й виникає необхідність розробки доступних та адаптивних програмних продуктів, здатних забезпечити не лише реєстрацію й бронювання послуг, але й комплексне управління замовленнями.

Реальні приклади успішного впровадження подібних систем свідчать про те, що інвестиції у розробку та адаптацію власного ПЗ можуть окупитися вже протягом першого року експлуатації. Наприклад, мережа фітнес-клубів «Sport Life» у 2023 р. на базі власної платформи, створеної спільно з українською IT-компанією, змогла підвищити загальний показник продажів абонементів на 18% порівняно з 2022 р., а також оптимізувати витрати на персонал на 10% завдяки автоматизації щоденних операцій і виключенню ручного введення даних.

Мета даної дипломної роботи полягає в розробці програмного забезпечення для управління замовленнями клієнтів спортивних закладів, яке б:

1. Забезпечило швидке та зручне оформлення замовлень як для клієнтів так і для адміністратора
2. Надало широкий набір функцій для керування каталогом послуг, налаштування гнучких тарифів і акцій.
3. Інтегрувалося з популярними в Україні платіжними системами та

касовими апаратами.

4. Мало вбудовані аналітичні інструменти для оцінки ефективності роботи клубу.

5. Відповідало законодавчим вимогам щодо безпеки персональних даних клієнтів.

6. Основні завдання дипломної роботи:

7. Проаналізувати потреби та бізнес-процеси клієнтів спортивних закладів на прикладі фітнес-центру «Fit&Go».

8. Оцінити існуючі програмні продукти (Mindbody, Glofox) та виділити їхні сильні й слабкі сторони на ринку України.

9. Розробити технічне завдання на програмний комплекс для управління замовленнями, враховуючи специфіку українських спортивних підприємств.

10. Створити прототип бази даних і розробити основні модулі системи (реєстрація клієнтів, оформлення замовлень, модуль звітності, інтеграція з платіжним шлюзом).

Запропоноване ПЗ поєднує найкращі світові практики з особливостями ринку України. Практичну значущість дослідження становить можливість впровадження розробленого продукту в роботу як окремих невеликих студій й фітнес-центрів, так і мережевих спортивних комплексів, що сприятиме підвищенню рівня обслуговування клієнтів, зростанню прибутковості та оптимізації бізнес-процесів.

Дипломна робота складається з таких розділів:

1. Огляд літератури та аналіз існуючих рішень — опис ринку програмного забезпечення для спортивних закладів, аналіз міжнародних та локальних систем.

2. Постановка задачі та технічне завдання — визначення об’єкта, предмета дослідження, вимог до системи, виготовлення моделі даних.

3. Проектування та розробка системи — опис структури бази даних, архітектури програмного комплексу, розробка ключових модулів, інтерфейсів.

4. Економічне обґрунтування — розрахунок собівартості розробки, оцінка ефективності впровадження.

5. Висновки та рекомендації — підсумки виконаної роботи, напрями подальших досліджень.

У результаті виконання усіх поставлених задач буде створене комплексне рішення для управління замовленнями клієнтів спортивних закладів, що сприятиме підвищенню ефективності бізнесу, покращенню якості обслуговування та розширенню сервісних можливостей українських фітнес-центрів і тренувальних студій.

Об’єкт дослідження: web-сайти управління замовленнями в діяльності спортивних закладів.

Предмет дослідження: алгоритм автоматизації роботи сервісу.

Мета дослідження: створення web-сайту.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ

1.1. Характеристика задачі

Ключова ідея дипломної роботи полягає в створенні зручного, надійного та адаптивного програмного забезпечення для управління замовленнями клієнтів спортивних закладів. Проте перш за все треба зрозуміти, які конкретні етапи та завдання варто виконати. Саме ці завдання мають найбільшу роль і розкривають сутність всього проекту, вони допоможуть зрозуміти шлях та етапи для поступового прокладання шляху для розробки зручного та комплексного рішення.

Для створення реально корисного інструменту для власників фітнес-клубів, тренерів та клієнтів була створена послідовність і опис основних завдань:

Дослідження потреб і бізнес-процесів спортивних закладів

Для того щоб написати дійсно полегшуючу життя програму, спершу треба зрозуміти, як уявляють собі ідеальний процес оформлення абонемента, бронювання занять чи внесення оплати керівники та адміністратори фітнес-центрів. Без розуміння цього всі подальші кроки будуть навмання і можуть не відповідати реальним потребам.

Щоб прокачатись в цьому питанні треба провести спілкування з адміністраторами пари спортивних клубів і зібрана інформація: як вони зараз приймають замовлення, з якими труднощами стикаються, які страждають процеси найчастіше. Проаналізувати, які функції є критично важливими, а які можуть бути опціональними на першому етапі, а також задокументувати всі знайдені вузькі місця: наприклад, ручне введення даних, відсутність єдиної системи обліку чи проблеми з нагадуваннями про заняття. Таким чином, треба сформувати чітке розуміння цінності майбутнього рішення та списку конкретних проблем, які воно має вирішити.

Оцінка існуючих програмних рішень

Щоб не винаходити велосипед, варто глянути на вже наявні інструменти та реалізовані можливості. Розуміння їх сильних і слабких сторін допоможе сформувати унікальну пропозицію, уникнути поширених помилок і взяти до уваги найкращі практики.

Треба проаналізувати міжнародні рішення, звертаючи увагу на функціонал, вартість, гнучкість впровадження. Оцінити локальні продукти і дивимося як вони інтегруються з платіжними системами, чи зручний інтерфейс для клієнта. Є сенс врахувати думки реальних користувачів: що подобається, а що ні; які болі досі не закриті. Зробити підсумки всіх переваг та недоліків, щоб потім закласти у власний продукт саме той набір функцій, який справді потрібен українському ринку, і виключити зайве.

Формулювання вимог та технічного завдання

Технічне завдання (ТЗ) — це так званий «дорожній план», який допоможе зорієнтуватися, що саме потрібно реалізувати, в які терміни та в якому форматі. Без чіткого ТЗ ризик розпилити ресурси на другорядні елементи зростає в рази.

На основі результатів попередніх етапів виписуються функціональні вимоги: наприклад, інтерфейс для реєстрації користувачів, модулі бронювання, обробка платежів, система нагадувань. Окремо варто описати нефункціональні вимоги: швидкодію, надійність, безпеку, доступність через різні пристрої платформи. Також не забути розбити весь проект на мікрозавдання: створити базу даних клієнтів, налаштувати API для LiqPay/WayForPay. Дуже важливо визначити пріоритети, терміни виконання кожного блока та критерії успішності (наприклад, «зменшення часу обробки замовлення з 5 хвилин до 1» або «можливість одночасного підключення не менше 50 клієнтів без збоїв»). Цей етап формалізує всі очікування, тому надалі команда знатиме, що саме робити й в якій послідовності.

Проєктування архітектури системи та моделювання даних

Щоб програма працювала швидко й коректно, потрібна продумана структура: і бази даних, і серверної, і клієнтської частин. Без такої карти є високий ризик заплутатися в коді та витратити багато часу на непотрібні

переробки.

Спочатку створюється ER-діаграма бази даних, де показуються зв’язки між клієнтами, замовленнями, тренерами, груповими заняттями, оплатами. Проєктуємо архітектуру бекенду (Laravel-контролери, моделі, сервіси для взаємодії з платіжними шлюзами) та фронтенду (структуру веб-сторінок або екранів мобільного додатку). Також першочергово важливо визначити, які технології й бібліотеки використовуємо: фреймворк, ORM, механізм черг на повідомлення, бібліотеку для графіків звітності тощо. Варто обговорити, як забезпечити масштабованість, наприклад, можливість підключати нові клуби чи додаткові модулі без кардинальних змін у коді. Таким чином, перед початком програмування буде зрозуміло, як взаємодіють усі компоненти.

Розробка прототипу інтерфейсу та візуалізація користувацького досвіду

Сьогодні користувачі очікують інтуїтивного інтерфейсу, де можна швидко знайти потрібну дію. Прототип дозволяє на ранньому етапі побачити, чи правильно розставлені елементи, чи не заплутує система людей нескінченними кнопками й формами. Гарною ідеєю буде намалювати друковані скетчі або зробити wireframe у спеціальному інструменті — описати головну сторінку, сторінку оформлення замовлення, особистий кабінет клієнта, модуль адміністратора. Слід запропонувати різні варіанти розташування форм, меню й навігації, та обговорити, що зручніше саме для керівника спортклубу, а що — для звичайного відвідувача. За потреби часто робиться click-through демо (наприклад, у Figma чи іншому сервісі), щоб пройтися по екранах та побачити поток дій у дії. Звичайно необхідно зібрати зворотний зв’язок: показати прототип кільком колегам чи знайомим, які не мають досвіду програмування, — таким чином можна упевнитися, що все зрозуміло. На цьому етапі ви фактично можна наглядно перевірити, чи людям приємно користуватися тим, над чим ведеться розробка.

Реалізація основних модулів системи

Код — це серцевина всього проєкту. Саме тут відбувається втілення всіх попередніх напрацювань: від моделі даних до взаємодії з платіжними шлюзами.

Кожен підмодуль має працювати стабільно й відповідати вимогам ТЗ.

Почнемо с невід'ємного елемента будь-якого проекту – база даних. Створення структури таблиць, індексів, можливих тригерів для зберігання інформації про клієнтів, замовлення, оплату, тренерів, групи і таке інше. Далі іде реєстрація / аутентифікація, гарно буде налаштувати механізм реєстрації нових користувачів, авторизації, відновлення пароля. Також не треба забувати за бронювання занять, реалізується воно через поток дій: клієнт обирає час і тренера, перевіряємо наявність вільних місць, зберігається замовлення. Враховуємо, що одне й те саме заняття можуть обрати одразу кілька людей. Потім йде підключення платежів, треба підключити українські платіжні системи LiqPay / WayForPay / Fondy. Налаштовується callback, щоб сервер отримував підтвердження успішної оплати. Окремо треба врахувати сценарій відміни платежу чи повернення коштів. Ще важливою клієнтською частиною є система знижок і лояльності. Можна додати, наприклад, знижки за довгострокове абонементування, бонуси за приведеного друга, купони. Прописуються ці правила нарахування балів і знижок у базі та на фронтенді. Важливо збирати статистику, тому додається модуль, який збирає статистику: кількість замовлень за період, доходи, заповнюваність групових занять тощо. Кожен з підмодулів створюється окремим блоком, але при цьому всі вони мають коректно взаємодіяти між собою. Наприклад, якщо клієнт сплатив онлайн, інформація має одразу відображатися в загальному журналі замовлень.

Тестування розробленого програмного продукту

Навіть найкращий код потрібно перевірити: чи він не зависає, чи немає дірок у безпеці, чи коректно працює логіка бронювання. Тестування підтверджує, що програмний продукт готовий до впровадження в реальному середовищі.

Функціональне тестування це важливий елемент тестування, він дозволяє перевірити, чи виконуються всі сценарії: реєстрація, оформлення замовлення, оплата, ануляція, повернення коштів. Юзабіліті-тестування, наступний етап комплексного тестування, для нього залучається кілька реальних користувачів — адміністраторів спортклубу і відвідувачів — і просимо їх виконати типові дії.

Потім іде збір зворотнього зв’язку: зрозумілий інтерфейс, зрозуміла навігація, відсутність незручних моментів. Також не слід забувати про навантажувальне тестування, для нього імітуємо велику кількість одночасних запитів, щоб переконатися, що сервер витримує навантаження. І звісно безпекове тестування. Перевіряється захист від SQL-ін’єкцій, XSS-атак, чи коректно обробляються токени безпеки, чи дані клієнтів у базі зашифровані або хоча б надійно захищені. Також для повного релізу важливо провести тестування на відповідність законодавству. Перевіряється, чи система зберігає персональні дані за правилами, передбаченими Законом України «Про захист персональних даних».

За результатами тестування формується список знайдених дефектів і поступово вони усуваються. Тільки коли всі критичні проблеми виправлені, можна переходити до впровадження.

Впровадження та оцінка ефективності на прикладі реального спортклубу

Справжня перевірка — коли програму запускають у реальних умовах. Тільки так можна побачити, чи досягнуті цілі: економія часу адміністратора, зменшення кількості помилок, зростання кількості онлайн-замовлень тощо.

Для початку обирається пілотний заклад та укладається домовленість про тестовий період (1–2 місяці). Закладу надаються інструкції з установки або посилання на тестовий сервер. Треба навчити адміністратора користуватися новим інтерфейсом: показати, як створювати нові послуги, бронювати клієнтів, формувати звіти. Паралельно з використанням програми фіксуються певні метрики:

- Скільки часу адміністраторам потрібно на обробку одного замовлення до й після впровадження?
- Яка динаміка продажів за період (абонементи, разові заняття)?
- Як змінилася кількість «прогульників» (no-show) завдяки автоматичним SMS-нагадуванням?
- Чи зросла задоволеність клієнтів (можна провести коротке опитування або опиратися на статистику відмов)?

Після чого проводиться аналіз отриманих даних й порівнюється із

запланованими показниками ефективності. Цей етап покаже «живу» картину, наскільки проект виправдав сподівання та де ще є над чим працювати.

Обробка результатів та формулювання висновків

Лише на завершальному етапі, коли всі компоненти розроблені, протестовані й випробувані, можна підбити підсумки та зробити висновки. Саме тут оцінюється, чи досягнуто поставленої мети, які результати отримано, та даються рекомендації для подальшого розвитку проєкту.

В кінці кінців підбиваються підсумки кожного етапу: що вдалося, а що викликало складнощі. Проводиться аналіз, наскільки розроблене ПЗ вирішило проблеми, виявлені на початку: чи дійсно скоротився час обробки замовлень, чи покращилася аналітика, чи є зворотний зв’язок від адміністраторів і клієнтів позитивним. Формулюються рекомендації по допрацюванню, які функції додати в наступних версіях, як масштабувати систему на мережу з кількома філіалами.

Підсумок

Кожен крок — це не просто технічні дії, а свідомий шлях до створення рішення, яке допоможе українським фітнес-клубам і тренувальним студіям автоматизувати бізнес-процеси, заощадити час адміністраторам і зробити користування комфортнішим для клієнтів.

1.2. Огляд існуючих web-сайтів

У сучасному цифровому середовищі спортивні заклади дедалі частіше обирають готові веб-платформи для управління замовленнями, з огляду на їхню масштабованість, доступність та багатофункціональність. Щоб об’єктивно оцінити, які рішення вже представлені на ринку, дослідник ретельно вивчив кілька найпоширеніших та найвідоміших веб-сайтів, що спеціалізуються на автоматизації роботи фітнес-центрів, спортклубів і тренувальних студій. Перш ніж звернутися до власної розробки, було важливо зрозуміти, як саме побудовані ці системи, які сервіси вони пропонують користувачам та чим відрізняються один від одного з точки зору зручності, гнучкості налаштувань та вартості.

Для аналізу були обрані чотири ключові платформи: Mindbody, Glofox та кілька невеликих стартапів із локального ринку. Критерії відбору були такими: по-перше, популярність серед клієнтів (кількість підключених спортклубів); по-друге, наявність української локалізації та інтеграції з локальними платіжними системами; по-третє, масштабованість пропонованих функцій.

Перш за все, варто звернути увагу на платформу Mindbody, яка є одним із лідерів світового ринку. Сайт Mindbody було вивчено шляхом реєстрації тестового облікового запису, ознайомлення з демо-версією інтерфейсу для адміністраторів і відвідувачів студій. Під час аналізу зверталася увага на такі аспекти:

- UI/UX: наскільки інтуїтивно зрозумілий дизайн, чи легко новому користувачу знайти потрібні функції (замовлення, реєстрація на заняття, перегляд графіку тренерів).
- Каталог послуг: як організовано перелік доступних класів та абонементів, чи є можливість фільтрувати заняття за тренерами, днями тижня, типом активності.
- Оплата та тарифи: які способи оплати доступні онлайн (картки, електронні гаманці, Apple Pay), чи реалізовано систему автоматичного виставлення рахунків.
- Аналітика: чи надаються готові звіти (доходи, відвідуваність, ефективність тренерів) і наскільки легко їх отримувати.
- Інтеграції: з якими сторонніми сервісами працює платформа (які платіжні шлюзи, які CRM, чи підключена підтримка SMS-розсилок тощо).

Приклади інтерфейсу сайту Mindbody наведені нижче (Рис. 1.1., Рис. 1.2., Рис. 1.3., Рис. 1.4.)

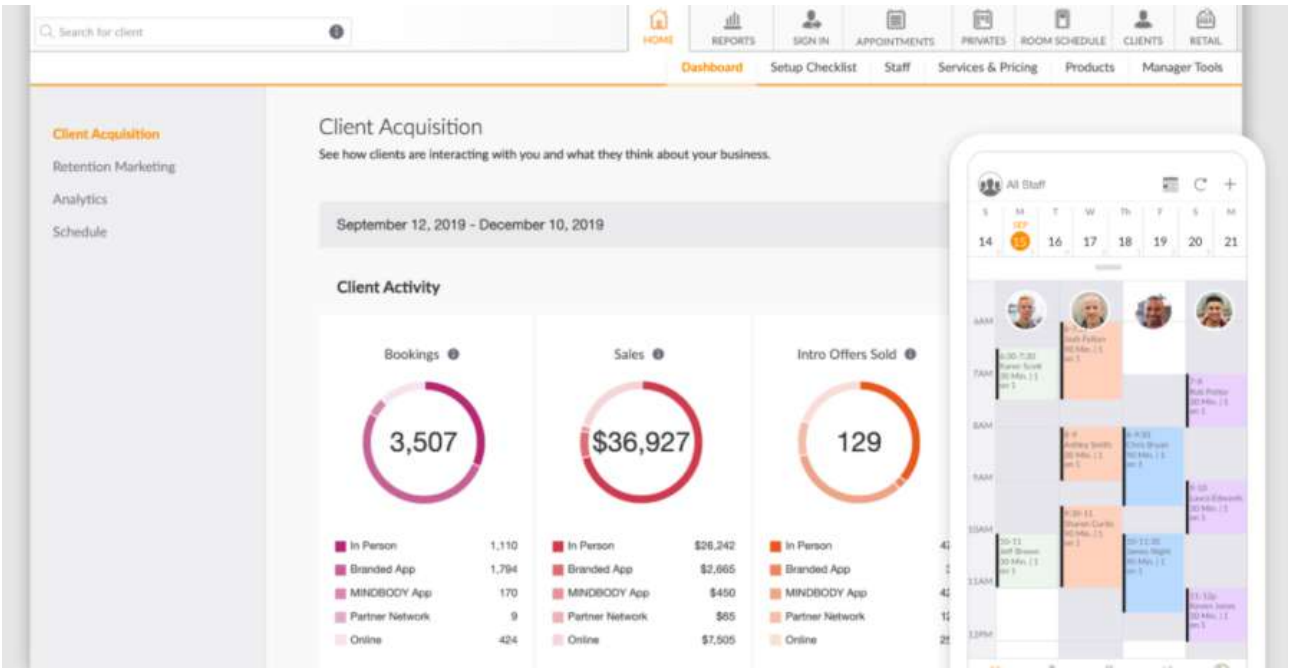

Рис. 1.1. Дашборд адміністративної частини сайту Mindbody

Рис. 1.2. Інструмент управління розкладом

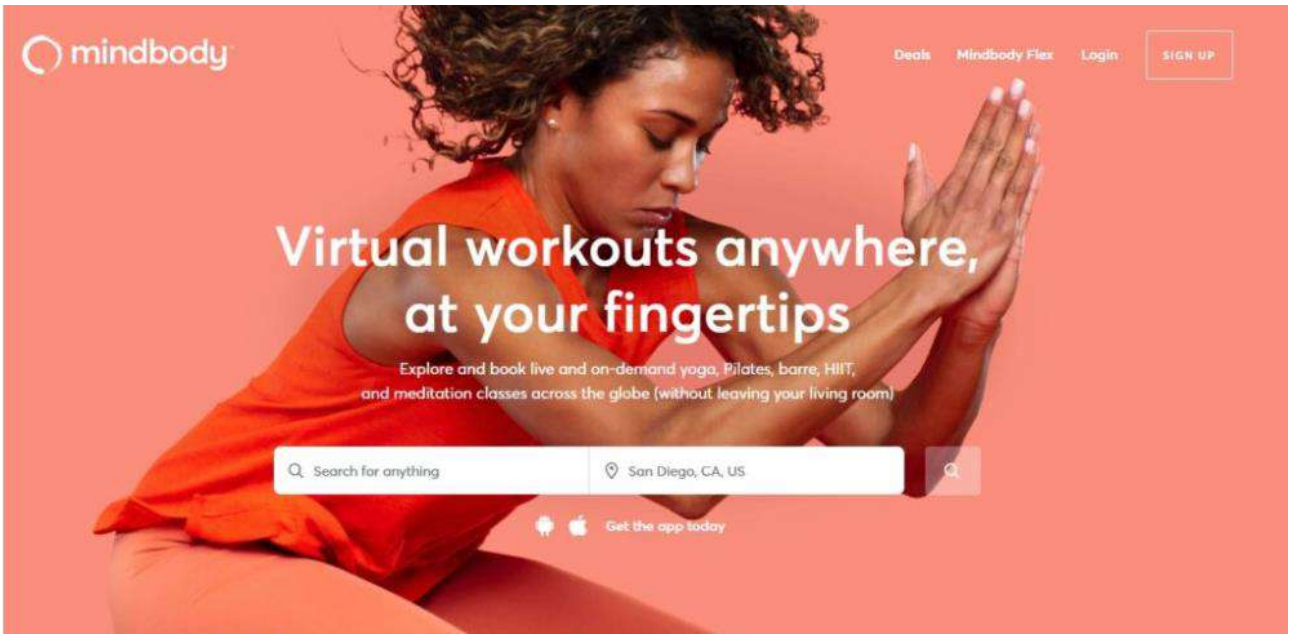

Рис. 1.3. Головна сторінка

MINDBODY PRICING - FITNESS	Starter $139 /mo GET STARTED	Accelerate $279 /mo GET STARTED	Ultimate $499 /mo GET STARTED	Ultimate Plus $699 /mo GET STARTED
List your business and services in the Mindbody app	✓	✓	✓	✓
Schedule classes, appointments, and resources	✓	✓	✓	✓
Manage clients and staff	✓	✓	✓	✓
Run comprehensive reports	Basic	✓	✓	✓
Enable mobile friendly booking and purchasing widget on your website	Add-on	✓	✓	✓
Offer self check-in for classes on an iPad kiosk	Add-on	✓	✓	✓
Streamline your intake process with digital forms	Add-on	✓	✓	✓
Reach the right audience with smart contact lists	✗	✓	✓	✓
Build emails with customizable templates	✗	✓	✓	✓
Prompt clients to post reviews	✗	✗	✓	✓
Automate your email & text marketing	✗	✗	✓	✓
Connect with clients using unlimited, 2-way, conversational SMS	✗	✗	✓	✓
Incentivize referrals	✗	✗	✓	✓
Create your own branded mobile app	✗	Add-on	Add-on	✓
Capture leads and missed calls with an AI assistant	✗	Add-on	Add-on	Add-on

Рис. 1.4. Прайс-лист сайту

У результаті виявилося, що Mindbody надає надзвичайно широкий набір інструментів: від ведення календаря занять до збірки детальної фінансової статистики. Однак суттєвими недоліками є висока вартість підписки, необхідність адаптації інтерфейсу під український ринок та відсутність нативної

підтримки деяких локальних платіжних шлюзів.

Другим об’єктом дослідження став Glofox. Цей сервіс відзначається простішим інтерфейсом, розробленим з урахуванням потреб невеликих і середніх фітнес-клубів. Під час аналізу сайт Glofox було вивчено через публічну демо-версію та огляд документації для розробників (API, SDK). З’ясувалося, що:

- Шаблони сайтів: Glofox пропонує готові шаблони web-інтерфейсу, які легко налаштовуються під бренд клієнта (кольори, логотип, фотоматеріали).
- Мобільний додаток: є можливість створити «білий лейбл» мобільної програми під власні потреби клубу.
- Гнучкість тарифікації: налаштування тарифів на заняття та абонементи за різними параметрами (кількість відвідувань, період дії, бонуси), а також програм лояльності.
- Підтримка локальних ринків: хоча Glofox орієнтований переважно на європейський ринок, там уже з’явилися користувачі з України, однак для повноцінної інтеграції з LiqPay або Fondy потрібна додаткова налаштування через API.

Зображення Glofox наведені нижче (Рис. 1.5., Рис. 1.6.)

Рис. 1.5. Дашборд адмінки Glofox

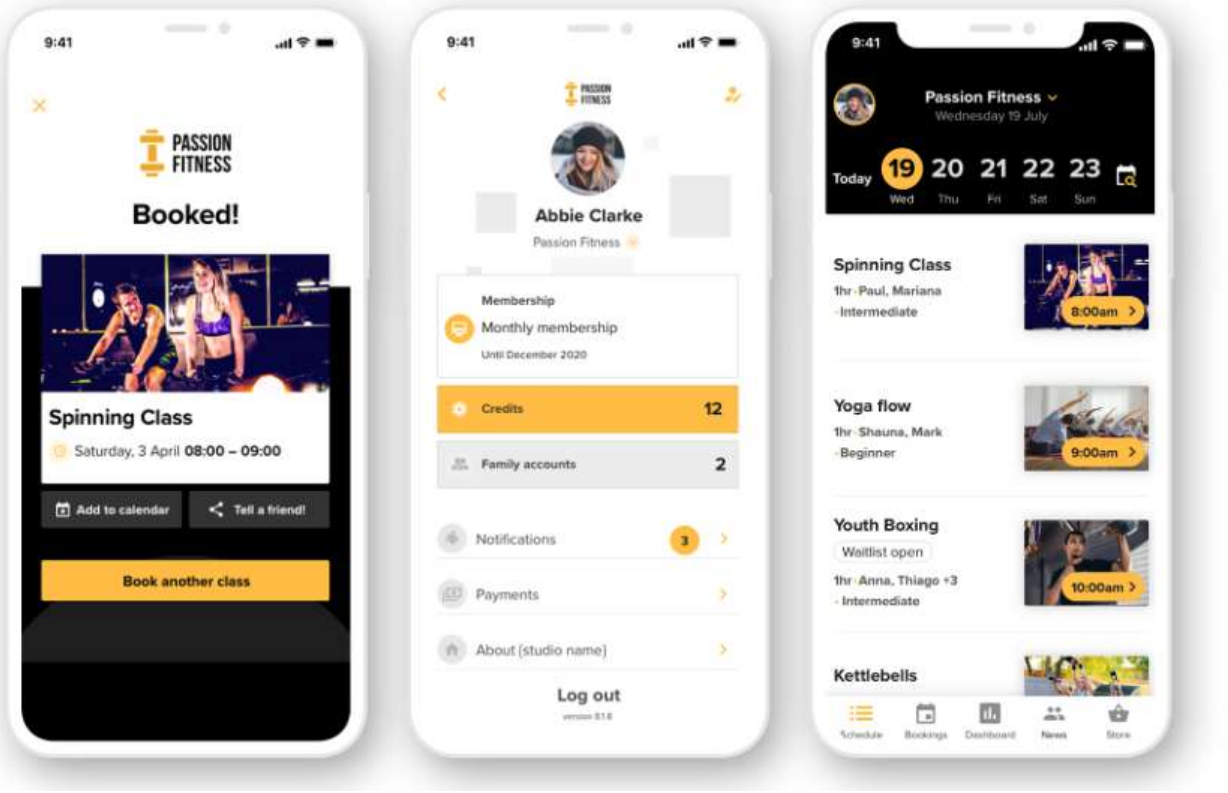

Рис. 1.6. Вигляд мобільного додатку Glofox

Основна перевага Glofox полягає в швидкому запуску та простоті адміністрування. Водночас відсутність «вбудованої» інтеграції з українськими платіжними системами змушує технічну команду знайти обхідні шляхи або замовити розробку власного модуля.

Окрім згаданих популярних платформ, дослідник звернув увагу на кілька локальних стартапів, які намагалися запропонувати компактні рішення для невеликих фітнес-студій. Найчастіше їхні веб-сайти мали обмежену функціональність, але пропонували низьку ціну або навіть безкоштовну тарифіку для перших користувачів. Усі ці сервіси були протестовані через реєстрацію демо-акаунтів, заповнення тестових даних та перевірку, наскільки швидко можна налаштувати розклад занять і прийом платежів. Загальна риса локальних проектів— це швидкий старт, але обмежені можливості масштабування та відсутність продуманої системи аналітики.

Щоб оцінити кожен із веб-сайтів якомога об’єктивніше, дотримувалася така послідовність:

1. Спочатку була проведена робота з офіційними ресурсами кожного сервісу: прочитані FAQ, переглянуті відеоінструкції, вивчені технічні вимоги до інтеграції. Це дало загальне уявлення про базовий функціонал, ставлення розробника до підтримки локальних ринків та наявні маркетингові обмеження.

2. Були створені тестові облікові записи в Mindbody, Glofox і кількох локальних платформах, щоб на власному досвіді відчути, як виглядають процеси реєстрації нових користувачів, бронювання занять, оплата послуг та робота з адміністративною панеллю. Це допомогло встановити, скільки кроків потрібно зробити адміністратору, щоби налаштувати новий спортклуб, та побачити реальні затримки або незручності інтерфейсу.

3. Для Mindbody і Glofox переглянуті незалежні огляди у тематичних форумах і відеоблогах, а також провів коротке опитування серед знайомих власників невеликих студій. Метою було дізнатися, як вони оцінюють ціну-якість, які проблеми виникають під час щоденної роботи, та чи є невирішені «болі» у цих платформах. На основі зібраних даних було складено таблицю, де порівнювалися:

- Час запуску (від реєстрації до першої броні клієнта).
- Вартість підписки (з урахуванням локальної конверсії валют).
- Наявність українськомовного інтерфейсу.
- Підтримка популярних платіжних шлюзів (LiqPay, Fondy, WayForPay).
- Обсяг доступної аналітики (щомісячні звіти, графіки відвідуваності, фінансові підсумки).
- Якість технічної підтримки (канали зв’язку, час відповіді, наявність українського сапорту).

На основі проведених тестів і зібраних відгуків сформувався перелік переваг і недоліків кожного рішення. Наприклад, Mindbody відзначено за потужність функцій, але одночасно — засуджено за складність адаптації до локальних умов. Glofox отримав позитивні відгуки за швидкість запуску й добрий користувацький досвід, проте виявився слабким у частині роботи з

українськими шлюзами.

Основні висновки огляду

Багатофункціональні іноземні платформи (Mindbody, Glofox) пропонують широкий спектр інструментів, але зазвичай не повністю адаптовані до українського ринку. Щоби працювати з українськими платіжними системами чи звітуватися перед місцевими органами, потрібні додаткові розробки або інтеграції.

Локальні рішення враховують особливості національного законодавства та інтегровані з українськими фінансовими сервісами, але часто відстають у зручності користувацького інтерфейсу і надають обмежений набір фіч.

Потреба в універсальному рішенні, яке поєднувало б переваги обох підходів: сучасний інтуїтивний інтерфейс із можливістю швидкого запуску, а також глибоку інтеграцію із локальними платіжними та законодавчими вимогами. Саме ця прогалина слугує обґрунтуванням для подальшої розробки власного програмного продукту.

Таким чином, процес аналізу web-сайтів показав, що на ринку немає готового рішення, яке б у повному обсязі задовольняло потреби як власника спортивної студії в Україні, так і його клієнтів. Саме це спонукало звернути увагу на створення власної веб-платформи, здатної швидко адаптуватися під локальні умови, надавати зручний користувацький досвід та боротися з існуючими проблемами конкурентів. За результатами огляду було визначено, які ключові модулі потрібно реалізувати в майбутньому продукті, щоб він складав конкуренцію як іноземним, так і локальним аналогам.

1.3. Аналіз вимог до web-сайту

Уявімо на хвилину, що користувач уперше заходить на веб-сайт спортивного клубу, аби записатися на тренування. Він очікує, що весь процес займе лічені хвилини: від реєстрації до остаточного підтвердження бронювання. Одночасно власник клубу сподівається отримати інструмент, який допомагатиме

йому контролювати інвентар, планувати завантаження тренерів та своєчасно отримувати звіти про фінансові показники. Саме для того, щоб поєднати ці очікування й реалізувати їх у вигляді одного веб-сайту, необхідно було чітко визначити комплекс вимог, які система мала задовольняти. Перш ніж розпочати креслити архітектуру й продумувати базу даних, дослідник провів низку інтерв'ю з керівниками спортивних закладів, адміністраторами та потенційними користувачами. Завдяки цьому з'ясувалося: користувачі «з вулиці» підходять до онлайн-бронювання зі скептичним настроєм, а адміністратори й тренери прагнуть мінімізувати ручну роботу та уникають зайвих кліків у панелі управління.

Основні функціональні вимоги визначаються наступними аспектами:

- Реєстрація та аутентифікація користувачів. Клієнт повинен мати можливість швидко створити особистий кабінет, підтвердити реєстрацію через email або SMS та ввійти до системи використовуючи email/пароль або авторизацію через соціальні мережі.
- Каталог послуг і бронювання занять. Сайт має містити зручний інтерфейс для вибору доступних групових занять, індивідуальних тренувань чи додаткових послуг (наприклад, прокат спортивного інвентарю). Клієнт повинен бачити актуальний розклад, вільні місця в кожній групі та легко додавати замовлення до корзини.
- Обробка платежів та історія транзакцій. Після вибору послуг користувач повинен отримати простий і зрозумілий механізм оплати через інтеграцію з українськими платіжними шлюзами (LiqPay, Fondy, WayForPay). Система має реєструвати статус кожного платежу, зберігати квитанції та формувати історію оплат у профілі клієнта.
- Персональні профілі та налаштування. Кожен зареєстрований клієнт має мати можливість редагувати особисті дані (контактні телефони, пошту), переглядати історію бронювань, перевіряти наявність бонусів, знижок чи абонементів і відмовлятися від броні за умов дотримання правил анулювання.
- Адміністративна панель для співробітників. Адміністратори

спортклубу повинні мати доступ до окремої частини сайту, де можна керувати каталогом послуг, змінювати розклад, додавати нових тренерів, налаштовувати акції та знижки, а також переглядати звіти з продажів. Тренери мають бачити власний графік занять, список клієнтів і статистику відвідуваності.

- Система сповіщень і нагадувань. Для мінімізації «прогульників» (no-show) передбачено надсилання автоматичних SMS та email-нагадувань про заплановані тренування. Адміністратор може налаштувати інтервали (наприклад, за 24 години та за 2 години до заняття) й текст повідомлень.

Нефункціональні вимоги охоплюють такі важливі аспекти:

- Швидкодія та масштабованість. Веб-сайт повинен коректно працювати за одночасної обробки щонайменше 100–150 запитів протягом пікової години. З огляду на сезонні та рекламні навантаження, система розрахована на динамічне збільшення кількості користувачів без значної втрати в продуктивності.

- Безпека даних. Оскільки зберігаються персональні дані клієнтів (ПІБ, номер телефону, email, історія платежів), система має відповідати вимогам Закону України «Про захист персональних даних». Для цього передбачено шифрування паролів (bcrypt), захист від SQL-ін’єкцій, XSS-атак та несанкціонованого доступу до API через застосування токенів (JWT).

- Юзабіліті та адаптивність інтерфейсу. Інтерфейс розробляється за принципом «mobile-first», тобто спочатку оптимізується під смартфони, потім під планшети й десктоп. Простий дизайн і зрозуміла логіка навігації дозволяють новачку у кілька кліків знайти потрібне заняття та оформити бронювання.

- Сумісність із локальними нормативами та обладнанням. Для зручності адміністраторів передбачена пряма інтеграція з РРО (реєстратор розрахункових операцій), що дозволяє миттєво друкувати касові чеки після онлайн-оплати або оплати на місці. Сайт підтримує українську, англійську та російську мови, автоматично підлаштовуючись під локалі користувача.

- Стабільність та доступність. Показник доступності (uptime) повинен бути не меншим ніж 99,5 % на місяць. У разі збою система автоматично

надсилатиме повідомлення технічній службі, а критичні помилки (наприклад, недоступність бази даних) — генерувати екстрені логи для швидкого усунення.

- Зручність адміністрування та оновлень. Система містить модулі «гарячого» оновлення (без зупинки сервісу), а також інтуїтивну консоль адміністрування для зміни тарифів, акцій, текстів інформаційних блоків і банерів без прямого втручання в код.

Насамперед виникла потреба в максимально гладкій процедурі реєстрації й входу. Клієнти спортивного клубу звикли, що багато сервісів дозволяють створити обліковий запис за допомогою кількох кліків: досить вказати електронну пошту, телефон або навіть скористатися обліковим записом у соціальних мережах. Вирішивши, що немає сенсу змушувати користувача вигадувати довгий пароль або довго шукати, де ввести дані, дослідник узяв за мету забезпечити можливість швидкої реєстрації через email з підтвердженням за допомогою одноразового коду, а також інтеграцію з кнопками «Увійти через Facebook» чи «Google». Однак при всій простоті досвід показав, що існує також баланс між максимальною зручністю й належним рівнем безпеки: не можна допускати, щоб пароль зберігався у відкритому вигляді чи передавався без додаткового шифрування. Саме тому, ухваливши рішення про зберігання паролів із застосуванням метода bcrypt, команда розробників мала змогу забезпечити необхідний рівень криптозахисту, не обмежуючи комфорт користувача.

Одразу після авторизації клієнт потрапляє до особистого кабінету, де йому мають бути доступні всі функції сайту. Необхідно було передбачити, що користувач бачитиме актуальний розклад групових занять і зможе ознайомитися з детальним описом кожного тренера: його кваліфікація, напрямок тренування, рейтинг від інших клієнтів. Зі свого боку, адміністратор отримує інструмент, який дозволяє легко додавати або видаляти нові заняття, регулювати кількість місць у групі, редагувати вартість абонементів і відстежувати рівень відвідуваності. Дослідник звернув особливу увагу на те, що даний модуль повинен працювати так само швидко як із десктопа, так і зі смартфона: процес вибору заняття не має відрізнятися за зручністю в мобільній версії, і жодної важливої кнопки не

повинно бути складно натиснути через маленький екран. Таким чином, поступово окреслилася ідея «мобільного-першого» підходу, який передбачає, що елементи інтерфейсу масштабуються, адаптуються під різні розміри екранів і зберігають мінімальну кількість зайвих кліків.

Не менш важливою виявилася потреба в бездоганній обробці платежів та ретельній фіксації кожної транзакції. Адміністратори спортклубів у розмові відзначали, що в багатьох випадках їм доводилося вручну перевіряти надходження коштів у банківських виписках, а клієнти скаржилися, що іноді сума стягувалася, але статус бронювання не оновлювався автоматично. На цьому етапі було вирішено інтегруватися з найбільш популярними українськими платіжними шлюзами — LiqPay, Fondy та WayForPay. Кожен з цих сервісів пропонує власний механізм підтвердження транзакції: вони надсилають відповідь на спеціальний callback-URL, де сервер повинен оперативно оновити статус замовлення у базі даних. Саме тоді клієнт одержує на екран повідомлення «Оплата пройшла успішно», а адміністратор бачить у панелі керування, що дане бронювання підтверджено. Дослідник наголошував на тому, що жодна транзакція не має залишатися «без відповіді», тому було закладено логіку повторної перевірки статусу в разі тимчасового збою: якщо платіжний шлюз не відповів миттєво, система через відкладене завдання перевіряє ще раз протягом двох хвилин, і лише у випадку остаточного невдавання помічає замовлення як «помилкове» або «відхилене».

Коли клієнт уже має зареєстрований профіль і може сплачувати за обрані послуги, важливо, щоб він чітко бачив історію власних дій на сайті. Звичаї клієнтів доводили: хтось заходить подивитися, чи справді він сплатив за абонемент, хтось нагадує сам собі, у який день була остання бронь, а дехто хоче перевірити свій баланс бонусів чи поточний рівень накопиченої знижки. Для цього дослідник родинав ідею збереження кожного кроку користувача в окремій таблиці бази даних — але не лише з точки зору технічного зберігання. Паралельно у візуальній частині сайту було продумано оформлення розділу «Моє портфоліо», де історія бронювань підкріплюється мініатюрними іконками видів

тренувань, а статус оплати відзначається кольором, що дає клієнту швидкий візуальний зворотний зв’язок. Крім того, якщо користувач захоче відмовитися від бронювання, система в автоматичному режимі перевіряє такі умови, як час до початку заняття (щоб не допускати анулювання за кілька хвилин чи за пів години до старту) та можливість повернення коштів чи нарахування бонусів. Якщо правила попередньо не дозволяють ануляцію, користувач отримає сповіщення «Бронювання не можна скасувати менш ніж за 12 годин до заняття».

Адміністраторська частина веб-сайту теж була ретельно продумана з урахуванням того, що власнику клубу та його команді потрібно часто змінювати умови: додавати нових тренерів, корегувати розклад або проводити рекламні акції. Тому було враховано, що панель управління неодмінно повинна містити так звану «дошку оголошень», на якій адміністратор бачить запити від клієнтів (наприклад, побажання додати додатковий вечірній слот), повідомлення про платежі, що з невідомих причин затрималися, та завчасну статистику про те, наскільки заповнений зал протягом тижня. Саме цей елемент дозволяє команді клубу отримувати короткий огляд «зверху», не вдаючись до глибокого входу у кожну окрему таблицю бази даних. Додатково в панелі передбачено просте поле для редагування текстових блоків на головній сторінці — щоб адміністратор міг змінити вітальний банер або оновити опис нової послуги без залучення веб-розробників.

Коли ж ішлося про систему сповіщень і нагадувань, дослідник спирався на досвід, отриманий під час огляду конкурентних платформ. Було помічено, що багато існуючих рішень надсилають лише одне повідомлення за 24 години до заняття, тоді як деяким клієнтам зручніше отримати другий «пінг» безпосередньо за дві години до початку тренування. Саме тому було вирішено зробити можливим адміністрування інтервалів і тексту кожного сповіщення окремо. Адміністратор у веб-інтерфейсі може ввести свій унікальний слоган чи додати корисну інструкцію перед першим повідомленням, а для кожного клієнта система автоматично підставить його ім’я: «Привіт, Олено! Нагадуємо, що сьогодні о 18:00 у тебе заняття з Йоги. Чекаємо на тебе!». У тому випадку, якщо тренування

було скасовано або перенесено, система одразу розсилає інформацію про новий час, а клієнт отримує змогу підтвердити свою присутність або обрати інший слот у режимі «два кліки».

Таким чином, аналіз вимог до веб-сайту охопив не тільки звичні технічні моменти, але й глибокий розгляд побажань усіх зацікавлених сторін: від власників тренерського бізнесу до кожного клієнта. Як наслідок, система мала стати не просто засобом бронювання, а цілісною екосистемою для взаємодії всіх учасників процесу — з високим рівнем безпеки, чітко налаштованою логікою бізнес-правил та приємним інтерфейсом, що дає відчуття «тепла» і довіри з першого кліку. У наступному розділі буде показано, як саме ці вимоги лягли в основу алгоритмічного проєктування та вибору архітектури майбутнього веб-сайту.

Висновки до розділу

У першому розділі було обґрунтовано актуальність розробки веб-сайту для управління замовленнями в спортивних закладах: сучасні фітнес-студії й тренажерні зали потребують гнучкого та автоматизованого інструменту, який поєднує зручність для клієнта і повний контроль бізнес-процесів для адміністратора. Через аналіз існуючих рішень (Mindbody, Glofox, 1С:Фітнес-Клуб та локальних стартапів) виявлено явний розрив між багатофункціональністю зарубіжних платформ і локальною адаптацією під українські законодавчі та платіжні вимоги. Чітке формулювання об’єкта, предмета й мети дослідження, а також послідовність завдань — від вивчення потреб «Fit&Go» до тестування прототипу в пілотному режимі — заклало міцну основу для подальших етапів проєкту.

РОЗДІЛ 2
ПРОЕКТУВАННЯ WEB-САЙТУ

У процесі створення веб-сайту для управління замовленнями клієнтів спортивних закладів перед дослідником постало завдання знайти таку послідовність дій і умов, яка б забезпечила безперебійне виконання всіх бізнес-процесів: від моменту, коли відвідувач заходить на головну сторінку, до того, як адміністратор отримує фінансову звітність і закриває зміну. Розробка алгоритму стала свого роду «костяком», який об’єднав у собі різні сценарії: користувацьку взаємодію (UI/UX), обробку транзакцій, роботу з базою даних і взаємодію з сторонніми сервісами (платіжними шлюзами й SMS-провайдерами). Логіка алгоритмів логічно поділилася на кілька взаємопов’язаних дисциплін: обробка реєстрації та аутентифікації, керування каталогом послуг і розкладом, обробка замовлень (від моменту вибору до остаточного підтвердження оплати), надсилання сповіщень та автоматична синхронізація даних між клієнтською й адміністративною частинами системи.

Перш ніж перейти до деталізації кожного з наведених сценаріїв, дослідник сформував загальну ієрархічну структуру алгоритму. На найвищому рівні вона складається зі вступної перевірки — визначення, чи зареєстрований користувач, а якщо ні, — перенаправлення до модуля реєстрації. Після цього відбувається завантаження «домашньої сторінки» з переглядом доступних видів тренувань. Якщо відвідувач обирає послугу, алгоритм переходить до етапу формування замовлення й розрахунку вартості. Далі логіка розгалужується: якщо клієнт має дійсний абонемент або достатній баланс бонусів, система автоматично застосовує відповідну знижку чи списує бонуси, і лише наприкінці — відправляє користувачеві прохання підтвердити оплату чи використання абонементу. Якщо оплата потрібна, алгоритм у межах того ж сеансу взаємодіє з платіжним шлюзом, чекаючи на підтвердження транзакції. За успішного результату — записує замовлення в базу з ознакою «підтверджено», ініціює відправку нагадування (чи двох нагадувань) і повертає клієнтові повідомлення про остаточну успішну

реєстрацію. У разі помилки оплати — алгоритм пропонує повторити спробу або звернутися до адміністратора, фіксує в журналі невдалу транзакцію, а саму бронь позначає як «непідтверджену».

Тепер розглянемо більш докладно кожен із ключових етапів алгоритмічної логіки.

Реєстрація та аутентифікація

Коли відвідувач потрапляє на сайт, алгоритм першочергово перевіряє наявність дійсної сесії. Якщо сесія відсутня, користувач потрапляє на сторінку входу, де йому пропонується обрати реєстрацію за email-адресою чи соціальними мережами. Алгоритм реєстрації починається з отримання введених даних (email, телефон, ім’я та обраного пароля). Після цього дані найперше перевіряються на відповідність базовим правилам (коректний формат email, строка пароля достатньої довжини). У разі правильності введених значень формується об’єкт користувача і записується в базу даних із хешованим (“bcrypt”) паролем. Наступним кроком алгоритму є відправлення на вказаний email одноразового коду або посилання для підтвердження. Поки користувач не підтвердить свій email, алгоритм блокує доступ до всіх функцій замовлення. Лише після успішної верифікації обліковий запис позначається як «активний».

Після активації аккаунта відбувається автоматичне створення особистого кабінету користувача й ініціюється базове налаштування профілю: завантажується статистика минулих бронювань (якщо вони були у випадку імпорту даних із зовнішньої системи), ініціалізуються змінні налаштувань — сповіщення за замовчуванням, вибір мови інтерфейсу. Якщо користувач обирає вхід через соціальну мережу (Google чи Facebook), алгоритм спочатку звертається до API відповідного провайдера, отримує підтвердження справжності користувача, визнає email як ключ і за потреби створює новий профіль у внутрішній базі. Якщо такий email вже є в системі, алгоритм просто активує сесію без необхідності повторної реєстрації.

Вибір послуги та формування замовлення

У момент, коли зареєстрований користувач авторизується і заходить на

сторінку з каталогом послуг, алгоритм динамічно генерує перелік доступних групових занять, індивідуальних тренувань чи інших опцій (наприклад, оренда інвентарю) на основі актуального розкладу, заданого адміністрацією. Тут система звертає увагу на добу тижня, конкретний час і вже існуючі бронювання, тож користувач бачить тільки ті слоти, де ще залишилися вільні місця. Як тільки клієнт натискає на конкретну послугу, алгоритм завантажує деталі заняття: опис тренера, рівень складності, необхідне обладнання, орієнтовну тривалість та вартість. На цьому етапі алгоритм також перевіряє, чи не існує у користувача активного конфлікту (наприклад, якщо клієнт спробує забронювати одночасно дві групові тренування в той самий інтервал часу, алгоритм видасть попередження і не дозволить перейти далі до оплати).

Якщо конфліктів не виявлено, алгоритм деактивує можливість одночасної зміни кількості місць у цій групі з боку інших користувачів (через механізм блокування рядків у базі даних або використання транзакцій у СУБД). Це запобігає одночасному перехопленню останнього місця двома клієнтами. Після того, як блокування займе останнє місце або ж залишаться вільні слоти, алгоритм формує об’єкт «Замовлення» та передає цю інформацію далі — у модуль розрахунку вартості. Саме під час цього етапу система перевіряє, які особисті знижки чи бонуси містить профіль користувача: якщо дійсний купон або накопичені бали дозволяють отримати пільгу, алгоритм автоматично перераховує остаточну суму, а на екрані клієнта відображається діалог із вибором остаточного методу оплати: «Оплатити онлайн» або «Використати абонемент». У результаті на екрані з’являється кнопка для переходу до платіжного шлюзу або підтвердження використання абонемента без проведення додаткової оплати.

Обробка оплати

Коли користувач обирає сплатити онлайн, алгоритм замикає об’єкт «Замовлення» на певний час (наприклад, 15 хвилин), протягом якого клієнт має завершити транзакцію. Далі він формує запит до обраного платіжного шлюзу (LiqPay, Fondy чи WayForPay), передаючи основні параметри: унікальний ідентифікатор замовлення на стороні системи, остаточну суму до списання,

валюту, callback-URL для зворотного підтвердження. Після переходу клієнта на сторінку платіжного шлюзу алгоритм вичікує подію «успішна транзакція» чи «помилка транзакції» у вигляді HTTP-виклику (webhook) із відповідними даними. Поки триває оплата, сервер періодично опитує статус транзакції (якщо платіж не повернув результат миттєво) і, залежно від відповіді, змінює статус «Замовлення» у базі: або на «Підтверджено», або на «Відхилено». Якщо платіж успішний, алгоритм негайно розблоковує місце в групі (бо вже заброньовано), відправляє клієнту підтверджувальну sms-повідомлення й email-лист із деталями заняття. Якщо ж платіж не пройшов, алгоритм повертає користувачу на сторінку з помилкою і пропонує повторити спробу або обрати інший метод оплати (або повідомити адміністратора).

Підтвердження замовлення та сповіщення

У випадку успішної оплати алгоритм переходить до етапу підтвердження замовлення. Він позначає у базі даних статус замовлення як «Активне», відправляє одночасно два види оповіщень: одне — адміністраторам (через внутрішню панель) із темою «Нове підтверджене замовлення», інше — клієнту з детальною інформацією про дату, час і тренера. Далі алгоритм розраховує час для автоматичного надсилання нагадувань: за 24 години до заняття й за 2 години до заняття. Після цього він записує у таблицю «Нагадування» два запити — один із міткою часу «T–24 години», інший — «T–2 години», а коли календарний час досягає зазначених міток, таймер на бекенді спрацьовує й ініціює відправлення SMS чи email. Якщо ж адміністратор у проміжку змінює розклад або тренер не може провести заняття (наприклад, через хворобу), алгоритм автоматично коригує статус «Замовлення» або відправляє клієнту новий час прямісінько з панелі адміністратора, не чекаючи задвоєних вручних дій.

Адміністративні процеси

Для кожного тренера та адміністратора передбачено окремі підпроцеси. Наприклад, коли адміністратор додає новий час групового заняття в розклад, алгоритм провокує міграцію у базі: він додає новий запис у таблицю «Групові заняття» із параметрами (дата, час, кількість місць, ідентифікатор тренера). Якщо

в цей момент користувачі онлайн дивляться розклад, алгоритм через WebSocket чи механізм «довгих запитів» (long polling) надсилає сигнал клієнту, що «розклад змінено», і автоматично оновлює відображення доступних слотів. Якщо адміністратор змінює деталі заняття (наприклад, переносить його на інший зал або зменшує кількість місць), алгоритм перевіряє, чи існують підтверджені замовлення на цю групу; якщо так, то спочатку скеровує сповіщення цим клієнтам із пропозицією змінити час або відмінити бронь із виплатою компенсації. Тільки після того, як клієнти отримали повідомлення або відмовилися, алгоритм остаточно редагує дані в таблиці.

Подібним чином, коли адміністратор формує звітність, алгоритм реалізує процедури збору даних: він проходить по всіх замовленнях за обраний період, підсумовує суми та створює агреговані статистичні дані (наприклад, кількість проданих абонементів, кількість підтверджених замовлень, середній дохід за один день). Результат алгоритму подається у вигляді JSON-об’єкта, який клієнтська частина відображає у вигляді діаграм і таблиць. Коли ж потрібен експорт у PDF чи Excel, алгоритм формує цей документ у фоновому режимі та надсилає адміністратору посилання на завантаження.

Обробка помилок і виключних ситуацій

У кожному з етапів алгоритм передбачає перевірку помилок. Наприклад, якщо базі даних не вдалося записати нове замовлення через тимчасову відсутність з’єднання, алгоритм створює чергу у Redis і регулярно намагається повторити запис, інформуючи адміністратора лише після трьох невдалих спроб. Якщо від платіжного шлюзу немає відповіді понад дві хвилини, алгоритм вважає транзакцію зупиненою й переводить замовлення у статус «Потребує перевірки». У випадку виникнення конфліктів з блокуванням місця (якщо двоє користувачів одночасно пробують забронювати останнє місце), алгоритм обирає першим того, чий запит прийшов раніше, а другий клієнт отримує інформацію «Наразі всі місця зайняті, спробуйте інший час».

Таким чином, розроблений алгоритм охопив увесь цикл взаємодії між користувачем і системою: від моменту створення облікового запису до миттєвого

оновлення розкладу й генерації фінансової звітності. У результаті вийшов комплекс взаємопов’язаних процедур, кожна з яких має чітко прописані вхідні й вихідні дані, точки відмови та механізми відновлення після збоїв.

Після того як було детально прописано алгоритми ключових сценаріїв взаємодії користувачів із системою, наступним кроком стало відшукання такої архітектурної моделі, яка б дозволила ці алгоритми реалізувати з мінімальними ризиками для подальшого розвитку і підтримки. Уявімо собі, що архітектура — це каркас будинку: за її допомогою визначаються «несучі стіни», розташування «інженерних комунікацій» і місце для «декоративних елементів». У нашому випадку каркасом стала трирівнева модель: клієнтський рівень (front-end), серверний рівень (back-end) і рівень зберігання даних (database & services).

На клієнтському рівні було ухвалено рішення відокремити логіку інтерфейсу від бізнес-логіки за допомогою сучасного JavaScript-фреймворку. Завдяки цьому можна забезпечити миттєве оновлення сторінок без перезавантаження, гнучке масштабування компонентів і легку розробку мобільної версії сайту. Інтерактивні елементи, як-от календар з вибором часу або форма оплати, відгукуються на дії користувача негайно завдяки асинхронним запитам до API. Візуально фронтенд відділений від бекенду чистим REST-інтерфейсом, що дозволяє в майбутньому розгорнути мобільний додаток, який використовуватиме ті самі ендпоїнти.

Серверний рівень спроектовано на базі MVC-підходу з використанням сучасного PHP-фреймворку. Контролери «приймають» HTTP-запити, сервіси обробляють бізнес-логіку згідно з алгоритмами, а моделі впорядковують взаємодію з базою даних. Такий розподіл обов’язків гарантує, що кожен програмний модуль за своєю суттю невеликий і легко тестується окремо. Наприклад, модуль аутентифікації відповідає лише за перевірку прав користувача і випуск JWT-токенів, модуль бронювання працює з таблицями розкладу та замовлень, а платіжний модуль спілкується виключно з API шлюзів і обробляє webhooks. Усі модулі об’єднані через шар сервісів, який виконує транзакції, керує чергами (Redis) для повторної перевірки статусу платежів та надсилає події на

відправку SMS і email.

Рівень зберігання даних базується на реляційній СУБД, обраній виходячи з критерію швидкої обробки складних запитів. Для прискорення читання «статичних» довідників використовується кешування в Redis, що значно знижує навантаження на основну базу. Міграції й структурні зміни проводяться через систему версіонування схеми, що забезпечує можливість «гарячого» оновлення без простоїв у роботі.

Ключовим принципом архітектури стало розмежування відповідальностей: клієнт відповідає за презентацію даних, бекенд — за правила бізнес-логіки, а база — за консистентність і цілісність інформації. Така парадигма дозволяє розробляти кожен модуль окремо, підключати нові сервіси без кардинальної переробки решти системи та забезпечує можливість горизонтального масштабування під зростаючу кількість користувачів. Усі ці рішення лягли в основу технічного завдання та стали відправною точкою для написання коду в наступному розділі.

Висновки до розділу

У другому розділі було деталізовано алгоритмічну логіку взаємодії користувача й системи, розроблено блок-схеми ключових сценаріїв (реєстрація, вибір послуги, оплата, сповіщення) та обрано трирівневу архітектуру. Використання REST-API, поділ на модулі (автентифікація, бронювання, платіжний шлюз, черги в Redis), асинхронні події та WebSocket-оповіщення гарантують швидку реакцію інтерфейсу та надійну обробку запитів. Такий підхід дозволяє масштабувати систему, і забезпечує чітке розмежування відповідальностей між компонентами.

РОЗДІЛ 3
ПРОЕКТУВАННЯ БАЗИ ДАНИХ WEB-САЙТУ

3.1. Обгрунтування вибору СУБД

У процесі проєктування бази даних для веб-сайту управління замовленнями спортивних закладів команда розробників зупинилася саме на MySQL. Рішення про цей вибір формувалося поступово, аби максимально врахувати як поточні вимоги проєкту, так і перспективи його розвитку.

По-перше, MySQL є перевіреним роками рішенням із сильними позиціями на ринку веб-додатків. Завдяки десятиліттям активної підтримки спільнотою та компанією Oracle, він пропонує повний набір інструментів для реалізації реляційної моделі даних: суворі транзакції з гарантіями ACID, контроль цілісності даних через зовнішні ключі й тригери, а також гнучкі можливості індексування. Саме ці властивості стали вирішальними для обробки фінансових операцій на сайті — від моменту реєстрації замовлення до остаточного збереження результату оплати.

По-друге, MySQL добре інтегрується з обраним стеком технологій. Серверна частина проєкту побудована на PHP-фреймворку, у якому є готові ORM-шари та механізми міграцій, що безшовно працюють із MySQL. Це дозволяє автоматизувати версіонування схеми — додавати або змінювати таблиці й індекси в середовищі розробки, а потім так само безпечно вивантажувати ці зміни на продакшн-сервер. Такий підхід мінімізує ризик ручних помилок і скорочує час на супровід бази даних у процесі оновлень.

Третій аргумент пов’язаний із продуктивністю та масштабованістю. MySQL за останні роки значно покращив механізм зберігання InnoDB: він оптимізує локування рядків і дає можливість налаштувати параметри кешування, що особливо важливо для сайтів із високим навантаженням у пікові години. У проєкті передбачена достатня кількість одночасних з’єднань — коли одночасно сотні клієнтів роблять запити до системи бронювання або формують фінансові

звіти, — тому можливість тонкої настройки буферів і налаштувань реплікації MySQL гарантує стійкість і швидкість обробки даних.

Також важливо, що MySQL легко масштабувати “горизонтально”: у міру зростання аудиторії можна активувати реплікацію на віддалені слейв-сервери, перерозподіливши навантаження на читання звітів та статистики. У майбутньому, якщо кількість клубів та користувачів зросте в десятки разів, можна буде додати додаткові репліки, а також застосувати шарований (sharded) підхід — усе це підтримується в екосистемі MySQL та родинних рішеннях, таких як MariaDB чи Percona.

Нарешті, у користь вибору MySQL зіграли економічні та організаційні чинники. Ця СУБД поширюється під безкоштовною ліцензією GPL, тому не потребує додаткових витрат на ліцензії. Крім того, у команді були фахівці, які вже мали досвід управління MySQL-кластерами та оптимізації запитів, що зменшило криву навчання і дозволило швидко розгорнути перші версії бази даних із дотриманням усіх найкращих практик.

Таким чином, MySQL виявився найбільш збалансованим вибором із погляду надійності, продуктивності, інтеграції з обраним стеком і подальших перспектив розширення. Саме він став основою для побудови структури таблиць, визначення зв’язків між ними та налаштування механізмів резервного копіювання і реплікації, що детально розкриється в наступних підрозділах.

3.2. Структура таблиць бази даних

У цьому підрозділі наведено основні таблиці, які лягли в основу реляційної СУБД MySQL, вибраної для проєкту. Їхня структура та зв’язки повністю відображають предметну модель: від обліку користувачів і компаній до ведення замовлень та розкладу занять.

Таблиця users

Ця таблиця містить облікові записи усіх користувачів системи – як клієнтів, так і співробітників спортивних закладів. В ній знаходиться основна інформація

о користувачах, така як: ім’я, прізвище, телефон, день народження. Структура таблиці «users» наведена у табл. 3.1.

Таблиця 3.1

Структура таблиці «users»

Назва	Тип	Атрибути	Опис
id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NOT NULL	Первинний ключ
first_name	VARCHAR(255)	NOT NULL	Ім’я
last_name	VARCHAR(255)	NOT NULL	Прізвище
email	VARCHAR(255)	NULL, DEFAULT “”	Адреса електронної пошти
email_verified_at	TIMESTAMP	NULL	Чи веріфікований email
password	VARCHAR(255)	NULL	Хеш пароля
phone	VARCHAR(15)	NOT NULL, UNIQUE	Номер телефону
created_by	VARCHAR(255)	NULL	Ким створено
remember_token	VARCHAR(100)	NULL	Токен для відновлення сесії
created_at	TIMESTAMP	NULL	Дата й час створення запису
updated_at	TIMESTAMP	NULL	Дата й час останнього оновлення
birthday	DATE	NULL	Дата народження

Таблиця companies

Компанії (спортивні клуби), які оперують у системі. Структура таблиці «companies» наведена у табл. 3.2.

Таблиця 3.2

Структура таблиці «companies»

Назва	Тип	Атрибути	Опис
id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NOT NULL	Первинний ключ
uuid	VARCHAR(255)	NOT NULL	Додаткове id локації
available_from	DATETIME	NOT NULL	Доступна з
available_to	DATETIME	NOT NULL	Доступна до
created_at	TIMESTAMP	Дата створення	Дата й час створення запису
updated_at	TIMESTAMP	Дата останнього оновлення	Дата й час останнього оновлення
owner_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NOT NULL, FOREIGN	Id власника
title	VARCHAR(255)	NOT NULL	Назва
created_by	VARCHAR(255)	NOT NULL	Ким створен
description	VARCHAR(255)	NULL	Опис
tariff_id	INT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NOT NULL, FOREIGN	Id тарифу
image	VARCHAR(255)	NULL	Зображення

Таблиця company_locations

Розрізнення філіалів у межах однієї компанії (якщо клуб має декілька адрес). В ній знаходиться інформація про локацію яка відноситься до певної компанії, така як: id компанії, до якої відностьс локація(для того щоб визначити зв’язок між сутностями), назва, опис, адреса, електронний поштовий адрес, телефони через які можна зв’язатися з представниками локації та інш. Структура таблиці «company_locations» наведена у табл. 3.3.

Таблиця 3.3

Структура таблиці «company_locations»

Назва	Тип	Атрибути	Опис
id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NOT NULL	Первинний ключ
created_at	TIMESTAMP	NULL	Дата й час створення запису
updated_at	TIMESTAMP	NULL	Дата й час останнього оновлення
company_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NOT NULL	Id компанії
title	VARCHAR(255)	NOT NULL	Назва
description	VARCHAR(255)	NULL	Опис
created_by	VARCHAR(255)	NULL	Ким створено
address	VARCHAR(255)	NULL	Адрес
email	VARCHAR(255)	NULL	Email адрес
phones	LONGTEXT	NULL	Телефони
image	VARCHAR(255)	NULL	Зображення
start_time	VARCHAR(10)	NULL	Відчиняється
end_time	VARCHAR(10)	NULL	Зачиняється
deleted_at	TIMESTAMP	NULL	Дата й час видалення

Таблиця company_staffs

Працівники клубу: адміністратори та тренери. В ній знаходиться інформація про працівників які відносяться до певної компанії, така як: id компанії, до якої відностьсяся працівник (для того щоб визначити зв’язок між сутностями), id користувача, роль в компанії, ціна як тренера (якщо працівник тренер), картинка, особистий опис працівника, ким він створений/коли створений та теги приставлені до працівників. Структура таблиці «company_staffs» наведена у табл. 3.4.

Таблиця 3.4

Структура таблиці «company_staffs»

Назва	Тип	Атрибути	Опис
id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NOT NULL	Первинний ключ
created_at	TIMESTAMP	NULL	Дата й час створення запису
updated_at	TIMESTAMP	NULL	Дата й час останнього оновлення
company_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NOT NULL	Id компанії
user_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NOT NULL	Id користувача
current_location_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NULL	Опис
created_by	VARCHAR(255)	NULL	Ким створено
role_in_company	INT	NOT NULL	Адрес
trainer_price	INT	NOT NULL, DEFAULT 0	Email адрес
deleted_by	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NULL	Телефони
image	VARCHAR(255)	NULL	Зображення
delete_reason	VARCHAR(10)	NULL	Відчиняється
bio	TEXT	NULL	Зачиняється
deleted_at	TIMESTAMP	NULL	Дата й час видалення
tags	TEXT	NULL	Теги

Таблиця clients

Клієнти спортивного закладу (окремо від працівників). Структура таблиці

«clients» наведена у таблиці 3.5.

Таблиця 3.5

Структура таблиці «clients»

Назва	Тип	Атрибути	Опис
id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NOT NULL	Первинний ключ
created_at	TIMESTAMP	NULL	Дата й час створення запису
updated_at	TIMESTAMP	NULL	Дата й час останнього оновлення
first_name	VARCHAR(255)	NOT NULL	Ім’я
last_name	VARCHAR(255)	NOT NULL	Прізвище
email	VARCHAR(255)	NULL	Адреса електронної пошти
created_by_user_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, , NULL	Чи веріфікований email
password	VARCHAR(255)	NULL	Хеш пароля
phone	VARCHAR(15)	NOT NULL, UNIQUE	Номер телефону
created_by	VARCHAR(255)	NULL	Ким створено
image	VARCHAR(255)	NULL	Зображення
birthday	DATETIME	NULL	Відчиняється
remember_token	VARCHAR(100)	NULL	Зачиняється
sex	TINYINT(1)	NOT NULL, DEFAULT 1	Дата й час видалення
status_id	INT	NOT NULL, DEFAULT 0	Теги
deleted_at	TIMESTAMP	NULL	Дата й час видалення

Таблиця tariffs

Тарифи та послуги, що пропонуються клієнтам. В ній знаходиться інформація про тарифи така як: назва, ціна. Структура таблиці «tariffs» наведена у таблиці 3.6.

Таблиця 3.6

Структура таблиці «tariffs»

Назва	Тип	Атрибути	Опис
id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, NOT NULL	Первинний ключ
created_at	TIMESTAMP	NULL	Дата й час створення запису
updated_at	TIMESTAMP	NULL	Дата й час останнього оновлення
title	VARCHAR(255)	NOT NULL	Опис
created_by	VARCHAR(255)	NULL	Первинний ключ
price	INT	NOT NULL, DEFAULT 0	Дата й час створення запису

Таблиця subscriptions

Записи про куплені клієнтами абонементи. Тут знаходиться вся інформація по абонементах, така як: id компанії, до якої відностьсся абонемент, чи ак (для того щоб визначити зв’язок між сутностями), чи активний він, назва, ціна, чи входить в нього тренер, зображення, чи можна його замовити онлайн, чи входить в нього нутріоніст, коли він був створений та оновлений та інш. Структура таблиці «subscriptions» наведена у таблиці 3.7.

Таблиця 3.7

Структура таблиці «subscriptions»

Назва	Тип	Атрибути	Опис
id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NOT NULL	Первинний ключ
created_at	TIMESTAMP	NULL	Дата й час створення запису
updated_at	TIMESTAMP	NULL	Дата й час останнього оновлення
deleted_at	TIMESTAMP	NULL	Дата й час видалення
company_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, NOT NULL	Id компанії
available_from	DATETIME	NOT NULL	Доступна з
available_to	DATETIME	NOT NULL	Доступна до
is_active	TINYINT(1)	NOT NULL, DEFAULT 1	Хеш пароля
title	VARCHAR(255)	NOT NULL	Опис
created_by	VARCHAR(255)	NULL	Ким створено
can_be_selected_by_staff	TINYINT(1)	NOT NULL, DEFAULT 1	Зображення
price	INT	NOT NULL	Дата й час створення запису
old_price	INT	NOT NULL, DEFAULT 0	Зачиняється
validity	INT	NOT NULL, DEFAULT 0	Дата й час видалення
has_trainer	TINYINT(1)	NOT NULL, DEFAULT 0	Теги

Таблиця orders

Замовлення на окремі заняття або послуги. Структура таблиці «orders»

наведена у таблиці 3.8.

Таблиця 3.8

Структура таблиці «orders»

Назва	Тип	Атрибути	Опис
id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NOT NULL	Первинний ключ
created_at	TIMESTAMP	NULL	Дата й час створення запису
updated_at	TIMESTAMP	NULL	Дата й час останнього оновлення
company_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, NOT NULL	Id компанії
company_location_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, NOT NULL	Id компанії
client_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, NOT NULL	Доступна з
payed_at	DATETIME	NULL	Доступна до
payed_by	INT	NULL	Хеш пароля
payment_source	INT	NOT NULL	Опис
created_by	VARCHAR(255)	NOT NULL	Ким створено
order_item_info	LONGTEXT	NULL	Зображення
price	INT	NOT NULL	Дата й час створення запису
created_by_user_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, NOT NULL, DEFAULT “0”	Зачиняється

Таблиця lessons_calendar

Розклад групових занять. Структура таблиці «lessons_calendar» наведена у

таблиці 3.9.

Таблиця 3.9

Структура таблиці «lessons_calendar»

Назва	Тип	Атрибути	Опис
id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NOT NULL	Первинний ключ
created_at	TIMESTAMP	NULL	Дата й час створення запису
updated_at	TIMESTAMP	NULL	Дата й час останнього оновлення
company_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, NOT NULL	Id компанії
lesson_schedules_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, NOT NULL	Id компанії
date	DATE	NOT NULL	Доступна з
time	TIME	NOT NULL	Доступна до
location_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, NOT NULL	Хеш пароля
trainer_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, NULL	Опис
sports_hall_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, NULL	Ким створено
title	TEXT	NOT NULL	Зображення
price	DECIMAL(8,2)	NOT NULL	Дата й час створення запису

Таблиця lessons_calendar_clients

Зв’язкова таблиця для бронювання клієнтами слотів у розкладі. Структура

таблиці «lessons_calendar_clients» наведена у таблиці 3.10.

Таблиця 3.10

Структура таблиці «lessons_calendar_clients»

Назва	Тип	Атрибути	Опис
id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, PRIMARY, NOT NULL	Первинний ключ
created_at	TIMESTAMP	NULL	Дата й час створення запису
updated_at	TIMESTAMP	NULL	Дата й час останнього оновлення
company_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, NOT NULL	Id компанії
lessons_calendar_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, NOT NULL	Id компанії
client_id	BIGINT	AUTO_INCREMENT, UNSIGNED, UNIQUE, NOT NULL	Доступна з
payment_method	TEXT	NULL	Доступна до
order_id	BIGINT	NULL	Хеш пароля
deleted_at	TIMESTAMP	NULL	Дата й час видалення

3.3. Реляційний зв’язок між таблицями

Вибудувавши структуру таблиць (підрозділ 3.2), слід тепер показати, як вони пов’язані між собою, щоб забезпечити цілісність даних та реалізувати всі описані в розділі 2 алгоритми. Нижче наведено опис ключових відносин «один-до-багатьох» та «багато-до-багатьох» і порядок, у якому таблиці взаємодіють під час типових сценаріїв.

Користувачі, клієнти та співробітники

users → clients (1:N)

Кожен запис у таблиці users може відповідати одному клієнту (строка в clients), але один клієнт належить лише одному користувачу. Це забезпечує зв’язок «обліковий запис → дані клієнта».

users → company_staffs (1:N)

Користувач із роллю тренера чи адміністратора має відповідний запис у company_staffs. Через цей зв’язок система відрізняє звичайного клієнта від співробітника клубу.

Компанії та їх підлеглі сутності

companies → company_locations (1:N)

Одна компанія може мати декілька філіалів: кожна локація зберігається в окремому записі company_locations, пов’язаному через company_id.

companies → company_staffs (1:N)

Всі співробітники, що працюють у межах конкретної компанії, зв’язуються зовнішнім ключем company_id у таблиці company_staffs.

companies → tariffs (1:N)

Кожна компанія самостійно визначає набір тарифів: «місячний абонемент», «разове відвідування», «пакет із п’яти занять» тощо.

companies → lessons_calendar (1:N)

Розклад занять формується для конкретної компанії – усі записи lessons_calendar мають поле company_id.

Абонементи та тарифи

clients → subscriptions (1:N)

Один клієнт може придбати послідовно кілька абонементів (наприклад, один місяць, потім ще один), отже в таблиці subscriptions для одного client_id може існувати декілька рядків.

tariffs → subscriptions (1:N)

Кожен тариф визначає умови одного виду абонемента. Запис subscription утримує зовнішній ключ tariff_id, щоб знати, на які умови він поширюється.

Бронювання занять і пов’язані замовлення

clients ↔ lessons_calendar (M:N)

Багато клієнтів можуть записатися на одне заняття, і кожен клієнт може відвідувати багато занять. Цей зв’язок реалізовано через проміжну таблицю lessons_calendar_clients, де кожен запис містить client_id і lesson_id.

clients → orders (1:N)

Кожне підтверджене бронювання оформляється також як замовлення в orders. Таблиця має поле client_id, що вказує на того, хто виконав замовлення.

lessons_calendar → orders (1:N)

Хоча бронювання клієнтів і зберігаються в lessons_calendar_clients, деталі оплати (сума, знижка, час оплати) логічніше відображати окремо в orders. Кожне замовлення прив’язане до одного заняття через lesson_id.

Приклад сценарію «бронювання → оплата → підтвердження»

Клієнт обирає слот у lessons_calendar.

Система створює запис у lessons_calendar_clients зі статусом booked.

Одночасно формується чернетка замовлення в orders зі статусом pending та полем amount.

Після успішної оплати через платіжний шлюз orders.status оновлюється на confirmed, а поле paid_at заповнюється часовою міткою.

Якщо оплата не проходить, orders.status змінюється на cancelled, а відповідний запис у lessons_calendar_clients позначається як cancelled або видаляється.

Усі зовнішні ключі налаштовані з опцією ON DELETE CASCADE для забезпечення коректного видалення залежних записів: наприклад, якщо видалити клієнта, автоматично прибереться його профіль у clients, записи в subscriptions, lessons_calendar_clients і orders.

Допоміжні таблиці Laravel

Окрім основних «предметних» таблиць, у проєкті також присутні стандартні Laravel-міграції:

migrations — історія виконаних міграцій.

failed_jobs — фіксація невдалих завдань черги.

password_resets — токени для відновлення пароля.

personal_access_tokens (якщо використовується Laravel Sanctum) — API-токени користувачів.

Ці допоміжні таблиці не беруть участі у предметній моделі, але забезпечують коректну роботу фонових задач, безпечну автентифікацію та можливість відкотів у разі помилок.

Реляційні відношення виглядають наступним чином (Рис. 3.1., Рис. 3.2., Рис. 3.3.).

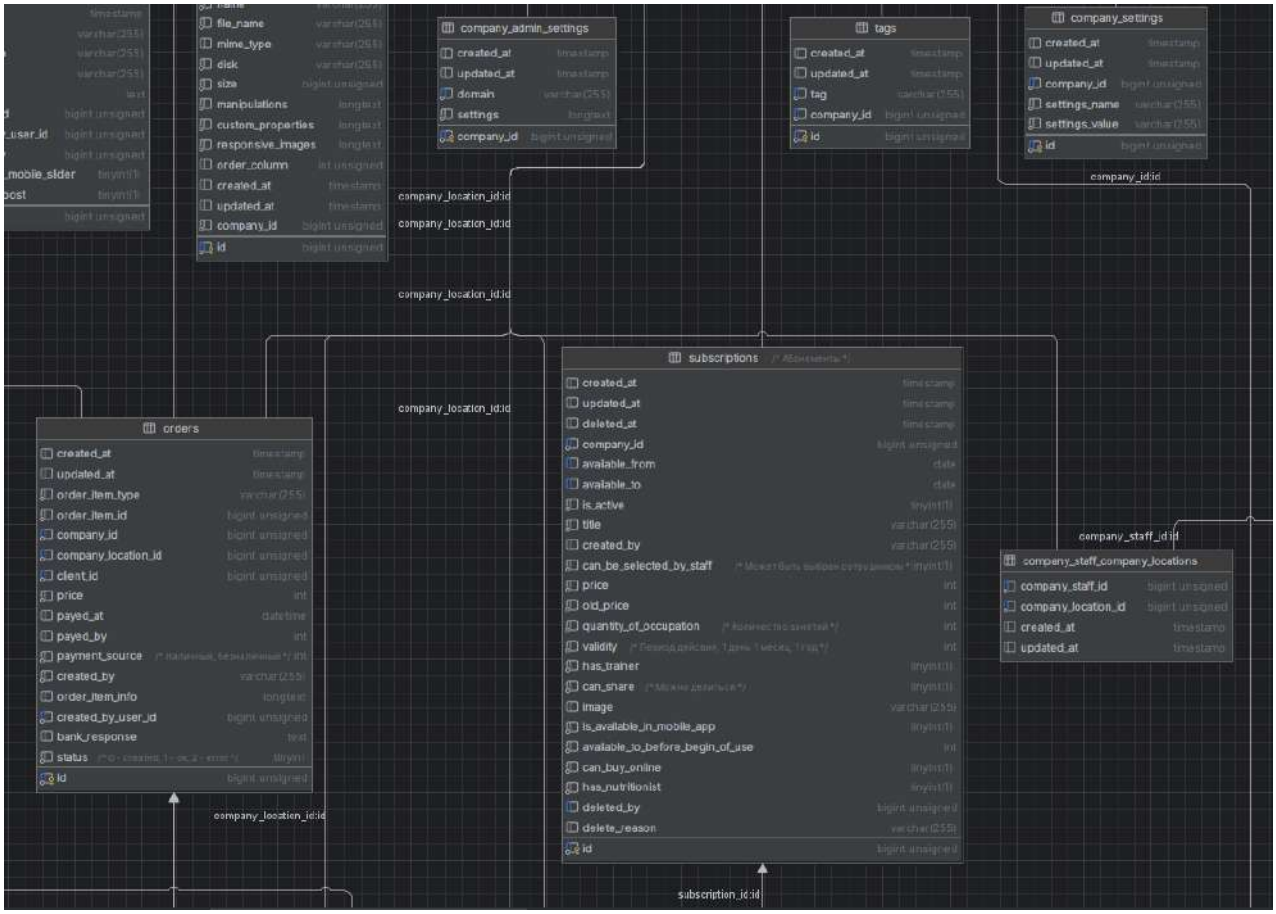

Рис. 3.1. Реляційні відношення 1

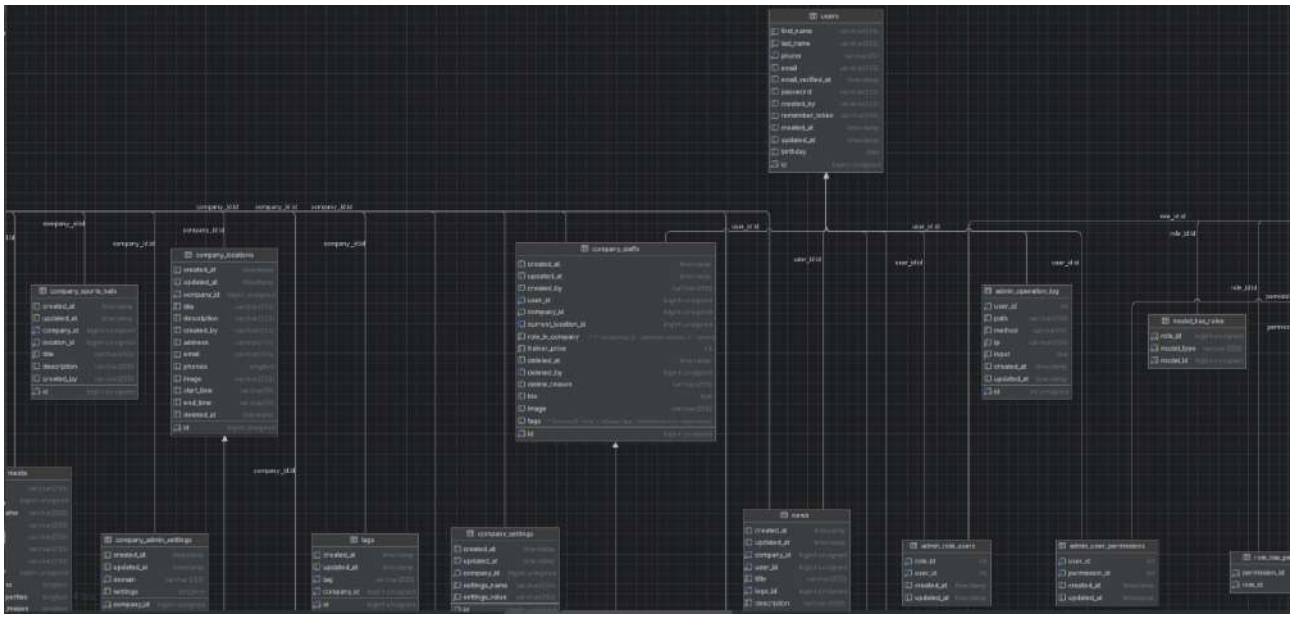

Рис. 3.2. Реляційні відношення 2

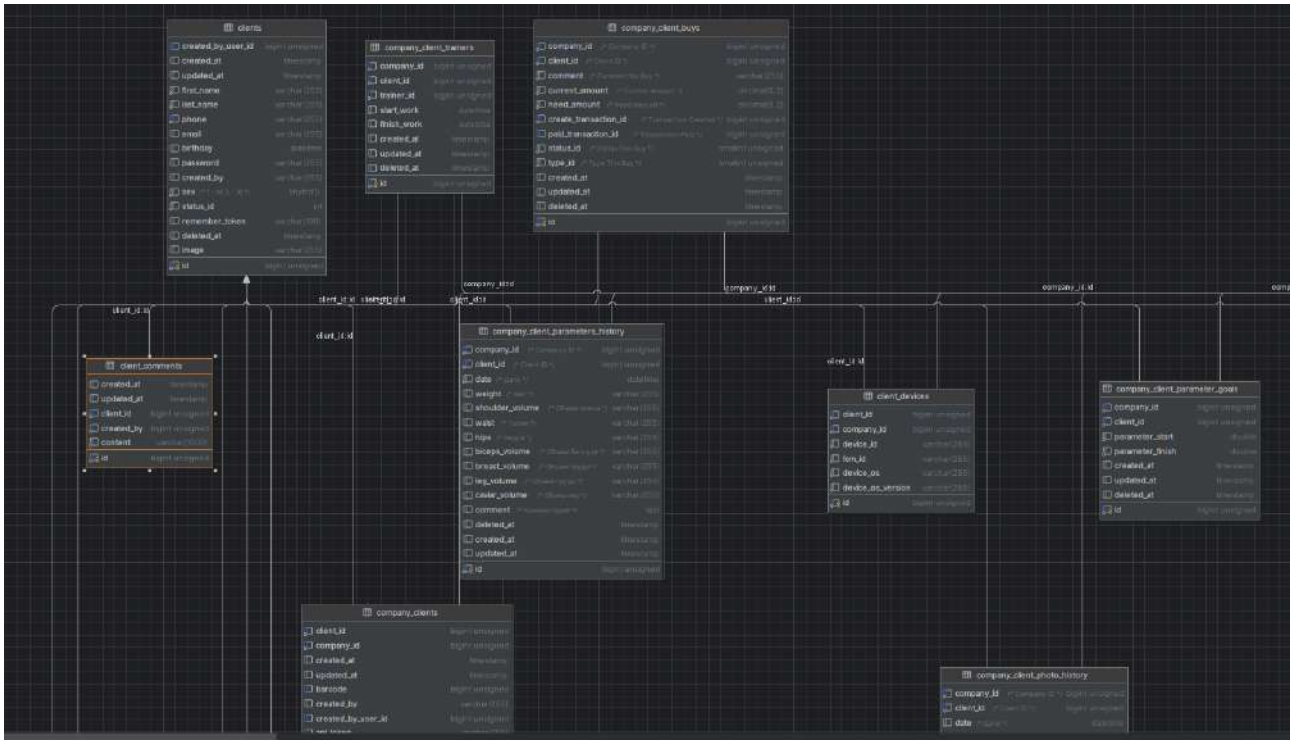

Рис. 3.3. Реляційні відношення 3

Висновки до розділу

Третій розділ присвячено проектуванню реляційної моделі в MySQL — вибору СУБД, опису структури таблиць і зв’язків між ними. Стандартизована

таблиця users слугує джерелом прав користувачів, companies, company_locations і company_staffs відображають структуру спортивного закладу, а clients, tariffs та subscriptions керують моделлю абонементів. Таблиці lessons_calendar, lessons_calendar_clients і orders побудовані для обробки бронювань і платежів із гарантією цілісності даних (ON DELETE CASCADE, транзакції InnoDB). Стандартні міграції Laravel і кешування Redis оптимізують розробку та продуктивність, а ретельне документування зв’язків забезпечує легкість підтримки.

РОЗДІЛ 4
РОЗРОБКА WEB-САЙТУ

4.1. Обгрунтування вибору технологій розробки

На етапі безпосередньої реалізації веб-сайту для управління замовленнями спортивного закладу постало завдання не просто реалізувати усі заплановані функції, а зробити це в такий спосіб, щоб код був зрозумілим, масштабованим і підтримуваним у довготривалій перспективі. Саме тому вибір технологій не був спонтанним — кожна складова технологічного стеку підбиралася під конкретні вимоги, які сформувалися на етапах аналізу та проєктування. Ключову роль у цьому рішенні відіграли такі фактори, як популярність фреймворку, наявність документації, зручність роботи для команди, гнучкість у розширенні функціональності та активність розробницької спільноти.

В якості основного серверного фреймворку було обрано Laravel. Це сучасний, елегантний PHP-фреймворк, який побудований на парадигмі MVC (Model-View-Controller) і надає чітке розділення логіки, шаблонів і даних. Laravel забезпечує чудову базу для побудови API, що було особливо важливо з огляду на те, що клієнтська частина сайту реалізована окремо на JavaScript-фреймворку Vue.js. Крім цього, Laravel містить потужну ORM-систему Eloquent, яка значно спрощує роботу з базою даних MySQL, дозволяючи формувати запити в зручній об'єктно-орієнтованій формі.

Одним із важливих інструментів, що значно прискорив розробку адміністративної частини, став Laravel-Admin — готова панель адміністратора, побудована на базі Laravel і адаптована для керування контентом, користувачами, замовленнями та іншими бізнес-об’єктами без потреби створювати бекенд «з нуля». Цей інструмент дав змогу розробити внутрішній інтерфейс для співробітників фітнес-центру за значно коротший термін і з меншими витратами. Наприклад, налаштування перегляду замовлень, оновлення статусу оплат або формування статистики по тренуваннях здійснюється без складного

програмування, оскільки Laravel-Admin підтримує готові компоненти для таблиць, фільтрів і графіків.

Для фронтенд-розробки обрано Vue.js — сучасний прогресивний JavaScript-фреймворк, що дозволяє створювати швидкі, динамічні та реактивні інтерфейси. На відміну від традиційного підходу з Blade-шаблонами Laravel, використання Vue.js дало змогу розділити фронтенд і бекенд, а також реалізувати SPA (Single Page Application), де кожна дія користувача (бронювання, перегляд розкладу, заповнення форми) виконується без перезавантаження сторінки. Такий підхід особливо важливий для збереження позитивного досвіду користувача в мобільній версії сайту, де кожен зайвий перехід між сторінками може викликати затримки або плутанину.

Для організації сучасного процесу збирання та обробки ресурсів (CSS, JS, зображень) у проєкті використовується Vite — інструмент нового покоління для фронтенд-збірки, який прийшов на зміну Webpack. Його перевага полягає в надзвичайно швидкому «гарячому перезавантаженні» (hot module replacement), завдяки якому розробник миттєво бачить зміни на екрані під час редагування коду. Це особливо зручно при тестуванні реактивних компонентів інтерфейсу, таких як календар бронювання або модальні вікна підтвердження.

Щоб забезпечити масштабованість і зручність розгортання сайту в різних середовищах, включно з локальним тестуванням, хмарними сервісами чи продакшн-сервером, у проєкті було використано Docker. Завдяки Docker створено ізольоване середовище для PHP, MySQL, Redis і фронтенд-збірки, що гарантує однакову поведінку коду на будь-якому комп’ютері незалежно від операційної системи. Також це дозволило зберігати конфігурації у вигляді коду (інфраструктура як код) — кожен член команди може за хвилину підняти весь проєкт, виконавши кілька команд у терміналі.

Окрему увагу було приділено моніторингу та діагностиці в процесі розробки. Для цього використано Laravel Telescope — офіційний пакет для відстеження запитів, помилок, запитів до бази даних, подій та іншої активності в реальному часі. За допомогою Telescope дослідник та інші учасники команди

могли виявляти, які запити сповільнюють систему, де виникають помилки у валідації або де варто додати кешування. Це значно підвищило якість коду ще до розгортання в робоче середовище.

Усі черги та фонові задачі, пов’язані з відправкою повідомлень, обробкою відкладених подій або повторною перевіркою транзакцій, було реалізовано за допомогою Redis. Він служить як високошвидкісний брокер черг (через Laravel Queue) і дозволяє обробляти завдання асинхронно, не блокуючи основний потік запитів. Це, зокрема, стосується відправки SMS-нагадувань, генерації PDF-файлів зі звітами або повторного запиту статусу оплати з боку платіжного шлюзу.

Усі зазначені технології були не лише логічно обґрунтовані, але й підтверджені успішним досвідом розробки у реальному середовищі. Їх комбінація дозволила створити сучасну, швидку, зручну та безпечну веб-платформу, яка легко масштабується, адаптується до потреб бізнесу та лишається дружньою як для адміністратора, так і для кінцевого користувача. У наступних підрозділах буде розглянуто, як саме реалізовано інтерфейс сайту та адміністративну панель, що базуються на цьому технологічному фундаменті.

Таким чином, побудова реляційних зв’язків дозволила гарантувати цілісність і консистентність даних, полегшила написання складних запитів (JOIN, агрегації) і надала чіткий каркас для подальшої розробки функціоналу звітності та аналітики. У наступному розділі ми перейдемо до безпосередньої реалізації інтерфейсу та адміністративної частини сайту.

4.2. Інтерфейс-користувача сайту

Інтерфейс користувача (UI) — це перше і найважливіше враження, яке отримує відвідувач веб-сайту спортивного клубу. У цьому проєкті ми керувалися принципом «мобільного-першого» дизайну, адже значна частина клієнтів обирають запис на тренування зі смартфонів. Водночас головним завданням було створити інтуїтивно зрозумілий та візуально привабливий інтерфейс, який би з однаковим комфортом працював як на екранах мобільних пристроїв, так і на

десктопах.

Почну з сайдбару, у нашому веб-додатку — це не просто меню, а справжній помічник для адміністраторів і співробітників спортивного закладу. Завдяки продуманому дизайну з темним фоном, контрастним текстом і яскравою зеленою лінією він не лише привертає увагу, але й забезпечує швидкий доступ до всіх ключових функцій. Іконки поруч із пунктами роблять навігацію інтуїтивною, а розгортаєма структура дозволяє легко знайти потрібний розділ. Вигляд сайдбару наведений нижче (Рис. 4.1.).

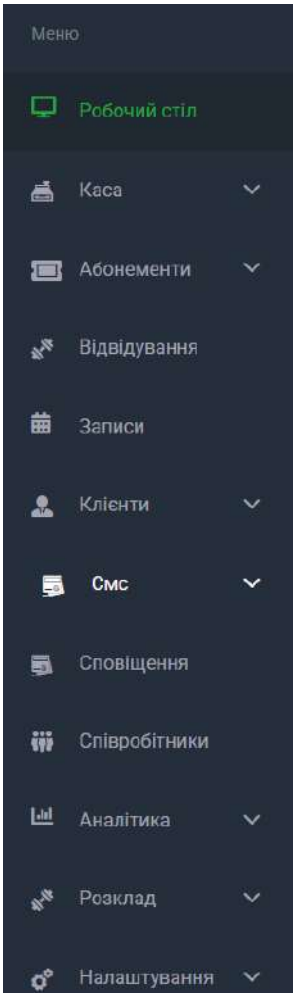

Рис. 4.1. Сайдбар

- Робочий стіл — центр управління. Уявіть собі командний пункт, звідки видно все, що відбувається у вашому спортивному закладі! "Робочий стіл" — це перше, що ви бачите, заходячи в систему. Він створений, щоб дати вам

миттєвий огляд стану справ без зайвих кліків. Простота і швидкість — ось що робить "Робочий стіл" незамінним. Жодних підпунктів, лише найважливіше на одному екрані. Ви можете одразу оцінити ситуацію і вирішити, куди рухатися далі — чи то до каси, чи до розкладу. Вигляд сторінки робочого чтолу наведена нижче (Рис. 4.2.)

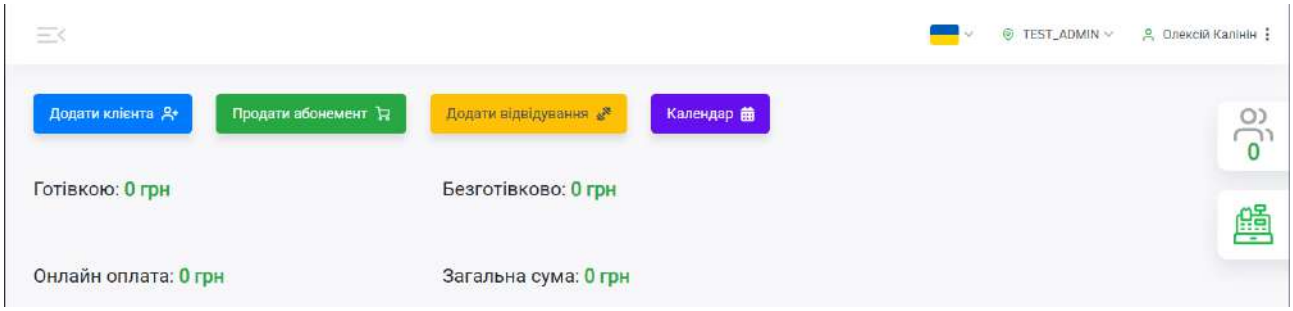

Рис. 4.2. Сторінка «Робочий стіл»

- Каса — Фінансовий мозок закладу. Ласкаво просимо до серця фінансових операцій! Розділ "Каса" — це місце, де ви керуєте всіма грошовими потоками: від готівки до онлайн-платежів. Завдяки розгортаємій структурі з чотирма підпунктами, ви завжди знайдете потрібний інструмент для роботи з фінансами. Повний контроль над транзакціями в реальному часі. Зручне управління чеками та безготівковими платежами. Можливість обробляти онлайн-замовлення без зайвих зусиль. Готівкові операції — Ваш щоденний фінансовий пульс. Це місце, де фіксується кожна копійка, що проходить через ваш заклад. Продаж абонементів, оплата разових занять чи покупка спортивного харчування — усе тут. Реєстрація нових операцій у кілька кліків. Історія транзакцій за день, тиждень чи місяць. Звіти для бухгалтерії — просто і швидко. Безготівкові операції — Сучасність у дії. Світ рухається до безготівкових платежів, і ми разом із ним. Цей підпункт — ваш провідник у світ карток, електронних гаманців і банківських переказів. Приймання платежів через термінал чи онлайн. Відстеження статусу кожної транзакції. Генерація чеків для клієнтів одним натисканням. Чекбокс — Гаманець вашого клієнта. Уявіть внутрішній рахунок клієнта, який працює як бонусна система чи передплачений гаманець. Цей

підпункт — ключ до таких операцій. Перегляд і поповнення балансу клієнтів. Списання коштів за послуги чи товари. Видача чеків для прозорості. Онлайн заняття — Ваш зв’язок із цифровим світом. Цей підпункт створений, щоб обробляти заявки швидко і без стресу. Перегляд нових заявок у реальному часі. Підтвердження платежів і статусів. Інтеграція з онлайн-платежами. Підрозділи каси та їх сторінки наведені нижче (Рис. 4.3., Рис. 4.4., Рис. 4.5.)

-

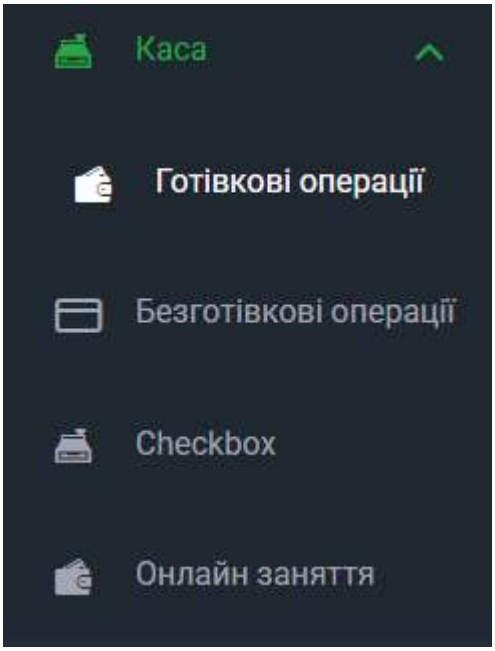

Рис. 4.3. Каса

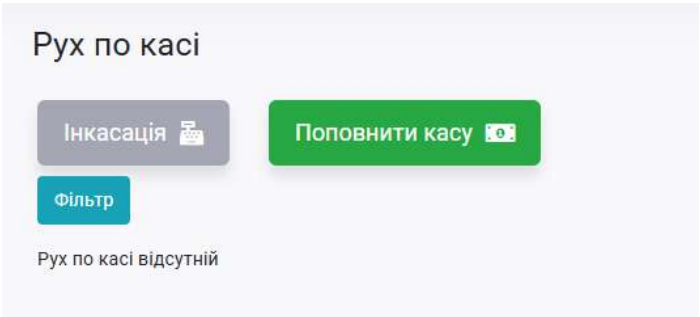

Рис. 4.4. Сторінка «Готівкові операції»

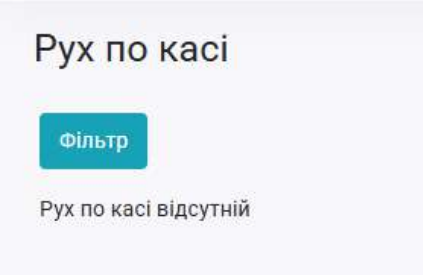

Рис. 4.5. Сторінка «Безготівкові операції»

- Абонементи — Серце клієнтських послуг. Цей розділ — справжня знахідка для тих, хто хоче організувати роботу з клієнтськими абонементами. Він відкриває доступ до всіх аспектів, пов’язаних із продажем, відстеженням і контролем терміну дії. Зелений фон підкреслює, що це активна зона, а розгорнута структура з трьома підпунктами дозволяє охопити весь процес. "Продажі абонементів" дають змогу легко оформляти нові угоди, фіксувати оплати та видавати цифрові квитанції. "Закинутість" допомагає виявляти клієнтів, які рідко відвідують, пропонуючи шанс повернути їх мотиваційними програмами. "Закінчились" же миттєво показує, чиї абонементи потребують оновлення, зберігаючи ваш дохід і лояльність клієнтів. Ця функція — ідеальний баланс між контролем і турботою про відвідувачів. Нижче наведений вигляд підрозділів «Абонементи» та їх сторінок (Рис. 4.6., Рис. 4.7., Рис. 4.8., Рис. 4.9., Рис. 4.10.)

-

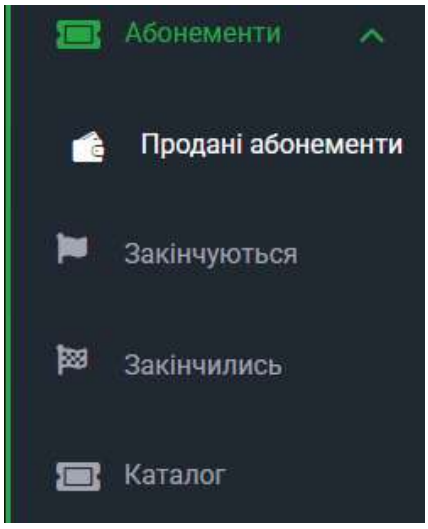

Рис. 4.6. Абонементи

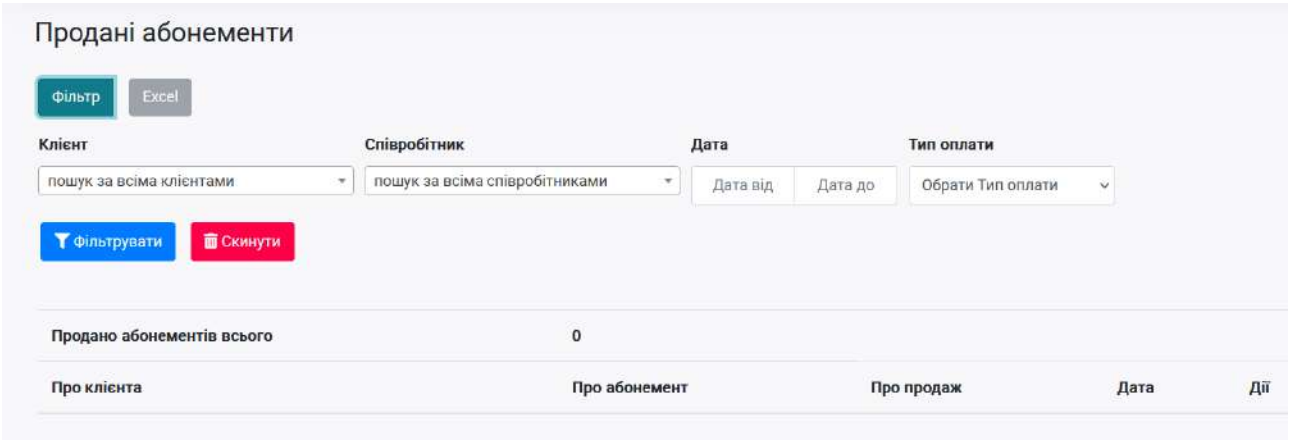

Рис. 4.7. Сторінка «Продані абонементи»

Рис. 4.8. Сторінка «Закінчуються»

Рис. 4.9. Сторінка «Закінчились»

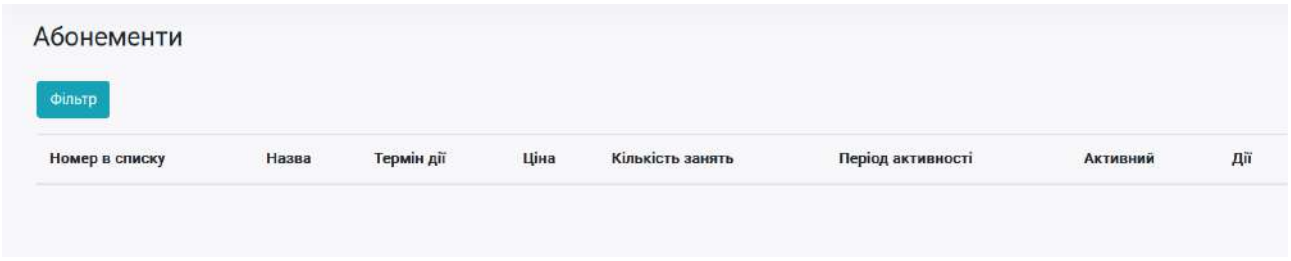

Рис. 4.10. Сторінка «Каталог»

- Відвідування — Пульс вашого закладу. Цей розділ — ваш око, що стежить за активністю клієнтів. З іконкою у вигляді гантелі він одразу асоціюється зі спортивним духом. Тут зібрано всю інформацію про те, хто і коли приходив на заняття, що дозволяє аналізувати завантаженість і планувати ресурси. Адміністратори можуть переглядати статистику за день, тиждень чи місяць, швидко виявляючи піки відвідуваності. Це також зручний інструмент для звітів, які можна використати для переговорів із тренерами чи інвесторами. Простота дизайну робить його доступним навіть для новачків, а чіткий текст забезпечує легке сприйняття. Приклад сторінки «Відвідування» наведений нижче (Рис. 4.11.)

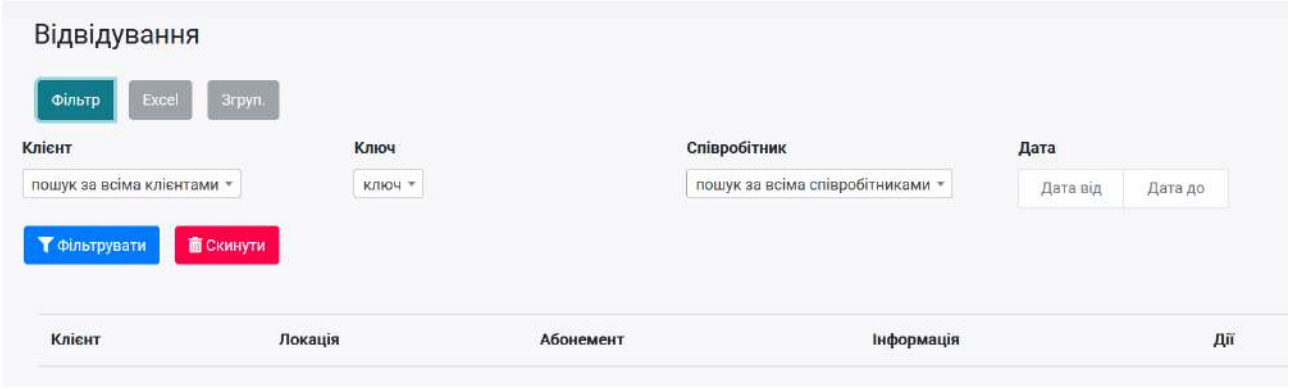

Рис. 4.11. Сторінка «Відвідування»

- Записи — Організація вашого часу. Розділ "Записи" — це ваш особистий календар для управління розкладом. Іконка у вигляді календаря натякає на його призначення: допомагати планувати заняття, тренування чи індивідуальні консультації. Тут адміністратори можуть створювати нові записи, редагувати існуючі або скасовувати їх за потребою. Клієнти, у свою чергу, можуть бачити доступні слоти і бронювати час онлайн, що економить час усім. Інтерфейс дозволяє сортувати записи за датою чи статусом, а також додавати нотатки, наприклад, про особливі побажання відвідувачів. Це місце, де порядок і зручність стають основою ефективної роботи закладу. Приклад сторінки

«Записи» наведений нижче (Рис. 4.12.)

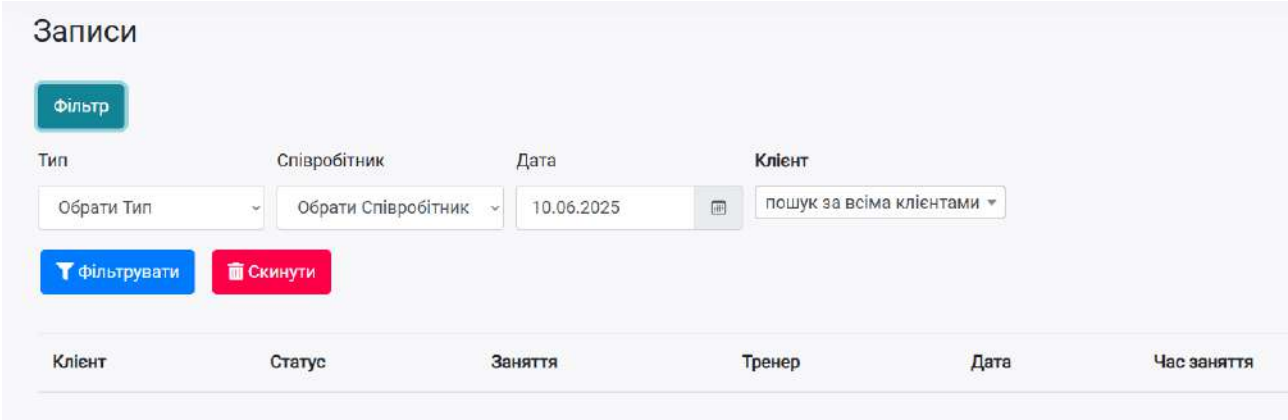

Рис. 4.12. сторінка «Записи»

- Клієнти — Серце вашої аудиторії. Цей розділ — справжній хаб для роботи з тими, хто робить ваш заклад живим. Іконка силуету людини одразу вказує на його призначення — управління даними клієнтів. Розгорнута структура з трьома підпунктами дозволяє охопити всі аспекти взаємодії. "Список" пропонує зручний перегляд усіх зареєстрованих відвідувачів із фільтрами за ім’ям, абонементом чи датою останнього відвідування. "Профіль" відкриває доступ до детальної інформації: контактні дані, історія покупок і персональні примітки, що допомагає персоналізувати підхід. "СМС" інтегрує функцію зв’язку, дозволяючи швидко надіслати повідомлення прямо з цього розділу. Це місце, де кожен клієнт стає не просто номером, а частиною вашої спільноти. Нижче наведений приклад клієнтів (Рис. 4.13., Рис. 4.14.)

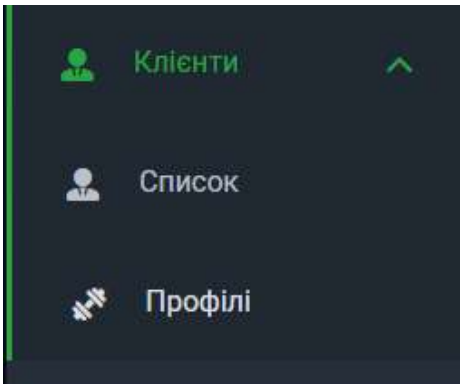

Рис. 4.13. Клієнти

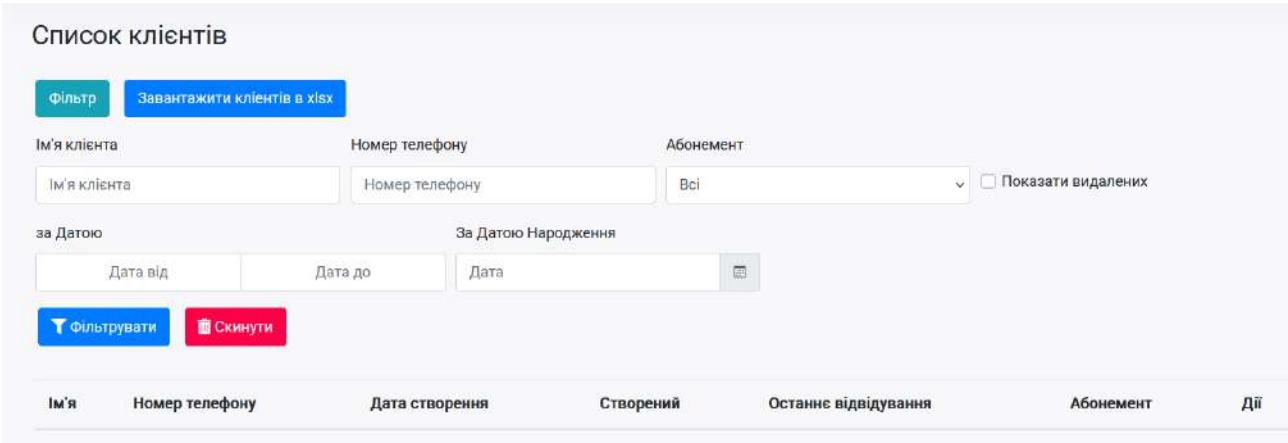

Рис. 4.14. сторінка «Список»

- СМС — Ваш голос до клієнтів. Розділ "СМС" — це ваш прямий канал комунікації з відвідувачами. Іконка конверта символізує швидкість і надійність повідомлень. Тут ви можете створювати масові розсилки про акції, нагадування про оплату чи запрошення на нові тренування. Інтерфейс простий: введіть текст, виберіть отримувачів із бази клієнтів і натисніть "Надіслати". Система автоматично фіксує час відправлення та статус доставки, що зручно для звітності. Це інструмент, який зміцнює зв'язок із аудиторією, підвищуючи її лояльність і залученість. Нижче наведена сторінка відправлених смс (Рис. 4.15.)

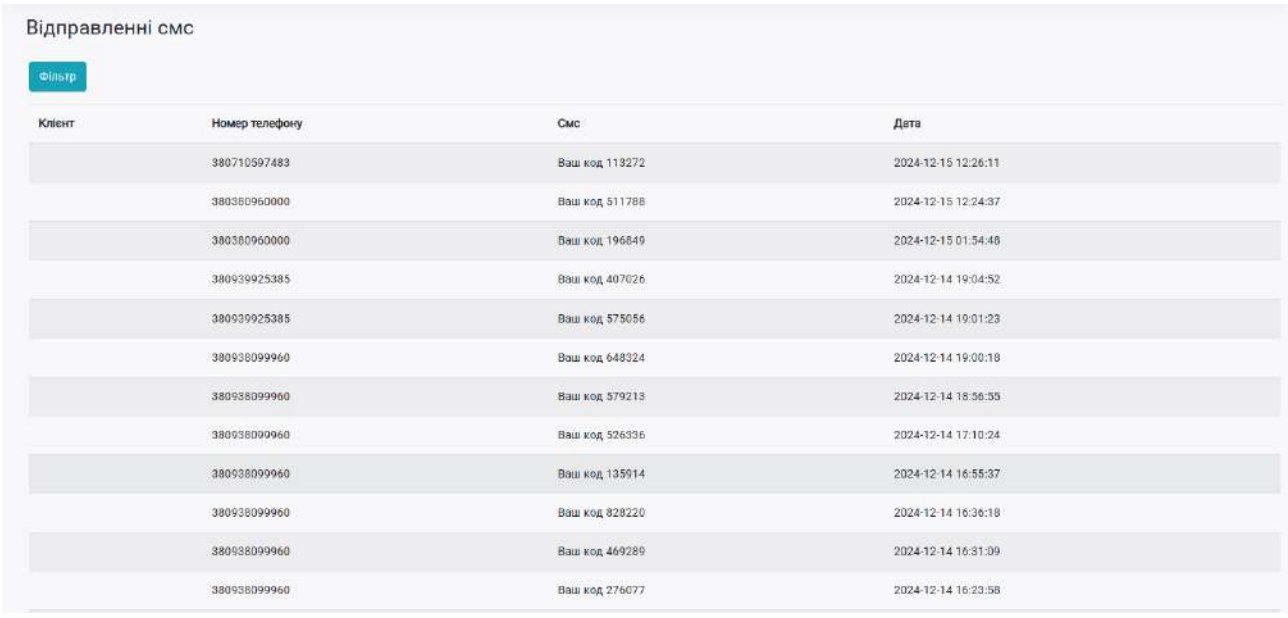

Рис. 4.15. Сторінка «Відправлені смс»

- Сповіщення — Завжди в курсі подій. Цей розділ — ваш особистий асистент, який тримає руку на пульсі. Іконка дзвіночка натякає на його роль — повідомляти про важливі оновлення. Тут відображаються нагадування про закінчення абонементів, нові заявки від клієнтів чи запити від співробітників. Кожен сповіщення супроводжується коротким описом і кнопкою для швидкої реакції, наприклад, перегляду деталів чи переходу до відповідного розділу. Дизайн дозволяє сортувати повідомлення за пріоритетом чи часом, щоб ви завжди могли зосередитися на найurgentніших завданнях. Це ваш спосіб не пропустити жодної деталі в напруженому робочому графіку. Приклад сторінки сповіщень наведений нижче (Рис. 4.16.)

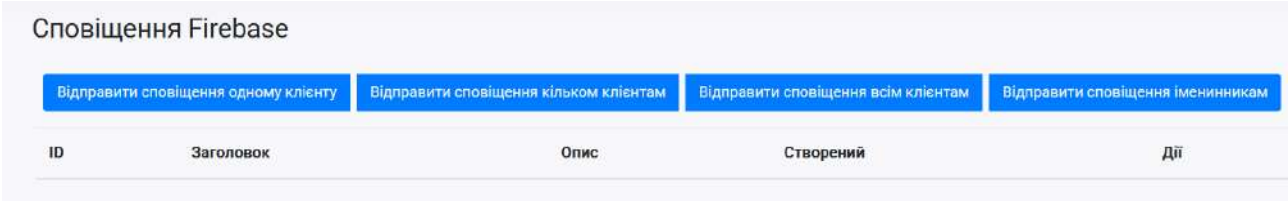

Рис. 4.16. сторінка «Сповіщення»

- Співробітники — Сила вашої команди. Цей розділ — центр управління вашою командою, де зібрані всі дані про тренерів, адміністраторів та інший персонал. Іконка групи людей одразу вказує на його призначення — об’єднувати та координувати зусилля співробітників. Тут ви можете переглядати список усіх працівників із деталями: посада, графік роботи та контактна інформація. Додавайте нових членів команди, редагуйте їхні профілі чи відстежуйте відпустки в одному місці. Інтерфейс простий і зручний, дозволяючи швидко знайти потрібного фахівця та призначити завдання. Це місце, де гармонія в команді стає запорукою успіху закладу. Приклад сторінки співробітників наведений нижче (Рис. 4.17.)

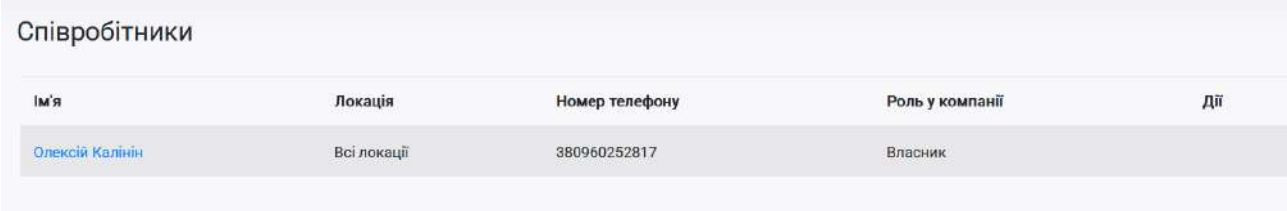

Рис. 4.17. сторінка «Співробітники»

- Аналітика — Ваш погляд у майбутнє. Розділ "Аналітика" — це ваш аналітичний центр, де дані перетворюються на цінні інсайти. Іконка стовпчастої діаграми підказує, що тут ви знайдете візуалізацію результатів роботи закладу. Розгорнута структура з двома підпунктами дозволяє глибоко зануритися в цифри. "Продукт" пропонує детальний аналіз продажів абонементів, товарів чи послуг, показуючи, які пропозиції приносять найбільший дохід, а які потребують доопрацювання. "Відвідування" відображає статистику активності клієнтів за різними періодами, допомагаючи оптимізувати розклад і ресурси. Інтерфейс підтримує фільтри та графіки, щоб ви могли легко інтерпретувати дані та планувати подальший розвиток. Це ваш компас для стратегічних рішень. Заображення аналітики продажів наведено нижче (Рис. 4.18.)

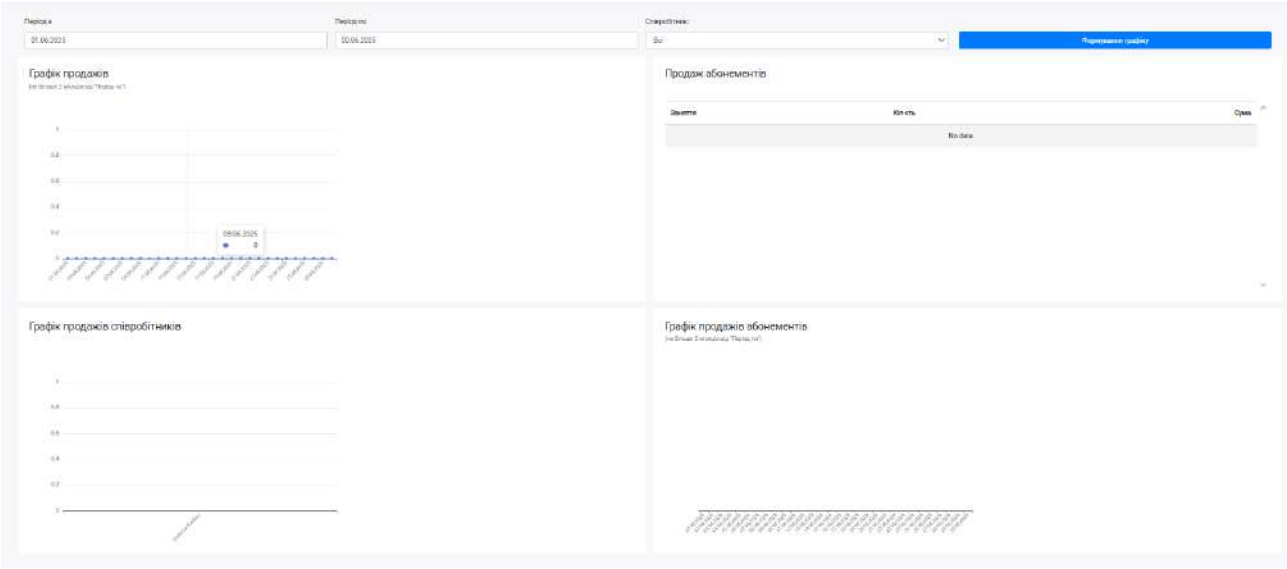

Рис. 4.18. Сторінка «Продажі»

4.3. Адміністративна частина сайту

Адміністраторська панель стала серцем системи управління — місцем, де менеджери спортклубу й тренери тримають руку на пульсі діяльності, налаштовують розклад, відстежують бронювання й платежі, а також оперативно реагують на побажання клієнтів. Для її реалізації в проєкті використано пакет Laravel-Admin, що надав готову основу для створення CRUD-інтерфейсів, фільтрів і графіків без необхідності писати великі обсяги шаблонного коду. Вигляд головної сторінки адмінки наведений нижче (Рис. 4.19.)

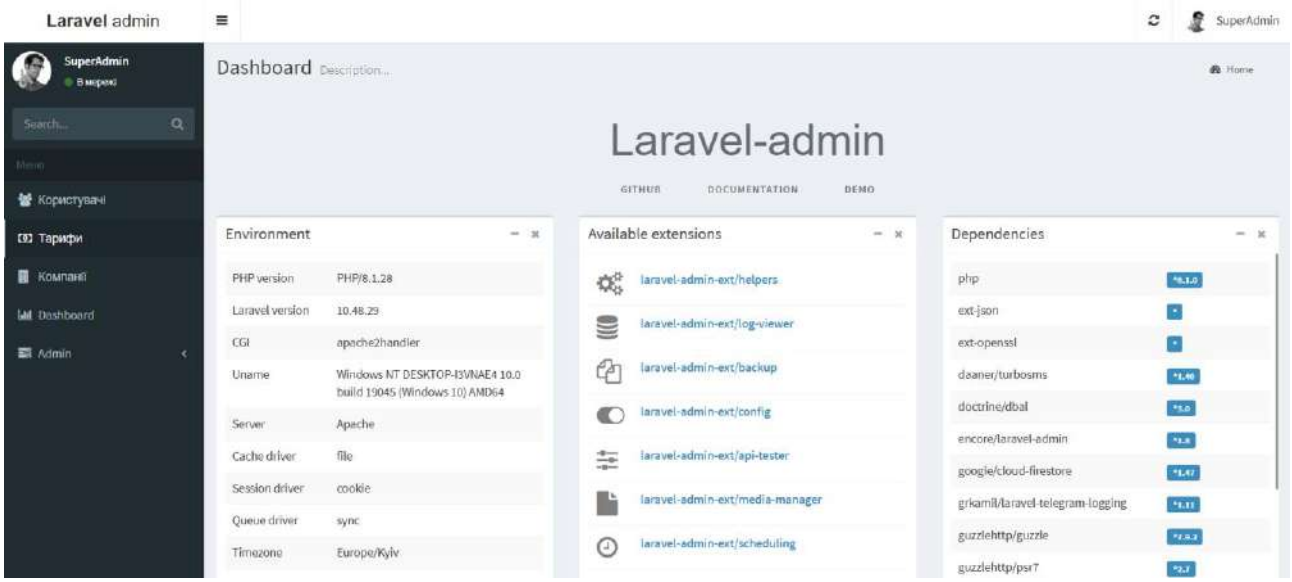

Рис. 4.19. Головна адмін-панелі

Відразу після входу адміністратор потрапляє на дашборд, де у верхній частині сторінки виводяться ключові показники: кількість активних клієнтів, число бронювань за добу, обсяг оплачених замовлень та середній відсоток заповненості групових занять. За реалізацію цих блоків відповідають спеціальні віджети Laravel-Admin, які збирають дані через агреговані запити до бази даних і відображають їх у вигляді інфографіки — невеликих кругових діаграм і лінійних графіків. Це дозволяє керівникові клубу в одну мить оцінити, чи зростає відвідуваність, чи необхідно додати додаткові слоти в пікові години, і чи справді

рекламна кампанія принесла нових клієнтів. Вигляд деяких сторінок адмін панелі будуть наведені нижче (Рис. 4.20., Рис. 4.21., Рис. 4.22.)

Рис. 4.20. Сторінка управління тарифами

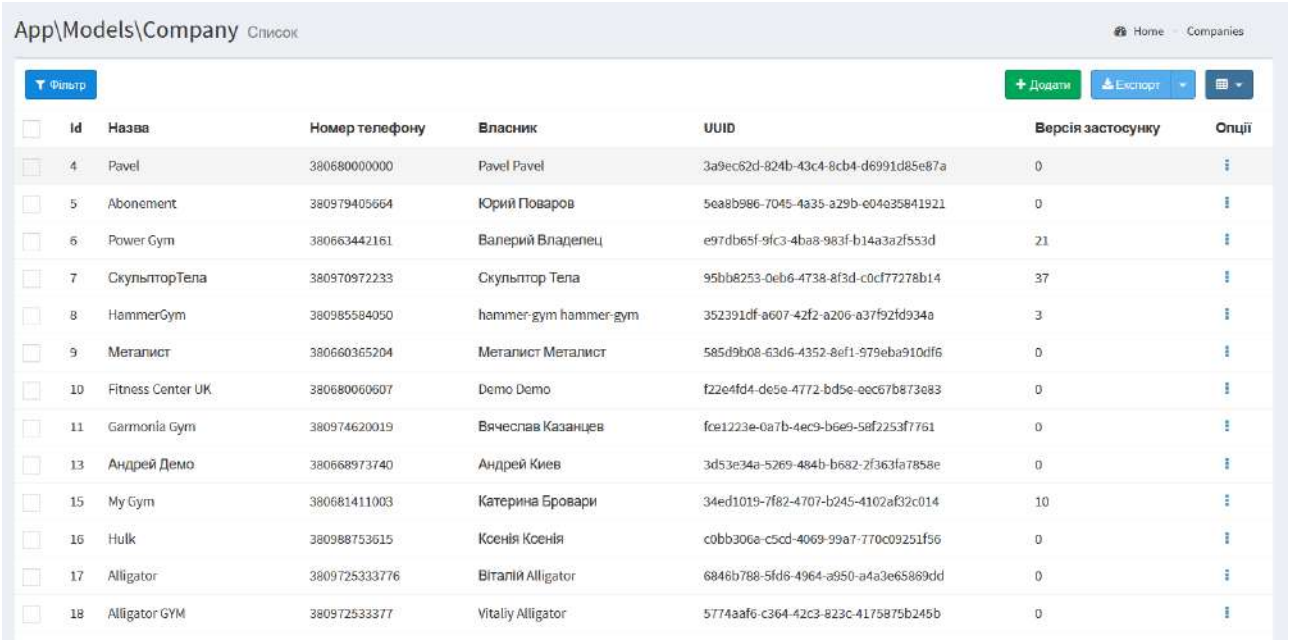

Рис. 4.21. Сторінка управляння компаніями

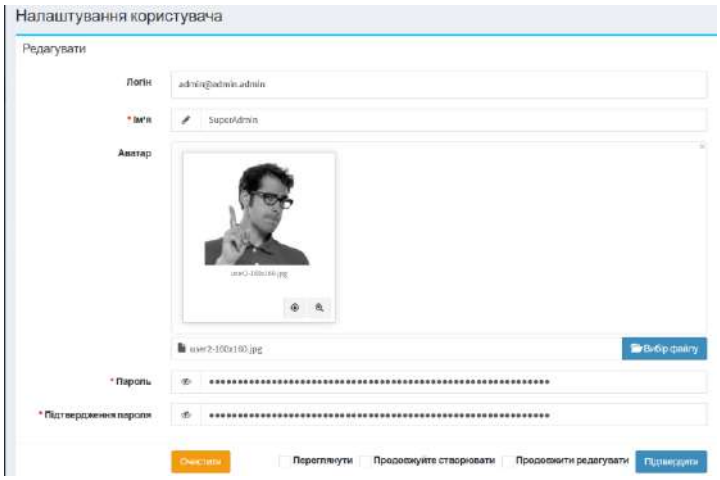

Рис. 4.22. Сторінка редугавуння користувача-адміністратора

Перехід до розділу «Користувачі» відкриває таблицю з користувачами клубу, у якій кожен рядок містить ім’я, контактні дані, статус абонемента та дату останнього візиту. Сторінка реалізована за допомогою компоненту Grid: тут можна шукати клієнтів за іменем чи email, фільтрувати за активним/простроченим абонементом, а також миттєво відправляти SMS-нагадування чи ручні повідомлення через вбудовану форму. Натискання на профіль клієнта відкриває деталі: історію бронювань, платежів, активні купони та накопичені бонусні бали. Саме тут менеджер може дозабронювати заняття від імені користувача або скасувати бронювання у разі зміни планів. Вигляд сторінки користувачів наведений нижче (Рис. 4.23.)

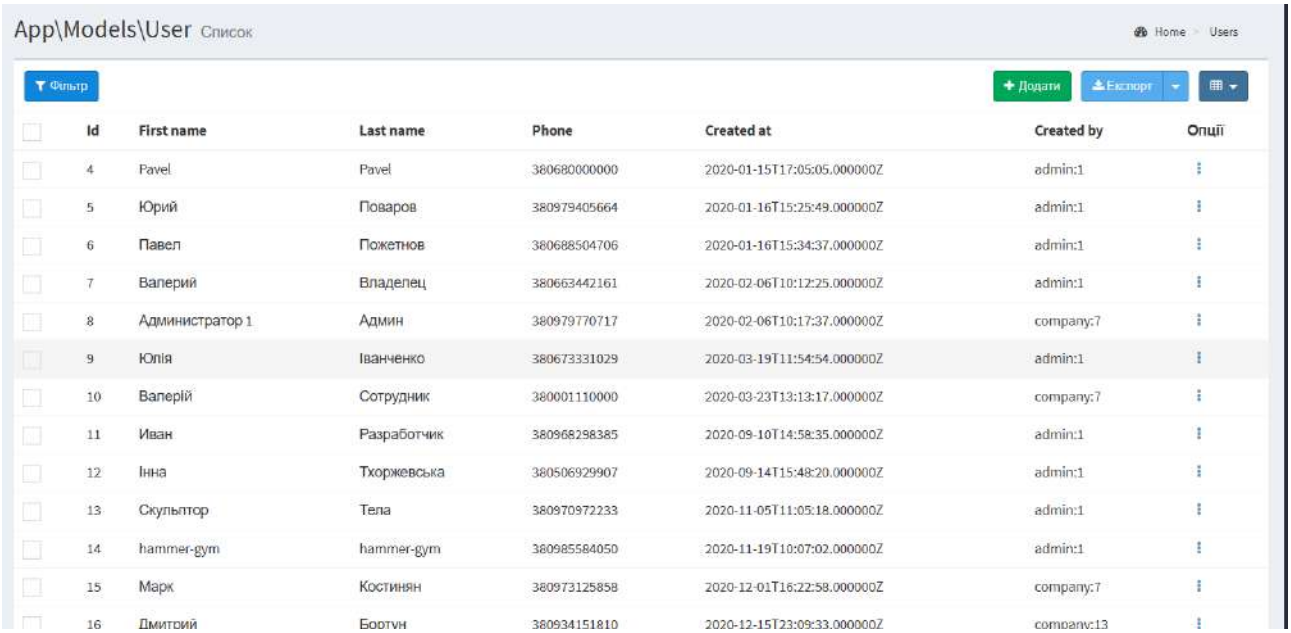

Рис. 4.23. Сторінка користувачів

Коли настає час обробки платежів, адмін-інтерфейс пропонує розділ «Замовлення», де всі транзакції групуються за статусами. Полотно таблиці відображає номер замовлення, клієнта, суму, застосовані знижки та часові мітки оплати. За потреби адміністратор може вручну підтвердити оплату, помітити помилкові транзакції як «відхилені» або запустити процес повернення коштів. У верхній частині сторінки розташований фільтр за датою та платіжним шлюзом, що дає змогу швидко знайти замовлення, що оброблялися через LiqPay чи Fondy.

Ще одна важлива частина адмін-панелі — «Звіти», де зібрані дашборди з аналітичною інформацією. Тут можна переглянути динаміку продажів за будь-який період, вирівнювання завантаженості по днях тижня, ефективність кожного тренера в балах відвідуваності та середню вартість одного відвідування. Для зручності доступні експорт у PDF чи Excel, а також можливість розсилки готових звітів на поштові скриньки керівництва за розкладом.

Нарешті, панель «Налаштування» дає змогу керувати довідниками: додавати нові платіжні шлюзи, оновлювати шаблони SMS і email-повідомлень, налаштовувати права доступу для різних ролей (наприклад, «менеджер», «тренер», «оператор»). За допомогою вбудованого редактора текстових блоків адміністратор може змінити вітальні банери на головній сторінці або оновити інформацію в футері без залучення розробників.

Також адмінка дозволяє управляти доступами користувачів, редагувати компанії та логувати дії юзера. Сторінка доступів наведена нижче (Рис. 4.24.)

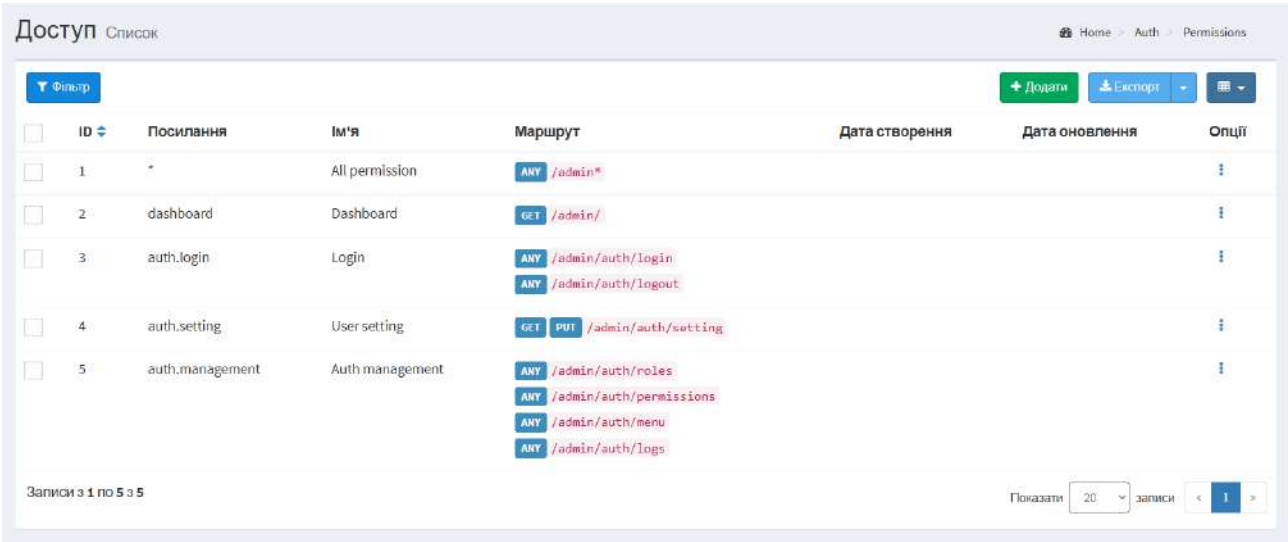

Рис. 4.24. Сторінка доступів

Таким чином, адміністративна частина сайту стала зручним інструментом для комплексного управління всіма аспектами діяльності фітнес-клубу: від роботи з клієнтами та бронюванням до фінансової аналітики і технічного супроводу. Використання Laravel-Admin значно пришвидшило розробку й

дозволило створити гнучкий і потужний бек-офіс, який легко адаптується до зростання бізнесу та змін у його процесах.

Висновки по розділу

У четвертому розділі обґрунтовано вибір стеку технологій (Laravel, Laravel-Admin, Telescope, Docker, Vite, Vue.js, Redis) та показано реалізацію інтерфейсу для користувачів і адміністраторів. Використання готових рішень (Laravel-Admin для бек-офісу, Vue.js + Vite для SPA) значно скоротило терміни розробки та підвищило якість UX/UI. Docker забезпечив репродуктивність середовища, а Telescope і Redis-черги гарантують прозорість і надійність фонових процесів. Втілення адаптивного, інтуїтивного дизайну, реактивних форм бронювання і модулів особистих кабінетів забезпечило високу зручність для клієнтів і ефективність роботи персоналу. Загалом, обрані технології дозволили створити сучасну, масштабовану та безпечну платформу, готову до подальшого розвитку та впровадження.

ВИСНОВКИ

У ході виконання кваліфікаційної бакалаврської роботи було послідовно вирішено низку завдань, спрямованих на створення сучасного веб-сайту для управління замовленнями клієнтів спортивних закладів. Перший розділ дав змогу чітко окреслити предмет і мету дослідження: визначити потреби як клієнтів, так і адміністраторів фітнес-клубів, дослідити існуючі зарубіжні та вітчизняні рішення, а також узагальнити їхні сильні та слабкі сторони. Аналіз конкурентів підтвердив відсутність готового продукту, який одночасно поєднував би багатофункціональність, зручний інтерфейс і відповідність українському законодавству та платіжним системам.

Другий розділ присвячений проєктуванню: розроблено детальні алгоритми основних сценаріїв — реєстрація, вибір послуги, обробка замовлення й оплата, сповіщення користувачів і адміністраторів. Вибір трирівневої архітектури (Vue.js → Laravel → MySQL/Redis) забезпечив чітке розмежування відповідальностей, гнучкість і масштабованість системи, що гарантує можливість подальшої інтеграції мобільних застосунків та нових сервісів.

У третьому розділі детально описано модель даних: обґрунтовано вибір СУБД MySQL як надійного, продуктивного та добре інтегрованого з PHP-стеком рішення; приведено структуру ключових таблиць (users, companies, clients, tariffs, subscriptions, lessons_calendar, orders тощо) та схему їхніх реляційних зв’язків. Така організація даних гарантує цілісність і консистентність інформації під час одночасної обробки великої кількості транзакцій і бронювань.

Четвертий розділ показав практичну реалізацію обраних рішень: стек Laravel, Laravel-Admin, Vue.js, Vite, Docker, Redis і Telescope дозволив побудувати комфортабельний інтерфейс для клієнтів і потужний бек-офіс для адміністраторів. Особливу увагу приділено адаптивному дизайну «mobile-first», реактивності UI, безпеці даних і моніторингу системи.

Загалом, реалізована платформа відповідає поставленим завданням: забезпечує швидку реєстрацію та бронювання тренувань, інтуїтивний особистий

кабінет із історією оплат, гнучке адміністрування розкладу й тарифів, надійну обробку платежів з інтеграцією українських шлюзів та відповідність вимогам законодавства про захист персональних даних. Архітектура й технології, вибрані на початкових етапах, дозволяють безболісно масштабувати систему, додаючи нові модулі та інтеграції. Отже, мету дослідження — розробка ефективного програмного забезпечення для управління замовленнями клієнтів спортивних закладів — досягнуто, а створена платформа має всі передумови для успішного впровадження в реальних умовах та подальшого розвитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Елізабет Робсон, Ерік Фрімен, Head First. HTML і CSS, - Фабула, 2025 – 620 с.

2. Гарвард Бізнес Ревью, Марко Янсіті, Сатья Наделла, Томас Х. Девенпорт, Цедал Нілі, HBR's 10 Must Reads on Data Strategy, - Harvard Business Review, 2025 – 150 с.

3. Ерік Фрімен, Елізабет Робсон, Head First. Програмування на JavaScript, - Фабула, 2022 – 532 с.

4. Роман Мельник, Програмування веб-застосувань (фронт-енд та бек-енд), - Львівська політехніка, 2018 – 134 с.

5. Ларрі Рокофф, The Language of SQL, - Longman (Pearson Education), 2021 – 165 с.

6. Люк Хеджер, Serverless Development on AWS: Building Enterprise-Scale Serverless Solutions, - O'Reilly Media, 2024 – 211 с.

7. Джеремі Тінлі, High Performance MySQL: Proven Strategies for Running MySQL at Scale, - O'Reilly Media, 2021 – 237 с.

8. Девід Вітні, Програмування для дітей. HTML, CSS та JavaScript, - Vivat, 2019 – 121 с.

9. Кріс Міннік, JavaScript All-in-One For Dummies, - John Wiley and Sons Ltd, 2023 – 701 с.

10. Кей С. Хорстманн, Modern JavaScript for the Impatient, - Longman (Pearson Education), 2020 – 126 с.

11. Лаура Томсон, Люк Вєллінг, PHP and MySQL Web Development, - Longman (Pearson Education), 2016 – 142 с.

12. Метт Стауффер, Laravel: Up & Running: A Framework for Building Modern PHP Apps, O'Reilly Media, 2023 – 426 с.

13. Майк МакГрат, CSS in easy steps, - In Easy Steps, 2020 – 53 с.

14. Олексій Васильєв, Алгоритми, - Ліра-К, 2022 – 224 с.

15. Геннадій Галісєєв, Системне програмування, - Університет

"Україна", 2019 – 44 с.

16. Володимир Дронов, HTML та CSS. 25 уроків для початківців, - PRINT2PRINT, 2020 – 246 с.

17. Лоренс Ларс Свекіс, Майке ван Путтен, Роб Персіваль, JavaScript із нуля до профі, - Expert insights, 2022 – 212 с.

18. Кайл Сімпсон, Познайомтеся, JavaScript, - PRINT2PRINT, 2022 – 133 с.

19. Devlin Basilan Duldulao, ASP.NET Core and Vue.js, - Packt, 2021 – 230 с.

20. Майк МакГрат, SQL in easy steps, - In Easy Steps, 2020 – 101 с.

21. Норман Ремсі, Programming Languages: Build, Prove, and Compare, - Cambridge University Press, 2022 – 469 с.

22. Ін Бай, SQL Server Database Programming with C#: Desktop and Web Applications, - Taylor & Francis, 2025 – 503 с.

23. Сайт: https://ru.scribd.com/document/722852208/Методичка-Redis-Дешко

24. Сайт: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-redis-i-zachem-on-nuzhen/

25. Майк МакГрат, HTML5 in easy steps, - In Easy Steps, 2017 – 168 с.

26. Сайт: https://dou.ua/lenta/articles/solid-principles/

27. Саверіо Перуджіні, Programming Languages: Concepts and Implementation, - Jones and Bartlett Publishers, 2021 – 697 с.

28. Райан Стівенс, SQL in 24 Hours, Sams Teach Yourself, - Longman (Pearson Education), 2022 – 110 с.

29. Сайт:https://journal.gen.tech/post/principi-solid-sho-ce-ta-yak-yih-zastosovuvati-kejsi-ta-porady

30. Сайт: https://dou.ua/forums/topic/39210/

ДОДАТКИ

Додаток А

Клас для отримання підписок

```
<?php

namespace App\Http\Controllers\Actions\Subscriptions;

use App\Components\CrmHelper;
use App\Constants\CompanySettings;
use App\Constants\CompanyStaffRoles;
use App\Constants\PaymentSources;
use App\Http\Controllers\Actions\BaseAction;
use App\Models\ClientSubscription;
use App\Models\CompanyStaff;
use App\Repositories\CompanyClientRepository;
use Auth;
use Carbon\Carbon;
use DB;

class ClientSubscriptionsAction extends BaseAction
{
    public $company_id;
    public $company_location_id;
    public $filter_client_id;
    public $filter_date_from;
    public $filter_date_to;
    public $filter_payment_source;
    public $filter_staff_id;

    public $per_page = 15;

  public $page = 1;

    public $clientSubsModel;
    public $viewParams;
```

Продовження Додатку А

```
        public function rules(): array
        {
            return [
                'company_id' => ['integer', 'required'],
                'company_location_id' => ['integer', 'required'],
                'per_page' => ['integer'],
                'page' => ['integer'],
                'filter_client_id' => ['nullable']
            ];
        }

        public function action()
        {
            $clients    =    app(CompanyClientRepository::class)-
>getClientList($this->company_id);
            $this->viewParams['clients']  =  ['' => __('пошук за
всіма клієнтами')] + $clients;

            $isOnlinePaymentShowingNeeded =
                CrmHelper::settings(



    CompanySettings::IS_ONLINE_PAYMENT_SHOWING_NEEDED,
                    false          );

            $staffs = $this->getStaffsList($this->company_id);

            if          (CrmHelper::getRoleInCompany()          ==
CompanyStaffRoles::ADMINISTRATOR) {
                $this->viewParams['staffs'] = $staffs;
            } else {
                $this->viewParams['staffs'] = ['' => __('пошук за
всіма співробітниками')] + $staffs;
```

```
    }

            $this->viewParams['items'] = [];
            $this->viewParams['back_url'] = url()->full();

            $subscriptions = ClientSubscription::query()
                ->where(['company_id' => $this->company_id])
                ->where(function ($query) {
                    $query->where(['company_location_id' => $this-
>company_location_id])
                        ->orWhere(['created_by' => null]);
                });

      if (CrmHelper::getRoleInCompany() != CompanyStaffRoles::OWNER
&& $isOnlinePaymentShowingNeeded) {
                $subscriptions->whereHas('order',         function
($query) {
                    $query->where('payment_source',          '!=',
PaymentSources::ONLINE);
                });
        }

            $this->clientSubsModel = $subscriptions
                ->with([
                    'user' => function ($query) {
                        $query->withTrashed();
                    },
                    'subscription' => function ($query) {
                        $query->withTrashed();
                    }, 'order'])
                ->orderByDesc('id');

            $this->viewParams['totalSubscriptionCount'] = $this-
```

Продовження Додатку А

```
    >clientSubsModel->count();

            if          (CrmHelper::getRoleInCompany()          ==
CompanyStaffRoles::ADMINISTRATOR) {
                $this->viewParams['filterIsUsed'] = true;

                $this->clientSubsModel->whereHas(

    'order',
                    function             ($query)             use
($isOnlinePaymentShowingNeeded) {
                        $query->where('created_by_user_id',   '=',
Auth::id());

                        if (!$isOnlinePaymentShowingNeeded) {
                            $query->orWhere('payment_source', '=',
PaymentSources::ONLINE);
                        }
    }
                );
            } else {
                $this->viewParams['filterIsUsed'] = false;
            }

            if ($this->filter_client_id) {
                $this->viewParams['filterIsUsed'] = true;

                $this->clientSubsModel->where([
                    'company_id' => $this->company_id,
                    'client_id' => $this->filter_client_id,
                ]);
            }
```

```
      if ($this->filter_date_from && $this->filter_date_to) {
                $this->viewParams['filterIsUsed'] = true;

                $this->clientSubsModel-
>whereBetween('created_at', [
                    Carbon::parse($this->filter_date_from)-
>startOfDay()->toDateTimeString(),
                    Carbon::parse($this->filter_date_to)-
>endOfDay()->toDateTimeString()
                ]);
            }

            if (!empty($this->filter_payment_source)) {
                $this->viewParams['filterIsUsed'] = true;

                $this->clientSubsModel->whereRaw(
                    DB::raw("order_id in (select id from `orders`
where `payment_source` = {$this->filter_payment_source})")
                );
            }

            if (!empty($this->filter_staff_id)) {
                $this->viewParams['filterIsUsed'] = true;
                $this->clientSubsModel->whereRaw(
                    DB::raw("order_id in (select id from `orders`
where `created_by_user_id` = {$this-

     >filter_staff_id})")
                );
            }

            $this->clientSubsModel    =    $this->clientSubsModel-
>paginate($this->per_page, ['*'], 'page', $this->page);
```

Продовження Додатку А

```
            $subsItem = [];
            $carbon = Carbon::now();
            foreach ($this->clientSubsModel->items() as $item) {
                $subsItem[$item->id]['id'] = $item->id;
                $subsItem[$item->id]['available_from'] = $item-
>formatDate('available_from') ?? '-';
                $subsItem[$item->id]['available_to'] = $item-
>formatDate('available_to') ?? '-';
                $subsItem[$item->id]['available_date'] =
$subsItem[$item->id]['available_from'] . ' / ' . $subsItem[$item-
>id]['available_to'];
                $subsItem[$item->id]['auto_prolongation'] = $item-
>subscription->auto_prolongation;
                $subsItem[$item->id]['created_at'] = $item-
>created_at;
                $subsItem[$item->id]['quantity_of_occupation'] =
$item->quantity_of_occupation == -1 ? __('Безліміт') : $item-
>quantity_of_occupation;

    $subsItem[$item->id]['active'] = ($item->available_to >
$carbon && $item->quantity_of_occupation !== 0) || ($item-
>available_to === null && $item->quantity_of_occupation !== 0);
                $subsItem[$item->id]['deleted_at'] = $item-
>deleted_at;

                $subsItem[$item->id]['client']['first_name'] =
$item->user->first_name;
                $subsItem[$item->id]['client']['last_name'] =
$item->user->last_name;
                $subsItem[$item->id]['client']['phone'] = $item-
>user->phone;
                $subsItem[$item->id]['client']['id'] = $item-
>user->id;
```

Продовження Додатку А

```
                $subsItem[$item->id]['subscription']['id']       =
$item->subscription->id;
                $subsItem[$item->id]['subscription']['title']    =
$item->subscription->title;
                $subsItem[$item-
>id]['subscription']['deleted_at']      =      $item->subscription-
>deleted_at;
                $subsItem[$item-
>id]['subscription']['is_freezed'] = $item->is_freezed;

                $subsItem[$item->id]['order']['id'] =

     $item->order_id;
                $subsItem[$item->id]['order']['amount']  =  $item-
>order->price ?? ' - ';
                $subsItem[$item->id]['discount'] = $item->discount
?? 0;
                $subsItem[$item->id]['order']['pay_source'] =
                    ($item->order->payment_source == '1')
                        ? __('Готівкою')
                        : (
                            ($item->order->payment_source == '2')
                        ? __('Безготівково')
                        : (
                            ($item->order->payment_source == '3')
                        ? __('Онлайн оплата')
                        : '-'
                        )
                        );
            }

            $this->viewParams['filteredSubscriptionCount'] =
```

Продовження Додатку А

```
    $this->clientSubsModel->total();

            $this->viewParams['clientSubsModel'] = $this->clientSubsModel;

    $this->viewParams['items'] = $subsItem;

        }

        public function getStaffsList($company_id): array
        {
            if (CrmHelper::getRoleInCompany() == CompanyStaffRoles::OWNER) {
                $staffs = CompanyStaff::query()
                    ->where(['company_id' => $company_id,])
                    ->with(['user'])
                    ->get();

                foreach ($staffs as $staff) {
                    $roleName = CompanyStaffRoles::getTitleRoleById($staff->role_in_company);
                    $response[$staff->user_id] = $staff->user->first_name . ' ' . $staff->user->last_name . '  ' . $staff->user->phone . ' (' . $roleName . ')';
                }
            } elseif (CrmHelper::getRoleInCompany() == CompanyStaffRoles::ADMINISTRATOR) {
                $staffs = CompanyStaff::query()->where([
                    'company_id' => $company_id,
                    'user_id' => Auth::id()
                ])->first();
```

Продовження Додатку А

```
    $roleName                                                    =
CompanyStaffRoles::getTitleRoleById(CrmHelper::getRoleInCompany())
;
                $response[$staffs->user_id]    =    $staffs->user-
>first_name . ' ' . $staffs->user->last_name . '  ' . $staffs->user-
>phone . ' (' . $roleName . ')';
    }

            return $response;
        }

    }
```

Додаток Б

Один з головних серверних класів

```
<?php

namespace Illuminate\Foundation\Http;

use Carbon\CarbonInterval;
use DateTimeInterface;
use Illuminate\Contracts\Debug\ExceptionHandler;
use Illuminate\Contracts\Foundation\Application;
use Illuminate\Contracts\Http\Kernel as KernelContract;
use Illuminate\Foundation\Http\Events\RequestHandled;
use Illuminate\Routing\Pipeline;
use Illuminate\Routing\Router;
use Illuminate\Support\Carbon;
use Illuminate\Support\Facades\Facade;
use Illuminate\Support\InteractsWithTime;
use InvalidArgumentException;
use Throwable;

class Kernel implements KernelContract
{
    use InteractsWithTime;

    /**
     * The application implementation.
     *
     * @var \Illuminate\Contracts\Foundation\Application
     */
    protected $app;

    /**
     * The router instance.
     *
```

Продовження Додатку Б

```
     * @var \Illuminate\Routing\Router
      */
     protected $router;

     /**
      * The bootstrap classes for the application.
      *
      * @var string[]
      */
     protected $bootstrappers = [

\Illuminate\Foundation\Bootstrap\LoadEnvironmentVariables::class,

\Illuminate\Foundation\Bootstrap\LoadConfiguration::class,

\Illuminate\Foundation\Bootstrap\HandleExceptions::class,

\Illuminate\Foundation\Bootstrap\RegisterFacades::class,

\Illuminate\Foundation\Bootstrap\RegisterProviders::class,

\Illuminate\Foundation\Bootstrap\BootProviders::class,
     ];

     /**
      * The application's middleware stack.
      *
      * @var array<int, class-string|string>
      */
     protected $middleware = [];

     /**
      * The application's route middleware groups.
```

Продовження Додатку Б

```
*
     * @var array<string, array<int, class-string|string>>
     */
    protected $middlewareGroups = [];

    /**
     * The application's route middleware.
     *
     * @var array<string, class-string|string>
     *
     * @deprecated
     */
    protected $routeMiddleware = [];

    /**
     * The application's middleware aliases.
     *
     * @var array<string, class-string|string>
     */
    protected $middlewareAliases = [];

    /**
     * All of the registered request duration handlers.
     *
     * @var array
     */
    protected $requestLifecycleDurationHandlers = [];

    /**
     * When the kernel starting handling the current request.
     *
     * @var \Illuminate\Support\Carbon|null
     */
    protected $requestStartedAt;
```

Продовження Додатку Б

```
        /**
         * The priority-sorted list of middleware.
         *
         * Forces non-global middleware to always be in the given
order.
         *
         * @var string[]
         */
        protected $middlewarePriority = [

\Illuminate\Foundation\Http\Middleware\HandlePrecognitiveRequests:
:class,
            \Illuminate\Cookie\Middleware\EncryptCookies::class,

\Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,
            \Illuminate\Session\Middleware\StartSession::class,

\Illuminate\View\Middleware\ShareErrorsFromSession::class,

\Illuminate\Contracts\Auth\Middleware\AuthenticatesRequests::class
,

\Illuminate\Routing\Middleware\ThrottleRequests::class,

\Illuminate\Routing\Middleware\ThrottleRequestsWithRedis::class,

\Illuminate\Contracts\Session\Middleware\AuthenticatesSessions::cl
ass,

\Illuminate\Routing\Middleware\SubstituteBindings::class,
            \Illuminate\Auth\Middleware\Authorize::class,
        ];
```

Продовження Додатку Б

```
    /**
     * Create a new HTTP kernel instance.
     *
     *  @param    \Illuminate\Contracts\Foundation\Application
$app
     * @param  \Illuminate\Routing\Router  $router
     * @return void
     */
    public function __construct(Application $app, Router
$router)
    {
        $this->app = $app;
        $this->router = $router;

        $this->syncMiddlewareToRouter();
    }

    /**
     * Handle an incoming HTTP request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function handle($request)
    {
        $this->requestStartedAt = Carbon::now();

        try {
            $request->enableHttpMethodParameterOverride();

            $response                =                $this-
>sendRequestThroughRouter($request);
        } catch (Throwable $e) {
```

Продовження Додатку Б

```
        $this->reportException($e);

            $response = $this->renderException($request, $e);
        }

        $this->app['events']->dispatch(
            new RequestHandled($request, $response)
        );

        return $response;
    }

    /**
     * Send the given request through the middleware / router.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    protected function sendRequestThroughRouter($request)
    {
        $this->app->instance('request', $request);

        Facade::clearResolvedInstance('request');

        $this->bootstrap();

        return (new Pipeline($this->app))
                    ->send($request)
                    ->through($this->app->shouldSkipMiddleware() ? [] : $this->middleware)
                    ->then($this->dispatchToRouter());
    }
```

Продовження Додатку Б

```
    /**
     * Bootstrap the application for HTTP requests.
     *
     * @return void
     */
    public function bootstrap()
    {
        if (! $this->app->hasBeenBootstrapped()) {
            $this->app->bootstrapWith($this-
>bootstrappers());
        }
    }

    /**
     * Get the route dispatcher callback.
     *
     * @return \Closure
     */
    protected function dispatchToRouter()
    {
        return function ($request) {
            $this->app->instance('request', $request);

            return $this->router->dispatch($request);
        };
    }

    /**
     * Call the terminate method on any terminable middleware.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Illuminate\Http\Response  $response
     * @return void
```

Продовження Додатку Б

```
     */
        public function terminate($request, $response)
        {
            $this->terminateMiddleware($request, $response);

            $this->app->terminate();

            if ($this->requestStartedAt === null) {
                return;
            }

            $this->requestStartedAt->setTimezone($this-
>app['config']->get('app.timezone') ?? 'UTC');

            foreach ($this->requestLifecycleDurationHandlers as
['threshold' => $threshold, 'handler' => $handler]) {
                $end ??= Carbon::now();

                if ($this->requestStartedAt-
>diffInMilliseconds($end) > $threshold) {
                    $handler($this->requestStartedAt, $request,
$response);
                }
            }

            $this->requestStartedAt = null;
        }

        /**
         * Call the terminate method on any terminable middleware.
         *
         * @param  \Illuminate\Http\Request  $request
         * @param  \Illuminate\Http\Response  $response
```

Продовження Додатку Б

```
     * @return void
         */
        protected      function      terminateMiddleware($request,
$response)
        {
            $middlewares = $this->app->shouldSkipMiddleware() ? []
: array_merge(
                $this->gatherRouteMiddleware($request),
                $this->middleware
            );

            foreach ($middlewares as $middleware) {
                if (! is_string($middleware)) {
                    continue;
                }

                [$name] = $this->parseMiddleware($middleware);

                $instance = $this->app->make($name);

                if (method_exists($instance, 'terminate')) {
                    $instance->terminate($request, $response);
                }
            }
        }

        /**
         * Register a callback to be invoked when the requests
lifecycle duration exceeds a given amount of time.
         *
         *                                                    @param
\DateTimeInterface|\Carbon\CarbonInterval|float|int  $threshold
         * @param  callable  $handler
```

Продовження Додатку Б

```
        * @return void
         */
        public function whenRequestLifecycleIsLongerThan($threshold, $handler)
        {
            $threshold = $threshold instanceof DateTimeInterface
                ? $this->secondsUntil($threshold) * 1000
                : $threshold;

            $threshold = $threshold instanceof CarbonInterval
                ? $threshold->totalMilliseconds
                : $threshold;

            $this->requestLifecycleDurationHandlers[] = [
                'threshold' => $threshold,
                'handler' => $handler,
            ];
        }

        /**
         * When the request being handled started.
         *
         * @return \Illuminate\Support\Carbon|null
         */
        public function requestStartedAt()
        {
            return $this->requestStartedAt;
        }

        /**
         * Gather the route middleware for the given request.
         *
         * @param  \Illuminate\Http\Request  $request
```

Продовження Додатку Б

```
     * @return array
      */
     protected function gatherRouteMiddleware($request)
     {
         if ($route = $request->route()) {
             return                                $this->router-
>gatherRouteMiddleware($route);
         }

         return [];
     }

     /**
      * Parse a middleware string to get the name and parameters.
      *
      * @param  string  $middleware
      * @return array
      */
     protected function parseMiddleware($middleware)
     {
         [$name,    $parameters]    =    array_pad(explode(':',
$middleware, 2), 2, []);

         if (is_string($parameters)) {
             $parameters = explode(',', $parameters);
         }

         return [$name, $parameters];
     }

     /**
      * Determine if the kernel has a given middleware.
      *
```

```
       * @param  string  $middleware
        * @return bool
        */
       public function hasMiddleware($middleware)
       {
           return in_array($middleware, $this->middleware);
       }

       /**
        * Add a new middleware to the beginning of the stack if it
does not already exist.
        *
        * @param  string  $middleware
        * @return $this
        */
       public function prependMiddleware($middleware)
       {
           if (array_search($middleware, $this->middleware) ===
false) {
               array_unshift($this->middleware, $middleware);
           }

           return $this;
       }

       /**
        * Add a new middleware to end of the stack if it does not
already exist.
        *
        * @param  string  $middleware
        * @return $this
        */
       public function pushMiddleware($middleware)
```

Продовження Додатку Б

```
        {
            if (array_search($middleware, $this->middleware) ===
false) {
                $this->middleware[] = $middleware;
            }

            return $this;
        }

        /**
         * Prepend the given middleware to the given middleware
group.
         *
         * @param  string  $group
         * @param  string  $middleware
         * @return $this
         *
         * @throws \InvalidArgumentException
         */
        public      function      prependMiddlewareToGroup($group,
$middleware)
        {
            if (! isset($this->middlewareGroups[$group])) {
                throw new InvalidArgumentException("The [{$group}]
middleware group has not been defined.");
            }

            if          (array_search($middleware,          $this-
>middlewareGroups[$group]) === false) {
                array_unshift($this->middlewareGroups[$group],
$middleware);
            }
```

Продовження Додатку Б

```
        $this->syncMiddlewareToRouter();

            return $this;
        }

        /**
         * Append the given middleware to the given middleware
group.
         *
         * @param  string  $group
         * @param  string  $middleware
         * @return $this
         *
         * @throws \InvalidArgumentException
         */
        public function appendMiddlewareToGroup($group,
$middleware)
        {
            if (! isset($this->middlewareGroups[$group])) {
                throw new InvalidArgumentException("The [{$group}]
middleware group has not been defined.");
            }

            if (array_search($middleware, $this-
>middlewareGroups[$group]) === false) {
                $this->middlewareGroups[$group][] = $middleware;
            }

            $this->syncMiddlewareToRouter();

            return $this;
        }
```

Продовження Додатку Б

```
    /**
     * Prepend the given middleware to the middleware priority
list.
     *
     * @param  string  $middleware
     * @return $this
     */
    public function prependToMiddlewarePriority($middleware)
    {
        if (! in_array($middleware, $this-
>middlewarePriority)) {
            array_unshift($this->middlewarePriority,
$middleware);
        }

        $this->syncMiddlewareToRouter();

        return $this;
    }

    /**
     * Append the given middleware to the middleware priority
list.
     *
     * @param  string  $middleware
     * @return $this
     */
    public function appendToMiddlewarePriority($middleware)
    {
        if (! in_array($middleware, $this-
>middlewarePriority)) {
            $this->middlewarePriority[] = $middleware;
        }
```

Продовження Додатку Б

```
        $this->syncMiddlewareToRouter();

            return $this;
        }

        /**
         * Sync the current state of the middleware to the router.
         *
         * @return void
         */
        protected function syncMiddlewareToRouter()
        {
            $this->router->middlewarePriority = $this->middlewarePriority;

            foreach ($this->middlewareGroups as $key => $middleware) {
                $this->router->middlewareGroup($key, $middleware);
            }

            foreach (array_merge($this->routeMiddleware, $this->middlewareAliases) as $key => $middleware) {
                $this->router->aliasMiddleware($key, $middleware);
            }
        }

        /**
         * Get the priority-sorted list of middleware.
         *
         * @return array
         */
        public function getMiddlewarePriority()
```

Продовження Додатку Б

```
{
        return $this->middlewarePriority;
    }

    /**
     * Get the bootstrap classes for the application.
     *
     * @return array
     */
    protected function bootstrappers()
    {
        return $this->bootstrappers;
    }

    /**
     * Report the exception to the exception handler.
     *
     * @param  \Throwable  $e
     * @return void
     */
    protected function reportException(Throwable $e)
    {
        $this->app[ExceptionHandler::class]->report($e);
    }

    /**
     * Render the exception to a response.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Throwable  $e
     * @return \Symfony\Component\HttpFoundation\Response
     */
    protected function renderException($request, Throwable $e)
```

Продовження Додатку Б

```
    {
            return            $this->app[ExceptionHandler::class]-
>render($request, $e);
        }

        /**
         * Get the application's route middleware groups.
         *
         * @return array
         */
        public function getMiddlewareGroups()
        {
            return $this->middlewareGroups;
        }

        /**
         * Get the application's route middleware aliases.
         *
         * @return array
         *
         * @deprecated
         */
        public function getRouteMiddleware()
        {
            return $this->getMiddlewareAliases();
        }

        /**
         * Get the application's route middleware aliases.
         *
         * @return array
         */
        public function getMiddlewareAliases()
```

Продовження Додатку Б

```
    {
            return   array_merge($this->routeMiddleware,   $this-
>middlewareAliases);
        }

        /**
         * Get the Laravel application instance.
         *
         * @return \Illuminate\Contracts\Foundation\Application
         */
        public function getApplication()
        {
            return $this->app;
        }

        /**
         * Set the Laravel application instance.
         *
         *  @param    \Illuminate\Contracts\Foundation\Application
$app
         * @return $this
         */
        public function setApplication(Application $app)
        {
            $this->app = $app;

            return $this;
        }
    }
```

Додаток В

Контроллер користувачів

```
<?php

namespace App\Admin\Controllers;

use App\Models\User;
use Encore\Admin\Controllers\AdminController;
use Encore\Admin\Form;
use Encore\Admin\Grid;
use Encore\Admin\Show;

class UsersController extends AdminController
{
    /**
     * Title for current resource.
     *
     * @var string
     */
    protected $title = User::class;

    /**
     * Make a grid builder.
     *
     * @return Grid
     */
    protected function grid()
    {
        $grid = new Grid(new User());

        $grid->column('id', __('Id'));
        $grid->column('first_name', __('First name'));
        $grid->column('last_name', __('Last name'));
        $grid->column('phone', __('Phone'));
```

```
                $grid->column('created_at', __('Created at'));
        $grid->column('created_by', __('Created by'));

        return $grid;
    }

    /**
     * Make a show builder.
     *
     * @param mixed $id
     * @return Show
     */
    protected function detail($id)
    {
        $show = new Show(User::findOrFail($id));

        $show->field('id', __('Id'));
        $show->field('first_name', __('First name'));
        $show->field('last_name', __('Last name'));
        $show->field('phone', __('Phone'));
        $show->field('email', __('Email'));
        $show->field('email_verified_at',  __('Email  verified
at'));
        $show->field('password', __('Password'));
        $show->field('remember_token', __('Remember token'));
        $show->field('created_at', __('Created at'));
        $show->field('updated_at', __('Updated at'));
        $show->field('created_by', __('Created by'));

        return $show;
    }

    /**
```

Продовження Додатку В

```
    * Make a form builder.
         *
         * @return Form
         */
        protected function form()
        {
            $form = new Form(new User());

            $form->text('first_name', __('First name'));
            $form->text('last_name', __('Last name'));
            $form->mobile('phone', __('Phone'));
            $form->email('email', __('Email'));
            $form->datetime('email_verified_at',         __('Email
verified at'))->default(date('Y-m-d H:i:s'));
            $form->password('password', __('Password'));
            $form->text('remember_token', __('Remember token'));
            $form->text('created_by', __('Created by'));

            return $form;
        }
    }
```