МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

**Допущено до захисту:**

Завідувач кафедри
комп'ютерних технологій та
економічної кібернетики
д. е. н., проф. П. М. Грицюк

_____
«____» _____ 2025 р.

# КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня «магістр»
за освітньо-професійною програмою
«Інформаційні технології в бізнесі»
спеціальності 126 «Інформаційні системи та технології»
на тему: **«Реалізація вебплатформи для запису на персональні тренування із застосуванням Node.js»**

**Виконав**:
здобувач вищої освіти 2 курсу,
групи ІТБ-61м
Юрко Василь Олександрович

**Керівник**:
к.е.н., доцент Волошин В.С.

**Рецензент**:
д.е.н., проф. Грицюк П. М.

Рівне – 2025

## РЕФЕРАТ

**Кваліфікаційна робота магістра: 54 с., 22 рис., 5 табл., 28 літературних джерел.**

Актуальність теми магістерської роботи зумовлена зростанням попиту на цифрові сервіси для організації персональних тренувань і фітнес-процесів. Сучасні вебплатформи дозволяють автоматизувати взаємодію між клієнтами та тренерами, оптимізувати розклад тренувань і забезпечити оперативний доступ до даних. Використання технології Node.js підвищує ефективність розробки завдяки високій продуктивності, масштабованості та можливості створення швидких серверних застосунків у середовищі JavaScript.

**Об'єкт дослідження** – процес організації взаємодії користувача з вебплатформою для персональних тренувань.

**Предмет дослідження** – методи та технології розробки вебзастосунку для запису на персональні тренування з використанням **Node.js**, Express, MySQL, а також HTML, CSS, JavaScript для формування інтерфейсу користувача.

**Мета роботи** – розробити вебплатформу, що забезпечує можливість перегляду тренерів, вибору доступного часу та оформлення заявки на персональне тренування із застосуванням сучасних вебтехнологій.

**У магістерській роботі:**

- обґрунтовано важливість цифровізації сфери персонального тренінгу;
- здійснено огляд існуючих платформ для онлайн-запису в спортзали та фітнес-центри;
- проведено аналіз кросплатформеного середовища Node.js для реалізації вебсервісів;
- розроблено структуру бази даних для обробки заявок;
- створено вебплатформу для запису на персональні тренування;
- забезпечено інтеграцію з MySQL для збереження та обробки даних.

**КЛЮЧОВІ СЛОВА:** ВЕБПЛАТФОРМА, ПЕРСОНАЛЬНІ ТРЕНУВАННЯ, ІНФОРМАЦІЙНА СИСТЕМА, NODE.JS, БАЗА ДАНИХ, JAVASCRIPT, EXPRESS, MYSQL.

# ЗМІСТ

# ВСТУП

У сучасному суспільстві цифровізація охоплює практично всі сфери життя, і фітнес-індустрія не є винятком. Попит на персональні тренування зростає, а разом із ним – запит на інструменти, які спрощують комунікацію між клієнтами та тренерами, полегшують процес бронювання занять і забезпечують прозорість та доступність розкладу. Традиційні способи взаємодії, такі як телефонні дзвінки чи ручне ведення календарів, поступово відходять у минуле, оскільки не відповідають швидкості, зручності та структурованості, якої очікує сучасний користувач.

Вебплатформи для персональних тренувань стають важливим елементом інфраструктури спортивних клубів та індивідуальних тренерів. Вони дозволяють оптимізувати робочий час, уникати помилок з бронюванням і надавати клієнтам можливість самостійно обирати тренера та доступні години. Такі системи також створюють умови для формування персонального підходу: користувач отримує можливість переглядати інформацію про тренерів, їхню спеціалізацію, доступні слоти та додаткові послуги.

Розробка вебплатформи, що поєднує функціональність, простоту використання та достатню гнучкість, є актуальним завданням у межах цифрової трансформації спортивних сервісів. Технологія Node.js у цьому контексті є одним з найбільш ефективних інструментів. Її використання забезпечує високу продуктивність серверної частини, можливість роботи з великою кількістю одночасних запитів і зручність створення REST API. Поєднання Node.js з HTML, CSS, JavaScript та реляційною базою даних MySQL дозволяє створити цілісний, масштабований і технологічно сучасний продукт.

**Метою магістерської роботи** є проєктування та розробка вебплатформи для запису на персональні тренування, що забезпечує перегляд тренерів, формування заявок, контроль доступних часових слотів та роботу з базою даних у режимі реального часу.

**Об’єкт дослідження** – процес управління записом на персональні тренування в цифровому середовищі.

**Предмет дослідження** – вебплатформа для запису на персональні тренування, розроблена з використанням Node.js та супутніх вебтехнологій.

Для досягнення поставленої мети у роботі необхідно вирішити такі завдання:

1. Визначити особливості цифрової взаємодії між тренером і клієнтом у сфері персонального тренінгу.
2. Проаналізувати існуючі рішення для онлайн-запису в спортивній індустрії.
3. Дослідити можливості Node.js та пов’заних технологій для створення серверної частини вебплатформи.
4. Побудувати схему бази даних, що забезпечує роботу із заявками, тренерами та користувачами.
5. Реалізувати функціонал реєстрації заявок, перегляду тренерів та бронювання часу з урахуванням унікальності тренерського слоту.
6. Розробити інтерфейс користувача та адміністративну частину вебплатформи.
7. Провести тестування роботи платформи й оцінити її ефективність.

Структура магістерської роботи відповідає логіці дослідження.

У **вступі** обґрунтовується актуальність теми, визначаються об’кт, предмет, мета та завдання. **Перший розділ** присвячений аналізу цифрових рішень у сфері персональних тренувань та огляду технологій, що можуть бути використані для реалізації платформи, з акцентом на перевагах Node.js.
У **другому розділі** наведено моделі даних та опис функціональних вимог, необхідних для роботи системи. **Третій розділ** містить опис розробленого програмного забезпечення, його архітектури, принципів роботи серверної частини, реалізації інтерфейсу та механізмів обмеження дубльованих

бронювань. У **висновках** наведено результати виконаної роботи та сформульовано перспективи подальшого розвитку платформи.

Під час розробки використовувалися такі програмні засоби, як **Visual Studio Code, Node.js, MySQL Workbench, Express.js, HTML, CSS, JavaScript** та інші інструменти веброзробки.

# РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ З НАДАННЯ ПОСЛУГ ПЕРСОНАЛЬНОГО ТРЕНУВАННЯ

## 1.1. Поняття та сутність персонального тренування як послуги

Персональне тренування в сучасній фітнес-індустрії розглядається як окремий вид послуги, що поєднує елементи фізичної підготовки, консультування з питань здоров' та способу життя, а також сервісну взаємодію між клієнтом і тренером. У міжнародній практиці персональний тренер визначається як фахівець, який, використовуючи індивідуальний підхід, оцінює фізичний стан клієнта, ставить цілі, розробляє безпечну й ефективну програму занять, навчає техніці виконання вправ, мотивує та відстежує прогрес.

Більшість профільних організацій (NSCA, ACE, NASM тощо) підкреслюють, що персональний тренер працює в межах визначеного «scope of practice», зосереджуючись на здорових клієнтах або тих, хто має медичний допуск. До його компетенцій відносять оцінку рівня фізичної підготовки, підбір навантажень, навчання правильній техніці, базові рекомендації щодо способу життя та, за потреби, перенаправлення до лікаря або дієтолога.

В українському контексті сервіс персональних тренувань активно просувається мережею фітнес-клубів як преміум-продукт, що забезпечує «максимальну ефективність тренування», індивідуальний підхід до клієнта та контроль техніки виконання вправ. Наприклад, мережа Sport Life акцентує увагу на структурованості заняття, контролі пауз відпочинку, підборі навантажень під цілі клієнта.

Аналогічно, інші українські клуби позиціонують персональне тренування як «шанс поліпшити фігуру і самопочуття, стати здоровішим, сильнішим, активізувати захисні сили організму».

Онлайн- та офлайн-опис послуги персональних тренувань у більшості клубів має спільні риси:

- діагностика та постановка цілей (схуднення, набір м’зової маси, реабілітація, підготовка до змагань тощо);
- розробка індивідуальної програми (план тренувань, підбір вправ, планування навантажень по тижнях або мікроциклах);
- контроль за технікою виконання (корекція рухових помилок, профілактика травм, навчання базовим принципам безпеки);
- мотиваційна та освітня складова (формування тренувальних звичок, пояснення логіки навантажень, елементарна освіта з харчування та відновлення);
- моніторинг прогресу (порівняння вихідних та поточних показників, корекція плану тренувань).

Сутність персонального тренування як послуги полягає не лише у «продажі часу тренера», а у створенні індивідуалізованої послуги із високою доданою вартістю. За результатами досліджень фітнес-індустрії, особистий тренер виступає центральною фігурою сервісної взаємодії: від його компетентності, емоційного інтелекту та здатності підтримувати контакт залежить лояльність клієнтів, їхня готовність дотримуватись рекомендацій та продовжувати співпрацю.

Окремо варто виділити емоційний компонент послуги. Персональні тренування часто сприймаються клієнтами як простір довіри та підтримки, де тренер виступає не тільки інструктором, а й мотиватором, партнером, подекуди – «коучем» зі змін способу життя. Саме поєднання експертності (знання анатомії, фізіології, методики тренувань) і сервісних навичок (комунікація, емпатія, керування очікуваннями клієнта) формує повноцінну сутність персонального тренування як послуги.

Сучасні тренди демонструють перехід від суто «залового» формату до гібридних моделей: поєднання офлайн-занять із онлайн-супроводом, мобільними застосунками, дистанційними консультаціями, відеотренуваннями.

Це розширює межі класичного персонального тренування, але зберігає його сутність – персоналізацію, трекінг прогресу, регулярну взаємодію клієнта і тренера.

З точки зору інформаційних технологій, персональне тренування як послуга є типовим прикладом процесу, який добре піддається цифровізації:

- є чітко визначені ролі (клієнт, тренер, адміністратор клубу);
- присутні сутності даних (клієнти, тренери, розклад, бронювання, історія тренувань);
- існують повторювані сценарії (запис на тренування, скасування, перенесення, оплата);
- є вимоги до обмеження ресурсів (один тренер не може одночасно тренувати двох клієнтів в один і той же час).

Таким чином, з позицій ІТ-фахівця, персональні тренування як послуга логічно трансформуються у модель, що реалізується через вебплатформу: клієнт працює з інтерфейсом, тренер – із власним кабінетом, а бекенд-система забезпечує керування даними та бізнес-логікою.

## 1.2. Аналіз сучасних онлайн платформ

Розвиток персонального тренінгу тісно пов’заний із появою та еволюцією онлайн-платформ, які автоматизують ключові процеси: від запису на тренування до ведення прогресу та дистанційного коучингу. Для аналізу доцільно виділити кілька типів рішень:

1) Системи онлайн-запису для тренерів та фітнес-студій

Сервіси на кшталт SimplyBook.me, Zenamu, EasyWeek, Time2book та інші позиціонуються як універсальні системи онлайн-бронювання для тренерів, студій, фітнес-клубів, шкіл танців тощо. Вони надають можливості: створення розкладу, прийому онлайн-заявок, управління клієнтською базою, інтеграції з оплатами, SMS/e-mail-нагадуваннями та CRM-функціями.

Для прикладу, SimplyBook.me прямо орієнтується на фітнес-інструкторів та тренерів, пропонуючи рішення для бронювання приватних тренувань і групових занять, управління заняттями та нагадувань клієнтам.

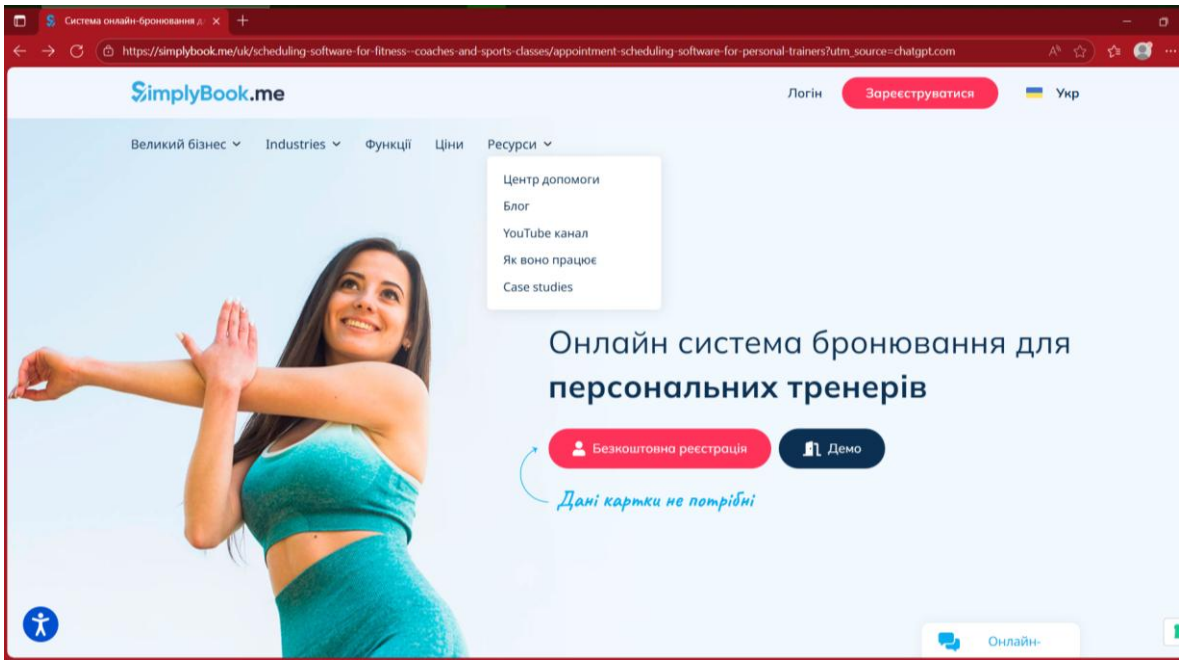


**Рис 1.1** Сторінка SimplyBook.me

Zenamu позиціонує себе як онлайн-систему для студій йоги, фітнесу та інших активностей, з акцентом на простому онлайн-розкладі та оплатах.

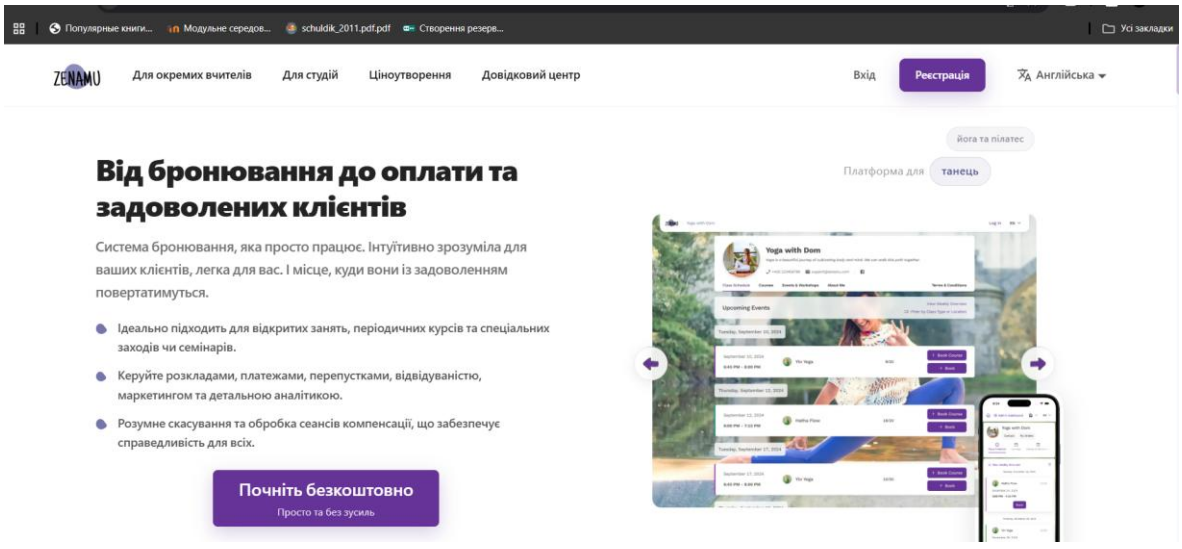


**Рис 1.2** Система Zenamu

EasyWeek та Time2book фокусуються на автоматизації записів, інтеграції календарів і спрощенні роботи для малого бізнесу у сфері фітнесу.

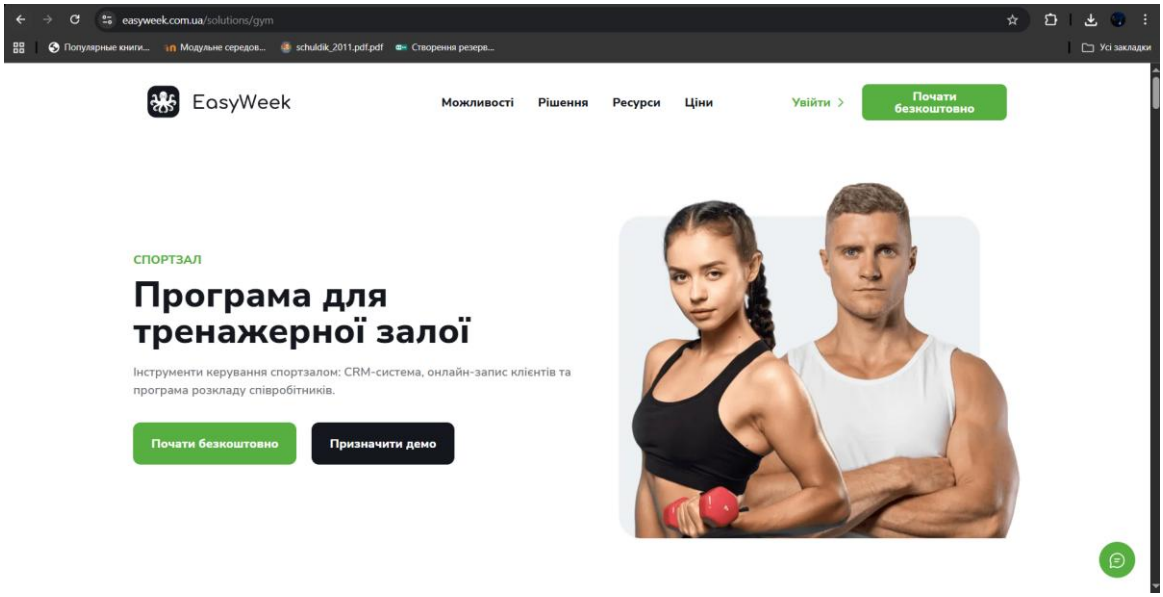


**Рис. 1.3** Платформа EasyWeek


**Рис. 1.4** Онлайн-сервіс Time2book

Переваги цих рішень:

- швидкий старт (SaaS-модель, немає потреби самостійно розгортати сервер);
- готові модулі онлайн-запису, календарів, сповіщень;
- інтеграція з платіжними сервісами;
- кросплатформеність (веб + мобільні версії).

Недоліки:

- відсутність глибокої кастомізації під конкретні бізнес-процеси;
- залежність від зовнішньої платформи (питання безпеки та зберігання даних);

- оплата за підпискою, що може бути критичним для невеликих студій чи індивідуальних тренерів;
- обмежені можливості щодо розширення функціоналу з боку розробника.

2) Програмні платформи для персональних тренерів

До цієї групи належать продукти на кшталт TrueCoach, FITR, Trainerize, які орієнтовані на тренерів, що працюють як офлайн, так і онлайн, та ведуть клієнтів дистанційно. TrueCoach позиціонується як «№1 платформа для персональних тренерів», що дає змогу створювати й надсилати програми, отримувати звіти клієнтів, зберігати історію тренувань і прогресу, а також автоматизувати рутинні дії.

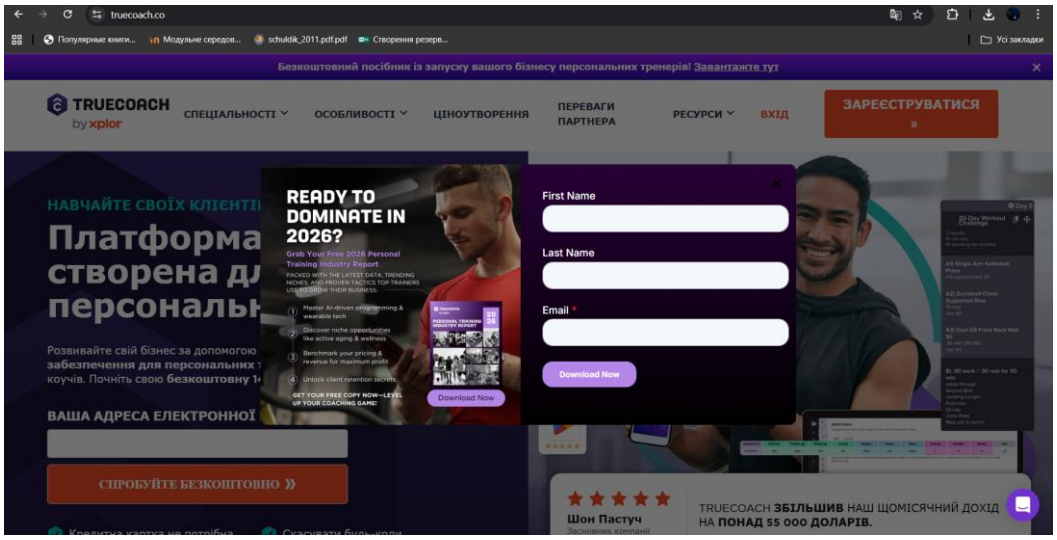


**Рис. 1.5** Платформа для персональних тренерів TrueCoach

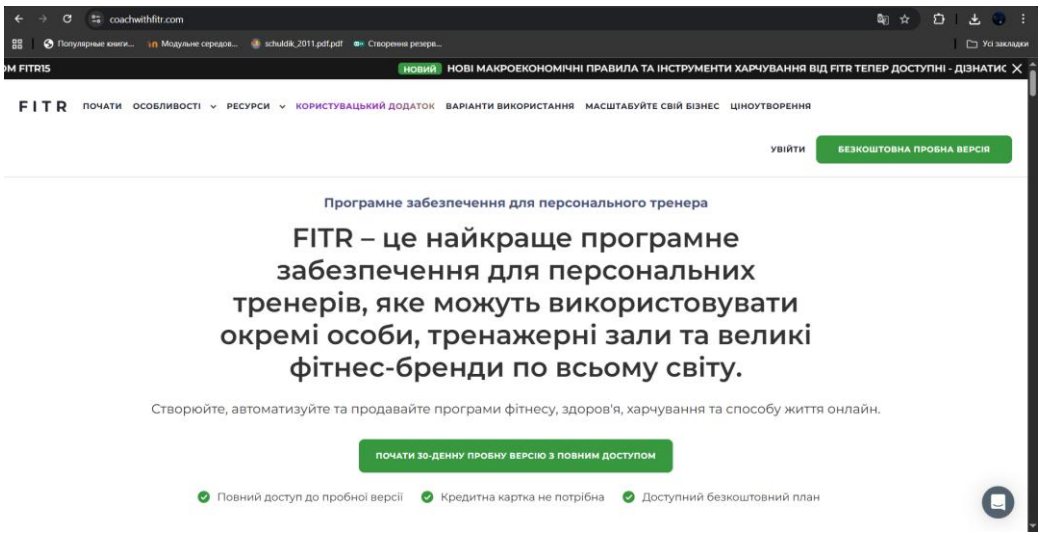


**Рис. 1.6** Система FITR

FITR робить акцент на створенні та продажу програм тренувань, харчування, комплексних lifestyle-протоколів для індивідуальних клієнтів і брендів.

Подібні системи фокусуються на:

- конструюванні індивідуальних і пакетних програм тренувань;
- відстеженні прогресу клієнта (вага, робочі ваги, обсяги навантаження тощо);
- комунікації з клієнтами (чат, коментарі, відео);
- іноді – внутрішній аналітиці (утримання клієнтів, аналіз виконання програм).

Ці платформи є більш «спеціалізованими» у порівнянні з універсальними booking-системами, однак мають схожі обмеження: закритий вихідний код, залежність від бізнес-моделі сервісу, складність повної інтеграції у власну інфраструктуру.

3) Локальні сайти фітнес-клубів з розділом «Персональний тренер»

Окрему категорію становлять вебсайти фітнес-мереж, які містять інформаційні розділи про персональні тренування, але не завжди мають повноцінний модуль онлайн-запису. Наприклад, розділ «Персональний тренер» мережі Sport Life подає опис переваг персональних тренувань, аргументи на користь роботи з тренером, але механіка запису часто зводиться до форми зворотного зв’зку чи загальної заявки.

Подібний підхід використовують й інші українські клуби (Green Hills fitness & spa, hiitworks тощо), роблячи акцент на маркетинговому представленні послуги, а не на глибокій цифровій автоматизації процесів.

Ці сайти цінні для нашого дослідження як джерело структури подачі інформації (як описують тренерів, програми, переваги), але вони не демонструють повноцінний сценарій взаємодії «користувач – вебплатформа – база даних – тренер» у вигляді, який ми будуємо на Node.js.

4) Наукові та прикладні дослідження онлайн-платформ для фітнесу

Важливим є не тільки аналіз комерційних сервісів, але й розгляд наукових робіт, які описують архітектуру та ефективність онлайн-систем для фітнесу. Так, у роботі «Online Gym Booking System (OGBS)» описано вебсистему, яка дає змогу клієнтам бронювати заняття та персональні тренування онлайн, а менеджменту – відстежувати завантаженість та оперативно опрацьовувати дані.

Дослідження демонструє типову архітектуру: вебінтерфейс, серверна частина, база даних, модуль управління розкладом – що багато в чому збігається з логікою нашого проєкту.

Інше дослідження, присвячене віртуальним фітнес-тренерам, показує, що використання цифрових платформ може підвищувати залученість студентів до фізичної активності, якщо застосовується грамотний дизайн інтерфейсу та механізми зворотного зв’зку.

Це ще раз підкреслює, що вебплатформа – не лише інструмент бронювання, а й компонент мотиваційного середовища.

Окрему увагу привертають аналітичні публікації щодо онлайн-тренінгу та віртуального коучингу, де розглядається перехід індустрії до «цифрової революції» у фітнесі: використання відеозв’зку, спеціалізованих застосунків для тренувань, платформ для віддаленого супроводу клієнтів.

Вони підтверджують, що попит на гнучкі, персоналізовані онлайн-рішення буде лише зростати.

З урахуванням аналізу, наведемо узагальнену порівняльну характеристику кількох типових платформ, близьких до тематики персональних тренувань (таблиця 1.1):

Таблиця 1.1

**Порівняльна характеристика онлайн-платформ у сфері персональних тренувань**

| **Платформа** | **Тип рішення** | **Основні функції** | **Модель монетизації** | **Особливості** |
|---|---|---|---|---|
| SimplyBook.me | SaaS-сервіс онлайн-запису | Онлайн-бронювання, розклад, | Підписка, тарифні плани | Орієнтація на різні послуги, у т.ч. фітнес і тренерів |

| | | | | |
|---|---|---|---|---|
| | | нагадування, онлайн-оплати | | |
| Zenamu | Онлайн-система бронювання для студій | Розклад занять, онлайн-запис, управління групами, оплати | Підписка | Фокус на студіях йоги, танців, фітнесу |
| TrueCoach | ПЗ для персональних тренерів | Планування тренувань, спостереження за прогресом, комунікація з клієнтами | Підписка, freemium-підхід | Орієнтація на дистанційне тренування, тисячі тренерів у світі |
| FITR | Платформа для продажу програм | Створення і продаж програм, трекінг прогресу, робота з брендами | Підписка, платні плани | Підтримка індивідуальних тренерів та великих фітнес-брендів |
| Sportlife.ua | Інформаційний розділ сайту клубу | Опис послуги, форми зворотного зв’зку, маркетинговий опис переваг | Входить до абонементів | Немає повноцінного модулю онлайн-бронювання під кожного тренера |

Таблиця демонструє, що на ринку домінують або універсальні SaaS-рішення з «надлишковим» для окремого клубу функціоналом, або глобальні платформи для тренерів, тісно інтегровані у їхні бізнес-процеси, але закриті для кастомної розробки. Локальні сайти українських мереж фітнес-клубів, натомість, часто мають лише базові форми зворотного зв’зку без глибокої автоматизації.
Саме тому розробка власної вебплатформи для запису на персональні тренування із застосуванням Node.js є технічно обґрунтованою. Вона дозволяє:

- реалізувати REST API, яке чітко відповідає бізнес-логіці конкретного закладу;
- будувати власну модель даних (користувачі, тренери, бронювання, прогрес);
- контролювати унікальність часових слотів (один тренер – один клієнт у конкретний час);
- легко розширювати функціонал (кабінет адміністратора, аналітика, інтеграції) без обмежень зовнішніх платформ.

## Висновки по розділу 1

У першому розділі було визначено, що персональне тренування є сучасною та ефективною формою фітнес-послуги, спрямованою на досягнення індивідуальних цілей клієнта за допомогою професійного супроводу тренера. Такий формат охоплює не лише підбір фізичних вправ, а й контроль прогресу, корекцію плану тренувань, мотиваційну підтримку та аналіз стану здоров'. Персональні тренування стають дедалі популярнішими завдяки індивідуалізованим підходам, що дозволяють підвищити ефективність занять та забезпечити стабільні результати.

Проведений огляд сучасних онлайн-платформ у сфері фітнесу засвідчив значну різноманітність цифрових сервісів, які пропонують функції планування тренувань, віддаленого супроводу, бронювання занять та відстеження прогресу. Разом з тим багато рішень є комерційно дорогими, громіздкими для індивідуальних тренерів або обмеженими функціонально. Це створює потребу в доступній, гнучкій та адаптивній вебплатформі, яка б поєднувала ключові можливості фітнес-сервісів і була орієнтована на взаємодію «клієнт–тренер».

Також здійснено порівняльний аналіз технологій, що використовуються у веброзробці, зокрема платформ і фреймворків для створення серверних частин вебзастосунків. Визначено, що Node.js є оптимальним вибором для побудови вебплатформи персональних тренувань завдяки своїй продуктивності, неблокуючій моделі виконання, простоті масштабування та можливості реалізувати клієнтську й серверну логіку однією мовою — JavaScript. Це дає змогу створити швидкий та зручний сервіс, який ефективно обробляє взаємодію користувачів у реальному часі.

Отже, розділ забезпечив теоретичне підґрунтя для подальшої розробки вебплатформи: було визначено сутність персональних тренувань як послуги, здійснено аналіз сучасних цифрових рішень у фітнес-індустрії та обґрунтовано вибір технологічного середовища Node.js. Отримані висновки використовуватимуться в наступних розділах роботи для побудови моделі даних, проектування та реалізації програмної частини системи.

# РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБПЛАТФОРМИ ДЛЯ ЗАПИСУ НА ПЕРСОНАЛЬНІ ТРЕНУВАННЯ

## 2.1. Порівняльна характеристика фреймворків Node.js та Laravel

Вибір фреймворку для серверної частини вебплатформи є ключовим технічним рішенням, оскільки від нього залежать продуктивність системи, можливості масштабування, зручність підтримки коду та подальше розширення функціоналу. У межах даної магістерської роботи розглядаються два популярних підходи: використання Node.js **та** Laravel як представника сучасних PHP-фреймворків.

Node.js – це середовище виконання JavaScript на стороні сервера, побудоване на базі високопродуктивного двигуна V8 (розробка компанії Google). Основна ідея Node.js полягає в використанні однопотокової, подієво-орієнтованої, неблокуючої моделі введення/виведення**,** що дає змогу обробляти велику кількість одночасних запитів без створення множини потоків операційної системи.

Подієва архітектура (event-driven architecture) передбачає наявність циклу подій**,** який постійно «слухає» нові запити та делегує роботу з повільними операціями (диск, мережа, база даних) у пул робітників (worker pool), не блокуючи основний потік. Завдяки такому підходу Node.js особливо ефективний у системах реального часу, де важливо швидко обробляти велику кількість одночасних підключень: чати, онлайн-ігри, біржі, системи бронювання тощо.

До ключових переваг Node.js належать:

- висока продуктивність для I/O-орієнтованих задач завдяки неблокуючому введенню/виведенню;
- можливість використовувати одну мову (JavaScript) як на клієнті, так і на сервері, що полегшує командну розробку та обмін знаннями;

- велика екосистема модулів NPM, яка включає тисячі готових пакетів (Express, cors, jsonwebtoken, bcrypt, multer та ін.);
- гнучкість при побудові REST API для односторінкових застосунків, мобільних клієнтів, інтеграцій.

У межах розроблюваної вебплатформи для запису на персональні тренування Node.js дозволяє ефективно обробляти запити користувачів до API (перегляд тренерів, створення заявок, отримання списку вправ чи тренувань), забезпечує швидку взаємодію з базою даних MySQL і потенційно підтримує розширення в напрямі реального часу (наприклад, показ вільних слотів у майже моментальному режимі).

Laravel – один із найпопулярніших PHP-фреймворків, який позиціонується як «framework for web artisans» і побудований на базі архітектурного підходу Model–View–Controller (MVC)**.**

Laravel надає розробникам розвинену екосистему, що містить:

- Eloquent ORM – об’ктно-реляційний відображувач, який дає змогу працювати з таблицями бази даних як з об’ктами, використовуючи виразний PHP-синтаксис замість сирого SQL;
- Blade – шаблонізатор для створення представлень (View) з підтримкою наслідування шаблонів, розділів, компонентів;
- зручну систему маршрутизації**,** що дозволяє швидко описувати HTTP-маршрути, у тому числі для побудови REST API;
- механізми безпеки**:** захист від CSRF-атак, XSS, SQL-ін’єкцій, вбудовану систему аутентифікації та авторизації;
- модулі для кешування, черг , повідомлень, тестування**,** що полегшують розробку великих корпоративних застосунків.

Laravel традиційно демонструє високу ефективність у розробці класичних вебзастосунків, де основний акцент робиться на формуванні HTML-сторінок на сервері, роботі з формами, звітуванні, CRUD-операціях, складній бізнес-логіці інтранет-систем. Завдяки суворішій структурі, він добре підходить для команд, яким важливі чіткі шаблони організації коду.

Аналіз сучасних оглядів та порівняльних статей показує, що Node.js і Laravel мають різні «зони комфорту», де вони проявляють свої сильні сторони. Node.js часто рекомендують для високонавантажених**,** реального часу та data-intensive застосунків**,** тоді як Laravel – для традиційних вебдодатків з акцентом на бізнес-логіку, безпеці та швидкій розробці**.**

Для задачі вебплатформи запису на персональні тренування важливими є:

- швидка реакція на запити (перегляд тренерів, перевірка вільних слотів);
- потенційна можливість розширення в бік real-time-оновлень;
- зручний REST API для перспективної інтеграції з мобільним застосунком.

Враховуючи ці аспекти, доцільність Node.js як бекенд-середовища зростає, оскільки неблокуюча архітектура добре масштабуються під численні одночасні звернення.

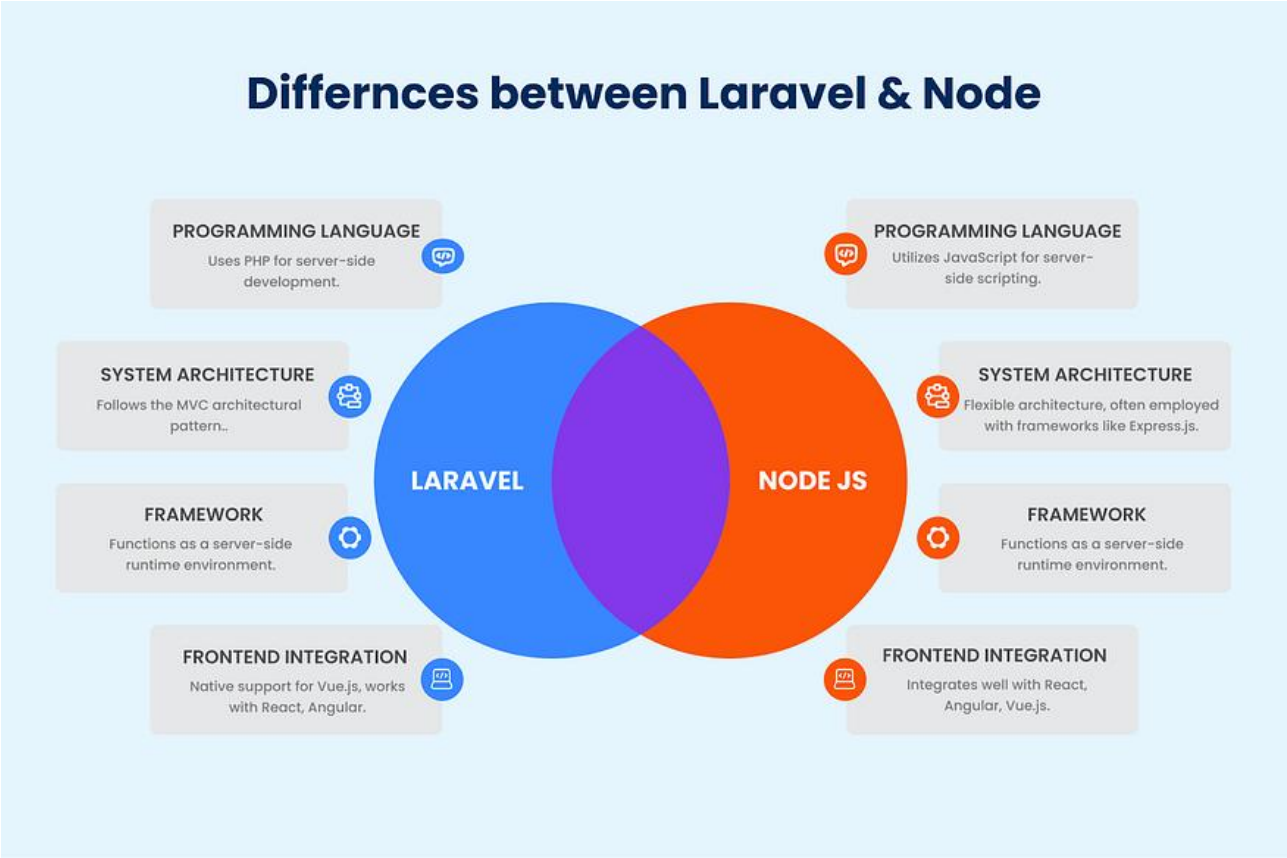


**Рис. 2.1** Схематичне порівняння архітектур Node.js та Laravel

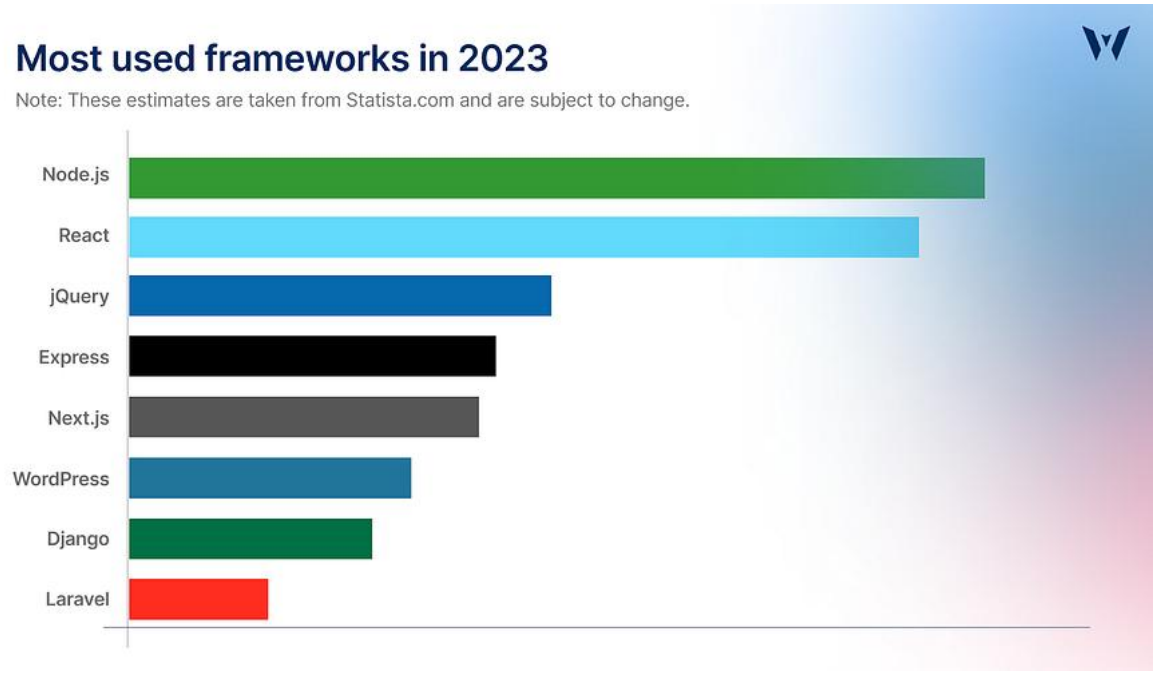


**Рис 2.2** Найбільш використовувані фреймворки у 2023 році за даними Statista

Для обґрунтування вибору технології доцільно зіставити Node.js та Laravel за кількома критеріями:

- продуктивність;
- масштабованість;
- простота розробки;
- екосистема;
- підтримка REST API;
- реальний час;
- безпека.

Таблиця 2.1

**Порівняльна характеристика Node.js та Laravel**

| Критерій | Node.js | Laravel |
|---|---|---|
| Мова програмування | JavaScript (одна мова для frontend + backend) | PHP |
| Архітектурна модель | Event-driven, non-blocking I/O, single-threaded event loop | MVC, синхронна обробка запитів |
| Продуктивність | Висока для I/O-завантажених і real-time задач | Висока для класичних вебдодатків, залежить від налаштування сервера |

| | | |
|---|---|---|
| Масштабованість | Легка горизонтальна масштабованість, багато одночасних з'днань | Добре масштабується, але часто орієнтована на традиційні вебсценарії |
| REST API | Природна інтеграція, мінімалістичні фреймворки (Express.js тощо) | Має вбудовані засоби для створення REST API (роутинг, контролери, ORM) |
| Реальний час (WebSocket) | Сильна підтримка, популярні бібліотеки (Socket.io тощо) | Можливо через додаткові сервіси, менш природно |
| Безпека | Залежить від обраних модулів, потребує уважної конфігурації | Має вбудовані механізми захисту, шаблони для аутентифікації |
| Екосистема | NPM – одна з найбільших екосистем модулів | Екосистема пакетів Composer, Laravel-сервіси та сторонні пакети |
| Крива навчання | Легка для тих, хто вже знає JS | Зручна для PHP-розробників, чітко структурована |

Сучасні публікації підкреслюють, що Node.js частіше обирають у проєктах, де необхідні швидка реакція на події, велика кількість одночасних підключень, стрімінг або чат, тоді як Laravel зручніший для структурованих інформаційних систем із багатим CRUD-функціоналом та високими вимогами до безпеки**.**

З урахуванням специфіки теми магістерської роботи – розробки вебплатформи для запису на персональні тренування – Node.js було обрано як основну серверну технологію з таких причин:

1. **Єдина мова розробки (JavaScript)** на клієнті та сервері спрощує супровід та розширення системи.
2. **Подієва, неблокуюча архітектура** дозволяє ефективно обробляти запити до API, що важливо для потенційного масштабування платформи.
3. **Природна підтримка REST API** та JSON-формату, що полегшує інтеграцію з майбутнім мобільним застосунком або сторонніми сервісами.
4. Можливість у перспективі додати **функціонал реального часу** (онлайн-статус тренера, миттєві оновлення вільних слотів, чат з тренером).
5. Уже реалізована в роботі структура бекенду (Express.js, MySQL) органічно вписується в архітектуру Node.js і демонструє реальну застосованість цієї технології.

Laravel, водночас, виступає **альтернативним рішенням**, що могло б бути використане в іншому варіанті проєкту, і тому його аналіз у даному підрозділі є важливою частиною обґрунтування вибору.

### 2.2. Логічна модель даних

Проєктування бази даних є фундаментальним етапом створення будь-якої інформаційної системи. Саме від якості логічної моделі залежить коректність зберігання даних, відсутність надлишковості, можливість виконання складних запитів і подальше масштабування платформи. Для вебплатформи запису на персональні тренування логічна модель даних повинна відображати структуру предметної області: користувачі, тренери, тренування, вправи, прогрес, заявки на заняття.

Логічне моделювання даних базується на підходах реляційної моделі **та** ER (Entity–Relationship) моделі**,** що передбачають виділення сутностей, їхніх атрибутів та зв’зків між ними. ER-діаграма дозволяє наочно представити структуру даних до етапу фізичного проєктування та реалізації в конкретній СУБД (у нашому випадку – MySQL).

У сучасних інструментах, таких як MySQL Workbench, підтримуються розширені EER-діаграми (Enhanced ER), які можуть включати додаткові концепції (спеціалізація, узагальнення тощо), але для даного проєкту достатньо класичної ER-моделі з правильно визначеними типами зв’зків: «один-до-багатьох» (1:N) та «багато-до-багатьох» (N:M) через проміжні таблиці.

Предметна область вебплатформи охоплює:

- користувачів (клієнти та тренери);
- тренування (окремі сесії, прив’зані до користувача);
- вправи (довідник базових рухів);
- структуру тренування;
- заявки на персональні тренування.

На основі аналізу цієї предметної області було виділено такі основні сутності реляційної бази даних:

1. **users** – користувачі системи.
2. **workouts** – тренувальні сесії.
3. **exercises** – довідник вправ.
4. **workout_exercises** – зв’зувальна таблиця між тренуваннями та вправами.
5. **progress** – записи про фізичний прогрес користувача.
6. **bookings** – заявки на персональні тренування.

Для коректного відображення структури даних встановлено такі ключові зв’зки:

- **users – workouts**: один користувач (клієнт) може мати багато тренувань. Це залежність 1:N, де поле workouts.user_id є зовнішнім ключем до users.id;
- **users – progress**: один користувач може мати багато записів прогресу;
- **workouts – workout_exercises**: одне тренування може включати багато записів у таблиці workout_exercises, кожен з яких описує конкретну вправу в цьому тренуванні;
- **exercises – workout_exercises**: одна вправа може використовуватись у багатьох тренуваннях. Через таблицю workout_exercises реалізується зв’зок N:M між workouts і exercises;
- **users – bookings**: один тренер може мати багато заявок;
- **bookings** додатково містить дані клієнта (ПІБ, контакти), що дозволяє обробляти заявки навіть від неавторизованих користувачів.

ER-схема бази даних повністю відповідає розробленій структурі інформаційної системи. Зв’зки між таблицями Users, Bookings, Workouts, Exercises, Workout_Exercises та Progress реалізовані відповідно до фактичної логіки проєкту. Сутність Bookings містить дані клієнта без прямої прив’зки до Users, що є коректним рішенням у межах обраної архітектури. Таким чином,

представлену ER-схему можна вважати завершеною та такою, що точно відображає функціональне наповнення створеної вебплатформи

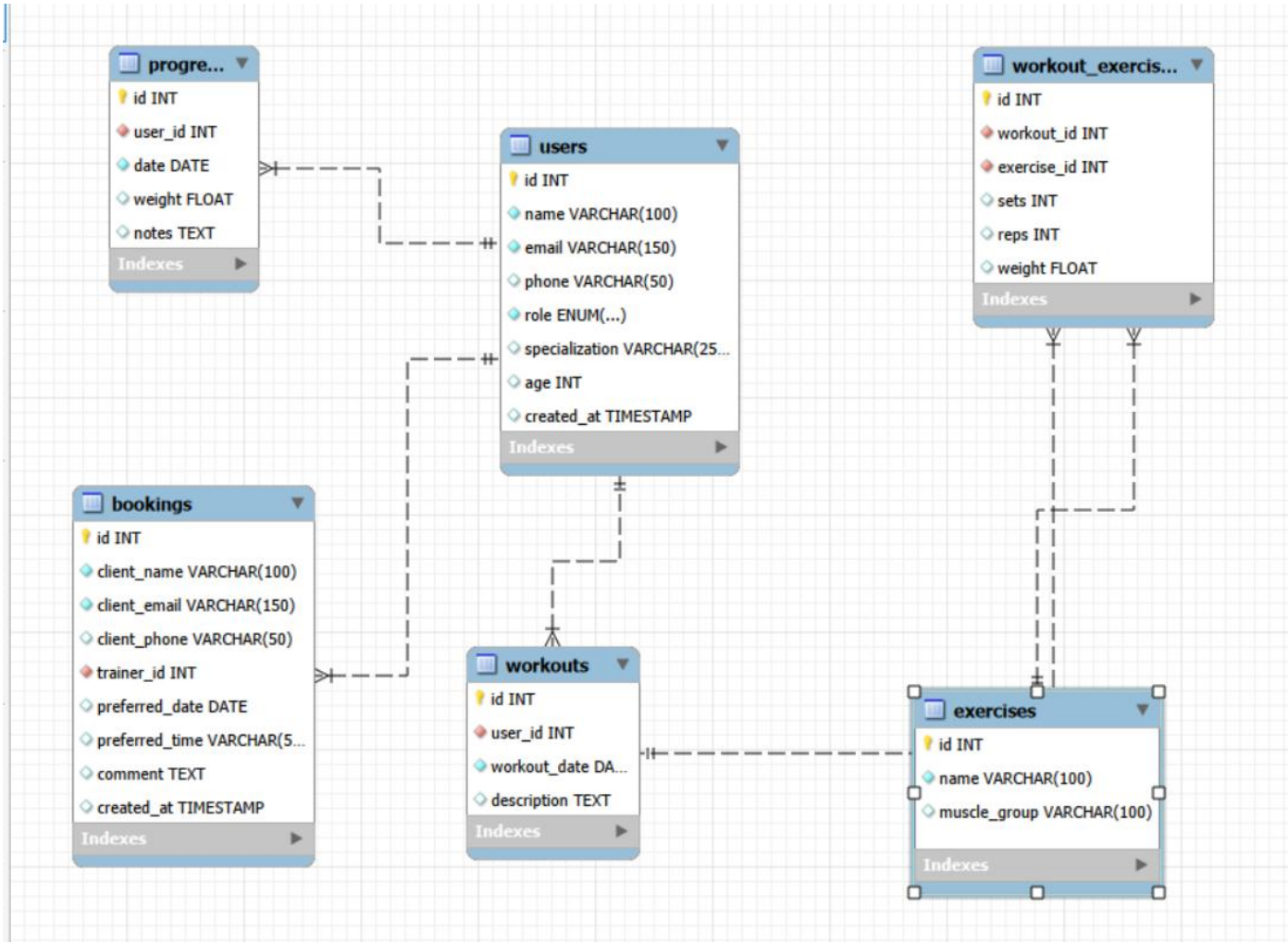


**Рис. 2.3** Логічна модель бази даних

Таблиця users зберігає інформацію про всіх користувачів платформи, як клієнтів, так і тренерів. Основні поля:

- id – первинний ключ, унікальний ідентифікатор користувача;
- name – ПІБ користувача;
- email – унікальна електронна адреса;
- age – вік;
- role – роль користувача в системі;
- specialization – спеціалізація тренера;
- phone – контактний номер;
- created_at – дата й час реєстрації.

Таблиця 2.2

**Основні поля талиці users**

| **Поле** | **Тип** | **Опис** |
|---|---|---|
| id | INT (PK, AI) | Унікальний ідентифікатор користувача |
| name | VARCHAR(100) | Ім’ та прізвище |
| email | VARCHAR(150) | Електронна адреса (унікальна) |
| age | INT | Вік |
| role | VARCHAR(50) | Роль (client/trainer/admin) |
| specialization | VARCHAR(255) | Спеціалізація тренера |
| phone | VARCHAR(50) | Телефон |
| created_at | TIMESTAMP | Дата створення запису |

Таблиця bookings реалізує зберігання заявок клієнтів на персональні тренування. Вона містить як дані клієнта, так і посилання на тренера, до якого подається заявка.

Основні поля:

- id – первинний ключ;
- client_name – ім’ клієнта;
- client_email – email клієнта;
- client_phone – телефон;
- trainer_id – зовнішній ключ на users.id;
- preferred_date – бажана дата тренування;
- preferred_time – бажаний час;
- comment – коментар клієнта;
- created_at – дата створення заявки.

Така структура дає змогу зберігати історію заявок навіть для тих користувачів, які ще не зареєстровані у системі, а також реалізувати валідацію вільних часових слотів для конкретного тренера.

- workouts – зберігає окремі тренувальні сесії, прив’зані до користувача через user_id.
- exercises – довідник вправ із зазначенням назви та цільової м’зової групи.

- workout_exercises – описує структуру конкретного тренування, які вправи, скільки підходів, повторень, робоча вага.
- progress – фіксує зміни у фізичному стані користувача (наприклад, маса тіла, нотатки щодо самопочуття).

Таблиця 2.3

**Приклад структури таблиці** workouts

| **Поле** | **Тип даних** | **Обмеження / ключі** | **Призначення** |
|---|---|---|---|
| id | INT | PRIMARY KEY, AUTO_INCREMENT | Унікальний ідентифікатор тренування |
| user_id | INT | FOREIGN KEY → users(id) | Посилання на клієнта / тренера |
| workout_date | DATE | NOT NULL | Дата проведення тренування |

Таблиця 2.4

**Приклад структури таблиці progress**

| **Поле** | **Тип даних** | **Обмеження / ключі** | **Призначення** |
|---|---|---|---|
| id | INT | PRIMARY KEY, AUTO_INCREMENT | Унікальний ідентифікатор запису прогресу |
| user_id | INT | FOREIGN KEY → users(id) | Посилання на користувача |
| date | DATE | NOT NULL | Дата фіксації прогресу |
| weight | FLOAT | NULL | Поточна вага користувача, кг |
| notes | TEXT | NULL | Коментар або опис самопочуття / навантаження |

При побудові логічної моделі бази даних було дотримано основні принципи нормалізації, що дозволяє уникнути дублювання даних і аномалій при вставці, оновленні чи видаленні записів.

Наприклад:

- окремий довідник exercises запобігає дублюванню опису вправ у кожному тренуванні;
- таблиця workout_exercises реалізує відношення N:M між тренуваннями та вправами;
- таблиця progress винесена окремо від workouts, що дозволяє реєструвати фізичні показники незалежно від конкретної сесії.

Цілісність даних забезпечується за допомогою:

- первинних ключів (PRIMARY KEY) для кожної таблиці;
- зовнішніх ключів (FOREIGN KEY) для зв’зків між таблицями;
- обмеження уникальності для поля email у таблиці users;
- перевірок у прикладному коді під час створення заявок, тренувань, записів прогресу.

Сформована логічна модель даних повністю відповідає функціональним вимогам, визначеним у роботі:

- підтримка реєстрації клієнта й тренера, зберігання їхніх профілів;
- можливість створення та зберігання тренувальних сесій з описом структури тренування;
- формування історії прогресу клієнта;
- запис на персональні тренування до конкретного тренера;
- потенційна підтримка перевірки вільних слотів через фільтрацію заявок і тренувань у базі.

Таким чином, логічна модель бази даних виступає зв’зуючим елементом між теоретичними аспектами предметної області (персональні тренування як послуга) та практичною реалізацією вебплатформи на основі Node.js та MySQL.

## Висновки по розділу 2

У другому розділі магістерської роботи було здійснено комплексне проектування вебплатформи для запису на персональні тренування, що забезпечило формування чіткої архітектури системи, вибір технологічного стеку та побудову логічної моделі даних.

Проведене порівняння фреймворків **Node.js** та **Laravel** показало, що обидві технології є потужними інструментами для створення вебзастосунків, проте мають різні архітектурні підходи та сфери оптимального використання. Laravel є зрілим серверним PHP-фреймворком з високим рівнем структурованості, однак Node.js забезпечує кращу продуктивність у задачах, що потребують обробки великої кількості паралельних запитів, а також дозволяє реалізувати застосунок із використанням однієї мови програмування — JavaScript — на клієнті та сервері. Це спрощує підтримку, пришвидшує розробку та забезпечує масштабованість, що є ключовим для сучасних онлайн-платформ у сфері фітнес-послуг.

На основі аналізу функціональних вимог, особливостей предметної області та очікуваних навантажень було обґрунтовано вибір саме **Node.js** як серверної платформи для створення вебсервісу. Такий вибір дозволив використовувати неблокуючу модель обробки подій, забезпечити швидку взаємодію з клієнтською частиною та реалізувати гнучку, розширювану архітектуру API.

У межах розділу створено логічну модель даних**,** яка визначає структуру ключових сутностей вебплатформи: користувачів, тренувань, вправ, прогресу, тренерів та записів на тренування. Модель забезпечує цілісність даних, підтримує зв’зки між об’ктами та дозволяє ефективно виконувати пошук, фільтрацію й обробку інформації. Побудована модель слугує основою для формування фізичної структури бази даних у MySQL та безпосередньої реалізації CRUD-операцій на стороні серверної логіки.

Виконаний аналіз і проєктні рішення дозволили чітко окреслити структуру майбутньої системи, визначити оптимальний інструментарій розробки та забезпечити підготовку до програмної реалізації, яка детально розглядається у третьому розділі.

# РОЗДІЛ 3. РОЗРОБКА ВЕБПЛАТФОРМИ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ NODE.JS

## 3.1. Архітектура та функціональні можливості Backend веб-платформи

Для реалізації вебплатформи персональних тренувань було обрано сучасний та надійний технологічний стек, який забезпечує ефективну взаємодію між серверною та клієнтською частинами системи, а також стабільну роботу з базою даних. Вибір конкретних технологій обумовлений вимогами до продуктивності, масштабованості, зручності розробки та підтримки програмного продукту.

Використання повноцінного Full-Stack підходу дозволило реалізувати логічно структуровану архітектуру вебзастосунку, у якій кожен компонент відповідає за визначену функціональну область. У подальших підпунктах наведено опис основних технологій, що були використані під час розробки серверної та клієнтської частин платформи, а також інструментів, застосованих у процесі створення та налагодження системи.

Було створено структуру проєкту у середовищі Visual Studio Code, яка включає такі основні елементи:

1. **index.js** — головний серверний файл на Node.js, у якому:
   - створюється HTTP-сервер на базі Express;
   - налаштовуються маршрути для роботи з тренерами, заявками та іншими сутностями;
   - здійснюється підключення до бази даних MySQL.
2. **public** — директорія статичних файлів:
   - index.html — головна сторінка платформи з розділами «Вправи», «Тренери», «Тренування» та «Персональний тренер»;
   - admin.html — адміністративна панель для перегляду заявок;

- main.js — скрипт, що відповідає за завантаження даних із сервера (список тренерів, тренувань тощо) та їх відображення на сторінці;
- script.js — логіка роботи форми запису на тренування (збір даних, відправка POST-запитів, обробка відповідей сервера);
- style.css — стилізація інтерфейсу, фонові зображення, оформлення кнопок та блоків;

3. **node_modules** — службова директорія з установленими модулями Node.js;
4. **package.json** — конфігураційний файл проєкту, де описано залежності та основні скрипти запуску.

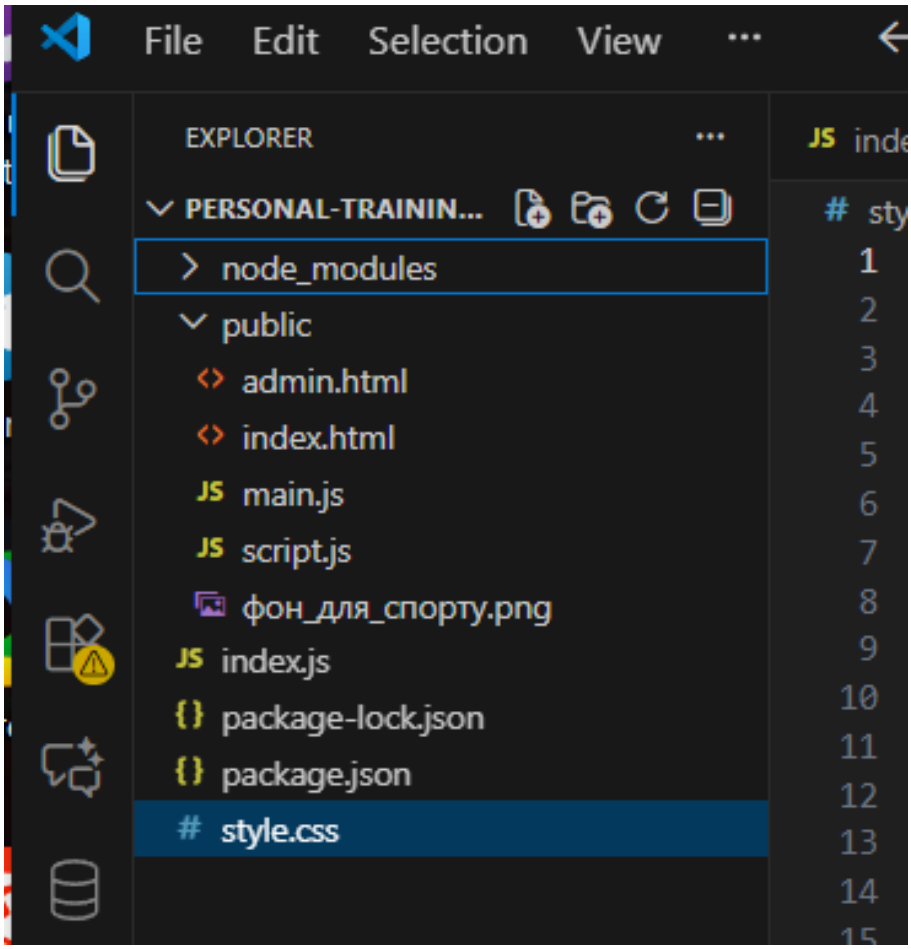


**Рис. 3.1** Структура проєкту вебплатформи у середовищі Visual Studio Code.

На рисунку 3.1 демонструється дерево файлів проєкту: видно головний файл index.js, папку public з HTML, CSS та JavaScript-файлами, а також

конфігураційний файл package.json, що підтверджує організацію застосунку за принципом розділення на серверну та клієнтську частини.

Серверна частина платформи написана у файлі index.js та запускається з терміналу середовища Visual Studio Code командою - **node index.js**

**Node.js** — серверне середовище виконання JavaScript, яке забезпечує асинхронну обробку запитів та ефективну роботу з мережею. Використовується для запуску серверної логіки вебплатформи.

**Express.js** — вебфреймворк для Node.js, який спрощує створення HTTP-сервера та реалізацію REST-API. Забезпечує маршрутизацію запитів, обробку даних та формування відповідей клієнту.

**MySQL** — реляційна система керування базами даних, призначена для зберігання та обробки структурованої інформації про користувачів, тренування, вправи та заявки.

Після успішного запуску в консолі відображається повідомлення на кшталт:

«Сервер запущено на порту 3000

Підключено до MySQL!»

Це свідчить про коректне підключення до бази даних MySQL та готовність серверної частини до обробки HTTP-запитів від клієнтів.

Інструменти розробки та адміністрування;

**Visual Studio Code** – інтегроване середовище розробки, яке використовується для створення, редагування та налагодження програмного коду проєкту.

**MySQL Workbench** – графічний інструмент для адміністрування бази даних, створення таблиць, виконання SQL-запитів і побудови ER-діаграм.

**npm (Node Package Manager)** – менеджер пакетів для керування залежностями проєкту та встановлення необхідних бібліотек.

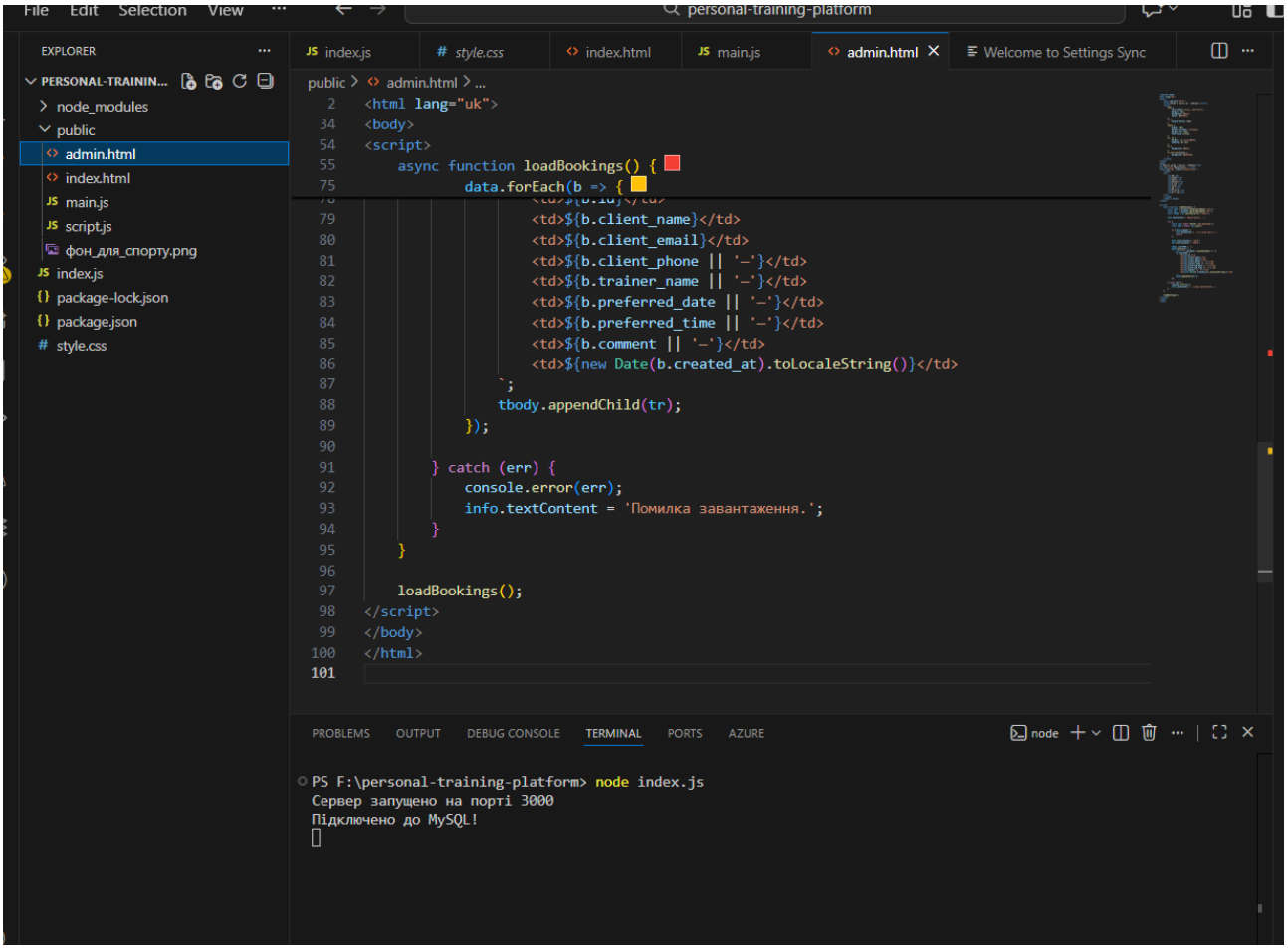


**Рис. 3.2** Успішний запуск сервера Node.js та підключення до MySQL у терміналі VS Code.

На рисунку 3.2 показано консольний вивід середовища розробки, де видно, що сервер прослуховує порт 3000, а також встановлено з’днання з базою даних personal_training.

Фрагмент серверного коду відповідає за отримання списку тренерів:

```
app.get(«/api/trainers», (req, res) => {
  const sql = `
    SELECT id, name, email, phone, specialization
    FROM users
    WHERE role = «trainer»
    ORDER BY name;
  `;
  db.query(sql, (err, rows) => {
    if (err) {
```

```
            console.error(«Помилка GET /api/trainers:», err);
            return res.status(500).json({ error: «Помилка сервера» });
        }
        res.json(rows);
    });
});
```

Даний маршрут:

- формує SQL-запит до таблиці users з фільтром role = «trainer»;
- повертає на Frontend список доступних тренерів у форматі JSON;
- у разі помилки виводить інформацію у консоль і повертає клієнту повідомлення про помилку сервера.

Отримані з цього ендпоінта дані використовуються на сторінці «Персональний тренер» для побудови карток тренерів та у випадаючому списку форми запису (див. рис. 3.9).

Формування нової заявки користувача на персональне тренування обробляється таким фрагментом коду:

```
app.post(«/api/bookings», (req, res) => {
    const {
        client_name,
        client_email,
        client_phone,
        trainer_id,
        preferred_date,
        preferred_time,
        comment
    } = req.body;
    const sql = `
        INSERT INTO bookings
        (client_name, client_email, client_phone, trainer_id, preferred_date, preferred_time, comment)
```

```
      VALUES (?, ?, ?, ?, ?, ?, ?)
    `;
    db.query(sql, [
      client_name,
      client_email,
      client_phone,
      trainer_id,
      preferred_date,
      preferred_time,
      comment
    ], (err) => {
      if (err) {
        console.error(«Помилка POST /api/bookings:», err);
        return res.status(500).json({ error: «Помилка сервера» });
      }
      res.json({ message: «Заявку успішно відправлено!» });
    });
});
```

Алгоритм роботи ендпоінта:

1. Отримання даних із тіла запиту (req.body), які були надіслані з форми на сторінці.
2. Формування параметризованого SQL-запиту INSERT до таблиці bookings.
3. Запис інформації про клієнта, тренера, бажану дату й час тренування та коментар.
4. У разі успіху – відправка JSON-відповіді { message: «Заявку успішно відправлено!» }, яку потім використовує клієнтська частина для виведення повідомлення користувачу.

Запис, що створюється через цей ендпоінт, надалі відображається як у адміністративній панелі (рис. 3.7), так і безпосередньо у MySQL Workbench (рис. 3.6, 3.8).

Додатково в Backend реалізовано перевірку, чи вільний тренер на обрану дату й час. У базі даних створено унікальне обмеження на пару полів trainer_id, preferred_date, preferred_time. Це запобігає дублюванню бронювань для одного й того ж тренера в один і той самий часовий слот.

У разі спроби записатися на вже зайнятий слот сервер повертає помилку, яку клієнтська частина обробляє та виводить користувачу відповідне повідомлення (див. рис. 3.5).

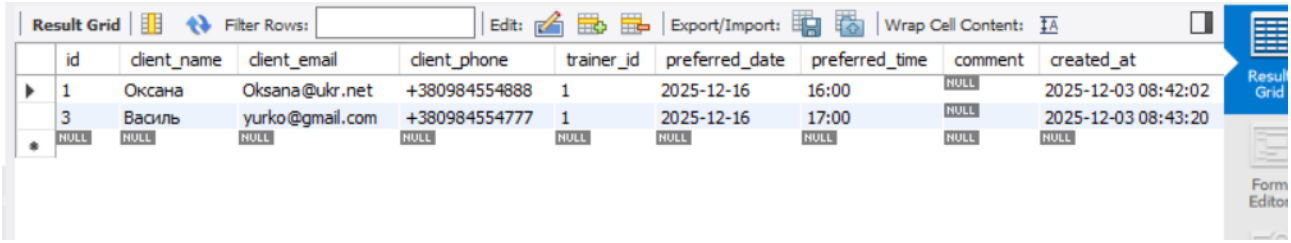


**Рис. 3.3** Таблиця bookings у MySQL Workbench

## 3.2. Frontend реалізація

Frontend-частина вебплатформи відповідає за формування інтерфейсу користувача, відображення даних, отриманих від сервера, а також забезпечення інтерактивної взаємодії з системою. Інтерфейс реалізовано на основі стандартних вебтехнологій — HTML5, CSS3 та JavaScript, що гарантує високу швидкодію, кросплатформеність та адаптивність.

Клієнтська частина (Frontend)

**HTML5** — мова розмітки, що використовується для формування структури вебсторінок платформи та організації контенту.

**CSS3** — мова стилів, яка відповідає за зовнішній вигляд інтерфейсу користувача, включаючи адаптивний дизайн, кольорову схему та оформлення елементів.

**JavaScript (ES6)** — мова програмування, що забезпечує динамічну взаємодію користувача з вебплатформою, обробку подій, валідацію форм та оновлення сторінок без перезавантаження.

**Fetch API** — засіб асинхронної взаємодії клієнтської частини з сервером, який дозволяє отримувати та надсилати дані у форматі JSON.

До складу фронтенду входять такі основні файли:

- index.html — головна сторінка.
- admin.html — адміністративна панель.
- style.css — стилізація.
- main.js — завантаження даних.
- script.js — відправлення заявок.

На рисунку 3.3 представлено вікно MySQL Workbench з вибраною таблицею bookings, де видно два успішно додані записи. У нижній частині вікна відображається історія виконаних команд, зокрема створення унікального обмеження unique_trainer_slot, яке забезпечує відсутність дублювання заявок на один і той самий часовий інтервал.

На головній сторінці користувач бачить:

- назву вебплатформи «Платформа персональних тренувань» та слоган «МОТИВАЦІЯ – ЦЕ СИЛА»;
- навігаційне меню у вигляді кнопок-перемикачів: «Вправи», «Тренери», «Тренування», «Персональний тренер»;
- список вправ із коротким описом (назва, цільова група м’зів, текстова рекомендація щодо виконання);
- фонове зображення тренажерної зали, що створює візуальний контекст фітнес-тематики.

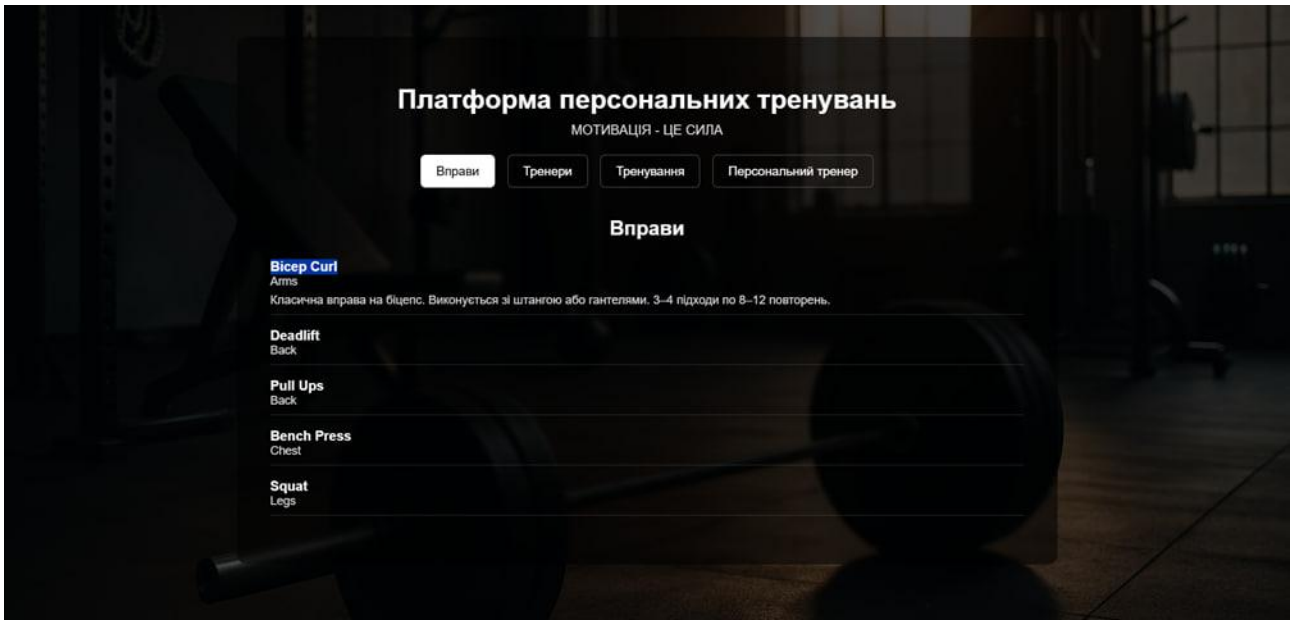


**Рис. 3.4** Інтерфейс головної сторінки вебплатформи з переліком вправ.

На рисунку 3.4 відображено варіант головної сторінки, де активним є розділ «Вправи». Зліва у списку показано приклади вправ (наприклад, *Bicep Curl*, *Deadlift*, *Squat*), а при наведенні чи виборі вправи демонструється розгорнутий опис. Це надає користувачу базову інформацію про типові елементи тренувань.

Форма запису на персональне тренування розміщена у розділі «Персональний тренер». Вона містить такі поля:

- **«Ім’я»** – ПІБ клієнта;
- **«Email»** – контактна електронна адреса;
- **«Телефон»** – номер телефону у міжнародному форматі;
- **«Обрати тренера»** – випадаючий список, сформований на основі даних з ендпоінта /api/trainers;
- **«Бажана дата»** – поле вибору дати тренування;
- **«Бажаний час»** – поле вибору години проведення;
- **«Коментар»** – необов’зкове поле для уточнення побажань клієнта (наприклад, «тренування лише в середу та по суботах»).

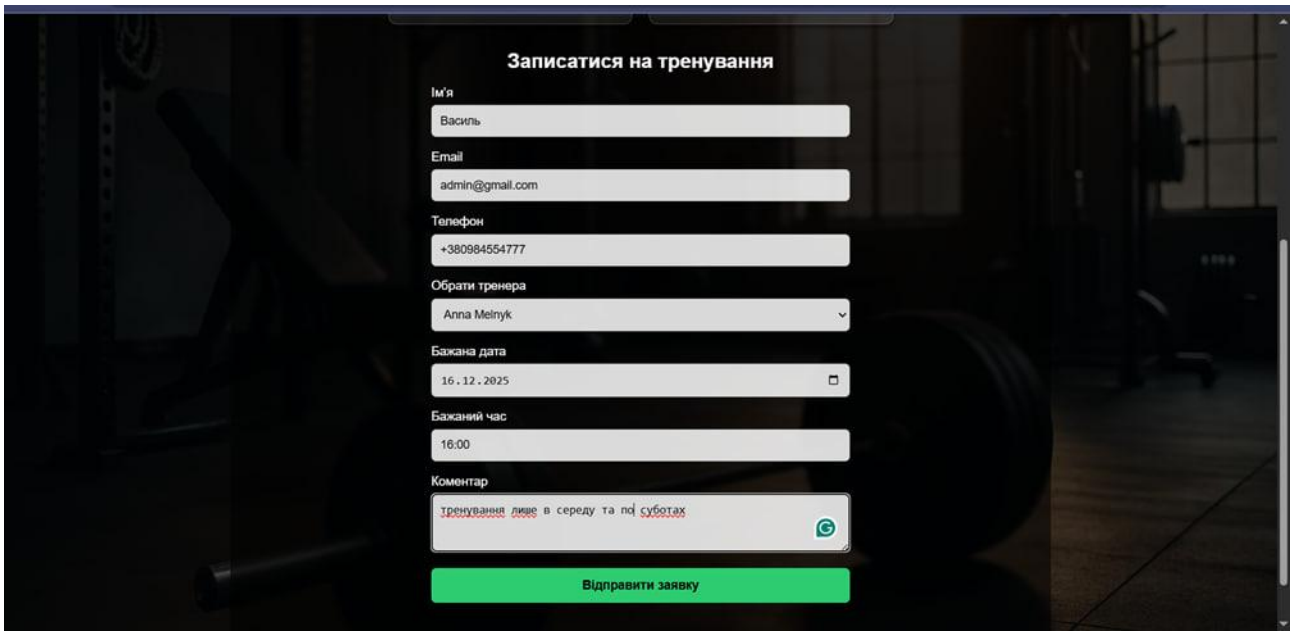


**Рис. 3.5** Форма подання заявки на персональне тренування з вибором тренера, дати та часу.

Після натискання кнопки **«Відправити заявку»** викликається JavaScript-функція, яка збирає дані з полів, формує тіло POST-запиту та надсилає його на адресу /api/bookings. У разі успішної відповіді сервера користувач бачить візуальне підтвердження.

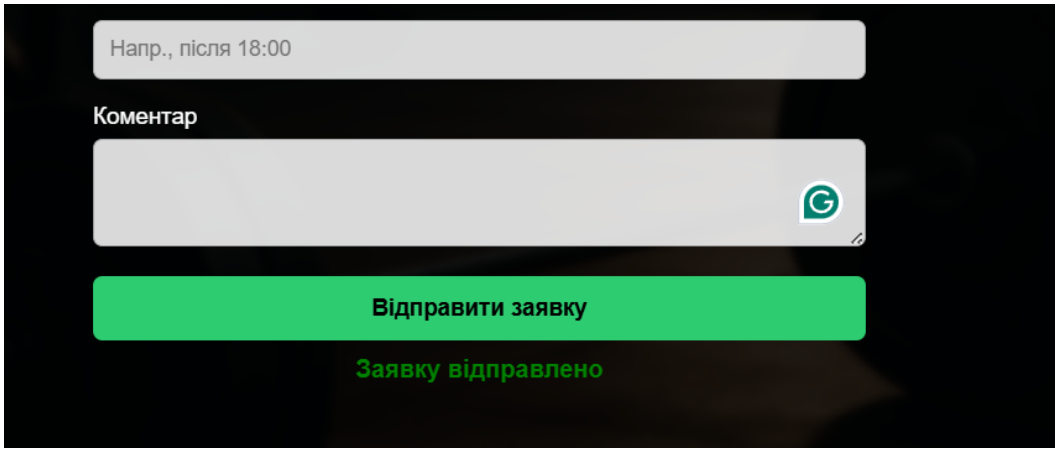


**Рис. 3.6** Повідомлення про успішне надсилання заявки на тренування.

На рисунку 3.6 у нижній частині форми відображено текст «Заявку відправлено», що підсвічується зеленим кольором. Це є результатом обробки позитивної відповіді сервера й виконує роль зворотного зв’зку для користувача, підтверджуючи, що дані було збережено.

Якщо ж тренер на обрану дату й час уже зайнятий, замість зеленого повідомлення виводиться попередження червоним текстом.

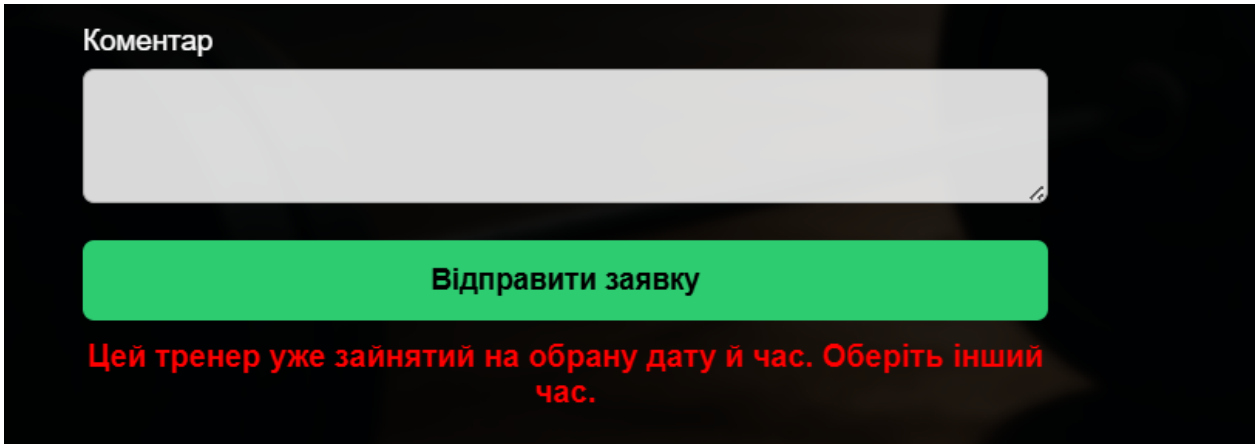


**Рис. 3.7** Повідомлення про помилку при спробі записатися на вже зайнятий часовий слот.

На рисунку 3.7 видно текст повідомлення «Цей тренер уже зайнятий на обрану дату й час. Оберіть інший час.». Це підтверджує коректність роботи бізнес-логіки щодо перевірки унікальності бронювання.

Адміністративна панель реалізована у файлі admin.html. Вона дає змогу переглядати всі наявні заявки на персональні тренування в табличному вигляді.

Отримані з сервера дані (ім’ клієнта, email, телефон, тренер, дата, час, коментар, час створення запису) відображаються на сторінці у вигляді HTML-таблиці.

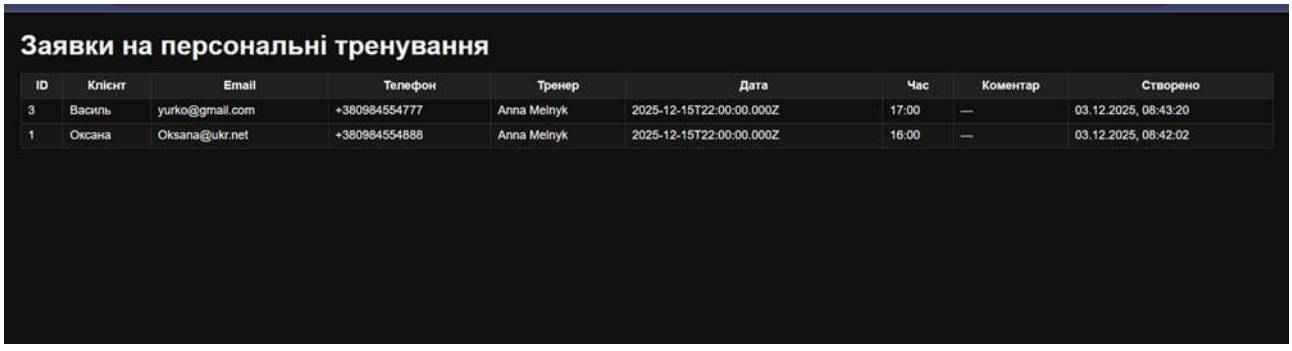


**Рис. 3.8** Адміністративна сторінка із переліком заявок на персональні тренування

На рисунку 3.8 показано, як адміністратор бачить дві заявки: від клієнтів Василя та Оксани. Для кожної заявки відображаються усі ключові параметри – контактні дані, обраний тренер, дата й час тренування, а також момент створення

заявки. Таким чином, адміністратор може оперативно аналізувати завантаження тренерів і при потребі зв’зуватися з клієнтами.

Окремий розділ «Тренування» містить список запланованих тренувальних сесій. Дані для цієї вкладки завантажуються з відповідної таблиці бази (workouts) за допомогою AJAX-запиту.

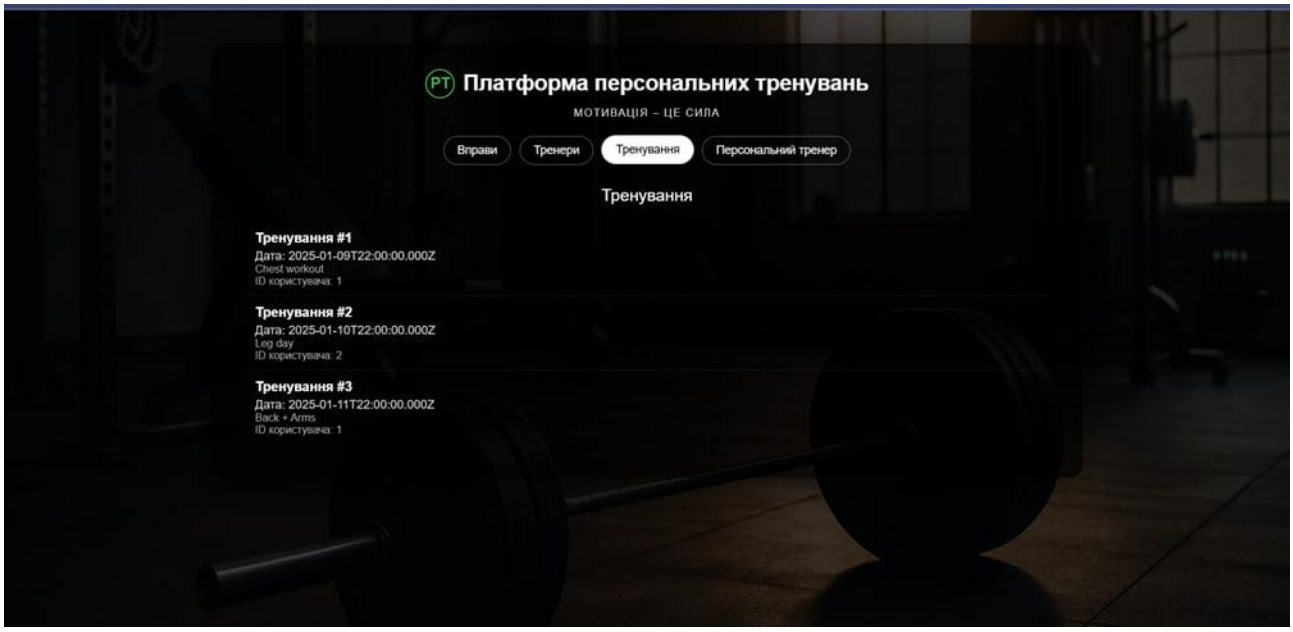


**Рис. 3.9** Вкладка «Тренування» з переліком запланованих тренувальних сесій.

На рисунку 3.9 видно, що для кожного тренування відображається номер, дата, коротка назва/опис (наприклад, *Chest workout*, *Leg day*), а також ID користувача. Такий перелік може використовуватися клієнтом як щоденник занять, а тренером – для планування навантажень.

Сторінка «Персональний тренер» поєднує два основних компоненти:

1. Список тренерів у вигляді карток з основною інформацією:
   - ПІБ;
   - спеціалізація (силові тренування, функціональний тренінг, набір м’зової маси, схуднення тощо);
   - контактний e-mail та телефон.

2. Форма запису для швидкого створення заявки без додаткових переходів.

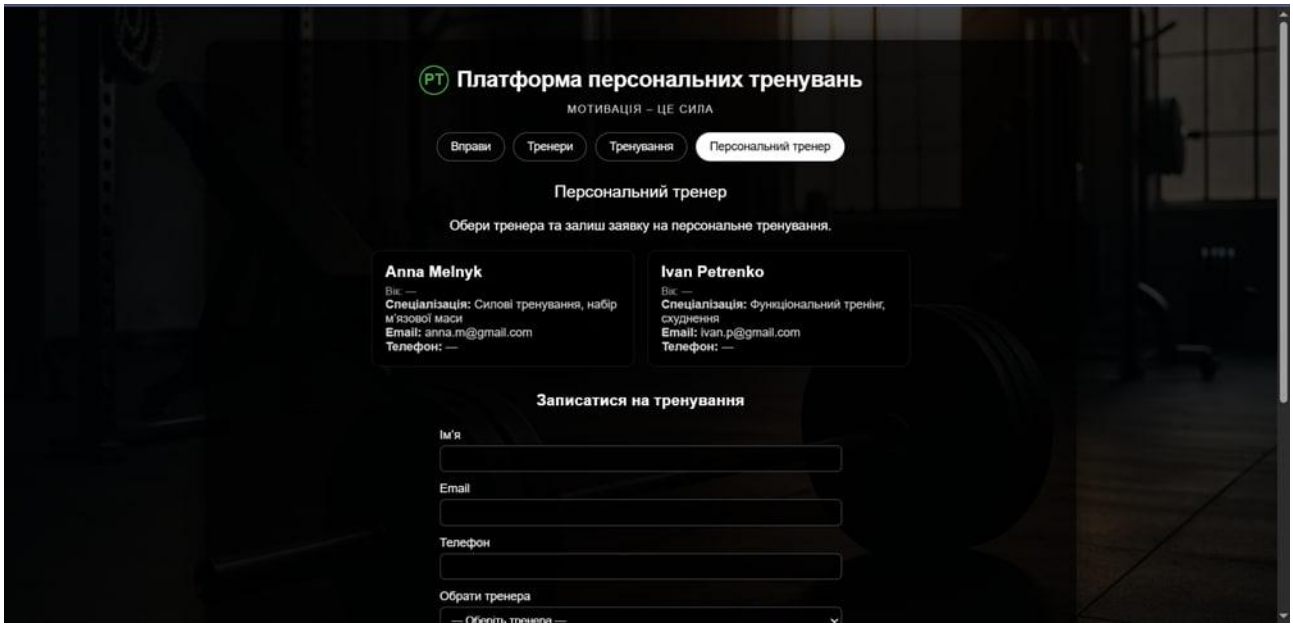


**Рис. 3.10** Вкладка «Персональний тренер» із картками тренерів та формою запису.

На рисунку 3.10 користувачу пропонується обрати одного з тренерів (наприклад, Anna Melnyk або Ivan Petrenko), ознайомитися з їх спеціалізацією й відразу нижче заповнити форму запису. Такий підхід зменшує кількість кліків і робить процес взаємодії з платформою інтуїтивно зрозумілим.

База даних **personal_training** містить шість ключових таблиць:

- users – інформація про користувачів платформи (клієнтів і тренерів);
- bookings – заявки на персональні тренування;
- exercises – перелік вправ;
- workouts – тренувальні сесії;
- workout_exercises – зв'зок між тренуваннями та окремими вправами (реалізація зв'зку «багато-до-багатьох»);
- progress – фіксація прогресу клієнтів (вага, обсяги, виконані навантаження тощо).

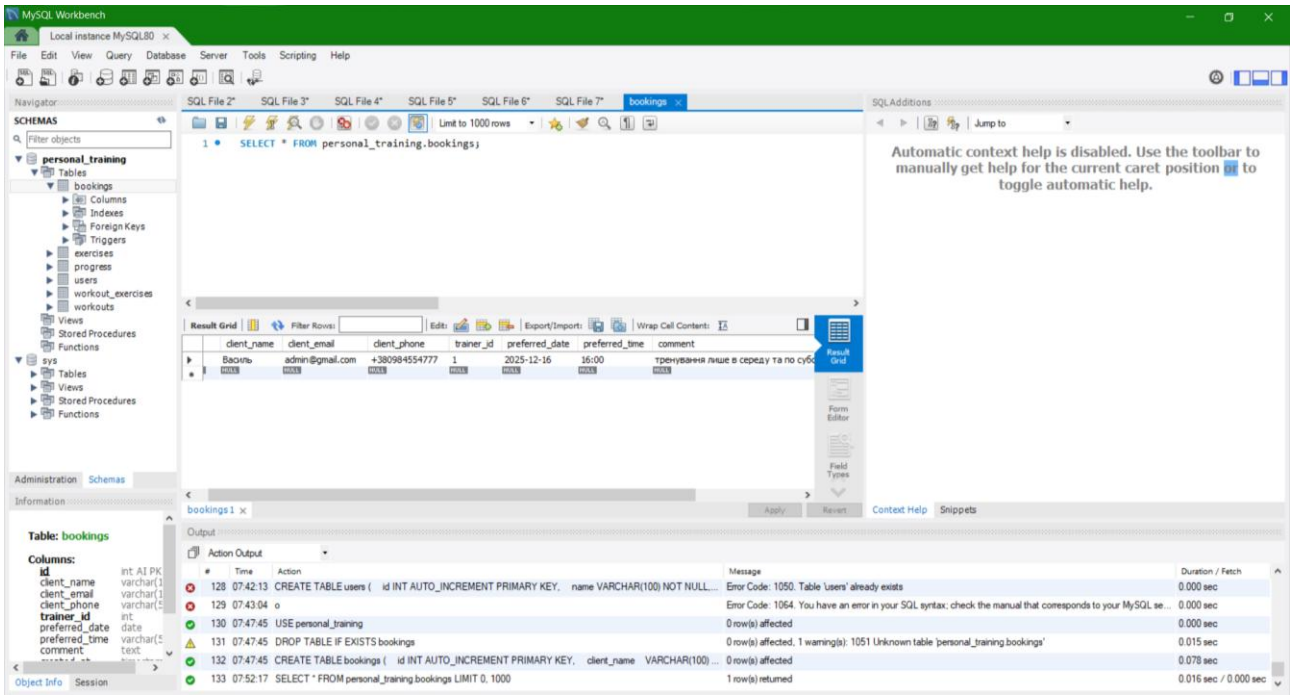


**Рис. 3.11** Структура бази даних personal_training у MySQL Workbench.

На рисунку 3.10 показано дерево об’єктів у MySQL Workbench: видно схему personal_training, у якій присутні таблиці bookings, exercises, progress, users, workout_exercises та workouts. Це підтверджує реалізацію багатотабличної структури, яка охоплює як облік заявок, так і зберігання інформації про тренування та прогрес.

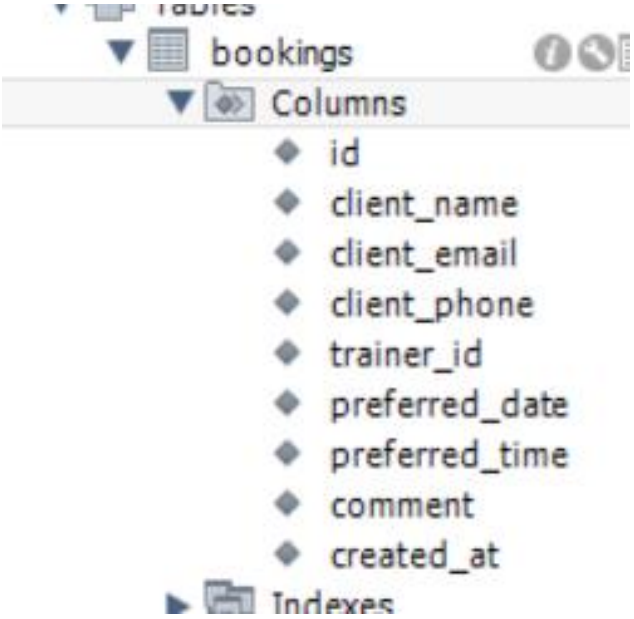


**Рис. 3.12** Вміст таблиці bookings у MySQL Workbench після створення першої заявки.

На рисунку 3.12 демонструється запис, сформований на основі прикладу з форми запису: видно значення полів client_name, client_email, client_phone, вибраного тренера, дати й часу, а також текстовий коментар. Це підтверджує коректність взаємодії Frontend – Backend – база даних.

При додаванні нових заявок таблиця bookings поповнюється відповідними записами. Одночасно застосовується описане вище унікальне обмеження на поєднання trainer_id, preferred_date, preferred_time.

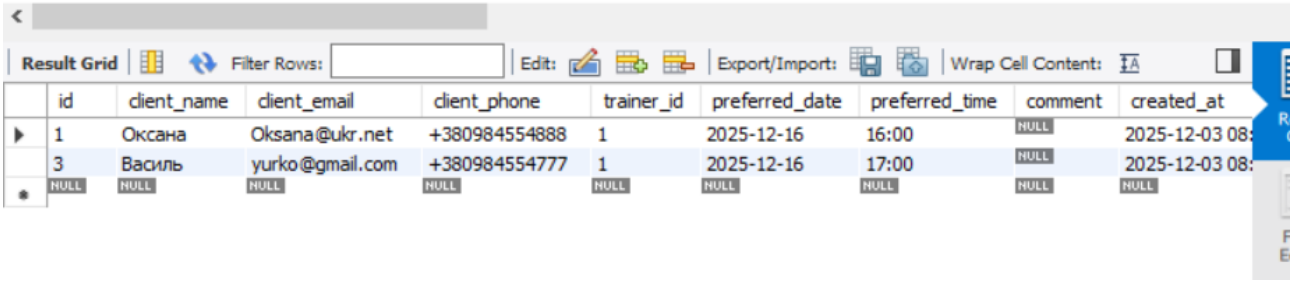


**Рис. 3.13** Вміст таблиці bookings з двома заявками та історією створення унікального обмеження.

На рисунку 3.13 видно два записані бронювання для тренера Anna Melnyk: на 16:00 та 17:00, а внизу – команду ADD CONSTRAINT unique_trainer_slot UNIQUE з повідомленням «0 duplicates», що свідчить про успішне накладання унікального ключа. Надалі спроба вставити третій запис із тим самим тренером, датою та часом призводить до помилки й відображається на інтерфейсі як повідомлення, показане на рисунку 3.6.

Аналогічно можуть бути заповнені таблиці workouts та progress, що забезпечують зберігання даних про виконані тренування та результати клієнтів.

JavaScript-файли **main.js** та **script.js** забезпечують повний цикл взаємодії між клієнтом і сервером:

- завантаження списку тренерів та тренувань;
- динамічне оновлення вмісту сторінок без перезавантаження;
- формування та відправлення заявок на тренування;
- обробку відповідей сервера (успіх / помилка) і виведення повідомлень користувачу;
- заповнення HTML-таблиць даними при відкритті адміністративної панелі.

Приклад фрагмента клієнтської логіки:

```
async function loadTrainers() {
   const response = await fetch(«/api/trainers»);
   const trainers = await response.json();

   const select = document.getElementById(«trainerSelect»);
   trainers.forEach(trainer => {
      const option = document.createElement(«option»);
      option.value = trainer.id;
      option.textContent = trainer.name;
      select.appendChild(option);
   });
}
```

Ця функція викликається під час завантаження сторінки й заповнює випадаючий список «Обрати тренера» актуальними даними з бази.

Нижче наведено фрагменти SQL-коду для створення основних таблиць бази даних **personal_training**, які забезпечують зберігання даних про користувачів, вправи, тренування, прогрес та заявки на персональні тренування. Структура реалізована на основі реляційної моделі з використанням первинних ключів (PRIMARY KEY), зовнішніх ключів (FOREIGN KEY) та обмежень цілісності.

Таблиця users є центральною сутністю системи та містить дані про користувачів платформи: клієнтів, тренерів і адміністраторів. Роль користувача визначається полем role (enum), що дозволяє розмежувати доступ і логіку відображення даних.

```
CREATE TABLE `users` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(100) NOT NULL,
  `email` varchar(150) NOT NULL,
  `role` enum('client','trainer','admin') NOT NULL DEFAULT 'client',
  PRIMARY KEY (`id`),
```

```
  UNIQUE KEY `email` (`email`)
);
```

id — первинний ключ користувача;

role — визначає тип користувача (клієнт/тренер/адмін);

UNIQUE(email) — запобігає дублюванню акаунтів за email.

Таблиця bookings використовується для реєстрації заявок клієнтів на персональні тренування. Вона містить контактні дані клієнта, обраного тренера та бажану дату/час. Для запобігання дублюванню записів реалізовано унікальне обмеження на комбінацію trainer_id + preferred_date + preferred_time.

```
CREATE TABLE `bookings` (
  `id` int NOT NULL AUTO_INCREMENT,
  `client_name` varchar(100) NOT NULL,
  `client_email` varchar(150) NOT NULL,
  `trainer_id` int NOT NULL,
  `preferred_date` date DEFAULT NULL,
  `preferred_time` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `unique_trainer_slot` (`trainer_id`,`preferred_date`,`preferred_time`),
  CONSTRAINT `bookings_ibfk_1`
    FOREIGN KEY (`trainer_id`) REFERENCES `users` (`id`)
);
```

trainer_id — зовнішній ключ, який посилається на таблицю users;

unique_trainer_slot — забезпечує бізнес-правило «тренер не може мати 2 заявки на один і той самий слот часу».

Таблиця exercises містить довідник фізичних вправ та групу м’язів, на яку орієнтована вправа. Ця таблиця використовується при формуванні тренувальних планів.

```
CREATE TABLE `exercises` (
  `id` int NOT NULL AUTO_INCREMENT,
```

```
  `name` varchar(100) NOT NULL,
  `muscle_group` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`)
);
```

Таблиця є довідником і використовується у зв’язці з тренуваннями через проміжну таблицю workout_exercises.

Таблиця workouts зберігає інформацію про тренувальні сесії конкретного користувача: дату та опис тренування. Пов’язана з таблицею users через поле user_id.

```
CREATE TABLE `workouts` (
  `id` int NOT NULL AUTO_INCREMENT,
  `user_id` int NOT NULL,
  `workout_date` date NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `workouts_ibfk_1`
    FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)
);:
```

user_id визначає, для якого користувача сформовано тренування;

зв’язок users (1) → workouts (N).

Таблиця workout_exercises реалізує зв’язок **багато-до-багатьох** між workouts та exercises і додатково зберігає параметри виконання (підходи, повторення, вага).

```
CREATE TABLE `workout_exercises` (
  `id` int NOT NULL AUTO_INCREMENT,
  `workout_id` int NOT NULL,
  `exercise_id` int NOT NULL,
  `sets` int DEFAULT NULL,
  `reps` int DEFAULT NULL,
  `weight` float DEFAULT NULL,
  PRIMARY KEY (`id`),
```

```
  CONSTRAINT `workout_exercises_ibfk_1`
    FOREIGN KEY (`workout_id`) REFERENCES `workouts` (`id`),
  CONSTRAINT `workout_exercises_ibfk_2`
    FOREIGN KEY (`exercise_id`) REFERENCES `exercises` (`id`)
);
```

Ця таблиця зберігає деталі конкретної вправи в межах конкретного тренування та дозволяє формувати тренування з довільною кількістю вправ.

Таблиця progress зберігає дані про прогрес користувача (наприклад, вага та примітки) із прив’язкою до конкретної дати. Дані використовуються для аналізу динаміки результатів.

```
CREATE TABLE `progress` (
  `id` int NOT NULL AUTO_INCREMENT,
  `user_id` int NOT NULL,
  `date` date NOT NULL,
  `weight` float DEFAULT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `progress_ibfk_1`
    FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)
);
```

user_id — посилання на користувача;

date — фіксує дату вимірювання/запису прогресу.

Отже, структура БД реалізує ключові бізнес-сутності платформи. Використання первинних та зовнішніх ключів забезпечує логічні зв’язки між таблицями, а унікальне обмеження в таблиці bookings гарантує відсутність дублювання записів на один і той самий часовий слот тренера.

## Висновки по розділу 3

У третьому розділі було виконано повну програмну реалізацію вебплатформи для запису на персональні тренування, яка об’єднує серверну частину, клієнтський інтерфейс та модуль взаємодії з базою даних. Проведена розробка охоплює всі етапи створення вебзастосунку — від побудови структури файлів та налаштування серверного оточення до формування інтерактивного інтерфейсу та впровадження механізмів обробки користувацьких запитів.

На основі технологій **Node.js**, **Express** та **MySQL** реалізовано гнучку та масштабовану серверну підсистему, яка забезпечує обробку всієї бізнес-логіки платформи. До основних функцій серверної частини належать:

- обробка HTTP-запитів, сформованих клієнтською частиною;
- робота з базою даних, включаючи виконання SQL-запитів, вставку нових записів і вибірку даних;
- валідація користувацьких даних перед записом у базу;
- запобігання дублюванню заявок шляхом перевірки зайнятості тренера у вибраний час;
- формування відповідей API для різних клієнтських сценаріїв.

Створений набір API-маршрутів повністю забезпечує необхідну взаємодію між користувачем, тренером та адміністратором. Зокрема, реалізовано маршрути для отримання списку тренерів, перегляду тренувань, створення нових бронювань і завантаження заявок у адміністративний модуль. Така структура API є достатньо простою для підтримки й водночас гнучкою для розширення у майбутніх версіях.

Клієнтська частина платформи була детально опрацьована. Було розроблено сучасний інтерфейс із використанням HTML, CSS та JavaScript, що забезпечує інтуїтивно зрозумілу взаємодію для кінцевих користувачів. Реалізовано динамічне завантаження даних з API, оновлення елементів сторінки без перезавантаження, візуальні повідомлення про результати дій та інтерактивну форму запису на тренування. Завдяки адаптивній верстці сторінки коректно відображаються на різних пристроях.

Окрему увагу було приділено **адміністративному модулю**, що дозволяє ефективно керувати заявками на тренування. У цьому розділі платформи адміністратор може переглядати всі подані заявки, що значно спрощує управління розкладом тренерів і взаємодію з клієнтами. Реалізація табличного відображення даних підтвердила правильність роботи API та надійну взаємодію з MySQL-сервером.

Під час розробки також була створена та протестована база даних **personal_training**, структура якої повністю відповідає логічній моделі, сформованій у попередньому розділі. Здійснено перевірку працездатності таблиць, коректності зв’язків між ними та відсутності конфліктів при записі даних. Система продемонструвала стабільну роботу під час додавання, вибірки та обробки інформації.

Узагальнюючи, реалізована програмна частина вебплатформи підтверджує правильність обраних підходів до проєктування архітектури та моделювання бази даних. Вебзастосунок успішно виконує всі основні функції, забезпечує автоматизацію процесу запису на тренування та створює зручний інструмент взаємодії між клієнтами, тренерами та адміністраторами. Розроблена система є масштабованою, придатною до подальшого розвитку та може слугувати основою для створення повнофункціональної комерційної платформи.

# ВИСНОВКИ

У результаті виконання роботи було розроблено повноцінну вебплатформу для запису на персональні тренування, яка охоплює весь цикл взаємодії між клієнтом та тренером — від перегляду інформаційних матеріалів до оформлення заявок та управління тренувальними процесами.

У першому розділі проведено аналіз предметної області, визначено основні проблеми традиційного процесу запису на тренування та обґрунтовано необхідність створення веборієнтованої системи. Було сформовано функціональні та нефункціональні вимоги, описано цільову аудиторію та ключові сценарії взаємодії користувачів.

У другому розділі виконано побудову логічної моделі даних, на основі якої створено ER-схему бази даних. Визначено основні сутності, їх атрибути та зв’зки, що забезпечили структуроване зберігання інформації про користувачів, тренування, вправи та прогрес. На основі цієї моделі реалізовано фізичну базу даних MySQL, яка забезпечує цілісність, узгодженість та розширюваність даних.

У третьому розділі було здійснено повну програмну реалізацію вебплатформи, що включає серверну частину на Node.js та клієнтський інтерфейс, створений засобами HTML, CSS та JavaScript. Реалізовано ключові модулі системи: перегляд вправ, вибір тренера, створення заявок на тренування, адміністративну панель, а також механізми обміну даними між Frontend та Backend частинами. Проведено тестування функціональності, перевірено коректність роботи API та збереження інформації у базі даних.

Розроблена система повністю задовольняє поставлені вимоги й демонструє стабільну роботу в усіх сценаріях використання. Платформа дозволяє автоматизувати процес запису на тренування, зменшує навантаження на персонал фітнес-центрів, підвищує зручність для користувачів та забезпечує можливість подальшого масштабування. Запропонована архітектура дає змогу

легко додавати новий функціонал — наприклад, систему онлайн-оплат, особисті кабінети тренерів та клієнтів, розширену аналітику прогресу.

Таким чином, у роботі виконано повний цикл створення програмного продукту — від аналізу предметної області й проєктування бази даних до реалізації вебзастосунку та його тестування. Отриманий результат має практичну цінність та може бути використаний як основа для реального впровадження у спортивних закладах або як база для подальших досліджень та удосконалень.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Amorserv. Benefits of using Node.js for web development [Електронний ресурс]. – Режим доступу: https://amorserv.com. – (Дата звернення: 25.11.2025).
2. Amrutiya A. Laravel vs Node.js: A comprehensive comparison with pros and cons [Електронний ресурс]. – Medium. – Режим доступу: https://medium.com. – (Дата звернення: 14.11.2025).
3. Cambridge University. MySQL: Implementing a Relational Database Design [Електронний ресурс]. – Режим доступу: https://training.cam.ac.uk. – (Дата звернення: 30.10.2025).
4. Codecordia. Laravel vs Node.js: Which backend is better for your startup? [Електронний ресурс]. – Режим доступу: https://codecordia.com. – (Дата звернення: 28.10.2025).
5. Dev.to. The ultimate guide to Laravel: A PHP framework for modern web applications [Електронний ресурс]. – Режим доступу: https://dev.to. – (Дата звернення: 17.10.2025).
6. EasyWeek. Online appointment software for personal trainers [Електронний ресурс]. – Режим доступу: https://easyweek.com.ua/solutions/gym. – (Дата звернення: 19.11.2025).
7. FITR. Personal Trainer Software [Електронний ресурс]. – Режим доступу: https://coachwithfitr.com. – (Дата звернення: 31.10.2025).
8. FitBudd. Embracing the Rise of Online Training Platforms: A Guide for Fitness Coaches [Електронний ресурс]. – 2024. – Режим доступу: https://fitbudd.com. – (Дата звернення: 29.10.2025).
9. Fitness.edu.au. Online Personal Training and Virtual Coaching: Adapting to the Digital Fitness Revolution [Електронний ресурс]. – 2024. – Режим доступу: https://fitness.edu.au. – (Дата звернення: 20.10.2025).

10. GymDesk Blog. Personal Trainer: Definition, Types, Qualifications, and Career Outlook [Електронний ресурс]. – 2025. – Режим доступу: https://gymdesk.com/blog. – (Дата звернення: 21.11.2025).
11. Jhavtech Studios. Best PHP Frameworks in 2024 (розділ про Laravel) [Електронний ресурс]. – Medium. – Режим доступу: https://medium.com. – (Дата звернення: 02.11.2025).
12. Kinsta. Laravel vs Node: A Head-to-Head Comparison [Електронний ресурс]. – Режим доступу: https://kinsta.com. – (Дата звернення: 12.10.2025).
13. Laravel Official Site. Laravel – The PHP Framework for Web Artisans [Електронний ресурс]. – Режим доступу: https://laravel.com. – (Дата звернення: 04.11.2025).
14. Melton D. I. The Current State of Personal Training: an Industry Perspective // *Journal of Strength and Conditioning Research*. – 2008.
15. Node.js Foundation. Don»t block the Event Loop (or the Worker Pool) [Електронний ресурс]. – Режим доступу: https://nodejs.org. – (Дата звернення: 16.11.2025).
16. Node.js Foundation. The Node.js Event Loop, Timers, and process.nextTick() [Електронний ресурс]. – Режим доступу: https://nodejs.org. – (Дата звернення: 19.10.2025).
17. Orbitwebtech. Reasons to choose Node.js for web development [Електронний ресурс]. – Режим доступу: https://orbitwebtech.com. – (Дата звернення: 26.10.2025).
18. Peanutsquare. 10 reasons why Laravel is best for web app development (2024) [Електронний ресурс]. – Режим доступу: https://peanutsquare.com. – (Дата звернення: 01.11.2025).
19. ProjectGuru. A guided introduction to MySQL Workbench and EER modeling [Електронний ресурс]. – Режим доступу: https://projectguru.in. – (Дата звернення: 24.11.2025).

20. ResearchGate. The effectiveness of a virtual fitness trainer app in motivating and engaging students for fitness activity by applying motor learning theory [Електронний ресурс]. – 2020. – Режим доступу: https://researchgate.net. – (Дата звернення: 02.12.2025).
21. SimplyBook.me. Online appointment scheduling software for personal trainers [Електронний ресурс]. – Режим доступу: https://simplybook.me. – (Дата звернення: 14.10.2025).
22. Sport Life. Персональний тренер – офіційний сайт мережі фітнес-клубів [Електронний ресурс]. – Режим доступу: https://sportlife.ua. – (Дата звернення: 17.11.2025).
23. Superprof Україна. Персональні тренування для кожного [Електронний ресурс]. – 2023. – Режим доступу: https://superprof.ua. – (Дата звернення: 12.11.2025).
24. Swovo. Node.js vs Laravel: Which is better? [Електронний ресурс]. – Режим доступу: https://swovo.com. – (Дата звернення: 26.11.2025).
25. Technogym. Personal trainer: story of a key role of a fitness centre [Електронний ресурс]. – 2024. – Режим доступу: https://technogym.com. – (Дата звернення: 11.10.2025).
26. Time2book. Personal trainer software [Електронний ресурс]. – Режим доступу: https://time2book.me/schedule. – (Дата звернення: 01.12.2025).
27. TrueCoach. #1 Personal Trainer Software [Електронний ресурс]. – Режим доступу: https://truecoach.co. – (Дата звернення: 16.10.2025).
28. Zenamu. Online Booking System for Fitness Centers & Sports Clubs [Електронний ресурс]. – Режим доступу: https://zenamu.com. – (Дата звернення: 04.12.2025).