Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

ВЕБЗАСТОСУНОК ГЕНЕРАЦІЇ ПРОГРАМ ТРЕНУВАНЬ НА ОСНОВІ ДАНИХ ФУНКЦІОНАЛЬНОСТІ КОРИСТУВАЧА

WEB APPLICATION FOR GENERATING TRAINING PROGRAMMES BASED ON USER FUNCTIONALITY DATA

спеціальність 122 Комп'ютерні науки

освітня програма «Комп'ютерні науки»

Виконав: здобувач вищої освіти
групи КНз-41
Веремейчик Антон Олегович

_____
(підпис)

Керівник:
Хиць Руслан Андрійович

_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«2» червня 2026 р.
Гарант ОП: д.п.н., професор
Тулашвілі Юрій Йосипович

_____
(підпис)

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерних наук*
Ступінь вищої освіти: *бакалавр*
Галузь знань*: 12 Інформаційні технології*
Спеціальність: *122 Комп'ютерні науки*
Освітня програма: *«Комп'ютерні науки»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Валерій ЛІЩИНА
«16» січня 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА
ПЕРШОГО (БАКАЛАВРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ

Веремейчик Антон Олегович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Вебзастосунок генерації програм тренувань на основі даних функціональності користувача*
Керівник: *ХИЦЬ Руслан Андрійович*
затверджено наказом закладу вищої освіти від «16» січня 2026 р. № 32/01-02
2. Строк подання кваліфікаційної роботи *«02» червня 2026 р.*
3.Вихідні дані до роботи: *аналітичне дослідження систем управління проектами.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Вступ
Розділ 1. Аналіз предметної області
Розділ 2. Структурна реалізація об'єкта проектування
Розділ 3. Програмна реалізація об'єкта проектування
Розділ 4. Спеціальні розрахунки
Висновки
Додатки
5. Перелік графічного матеріалу: _____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Ліщина В. О.*		
Теоретичне дослідження	*Ліщина В. О.*		
Практична реалізація	*Ліщина В. О.*		
Спеціальні розрахунки	*Ліщина В. О.*		
Показник запозичень тексту	_____%		
Інструментальна перевірка	*Кошелюк В. А.*		
Нормоконтроль	*Сачук В. О.*		
Гарант ОПП	*Тулашвілі Ю.Й.*		

7. Дата видачі завдання *«16» січня 2026 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1.	*Провести огляд літературних джерел по темі кваліфікаційної роботи*	*до 10.02.2026р*	
2.	*Провести аналіз загальної проблеми і вибір напрямків дослідження*	*до 24.02.2026р.*	
3.	*Розробити функціональну схему роботи програмного продукту*	*до 10.03.2026р*	
4.	*Описати засоби розробки об'єкта проектування*	*до 31.03.2026р.*	
5.	*Практична реалізація об'єкта проектування*	*до 05.05.2026р.*	
7.	*Провести спеціальні розрахунки*	*до 21.05.2026р.*	
8.	*Здача чистового варіанту кваліфікаційної роботи на кафедру*	*до 02.06.2026р.*	

Здобувач вищої освіти _____ Антон ВЕРЕМЕЙЧИК

Керівник роботи _____ Руслан ХИЦЬ

АНОТАЦІЯ

Веремейчик А. О. Вебзастосунок генерації програм тренувань на основі даних функціональності користувача. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп’ютерні науки». Луцький національний технічний університет. Луцьк, 2026.

У роботі розглянуто програмне рішення «вебзастосунок генерації програм тренувань». Рішення призначене для автоматизації процесів, пов’язаних із персоналізацією навантаження, добором вправ і контролем прогресу.

Ключові слова: тренування, вебзастосунок, персоналізація, вправи, функціональні показники.

ABSTRACT

Anton Veremeichyk. Web application for generating training programmes based on user functionality data. Manuscript.

Bachelor’s qualifying thesis of the OP "Computer Sciences". Lutsk National Technical University. Lutsk, 2026.

The thesis considers a web application for generating training programmes based on user functionality data intended to automate processes related to workload personalisation, exercise selection, and progress tracking. The relevance of the topic is substantiated, and the main system requirements, architecture, data model, and user scenarios are defined. The implemented functionality includes user profile, functional metrics, goals, limitations, exercises, calendar, and progress. The main scenarios were tested, and an approach to deployment was prepared. The practical value of the work lies in the ability to support safer and more reasonable planning of individual training.

Keywords: training, web application, personalisation, exercises, functional metrics.

ЗМІСТ

ВСТУП

Актуальність теми кваліфікаційної роботи зумовлена зростанням ролі інформаційних технологій у вирішенні прикладних завдань, пов’язаних із вебзастосунку генерації програм тренувань на основі даних функціональності користувача. Сучасні програмні системи мають не лише автоматизувати окремі операції, а й забезпечувати цілісну підтримку користувача: від введення початкових даних до отримання зрозумілого результату, контролю виконання дій та подальшого використання накопиченої інформації. У таких умовах особливого значення набувають веборієнтовані рішення, які доступні з різних пристроїв, легко масштабуються та можуть бути інтегровані з іншими сервісами.

Практична потреба у розробленні такої системи пояснюється тим, що популярність самостійних занять спортом зростає, однак користувачі часто не мають достатніх знань для коректного вибору навантаження, послідовності вправ і режиму відновлення. За відсутності спеціалізованого програмного інструмента значна частина роботи виконується вручну або за допомогою універсальних сервісів, які не враховують специфіку предметної області. Це призводить до втрати часу, дублювання інформації, складності контролю актуального стану даних і зниження якості прийняття рішень. Тому створення цільового застосунку є обґрунтованим як з погляду оптимізації користувацьких процесів, так і з погляду набуття практичних навичок проєктування сучасних програмних продуктів.

Метою кваліфікаційної роботи є підвищення якості планування фізичних занять шляхом створення вебзастосунку, який аналізує вхідні дані користувача та формує адаптовану програму тренувань. Досягнення цієї мети передбачає не лише реалізацію окремих функцій, а й обґрунтування архітектури системи, визначення ролей користувачів, опис інформаційних потоків, вибір технологічного стеку та перевірку працездатності створеного рішення на типових сценаріях використання.

Об’єктом дослідження є процеси формування індивідуальних тренувальних програм з урахуванням фізичного стану та цілей користувача. У межах роботи цей об’єкт розглядається як сукупність взаємопов’язаних дій користувача, інформаційних сутностей, правил обробки даних і результатів, які повинні бути відображені у програмній системі. Такий підхід дає змогу перейти від загального опису проблеми до конкретної моделі застосунку, придатної для реалізації та подальшого тестування.

Предметом дослідження є моделі подання функціональних показників користувача й програмні засоби генерації персоналізованих тренувальних рекомендацій. Предметна область потребує врахування функціональних вимог, обмежень безпеки та ергономічності користування, способів зберігання даних, логіки взаємодії між клієнтською та серверною частинами, а також засобів перевірки правильності роботи основних модулів. Саме ці складові визначають якість майбутнього програмного продукту та його придатність до практичного застосування.

Для досягнення поставленої мети необхідно розв’язати такі основні завдання: проаналізувати сучасний стан предметної області та наявні підходи до вирішення подібних задач; визначити потреби цільових користувачів і сформувати перелік функціональних вимог; розабезпечити структуру даних і логіку взаємодії компонентів; спроєктувати користувацький інтерфейс; реалізувати основні програмні модулі; провести перевірку роботи системи на типових сценаріях; сформулювати висновки щодо ефективності запропонованого рішення.

Функціональне наповнення майбутньої системи має бути безпосередньо пов’язане з тематикою роботи. Для цього передбачається реалізувати такі можливості: збір антропометричних і функціональних даних, вибір мети тренувань, підбір вправ, налаштування інтенсивності, формування календаря занять і можливість коригування програми. Зазначені функції повинні працювати узгоджено, забезпечувати логічний шлях користувача та створювати основу для подальшого розвитку програмного продукту. Особливу увагу

необхідно приділити простоті введення даних, зрозумілості результатів і зменшенню кількості надлишкових дій під час виконання основних операцій.

Методологічною основою роботи є методи аналізу предметної області, правил прийняття рішень, проєктування користувацьких сценаріїв, розробки вебінтерфейсу, організації сховища даних і перевірки коректності рекомендацій. Використання цих методів дає змогу послідовно пройти етапи від аналізу проблеми до створення працездатного програмного рішення. На етапі проєктування важливо визначити інформаційні сутності, зв’язки між ними, сценарії використання та обмеження системи. На етапі реалізації необхідно забезпечити коректну взаємодію компонентів, обробку помилкових ситуацій і достатній рівень надійності під час виконання основних операцій.

Практична цінність отриманого результату полягає в тому, що результат може використовуватися як допоміжний інструмент для початківців, фітнес-клубів або тренерів під час первинного складання планів занять. Запропоноване рішення має забезпечити зменшення рутинних дій, підвищення прозорості роботи з даними та створення більш зручного цифрового середовища для кінцевого користувача. Крім того, виконання роботи сприяє закріпленню навичок повного циклу розробки: від аналізу вимог і проєктування до програмної реалізації, тестування та документування.

РОЗДІЛ 1
ПРЕДМЕТНА ОБЛАСТЬ

1.1 Цифрова підтримка персональних тренувань

Цифрова підтримка тренувань потребує не механічного добору вправ, а узгодження цілі, рівня підготовки, доступного обладнання, обмежень користувача та очікуваної частоти занять. У такій постановці сервіс розглядається як інструмент прийняття прикладного рішення, де кожна рекомендація має пояснюватися вхідними параметрами й бути придатною для повторної перевірки.

Для коректної роботи сервісу важливо розмежувати довідкові дані та персональні параметри користувача. Довідник вправ може бути спільним для всіх, тоді як профіль, обмеження, історія виконання і темп прогресу залишаються індивідуальними.

Персоналізоване тренування вимагає обліку мети, рівня підготовки, доступного часу, обмежень за здоров’ям і попереднього досвіду користувача. У типових умовах ці дані часто залишаються розрізненими, тому програма занять формується інтуїтивно і не завжди відповідає фактичному стану людини.

Предметна область охоплює профіль користувача, вправи, м’язові групи, рівні складності, підходи, повторення, тривалість, відпочинок, історію виконання і показники прогресу. Ці сутності утворюють основу для алгоритму, який добирає вправи й контролює навантаження.

Користувач очікує від вебзастосунку не довідника вправ, а завершеного плану дій. Система має допомогти вибрати ціль, сформувати програму, пояснити структуру тренування, зафіксувати виконання і дати підстави для подальшого коригування.

Основною проблемою є поєднання безпечного рівня навантаження з достатньою різноманітністю вправ. Якщо програма надто проста, вона не підтримує прогрес; якщо вона надмірна, користувач може швидко втратити мотивацію або отримати перевантаження.

Об’єктом дослідження є процес формування персональної тренувальної програми. Предметом дослідження є програмні методи збереження профілю, добору вправ, розрахунку тренувального навантаження і подання результату у вебінтерфейсі.

Метою роботи є створення вебзастосунку, який формує тренувальну програму на основі параметрів користувача і забезпечує зрозумілий супровід основного сценарію від введення даних до отримання плану занять.

1.2 Порівняння рішень для планування навантаження

Порівняння аналогів доповнено оцінюванням прозорості алгоритму, гнучкості налаштувань, зручності зміни плану після отримання результатів і здатності системи працювати з неповними даними. Саме ці критерії визначають, чи може користувач довіряти сформованій програмі та коригувати її без втрати логіки попереднього планування.

Під час аналізу аналогів окремо оцінюються не тільки наявні функції, а й спосіб пояснення рекомендацій. Для користувача має значення, чи зрозуміло, чому система пропонує певну вправу, тривалість або кількість підходів.

Сучасні цифрові рішення у сфері фітнесу можна поділити на мобільні трекери активності, каталоги вправ, сервіси онлайн-тренувань і платформи для складання програм. Кожна група має власні переваги, але не всі вони забезпечують прозору персоналізацію.

Трекери активності добре фіксують кроки, пульс, витрати енергії і загальну динаміку руху. Їхнє обмеження полягає в тому, що вони здебільшого аналізують виконані дії, а не створюють структурований план майбутніх тренувань.

Каталоги вправ корисні як база рухів, описів і технік. Водночас користувачу часто доводиться самостійно визначати, які вправи поєднувати, скільки підходів виконувати і як змінювати програму після кількох занять.

Сервіси з готовими тренувальними програмами зазвичай орієнтовані на широкий сегмент користувачів. Для цієї роботи важливо, щоб план залежав від

конкретних параметрів користувача і міг бути пояснений через ціль, рівень і обмеження.

Аналіз аналогів показує потребу в системі, яка поєднує профіль, каталог вправ, генератор плану, контроль навантаження і журнал виконання. Такий підхід дає змогу створити не статичний список вправ, а керований тренувальний сценарій.

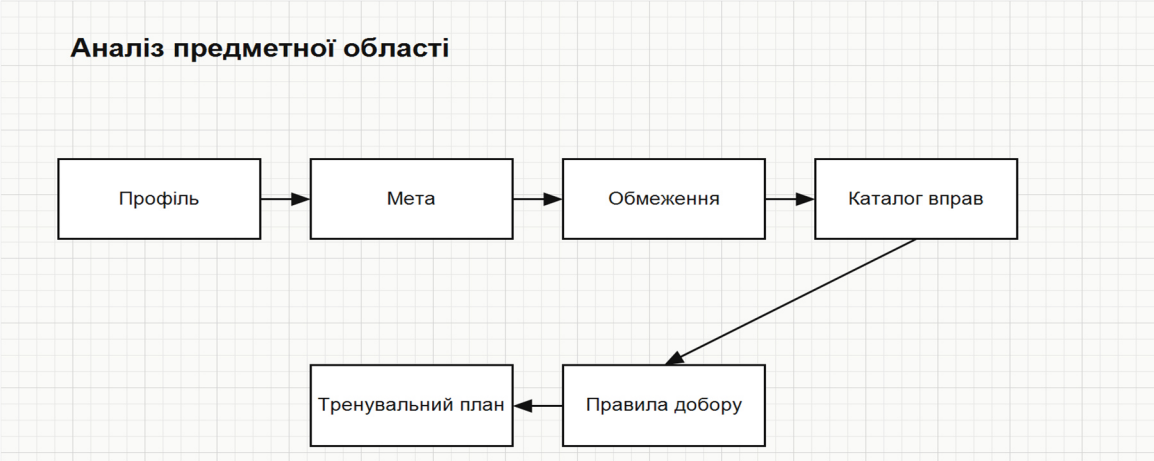

Рисунок 1.1 – Узагальнена схема аналізу предметної області

На рисунку 1.1 подано предметну область тренувального сервісу як послідовність перетворення даних користувача у персоналізований план. Схема показує, що вихідним пунктом є профіль, мета занять і обмеження, після чого система звертається до каталогу вправ, застосовує правила добору та формує тренувальний план. Такий рисунок пояснює, чому у роботі окремо розглядаються профіль користувача, довідник вправ і генератор рекомендацій.

1.3 Вимоги до сервісу формування тренувальних програм

Вимоги до сервісу охоплюють не лише функціональність створення плану, а й якість взаємодії. Зрозумілі повідомлення про помилки, стабільне збереження профілю, наочне подання вправ і відокремлення обов’язкових параметрів від

допоміжних. Це зменшує ризик випадкового формування некоректного тренувального навантаження.

Вимоги до системи мають враховувати регулярний характер використання. Користувач повертається до програми після кожного заняття, тому інтерфейс повинен швидко показувати поточний стан плану, останній результат і наступну дію.

Функціональні вимоги охоплюють створення профілю, вибір мети, введення рівня підготовки, зазначення обмежень, перегляд каталогу вправ, генерацію програми, редагування плану і фіксацію виконання.

Профіль користувача має містити ціль тренувань, бажану частоту занять, рівень підготовки, доступний інвентар, обмеження за вправами і бажану тривалість сесії. Саме ці параметри визначають добір вправ і початковий обсяг навантаження.

Каталог вправ повинен зберігати назву, опис, техніку виконання, м’язові групи, складність, потрібний інвентар, рекомендовані повторення і попередження. Така структура потрібна для фільтрації і безпечного формування програми.

До нефункціональних вимог належать швидка відповідь системи, зрозумілий інтерфейс, коректна обробка неповних даних, захист профілю користувача і можливість подальшого розширення каталогу вправ.

Окремо враховується пояснюваність результату. Користувач має бачити, чому система запропонувала конкретні вправи, як сформовано кількість підходів і які параметри можна змінити.

1.4 Постановка завдання для розробки

Формалізація задачі деталізує межі прототипу. Система приймає профіль користувача, фільтрує вправи, групує їх за метою та рівнем складності, формує план і повертає результат у вигляді зрозумілої послідовності дій. Такий опис задає основу для подальшого проєктування модулів і тестових сценаріїв.

Формалізація задачі дозволяє відокремити обов’язкову функціональність від можливих майбутніх розширень. У межах прототипу головним є стабільний сценарій генерації, перегляду і збереження програми.

Задача дослідження полягає у створенні вебзастосунку, який автоматизує формування персональної тренувальної програми на основі профілю користувача, каталогу вправ і правил добору навантаження.

Для розв’язання задачі потрібно описати предметну область, побудувати модель даних, визначити ролі користувачів, спроєктувати API, реалізувати генератор програми і підготувати інтерфейс для роботи з планом занять.

Система має враховувати неповні або суперечливі дані. Якщо користувач обирає обмеження, які суттєво звужують набір вправ, застосунок повинен пояснити причину і запропонувати змінити параметри.

Для досягнення поставленої мети необхідно розв’язати такі основні завдання: проаналізувати сучасний стан предметної області та наявні підходи до вирішення подібних задач; визначити потреби цільових користувачів і сформувати перелік функціональних вимог; розабезпечити структуру даних і логіку взаємодії компонентів; спроєктувати користувацький інтерфейс; реалізувати основні програмні модулі; провести перевірку роботи системи на типових сценаріях; сформулювати висновки щодо ефективності запропонованого рішення.

Очікуваним результатом є прототип, який дозволяє створити профіль, отримати тренувальний план, переглянути деталі вправ, зберегти програму і зафіксувати виконання.

РОЗДІЛ 2
МОДЕЛЮВАННЯ СИСТЕМИ ТА ПРОЄКТНІ РІШЕННЯ

2.1 Компонентна схема вебзастосунку

Під час проєктування компонентів розмежовано відповідальність інтерфейсу, серверної логіки, генератора плану та сховища даних. Завдяки цьому зміна правил добору вправ не вимагає перебудови екранних форм, а оновлення інтерфейсу не впливає на збережені дані профілю.

Компонентна модель також враховує можливість заміни окремих модулів. Правила генерації тренувань можна вдосконалювати незалежно від екрану профілю або каталогу вправ.

Архітектура застосунку побудована як клієнт-серверна система з окремими рівнями інтерфейсу, API, бізнес-логіки і сховища даних. Такий поділ зменшує зв’язаність модулів і спрощує розвиток генератора тренувань.

Клієнтська частина відповідає за профіль, форми налаштування, перегляд вправ, відображення згенерованої програми і взаємодію з журналом виконання. Вона передає дані до API і показує користувачу результат у структурованому вигляді [1].

Серверна частина виконує валідацію, добір вправ, розрахунок навантаження, збереження планів і підготовку відповіді для інтерфейсу. Саме на цьому рівні концентрується прикладна логіка формування тренувальної програми.

База даних зберігає користувачів, вправи, програми, тренувальні сесії, історію виконання і службові довідники. Структура сховища не дублює вправи в кожному плані, а використовує зв’язки між програмою і каталогом.

На архітектурній схемі розмежовано клієнтський інтерфейс, REST API, модулі профілю, генератор плану, каталог вправ, сховище даних, аналітику прогресу та журнал дій [8]. Рисунок уточнює, що інтерфейс не виконує розрахунки самостійно, а передає параметри на сервер, де правила добору вправ працюють із довідковими та персональними даними.

Рисунок 2.1 – Архітектура програмної системи

Рисунок 2.1 подає узагальнену компонентну схему і показує, як профіль користувача переходить у запит до генератора, а результат повертається в інтерфейс як тренувальний план.

2.2 Сценарії користувача і модель даних

Сценарії користувача розглядаються як послідовність перевірюваних переходів, заповнення профілю, вибір цілі, запуск генерації, перегляд результату, коригування параметрів і збереження плану. Модель даних підтримує цей маршрут через окремі сутності профілю, вправи, плану, тренувального дня та результату виконання.

Модель даних проєктується так, щоб зберігати не лише підсумковий план, а й підстави його формування. Це допомагає повторно відтворити рішення генератора після зміни профілю або каталогу вправ.

Основний сценарій починається з налаштування профілю. Користувач обирає ціль, частоту занять, рівень підготовки, інвентар і обмеження, після чого система формує початковий варіант програми.

У системі виділяються ролі користувача та адміністратора. Користувач працює з профілем, планом і журналом виконання, а адміністратор підтримує каталог вправ, категорії, рівні складності і службові параметри.

Модель даних включає сутності користувача, профілю, вправи, тренувального плану, сесії, підходу і запису прогресу. Такий склад дозволяє відстежувати, які вправи включені до програми і як змінюється виконання.

Тренувальний план зберігається як набір сесій, а кожна сесія містить вправи з параметрами підходів, повторень і відпочинку. Це дозволяє редагувати окрему частину програми без повного перестворення плану.

Рисунки 2.2-2.4 пояснюють варіанти використання, логічну модель даних і послідовність генерації програми.

Рисунок 2.2 – UML-діаграма варіантів використання системи

UML-діаграма показує основні варіанти використання для користувача: заповнення профілю, вибір мети, генерацію плану, перегляд вправ, фіксацію результату та перегляд прогресу. Пунктирні залежності відображають, що генерація плану спирається на попередньо введені параметри й дані каталогу, а не є ізольованою дією [16].

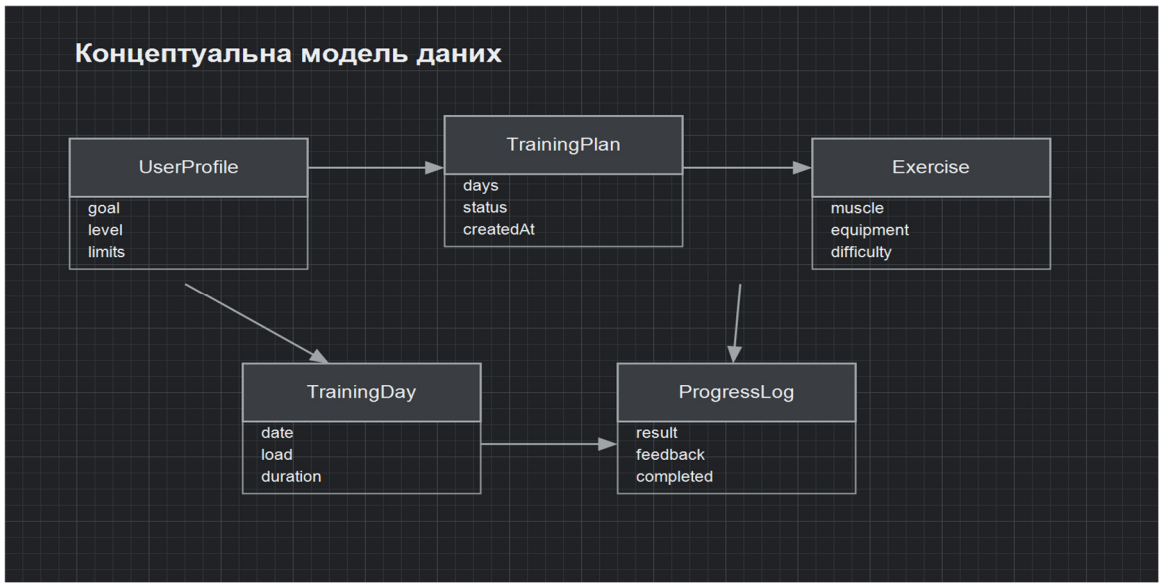

Рисунок 2.3 – Концептуальна модель даних

Концептуальна модель даних містить сутності UserProfile, TrainingPlan, Exercise, TrainingDay і ProgressLog. На рисунку показано, як персональні параметри користувача пов’язані з планом, план деталізується за тренувальними днями, вправи підбираються з каталогу, а виконання фіксується в журналі прогресу [2-4].

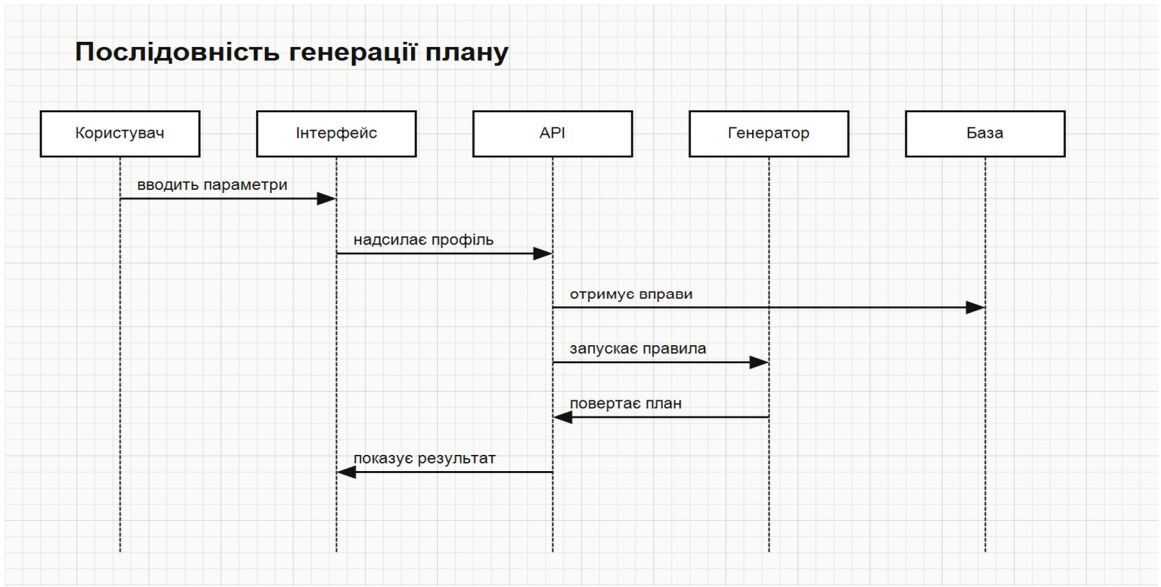

Рисунок 2.4 – Послідовність виконання основного сценарію

Діаграма послідовності відображає основний сценарій генерації плану: користувач вводить параметри, інтерфейс надсилає профіль до API, сервер отримує вправи зі сховища, передає їх генератору правил і повертає готовий результат на екран. Така схема пояснює порядок взаємодії компонентів під час виконання ключової функції.

2.3 Інтерфейс, стек технологій і критерії якості

Інтерфейсні рішення уточнено з позиції щоденного використання. Користувач має швидко бачити поточну мету, заплановане навантаження, найближчу дію та причину відхилення вправ. Технологічний стек обрано так, щоб демонстраційний прототип залишався простим для запуску й водночас відображав реальну клієнт-серверну взаємодію.

Критерії якості пов’язані з безперервністю користувацького сценарію. Якщо користувач редагує профіль або відкриває вправу, система має зберігати контекст поточного тренувального плану.

Інтерфейс проєктується як послідовність робочих екранів авторизації, головна панель, профіль, каталог вправ, згенерована програма, картка вправи і журнал виконання.

Екран результату має показувати програму за днями або сесіями, основні параметри навантаження, попередження і дії редагування. Користувач повинен швидко зрозуміти, що виконувати і чому це включено до плану.

Технологічний стек обирається з урахуванням REST API, динамічного інтерфейсу, бази даних і можливості локального запуску. Архітектура не прив’язує генератор до конкретного екрана, тому логіку можна повторно використовувати [5].

Критеріями якості є коректність фільтрації вправ, стабільність генерації, зрозумілі повідомлення, швидке завантаження і можливість відновити попередній план.

Рисунок 2.5 показує макет взаємодії користувача з профілем, каталогом і результатом генерації.

Рисунок 2.5 – Узагальнений макет екранних форм і навігації

Навігаційна схема демонструє зв’язки між головною сторінкою, профілем, каталогом вправ, планом тижня, прогресом, налаштуваннями та довідником. Рисунок показує, що користувач може повернутися до профілю або каталогу після перегляду плану, тому інтерфейс підтримує повторне коригування параметрів.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ТРЕНУВАЛЬНОГО СЕРВІСУ

3.1 Організація проєкту та початкові модулі

Початкові модулі реалізуємо так, щоб вони підтримували наскрізний сценарій без зайвої залежності від зовнішніх сервісів. На цьому етапі ми готуємо структуру даних, базові маршрути API, демонстраційні записи та мінімальний інтерфейс, який дозволяє перевірити логіку вже під час розробки [6-7].

Початкові модулі реалізуються так, щоб демонстраційний запуск не залежав від зовнішніх сервісів. Це спрощує перевірку роботи прототипу в навчальному середовищі.

Реалізуємо прототип тренувального сервісу з окремими модулями профілю, каталогу вправ, генератора програми, журналу виконання і адміністративних довідників.

Структуру коду організовуємо так, щоб інтерфейс, API, сервіси і сховище даних мали окремі зони відповідальності. Це спрощує тестування і дає змогу змінювати генератор без переписування форм.

У клієнтській частині створюємо екрани входу, панелі користувача, списку вправ, форми редагування, детального перегляду і результатів програми.

У серверній частині реалізуємо маршрути для створення записів, перевірки даних, запуску генератора і збереження результату.

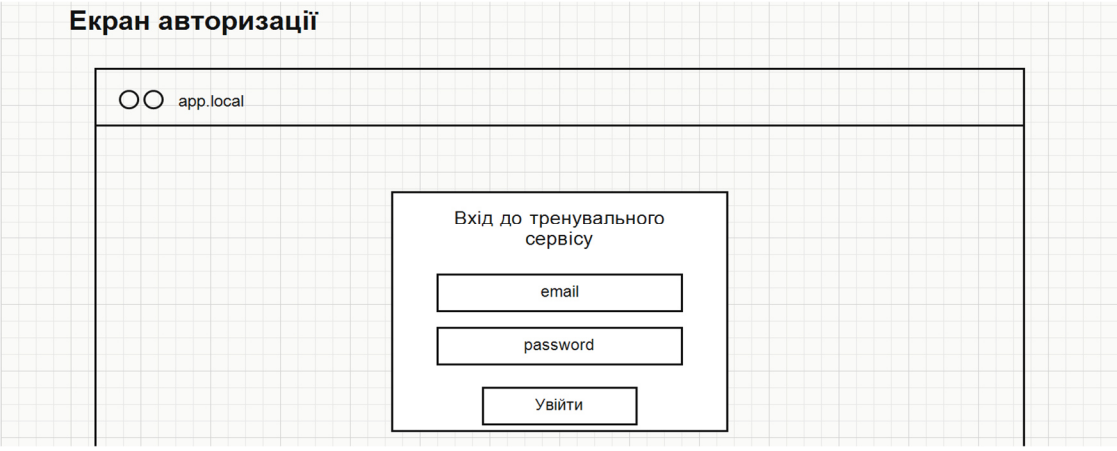

Рисунок 3.1 – Макет екрана авторизації користувача

Рисунки 3.1-3.2 демонструють початкові екрани системи, а лістинг 3.1 показує базову модель даних для тренувального плану.

Макет авторизації містить мінімальний набір елементів для входу до тренувального сервісу. Поля електронної пошти й пароля та кнопку підтвердження. Рисунок використовується для пояснення стартової точки користувацького сценарію, після якої система може прив’язувати збережений профіль і результати тренувань до конкретного користувача.

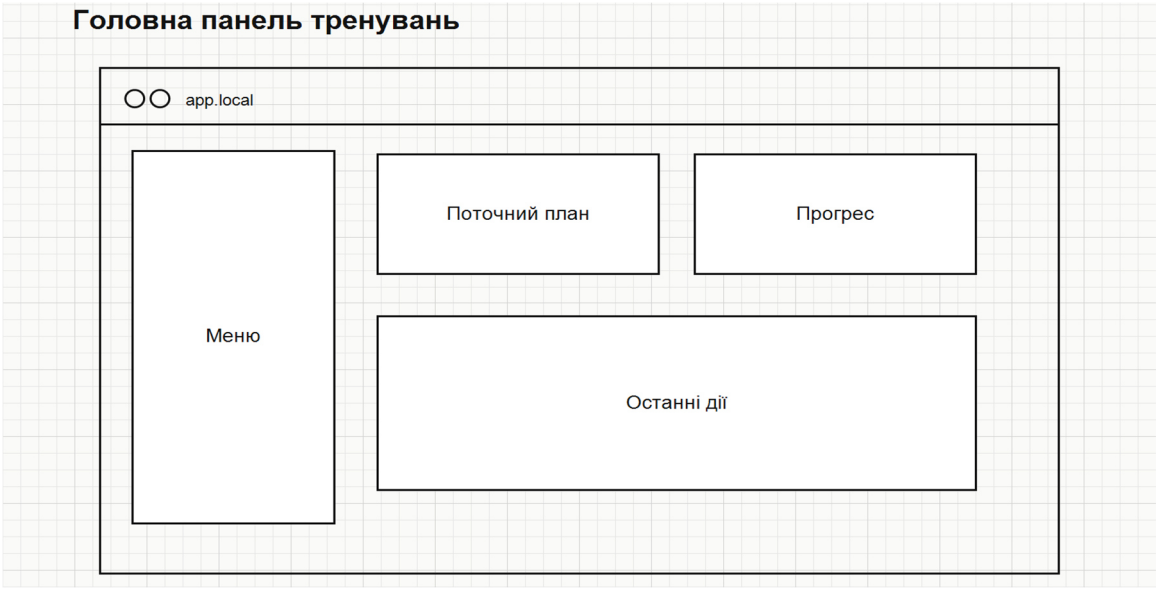

Рисунок 3.2 – Головна панель користувача застосунку

Головна панель тренувань зосереджує поточний план, показник прогресу, меню розділів і блок останніх дій. На рисунку відображено екран, з якого користувач переходить до основних функцій: редагування профілю, перегляду вправ, аналізу прогресу та запуску генерації оновленої програми.

У лістингу 3.1 наведено структуру даних, через яку система зберігає основні параметри предметної області для роботи.

Лістинг 3.1 – Опис базової моделі даних для об’єкта «тренувальний план»

```
export interface MainRecord {
id: string;
title: string;
status: 'draft' | 'active' | 'done' | 'archived';
ownerId: string;
```

```
createdAt: string;
updatedAt: string;
fields: {
description?: string;
domainObject: 'тренувальний план';
mainModule: 'вправи';
resultType: 'прогрес тренувань';
};
}
export const emptyRecord = (ownerId: string): MainRecord => ({
id: crypto.randomUUID(),
title: '',
status: 'draft',
ownerId,
createdAt: new Date().toISOString(),
updatedAt: new Date().toISOString(),
fields: {
domainObject: 'тренувальний план',
mainModule: 'вправи',
resultType: 'прогрес тренувань'
}
});
```

кінець лістингу 3.1

Фрагмент коду демонструє практичний метод реалізації: вхідні дані приводяться до потрібної структури, виконується контроль помилок, після чого результат передається наступному модулю системи.

3.2 Серверні методи, дані і генератор тренувань

Серверні методи описуємо через вхідні параметри, перевірку, формування відповіді та обробку виняткових ситуацій. Показзано роль кожного методу у створенні тренувального плану, зокрема фільтрацію вправ, розподіл навантаження та повернення структурованого результату клієнтській частині.

Серверні методи повертають не тільки результат, а й службові ознаки виконання. Завдяки цьому клієнтська частина може відобразити успішне створення плану, помилку валідації або потребу змінити параметри.

У серверній частині реалізуємо модель даних для користувача, вправи, профілю, тренувального плану, сесії та журналу виконання.

Модель вправи містить назву, м’язову групу, рівень складності, інвентар, опис техніки і рекомендований діапазон повторень. Ці дані використовуємо під час фільтрації і добору.

Генератор тренувань приймає профіль користувача, відбирає допустимі вправи, групує їх за сесіями і визначає початкові параметри навантаження.

Під час створення записів виконуємо серверну валідацію, перевіряємо назву, статус, допустимі значення, наявність профілю і право користувача на зміну даних.

Після запуску основного сценарію сервер повертає структурований план, який клієнтська частина може відобразити як програму занять.

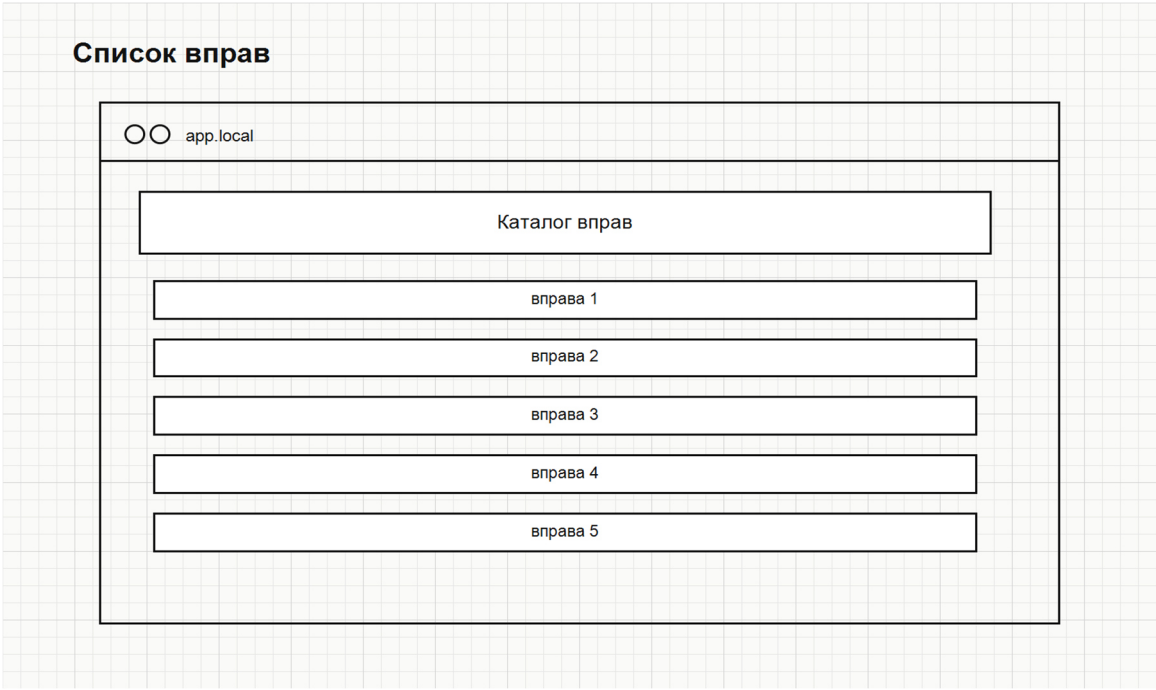

Рисунок 3.3 – Інтерфейс списку об’єктів системи

На рисунку 3.3 показуємо список вправ або сформованих програм. Екран використовується для швидкого перегляду ключових характеристик, переходу до деталей і запуску дій редагування.

Інтерфейс списку вправ подано як таблицю або перелік записів каталогу, де кожна вправа може мати назву, групу м’язів, складність і доступне

обладнання. Рисунок показує, що каталог є не декоративним блоком, а джерелом даних для генератора тренувального плану.

Список побудовано так, щоб користувач бачив статус, назву, короткі параметри і доступні команди без зайвого переходу між сторінками.

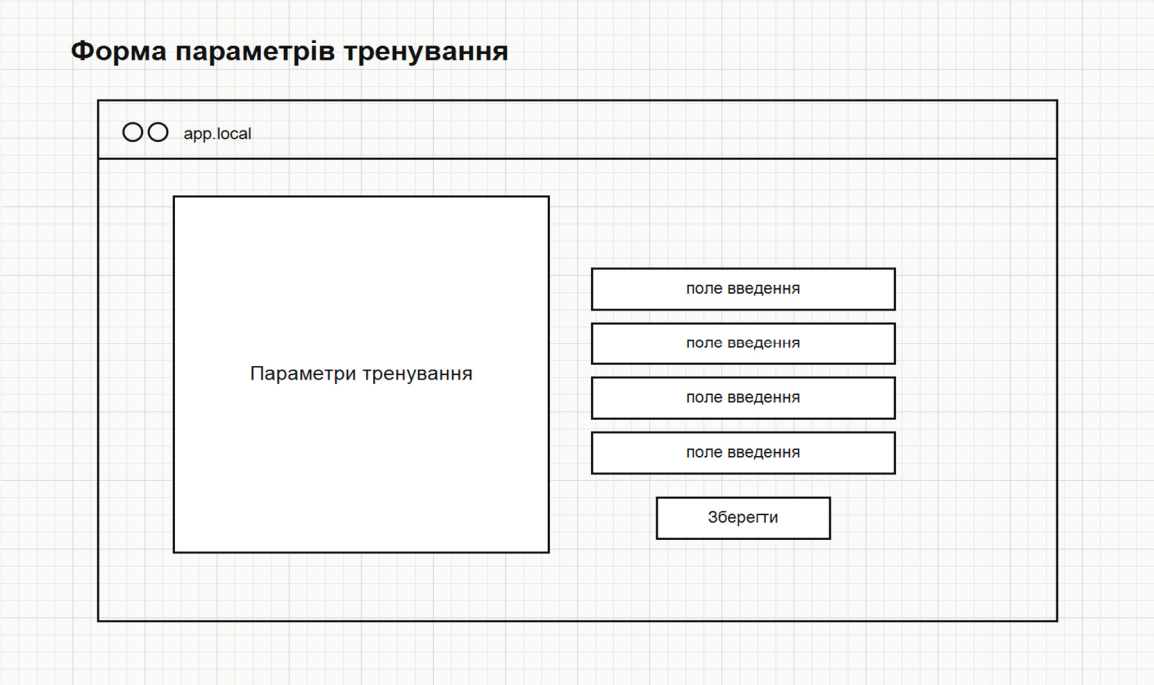

Рисунок 3.4 – Форма створення або редагування запису

На рисунку 3.4 демонструємо форму створення або редагування тренувальної програми. Поля згруповано за змістом, щоб користувач послідовно вводив основні та службові дані. Форма параметрів тренування відображає поля, через які користувач задає мету, рівень підготовки, обмеження, тривалість і бажану частоту занять. Цей рисунок пояснює, які саме дані потрапляють до серверної валідації та впливають на подальший добір вправ.

Форма працює разом із серверною валідацією: клієнтська частина допомагає уникнути очевидних помилок, а сервер контролює коректність перед збереженням.

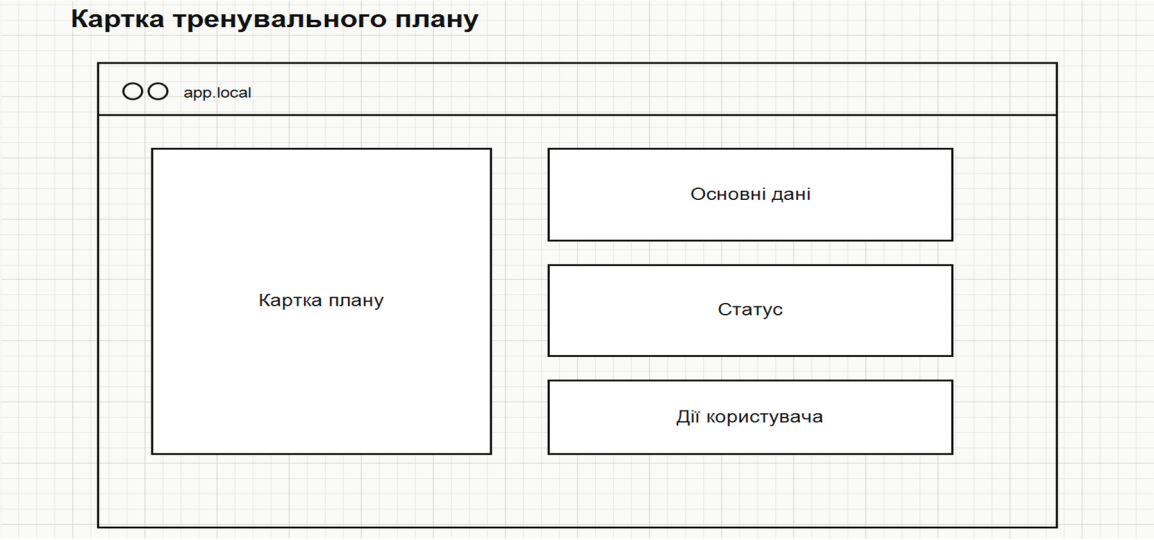

Рисунок 3.5 – Картка детального перегляду об’єкта

На рисунку 3.5 подано картку детального перегляду тренувальної програми. Вона відкриває повну інформацію і дає змогу перейти до редагування або пов’язаних дій. Картка тренувального плану показує підсумковий результат генерації: основні дані плану, статус, перелік дій користувача та можливість перегляду деталей. Рисунок демонструє, як сформований план подається не як суцільний текст, а як структурований об’єкт інтерфейсу.

Картка використовується як проміжний екран між загальним списком і конкретною операцією, тому поєднує інформаційний та навігаційний блоки.

Лістинг 3.2 – Метод API для створення запису в системі

```
router.post('/records', authorize('користувач'), async (req, res, next) => {
try {
const payload = validateRecordPayload(req.body);
const record = await recordService.createRecord({
...payload,
ownerId: req.user.id,
source: 'система персоналізованого планування тренувань'
});
res.status(201).json({
ok: true,
message: 'Запис успішно створено',
data: record
});
} catch (error) {
next(error);
}});
```

кінець лістингу 3.2

У лістингу 3.2 наведено серверний маршрут, що приймає запит, перевіряє дані та повертає клієнтській частині структуровану відповідь для роботи.

У лістингу 3.3 показано функцію перевірки даних перед збереженням. Ми контролюємо обов’язкові поля, статуси і текстові значення.

Лістинг 3.3 – Функція серверної валідації вхідних даних

```
function validateRecordPayload(payload) {
const errors = {};
if (!payload.title || payload.title.trim().length < 3) {
errors.title = 'Назва повинна містити щонайменше 3 символи';
}
if (!payload.status) {
errors.status = 'Необхідно вказати статус запису';
}
if (Object.keys(errors).length > 0) {
throw new ValidationError('Помилка перевірки даних', errors);
}
return {
title: payload.title.trim(),
status: payload.status,
description: payload.description?.trim() ?? ''
};}
```

кінець лістингу 3.3

У лістингу 3.4 наведено приклад сервісної логіки, у якій бізнес-правила відокремлені від інтерфейсу користувача.

Лістинг 3.4 – Сервіс виконання основного сценарію «генерація плану»

```
async function runMainScenario(recordId, userId) {
const record = await repository.findById(recordId);
assertAccess(record, userId);
const steps = 'анкета, введення показників, вибір мети, генерація плану, виконання,
коригування'.split(',').map(step => step.trim());
const result = await scenarioEngine.execute({
record,
steps,
action: 'генерація плану',
expectedResult: 'прогрес тренувань'
});
await auditLog.write({
userId,
recordId,
action: 'генерація плану',
status: result.status
});
return result;
}
```

кінець лістингу 3.4

Для тренувального сервісу цю перевірку розширюємо правилами для рівня підготовки, вправ, підходів і обмежень користувача. У лістингу наведено перевірку вхідних даних перед виконанням основної операції.

3.3 Клієнтські екрани та взаємодія користувача

Клієнтські екрани розглядаємо як практичне відображення серверної логіки. Ми показуємо, як користувач переходить від профілю до каталогу вправ, як запускає генерацію, як бачить сформовану програму та як система повідомляє про некоректні або неповні дані.

Клієнтські екрани пов’язані між собою єдиним станом програми. Після збереження профілю або редагування вправи користувач бачить оновлений план без повторного проходження всього сценарію.

Клієнтський інтерфейс реалізуємо як набір екранів, що підтримують шлях користувача від профілю до готової програми.

На головній панелі показуємо останній план, швидкий запуск генерації, поточний стан профілю і коротку статистику виконання.

Список вправ підтримує пошук, фільтрацію за м’язовою групою, рівнем і інвентарем. Це допомагає користувачу перевірити, з яких рухів формується програма.

Екран результатів подає програму за сесіями, показує вправи, підходи, повторення і відпочинок. Детальна картка відкриває техніку виконання і попередження.

Форми передають дані до API, показують стан завантаження і повертають помилки у зрозумілому вигляді.

На рисунку 3.6 показуємо екран результатів, де користувач бачить згенеровану програму, структуру занять і ключові показники навантаження.

Рисунок 3.6 – Екран результатів та аналітики

Екран прогресу та аналітики містить графічне подання динаміки навантаження або виконання занять. Рисунок пояснює, як сервіс може відображати накопичені результати, щоб користувач бачив зміни у виконанні плану й міг приймати рішення щодо подальшого коригування. Екран допомагає оцінити підсумок основного сценарію і перейти до коригування даних або збереження результату.

Рисунок 3.7 – Адміністративні налаштування застосунку

На рисунку 3.7 показуємо адміністративні налаштування каталогу вправ, рівнів складності і службових статусів.

Адміністративний екран дозволяє підтримувати довідники без прямого редагування бази даних.

Адміністративні налаштування охоплюють правила генерації, довідники, журнал змін і параметри доступу. Рисунок показує службову частину прототипу, через яку можна підтримувати актуальність каталогу вправ і контролювати логіку формування тренувальних програм.

Лістинг 3.5 – Клієнтський метод збереження даних через REST API

```
export async function saveRecord(formState) {
const response = await fetch('/api/records', {
method: 'POST',
headers: { 'Content-Type': 'application/json' },
body: JSON.stringify(formState)
});
const body = await response.json();
if (!response.ok || !body.ok) {
throw new ApiError(body.message, body.errors);
}
return body.data;
}
```

кінець лістингу 3.5

У лістингу 3.5 демонструємо клієнтський метод збереження даних через REST API. Ми формуємо запит, передаємо JSON-об’єкт і перевіряємо відповідь сервера. У лістингу наведено серверний маршрут, що приймає запит, перевіряє дані та повертає клієнтській частині структуровану відповідь.

Метод ізольований від конкретної форми, тому може використовуватися на різних екранах створення або редагування [10].

У лістингу 3.6 наведено обробник відправлення форми. Ми вмикаємо стан завантаження, очищуємо попередні помилки і викликаємо метод збереження. У лістингу наведено зв’язок екранної форми з прикладною логікою та відображенням результату.

Після успішної операції інтерфейс переходить до створеного запису, а у разі помилки показує користувачу зрозуміле повідомлення.

Лістинг 3.6 – Обробник форми користувацького інтерфейсу

```
async function handleSubmit(event) {
event.preventDefault();
setLoading(true);
setErrors({});
try {
const saved = await saveRecord(formState);
notify('Дані збережено');
navigate('/records/' + saved.id);
} catch (error) {
setErrors(error.details ?? {});
notify(error.message ?? 'Не вдалося виконати операцію');
} finally {
setLoading(false);
}}
```

кінець лістингу 3.6

Демонстраційні дані добираються так, щоб показати різні рівні підготовки і різні цілі. Це дозволяє наочно перевірити, як змінюється програма після коригування профілю.

Для демонстрації готуємо набір вправ, профілі різного рівня, кілька цілей тренувань і приклади сформованих програм.

Тестові профілі охоплюють початковий, середній і підвищений рівень підготовки, а також різні часові обмеження.

Перед запуском перевіряємо API, відображення форм, роботу генератора, збереження плану і відкриття детальної картки вправи.

РОЗДІЛ 4
ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ ТА ВПРОВАДЖЕННЯ

4.1 Сценарії контролю і середовище тестування

План тестування деталізуємо за типами перевірок. Позитивні сценарії підтверджують основний маршрут, негативні демонструють реакцію на помилки, граничні перевіряють крайні значення параметрів, а інтеграційні показують зв’язок між клієнтом, API та сховищем даних.

Під час підготовки тестів окремо фіксуються очікувані результати для кожного профілю. Це дає змогу порівняти фактичний план із передбаченою логікою генерації.

Перевіряємо прототип як завершений інструмент формування тренувальної програми.

Тестові сценарії поділяємо на позитивні, негативні, граничні та інтеграційні. Так окремо перевіряємо профіль, каталог, генератор, інтерфейс і збереження результату.

Позитивні сценарії охоплюють створення профілю, запуск генерації, перегляд програми і фіксацію виконання.

Негативні сценарії перевіряють порожній профіль, некоректні значення, нестачу вправ у каталозі і відсутність права доступу.

Рисунок 4.1 і таблиця 4.1 фіксують схему перевірки та контрольні тестові випадки.

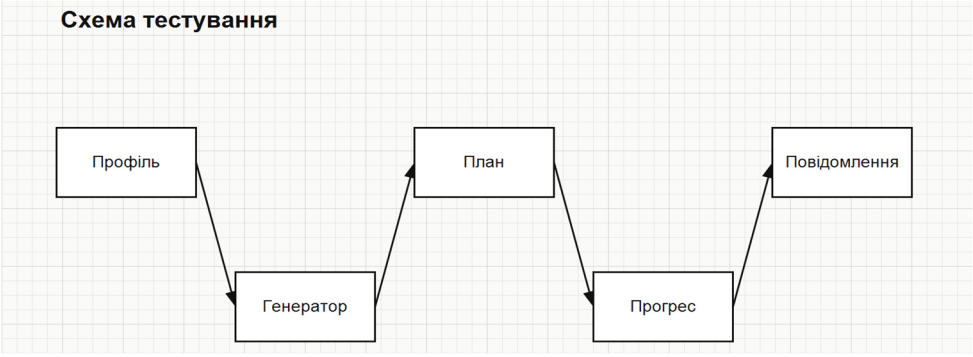

Рисунок 4.1 – Схема тестування застосунку

Схема тестування відображає контрольний маршрут перевірки: профіль користувача, генератор, план, прогрес і повідомлення системи. Стрілки показують, у якій послідовності перевіряється основна функціональність і де фіксуються очікувані результати.

Таблиця 4.1 – Контрольні тестові випадки

ID	Перевірка	Очікуваний результат
TC-01	Вхід тестового користувача	Користувач переходить до основної панелі
TC-02	Переміщення картки	Система формує результат
TC-03	Некоректні дані у формі	Показуємо повідомлення про помилку
TC-04	Спроба службової дії без прав	Доступ заблоковано
TC-05	Повторний запуск сценарію	Дані не дублюються, стан оновлюється

У розділі тестування ми перевіряємо вебзастосунок генерації тренувальних програм як завершений прототип; враховуємо зовнішні залежності та виділення ключових об’єктів. Основний сценарій охоплює добір вправ, розрахунок навантаження і створення плану, збереження даних і отримання результату «персональна програма тренувань».

4.2 Результати перевірки функцій

Результати перевірки подаємо через зіставлення очікуваної та фактичної поведінки. Для тренувального сервісу особливе значення має не тільки факт створення плану, а й коректність його складу, відповідність обмеженням користувача та зрозумілість повідомлень інтерфейсу.

Результати перевірки оцінюються не тільки за фактом виконання запиту, а й за якістю відповіді. Повідомлення системи мають пояснювати користувачу, що саме відбулося і яку дію можна виконати далі.

Функціональне тестування підтвердило, що система створює програму для профілів із коректними параметрами і не формує випадковий результат за суперечливих умов.

Перевірка генератора показала, що добір вправ залежить від рівня, мети, інвентарю і обмежень користувача.

Інтеграційні сценарії підтвердили зв’язок між профілем, каталогом і результатом генерації.

Виявлені зауваження стосуються розширення каталогу вправ і подальшого вдосконалення правил прогресії навантаження.

Рисунок 4.2 і таблиця 4.2 узагальнюють результати контрольної перевірки.

Рисунок 4.2 – Узагальнені результати контрольного тестування

Рисунок подає результати контрольної перевірки у вигляді матриці сценаріїв, вхідних даних, очікуваної поведінки та статусу виконання. Для тренувального сервісу така форма дозволяє окремо зафіксувати перевірку профілю, обмежень, порожніх даних, генерації плану та відображення прогресу.

Таблиця 4.2 – Критерії готовності до демонстрації

Критерій	Стан	Коментар
Основний сценарій	Виконуємо	генерація програми тренувань
Тестові дані	Підготовлено	тренувальний план
Помилки введення	Перевіряємо	Система відображає повідомлення
Журнал подій	Доступний	Фіксуються ключові дії
Розгортання	Підготовлено	Тестове середовище

Позитивний сценарій підтверджує, що користувач може виконати дію «добір вправ, розрахунок навантаження і створення плану» без помилок; працює журналювання ключових дій та узгодження рисунків із текстом. Фіксуємо очікуваний результат, фактичну поведінку і стан даних після завершення операції.

Порядок запуску описуємо як повторювану послідовність дій: підготовка середовища, встановлення залежностей, запуск серверної частини, відкриття клієнтського інтерфейсу та контрольний прохід демонстраційного сценарію. Це робить розгортання відтворюваним для перевірки роботи.

Контрольна демонстрація завершується проходженням повного маршруту користувача, створення профілю, генерація плану, відкриття вправи, збереження результату і перегляд наступного тренування.

Розгортання виконуємо після перевірки функціональних сценаріїв. Ми запускаємо сервер, клієнтську частину і тестову базу вправ.

Контрольний запуск починаємо зі створення профілю, після чого генеруємо програму, відкриваємо вправу і зберігаємо результат.

Під час запуску контролюємо відповіді API, повідомлення про помилки, стан початкових даних і доступність інтерфейсу.

Після контрольної демонстрації прототип готовий до показу основної функціональності.

ВИСНОВКИ

У кваліфікаційній роботі було вирішено прикладну задачу, пов’язану з розробленням вебзастосунку для генерації персоналізованих тренувальних програм. У процесі виконання роботи проаналізовано предметну область, визначено потребу в обліку мети, рівня підготовки, обмежень, доступного інвентарю та історії виконання.

Досліджено сучасні підходи до цифрової підтримки тренувань, визначено основні інформаційні сутності та сформовано вимоги до сервісу. Результатом цього етапу стало уточнення ролі профілю користувача, каталогу вправ і правил формування навантаження.

Виконано проєктування системи. Описано компонентну схему вебзастосунку, сценарії користувача, модель даних, структуру інтерфейсу, технологічний стек і критерії якості, які забезпечують перехід від аналізу до програмної реалізації.

Реалізовано основні модулі прототипу: профіль користувача, каталог вправ, серверні методи, генератор тренувальної програми, клієнтські екрани та демонстраційні дані. Реалізація підтримує отримання плану, перегляд вправ і збереження результату.

Проведено тестування і підготовку проєкту до запуску. Перевірено позитивні, негативні, граничні та інтеграційні сценарії, а також контрольний прохід від створення профілю до перегляду сформованої програми.

Практичне значення роботи полягає у створенні навчального прототипу, який може бути розширений засобами прогресії навантаження, аналітики виконання і персональних рекомендацій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ACSM. Exercise Preparticipation Health Screening. American College of Sports Medicine. URL: https://www.acsm.org/education-resources/trending-topics-resources/exercise-preparticipation-screening (дата звернення: 12.02.2026).

2. Android Developers. App architecture guide. Google. URL: https://developer.android.com/topic/architecture (дата звернення: 20.04.2026).

3. Apple Developer. HealthKit. Apple Inc.. URL: https://developer.apple.com/health-fitness/ (дата звернення: 20.04.2026).

4. Docker Docs. Get started. Docker Inc.. URL: https://docs.docker.com/get-started/ (дата звернення: 20.04.2026).

5. Express.js. Routing guide. OpenJS Foundation. URL: https://expressjs.com/en/guide/routing/ (дата звернення: 20.04.2026).

6. FastAPI. Documentation. FastAPI. URL: https://fastapi.tiangolo.com/ (дата звернення: 20.04.2026).

7. Fitbit Web API. Google. URL: https://dev.fitbit.com/build/reference/web-api/ (дата звернення: 20.04.2026).

8. Garmin Health API. Garmin. URL: https://developer.garmin.com/health-api/overview/ (дата звернення: 12.02.2026).

9. HTTP overview. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview (дата звернення: 20.04.2026).

10. Node.js. Introduction to Node.js. OpenJS Foundation. URL: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs (дата звернення: 20.04.2026).

11. OpenAPI Specification. SmartBear Software. URL: https://swagger.io/specification/ (дата звернення: 20.04.2026).

12. OWASP Top 10 Web Application Security Risks. OWASP Foundation. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 20.04.2026).

13. PostgreSQL Documentation. PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/current/ (дата звернення: 20.04.2026).

14. React. Learn React. Meta Open Source. URL: https://react.dev/learn (дата звернення: 20.04.2026).

15. Web Content Accessibility Guidelines (WCAG) 2.2. W3C. URL: https://www.w3.org/TR/WCAG22/ (дата звернення: 20.04.2026).

16. WHO. Physical activity. World Health Organization. URL: https://www.who.int/news-room/fact-sheets/detail/physical-activity (дата звернення: 12.02.2026).

ДОДАТКИ

ДОДАТОК А

Фрагменти програмного коду ілюструють реалізацію основних методів розробки для вебзастосунку генерації програм тренувань. Лістинги відображають структуру моделі даних, API-методу та сервісу виконання основної операції.

Лістинг А.1 – Опис базової моделі даних.

```
export interface WorkoutPlan {
  id: string;
  title: string;
  status: 'draft' | 'active' | 'done';
  createdAt: string;
  updatedAt: string;
  payload: {
    objectName: 'тренувальний план';
    actionName: 'генерація тренувального плану';
  };
}
```

Лістинг А.2 – Приклад API-методу створення запису.

```
router.post('/api/workoutplan', async (req, res, next) => {
  try {
    const data = validatePayload(req.body);
    const result = await service.create(data);
    res.status(201).json({ ok: true, data: result });
  } catch (error) {
    next(error);
  }
});
```

Лістинг А.3 – Сервіс виконання основної операції.

```
async function runMainAction(id, userId) {
  const record = await repository.findById(id);
  accessPolicy.assertCanUpdate(record, userId);
  const result = await workflow.execute('генерація тренувального плану', record);
  await auditLog.write({ id, userId, action: 'генерація тренувального плану' });
  return result;
}
```