**Міністерство освіти і науки України**

**Луцький національний технічний університет**

**Факультет комп'ютерних та інформаційних технологій**

**Кафедра інженерії програмного забезпечення**

# КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

## РОЗРОБКА API ДЛЯ МОБІЛЬНОГО ДОДАТКУ «COACHNOTE»

## API DEVELOPMENT FOR THE MOBILE APP «COACHNOTE»

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗс-31
**Мартинюк Ілля Андрійович**
_____
(підпис)

Керівник:
к.т.н., доцент
Повстяна Юлія Славомирівна
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«8» 06 2024 р.
к.т.н., доцент
Гарант освітньої програми:
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2024 року

**ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Н. Ліщина
«2» 02 2024 р.

# ЗАВДАННЯ
## НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

*Мартинюку Іллі Андрійовичу*
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: *розробка API для мобільного додатку «CoachNote»*

Керівник роботи: *к.т.н., доцент, Повстяна Юлія Славомирівна*
затверджені наказом закладу вищої освіти від «*30» грудня 2023 р.* № *477/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«5» червня 2024 р.*
3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
*Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.*

5. Перелік графічного матеріалу: *10 рисунків.*

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| *Аналіз предметної області* | *Повстяна Ю. С.* | | |
| *Специфікації вимог до розробленої системи* | *Повстяна Ю. С.* | | |
| *Розробка об'єкта проєктування* | *Повстяна Ю. С.* | | |
| *Нормоконтроль* | *Повстяна Ю. С.* | | |
| *Гарант ОП* | *Ліщина Н. М.* | | |
| *Показник запозичень тексту* | *5,1 %* | | |
| *Академічна доброчесність* | *Яцук А. А.* | | |

7. Дата видачі завдання *«2» лютого 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

| № З/п | Назва етапів кваліфікаційної роботи бакалавра | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | *Огляд літературних джерел по темі кваліфікаційної роботи бакалавра* | *07.02.2024 р.* | вик |
| 2 | *Аналіз проблеми розробки та впровадження об'єкту проектування* | *27.02.2024 р.* | вик |
| 3 | *Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання* | *12.03.2024 р.* | вик |
| 4 | *Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних* | *17.04.2024 р.* | вик |
| 5 | *Практична реалізація об'єкта проектування та розробка бази даних* | *18.05.2024 р.* | вик |
| 6 | *Тестування та налагодження об'єкта проектування* | *01.06.2024 р.* | вик |
| 7 | *Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру* | *05.06.2024 р.* | вик |
| | | | |
| | | | |

Здобувач вищої освіти _____ (підпис) *Мартинюк І. А.* (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) *Повстяна Ю. С.* (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

**Рецензія**

**на кваліфікаційну роботу бакалавра**

**Здобувач вищої освіти:**
Мартинюк Ілля Андрійович.

**Тема:** розробка API для мобільного додатку «CoachNote».

**Коротка характеристика кваліфікаційної роботи:**
робота присвячена розробці мобільного додатку для обліку клієнтів тренерів у спортивних залах. Вона включає проектування та реалізацію API, а також тестування і налаштування програмного інтерфейсу, що забезпечує ефективне управління тренувальним процесом.

**Самостійні розробки і пропозиції автора:**
API для додатку «CoachNote».

**Практичне значення роботи:**
практичне значення роботи полягає у створенні ефективного інструменту для тренерів спортивних залів, який полегшує облік клієнтів, управління тренуваннями та сприяє підвищенню якості наданих послуг.

**Загальний висновок:**
Роботу виконано у повному обсязі, у відповідності із завданням. Здобувач освіти Мартинюк Ілля Андрійович заслуговує позитивну оцінку.

Рецензент: Ліщук В.О. к.т.н. доц.
(прізвище, ім'я, по-батькові, посада)

**Рецензент кваліфікаційної роботи** _____
(підпис)

«7» 06 2024 р.

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

**ВІДГУК**
**КЕРІВНИКА НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА**

Здобувач вищої освіти **Мартинюк Ілля Андрійович**
(прізвище, ім'я, по батькові)

група ІПЗс-31

**Тема кваліфікаційної роботи:** розробка API для мобільного додатку «Coach Note».

**Керівник** к.т.н., доц. Повстяна Ю. С.

**Актуальність теми.**
Розробка мобільного додатку для обліку клієнтів тренерів у спортзалах є актуальною, оскільки вона відповідає зростаючому попиту на ефективні та зручні інструменти для управління відповідними процесами.

**Об'єкт дослідження.**
Інформаційні системи для обліку клієнтів у спортивних залах.

**Характеристика теоретичного рівня, наявності самостійних розробок і практичної значимості роботи.**
Робота виконана на сучасних теоретичних засадах розробки інформаційних систем, включаючи принципи побудови REST API, управління базами даних та використання кешування для оптимізації продуктивності. В рамках дослідження було самостійно розроблено архітектуру бази даних, API для взаємодії з мобільним додатком, а також механізми кешування та контейнеризації компонентів системи. Створене API для мобільного додатку «Coach Note» дозволить підвищити ефективність роботи тренерів, покращити управління клієнтською базою та оптимізувати процеси планування та проведення тренувань.

**Загальний висновок**
Роботу виконано у повному обсязі у відповідності до поставлених завдань, а здобувач Мартинюк І. А. заслуговує на оцінку «відмінно».

**Керівник** _____ **Повстяна Ю.С.**
(підпис) (прізвище, ім'я, по батькові)

«7» 06 2024 р.

# АНОТАЦІЯ

Мартинюк І. А. Розробка API для мобільного додатку «CoachNote». Рукопис.

Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2024.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків і пропозицій, списку використаних джерел та додатків.

У першому розділі здійснено аналіз актуальності розробки API для додатку обліку клієнтів тренерів і сформульовано завдання. В другому розділі визначено вимоги до програмного інтерфейсу, а також засоби, методи і алгоритми розробки. У третьому розділі розроблено API для додатку «Coach Note» . У висновках узагальнено інформацію, відображену у попередніх частинах. Результати розробки демонструють можливості автоматизації обліку та оптимізації процесів в спортзалах.

Ключові слова: Python, API, БД, Flask, SqlAlchemy.

# ABSTRACT

Martyniuk I. A. API development for mobile application "Coach Note". Manuscript.

Bachelor’s qualifying thesis of the educational program "Software Engineering". Lutsk National Technical University. Lutsk, 2024.

The bachelor's qualifying thesis consists of an introduction, three sections, conclusions and proposals, a list of used sources and annexes.

In the first section, an analysis of the relevance of the development of an API for the accounting application of trainers clients was carried out and the task was formulated. The second section defines the requirements for the program interface, as well as means, methods and development algorithms. In the third chapter, API of "Coach Note" was developed and tested. The conclusions summarize the information presented in the previous parts. The development results demonstrate the possibilities of automatization and optimization of processes in gyms.

Keywords: Python, API, DB, Flask, SqlAlchemy.

**ЗМІСТ**

# ВСТУП

Актуальність роботи. Наші часи характеризуються зростаючим інтересом до здорового способу життя, що, однозначно, призводить до збільшення кількості клієнтів в спортивних залах та фітнес-клубах. Це, в свою чергу, створює нові виклики для менеджменту у таких закладах, особливо для обліку клієнтів та організації тренувального процесу. У зв'язку з цим виникає необхідність у впровадженні ефективних інформаційних систем, які допомагають тренерам вести облік своїх клієнтів, відстежувати їх прогрес та оптимізувати графіки тренувань.

На сьогоднішній день існує багато програмних рішень для управління фітнес-клубами. Однак багато з них мають обмежені функціональні можливості або ж високий поріг входження через складність у використанні. Зокрема, багато систем не надають можливості зручного мобільного доступу або не забезпечують належного рівня інтеграції з іншими системами управління. Це створює суттєві прогалини в ефективності використання таких програм і підкреслює необхідність розробки нових, більш адаптивних та функціональних рішень.

Світові тенденції свідчать про активне впровадження мобільних технологій у різних сферах життя, включаючи спорт та фітнес. Мобільні додатки стають невід'ємною частиною роботи багатьох тренерів та клієнтів, забезпечуючи швидкий доступ до необхідної інформації та спрощуючи комунікацію між ними. В таких умовах актуальність розробки API для мобільного додатку для обліку клієнтів тренерів у спортзалах стає очевидною, оскільки це дозволяє розробляти відповідні телефонні програми. Які, у свою чергу, зможуть підвищити ефективність роботи тренерів та покращити досвід клієнтів.

Метою даної роботи є створення API для мобільного додатку, який би дозволяв тренерам ефективно вести облік своїх клієнтів, відстежувати їх прогрес, планувати тренування та підтримувати комунікацію з ними. Основна увага

приділяється розробці, яка забезпечуватиме надійну взаємодію мобільного додатку з базою даних, зберігаючи при цьому високу продуктивність та безпеку.

Поставлені завдання для даної роботи:

- проаналізувати актуальність проекту;
- побудувати UML-діаграми;
- спроектувати базу даних в dbdiagram.io;
- спроектувати структуру API;
- реалізувати API;
- протестувати API за допомогою Postman.

Об’єктом роботи виступає API для мобільного додатку «CoachNote».

Предметом кваліфікаційної роботи є розробка API для мобільного додатку «CoachNote».

Область застосування розробленого додатку включає фітнес-клуби та спортивні зали, де тренери потребують зручного інструменту для управління своєю діяльністю та взаємодії з клієнтами.

Ця робота також розглядає взаємозв’язок з іншими подібними рішеннями та дослідженнями в цій галузі. Використання сучасних технологій, таких як Flask для розробки API, SQLAlchemy для взаємодії з базою даних, PostgreSQL як надійної системи управління базами даних та Redis для кешування та збереження тимчасових даних, дозволяє досягти високих результатів. Крім того, впровадження Docker Compose для контейнеризації компонентів системи сприяє більшій гнучкості та спрощує процес розгортання додатку.

Таким чином, розробка даного програмного інтерфейсу є своєчасною та актуальною відповіддю на сучасні вимоги ринку. Вона спрямована на заповнення існуючих прогалин у програмних рішеннях для управління фітнес-клубами, надаючи тренерам зручний та ефективний інструмент для їхньої роботи. Це дослідження не лише впроваджує нові технологічні рішення, але й робить значний внесок у розвиток інформаційних систем у сфері спорту та фітнесу, створюючи нові можливості для покращення якості послуг та задоволення потреб клієнтів.

# РОЗДІЛ 1
# АНАЛІЗ API ДОДАТКІВ ДЛЯ ТРЕНЕРІВ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

## 1.1 Аналіз сучасного стану проблеми

У динамічному світі сучасного спорту, для тренерів у спортзалах ефективне управління тренуваннями та клієнтами стає ключовим фактором, що визначає їх успіх. Проте, незважаючи на зростаючу популярність занять спортом та постійні зміни в очікуваннях клієнтів, багато тренерів все ще покладаються на застарілі методи обліку клієнтів та планування тренувань. Ці методи, зазвичай засновані на паперових нотатках, електронних таблицях або простих системах бронювання, часто не відповідають сучасним вимогам спортивного ринку.

Традиційні методи обліку клієнтів та планування тренувань можуть бути дуже громіздкими та незручними у використанні. Тренерам доводиться витрачати багато часу на рутинні завдання, такі як ведення записів, оновлення інформації та координація з клієнтами. Це відволікає їх від основної роботи – надання персоналізованих тренувань та якісного обслуговування клієнтів.

Паперові нотатки та електронні таблиці часто схильні до помилок, що може призвести до плутанини та розчарування як для тренерів, так і для клієнтів. Неточні записи можуть призвести до пропущених тренувань, їх оплат, неправильного планування та незбалансованих програм тренувань.

Звичні методи не пропонують гнучкості, необхідної для задоволення потреб сучасних клієнтів. Тренерам складно адаптувати програми тренувань під мінливі графіки та індивідуальні цілі клієнтів. Ці методи методи не надають тренерам можливості відстежувати прогрес клієнтів та аналізувати дані про тренування, отримувати статистику, та формувати графіки по ній. Це обмежує їхню здатність оцінювати ефективність своїх програм та вносити необхідні корективи.

В наслідок – отримуємо незадоволеність клієнтів, та складності в роботі. Клієнти, які стикаються з плутаниною, неточними записами та відсутністю

гнучкості, ймовірно, будуть розчаровані та шукатимуть альтернативні варіанти. Неефективне управління тренуваннями та клієнтами може призвести до втрати цінних клієнтів, що негативно впливає на дохід та репутацію тренера. В той час тренери, які витрачають багато часу на рутинні завдання та стикаються з невдоволеними клієнтами, ризикують професійним вигоранням, втратою можливого прибутку, та втратою часу в наслідок нераціонального його використання на рутинні справи.

Щоб успішно конкурувати на сучасному спортивному ринку, тренерам у спортзалах необхідні інноваційні рішення для управління тренуваннями та клієнтами. Тренери повинні мати можливість легко вести облік клієнтів, планувати тренування та спілкуватися з клієнтами за допомогою єдиної платформи. Системи управління повинні забезпечувати точні та актуальні дані про клієнтів та тренування, щоб уникнути плутанини та розчарування. Тренери повинні мати можливість легко адаптувати програми тренувань під індивідуальні потреби та цілі клієнтів. Та врешті-решт – системи управління повинні надавати тренерам доступ до даних про прогрес клієнтів та тренування, щоб вони могли оцінювати ефективність та вносити необхідні корективи.

Розробка інноваційного рішення для управління тренуваннями та клієнтами може допомогти тренерам у спортзалах значно покращити свою ефективність, задоволеність клієнтів та загальний успіх і продуктивність.

На ринку уже існують певні додатки, які містять потрібний функціонал, кожен з яких має свої сильні та слабкі сторони.

MyFitnessPal – це універсальний додаток для здоров’я та фітнесу, який використовується мільйонами людей у всьому світі. Він пропонує широкий спектр функцій (рис. 1.1), які можуть бути корисними для тренерів. Дозволяє записувати та відстежувати свої тренування, включаючи кардіо, силові вправи та йогу. Має функціонал щоб заповнювати інформацію про те що ви їсте та п’єте, щоб відстежувати споживання калорій, макронутрієнтів та мікронутрієнтів. За допомогою нього ви відстежуєте свою вагу, розміри та інші показники прогресу, щоб бачити, як ви розвиваєтесь.

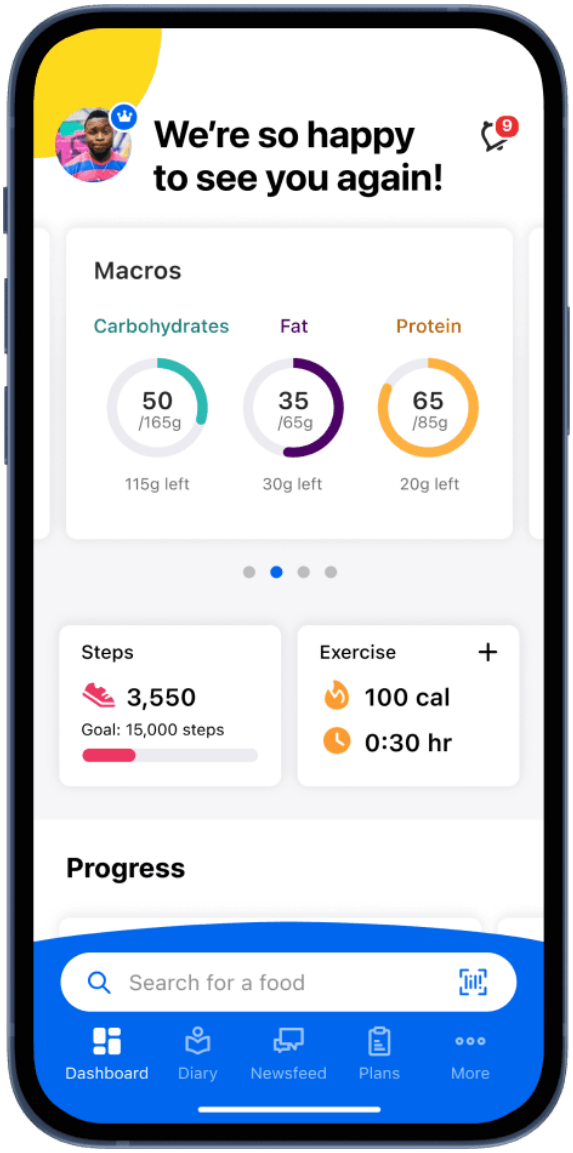


Рисунок 1.1 – Сторінка «Dashboard» в додатку MyFitnessPal [1]

Додаток має доступ до великої бібліотеки вправ з інструкціями та відео, доступ до великої бібліотеки продуктів харчування з інформацією про харчову цінність (рис. 1.2), що дуже корисно для клієнтів. Також MyFitnessPal має велике ком'юніті користувачів, що тебе підтримуватиме та вмотивовуватиме.

Спорт-додаток дуже популярний і широко використовується, що, однозначно, є одною з найбільших його переваг. Широкий спектр функцій не залишить користувачів байдужими, та не змусить обходитись власними методами при специфічних потребах. Велика бібліотека вправ і продуктів харчування завжди дасть вибрати те, що потрібно саме вам. Також серед переваг є наявність безкоштовна версія.

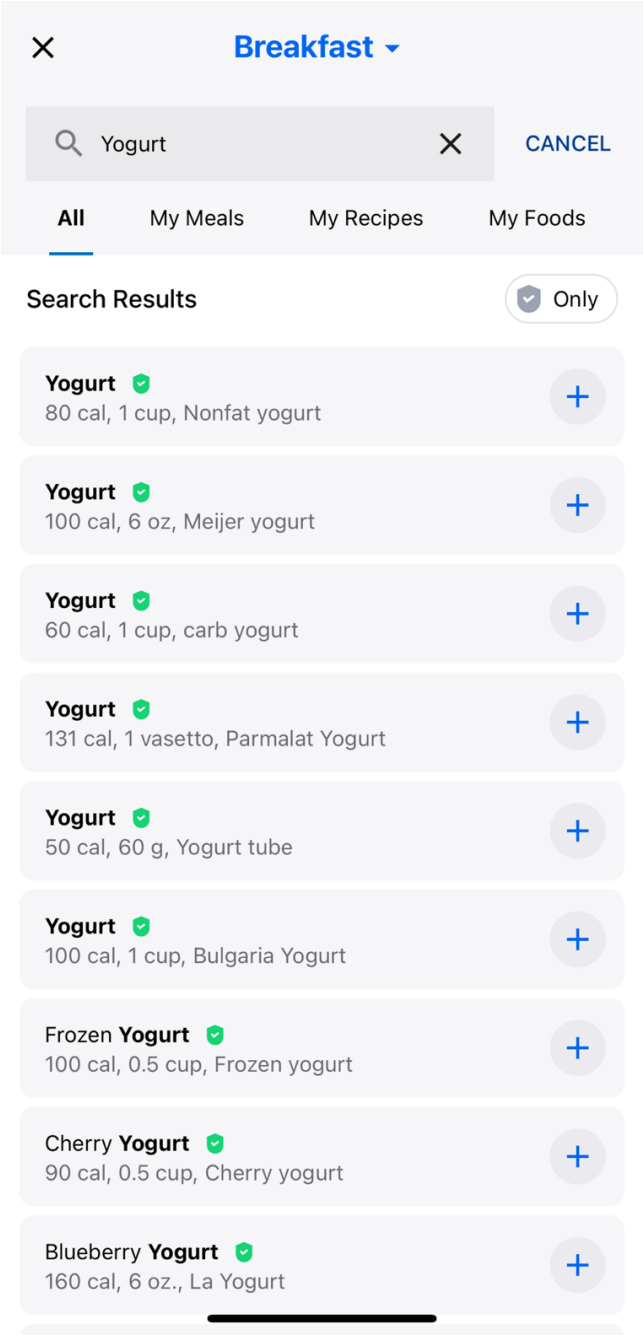


Рисунок 1.2 – Приклад списку продуктів в додатку MyFitnessPal [1]

Серед його недоліків є складність функціоналу через його об’ємність. Це може викликати складнощі у користуванні. Особливо для новачків. Також додаток не зосереджений на потребах тренерів, таких як планування тренувань для клієнтів та відстеження їхнього прогресу та оплат за заняття.

Trainerize – це мобільний додаток та веб-платформа, спеціально розроблені для персональних тренерів та фітнес-студій, щоб допомагати їм керувати своїм

бізнесом та спілкуватися з клієнтами. Він пропонує широкий спектр функцій, які роблять його цінним інструментом для будь-якого фітнес-професіонала.

Тренери створюють власні плани тренувань (рис. 1.3) з відео, інструкціями та варіантами налаштування. Вони можуть продавати свої плани занять клієнтам безпосередньо через додаток. Також є можливість відстежувати відвідування тренувань, виконання вправ, вагу та інші показники прогресу клієнтів. Можна надсилати повідомлення клієнтам, відповідати на запитання та спілкуватися з ними в чаті. Наявний календар доступності, щоб клієнти могли бронювати тренування онлайн. Тренери отримують доступ до звітів про доходи, клієнтів та використання, щоб відстежувати свій прогрес і приймати обґрунтовані рішення.

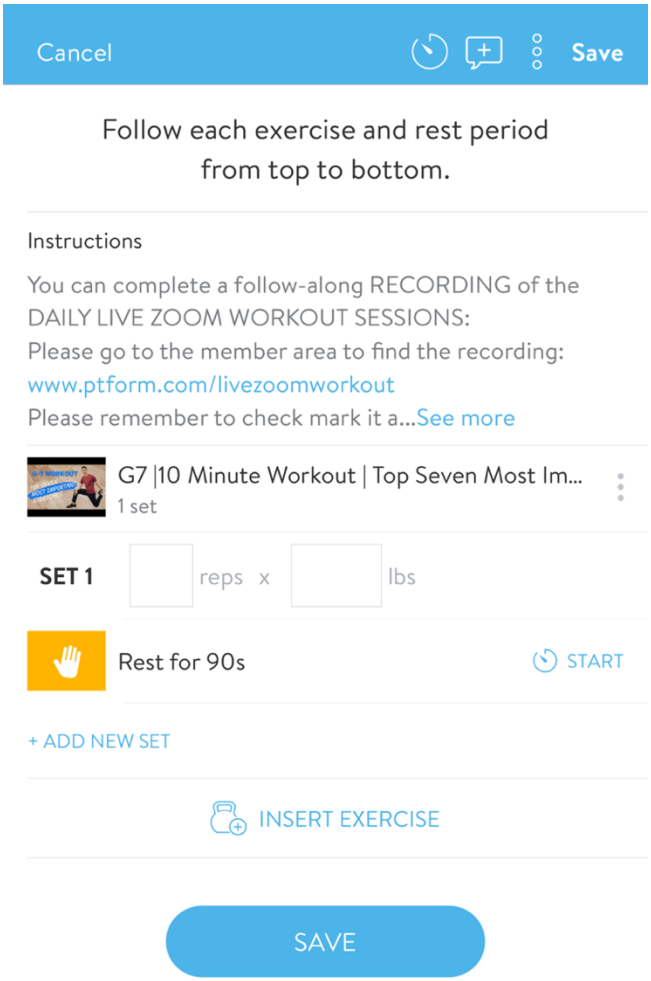


Рисунок 1.3 – Приклад створення заняття в додатку Trainerize [2]

Зі сторони клієнтів є можливість переглядати та виконувати плани тренувань, створені вашим тренером (рис. 1.4), відстежувати свої тренування, виконання вправ, вагу та інші показники.

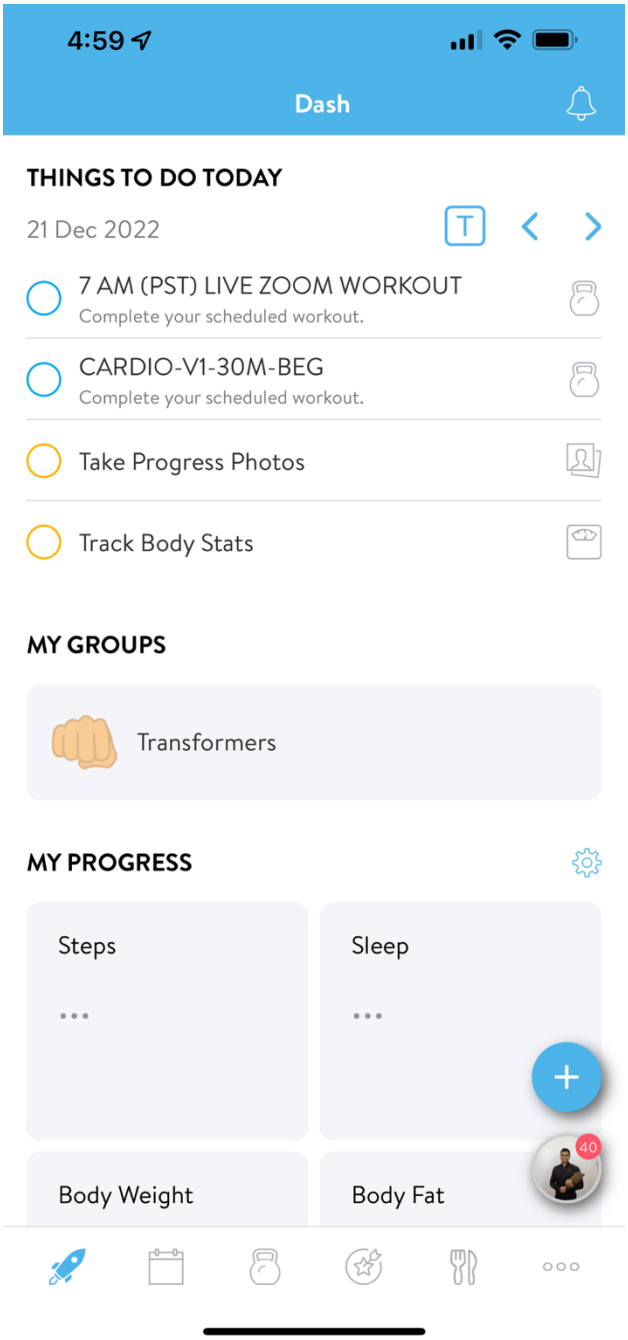


Рисунок 1.4 – Приклад списку занять в додатку Trainerize [2]

Також є можливість листування з тренером, що значно покращує комунікацію.

Додаток Trainerize дійсно збільшує ефективність тренувань. Тренери можуть економити час на адміністративних завданнях і зосередитися на роботі з клієнтами. Клієнти можуть легко відстежувати свій прогрес, спілкуватися зі своїми тренерами та бронювати тренування.

Тренери можуть продавати онлайн-тренування та приймати платежі безпосередньо через додаток. Це дуже зручний і безпечний функціонал як для тренерів так і для їх клієнтів.

Trainerize підходить як для персональних тренерів, так і для фітнес-студій з великою кількістю клієнтів. І ще серед переваг те, що додаток має безкоштовний план, хоч і з обмеженим функціоналом.

Хоча існуючі мобільні додатки для тренерів, такі як MyFitnessPal, та Trainerize пропонують широкий спектр функцій, все ж таки є деякі невирішені бізнес-проблеми. Багато існуючих додатків пропонують загальні плани тренувань та функції відстеження.

Деякі додатки ускладнюють спілкування між тренерами та клієнтами, а також співпрацю між кількома тренерами, які працюють з одним клієнтом. Потрібний додаток, який покращить комунікацію, пропонуючи функції чату, обміну нотатками та спільного календаря, а також дозволяючи тренерам ділитися даними про прогрес клієнта та надавати спільні відгуки.

Тож робота є надзвичайно актуальною в контексті сучасних тенденцій у фітнес-індустрії.

Аналіз конкурентів, таких як MyFitnessPal і Trainerize, виявив, що багато існуючих додатків зосереджуються переважно на відстеженні тренувань та прогресу. Але їхні програмні рішення не пропонують достатньої мотивації та підтримки, необхідної для досягнення клієнтами своїх цілей, не дають достатнього рівня контролю процесів.

Даний додаток може вирішити цю проблему, впроваджуючи функції, які сприяють встановленню цілей, відстеженню навичок, спрощення нагляду тренерів. Ці елементи допоможуть клієнтам залишатися мотивованими та

відповідальними, що є важливим аспектом у досягненні довгострокових результатів.

Зважаючи на вищезазначені невирішені бізнес-проблеми, існує значний потенціал для створення нового мобільного додатку для тренерів, який запропонує більш персоналізований, інтегрований, мобільний та підтримувальний досвід як для тренерів, так і для клієнтів. Додаток відрізнятиметься від конкурентів завдяки унікальним функціям, інноваційному дизайну та чіткій цінності для користувачів. Зокрема, функції встановлення цілей і відстеження навичок дозволять тренерам та клієнтам спільно працювати над досягненням конкретних фітнес-результатів, створюючи індивідуальні плани тренувань, що відповідають особистим потребам та рівню фізичної підготовки клієнтів.

У результаті проведеного дослідження та розробки мобільного додатку, було повністю виконано завдання на кваліфікаційну роботу бакалавра. Отримані результати демонструють високі кількісні та якісні показники, включаючи швидкість обробки запитів, надійність системи та її масштабованість. Робота має тісний зв'язок з науково-дослідними розробками кафедр університету та інших організацій, що займаються питаннями розробки інформаційних систем та мобільних додатків. Отримані результати можуть знайти відображення у наукових статтях та патентах, а також можуть бути використані для подальших досліджень і вдосконалення існуючих рішень у цій галузі.

Подальші шляхи вдосконалення розроблених у роботі рішень можуть включати інтеграцію з іншими системами управління, розширення функціоналу API для задоволення потреб ширшого кола користувачів, а також оптимізацію алгоритмів обробки даних для підвищення продуктивності. Додаткові дослідження можуть бути спрямовані на впровадження нових технологій, таких як машинне навчання для аналізу прогресу клієнтів, або на розробку модулів для персоналізації тренувальних програм. Таким чином, дана робота робить значний внесок у розвиток інформаційних систем для фітнес-індустрії і має значний потенціал для подальшого розвитку.

## 1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Тож було вирішено створити додаток Coach Note, розробити API для нього. Його реалізацію розбиваємо на такі кроки:

– проаналізувати актуальність проекту;
– побудувати UML-діаграми;
– спроектувати базу даних в dbdiagram.io;
– спроектувати структуру API;
– реалізувати API;
– протестувати API за допомогою Postman.

## Висновки до розділу 1

У цьому розділі було проведено пошук та дослідження існуючих мобільних додатків, їх API для тренерів у спортзалах. Було проаналізовано їхні функції, переваги та недоліки.

Також було виявлено деякі невирішені бізнес-проблеми, які існують у цій сфері. Майбутній додаток може вирішити цю проблему, пропонуючи функції, такі як встановлення цілей, відстеження прогресу, аналіз статистики.

На основі проведеного аналізу було визначено, що існує значний потенціал для створення API для мобільного додатку для тренерів, яке пропонуватиме більш персоналізований, інтегрований, мобільний та підтримувальний досвід для тренерів та клієнтів.

Було сформульовано конкретні цілі атестаційної роботи, які полягають у розробці API мобільного додатку для «CoachNote» з певними функціональними можливостями.

Вважаю, що виконання цих цілей дозволить створити цінний інструмент для тренерів у спортзалах, який допоможе їм покращити свою роботу та краще обслуговувати своїх клієнтів.

# РОЗДІЛ 2
# СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

## 2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Першим кроком у процесі виявлення та аналізу вимог є визначення зацікавлених сторін. Зацікавлені сторони – це люди або групи людей, які мають певний інтерес до мобільного додатку. У даному проекті до зацікавлених сторін належать тренери спортзалів та клієнти тренерів.

Тренери спортзалів – це основні користувачі мобільного додатку, які використовуватимуть його для створення та ведення планів тренувань, відстеження прогресу клієнтів, оплат за заняття, спілкування з клієнтами та інших завдань.

Клієнти тренерів – це непрямі користувачі додатку, але вони відчують на собі користь від використання його тренерами. Тренери матимуть більше вільного часу та працюватимуть більш ефективно внаслідок автоматизації певних процесів.

Після того, як зацікавлені сторони визначені, можна розпочати збір вимог. Існує декілька способів збору вимог, включаючи інтерв’ю, опитування, аналіз даних та спостереження.

Інтерв’ю – це хороший спосіб отримати детальну інформацію про потреби та очікування зацікавлених сторін. Його мінус у тому, що ви не зможете провести його для багатьох потенційних клієнтів, так як це дуже затратно по часу. Тому варто поєднувати його із опитуваннями, щоб в загальному розуміти чи думка більш широкої вибірки людей співпадає із думкою тих, з якими проводили інтерв’ю.

Можна проаналізувати існуючі дані, такі як записи про тренування або відгуки клієнтів, щоб отримати інформацію про вимоги. Такий спосіб хороший тим, що ви, як спеціаліст, можете побачити додаткові можливості для автоматизації, які не завжди помітить користувач. Те ж саме стосується способу спостереження.

Спостереження за тренерами та клієнтами під час використання ними існуючих систем теж може допомогти виявити нові вимоги.

Тож, скориставшись цими методами, я описав такі функціональні вимоги:

– реєстрація;
– логінізація;
– створення та редагування профілю;
– створення та редагування планів тренувань;
– відстеження прогресу;
– ведення та планування тренувань;
– система оплат.

Реєстрація – користувачі повинні мати можливість реєструватися за допомогою телефону, без використання логіну, електронної пошти, паролю.

При реєстрації на телефон приходить код підтвердження (рис. 2.1). Після його введення користувача авторизовує.

Логінізація – аналогічно реєстрації, за допомогою номеру телефону та коду підтвердження. Це дозволяє не тримати в пам’яті пароль, який можна забути, або, що ще гірше, його можуть отримати зловмисники і скористатися ним. Також повинна бути підтримка авторизації через сервіс Google, Facebook або AppleID, що теж є швидким та безпечним методом авторизації.

Створення та редагування профілю – тренери повинні мати можливість додавати інформацію про себе.

Включаючи біографію, досвід роботи, спеціалізацію, сертифікації та контактну інформацію. Також повинна бути можливість завантаження фотографії профілю.

Створення та редагування планів тренувань – користувачі можуть створювати індивідуальні та групові плани тренувань. Додавати в них вправи, встановлювати кількість повторень для кожної, кількість підходів, вагу, час та інші параметри.

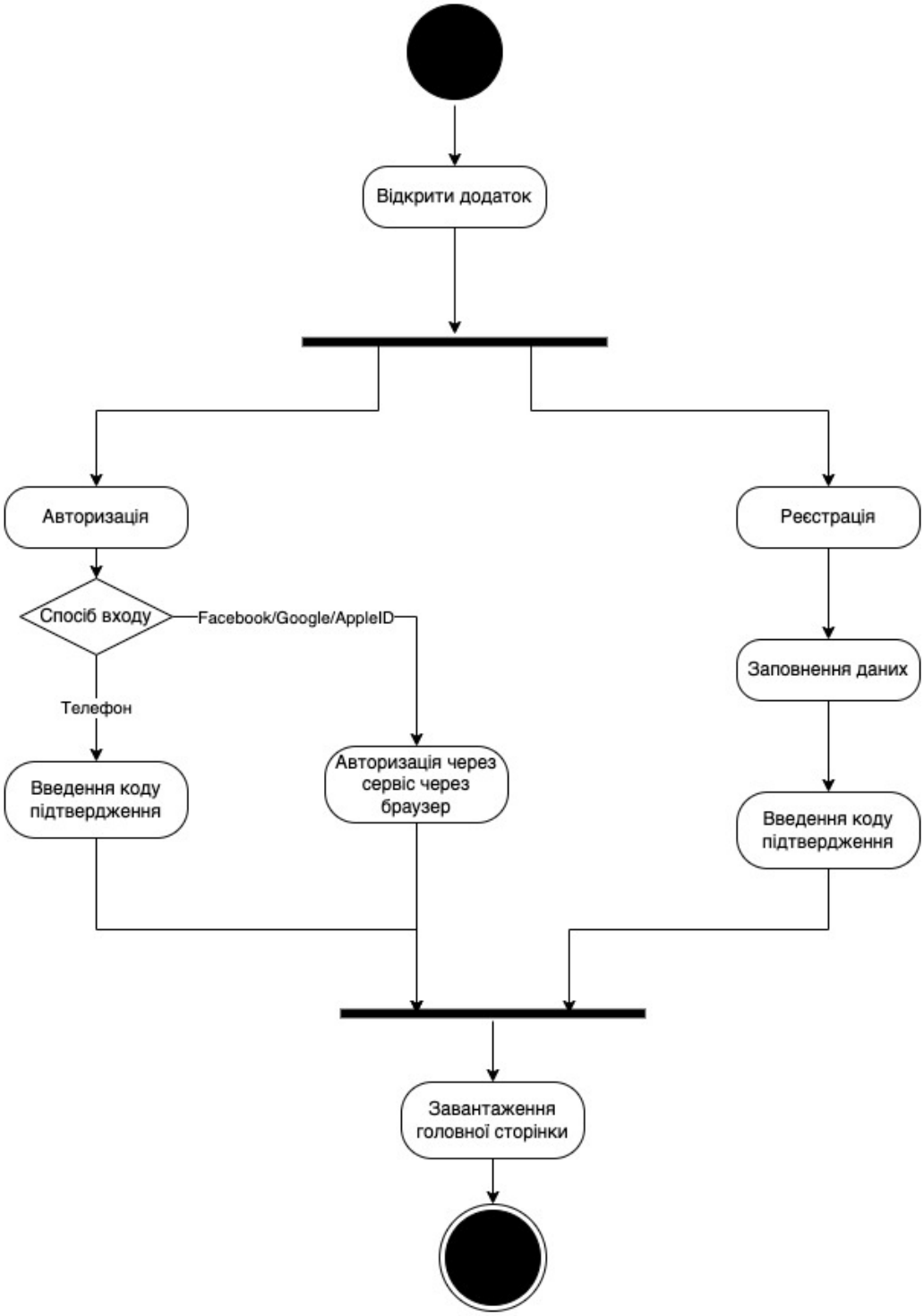


Рисунок 2.1 – UML-діаграма модулю авторизації та реєстрації

Відстеження прогресу – відображення статистики прогресу клієнтів. Для цього збирати такі показники як вага, заміри тіла, максимальні показники виконання вправ, фотографії тіла. Відповідно до прогресу відображати графіки та діаграми.

Ведення та планування тренувань – тренери повинні мати можливість переглядати та редагувати свій календар тренувань, додавати в нього нові тренування (рис. 2.2). Відповідно треба розробити опціональну (тобто вмикати лише за бажанням користувача) систему нагадувань про тренування.

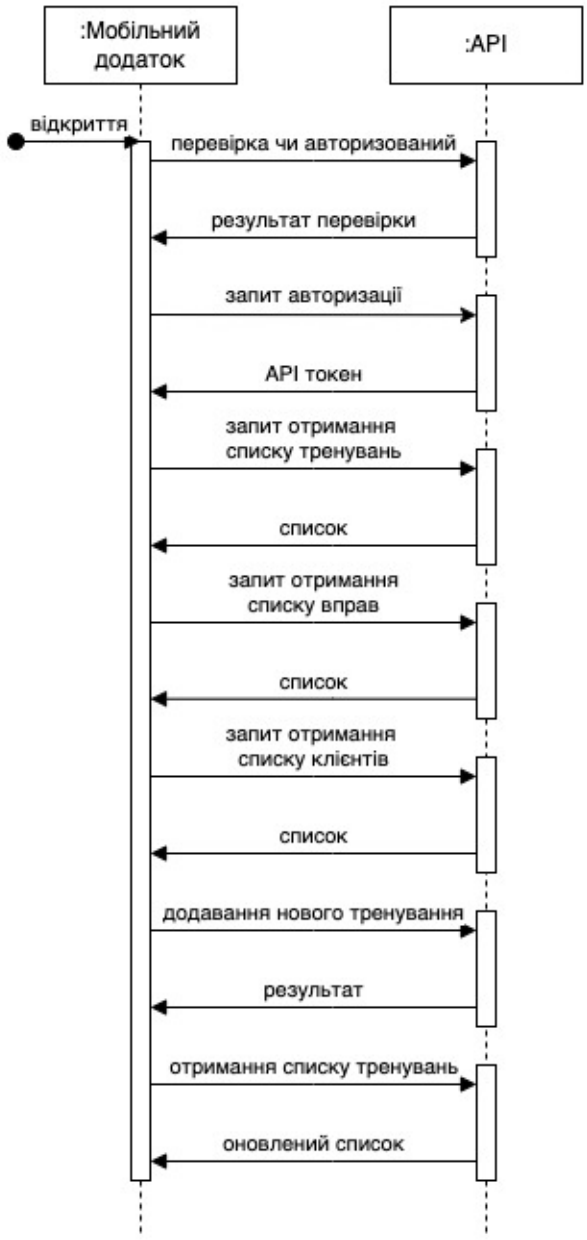


Рисунок 2.2 – UML-діаграма модуля тренувань

Система оплат – розробити функціонал для внесення даних про оплати клієнтів. Також налаштувати сповіщення клієнтам про необхідність оплати за заняття, якщо закінчується абонемент.

## 2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для розробки даного проекту існує широкий спектр інструментів і технологій. Основні вимоги до такого API включають надійність, масштабованість, безпеку, швидкість у розробці та високу продуктивність. Розглянемо основні технології, які підходять для цих цілей, їх переваги та недоліки.

Я прийняв рішення використати для розробки такі інструменти:

– Python;
– Flask;
– SQLAlchemy;
– PostgreSQL;
– Redis;
– Docker-Compose.

Python – це одна із найбільш популярних мов програмування в світі завдяки своїй простоті у використанні, багатій екосистемі бібліотек та фреймворків, а також активній спільноті розробників. Python дозволяє писати чистий і читабельний код, що спрощує розробку та підтримку програмного забезпечення. Також, існує багато бібліотек для різних задач, від обробки даних до веб-розробки. Для веб-розробки має потужні веб-фреймворки, такі як Flask [3], Django/DRF (Django Rest Framework) [4] та FastAPI [5], які дозволяють швидко створювати веб-додатки та API.

Flask – це легкий веб-фреймворк для Python, який дозволяє швидко розробляти веб-додатки і API. Flask надає лише базові функції, дозволяючи розробникам вибирати додаткові бібліотеки та модулі за потребою. Він не перевантажує проект непотрібними розширеннями, що йдуть разом з фреймворком як, для прикладу, у Django. Завдяки своїй структурі, Flask дозволяє легко налаштовувати та розширювати функціональність додатку. Він доволі швидкий у виконанні, хоча в порівнянні з асинхронним FastAPI він повільніший.

Але швидший в розробці та тестуванні, так як асинхронний код дебажити набагато тяжче за синхронний. Тому Flask являється хорошим оптимальним рішенням для реалізації вимог, описаних вище.

SQLAlchemy – це потужний ORM (Object-Relational Mapping) для Python, який забезпечує абстракцію для роботи з базами даних. Він підтримує роботу з багатьма СУБД, має інтуїтивний синтаксис, дозволяє легко створювати і виконувати SQL-запити та дозволяє моделювати складні зв’язки між даними, формувати складні запити. Вбудований у Django ORM сильно уступає SQLAlchemy по функціональності та продуктивності.

PostgreSQL – це потужна об’єктно-реляційна СУБД з відкритим вихідним кодом [6]. Підтримується багатьма ORM, має можливість формувати складні запити, транзакції, індекси і розширення. PostgreSQL добре масштабується і може обробляти великі обсяги даних, відома своєю стабільністю і надійністю. В порівнянні з MySQL – має набагато більше функціоналу, частіше оновлюється, та має велике ком’юніті, що дозволяє швидко вирішувати проблеми, що виникають при користуванні.

Redis – це in-memory ключ-значення база даних, яка використовується для кешування, управління сесіями та токенами і зберігання інших тимчасових даних. Ця БД фактично не має конкурентів серед інших для такого застосування. Redis працює в оперативній пам’яті, що забезпечує високу швидкість операцій читання та запису. Він підтримує різні структури даних, такі як строки, списки, множини, хеші тощо. Також Redis простий у встановленні та налаштуванні, забезпечує стабільну роботу.

Docker-Compose – це інструмент для автоматизації розгортання багатоконтейнерних Docker-додатків. Docker-Compose дозволяє визначати і запускати багатоконтейнерні додатки з одного конфігураційного файлу. За допомогою команд Docker-Compose легко масштабувати додатки. Також він забезпечує відтворюваність середовища розробки та спрощує його взаємодію з базами даних та іншими сервісами.

## Висновки до розділу 2

У цьому розділі було розроблено UML-схему функціоналу та проведено аналіз різних технологій, які могли б бути використані для розробки API мобільного додатку для тренерів спортзалів. Після ретельного аналізу було обрано наступні технології: Python, Flask, SQLAlchemy, Postgres, Redis та Docker-Compose. Вибір цих технологій ґрунтувався на їх простоті використання, продуктивності, масштабованості та гнучкості.

Вибрані технології дозволили створити API, яке є надійним, масштабованим та простим у використанні. API відповідає всім вимогам і може бути легко інтегрований з мобільним додатком. В даному випадку було обрано технології, які, на мою думку, найкраще відповідають вимогам проекту.

Однак інші технології також могли б бути використані для розробки API. У майбутньому може знадобитися переглянути вибір технологій, якщо вимоги до проекту зміняться.

# РОЗДІЛ 3
# РОЗРОБКА API ДЛЯ МОБІЛЬНОГО ДОДАТКУ «COACHNOTE»

## 3.1 Розробка бази даних

Перше до чого я приступив на етапі розробки – проектування БД. Для розробки схеми я вирішив використати dbdiagram – онлайн-інструмент для візуалізації та проектування баз даних [7]. Він надає зручний графічний інтерфейс для візуалізації та редагування структури бази даних. Після завершення проєктування схема була експортована у вигляді SQL-коду для створення таблиць у базі даних PostgreSQL.

Для підключення до серверу Postgres та виконання маніпуляцій з базою даних я використав DataGrip – багатофункціональний IDE для розробки та роботи з базами даних, який підтримує широкий спектр СУБД, включаючи Oracle, MySQL, PostgreSQL, SQL Server, Sybase ASE, DB2, Hive та MongoDB. Цей IDE дозволяє писати та редагувати SQL-запити за допомогою зручного візуального редактора, який підсвічує синтаксис, пропонує автозаповнення та перевірку помилок. Також в ньому можна змінювати, створювати та переглядати дані без використання коду, а просто використовуючи інтерфейс.

Проектування БД – це перший крок, так як API в основному це просто CRUD (Create Read Update Delete) операції для існуючих об’єктів – таблиць БД [8]. Тому для початку потрібно зрозуміти які об’єкти ми матимемо, які поля у них будуть, яких типів та які зв’язки між ними.

Для реалізації описаних в попередньому розділі функціональних вимог API мобільного додатку було спроектовано схему бази даних (рис. 3.1). Вона розроблялась так, щоб була можливість в подальшому розширювати або модифікувати її для включення додаткових функціональних вимог.

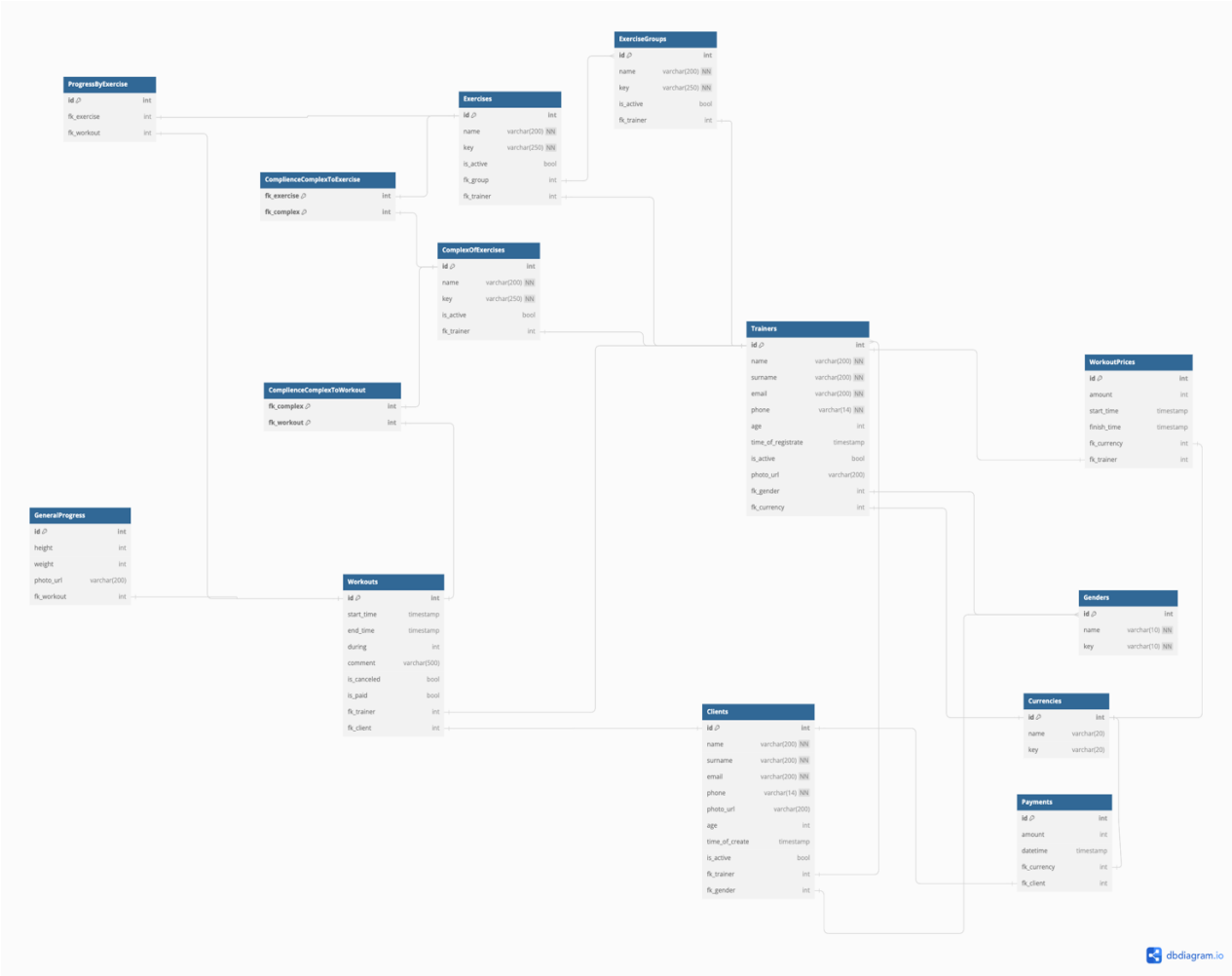


Рисунок 3.1 – Схема бази даних API

БД включає в себе такі таблиці:

– Clients (клієнти);
– Trainers (тренери);
– Genders (статі);
– GeneralProgress (загальний прогрес);
– ProgressByExercise (прогрес за вправою);
– Exercises (вправи);
– ExerciseGroups (групи вправ);
– ComplexOfExercises (комплекси вправ);
– ComplienceComplexToExercise (відповідність комплекс-вправа);
– Workouts (тренування);

– ComplienceComplexToWorkout (відповідність комплекс-тренування);

– Payments (платежі);

– WorkoutPrices (ціни тренувань);

– Currencies (валюти).

### 3.2 Практична реалізація API

Створення Flask проекту починається з організації файлової структури, що є основою для подальшої розробки та підтримки додатку. Першим кроком є створення нової директорії, яка міститиме всі файли та папки вашого Flask проекту. Директорія може бути створена будь-яким зручним способом: через графічний інтерфейс або командний рядок.

У створеній директорії створюється файл app.py, який буде основним файлом вашого Flask додатку. Цей файл містить початковий код для налаштування та запуску додатку (ліст. 3.1). У файлі app.py описується функція-фабрика, яка створює і налаштовує Flask додаток. Функція-фабрика дозволяє створювати інстанси додатку з різними конфігураціями, що зручно для розробки, тестування та продакшн середовищ.

Лістинг 3.1 – Ініціалізація додатку

```
from flask import Flask
from flask_login import LoginManager
from flask_marshmallow import Marshmallow
from flask_sqlalchemy import SQLAlchemy

from config import Config
from models import Base


db = SQLAlchemy(model_class=Base)

login_manager = LoginManager()

ma = Marshmallow()
```

```
def create_app():
    app = Flask(__name__)
    app.config.from_object(Config)

    db.init_app(app)

    login_manager.init_app(app)

    ma.init_app(app)

    from blueprints.auth import auth_blueprint

    app.register_blueprint(auth_blueprint)

    return app
```

Кінець лістингу 3.1

В даному випадку ми ініціалізовуємо в ньому з'єднання з БД через ORM SqlAlchemy, LoginManager для Flask-a, серіалізатор Marshmallow та реєструємо модуль (блупрінт) авторизації. Після цього ми описуємо файл server.py (ліст. 3.2) і вже можемо запускати Flask-сервер.

Лістинг 3.2 – Запуск Flask-сервера

```
from app import create_app

app = create_app()

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, use_reloader=False)
```

Кінець лістингу 3.2

Але без жодного описаного роута API не матиме жодного значення. На запити буде повертати 404 помилку. Тому додаємо перший роут авторизації в блупрінт, який ми вже зареєстрували в app.py файлі. Авторизація складатиметься з двох частин: запит на авторизацію (ліст. 3.3) та запит з кодом підтвердження (ліст. 3.4).

Лістинг 3.3 – Запит на авторизацію

```
@auth_blueprint.route("/sign_in", methods=["POST"])
def sign_in():
    form = SignInForm()

    if form.validate_on_submit():
        confirm_code = send_confirm_code(form.phone.data)

        redis.set(
            f"{form.phone.data}_login",
            value=pickle.dumps(
                {
                    "sign_in_data": {
                        "phone": form.phone.data,
                    },
                    "confirm_code": confirm_code,
                }
            ),
            ex=Config.SECONDS_IN_MINUTE * 10
        )

        return jsonify(success=True), 201

    return jsonify(success=False, errors=form.errors), 422
```

Кінець лістингу 3.3

Лістинг 3.4 – Підтвердження авторизації

```
@auth_blueprint.route("/confirm_sign_in", methods=["POST"])
def confirm_sign_in():
    form = ConfirmCodeForm()

    if form.validate_on_submit():
        sign_in_data = redis.get(f"{form.phone.data}_login")

        if not sign_in_data:
            return jsonify(
                message="Час на підтвердження входу вийшов",
                success=False
            )

        sign_in_data = pickle.loads(sign_in_data)

        if sign_in_data.get("confirm_code") != form.confirm_code.data:
            return jsonify(
                message="Не вірний код підтвердження",
                success=False
            )

        trainer_id = get_id_by_phone(form.phone.data)
```

```
        return jsonify(
            success=True,
            token=generate_token(trainer_id)
        ), 201

    return jsonify(success=False, errors=form.errors), 422
```

Кінець лістингу 3.4

Користувач отримує код підтвердження (наприклад, через SMS) і вводить його в інтерфейсі додатку. Код підтвердження передається до сервера для перевірки. Сервер перевіряє код підтвердження на валідність. Якщо код підтвердження є правильним, сервер генерує API-ключ для користувача. Після успішної перевірки коду підтвердження, сервер повертає згенерований API-ключ у відповідь на запит. Користувач отримує цей ключ і використовує його для авторизації при подальших запитах до захищених маршрутів API.

Користувач додає отриманий API-ключ до заголовків (headers) своїх запитів для доступу до login_required роутів. Це може бути зроблено за допомогою заголовку Authorization або іншого визначеного вами заголовку. Таким чином користувачам не потрібно зберігати або вводити паролі для кожного запиту. Замість цього вони використовують API-ключі, що зменшує ризик витоку вразливих даних. API-ключі можуть бути відкликані або змінені без необхідності змінювати паролі. Це спрощує керування доступом. Також вони можуть мати різні рівні доступу, що дозволяє легко керувати правами доступу до різних частин вашого API.

Авторизація через API-ключі є зручним і безпечним способом надання доступу до захищених маршрутів вашого API.

Наступним є модуль взаємодії з клієнтами. Він забезпечує операції (створення, читання, оновлення) для роботи з даними про клієнтів тренерів. Цей модуль складається з чотирьох основних роутів, кожен з яких виконує певну функцію.

Отримання списку клієнтів – /client/list GET. Запит /client/list призначений для отримання повного списку всіх клієнтів у системі. Він є дуже важливим для

тренерів та адміністраторів, оскільки він дозволяє їм бачити всю інформацію про клієнтів, включаючи їхні контактні дані, історію платежів, статус активності та інші деталі (ліст. 3.5). Це спрощує управління клієнтською базою та надає необхідну інформацію для ефективної роботи.

Лістинг 3.5 – Відповідь /client/list


```
{
  "clients": [
    {
      "age": 21,
      "email": "illymartynyk@gmail.com",
      "gender": null,
      "id": 3,
      "is_active": null,
      "name": "Тест",
      "payment": {
        "history": [],
        "info": null
      },
      "phone": "+380501659176",
      "photo_url": null,
      "surname": "Тест",
      "time_of_create": null,
      "trainings": []
    },
    {
      "age": 21,
      "email": "illymartynyk@gmail.com",
      "gender": {
        "key": "man",
        "name": "Чоловік"
      },
      "id": 1,
      "is_active": true,
      "name": "Ілля",
      "payment": {
        "history": [],
        "info": null
      },
      "phone": "+380501659176",
      "photo_url": "",
      "surname": "Мартинюк",
      "time_of_create": "Thu, 01 Jan 1970 00:00:00 GMT",
      "trainings": []
    }
  ],
```

```
  "count_of_clients": 2
}
```

Кінець лістингу 3.5

Маршрут для створення нового клієнта – /client/new. Він використовується для додавання нового клієнта до системи і реалізується за допомогою методу POST. Коли цей маршрут викликається, сервер отримує дані у форматі JSON (ліст. 3.6), що містять інформацію про клієнта, таку як вік, електронну пошту, гендер, ім’я, телефон і прізвище. Ці дані витягуються з JSON-об’єкта і перевіряються на коректність, що включає валідацію правильності формату електронної пошти, віку, телефону та інших обов’язкових полів.

Лістинг 3.6 – Тіло запиту /client/new

```
{
  "age": 21,
  "email": "illymartynyk@gmail.com",
  "key": "man",
  "name": "Тест",
  "phone": "+380501659176",
  "surname": "Тест"
}
```

Кінець лістингу 3.6

Після валідації дані використовуються для створення нового клієнта у базі даних. При цьому унікальний запис id генерується автоматично. Новий запис про клієнта додається до бази даних, і сервер формує відповідь у форматі JSON (ліст. 3.7), що містить унікальний ідентифікатор клієнта та статус успішності операції. Відповідь повертається клієнту з кодом статусу HTTP 201 Created, що вказує на успішне створення нового запису.

Лістинг 3.7 – Відповідь запиту /client/new

```
{
  "client_id": 3,
  "success": true
}
```

Кінець лістингу 3.7

Ця реалізація забезпечує ефективний і надійний спосіб для тренерів та адміністраторів додавати нових клієнтів до системи, підтримуючи актуальну і повну базу даних клієнтів. За допомогою отриманого id, frontend частина додатку може відразу взаємодіяти з новим об’єктом.

Маршрут для деактивації клієнта – /client/<id_>/disable. Він блокує клієнта в системі, що здійснюється шляхом зміни значення поля is_active у записі цього клієнта в таблиці Client. Цей маршрут викликається за допомогою методу POST і приймає параметр id_, що є унікальним ідентифікатором клієнта. Коли викликається цей маршрут, сервер отримує запит і витягує id_ клієнта з URL. Після цього здійснюється запит до бази даних для знаходження відповідного клієнта. Якщо клієнт знайдений, значення його поля is_active змінюється на False, що означає деактивацію клієнта. Потім оновлені дані зберігаються в базі даних.

Після успішної деактивації клієнта сервер формує відповідь у форматі JSON, що містить повідомлення та статус успішності операції, встановлений у true (ліст. 3.8). Ця відповідь повертається клієнту з кодом статусу HTTP, що підтверджує успішне виконання операції. Такий підхід дозволяє адміністраторам та тренерам легко і швидко деактивувати клієнтів у системі, зберігаючи актуальність даних про активних користувачів та підтримуючи порядок у клієнтській базі.

Лістинг 3.8 – Відповідь запиту /client/<id_>/disable

```
{
  "message": "Client disabled.",
  "success": true
}
```

Кінець лістингу 3.8

Наступним я розробляв модуль роботи з тренуваннями. Він є ключовим компонентом системи управління тренувальним процесом, що забезпечує ефективне та організоване ведення тренувань для клієнтів. Цей модуль дозволяє тренерам планувати, відслідковувати та аналізувати тренувальні заняття,

забезпечуючи тим самим індивідуальний підхід до кожного клієнта і підвищуючи ефективність процесу.

Основні функції модуля включають створення та управління розкладом тренувань, реєстрацію відвідувань клієнтів, зберігання історії тренувань, а також відслідковування прогресу клієнтів. Тренери можуть легко створювати нові тренувальні сесії, призначати їх конкретним клієнтам або групам, а також вносити зміни до розкладу в режимі реального часу. Це забезпечує гнучкість і зручність у плануванні, дозволяючи швидко адаптуватися до змін у розкладі або потреб клієнтів.

Модуль також дозволяє зберігати детальну інформацію про кожне тренування, включаючи дату, час, тривалість, тип тренування, а також індивідуальні коментарі та оцінки тренера. Це сприяє точному відслідковуванню прогресу клієнтів і надає можливість аналізувати ефективність різних видів тренувань. Крім того, збережена історія тренувань допомагає тренерам вносити корективи до плану тренувань, базуючись на попередніх результатах і досягненнях клієнтів.

Інтеграція модуля роботи з тренуваннями з іншими модулями системи, забезпечує комплексний підхід до управління тренувальним процесом. Таким чином, модуль роботи з тренуваннями є необхідним інструментом для тренерів і адміністраторів, що забезпечує організацію, ефективність і персоналізацію тренувального процесу, сприяючи покращенню результатів клієнтів і загальної якості послуг, що надаються.

Першим і найголовнішим в цьому модулі є маршрут /trainings/list. Він повертає список тренувань з можливістю фільтрації та пагінації, що дозволяє тренерам і адміністраторам управляти розкладом тренувань. Коли викликається цей запит, сервер приймає такі параметри як номер сторінки (page), кількість елементів на сторінці (per_page), ідентифікатор клієнта (client_id) та період вибірки (period). Ці параметри допомагають визначити, які саме тренування потрібно повернути.

Параметри page та per_page використовуються для визначення діапазону тренувань, які слід включити у відповідь, забезпечуючи пагінацію результатів щоб зменшити навантаження на сервер і отримувати дані невеликими частинами. Якщо вказано параметр client_id, сервер фільтрує тренування для конкретного клієнта. Параметр period дозволяє визначити часовий інтервал для вибірки тренувань, підтримуючи різні формати, включаючи конкретні дати, часові відрізки та попередньо визначені періоди, такі як сьогодні, цей тиждень, цей місяць тощо.

Після обробки параметрів сервер виконує запит до бази даних для отримання відповідних записів тренувань. Кожен запис містить детальну інформацію про тренування, включаючи дані про клієнта, коментар тренера, тривалість, час початку та закінчення, статус тренування та інші метадані. Зібрані дані формуються у відповідь у форматі JSON, який повертається клієнту (ліст. 3.9). Такий підхід забезпечує гнучке і зручне управління тренуваннями, дозволяючи легко отримувати потрібну інформацію у різних контекстах, що сприяє ефективному плануванню та аналізу тренувального процесу.

Лістинг 3.9 – Відповідь запиту /trainings/list


```
{
  "trainings": [
    {
      "client": {
        "id": 1,
        "name": "Ілля",
        "photo_url": "",
        "surname": "Мартинюк"
      },
      "comment": "test",
      "duration": 60,
      "end_time": "Wed, 19 Jun 2024 19:30:00 GMT",
      "id": 8,
      "is_paid": null,
      "start_time": "Wed, 19 Jun 2024 18:30:00 GMT",
      "status": {
        "key": "planned",
        "title": "Заплановане"
      }
    },
    {
```

```
      "client": {
        "id": 1,
        "name": "Ілля",
        "photo_url": "",
        "surname": "Мартинюк"
      },
      "comment": "test",
      "duration": 60,
      "end_time": "Fri, 19 Apr 2024 19:30:00 GMT",
      "id": 7,
      "is_paid": null,
      "start_time": "Fri, 19 Apr 2024 18:30:00 GMT",
      "status": {
        "key": "completed",
        "title": "Завершене"
      }
    }
  ]
}
```


Кінець лістингу 3.9

Маршрут для додавання нового тренування – /trainings/new. Він використовується для створення нових тренувань у системі, що дозволяє тренерам додавати тренування для одного або декількох клієнтів одночасно, зменшуючи кількість запитів з фронтенду. Коли цей запит викликається, сервер отримує тіло запиту у форматі JSON (ліст. 3.10), що містить список ідентифікаторів клієнтів, час початку та закінчення тренування, дати та коментар. Спочатку сервер витягує ці дані та перевіряє їх на коректність, включаючи валідацію часу, дат та ідентифікаторів клієнтів.

Лістинг 3.10 – Тіло запиту /trainings/new


```
{
  "client_ids": [
    1
  ],
  "start_time": "18:30:00",
  "end_time": "19:30:00",
  "dates": [
    "2024-04-19"
  ],
  "comment": "test"
}
```


Кінець лістингу 3.10

Після успішної валідації сервер створює окремі записи для кожного клієнта на кожну вказану дату. Це означає, що для кількох клієнтів і кількох дат буде створено відповідну кількість записів тренувань. Кожен запис містить ідентифікатор клієнта, час початку та закінчення тренування, дату та коментар. Ці записи додаються до бази даних, що забезпечує збереження інформації про тренування.

Після створення всіх необхідних записів сервер формує відповідь у форматі JSON (ліст. 3.11), яка містить повідомлення про успішне створення тренування та статус успішності операції. Ця відповідь повертається клієнту з HTTP-статусом, що підтверджує успішне виконання запиту. Реалізація цього маршруту забезпечує ефективне та зручне додавання тренувань для кількох клієнтів одночасно, що покращує управління розкладом і організацію тренувального процесу.

Лістинг 3.11 – Відповідь запиту /trainings/new

```
{
  "message": "Training created.",
  "success": true
}
```

Кінець лістингу 3.11

Наступним я розробив модуль оплат. Він є критично важливим компонентом будь-якої аналогічної інформаційної системи, особливо в контексті фітнес-клубів, тренувальних студій та інших організацій, що надають послуги на основі підписки або оплати за заняття. Цей модуль забезпечує можливість ефективного управління фінансовими транзакціями, що є фундаментальним для стабільного функціонування бізнесу. Завдяки модулю оплат можна легко відстежувати надходження коштів від клієнтів, що дозволяє менеджменту мати чітке уявлення про фінансовий стан компанії.

Інтеграція модуля оплат також підвищує прозорість і довіру клієнтів, оскільки вони можуть бачити свої транзакції, отримувати підтвердження оплати. Це знижує ризик виникнення непорозумінь і конфліктів, пов’язаних з оплатами,

та покращує взаємодію з клієнтами. Крім того, автоматизація процесу обробки оплат зменшує навантаження організоційної роботи тренерів, дозволяючи їм зосередитись на наданні якісних послуг.

Крім того, модуль оплат в майбутньому дозволить ефективно управляти акціями, знижками та іншими маркетинговими стимулами. Це допоможе залучати нових клієнтів і утримувати існуючих, створюючи конкурентні переваги. Фінансові дані, зібрані цим модулем, також можуть бути використані для аналітики та прийняття стратегічних рішень, що сприяє довгостроковому розвитку бізнесу.

Таким чином, модуль оплат не тільки забезпечує безперебійний фінансовий потік, але й підвищує ефективність управління, покращує взаємодію з клієнтами та сприяє зростанню бізнесу. Його важливість важко переоцінити, оскільки він є основою для стабільності та успіху будь-якої організації, що працює з клієнтськими платежами.

Головним роутом цього модулю є /payment/list – отримання списку оплат. Він виконує ключову функцію, надаючи тренерам можливість отримувати інформацію про оплати їх клієнтів.

### 3.3 Тестування та налагодження API

Процес тестування та налагодження API є важливим етапом розробки будь-якого продукту. Він включає в себе різні методи, які допомагають виявити і виправити помилки, дослідити причини їх виникнення, а також забезпечити надійну і стабільну роботу системи.

Для тестування API використовувався Postman – популярна програма, що дозволяє здійснювати HTTP-запити (рис. 3.2) та аналізувати відповіді серверів [9]. Використання Postman допомагає швидко перевірити коректність роботи API на різних етапах розробки. Відповіді HTTP-запитів у ньому виглядають більш інтерактивно та зрозуміло.

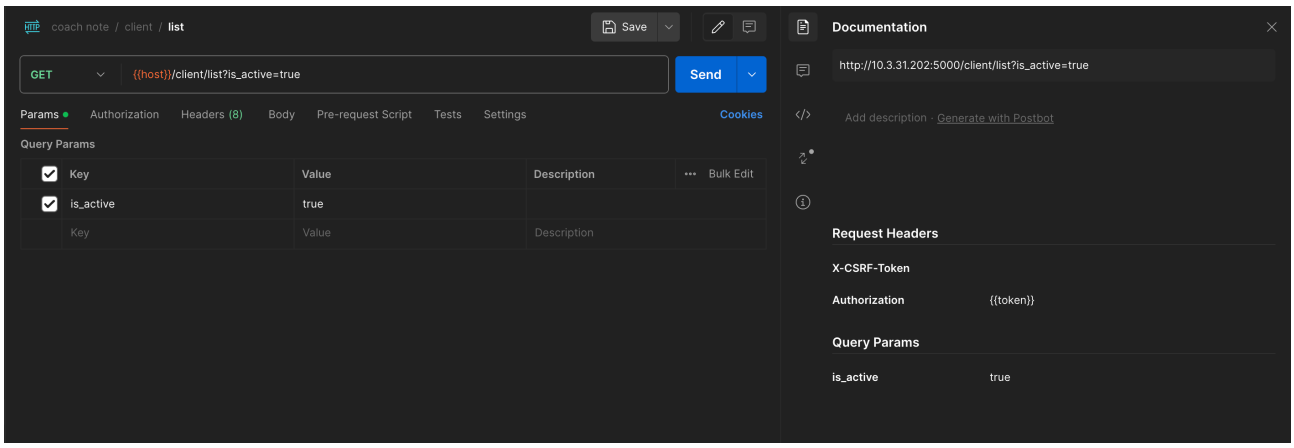


Рисунок 3.2 – Приклад запиту Postman

Колекції запитів і їх прикладів відповідей (рис. 3.3) можна легко поширювати іншим користувачам, що значно спрощує роботу з API команді розробки.

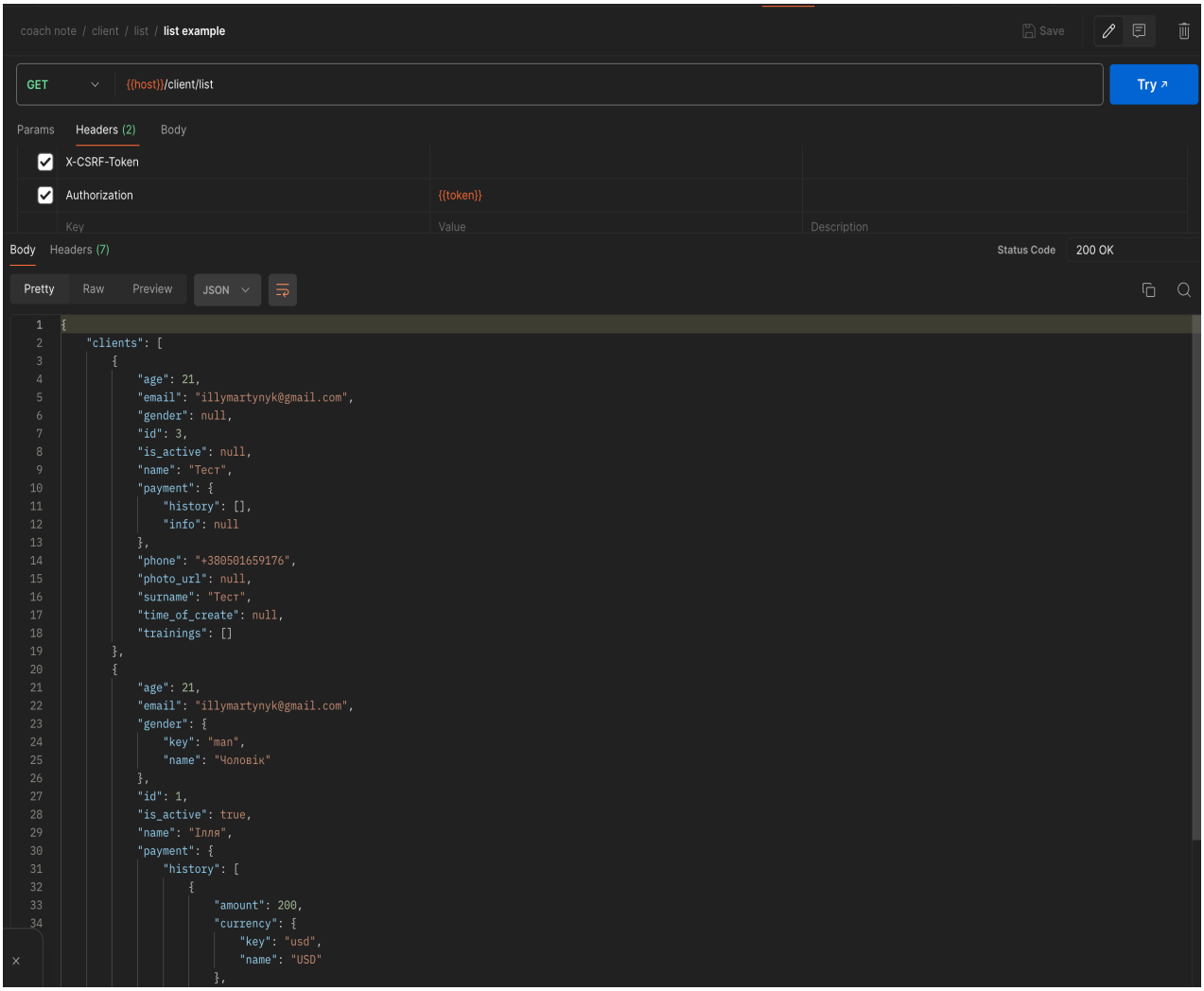


Рисунок 3.3 – Приклад збереженої відповіді запиту в Postman

Використання змінних оточення для зберігання конфігураційних параметрів (наприклад, базового URL сервера, ключів доступу та іншого) дозволяє легко змінювати налаштування для різних середовищ (розробка, тестування, продакшн) і дозволяє поширювати вашу колекцію запитів без ризиків поширення сенсативних даних.

Також ця програма містить інструменти для автоматичного тестування певних сценаріїв. Для кожного запиту Postman колекції можна додати автоматизовані тести на мові JavaScript, які автоматично перевірятимуть коректність відповіді сервера [10].

На рисунку 3.4 зображено запит на отримання списку клієнтів для поточного користувача-тренера - /client/list. Запит приймає параметр is_active, який дозволяє отримувати тільки активних або тільки неактивних користувачів. За замовчуванням значення параметру – null, та запит повертає всіх користувачів.

Також в запиті використано 2 змінні оточення – host і token. Для позначення їх як змінних, а не значень використовуються подвійні фігурні дужки.

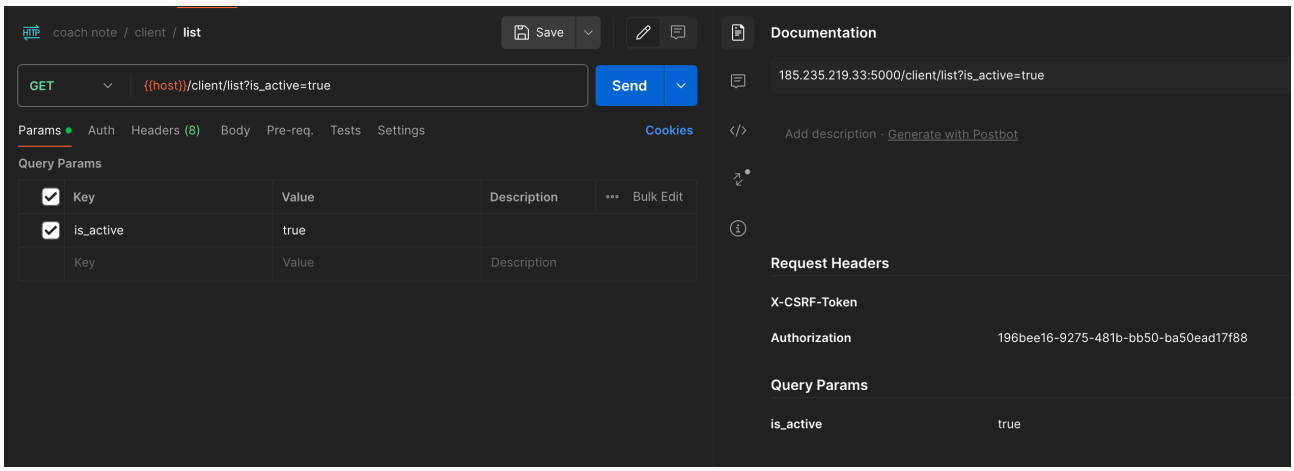


Рисунок 3.4 – Запит на отримання списку клієнтів в Postman

Налаштування API здійснювалося через конфігураційний файл config.py (ліст. 3.12), що використовує змінні оточення для зберігання конфіденційних даних і параметрів середовища.

Лістинг 3.12 – Конфігураційний файл config.py

```
import os


class Config:
    SECONDS_IN_MINUTE = 60
    SECONDS_IN_HOUR = SECONDS_IN_MINUTE * 60
    SECONDS_IN_DAY = SECONDS_IN_HOUR * 24
    SECONDS_IN_YEAR = SECONDS_IN_DAY * 365

    DEBUG = read_bool_from_os_env("DEBUG")
    DATABASE_HOST = os.environ.get("DATABASE_HOST")
    DATABASE_USER = os.environ.get("DATABASE_USER")
    DATABASE_PASSWORD = os.environ.get("DATABASE_PASSWORD")
    DATABASE_NAME = os.environ.get("DATABASE_NAME")
    DATABASE_URL =
f"postgresql://{DATABASE_USER}:{DATABASE_PASSWORD}@{DATABASE_HOST}:5432/{
DATABASE_NAME}"
    SQLALCHEMY_DATABASE_URI = DATABASE_URL
    REDIS_HOST = os.environ.get("REDIS_HOST")
    REDIS_PORT = os.environ.get("REDIS_PORT")
    REDIS_PASSWORD = os.environ.get("REDIS_PASSWORD")

    SESSION_TIMEOUT = SECONDS_IN_DAY * 10

    MAX_CONTENT_LENGTH_MBYTE = 20

    WTF_CSRF_ENABLED = False

    # Rate Limiting
    RATE_LIMIT_AUTHENTICATED_QUOTA = 10000
    RATE_LIMIT_UNKNOWN_QUOTA = 120
    RATE_LIMIT_WINDOW = SECONDS_IN_HOUR
    RATE_LIMIT_PREFIX = "RateLimit:"

    SECRET_KEY = os.environ.get("SECRET_KEY")
    ACCESS_TOKEN_SALT = os.environ.get("ACCESS_TOKEN_SALT")
    TOKEN_PREFIX = "token:"

    DATE_TIME_FORMAT = "%Y-%m-%d %H:%M:%S"
    DATE_FORMAT = "%Y-%m-%d"
```

Кінець лістингу 3.12

Для ефективного налагодження API використовувалося логування помилок та ключових подій. В Python це досягалося за допомогою вбудованого модуля logging, який підключається в функції-фабриці додатку (ліст. 3.13) [11]. Модуль надає можливість записувати логи різних рівнів (DEBUG, INFO,

WARNING та CRITICAL), відсікати їх по рівнях для різних середовищ розробки та зберігати в відповідні файли.

Лістинг 3.13 – Підключення модуля логування для додатку

```
from flask import Flask
from flask_cors import CORS
from flask_login import LoginManager
from flask_marshmallow import Marshmallow
from flask_sqlalchemy import SQLAlchemy

from config import Config, create_logger
...


def create_app():
    ...

    app.logger = create_logger(__name__)

    ...

    return app
```

Кінець лістингу 3.13

## Висновки до розділу 3

У третьому розділі було проведено комплексну розробку і реалізацію ключових компонентів API мобільного додатку для обліку клієнтів. Спочатку здійснено детальне проєктування та створення бази даних, яка забезпечує ефективне зберігання та управління інформацією про клієнтів, тренерів, тренування та інші пов’язані дані. Скориставшись сервісом dbdiagram.io вдалося швидко і ефективно сформувати схему БД, візуалізувати її. Використання PostgreSQL як системи управління базами даних забезпечило надійність, масштабованість та високу продуктивність роботи з великими обсягами даних.

Далі було розроблено API, яке забезпечує взаємодію мобільного додатку з базою даних. Використання Flask у поєднанні з SQLAlchemy дозволило створити гнучкий і розширюваний API, що відповідає принципам REST [12]. Особлива увага приділялася безпеці, оптимізації запитів та забезпеченню швидкого доступу до даних. Завдяки використанню Redis як кешуючого механізму,

вдалося значно покращити швидкість обробки запитів та знизити навантаження на основну базу даних. Також Redis виступив сховищем для API ключів клієнтів.

Заключним етапом стало тестування та налагодження API. Було проведено різноманітні види тестування, включаючи модульне, інтеграційне та навантажувальне тестування, а також тестові сценарії Postman. Це дозволило виявити та виправити потенційні помилки й забезпечити стабільну роботу. Налагодження включало оптимізацію коду та конфігурацій серверного оточення, що також виконувалося за допомогою Docker Compose, забезпечуючи ізольоване середовище для кожного компонента системи.

Таким чином, реалізація всіх запланованих етапів дозволила створити повнофункціональне API для мобільного додатку, яке відповідає сучасним вимогам до продуктивності, безпеки та масштабованості. Це API забезпечує ефективну роботу додатку та задоволення потреб користувачів.

# ВИСНОВКИ

В результаті виконання роботи вдалось розробити API для мобільного додатку «CoachNote», який відповідає сучасним вимогам до продуктивності, безпеки та зручності використання.

Було проведено аналіз актуальності проекту та наявних конкурентів, таких як MyFitnessPal і Trainerize, що показав наявність певних прогалин у існуючих рішеннях, які були враховані при розробці власного додатку. Використання сучасного стеку технологій (Python, Flask, PostgreSQL, SQLAlchemy, Redis, Docker Compose) дозволило досягти високих та якісних показників, включаючи швидкість обробки запитів, надійність системи та її масштабованість. Було розроблено UML-діаграми функціоналу і схеми бази даних, які забезпечили чітке розуміння архітектури системи і полегшили процес її реалізації.

Було спроектовано базу даних з використанням інструменту dbdiagram.io, який дозволив якісно та швидко візуалізувати всі таблиці, зв’язки, поля та їх типи. Це значно зекономило час розробки, та спростило розуміння системи, що розроблялася.

Також, у ході роботи, було спроектовано структуру API, яка включає розробку архітектури відносної до бази даних для отримання інформації про клієнтів, тренерів та тренувальні заняття. Таким чином було визначено маршрути (роути) для виконання основних CRUD-операцій (створення, читання, оновлення, видалення). Спроектована структура API дозволяє легко інтегрувати додаткові функції та адаптувати систему до потреб користувачів, забезпечуючи гнучкість та надійність у роботі з даними.

Відповідно до спроектованої структури було реалізовано програмний інтерфейс для додатку «CoachNote». Основний функціонал API забезпечує ефективне управління інформацією про клієнтів і тренерів. API було реалізовано з використанням Flask, забезпечуючи надійний та масштабований спосіб взаємодії між мобільним додатком та сервером. При цьому особлива увага приділялася забезпеченню безпеки даних та оптимізації продуктивності за рахунок використання Redis для кешування та PostgreSQL як основної бази

даних. Всі компоненти системи контейнеризовано за допомогою Docker Compose, що спрощує розгортання та управління додатком в різних середовищах.

API було ретельно протестовано за допомогою інструменту Postman, що дозволило перевірити коректність виконання всіх операцій, таких як створення, читання, оновлення та видалення даних про клієнтів, тренерів та тренувальні сесії. Тестування включало перевірку відповідності API специфікаціям, обробку помилок, а також перевірку надійності та безпеки передачі даних. Особлива увага приділялася тестуванню продуктивності та оптимізації роботи API, що дозволило забезпечити швидку і стабільну взаємодію між клієнтським додатком та серверною частиною. Використання Postman дозволило виявити та усунути потенційні недоліки, забезпечивши високу якість та надійність програмного інтерфейсу для додатку «CoachNote».

Подальші шляхи вдосконалення розроблених у роботі рішень можуть включати інтеграцію з іншими системами управління, розширення функціоналу програмного інтерфейсу для задоволення потреб ширшого кола користувачів, а також оптимізацію алгоритмів обробки даних для підвищення продуктивності. Додаткові дослідження можуть бути спрямовані на впровадження нових технологій, таких як машинне навчання для аналізу прогресу клієнтів, або на розробку модулів для персоналізації тренувальних програм. Таким чином, дана робота робить значний внесок у розвиток інформаційних систем для фітнес-індустрії і має значний потенціал для подальшого розвитку.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційний сайт сервісу MyFitnessPal. URL: https://www.myfitnesspal.com/ (дата звернення: 19.05.2024).

2. Офіційний сайт додатку Trainerize з описом функціоналу. URL: https://www.trainerize.com/features (дата звернення: 19.05.2024).

3. Офіційний сайт Flask. URL: https://flask.palletsprojects.com/en/3.0.x/ (дата звернення: 19.05.2024).

4. Офіційний сайт Django. URL: https://www.djangoproject.com/ (дата звернення: 19.05.2024).

5. Офіційний сайт FastAPI. URL: https://fastapi.tiangolo.com/ (дата звернення: 19.05.2024)

6. Офіційний сайт СУБД Postgres з документацією та додатком для розгортання. URL: https://www.postgresql.org/ (дата звернення: 21.05.2024).

7. Сервіс проектування баз даних dbdiagram.io. URL: https://dbdiagram.io/ (дата звернення: 22.05.2024).

8. Навчальна стаття по CRUD. URL: https://www.codecademy.com/article/what-is-crud (дата звернення: 21.05.2024).

9. Офіційний сайт додатку Postman. URL: https://www.postman.com/ (дата звернення: 24.05.2024).

10. Туторіал по JavaScript. URL: https://www.w3schools.com/js/ (дата звернення: 24.05.2024).

11. Документація по Python модулю logging. URL: https://docs.python.org/uk/3/library/logging.html (дата звернення: 24.05.2024).

12. Конвенції по REST. URL: https://medium.com/@nadinCodeHat/rest-api-naming-conventions-and-best-practices-1c4e781eb6a5 (дата звернення: 24.05.2024).