ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка застосунку для фітнес-центру з використанням мови Kotlin та C#»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Родіон ГРАМУШНЯК
(підпис)

Виконав: здобувач вищої освіти групи ПД-41
Родіон ГРАМУШНЯК

Керівник: Олена НЕГОДЕНКО
к.т.н., доцент

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Грамушняку Родіону Олександровичу

1. Тема кваліфікаційної роботи: «Розробка застосунку для фітнес-центру з використанням мови C# та Kotlin»
керівник кваліфікаційної роботи к.т.н., доцент Олена НЕГОДЕНКО,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, механіки роботи Android застосунків.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної області та постановка задач.
2. Технічне завдання та вимоги.
3. Вибір засобів реалізації.
4. Проектування та розробка.
5. Огляд роботи Android застосунку.
6. Тестування Android застосунку.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз інтерфейсних рішень аналогів.
2. Аналіз аналогів.
3. Вимоги до застосунку.
4. Програмні засоби реалізації.
5. UseCase діаграма.
6. Схема архітектури серверного застосунку.
7. Схема BookingController.
8. Екранні форми входу в профіль.
9. Екранні форми навігаційної системи.
10. Демонстрація роботи застосунку.
11. Апробація результатів дослідження.

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів застосунків фітнес центрів	07.03-13.03.2024	
4	Моделюванні функціональних вимог користувачів системи. Визначення нефункціональних вимог	14.03-15.03.2024	
5	Розробка архітектури сервісу та Android застосунку.	16.03-25.03.2024	
6	Проектування та розробка логіки роботи сервісу.	26.03-04.04.2024	
7	Проектування та розробка логіки роботи Android застосунку, для онлайн запису на тренування в фітнес центр.	05.04-25.04.2024	
7	Тестування застосунку	26.04-28.04.2024	
9	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
10	Розробка демонстраційних матеріалів	06.05-12.05.2024	
11	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Родіон ГРАМУШНЯК
(підпис)

Керівник
кваліфікаційної роботи _____ Олена НЕГОДЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 61 стор., 28 рис., 8 табл., 9 джерел.

Розробка застосунку для фітнес центру з використанням мови C# та Kotlin.

Мета роботи – покращити взаємодію клієнтів до послуг фітнес центру за допомогою застосунку, створеного мовами C# та Kotlin.

Об'єкт дослідження – процес взаємодії клієнтів до послуг фітнес центру.

Предмет дослідження – застосунок для взаємодії клієнтів до послуг фітнес центру.

Короткий зміст роботи: В роботі проаналізовано алгоритми та методи для створення онлайн-запису в контексті спортивного фітнес центру. Проаналізовано Andorid застосунки фітнес центрів для створення онлайн запису: Apolo Next, Altegio, SportLife, ClassPass. Розроблено архітектуру серверної частини та Android застосунку програмно реалізовані ключові функціональні можливості, зокрема: перегляд інформації про фітнес центр, перегляд списку тренерів, створення онлайн-запису на тренування, перегляд історії тренувань, надсилання нагадувань про заплановане тренування, реєстрація та вхід в профіль. Проведено функціональне тестування Android застосунку. Та розроблені автотести для серверної частини. В роботі використано Android SDK та Kotlin для реалізації Android застосунку, C#, ASP.net, CORE.net для реалізації серверної частини.

Сферою використання застосунку є адміністрування записів у сфері фітнесу та спорту.

КЛЮЧОВІ СЛОВА: FIGHT ACADEMY, C#, ASP.NET, ENTITY FRAMEWORK, POSTGRE SQL, ANDROID SDK, KOTLIN COROUTINES.

ЗМІСТ

ВСТУП 10

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ 11

1.1 Функціональні можливості застосунку FightAcademy 12

1.1.1 Огляд застосунку Apolo Next 13

1.1.2 Огляд застосунку Altegio 18

1.1.3 Огляд застосунку SportLife 21

1.1.4 Огляд застосунку ClassPass 25

2 ТЕХНІЧНЕ ЗАВДАННЯ ТА ВИМОГИ 30

2.1 Технічне завдання 30

2.2 Функціональні вимоги 30

2.2.1 Валідація введеного користувачем паролю 31

2.2.2 Запис на тренування 31

2.2.3. Перегляд історії записів 32

2.2.4. Відправлення нагадувань через сповіщення 32

2.2.5. Функціональна вимога логіну 32

2.2.6. Функціональна вимога реєстрації 33

2.3. Нефункціональні вимоги 33

2.3.1. Надійність та стійкість 34

2.3.2. Зрозумілість інтерфейсу 34

2.3.3. Масштабування системи 34

2.4. Системні вимоги 34

3 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ 35

3.1 Середовище розробки JetBrains Rider 35

3.2 Середовище розробки Android Studio 35

3.3 Мова програмування – C# 36

3.4 ASP.NET - Фреймворк для побудування API 37
3.5 Entity Framework (EF) 38
3.6 Kotlin та корутини 38
3.7 Kotlin Flow 39
4 ПРОЕКТУВАННЯ ТА РОЗРОБКА 41
4.1 Проектування діаграми використання 41
4.2 Проектування внутрішньої архітектури сервісу 43
4.3 Проектування діаграми залежностей сервісу FightAcademy.Client 45
4.4 Робота контролерів 48
4.5 Підключення та робота з PostgeSQL 49
4.6 Проектування внутрішньої архітектури застосунку 51
4.6 Розробка екрану тренерів в застосунку з архітектурою MVVM 52
5 ОГЛЯД РОБОТИ ЗАСТОСУНКУ 55
5.1 Перевірка роботи реєстрації та автентифікація 55
5.2 Початковий екран 56
5.3 Екран тренерів 58
5.4 Екран активностей 60
5.5 Екран налаштувань 65
6 ТЕСТУВАННЯ ЗАСТОСУНКУ 67
6.1 Опис підходу до тестування застосунку 67
6.2 Тестування програми 67
ВИСНОВКИ 72
ПЕРЕЛІК ПОСИЛАНЬ 73
ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛ
(Презентація) 75

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Back-end – бізнес-логіка веб-сайту чи застосунку.

IDE – Integrated Drive Electronics, інтегроване середовище розробки

Android-застосунок - це програмне забезпечення, призначене для використання на мобільних пристроях, що працюють під управлінням операційної системи Android.

UI - User Interface (користувацький інтерфейс) - визначення елементів і взаємодія користувача з застосунком.

БД – база даних.

FightAcademy - назва Android-застосунку для фітнес центру

ВСТУП

У сучасному світі спорт і здоровий спосіб життя набувають все більшого значення для людей усіх вікових груп. З цього приводу створення онлайн платформ для запису на тренування та адміністрування цих записів стає надзвичайно актуальним завданням. Застосунок FightAcademy розроблений з метою забезпечення зручності і ефективності у веденні розкладу тренувань та контролю за ними.

Створення застостосунку FightAcademy відповідає актуальним потребам спортивних центрів та клієнтів у зручних та ефективних інструментах для планування та запису на тренування.

Мета роботи – покращити взаємодію клієнтів до послуг фітнес центру за допомогою мобільного застосунку, створеного мовами C# та Kotlin.

Об’єкт дослідження – процес взаємодії клієнтів до послуг фітнес центру.

Предмет дослідження – застосунок для взаємодії клієнтів до послуг фітнес центру.

Для досягнення поставленої мети були сформульовані такі задачі:

1. Проаналізувати ринок послуг фітнес центрів та встановити основні потреби користувачів.

2. Розробити вимоги до додатку, встановити функціональні та технічні вимог до застосунку та серверної системи з урахуванням потреб користувачів.

3. Проаналізувати та вибрати засоби та технології розробки.

4. Розробити модель архітектури та спроектувати застосунок за допомогою діаграми класів та прецедентів.

5. Розробити застосунок за допомогою обраних інструментів та визначених вимог.

6. Провести тестування для перевірки працездатності та відповідності застосунку вимогам.

Проект розроблено з використанням мови програмування C# та фреймворку ASP.NET, які в даний час вважаються одними з найпопулярніших і ефективних засобів для створення ентерпрайз бізнес-застосунків. Обрана мова програмування відповідає всім вимогам проекту та ідеально підходить для створення високопродуктивних і надійних програмних рішень. Разом з ASP.NET вони утворюють потужний інструментарій для розробки різноманітних застосунків, включаючи серверні API.

Проект також використовував мову програмування Kotlin, що є однією з перспективних та швидко розвиваючихся мов для розробки програмного забезпечення. Вибір Kotlin був зумовлений його високою ефективністю та потужністю, які сприяють швидкому і надійному розробці програм.

У фіналі роботи очікується отримання повноцінного та готового до використання застосунку, який задовольняє потреби як клієнтів, так і адміністраторів, та допоможе забезпечити ефективну організацію тренувань та контроль за ними. Цей застосунок повинен відповідати вимогам, сформульованим на етапі його розробки, та вирішувати ключові завдання, пов'язані з організацією тренувань та управлінням записами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ

Актуальність теми онлайн записів визначається кількома ключовими факторами, які відображають сучасні тенденції та потреби суспільства.

Зручність і доступність: У сучасному світі люди все більше цінують свій час і зручність. Онлайн записи дозволяють клієнтам бронювати послуги в будь-який час і з будь-якого місця за допомогою смартфона або комп'ютера, що дуже зручно та ефективно.

Ефективне управління часом: Записи в режимі онлайн дозволяють організаціям ефективно управляти своїм розкладом і ресурсами. Вони можуть оптимізувати використання часу та персоналу, мінімізуючи час, який витрачається на адміністративні процеси.

Збільшення клієнтської бази: Онлайн записи дозволяють привернути нових клієнтів, які шукають зручні інструменти для замовлення послуг. Це може допомогти розширити аудиторію та збільшити прибуток.

Підвищення рівня задоволеності клієнтів: Швидкий і зручний процес запису на послуги може позитивно позначитися на враженнях клієнтів і зробити їх більш задоволеними від взаємодії з бізнесом.

Адаптація до сучасних технологій: Запровадження онлайн записів демонструє технологічний прогрес та відповідає сучасним очікуванням клієнтів, які все частіше очікують доступ до послуг через інтернет.

Отже, виробники послуг у різних галузях, включаючи спорт, фітнес, медицину та інші, активно використовують онлайн записи для полегшення процесу замовлення послуг та підвищення задоволеності клієнтів. Ця тема є вкрай актуальною в сучасному цифровому світі і продовжує збирати на себе значні інвестиції та увагу.

Тема створення застосунку FightAcademy для онлайн запису та адміністрування записів є дуже актуальною та важливою в контексті сучасних спортивних тенденцій. Існує декілька об'єктивних причин, які підтверджують цю актуальність.

По-перше, спорт та фітнес набули значної популярності серед широкого загалу населення. Люди все більше усвідомлюють важливість здорового способу життя та регулярних тренувань, включаючи бойові мистецтва. Таким чином, існує великий попит на зручні інструменти для планування та запису на тренування.

По-друге, віддалений доступ до послуг стає все більш важливим у сучасному світі. Онлайн платформи, які дозволяють клієнтам записуватися на тренування з будь-якого місця та в будь-який час, відповідають цій потребі. Вони забезпечують зручність і доступність для клієнтів у всіх аспектах їхнього спортивного життя.

По-третє, конкуренція на ринку фітнес-послуг зростає. Спортивні центри та студії бойових мистецтв змушені конкурувати за клієнтів, тому важливо мати ефективні інструменти для привернення та збереження аудиторії. Застосунок для онлайн запису та адміністрування записів може стати ключовим конкурентним перевагою у цьому контексті.

1.1 Функціональні можливості застосунку FightAcademy

Застосунок призначений для сприяння організації тренувань та покращення спортивного досвіду в галузі бойових мистецтв. Однією з ключових можливостей цієї платформи є можливість легко та швидко запланувати візит на тренування. Користувачі можуть переглянути розклад доступних занять, обрати зручний час і інструктора, а потім забронювати місце за декілька клацань.

Крім того, FightAcademy надає зручну історію відвідувань, що дозволяє користувачам відстежувати свій прогрес та переглядати попередні записи на

тренування. Це допомагає спортсменам аналізувати свої досягнення та встановлювати нові цілі для подальшого розвитку.

У програмі також доступний список тренерів з описом їхньої кваліфікації та досвіду. Користувачі можуть дізнатися більше про кожного тренера, переглянути їхні навички та особисті дані, щоб знайти найбільш підходящого інструктора для своїх потреб.

Одним із переваг FightAcademy є можливість отримувати сповіщення про актуальні записи на заняття. Користувачі можуть налаштувати свої уподобання та отримувати повідомлення про наближення тренувань, що дозволяє їм завжди бути в курсі та планувати свій час ефективно.

У цілому, FightAcademy створює сприятливе середовище для спортсменів у всіх аспектах їхнього спортивного життя, допомагаючи їм досягати нових висот у бойових мистецтвах або фізичних здібностях.

1.1.1 Огляд застосунку Apolo Next

Apolo Next – це застосунок, призначений для перегляду тренувань онлайн. Застосунок Apollo Next призначений полегшити вам відвідування спортзалу Apollo Next. Він допомагає заощадити час, знайти потрібні заняття та тренера, а також керувати своїм членством. Apollo Next пропонує переглянути часи роботи залу та найближчі заняття.

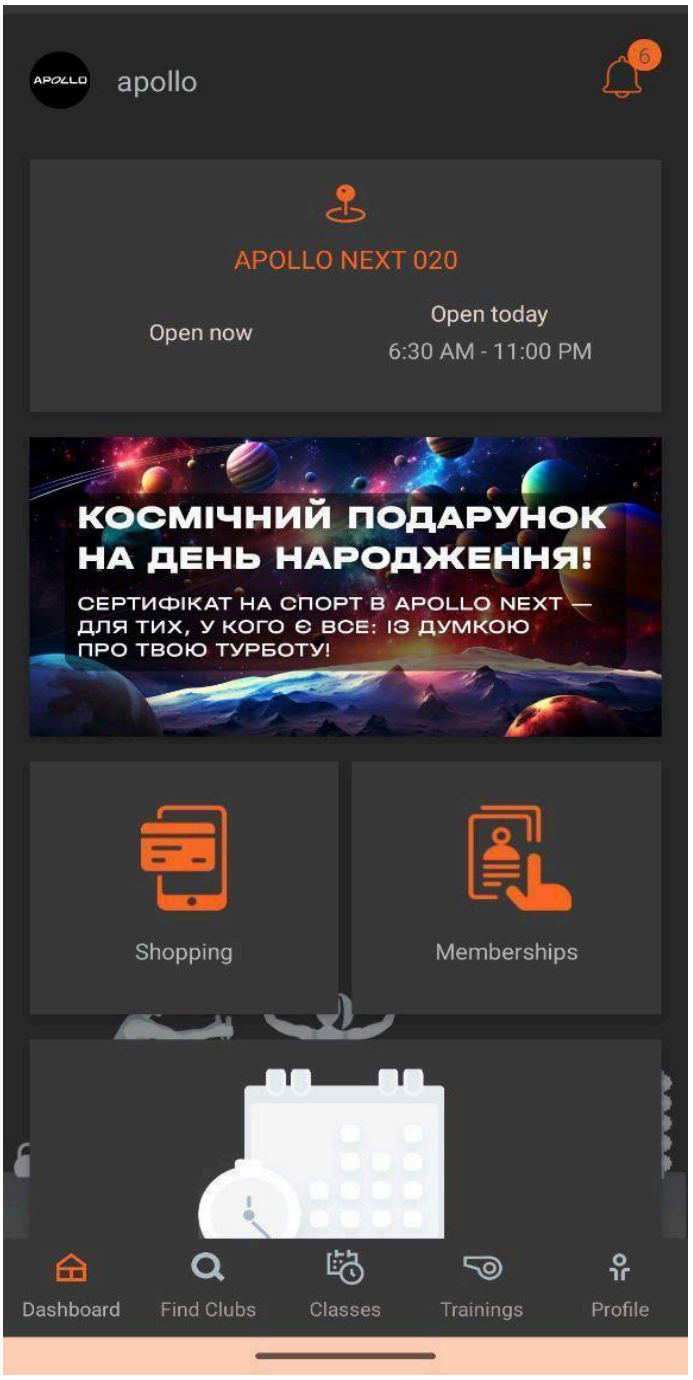


Рис. 1.1 Головна сторінка застосунку Apollo Next

Інтуїтивно зрозумілий та простий процес запису онлайн дозволяє клієнтам швидко та легко забронювати своє місце на тренуванні, вказавши дату та час, а також інші необхідні деталі.

Функціональні можливості:

- Переглянути розклад занять: Переглядайте розклад усіх відкритих

клубів Apollo Next, щоб знайти ідеальний час, місце та тип тренування.

- Знайти тренера: Застосунок може допомогти вам знайти тренера, який найкраще відповідає вашим потребам.
- Забронювати заняття: Забронюйте місце на групових або персональних заняттях, не виходячи з дому. Ви навіть можете записатися на безкоштовне персональне тренування.
- Зарезервувати шафку або сейф: Зарезервуйте особисту шафку для зберігання речей під час тренування або сейф для цінних речей.
- Керувати підпискою: Переглядайте та керуйте своєю підпискою Apollo Next, зокрема здійснюйте автоматичні платежі.

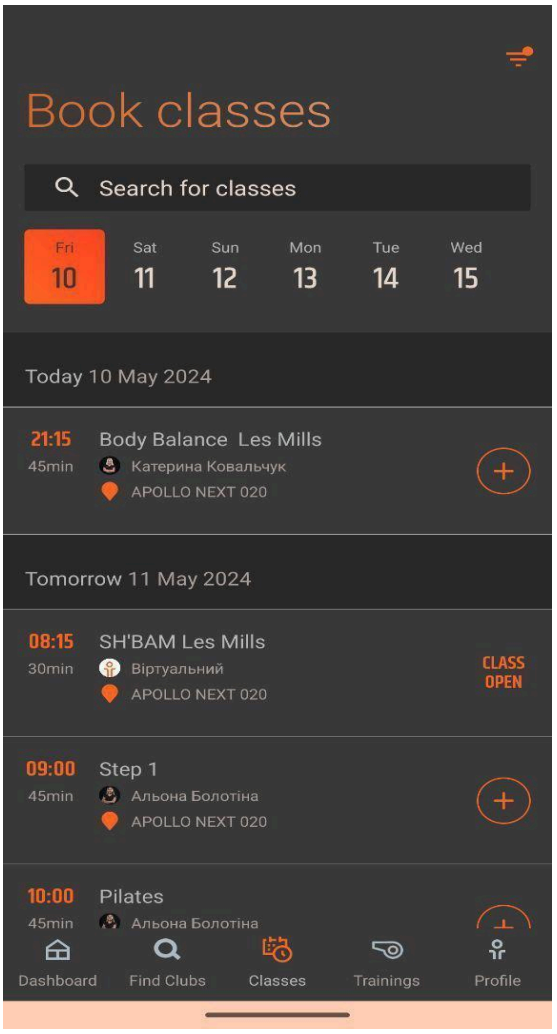


Рис. 1.2 Екран занять

В цілому, застосунок Apollo Next призначений полегшити вам відвідування спортзалу Apollo Next. Він допомагає заощадити час, знайти потрібні заняття та тренера, а також керувати своїм членством. Проте цей застосунок не є якісним, бо при використанні знайдено ряд проблем серед яких: застарілий та заплутаний дизайн (рис. 1.3), відсутність сповіщень (рис 1.4).

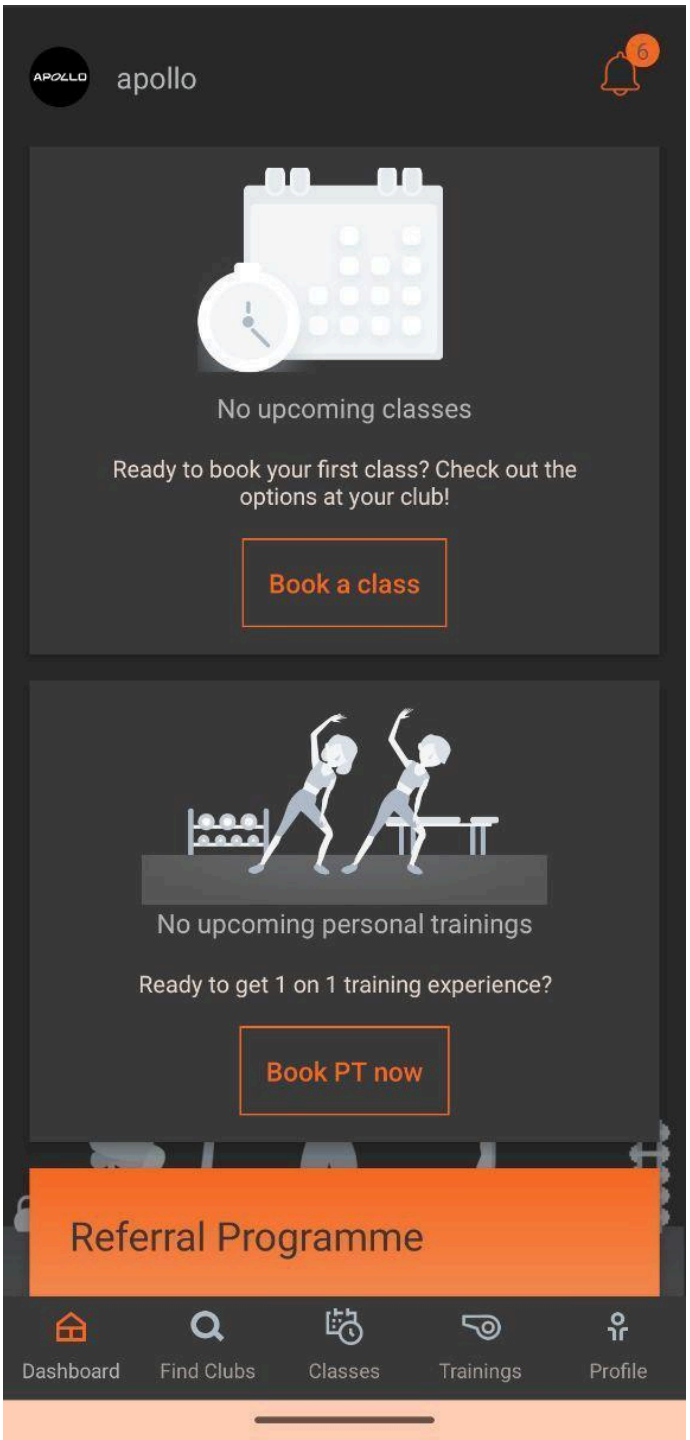


Рис. 1.3 Приклад застарілого інтерфейсу застосунку

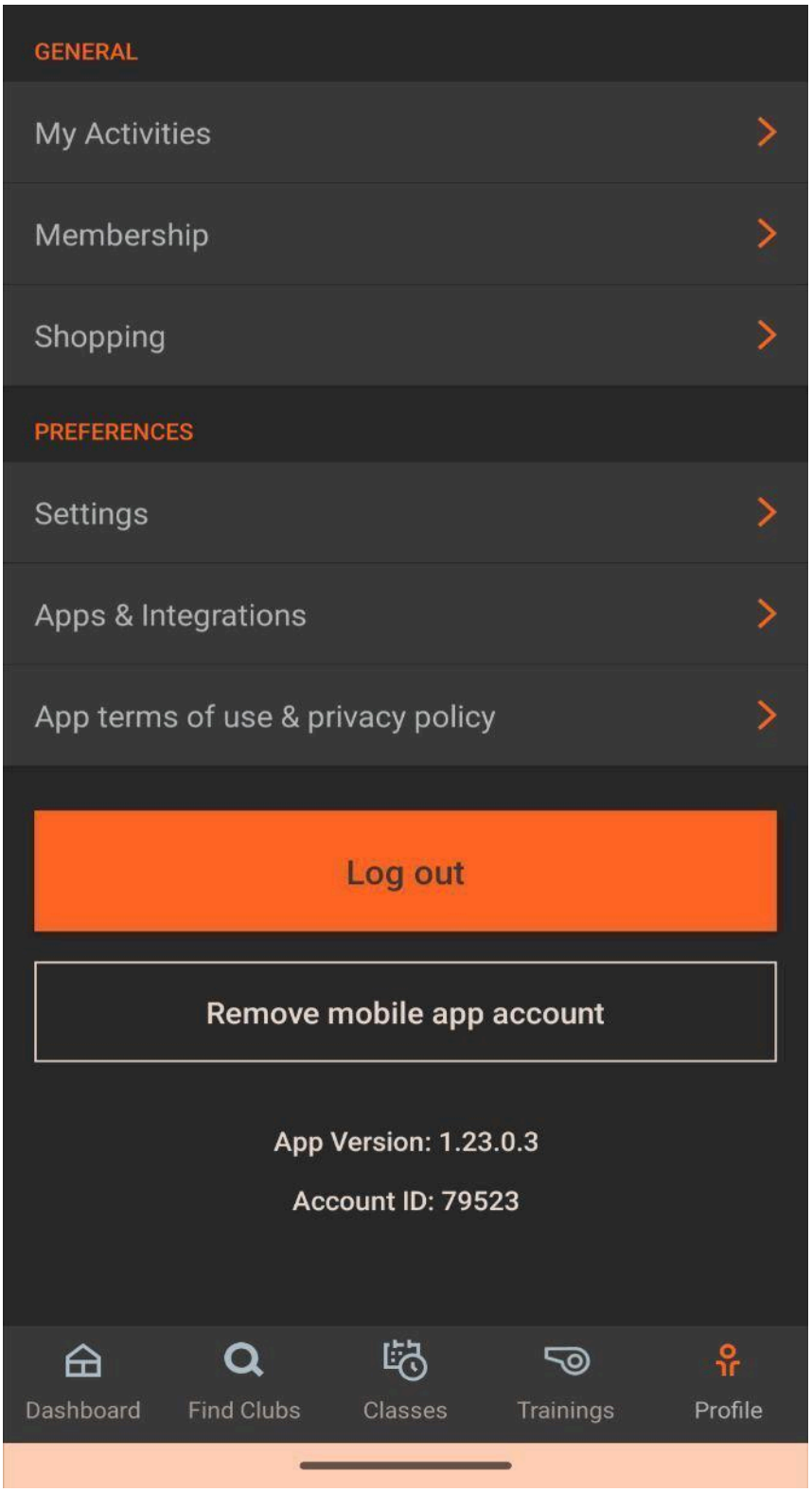


Рис. 1.4 Меню налаштувань профілю застосунку

1.1.2 Огляд застосунку Altegio

Altegio - це хмарна платформа для управління та автоматизації бізнесу в сфері послуг. Вона пропонує широкий спектр інструментів, які допомагають підприємствам економити час, підвищувати ефективність та покращувати обслуговування клієнтів.

Функціональні можливості веб-застосунку включають:

- Онлайн-запис: Дозвольте клієнтам бронювати час онлайн через веб-сайт, мобільний застосунок або соціальні мережі.
- Управління розкладом: Створіть та керуйте розкладом своїх співробітників і ресурсів, щоб оптимізувати завантаженість та уникнути накладок.
- CRM-система: Зберігайте та керуйте даними про клієнтів, включаючи історію записів, уподобання та контактну інформацію.
- Автоматизація маркетингу: Надсилайте автоматичні нагадування про записи, рекламні розсилки та інші маркетингові повідомлення.
- Аналітика: Відстежуйте ключові показники ефективності (KPI) свого бізнесу, щоб приймати обґрунтовані рішення.
- Інтеграція з іншими системами: Altegio можна інтегрувати з іншими системами, такими як бухгалтерське програмне забезпечення та системи керування стосунками з клієнтами (CRM).

Altegio представляє собою веб-застосунок, який досить складний в налаштуванні. Залежно від розміру та складності вашого бізнесу налаштування та інтеграція Altegio може потребувати часу та ресурсів. Недостатня профільність для фітнес клубів, не дозволяє реалізувати всі бажання замовника. Лише декілька базових налаштувань. Відсутність фільтрування по обраній даті.

Altegio пропонує різні платні підписки, і деякі підприємства можуть вважати його занадто дорогим.

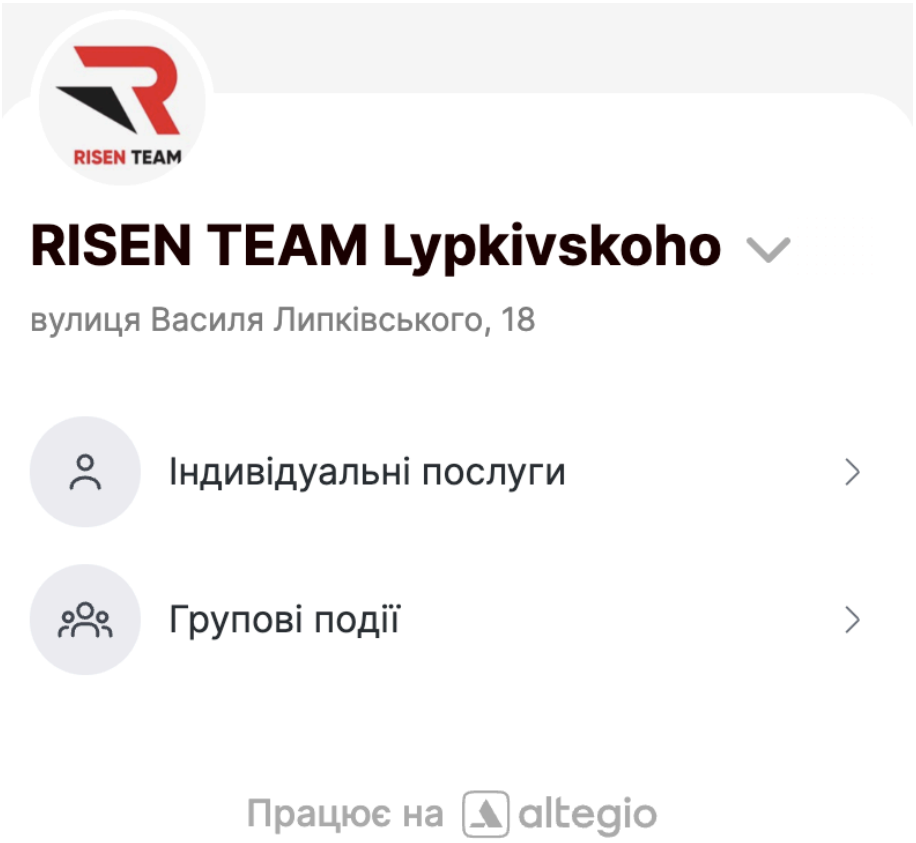


Рис. 1.5 Головне меню веб-застосунку на основі Altegio

Відсутність контенту на головній сторінці веб-застосунку та пусте меню можуть стати серйозною перешкодою для користувачів. Коли головна сторінка не містить достатньої інформації або функціональності, користувачі можуть відчувати розчарування та втрату інтересу до веб-застосунку. Пусте меню також може створювати враження недопрацьованості та незавершеності, що може підірвати довіру користувачів до продукту.

Без вмісту на головній сторінці користувачам може бути важко зрозуміти, як користуватися застосунку або які можливості він пропонує. Це може вплинути на їхній перший враження від застосунку та викликати негативні емоції. В результаті, відсутність контенту та пусте меню можуть призвести до зменшення активності користувачів у веб-застосунку, їхньої невдоволеності та низької відданості продукту.

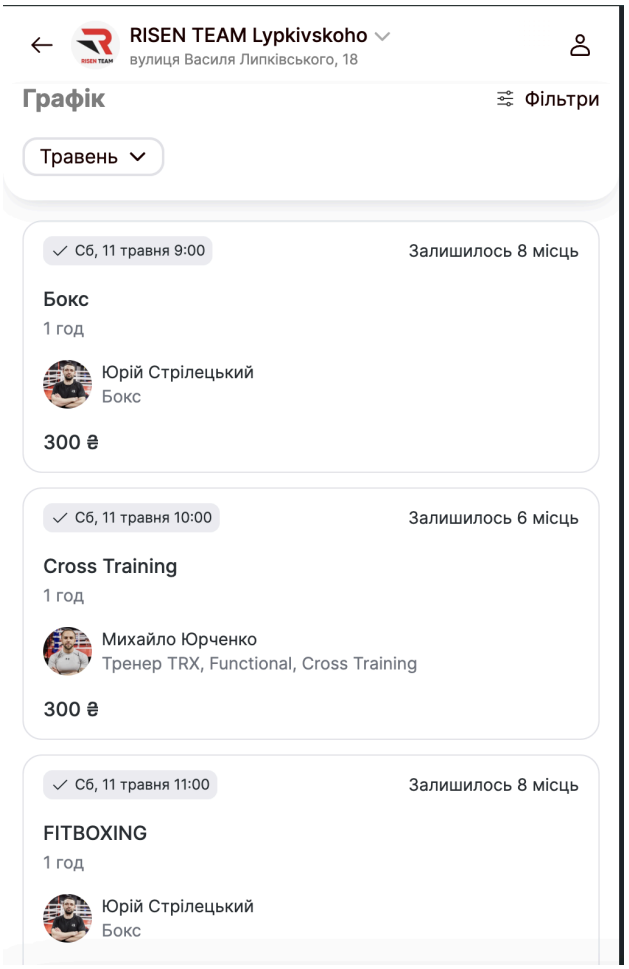


Рис. 1.5 Відсутність фільтрування по обраній даті

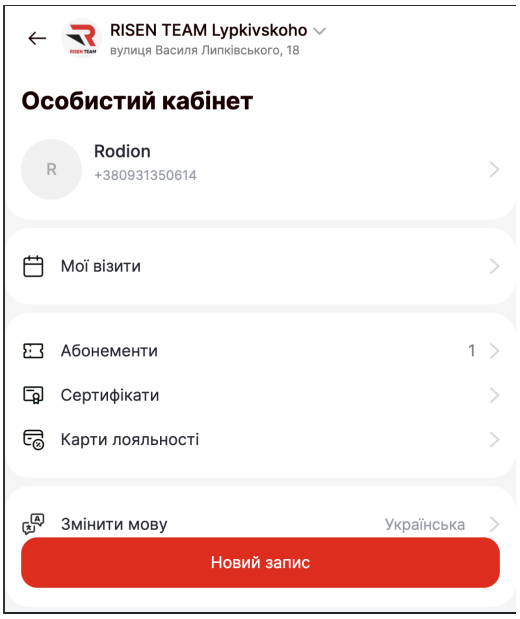


Рис. 1.6 Демонстрація профілю на платформі Altegio

1.1.3 Огляд застосунку SportLife

SportLife – це мобільний застосунок, який допомагає користувачам знаходити та бронювати заняття в спортивних клубах, а також відстежувати свій прогрес. Застосунок доступний для iOS та Android.

Функціональні можливості застосунку включають:

- Пошук та бронювання занять: Користувачі можуть шукати заняття за типом, тренером, часом, ціною та іншими критеріями.
- Відстеження прогресу: Користувачі можуть відстежувати свої тренування, калорії, пройдену відстань та інші показники.
- Придбання абонементів: Користувачі можуть купувати абонементи на спортивні зали та інші послуги безпосередньо в застосунку.
- Отримання нагадувань: Користувачі можуть отримувати нагадування про майбутні заняття.
- Перегляд розкладу: Користувачі можуть переглядати розклад занять свого спортивного клубу.

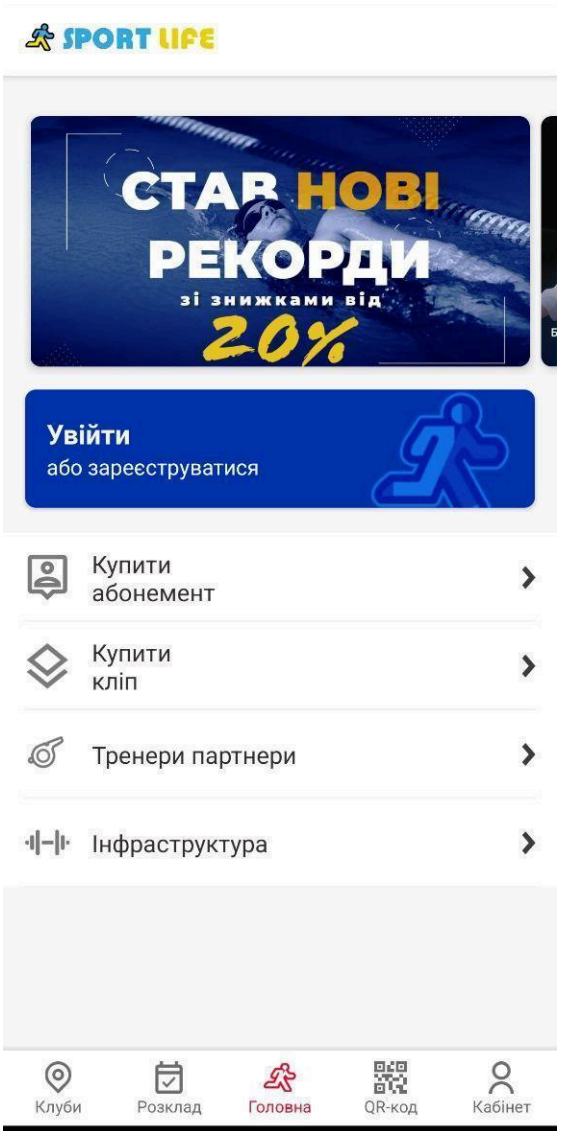


Рис. 1.6 Головне меню застосунку SportLife

Пусті сірі поля в застосунку (рис. 1.6) можуть створювати враження незавершеності та недбалості, що може вплинути на відчуття користувача щодо якості продукту. Коли поля не містять відповідної інформації або функціональності, користувачі можуть відчувати замішання та розчарування.

Також вони можуть також викликати плутанину у користувачів, що може призвести до погіршення їхнього досвіду використання застосунку. Це особливо важливо для важливих функцій або даних, які користувач очікує побачити. Відсутність інформації або можливостей у пустих полях може також знизити

корисність застосунку та зменшити його привабливість для користувачів.

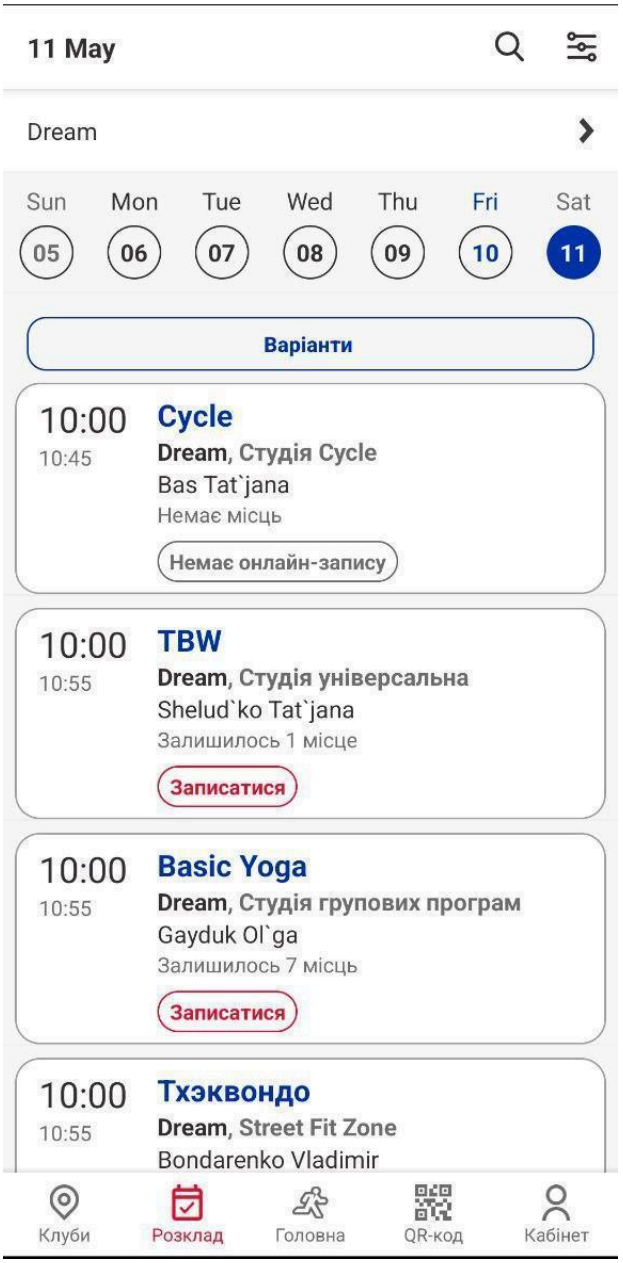


Рис. 1.7 Розклад в застосунку SportLife

Застосунок має проблему повільного завантаження сторінок, що може стати серйозною проблемою для користувачів. Коли сторінка завантажується довго, це може призвести до втрати зацікавленості користувача та навіть до того, що він покине застосунок. Крім того, повільне завантаження може

призвести до збільшення рівня стресу в користувачів, що в свою чергу може вплинути на їхній загальний досвід використання сервісу. В результаті, довге завантаження може вплинути на репутацію та ефективність застосунку, а також на його конверсію та витрати.

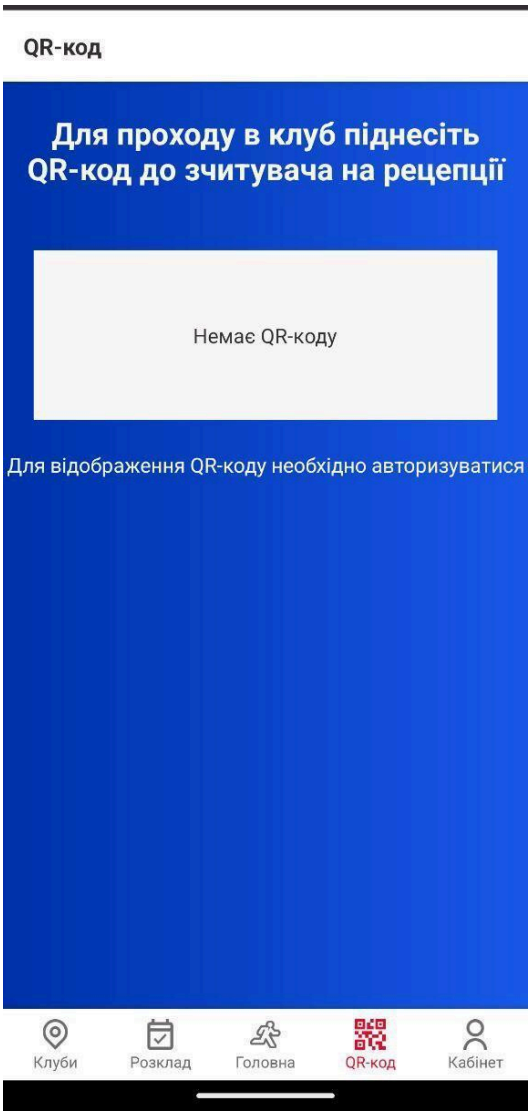


Рис. 1.8 Екран зчитування QR-коду

Яскраві та насичені кольори в застосунку можуть створювати проблеми для користувачів. Надмірна використання різких кольорів може викликати візуальне перевантаження, що робить важчим сприйняття інформації та навігацію для користувача. Крім того, яскраві кольори можуть бути відволікаючими, зменшуючи зосередженість і здатність користувача до

ефективної взаємодії з застосунком. В результаті, недбале використання яскравих кольорів може вплинути на користувацький досвід, зробити застосунок менш привабливим та знизити загальну ефективність його використання. (Рис. 1.8)

1.1.4 Огляд застосунку ClassPass

ClassPass – це застосунок-абонемент, який надає користувачам доступ до різних фітнес-клубів і студій. Він доступний у багатьох містах по всьому світу.

Переваги ClassPass:

1. Різноманіття: ClassPass надає доступ до широкого спектру фітнес-занять, включаючи йогу, пілатес, силові тренування, кардіо та багато іншого.
2. Зручність: ClassPass дозволяє користувачам спробувати різні фітнес-клуби та студії, не купуючи окремі членства.
3. Економія: ClassPass може бути більш економним варіантом, ніж членство в одному фітнес-клубі, якщо ви ходите на багато занять на тиждень.
4. Мотивація: ClassPass може допомогти користувачам залишатися мотивованими, пропонуючи їм нові та цікаві заняття.

Однак, незважаючи на переваги, застосунок та платформа не працюють в Україні.

Переваги застосунку:

Чистий та привабливий дизайн: Головна сторінка має чистий і привабливий дизайн, який сприяє зосередженню уваги користувача і не перенавантажує його зайвою інформацією.

Чітка навігація: Важливі елементи навігації, такі як меню, пошукове

поле або кнопки для доступу до ключових функцій, легко доступні та зрозумілі для користувача.

Корисний вміст: Головна сторінка містить корисний вміст, який відповідає потребам користувачів.

Персоналізований підхід: Інформація та рекомендації на головній сторінці персоналізовані відповідно до індивідуальних потреб та інтересів користувача, що зробить застосунок більш привабливим та корисним для нього.

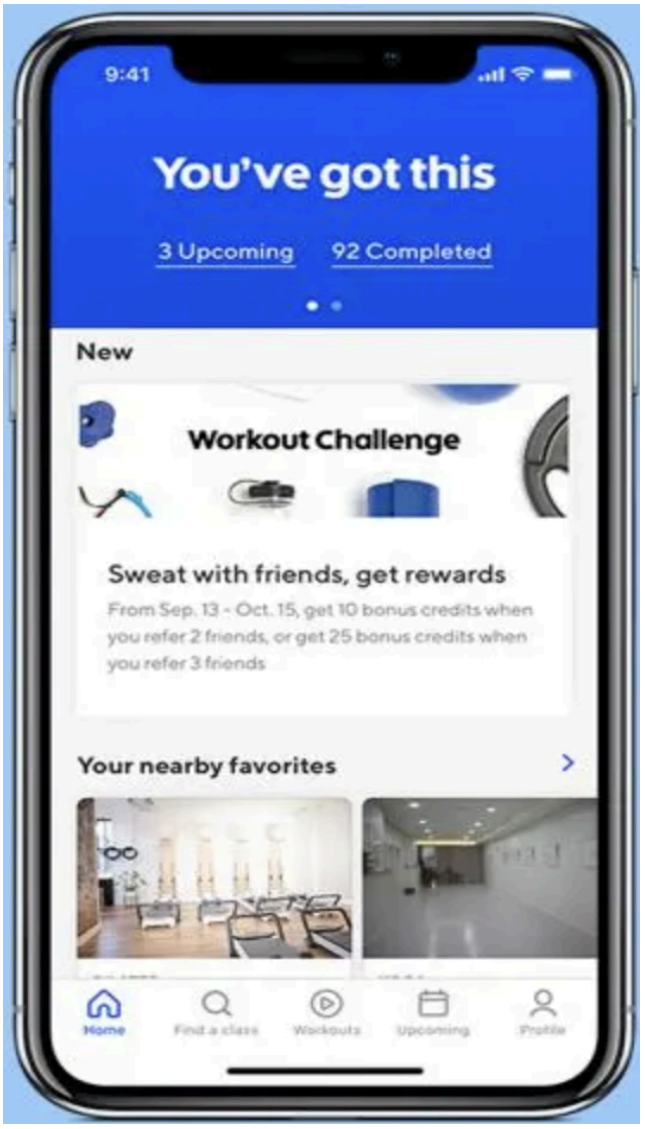


Рис. 1.9 Головна сторінка застосунку ClassPass

Таблиця 1.1

Зведена таблиця з характеристиками існуючих українських застосунків

Критерії оцінювання	**Apolo Next**	**Altegio**	**SportLife**	**ClassPass**	**Fight Academy**
Мови застосунку	Англ.	Укр.	Укр.	Англ.	Англ.
Каталог занять	+	+	+	+	+
Перегляд тренерів	+	+	+	+	+
Фільтрація по типу заняття	-	+	-	+	+
Фільтрація по тренеру	+	+	+	+	+
Фільтрація по даті	+	-	+	-	+
Історія відвідувань	+	+	+	-	+
Не потребує стабільного інтернет зв’язку	+	-	+	+	+
Не потребує помісячну оплату сервіса	+	-	+	-	+
Синхронізація між пристроями	+	+	+	+	+
Сучасний та інтуїтивно зрозумілий дизайн	-	-	-	+	+
Придбання абонементів	-	+	+	+	+
Сповіщення для про існуючий запис	-	+	+	+	+
Нативно інтегрована навігація	-	-	+	+	+

Наведена вище таблиця з характеристиками існуючих українських та міжнародних застосунків, для запису оналайн, надає корисний огляд доступних на ринку рішень. Вона дозволяє порівняти різні застосунки за їх функціональністю, дизайном та зручністю користування.

На основі аналізу таблиці можна зробити наступні спостереження:

- Спочатку, важливо зауважити, що на ринку українських застосунків для онлайн запису панує висока конкуренція. Існує велика кількість різноманітних застосунків, які пропонують схожі функціональні можливості.
- Далі, серед найбільш популярних функцій можна виділити можливість вибору тренера, типу тренування та перегляду історії відвідувань.
- Нарешті, варто відзначити, що дизайн застосунків різний. Деякі з них мають сучасний та елегантний дизайн, тоді як інші відрізняються більш простим та менш зрозумілим виглядом.

2 ТЕХНІЧНЕ ЗАВДАННЯ ТА ВИМОГИ

2.1 Технічне завдання

Для забезпечення зручності користувачів і ефективного управління тренуваннями у FightAcademy буде розроблений застосунок для платформи Android. Мобільний застосунок - це програмне забезпечення, яке встановлюється на смартфони та планшети з операційною системою Android, надаючи користувачам зручний спосіб доступу до сервісу.

Розробка Android застосунку для FightAcademy включає в себе вибір оптимальних технологій та інструментів розробки, які забезпечать найвищу якість та продуктивність застосунку. Застосунок буде розроблений з використанням мови програмування Kotlin, яка є офіційною мовою для розробки Android застосунків та відома своєю ефективністю та чистотою коду.

Основні функції застосунку включатимуть у себе можливість швидко записатися на тренування, переглянути розклад тренувань та доступність місць, а також отримувати сповіщення про створений запис. Завдяки мобільному застосунку користувачі матимуть зручний та швидкий доступ до всіх функцій FightAcademy прямо зі свого смартфона, що підвищить зручність та доступність сервісу.

2.2 Функціональні вимоги

Створення веб-застосунку для FightAcademy, який дозволить реєструватися на тренування та керувати цими записами, вимагає детального визначення умов для забезпечення його успішної та ефективної роботи. Після аналізу недоліків та відгуків користувачів щодо наявних рішень можна визначити основні вимоги для веб-застосунку FightAcademy.

2.2.1 Валідація введеного користувачем паролю

Валідація паролю при реєстрації передбачає перевірку введеного користувачем паролю на дотримання певних критеріїв безпеки та надійності. Це необхідно для запобігання використанню слабких паролів, які можуть стати об'єктом зламу або несанкціонованого доступу до облікового запису. Основні етапи функціональної вимоги валідації паролю включають:

Мінімальна довжина паролю: Система перевіряє, чи не менше заданої мінімальної кількості символів у паролі. Ця кількість становить не менше 8 символів.

Складність паролю: Система перевіряє, чи в паролі присутні символи різних типів, такі як великі та малі літери, цифри та спеціальні символи. Пароль має вимагати наявності принаймні однієї великої літери, однієї малої літери та однієї цифри.

2.2.2 Запис на тренування

Функціональна вимога запису на тренування передбачає можливість користувачам системи FightAcademy обирати та реєструватися на доступні тренування через застосунок. Основні аспекти цієї функціональної вимоги включають:

Перегляд розкладу тренувань: Користувачам надається можливість переглядати розклад доступних тренувань на місяць.

Вибір тренування: Користувач може обирати бажане тренування з переліку доступних опцій, враховуючи дату, тренера і тип тренування.

Реєстрація на тренування: Після обрання бажаного тренування користувач може здійснити реєстрацію на нього, натиснувши на відповідну кнопку. У разі успішної реєстрації користувач отримує підтвердження та додаткову інформацію про тренування.

2.2.3. Перегляд історії записів

Користувачам доступний перегляд їхньої історії записів на тренування, де вони можуть переглянути вже здійснені реєстрації, їх статус та ціну тренування.

2.2.4. Відправлення нагадувань через сповіщення

Система має автоматично надсилати користувачам нагадування за певний час до початку тренування. Ці нагадування будуть у вигляді сповіщень, які з'являються на екрані користувача, надихаючи його на виконання запланованих цілей та тренувань.

2.2.5. Функціональна вимога логіну

Вимога передбачає можливість аутентифікації користувачів у системі FightAcademy для забезпечення безпеки та доступу до персоналізованих функцій. Основні аспекти цієї функціональної вимоги включають:

1. Форма входу: Система повинна надавати форму для введення ідентифікаційних даних, таких як номер телефону та пароль.
2. Перевірка ідентифікаційних даних: Після введення ідентифікаційних даних система повинна перевіряти їхню правильність у базі даних. Якщо вони правильні, користувачу надається доступ до системи.
3. Обробка помилок: У разі неправильного введення ідентифікаційних даних система повинна повідомляти користувача про помилку.
4. Запам'ятовування сесії: Після успішного входу користувач повинен мати можливість залишатися в застосунку протягом тривалості сесії без потреби повторного входу.
5. Вихід з облікового запису: Система повинна надавати користувачеві можливість безпечного виходу з облікового запису для забезпечення конфіденційності та безпеки даних.

2.2.6. Функціональна вимога реєстрації

Функціональна вимога реєстрації передбачає можливість користувачам створювати облікові записи в системі FightAcademy. Основні аспекти цієї функціональної вимоги включають:

1. Форма реєстрації: Система має надавати форму для введення необхідних даних для реєстрації, таких як ім'я, прізвище, адреса електронної пошти, пароль тощо.
2. Перевірка унікальності даних: Під час реєстрації система повинна перевіряти унікальність введених користувачем даних, таких як адреса електронної пошти, щоб уникнути конфліктів облікових записів.
3. Підтвердження облікового запису: Після введення даних користувач повинен отримати підтвердження про успішну реєстрацію, наприклад, шляхом відправки листа на вказану електронну пошту з посиланням для активації облікового запису.
4. Захист паролю: Система повинна надавати можливість встановлення надійного пароля для захисту облікового запису користувача.
5. Обробка помилок: У разі неправильного введення даних система повинна повідомляти користувача про помилку і надавати можливість їх змінити або виправити.
6. Підтвердження реєстрації: Після успішної реєстрації користувач повинен мати можливість входу до свого облікового запису.

2.3. Нефункціональні вимоги

Нефункціональні вимоги важливі, оскільки вони визначають якість та ефективність системи в цілому, враховуючи аспекти, що не є прямо пов'язаними з її функціональністю. Ці вимоги допомагають забезпечити те, що система відповідає потребам користувачів та відповідає стандартам якості.

Ці вимоги, які характеризують атрибути системи, такі як її продуктивність, надійність, безпека, доступність та інші аспекти, які не стосуються конкретних

функцій програмного забезпечення, але впливають на його загальну ефективність та якість.

2.3.1. Надійність та стійкість

При введенні неправильних даних або команд, система FightAcademy має продовжувати свою роботу і правильно реагувати на запити користувачів. У випадку критичної помилки система повинна інформувати користувача про виниклий збій. Коли застосунок раптово втратить зв'язок, користувач має бути попередженим.

2.3.2. Зрозумілість інтерфейсу

Користувач повинен мати зручний доступ до функціоналу застосунку FightAcademy. Інтерфейс і команди мають бути дизайнерсько оновлені та легкі для сприйняття, щоб користувач з легкістю міг ознайомитися з застосунком і почати його використання без зайвих зусиль.

2.3.3. Масштабування системи

Середовище застосунка повинно бути гнучким щодо включення нових сервісів у систему без потреби внесення змін у наявний код або конфігурацію.

2.4. Системні вимоги

Операційна система: Android 6.0 (Marshmallow) або вище.

Оперативна пам'ять: Мінімум 2 ГБ ОЗУ для оптимальної продуктивності.

Внутрішня пам'ять: Мінімум 100 МБ вільного місця на пристрої для встановлення застосунку.

Роздільна здатність екрану: Рекомендовано HD (1280x720 пікселів) або вище

Інтернет-підключення: Для доступу до онлайн функцій та оновлень застосунку.

3 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

3.1 Середовище розробки JetBrains Rider

JetBrains Rider - інтегроване середовище розробки, спрямоване на платформу .NET та різноманітні мови програмування, такі як C#, F#, і VB.NET. Відзначається високою продуктивністю та швидкістю розробки завдяки розширеному набору інструментів та підтримці мультиплатформенності. Rider надає повноцінну підтримку для .NET Core, включаючи можливість розробки під Linux і macOS, з вбудованим дебагером, який підтримує практично всі типи .NET-додатків, таких як ASP.NET, Unity, і Xamarin.

Серед ключових функцій Rider варто відзначити розширену підтримку рефакторингу коду, автозавершення кодування, інтеграцію з іншими інструментами розробки, такими як git або SVN для систем контролю версій, а також підтримку різних мов програмування в одному проекті. Крім того, середовище розробки дозволяє працювати з базою даних безпосередньо з проекту, не потребуючи окремого додатку чи плагіну, що значно спрощує роботу зі сховищем даних під час розробки.

3.2 Середовище розробки Android Studio

Android Studio - це інтегроване середовище розробки для платформи Android, розроблене компанією Google. Воно надає розширений набір інструментів для ефективної розробки Android-застосунків, включаючи редагування коду, візуальний дизайн і тестування. Студія спрощує розробку за допомогою вбудованих шаблонів, автозавершення коду, інтегрованих інструментів для відлагодження, а також підтримки різних мов програмування, таких як Java і Kotlin.

Однією з ключових переваг Android Studio є можливість створення різних типів застосунків - від простих одноекранних застосунка до складних мультиплатформних проектів. Вона також має інтегровану систему керування версіями для спільної роботи над проектами та спрощення відстеження змін.

Додатково, Android Studio підтримує роботу з різними пристроями та версіями Android, що дозволяє розробникам перевіряти сумісність своїх застосунків з широким спектром пристроїв. Вона також має вбудовані інструменти для оптимізації застосунків під час розробки, що допомагає підвищити продуктивність та швидкість розробки.

3.3 Мова програмування – C#

Широкі можливості розробки: C# є однією з основних мов програмування для платформи .NET, що відкриває широкі можливості розробки для різних типів застосунків, включаючи веб-сервіси, мобільні застосунки, ігри та багато іншого.

Велика екосистема: C# має велику екосистему, яка включає різноманітні бібліотеки, фреймворки та інструменти розробки. Це дозволяє розробникам швидко створювати функціональні та ефективні сервіси, використовуючи готові рішення.

Висока продуктивність: C# володіє високою продуктивністю завдяки своєму компільованому характеру та оптимізованому виконанню коду. Це дозволяє створювати швидкі та ефективні сервіси, які працюють надійно навіть у великих навантажених середовищах.

Інтеграція з платформою .NET: Обираючи C#, ви отримуєте доступ до багатьох рішень та сервісів, що надаються платформою .NET, таких як ASP.NET для веб-розробки, WCF для створення служб, та інші.

3.4 ASP.NET - Фреймворк для побудування API

ASP.NET є фреймворком для розробки веб-застосунків, включаючи веб-сайти, веб-служби та веб-застосунки. Однією з ключових можливостей ASP.NET є його здатність побудувати API (інтерфейси програмування застосунків) шляхом використання різних технологій, зокрема ASP.NET Core.

ASP.NET дозволяє розробникам створювати API для взаємодії з різними клієнтами, такими як веб-сайти, мобільні застосунки, десктопні програми та інші служби.

Маршрутизація: ASP.NET надає механізми для визначення маршрутів, які вказують, які URL-адреси мають відповідати конкретним діям у вашому API. Це дозволяє визначати логіку обробки запитів на основі шляхів URL.

Модуль REST: Багато API, побудовані на ASP.NET, використовують архітектурний стиль REST (представлення стану передачі), що дозволяє створювати легкі та ефективні інтерфейси для взаємодії з клієнтами.

Серіалізація даних: ASP.NET надає можливості для серіалізації та десеріалізації даних у різних форматах, таких як JSON або XML. Це дозволяє обмінюватися даними з клієнтами у зручному для них форматі.

Автентифікація та авторизація: ASP.NET надає засоби для реалізації систем аутентифікації та авторизації, які дозволяють контролювати доступ до різних ресурсів у вашому API.

Інтеграція з іншими технологіями: ASP.NET може бути легко інтегрований з іншими технологіями та сервісами, такими як бази даних, служби розсилання повідомлень, системи кешування та багато іншого.

Основна мета створення API на ASP.NET полягає в забезпеченні ефективної, безпечної та надійної взаємодії між різними додатками та сервісами. Це дозволяє створювати розширені та сучасні рішення, що відповідають потребам сучасного ринку програмного забезпечення.

3.5 Entity Framework (EF)

Entity Framework (EF) - це об'єктно-орієнтований фреймворк для доступу до даних в програмах .NET. Він дозволяє розробникам взаємодіяти з базами даних за допомогою об'єктно-орієнтованого підходу, що спрощує процес роботи з даними.

Entity Framework дозволяє використовувати об'єктно-орієнтований підхід при роботі з базою даних. Замість того, щоб взаємодіяти з таблицями та SQL-запитами, розробник може працювати з класами та об'єктами, що відображають дані з бази даних.

Entity Framework автоматично створює відображення між класами програми та таблицями бази даних. Це означає, що розробникам не потрібно вручну писати SQL-запити для вибору, вставки, оновлення чи видалення даних - Entity Framework доглядає за цим за них.

Entity Framework надає зручний інтерфейс для виконання операцій з базою даних через мову запитів LINQ (Language Integrated Query). Це дозволяє розробникам писати запити до бази даних, використовуючи звичайний синтаксис C#, що робить код більш зрозумілим та підтримує розуміння.

3.6 Kotlin та корутини

Kotlin - це сучасна мова програмування, яка набула великої популярності серед розробників Android за останні кілька років. Вона розроблена компанією JetBrains і має багато переваг для розробки Android-застосунків.

Корутини - це механізм управління потоками виконання, що дозволяє програмі виконувати велику кількість завдань швидше та ефективніше. Ось кілька переваг корутин:

Корутини дозволяють легко створювати та керувати багато потоками виконання без необхідності великої кількості ресурсів. Вони дозволяють розпаралелювати виконання завдань та ефективно керувати ними.

У порівнянні зі створенням нових потоків, що може призвести до збільшення накладних витрат через збільшення витрат пам'яті та ресурсів, корутини мають менші накладні витрати. Вони використовують менше ресурсів, оскільки керування потоками відбувається на рівні програми, а не операційної системи.

Корутини дозволяють ефективно використовувати ресурси, такі як процесорний час та оперативна пам'ять. Вони дозволяють програмі зосередитися на виконанні активних завдань, уникнувши зайвої простоювання через блокування або очікування.

3.7 Kotlin Flow

Коли мова йде про Kotlin, "флоу" (Flow) - це компонент, що вводиться в Kotlin для роботи з асинхронним програмуванням та реактивним програмуванням.

Асинхронна обробка даних: Флоу дозволяє асинхронно отримувати та обробляти дані. Він дозволяє виконувати довгі операції без блокування основного потоку виконання, що дозволяє застосунку залишатися реактивним та ефективним.

Підтримка корутин: Флоу інтегрується з корутинами в Kotlin, що робить його досить потужним інструментом для асинхронного програмування. Ви можете легко використовувати флоу разом з корутинами для створення ефективних та елегантних рішень.

Безпека типів: Флоу має безпеку типів за замовчуванням, що допомагає уникнути помилок під час компіляції. Це забезпечує більшу надійність та стабільність вашого коду.

Легка інтеграція з іншими компонентами Kotlin: Флоу легко інтегрується з іншими компонентами Kotlin, такими як корутини, колекції та інші. Це дозволяє розробникам писати компактний та ефективний код без додаткових зусиль.

4 ПРОЕКТУВАННЯ ТА РОЗРОБКА

4.1 Проектування діаграми використання

Діаграми використання сприяють з'ясуванню контексту та потреб всієї системи чи її частин, і демонструють взаємодію між системою та особами, які з нею взаємодіють. Тут особа - це будь-яка участь у взаємодії з системою. Ці діаграми використання розкривають функціональність та область застосування системи, і акцентують на тому, як саме учасники з нею взаємодіють та які дії вони здійснюють.

Варіанти використання також можуть бути корисними на етапі збору вимог, аналізу, проектування та тестування системи. Вони допомагають визначити класи, необхідні системі, розробити тести та інші важливі аспекти розробки. Проте варто пам'ятати, що ці діаграми не надають інформації про те, як саме система працює всередині.

В даному застосунку Актором буде виступати застосунок-клієнт, який працювати із Android застосунком. Окрім акторів треба визначити основні варіанти використання системи:

- Робота з аутентифікацією
- Перегляд історії візитів
- Перегляд тренерів
- Планування візиту
- Перегляд інформації про фітнес центр

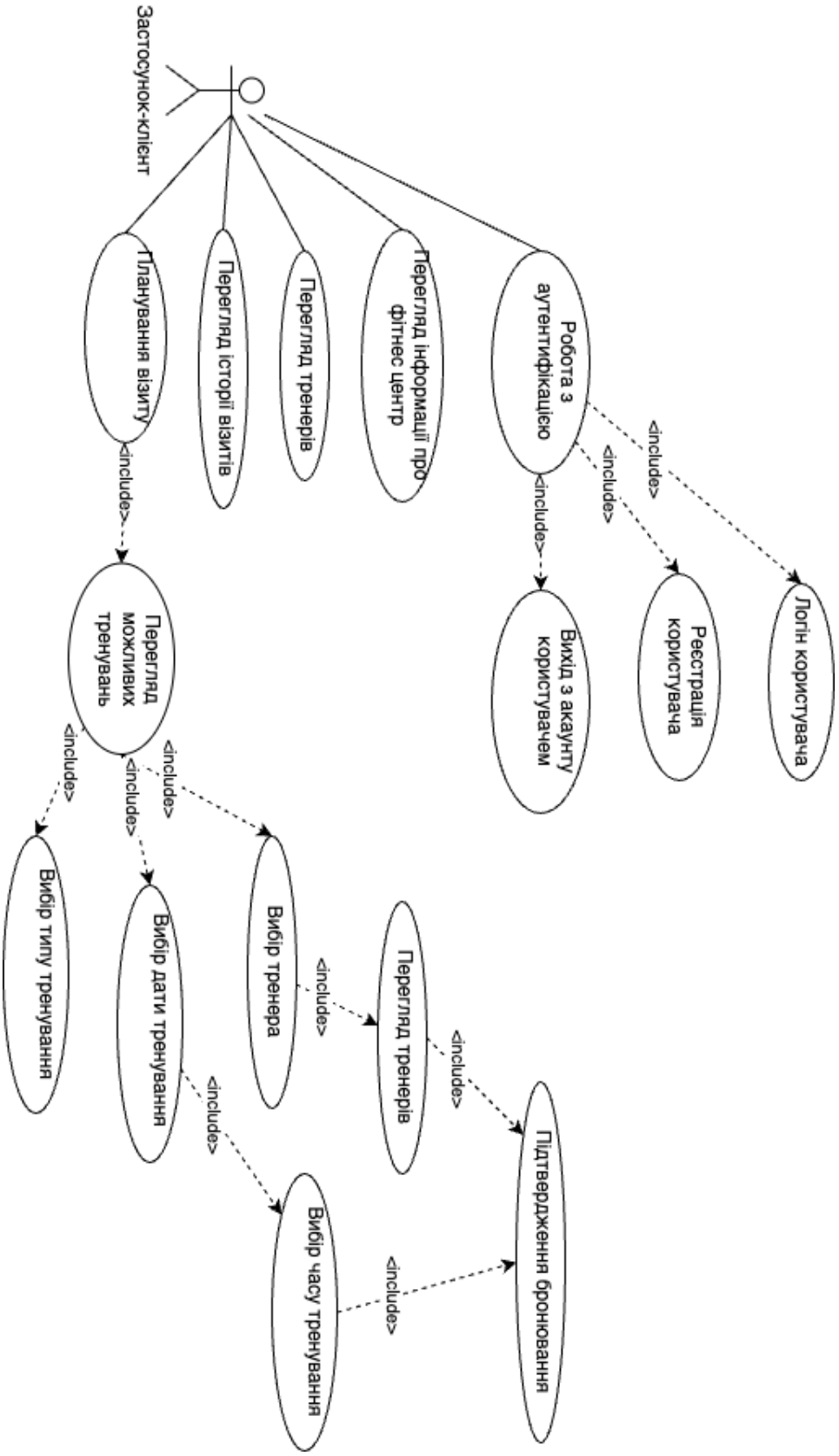


Рис. 4.1 Use Case діаграма Android застосунку FightAcademy

У цій діаграмі відображено всі ключові можливості застосунку "FightAcademy", доступні для користувачів через мобільний застосунок. Основною функцією є взаємодія з даними за допомогою API-ендпоінтів, що дозволяє користувачам отримати доступ до різноманітних функцій, пов'язаних з академією бою.

4.2 Проектування внутрішньої архітектури сервісу

Для розробки сервісу фітнес центру було обране архітектурне підходу DDD (Domain-Driven Design). DDD - це методологія розробки програмного забезпечення, яка ставить доменну модель у центр уваги. Згідно з DDD, програма розділяється на різні контексти, які відображають різні доменні області. Вони включають у себе доменну модель, яка описує ключові концепції, правила та логіку бізнесу. DDD включає такі ключові компоненти:

1. Доменна модель: відображає основні концепції, правила та обмеження бізнесу.
2. Сервіси застосунку: відповідають за виконання конкретних функцій в доменній моделі.
3. Репозиторії: забезпечують доступ до доменних об'єктів у базі даних.

DDD дозволяє зосередитися на ключових аспектах бізнесу та розробляти програмне забезпечення, яке краще відповідає потребам користувачів. Графічний приклад архітектури DDD зображено на рисунку 4.3.

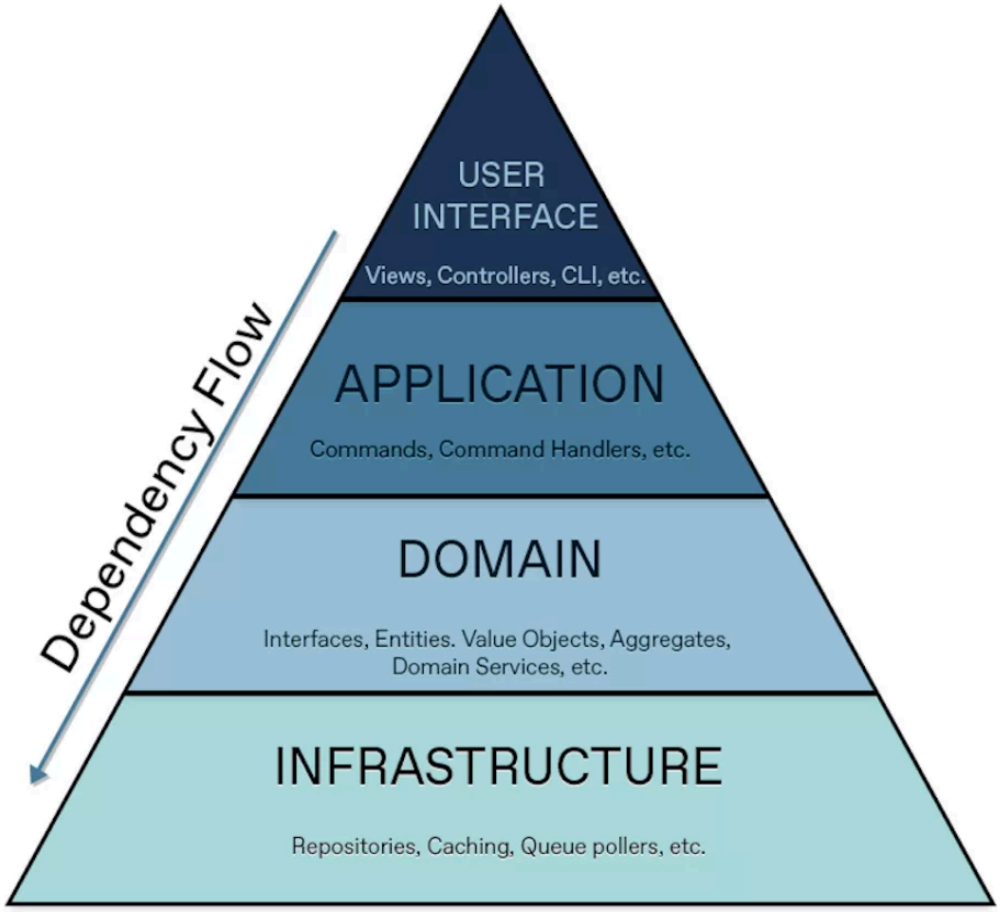


Рис. 4.3 Графічний вигляд архітектури DDD

Переваги архітектури Domain-Driven Design (DDD) починаються з акцентування уваги на доменній моделі, що дозволяє краще відтворити ключові концепції та правила бізнесу у програмному забезпеченні. Це сприяє зменшенню розбіжностей між мовою бізнесу та програмним кодом.

DDD дозволяє відділити доменну логіку від інфраструктурного коду, що забезпечує більшу гнучкість та розширюваність системи. Крім того, завдяки активному взаємодії з експертами з домену, DDD допомагає краще зрозуміти бізнес-потреби та створити більш ефективні рішення.

DDD також стимулює використання мови програмування та архітектурних патернів, що найбільш відповідають доменним вимогам. Це сприяє створенню більш ефективного та оптимізованого програмного забезпечення.

4.3 Проектування діаграми залежностей сервісу FightAcademy.Client

Діаграма залежностей відображає взаємозв'язки між компонентами системи та показує, які компоненти використовують інші для виконання своїх функцій. У вашому випадку:

Контролер: Це компонент, який приймає запити від користувача та взаємодіє з іншими компонентами для обробки цих запитів. Контролер отримує дані від користувача, обробляє їх та викликає відповідні методи в сервісі для виконання бізнес-логіки.

Сервіс: Це компонент, який містить бізнес-логіку застосунку. Сервіс отримує запити від контролера, обробляє їх, виконує необхідні операції з даними та повертає результат контролеру. У вашому випадку, сервіс взаємодіє з базою даних, виконуючи запити на отримання, оновлення, створення або видалення даних.

База даних: Це місце, де зберігаються дані застосунку. Сервіс взаємодіє з базою даних для отримання необхідної інформації або зберігання нових даних, відповідно до запитів, які надходять від контролера.

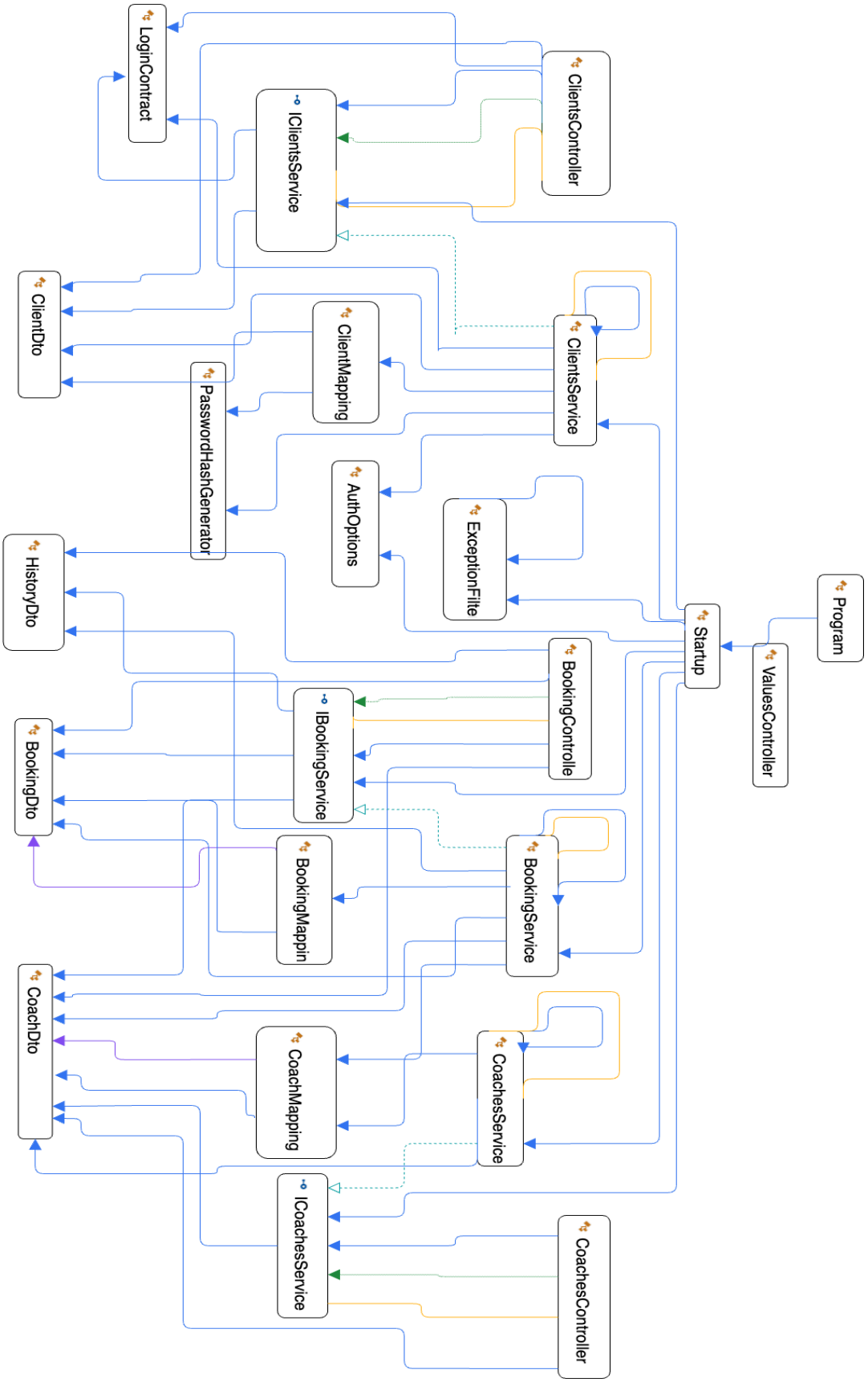


Рис. 4.4 Схема залежностей FightAcademy.Client

Для прикладу розглянемо UML діаграму роботи сервісу бронювання. В якому BookingController використовує IBookingService.

Контролери та сервіси - це ключові компоненти в архітектурі застосунку, що виконують різні функції.

Контролери взаємодіють з зовнішнім світом, таким як клієнти (наприклад, веб-браузери або мобільні застосунку), приймаючи HTTP-запити. Вони обробляють ці запити, взаємодіючи з сервісами для отримання необхідної інформації або виконання певних операцій.

Контролери відповідають за перенаправлення потоку управління, обробку вхідних даних, валідацію та відповідь на клієнтські запити.

Сервіси містять бізнес-логіку застосунку, вони виконують конкретні операції, які вимагають обробки даних або взаємодії з базою даних. Вони можуть виконувати операції, такі як обчислення, перетворення даних, валідація, взаємодія з іншими службами або збереження/отримання даних з бази даних. Сервіси незалежні від методу взаємодії з клієнтами, вони можуть бути викликані контролерами або іншими сервісами. Також сервіси взаємодіють з базою даних для отримання, збереження, оновлення або видалення даних.

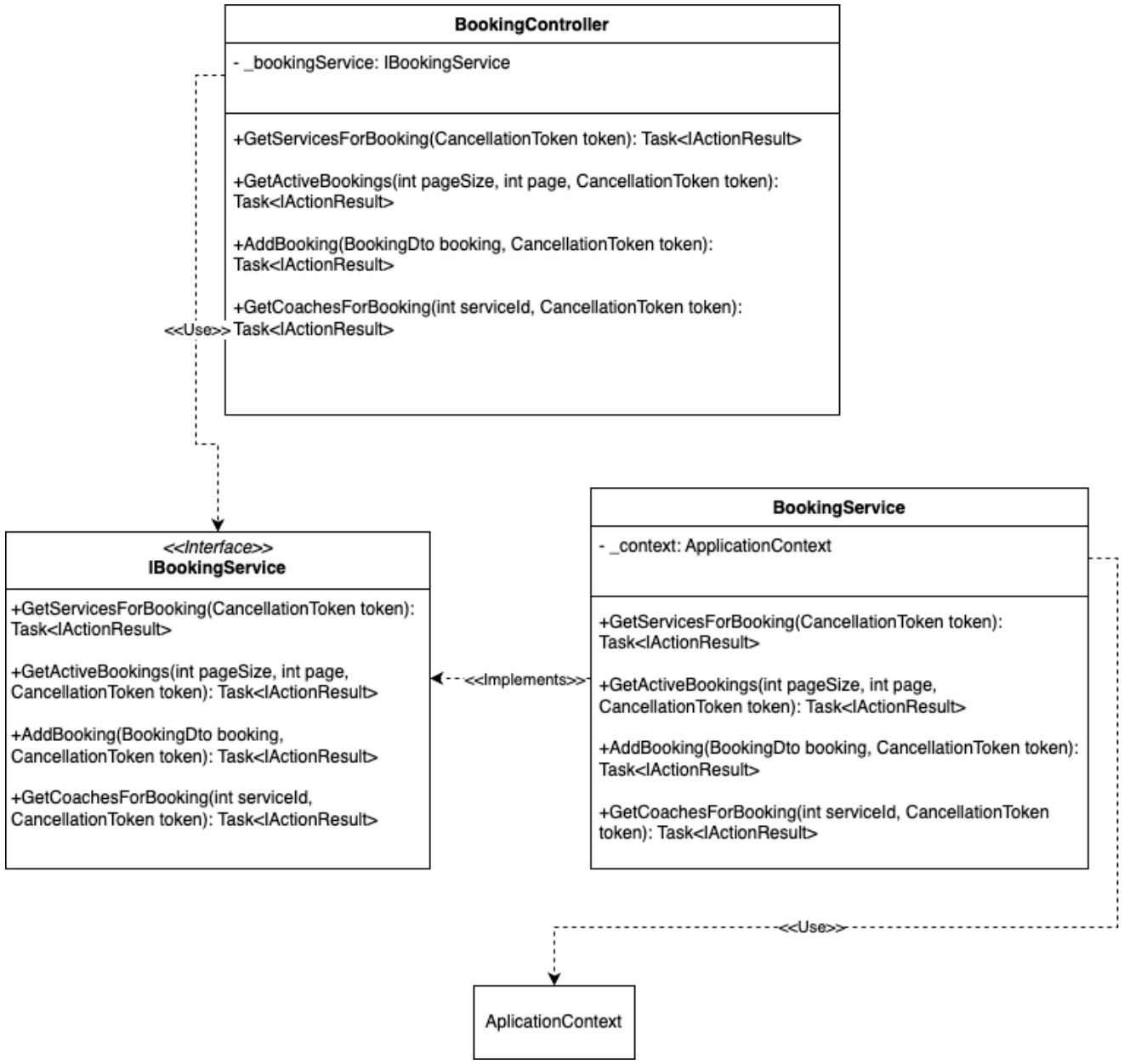


Рис. 4.5 Схема BookingController

4.4 Робота контролерів

Методи контролерів викликаються при надходженні HTTP-запитів від клієнтів. Коли клієнт (наприклад, веб-браузер або мобільний застосунок) надсилає HTTP-запит на сервер, серверний застосунок визначає, який контролер та метод повинні бути викликані для обробки цього запиту.

Виклик методів контролерів відбувається наступним чином:

Маршрутизація: Серверний застосунок аналізує URL-адресу HTTP-запиту та визначає, який контролер та метод повинні бути викликані для обробки запиту. Цей процес відомий як маршрутизація.

Створення екземпляра контролера: Після визначення контролера, сервер створює його екземпляр. Це може відбуватися шляхом виклику конструктора контролера або з використанням пула об'єктів для покращення продуктивності.

Виклик методу контролера: Після створення екземпляра контролера сервер викликає відповідний метод контролера для обробки запиту. Аргументи методу можуть бути отримані з параметрів запиту, таких як шлях URL, параметри запиту або тіло запиту (для POST-запитів).

Обробка запиту та формування відповіді: Метод контролера виконує необхідні операції для обробки запиту, такі як взаємодія з сервісами, обробка даних, валідація тощо. Після цього він формує відповідь, яка може бути відправлена клієнту. Відповідь містить JSON-дані для відправки даних до клієнта або інші формати відповідно до потреб.

4.5 Підключення та робота з PostgeSQL

Для роботи з базою даних через Entity Framework Core, використовується підхід з використанням "контексту застосунку" (Application Context).

Визначені класи, які відповідають сутностям потрібної бази даних. Ці класи відображають таблиці бази даних і містять властивості, що відповідають стовбцям таблиць.

Створено клас, який наслідується від класу DbContext. Цей клас буде представляти контекст застосунку, який відповідає базі даних. У цьому класі вказано набір властивостей типу DbSet<T>, де T - це ваші моделі даних. Ці властивості представляють таблиці бази даних.

У файлі appsettings.json або в конфігураційних файлах визначено рядок підключення до бази даних.

У застосунку використано вбудований контейнер залежностей для внедрення контексту застосунку. Це робиться в файлі Startup.cs за допомогою методу AddDbContext сервісу IServiceCollection.

Далі це використовується з контексту застосунку в інших класах застосунку. Є можливість створювати, оновлювати, видаляти та запитувати дані з бази даних за допомогою методів, які надає контекст.

Після внесення змін до об'єктів моделей даних ви викликається метод SaveChanges() контексту застосунку, щоб зберегти ці зміни у базі даних.

Цей підхід дозволяє вам легко працювати з базою даних у .NET Core застосунків за допомогою Entity Framework Core, забезпечуючи зручний та потужний спосіб взаємодії з даними.

```
using ...

namespace FightAcademy.Client
{
#pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
    1 usage  R Hramushniak +1
    public class Startup
    {
        R Hramushniak
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        2 usages
        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        R Hramushniak +1
        public void ConfigureServices(IServiceCollection services)
        {
            ////add httpClient
            services.AddHttpClient();

            // add services
            services.AddTransient<ICoachesService, CoachesService>();
            services.AddTransient<IBookingService, BookingService>();
            services.AddTransient<IClientsService, ClientsService>();

            ////add authentication by jwt
            services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
                .AddJwtBearer(options =>
                {
                    options.RequireHttpsMetadata = false;
                    options.TokenValidationParameters = new TokenValidationParameters
                    {
                        ValidateIssuer = true,
                        ValidIssuer = AuthOptions.Issuer,
                        ValidateAudience = true,
                        ValidAudience = AuthOptions.Audience,
                        ValidateLifetime = true,
```

Рис. 4.6 Приклад налаштування Startup класу

4.6 Проектування внутрішньої архітектури застосунку

MVVM (Model-View-ViewModel) — це архітектурний патерн, який в основному використовується в розробці інтерфейсу користувача, зокрема у фреймворках, таких як Android. Він має на меті відокремити інтерфейс користувача (View) від бізнес-логіки (ViewModel) і даних (Model). ViewModel діє як посередник між View і Model, відкриваючи дані та команди, до яких View може прив’язуватися. Це розділення забезпечує кращу тестоздатність, ремонтопридатність і гнучкість.

Мета полягає в тому, щоб розділити відповідальність, зробити її тестованою та уникнути будь-яких сильних залежностей від інтерфейсу користувача, фреймворків і баз даних. Існує можливість змінювати залежність, не впливаючи на всю структуру. ми повинні дотримуватися розділення інтересів . Це допомагає мати ефективний дизайн програми, розділяючи логіку, дані та інтерфейс користувача. Переваги підходу:

1. Реактивна та багаторівнева архітектура.
2. Односпрямований потік даних (UDF) на всіх рівнях програми.
3. Рівень інтерфейсу користувача з власниками стану для керування складністю інтерфейсу користувача.
4. Співпрограми та потоки.
5. Кращі практики впровадження залежностей.

Проект розділений на три рівні: дані, домен та презентація

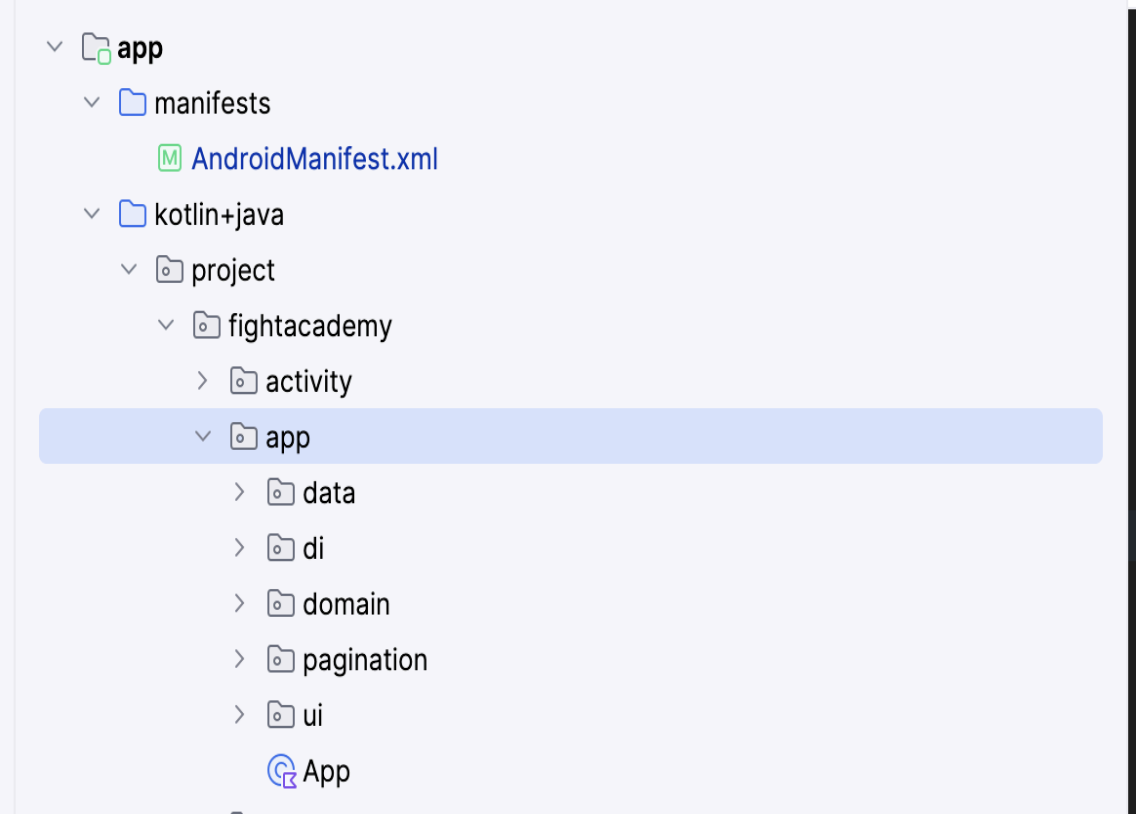


Рис. 4.7 Структура Android застосунку розділена по рівням

4.6 Розробка екрану тренерів в застосунку з архітектурою MVVM

У клієнтському застосунку FightAcademy, який використовує клієнтську архітектуру MVVM (Model-View-ViewModel), екран перегляду тренерів організований наступним чином:

Модель(CoachUiModel) представляє собою дані про тренерів, отримані з сервера, які готові бути відображені на екрані. Вона містить дані, такі як ім'я, фотографія, опис, номер телефону тощо.

Вю-модель(ViewModel) відповідає за управління відображенням даних на екрані та обробку взаємодії користувача. Вона отримує дані з моделі, форматує їх та надає їх до відображення відповідним компонентам інтерфейсу користувача.

Вона також містить логіку, необхідну для виконання операцій, пов'язаних з роботою пагінованого списку.

Юзкейс(UseCase) визначає операції, які можуть бути виконані на цьому екрані, такі як отримання списку тренерів з сервера, отримання детальної інформації про тренера. Він слугує посередником між вьюмоделлю та репозиторієм, де виконуються фактичні операції над даними.

Репозиторій(Repository) відповідає за доступ до даних, які можуть бути отримані з сервера, локальної бази даних або іншого джерела. Він містить методи для виконання операцій з отриманням, збереженням, оновленням та видаленням даних. На екрані перегляду тренерів репозиторій використовується для отримання списку тренерів.

Сервіс(Service) відповідає за виконання інтернет операцій, які необхідно виконати для забезпечення роботи застосунку.

Мапери(Mappers) використовуються для перетворення даних з одного формату в інший, наприклад, з формату, отриманого з сервера, в формат, придатний для відображення на екрані. Вони допомагають розділити логіку мапування даних від логіки відображення даних відповідно до принципу єдиної відповідальності.

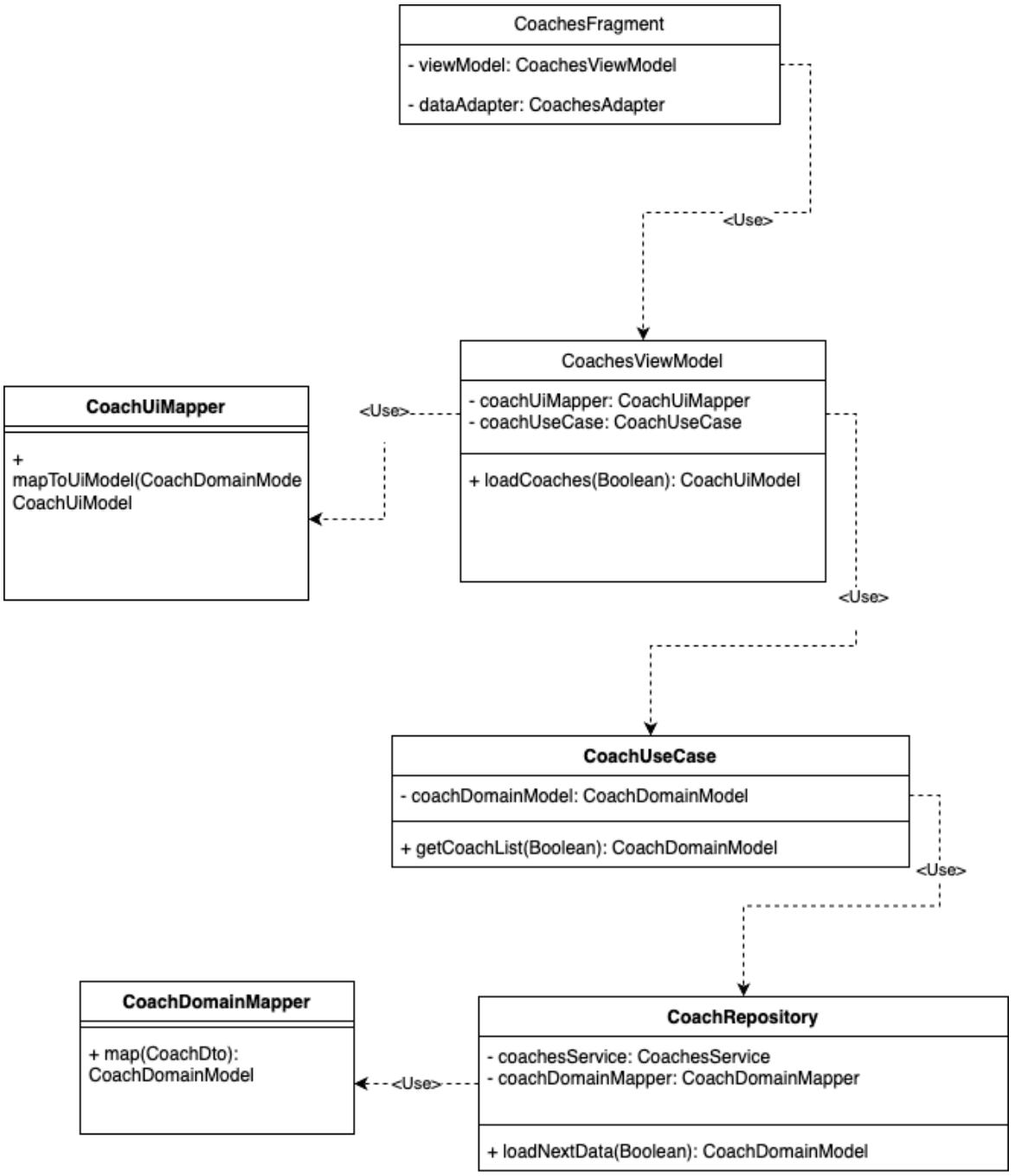


Рис. 4.8 UML діаграма роботи екрану для перегляду тренерів

Хоча архітектура Model-View-ViewModel (MVVM) не єдиний можливий варіант для розробки Android застосунку FightAcademy, вона виявляється особливою відмінністю завдяки своїм перевагам. MVVM пропонує чітке розділення обов'язків між компонентами застосунку, що забезпечує більшу гнучкість та зручність у розробці та підтримці коду.

5 ОГЛЯД РОБОТИ ЗАСТОСУНКУ

Необхідно перевірити відповідність реалізованого Android застосунку FightAcademy вимогам, а також їхню роботоздатність, провести тестування для підтвердження їхньої ефективності та надійності.

5.1 Перевірка роботи реєстрації та автентифікація

Екран вводу логіну та паролю, це перше, що бачить анонімний користувач після завантаження застосунку (рис 3.1). Користувач може увійти в свій акаунт, при коректному вводі номеру телефону та паролю. Або створити новий акаунт, вказавши номер телефону, ім’я та пароль.

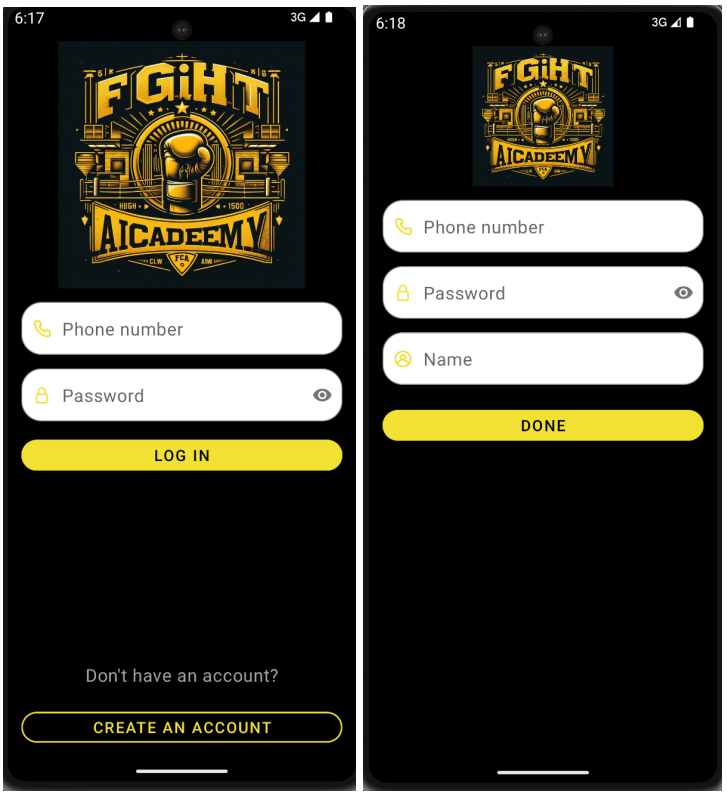


Рис. 5.1 Екран логіну та реєстрації

При вводі неправильного номеру телефону, показується повідомлення про це. Користувач має виправити номер на коректний. При вводі недостатньо міцного паролю, показується повідомлення, що потрібно добавити для його вдосконалення.

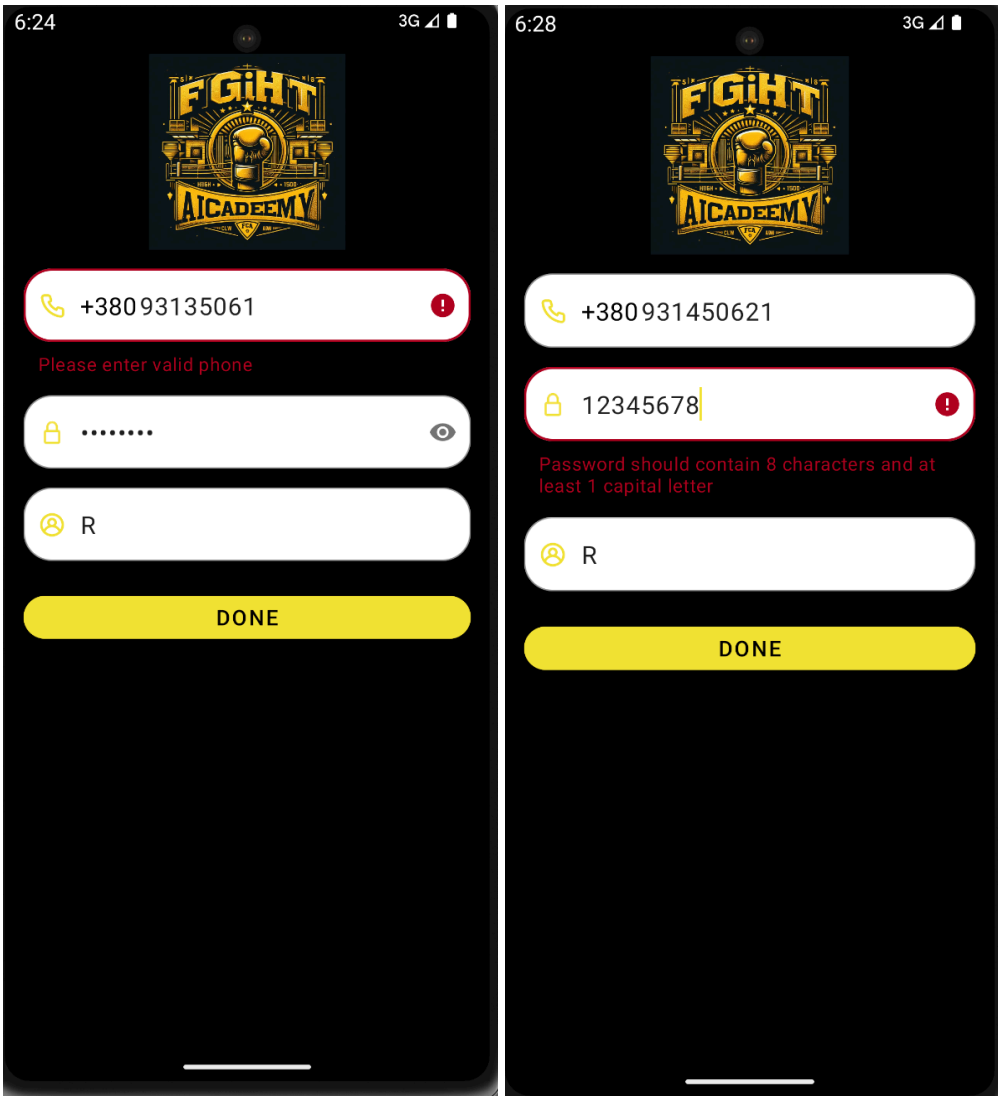


Рис. 5.2 Помилки вводу полів на екрані реєстрації

5.2 Початковий екран

Після вдалого входу в акаунт, користувача вітає головна сторінка, що демонструє всі переваги та особливості, які пропонує фітнес центр FightAcademy. Ця сторінка призначена для враження та мотивації користувачів

здійснювати відвідування спортивного закладу.

На головній сторінці користувач зустрічає динамічний контент, що може бути налаштований через спеціальний сервіс. Цей сервіс дозволяє адміністраторам або власникам фітнес центру змінювати картинки та тексти на головній сторінці в будь-який зручний для них спосіб. Наприклад, вони можуть оновлювати інформацію про акції та знижки, додавати фотографії тренувань або нові послуги, або ж розміщувати мотивуючі цитати та відеоролики.

Такий динамічний підхід до контенту дозволяє тримати інформацію на головній сторінці актуальною та цікавою для клієнтів, що сприяє підвищенню їхньої зацікавленості у відвідуванні фітнес центру та позитивному враженню від нього.

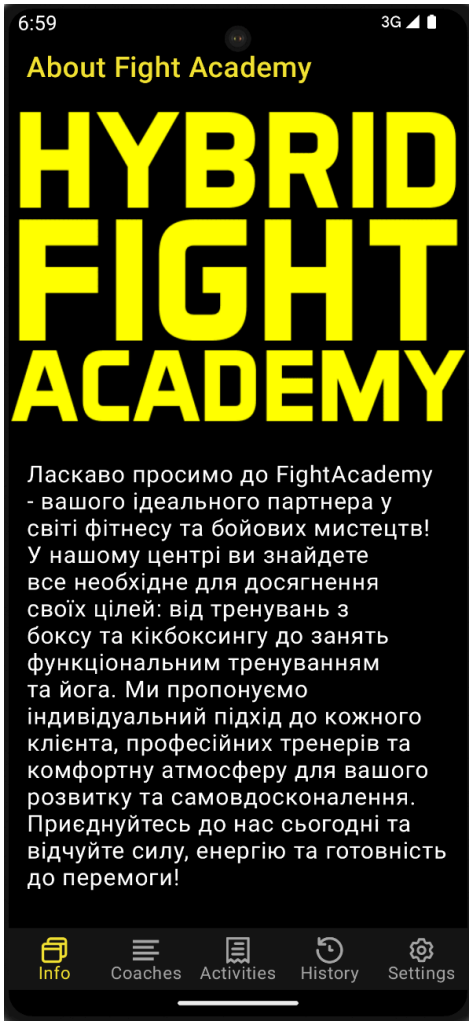


Рис. 5.3 Екран інформації про сервіс

5.3 Екран тренерів

На екрані тренерів користувач отримує можливість переглянути повний список доступних тренерів у FightAcademy. Кожен тренер представлений у вигляді картки, яка містить інформацію про тренера, його фотографію та контактні дані. Крім того, для зручності користувача, на кожній картці тренера також вказаний його номер телефону, за допомогою якого можна зв'язатися для отримання додаткової інформації або запису на тренування. Взагалі, цей екран дозволяє клієнтам отримати повний обсяг інформації про тренерів та зручно вибрати відповідного для себе інструктора. При кліку на номер, відбувається перенаправлення для зручного дзвінку.

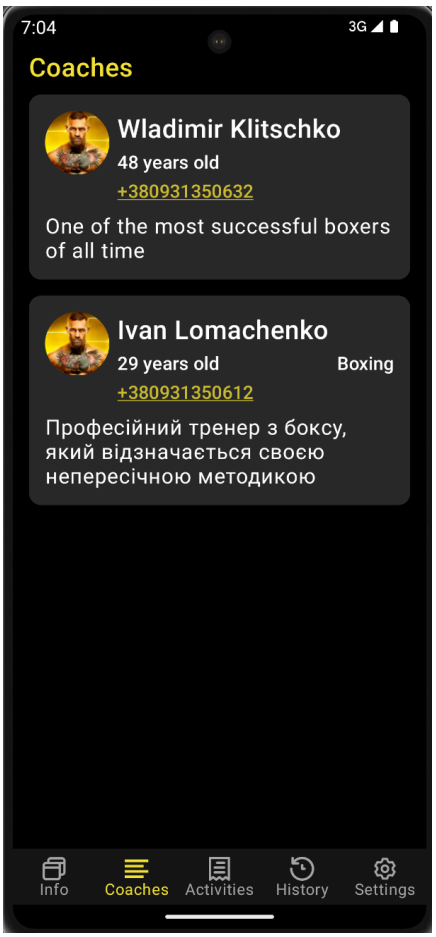


Рис. 5.4 Екран тренерів

5.4 Екран активностей

Екран активностей, який дозволяє користувачеві переглянути заплановані візити, відображається у застосунку FightAcademy. На цьому екрані користувач може переглянути список всіх запланованих тренувань та відвідувань у фітнес-центрі.

При першому відвідуванні цього екрану, коли ще немає запланованих візитів, користувач побачить повідомлення, що нічого не заплановано. Проте, зазначеною функцією в даному випадку є можливість додавання нових тренувань або запису на них. При натисканні на кнопку "плюс" користувачеві надається можливість перейти до екрану планування тренувань, де він може обрати дату, час та тип тренування для запису.

Цей екран активностей дозволяє користувачеві зручно та швидко переглядати свої заплановані візити та здійснювати нові записи за необхідності. Він спрощує процес організації часу та планування тренувань у фітнес-центрі FightAcademy.

Рис. 5.5 Пустий екран активностей

Екран бронювання тренування у застосунку FightAcademy призначений для зручного і швидкого планування тренувань користувачами. При переході на цей екран, спочатку доступна лише кнопка для обрання типу тренування. Інші кнопки, такі як обрання тренера та вибір дати з часом, залишаються недоступними, доки не буде обрано попереднє значення.

Рис. 5.5 Початковий стан екрану бронювання

Кнопка для обрання типу тренування дозволяє користувачеві вибрати необхідний тип тренування, такий як аеробіка, силові тренування, йога тощо. Після обрання типу тренування, інші кнопки стають активними, що дозволяє користувачеві продовжити процес планування.

Кнопка для обрання тренера дає користувачеві можливість вибрати конкретного тренера для тренування. Вибір тренера може залежати від його доступності, спеціалізації або інших параметрів, які визначаються в системі.

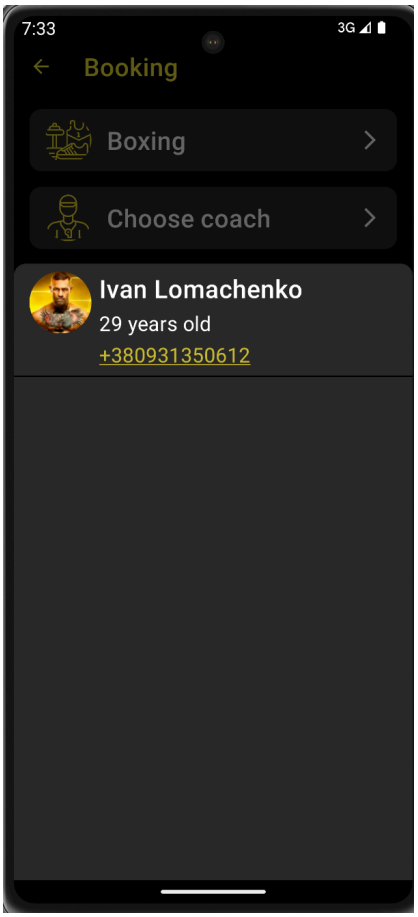


Рис. 5.6 Діалог вибору тренера

Кнопка для обрання дати з часом дозволяє користувачеві обрати певну дату та час для проведення тренування. Це дозволяє користувачеві забезпечити собі зручний час для відвідування фітнес-центру і узгодити його зі своїм розкладом.

Цей екран бронювання тренування має на меті спростити процес планування тренувань для користувачів, надаючи їм можливість вибирати тип, тренера та час, який найбільше відповідає їхнім потребам та розкладу.

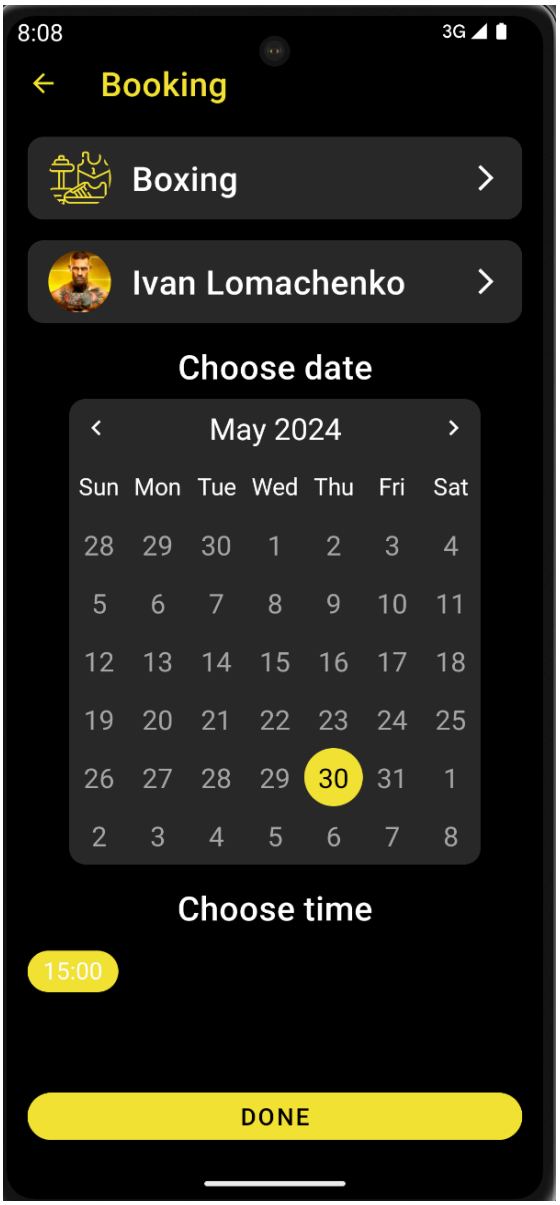


Рис. 5.7 Екран створення бронювання із заповниними полями

Після успішного бронювання тренування на екрані активностей з'являється новий запис, який містить інформацію про заплановане тренування рис. 5.8.

Цей новий запис на екрані активностей служить для зручного відстеження запланованих тренувань і надає користувачеві усю необхідну інформацію щодо дати, часу, ціни та тренера. Він дозволяє користувачам легко керувати своїм розкладом і вчасно прибувати на тренування.

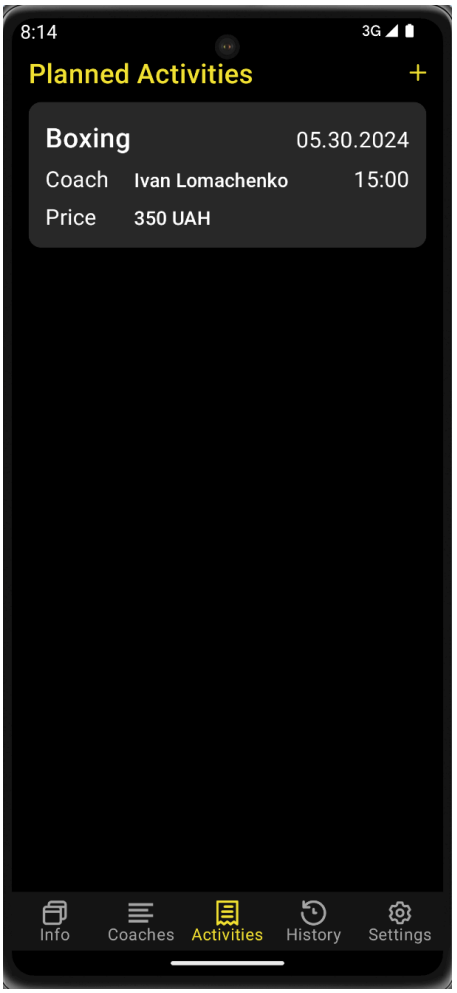


Рис. 5.7 Екран бронювань із новим записом

5.5 Екран налаштувань

На екрані налаштувань користувач має можливість здійснити кілька дій.

1. Вихід з акаунту: Кнопка, яка дозволяє користувачеві вийти зі свого облікового запису. Після натискання на цю кнопку користувач буде повернутий на екран входу або на екран привітання, залежно від конкретної реалізації застосунку.
2. Сповіщення про тренування: Перемикач або кнопка, яка дозволяє користувачеві увімкнути або вимкнути сповіщення про майбутні тренування. Якщо сповіщення включені, користувач отримуватиме повідомлення за певний час до початку кожного запланованого

тренування, що дозволяє йому не забувати про них і прибувати на них вчасно.

Ці налаштування дозволяють користувачам персоналізувати свій досвід використання застосунку FightAcademy, забезпечуючи зручність і контроль над їхнім обліковим записом та сповіщеннями про тренування.

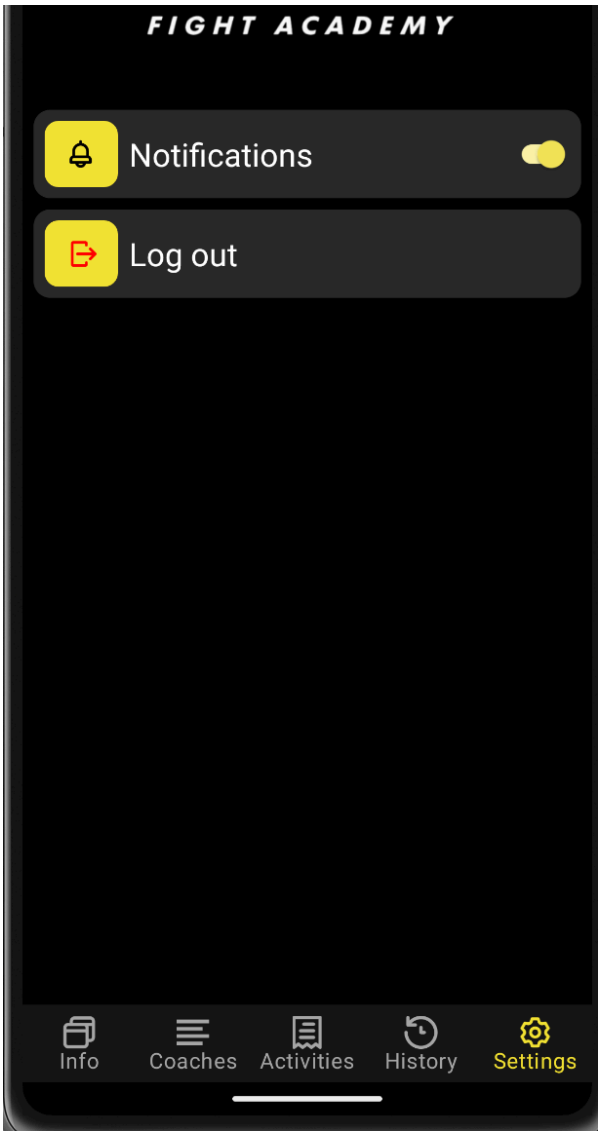


Рис. 5.9 Екран налаштувань

6 ТЕСТУВАННЯ ЗАСТОСУНКУ

6.1 Опис підходу до тестування застосунку

Тестування застосунку є ключовим етапом у розробці, оскільки воно дозволяє виявити помилки та недоліки, які можуть негативно позначитися на враженні користувачів. Система повинна працювати стабільно та без перебоїв. Щоб цього досягти, необхідно провести тестування програмного продукту. Для цієї мети складаються тест-кейси, які визначають послідовність дій для перевірки функціоналу.

6.2 Тестування програми

Таблиця 6.1

Тест кейс позитивної автентифікації

Тест кейс 01. Позитивна автентифікація користувача			
Передумови: відкритий застосунок на екрані логіну			
Кроки	Дані тесту	Очікуваний результат	ОК/ NOK
Введення номера телефону в поле	Валідний номер телефону	Не з’являється помилка формату номера телефону	ОК
Введення паролю в поле паролю	Валідний пароль від акаунту корст.	Не з’являється помилка паролю	ОК
Клік на кнопку логіну	Коректні попередні кроки	Користувача направляє на екран іформації і завантажуються його дані	ОК

Таблиця 6.2

Тест кейс екрану "Тренери"

Тест кейс 02. Позитивний перегляд списку тренерів			
Передумови: відкрити застосунок FightAcademy та перейти на екран "Тренери"			
Кроки	Дані тесту	Очікуваний результат	ОК/ NOK
Перейти на екран "Тренери"	Користувач увійшов у свій акаунт	Відображення списку доступних тренерів	ОК
Прокрутити список тренерів вниз	Відображено багато тренерів	Плавна прокрутка списку без затримок або проблем з відображенням	ОК
Натиснути на номер телефону тренера	Відображено номер телефону тренера	Користувача направляє перенаправляє на інший застосунок для здійснення дзвінка	ОК

Таблиця 6.3

Тест кейс екрану "Активності"

Тест кейс 03. Позитивний перегляд активних записів			
Передумови: відкрити застосунок FightAcademy та перейти на екран "Активні записи"			
Кроки	Дані тесту	Очікуваний результат	ОК/ NOK
Перейти на екран "Активні записи"	Користувач увійшов у свій акаунт	Відображення списку активних записів на тренування	ОК

Таблиця 6.4

Тест кейс екрану "Історія"

Тест кейс 04. Позитивний перегляд активних записів			
Передумови: Відкрити застосунок FightAcademy та перейти на екран "Історія"			
Кроки	Дані тесту	Очікуваний результат	ОК/ NOK
Перейти на екран "Історія"	Користувач увійшов у свій акаунт та відвідав тренування	Відображення списку історії візитів з попередніми тренуваннями	ОК
Прокрутити список візитів вниз	Користувач увійшов у свій акаунт та відвідав багато тренувань тренування	Плавна прокрутка списку без затримок	ОК

Таблиця 6.5

Тест кейс екрану "Історія"

Тест кейс 05. Позитивний перегляд активних записів			
Передумови: Відкрити застосунок FightAcademy та перейти на екран налаштувань			
Кроки	Дані тесту	Очікуваний результат	ОК/ NOK
Перейти на екран "Налаштування"	Користувач увійшов у свій акаунт	Відкриття екрану з налаштуваннями профілю користувача	ОК
Відключити сповіщення про тренування		Успішне вимкнення сповіщень та підтвердження збереження змін	ОК

Таблиця 6.6

Тест кейс екрану "Booking"

Тест кейс 06. Позитивний перегляд активних записів			
Передумови: Відкрити застосунок FightAcademy та перейти на екран активностей			
Кроки	Дані тесту	Очікуваний результат	ОК/ NOK
Перейти на екран "Booking"	Користувач увійшов у свій акаунт та є доступні опції для запису	Відображення списку доступних тренувань та опцій запису на них	ОК
Вибрати конкретне тренування для запису		Стан кнопки для обирання тренеру змінився на доступну	ОК
Вибрати конретну дату та час		З’вляється кнопка “Готово”	ОК
Натистинути на кнопку “Готово”		Користувач направлений на екран “Активності”, на якому в список добавлений новий запис	ОК

Тестування зручності використання визначає, наскільки інтерфейс додатку відповідає потребам та очікуванням користувачів. Цей тип тестування оцінює зручність взаємодії з програмним продуктом, включаючи навігацію, зрозумілість тексту, розташування елементів інтерфейсу, а також відповідність кольоровій палітрі та дизайну. Тест-кейси для оцінки зручності використання представлені у таблиці 6.7. Під час тестування застосунку FightAcademy всі перелічені тести успішно пройшли з результатом "passed".

Таблиця 6.7

Юзабіліті-тестування

Тест кейс 07. Юзабіліті-тестування		
Дії	Очікуваний результат	ОК/ NOK
Перевірити текстові елементи інтерфейсу, такі як заголовки, кнопки та посилання, на зрозумілість та читабельність.	Текст повинен легко читатися, відповідати дизайну та діям, за якими закріплений	ОК
Оцінити розташування елементів, таких як кнопки, поля для введення, списки та інші.	Елементи повинні бути розміщені зручно та логічнодля користувача, без перенасичення або зайвих відступів	ОК
Спробувати перейти на різні екрани застосунку за допомогою навігаційних елементів	Навігація повинна бути логічною та інтуїтивно зрозумілою, користувач повинен легко знайти потрібний функціонал	ОК

ВИСНОВКИ

Розроблений Android застосунок для онлайн запису на спортивні заняття засвідчує важливість сучасних технологій у галузі фітнесу. Запит на зручні та ефективні інструменти для бронювання занять стає дедалі більш актуальним у сучасному світі. Реалізація цього застосунку відповідає сучасним вимогам і допомагає спростити процес запису на тренування. Такі інструменти стають невід'ємною частиною роботи фітнес-залів, сприяючи їхньому успішному функціонуванню та покращенню взаємодії з клієнтами.

В ході виконання дипломної роботи було виконано наступні завдання:

1. Проведено аналіз обраної предметної області в сфері онлайн бронювань спортивних тренувань. Досліджено існуючі застосунки та сервіси з аналогічних фітнес центрів. Визначено ключові функції сервісів, їх переваги та недоліки, які враховано при розробці застосунку FightAcademy.
2. Розроблено основні вимоги до функціональних та нефункціональних вимог до застосунку та серверної системи. Проведено моделювання застосунку для взаємодії клієнтів фітнес-центру з використанням діаграми прецедентів.
3. Проведено аналіз засобів розробки програмного забезпечення. Обгрунтовано вибір мови програмування C# та фреймворку .net для розробки застосунку.
4. Розроблено застосунок “FightAcademy” з використанням Kotlin, C# та фреймворку .net.
5. Проведено комплексне тестування застосунку для виконання всіх функціональних та нефункціональних вимог.
6. Проект пройшов апробацію на Науково-технічній конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті» Київ, ДУІКТ, 15 травня 2024 року.

ПЕРЕЛІК ПОСИЛАНЬ

1. ЯКІ МОБІЛЬНІ ДОДАТКИ Є НАЙБІЛЬШ ПОПУЛЯРНИМИ. Київський міжнародний інститут соціології. 2021. URL: https://www.kiis.com.ua/?lang=ukr&cat=reports&id=1072&page=1.

2. Google Play. Sport Life Fitness. URL: https://play.google.com/store/apps/details?id=ua.sportlife.customer.

3. Google Play. APOLLO NEXT. URL: https://play.google.com/store/apps/details?id=com.perfectgym.perfectgymgo2.apollo.

4. Google Play. ClassPass. URL: https://play.google.com/store/apps/details?id=com.classpass.classpass&hl=uk&gl=US

5. Overview of ASP.NET Core MVC. Microsoft. 2023. URL: https://learn.microsoft.com/enus/aspnet/core/mvc/overview?view=aspnetcore-7.0.

6. Overview of Entity Framework Core - EF Core. Microsoft. 2023. URL: https://learn.microsoft.com/enus/ef/core/.

7. Introduction to Material Design in Android/ GeeksforGeeks. URL: https://www.geeksforgeeks.org/introduction-to-material-design-in-android/.

8. Metanit. URL: https://metanit.com/.

9. Android Developers: Android Mobile App Developer Tools. URL: https://developer.android.com/.

10. M. Porter, "Competitive Strategy: Techniques for Analyzing Industries and Competitors," Free Press, 2008.

11. E. Brynjolfsson and A. McAfee, "The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies," W. W. Norton & Company, 2016.

12. J. Durkee, "Mobile Application Development for Android Platform: A Literature Review," International Journal of Advanced Computer Science and Applications, vol. 10, no. 1, pp. 355-363, 2019.

13. M. Asif and A. Hussain, "Android Mobile App Development Framework: A Review Study," Journal of Computer and Communications, vol. 7, no. 10, pp. 41-48,

2019.
14. S. Arumugam and V. Arumugam, "Android Application Development: A Comprehensive Review," International Journal of Innovative Technology and Exploring Engineering, vol. 8, no. 7, pp. 2545-2552, 2019.
15. T. K. R. Mathaiya and A. A. M. J. Alzebaidi, "Review on Developing Android Mobile Applications Using Android Studio and Database Interface," International Journal of Engineering Research and Applications, vol. 9, no. 9, pp. 63-70, 2019.
16. Грамушняк Р.О., Негоденко О.В. Аналіз і порівняння бібліотек для реактивного програмування RxJava та Kotlin Flow для розробки Android застосунку FightAcademy. IV Всеукраїнська Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 206-207.
17. Грамушняк Р.О., Негоденко О.В. Оптимізація фітнес-центру: вплив та переваги використання Android застосунку. IV Всеукраїнська Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 172-173.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛ
(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка застосунку для фітнес центру з використанням мови C# та Kotlin

Виконав студент 4 курсу
групи ПД-41
Грамушняк Родіон Олександрович
Керівник роботи
К.т.н, доц, доцент кафедри ІПЗ Негоденко Олена Василівна

Київ – 2024

1

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

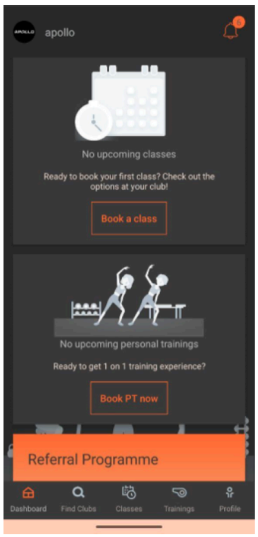
- **Мета роботи** - покращити взаємодію клієнтів до послуг фітнес центру за допомогою застосунку, створеного мовами C# та Kotlin.
- **Об’єкт дослідження** - процес взаємодії клієнтів до послуг фітнес центру.
- **Предмет дослідження** – застосунок для взаємодії клієнтів до послуг фітнес центру.

2

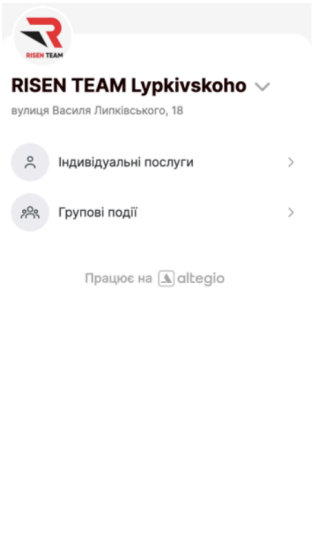
ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати ринок послуг фітнес центрів та встановити основні потреби користувачів.
2. Розробити вимоги до додатку, встановити функціональні та технічні вимог до застосунку та серверної системи з урахуванням потреб користувачів.
3. Проаналізувати та вибрати засоби та технології розробки.
4. Розробити модель архітектури та спроектувати застосунок за допомогою діаграми класів та прецедентів.
5. Розробити застосунок за допомогою вибраних інструменів та визначених вимог.
6. Провести тестування для перевірки працездатності та відповідності застосунку вимогам.

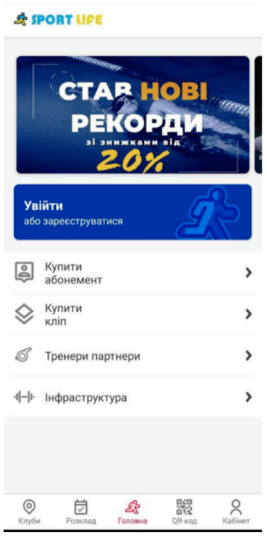
АНАЛІЗ ІНТЕРФЕЙСНИХ РІШЕНЬ АНАЛОГІВ



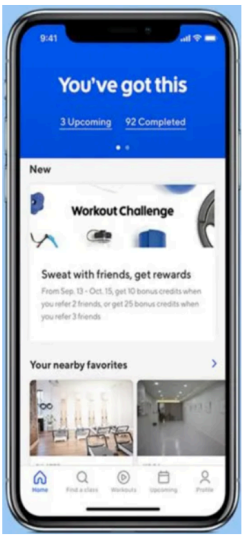
Apolo Next



Altegio



SportLife



ClassPass

АНАЛІЗ АНАЛОГІВ

Критерії оцінювання	Apolo Next	Altegio	SportLife	ClassPass	Fight Academy
Мови додатку	Англ.	Укр.	Англ.	Англ.	Англ.
Каталог занять	+	+	+	+	+
Фільтрація по типу заняття	-	+	-	+	+
Не потребує стабільного інтернет зв'язку	+	+	-	+	+
Історія відвідувань	+	+	+	-	+
Сповіщення для про існуючий запис	-	+	+	+	+
Дизайн та зручність користування	Застосунок має застарілі колірні схеми, відсутність анімації та елементів руху	Має мінімальний дизайн тільки з двома кнопками.	Сірий пустий простір створює відчуття порожнечі та незавершеності додатку.	Інформація чітка та лаконічна.	Зручна навігація додатку. Використано весь простір екрану. Заокруглені форми та плавність анімацій.

ВИМОГИ ДО ДОДАТКУ

Функціональні:

- Валідація введеного користувачем паролю.
- Запис на тренування.
- Перегляд історії записів.
- Відправлення нагадувань через сповіщення.

Нефункціональні:

- Надійність та стійкість.
- Зрозумілість.
- Масштабування.
- Дизайн має бути інтуїтивно зрозумілим та відповідати новим тенденціям.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Kotlin Coroutines

Android SDK

Microsoft .NET

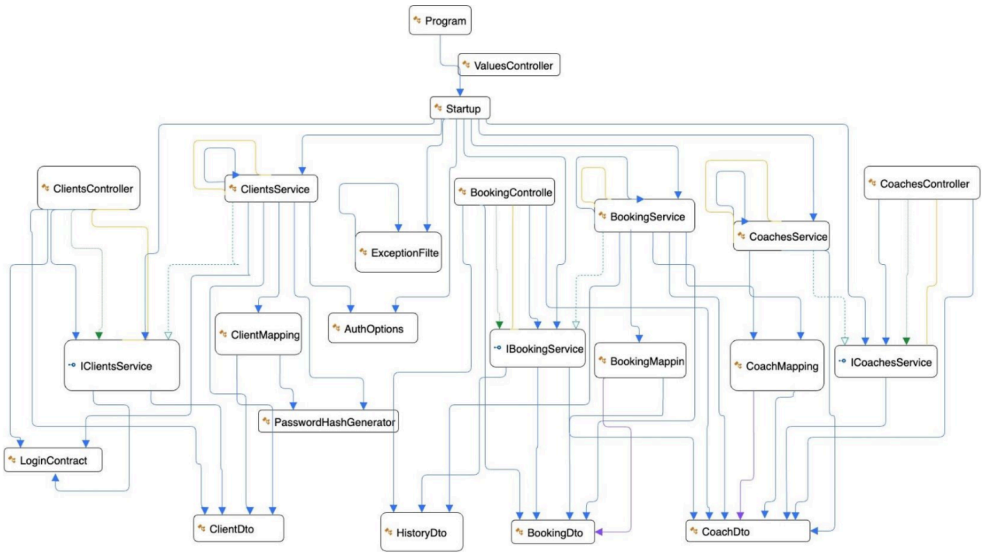
PostgreSQL

7

USE CASE ДІАГРАМА

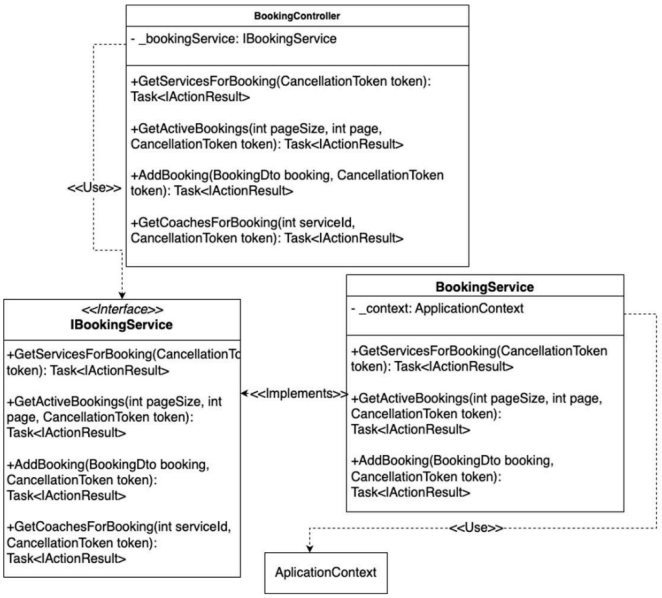
8

СХЕМА АРХІТЕКТУРИ ДОДАТКУ



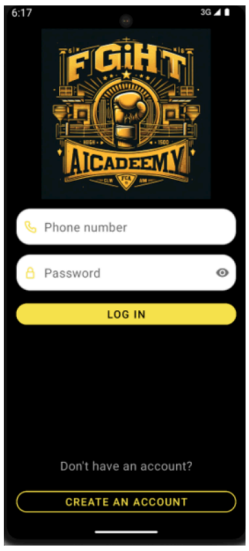
9

СХЕМА BOOKING CONTROLLER

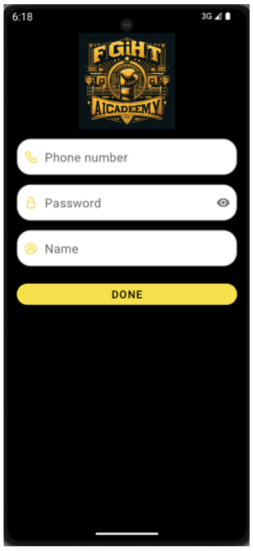


10

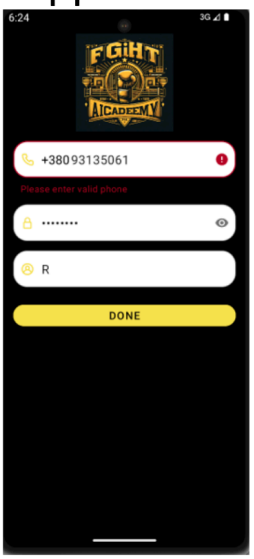
ЕКРАННІ ФОРМИ ВХОДУ В ПРОФІЛЬ



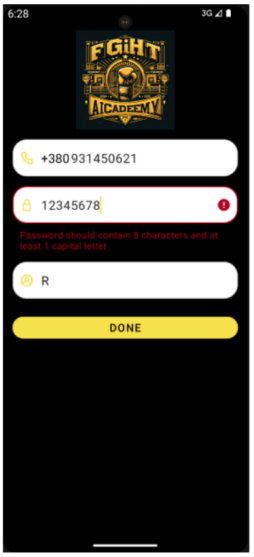
Логін



Реєстрація

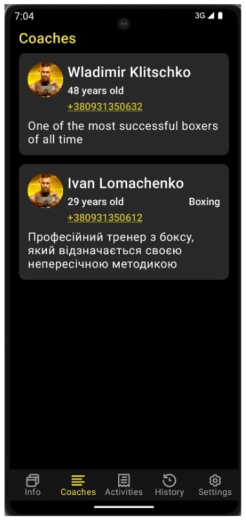


Валідація номеру телефона



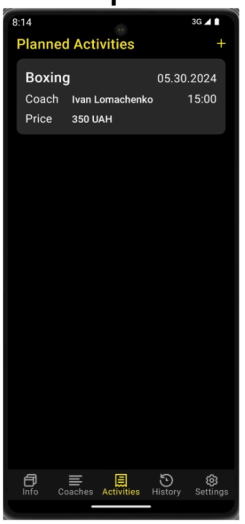
Валідація паролю

ЕКРАННІ ФОРМИ НАВІГАЦІЙНОЇ СИСТЕМИ

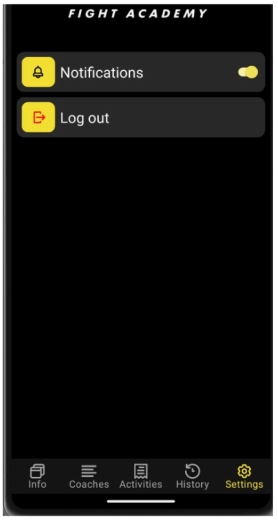


Список тренерів

Вибір бронювання

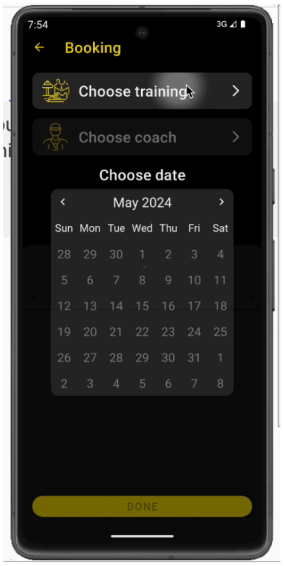


Забронювані активності



Налаштування

ДЕМОНСТРАЦІЯ РОБОТИ ЗАСТОСУНКУ



13

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Грамушняк Р.О., Негоденко О.В. Оптимізація фітнес-центру: вплив та переваги використання Android застосунку. IV Всеукраїнська Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 172-173.

2. Грамушняк Р.О., Негоденко О.В. Аналіз і порівняння бібліотек для реактивного програмування RxJava та Kotlin Flow для розробки Android застосунку FightAcademy. IV Всеукраїнська Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 206-207.

14

ВИСНОВКИ

1. Проведено аналіз обраної предметної області в сфері онлайн бронювань спортивних тренувань. Досліджено існуючі додатки та сервіси з аналогічних фітнес центрів. Визначено ключові функції сервісів, їх переваги та недоліки, які враховано при розробці застосунку FightAcademy.
2. Розроблено основні вимоги до функціональних та нефункціональних вимог до застосунку та серверної системи.
3. Проведено аналіз засобів розробки програмного забезпечення. Обґрунтовано вибір мови програмування C# та фреймворку .net для розробки застосунку.
4. Розроблено застосунок “FightAcademy” з використанням Kotlin, C# та фреймворку .net.
5. Проведено комплексне тестування застосунку для виконання всіх функціональних та нефункціональних вимог.