Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

# Кваліфікаційна робота бакалавра

на тему: Розробка мобільного фітнес додатку

Виконав студент групи ІПЗ20-1

Спеціальність 121 «Інженерія програмного забезпечення»

Корота Кирило Віталійович

Керівник к.т.н. доц. Мала Ю. А.

Рецензент _____

(місце роботи)

_____

(посада)

_____

(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

# АНОТАЦІЯ

*Корота К. В.* Розробка мобільного фітнес додатку.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2024.

Дипломна робота присвячена розробці мобільного фітнес додатку, спрямованого на покращення якості життя користувачів шляхом формування здорових звичок та підтримки активного способу життя.

У теоретичній частині роботи проаналізовано сучасні тенденції розвитку мобільних фітнес додатків, досліджено потреби та очікування користувачів, розглянуто особливості дизайну та функціоналу подібних програмних продуктів. Визначено основні вимоги до мобільного фітнес додатку, що забезпечують його ефективність та зручність використання.

Практична частина роботи включає розробку архітектури та дизайну мобільного фітнес додатку, створення користувацького інтерфейсу, реалізацію основних функцій, таких як відстеження фізичної активності, планування тренувань. Проведено тестування розробленого додатку з метою виявлення та усунення недоліків, оцінки його функціональності та зручності використання.

Результатом дипломної роботи є створення повнофункціонального мобільного фітнес додатку, який відповідає сучасним вимогам та потребам користувачів. Розроблений додаток має потенціал для подальшого розвитку та вдосконалення.

Ключові слова: мобільний додаток, фітнес, здоров'я, фізична активність, тренування, моніторинг.

# ABSTRACT

*Korota K. V.* Development of a mobile fitness application.

Qualification work for obtaining a bachelor's degree in specialty 121 "Software Engineering". – University of Customs and Finance, Dnipro, 2024.

The thesis is devoted to the development of a mobile fitness application aimed at improving the quality of users' lives by forming healthy habits and supporting an active lifestyle.

The theoretical part of the work analyzes current trends in the development of mobile fitness applications, examines the needs and expectations of users, and considers the features of the design and functionality of similar software products. The main requirements for a mobile fitness application that ensure its effectiveness and ease of use are determined.

The practical part of the work includes the development of the architecture and design of a mobile fitness application, the creation of a user interface, the implementation of basic functions such as tracking physical activity, and planning workouts. The developed application was tested to identify and eliminate shortcomings, evaluate its functionality and ease of use.

The result of the thesis is the creation of a fully functional mobile fitness application that meets modern requirements and user needs. The developed application has the potential for further development and improvement.

Keywords: mobile application, fitness, health, physical activity, training, monitoring.

# ЗМІСТ

# ВСТУП

Сучасний ритм життя вимагає від людини постійної уваги до власного здоров'я та фізичної форми. Зростаюча популярність здорового способу життя та фітнесу сприяє розвитку технологій, що допомагають людям досягати своїх цілей у цій сфері. Мобільні додатки стали невід'ємним інструментом для багатьох, хто прагне підтримувати активний спосіб життя, слідкувати за своїм харчуванням та тренуваннями. Серед них особливе місце займають фітнес-додатки, які надають можливість користувачам стежити за своїм здоров'ям, фізичною активністю та досягати поставлених цілей у сфері спорту та здорового способу життя. Актуальність розробки мобільних фітнес-додатків зумовлена зростаючим інтересом суспільства до здорового способу життя, підвищенням рівня обізнаності щодо важливості фізичної активності та прагненням людей до самовдосконалення.

Актуальність розробки мобільного фітнес-додатку зумовлена кількома факторами. Це зростаючий попит на персоналізовані та зручні інструменти для відстеження фізичної активності та прогресу у тренуваннях. Та мобільні додатки дозволяють забезпечити доступність фітнес-контенту та підтримку користувачів у будь-який час та в будь-якому місці.

Метою даної дипломної роботи є розробка мобільного фітнес додатку, який забезпечить користувачам комплексний підхід до тренувань та моніторингу їхнього фізичного стану.

Для досягнення мети необхідно вирішити такі задачі:

1. Дослідити прикладну тему та інструменти для створення додатку.
2. Визначити оптимальні технології та підходи до розробки.
3. Створити дизайн додатку за допомогою Figma.
4. Розробити архітектуру додатку та спроектувати базу даних.
5. Налаштувати інфраструктуру проекту.
6. Розробити мобільний додаток відповідно до визначених вимог.

Об'єктом дослідження є процес розробки мобільного фітнес-додатку.

Предметом дослідження є методи та інструменти розробки мобільного фітнес-додатку, дизайн, архітектура та функціональні можливості додатку.

Рівень розв'язання завдання розробки мобільних фітнес-додатків характеризується високою конкуренцією та різноманітністю існуючих рішень. На ринку представлено безліч додатків, які пропонують різні функціональні можливості, починаючи від відстеження фізичної активності та харчування, інтерактивними елементами гейміфікації. Проте, незважаючи на широкий вибір, існуючі додатки часто мають обмежену функціональність, складний інтерфейс користувача або недостатню персоналізацію, що ускладнює їх використання та знижує ефективність.

Для досягнення поставленої мети та створення конкурентоспроможного продукту, важливо враховувати сучасні тенденції у розробці мобільних додатків та використовувати передові технології. У рамках даної роботи планується застосувати сучасні підходи до проектування інтерфейсу користувача, забезпечити високий рівень персоналізації та адаптивності додатку, а також інтегрувати його з іншими сервісами та пристроями для розширення функціональних можливостей.

Окрему увагу буде приділено створенню інфраструктури проекту, що включатиме налаштування залежностей, написання власних утиліт для розробки, створення засобів для обробки бази даних, розробку дизайн системи та загальних UI компонентів. Важливим етапом буде розробка мобільного додатку відповідно до визначених вимог та поставлених задач. Завдяки комплексному підходу до розробки, додаток буде зручним, функціональним та ефективним інструментом для користувачів.

Сучасний рівень розв'язання завдання розробки мобільних фітнес додатків характеризується використанням передових технологій та інструментів. Для створення дизайну додатків широко застосовується Figma, яка дозволяє створювати інтерактивні прототипи та забезпечує ефективну співпрацю між дизайнерами та розробниками. Розробка архітектури додатків здійснюється з використанням сучасних підходів, таких як MVVM (Model-

View-Viewmodel), що забезпечує розділення відповідальності та полегшує тестування. Для проектування баз даних використовуються NoSQL бази даних, що дозволяє вибрати оптимальний варіант залежно від потреб проекту.

Створення інфраструктури проекту включає використання систем контролю версій, таких як Git, що забезпечує ефективне відстеження змін у коді. Для автоматизації збірки проекту використовуються інструменти, такі як Gradle або Fastlane, що прискорює процес розробки та забезпечує стабільність роботи додатку. Розробка мобільних додатків здійснюється з використанням крос платформеного фреймворка, React Native, що дозволяє створювати додатки для різних платформ з використанням єдиної кодової бази.

Очікується, що розроблений мобільний фітнес додаток буде мати такі основні технічні характеристики: підтримка різних платформ (Android та iOS), зручний та інтуїтивно зрозумілий інтерфейс, можливість відстеження фізичної активності, персоналізація тренувань. Реалізація даного проекту матиме значний технічно-економічний ефект, оскільки додаток може залучити широку аудиторію користувачів, що сприятиме підвищенню рівня фізичної активності населення та покращенню якості життя. Крім того, додаток може стати джерелом додаткового доходу.

# РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Дослідження предметної області та визначення ключових понять розробки мобільних додатків

Мобільні додатки стали невід'ємною частиною нашого повсякденного життя, змінюючи спосіб, яким ми взаємодіємо з технологіями, інформацією та один з одним. Від соціальних мереж і розваг до фінансів, освіти та охорони здоров'я, мобільні додатки проникли практично в усі сфери людської діяльності. Зростання популярності мобільних пристроїв, таких як смартфони та планшети, призвело до вибухового розвитку ринку мобільних додатків, створюючи величезні можливості для бізнесу, розробників та користувачів.

Розробка мобільних додатків є складним і багатомірним процесом, що вимагає глибокого розуміння предметної області, технологічних інструментів та потреб користувачів. Вона охоплює широкий спектр дисциплін, включаючи програмування, дизайн інтерфейсу користувача, тестування, маркетинг та розповсюдження. Успіх мобільного додатку залежить від багатьох факторів, таких як його функціональність, зручність використання, продуктивність, безпека та відповідність очікуванням цільової аудиторії.

Метою даного дослідження є всебічне вивчення предметної області розробки мобільних додатків та визначення ключових понять, що лежать в її основі. Ми розглянемо основні етапи життєвого циклу розробки мобільного додатку, починаючи від формулювання ідеї та аналізу ринку до проектування, розробки, тестування, запуску та подальшої підтримки. Особливу увагу буде приділено сучасним технологіям та інструментам, що використовуються в розробці мобільних додатків, таким як мови програмування, фреймворки, бібліотеки та платформи.

У ході дослідження ми також проаналізуємо основні тенденції та виклики, що стоять перед розробниками мобільних додатків у сучасному світі. Ми розглянемо такі питання, як фрагментація платформ, забезпечення безпеки

даних, монетизація додатків, оптимізація продуктивності та адаптація до різних розмірів екранів та роздільних здатностей.

Життєвий цикл розробки мобільного додатку – це комплексний процес, що складається з кількох взаємопов'язаних етапів, кожен з яких відіграє важливу роль у створенні успішного продукту. Розглянемо основні етапи цього процесу:

1. Ідея та концепція: Все починається з ідеї – бачення того, яким має бути додаток, яку проблему він вирішуватиме та яку цінність надаватиме користувачам. На цьому етапі важливо провести дослідження ринку, вивчити конкурентів та визначити цільову аудиторію.

2. Планування та аналіз: На цьому етапі формується детальний план розробки, включаючи визначення функціональних вимог, технічних специфікацій, термінів та бюджету. Проводиться аналіз ризиків та можливостей, визначаються ключові показники ефективності (KPI) для оцінки успішності проекту.

3. Проектування: Цей етап включає розробку архітектури додатку, дизайн інтерфейсу користувача (UI) та досвіду користувача (UX). Дизайн має бути інтуїтивно зрозумілим, привабливим та відповідати потребам цільової аудиторії. Також на цьому етапі створюються прототипи та макети для візуалізації майбутнього продукту.

4. Розробка: На цьому етапі відбувається безпосереднє програмування додатку, створення серверної частини (якщо потрібно), інтеграція з API та базами даних. Розробники використовують обрані мови програмування, фреймворки та інструменти для реалізації функціональності додатку [1].

5. Тестування: Тестування – критично важливий етап, на якому виявляються та виправляються помилки, перевіряється продуктивність, безпека та сумісність додатку з різними пристроями та операційними системами. Тестування може бути ручним або автоматизованим, включати альфа- та бета-тестування із залученням реальних користувачів.

6. Запуск та розповсюдження: Після успішного тестування додаток готовий до запуску. Його публікують у відповідних магазинах додатків (App Store, Google Play), проводять маркетингові кампанії для просування та залучення користувачів.

7. Підтримка та оновлення: Після запуску додаток потребує постійної підтримки, виправлення помилок, оновлення функціональності та адаптації до нових версій операційних систем. Важливо збирати відгуки користувачів та аналізувати дані використання для покращення продукту.

8. Монетизація (опціонально): Якщо метою розробки є отримання прибутку, на цьому етапі визначається модель монетизації (реклама, платні підписки, покупки всередині додатку тощо) та впроваджуються відповідні механізми.

Життєвий цикл розробки мобільного додатку не є лінійним, а скоріше ітеративним процесом, де деякі етапи можуть повторюватися або відбуватися паралельно. Успіх проекту залежить від ефективної комунікації між усіма учасниками, чіткого планування, якісного виконання кожного етапу та постійного вдосконалення продукту на основі зворотного зв'язку від користувачів.

Сучасна розробка мобільних додатків спирається на широкий спектр технологій та інструментів, що постійно розвиваються. Розглянемо ключові з них:

Мови програмування:

- Swift. Мова програмування від Apple, спеціально розроблена для створення додатків під iOS, macOS, watchOS та tvOS. Відрізняється високою продуктивністю, безпекою та сучасним синтаксисом [2].
- Kotlin. Офіційна мова програмування для Android, що пропонує лаконічний синтаксис, безпеку типів та повну сумісність з Java [3].
- JavaScript. Універсальна мова, що використовується в багатьох кросплатформенних фреймворках (React Native, Ionic) для розробки додатків під iOS та Android одночасно [4].

- Dart. Мова програмування, що використовується у фреймворку Flutter для створення високопродуктивних та візуально привабливих кросплатформенних додатків [5].

Фреймворки:

- React Native. Популярний фреймворк від Facebook, що дозволяє створювати нативні додатки під iOS та Android, використовуючи JavaScript та React [6].
- Flutter. Фреймворк від Google, що використовує Dart для створення швидких та візуально привабливих кросплатформенних додатків з власним рендерингом інтерфейсу [7].
- Xamarin. Фреймворк від Microsoft, що дозволяє створювати нативні додатки під iOS, Android та Windows за допомогою C# [8].
- Ionic. Фреймворк на основі Angular та веб-технологій (HTML, CSS, JavaScript), що дозволяє створювати кросплатформенні додатки з акцентом на веб-компоненти [9].

Бібліотеки:

- Retrofit. Популярна бібліотека для Android, що спрощує роботу з мережевими запитами та API [10].
- Alamofire. Аналог Retrofit для iOS, спрощує роботу з мережевими запитами [11].
- Glide. Бібліотека для Android, що ефективно завантажує та кешує зображення [12].
- SDWebImage. Аналог Glide для iOS [13].
- Firebase. Платформа від Google, що надає широкий спектр сервісів для розробки мобільних додатків, включаючи базу даних реального часу, аутентифікацію, зберігання файлів, хмарні функції, аналітику та багато іншого [14].

Платформи:

- Android Studio. Офіційне інтегроване середовище розробки (IDE) для Android, що надає всі необхідні інструменти для створення, тестування та налагодження додатків [15].
- Xcode. IDE від Apple для розробки під iOS, macOS, watchOS та tvOS [16].

Інші інструменти:

- Git. Система контролю версій, що дозволяє відстежувати зміни в коді та співпрацювати з іншими розробниками [17].
- Figma. Інструменти для дизайну інтерфейсів користувача [18].
- Postman. Інструмент для тестування API [19].
- Jenkins. Інструменти для автоматизації збірки та розгортання додатків [20].

Вибір конкретних технологій та інструментів залежить від багатьох факторів, таких як платформи, під які розробляється додаток, вимоги до продуктивності та функціональності, досвід та вподобання команди розробників.

## 1.2 Обґрунтування актуальності розробки мобільного фітнес додатку

У сучасному світі, де здоров'я та фізична активність набувають все більшого значення, мобільні фітнес-додатки стають незамінним інструментом для досягнення та підтримки здорового способу життя. Вони надають користувачам широкий спектр можливостей – від відстеження фізичної активності та харчування до персоналізованих тренувальних програм та мотиваційної підтримки. Зростаюча популярність мобільних пристроїв та підвищення обізнаності людей щодо важливості фізичного здоров'я роблять розробку мобільних фітнес-додатків надзвичайно актуальною і перспективною.

Актуальність розробки фітнес-додатку зумовлена низкою факторів:

- Зростаюча зацікавленість у здоровому способі життя. Сучасні люди все більше усвідомлюють важливість фізичної активності та правильного харчування для підтримки здоров'я та профілактики захворювань. Мобільні фітнес-додатки допомагають їм відстежувати свої досягнення, ставити цілі та підтримувати мотивацію. Сучасний спосіб життя, що характеризується сидячою роботою, стресом та неправильним харчуванням, призводить до збільшення ризику розвитку хронічних захворювань, таких як ожиріння, діабет, серцево-судинні захворювання та інші. У зв'язку з цим, все більше людей усвідомлюють важливість фізичної активності та здорового харчування для підтримки свого здоров'я та профілактики захворювань. Мобільні фітнес-додатки допомагають користувачам зробити перші кроки до здорового способу життя, надаючи їм необхідну інформацію, інструменти та мотивацію.
- Зручність та доступність. Мобільні додатки забезпечують зручний доступ до фітнес-інформації та інструментів у будь-який час та в будь-якому місці. Користувачі можуть відстежувати свої тренування, харчування та прогрес, не відвідуючи спортзал чи консультуючись з тренером.
- Персоналізація. Сучасні фітнес-додатки пропонують персоналізовані тренувальні програми, плани харчування та рекомендації, враховуючи індивідуальні особливості та потреби користувачів. Це дозволяє досягати кращих результатів та підвищує задоволеність від використання додатку.
- Мотивація та соціальна підтримка. Багато фітнес-додатків включають елементи гейміфікації, соціальні функції та спільноти, що допомагають користувачам підтримувати мотивацію, змагатися з друзями та отримувати підтримку від однодумців.
- Технологічний прогрес. Розвиток технологій, таких як штучний інтелект, машинне навчання та носимі пристрої, відкриває нові можливості для створення більш інтелектуальних та ефективних фітнес-додатків. Вони

можуть аналізувати дані користувачів, надавати персоналізовані рекомендації та прогнозувати результати.

Розвиток мобільних технологій та зростання доступності смартфонів та планшетів сприяють популяризації мобільних фітнес-додатків. Сучасні мобільні пристрої оснащені різноманітними сенсорами та датчиками, що дозволяють відстежувати фізичну активність, пульс, сон та інші параметри здоров'я. Це відкриває нові можливості для розробки інноваційних та персоналізованих фітнес-додатків, які можуть адаптуватися до індивідуальних потреб та цілей користувачів.

За даними досліджень, до 2026 року його обсяг може досягти 14,7 мільярда доларів США. Кількість завантажень фітнес-додатків зростає з кожним роком, а кількість активних користувачів вже перевищує сотні мільйонів. Ці цифри свідчать про величезний попит на такі додатки та підтверджують їхню важливість у сучасному суспільстві.

Однією з ключових переваг мобільних фітнес-додатків є можливість персоналізації тренувальних програм та рекомендацій щодо харчування. Користувачі можуть вибирати тренування відповідно до своїх уподобань, рівня фізичної підготовки та цілей. Додатки можуть враховувати індивідуальні особливості користувачів, такі як вік, стать, вага, рівень активності та хронічні захворювання, що дозволяє створювати максимально ефективні та безпечні програми тренувань.

Мобільні фітнес рішення не лише надають користувачам інструменти для відстеження прогресу, але й мотивують їх досягати своїх цілей. Вони можуть використовувати різноманітні методи мотивації, такі як нагороди, бейджі, рейтинги, соціальні виклики та персоналізовані повідомлення. Крім того, багато додатків мають функції соціальної взаємодії, що дозволяють користувачам ділитися своїми досягненнями, підтримувати один одного та брати участь у спільних викликах. Це створює відчуття спільноти та допомагає користувачам залишатися мотивованими та досягати кращих результатів.

До мобільного додатку дуже легко отримати доступ будь-де та будь-коли, що робить їх зручним інструментом для людей з різним способом життя та графіком роботи. Користувачі можуть тренуватися вдома, у спортзалі або на свіжому повітрі, використовуючи свій смартфон або планшет. Додатки також можуть синхронізуватися з іншими пристроями, такими як фітнес-трекери та розумні годинники, що дозволяє користувачам отримувати більш повну картину своєї фізичної активності та здоров'я.

Розробка мобільного додатку є актуальною і перспективною ідеєю, зумовленою зростаючим інтересом до здорового способу життя, зручністю та доступністю мобільних технологій, а також розвитком інноваційних технологій. Мобільні фітнес-додатки мають значний потенціал для задоволення потреб користувачів, створення бізнес-можливостей та позитивного впливу на суспільство.

Однак, розробка успішного додатку вимагає ретельного планування, аналізу ринку, вивчення потреб цільової аудиторії, використання сучасних технологій та інструментів, а також постійного вдосконалення продукту на основі зворотного зв'язку від користувачів. Тільки так можна створити додаток, який буде корисним, зручним та ефективним для користувачів, а також прибутковим для розробників.

## 1.3. Постановка задачі та визначення основних цілей дипломної роботи

Завданням дипломної роботи є розробка мобільного фітнес додатку, який допоможе користувачам створювати та дотримуватися персоналізованих програм тренувань, відстежувати свій прогрес.

Основні задачі дипломної роботи:

1. Аналіз ринку та користувацьких потреб:

- Провести дослідження існуючих фітнес додатків на ринку, визначити їх сильні та слабкі сторони.

- Вивчити потреби та очікування цільової аудиторії щодо функціональності та дизайну фітнес додатку.

2. Розробка концепції додатку:

- Визначити основні функції та можливості додатку, які забезпечуватимуть його конкурентоспроможність.
- Розробити архітектуру додатку, включаючи базу даних для зберігання даних користувачів та інформації про тренування.

3. Дизайн та інтерфейс користувача:

- Створити інтуїтивно зрозумілий та привабливий дизайн додатку, який буде забезпечувати легкість використання та позитивний користувацький досвід.
- Розробити макети основних екранів додатку та визначити логіку навігації між ними.

4. Реалізація додатку:

- Налаштування інфраструктури проекта та робота із залежностями
- Написати код для реалізації основних функцій додатку.

5. Тестування та оптимізація:

- Розробити тестові кейси для додатку
- Провести тестування додатку на різних пристроях та операційних системах для забезпечення його стабільної роботи.

У першому розділі було проведено детальний аналіз предметної області розробки мобільних додатків, включаючи ключові етапи життєвого циклу, від формулювання ідеї до підтримки та оновлення, а також основні технології та інструменти, такі як мови програмування, бібліотеки та платформи. Було досліджено актуальність розробки мобільних фітнес-додатків, враховуючи зростаючий інтерес до здорового способу життя, зручність та доступність мобільних технологій, а також розвиток інноваційних технологій у сфері фітнесу та здоров'я, таких як штучний інтелект, машинне навчання та носимі пристрої.

З'ясовано, що мобільні фітнес-додатки мають значний потенціал для задоволення потреб користувачів, пропонуючи персоналізовані тренувальні програми, плани харчування, відстеження прогресу, мотиваційні механізми та зручний доступ до інформації про здоров'я та фізичну активність. Вони також можуть враховувати індивідуальні особливості користувачів, такі як вік, стать, вага, рівень активності та хронічні захворювання, що дозволяє створювати максимально ефективні та безпечні програми тренувань.

Розуміння основних аспектів розробки мобільних додатків, включаючи технологічний стек, життєвий цикл та обґрунтування актуальності створення фітнес-додатку, є важливим підґрунтям для подальшого визначення вимог, проектування та розробки конкретного продукту, що задовольнить потреби цільової аудиторії та сприятиме покращенню їхнього здоров'я та якості життя. Крім того, врахування сучасних тенденцій та викликів у розробці мобільних додатків, таких як фрагментація платформ, забезпечення безпеки даних, монетизація додатків, оптимізація продуктивності та адаптація до різних розмірів екранів та роздільних здатностей, є критичним для успіху проекту.

# РОЗДІЛ 2 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ВИБІР МЕТОДІВ РІШЕННЯ

## 2.1 Аналіз альтернативних рішень поставленої задачі

Розробка ефективного та зручного мобільного фітнес-додатку є складним та багатогранним завданням, що вимагає глибокого розуміння потреб користувачів, сучасних технологій та тенденцій у сфері фітнесу. Успіх такого додатку залежить від безлічі факторів, включаючи зручний інтерфейс, персоналізований підхід, мотиваційні елементи та інтеграцію з іншими сервісами та пристроями.

Буде проведено комплексний аналіз існуючих мобільних фітнес-додатків, які вже зарекомендували себе на ринку. Аналіз охопить широкий спектр аспектів, включаючи:

- Функціональність. Детальний розгляд основних та додаткових функцій додатків, таких як відстеження активності, планування тренувань, моніторинг харчування, аналіз прогресу, соціальні функції та ін.
- Інтерфейс та досвід користувача. оцінка зручності, інтуїтивності та естетичної привабливості інтерфейсу, а також загального враження від використання додатку.
- Технології. Дослідження технологічного стеку, використаного у розробці додатків, включаючи мови програмування, фреймворки, бази даних, хмарні сервіси та ін.
- Бізнес-моделі. Аналіз різних підходів до монетизації додатків, таких як платна підписка, реклама, покупки всередині додатку тощо.
- Цільова аудиторія. Визначення основних груп користувачів, на яких орієнтовані додатки, та їх потреб.
- Переваги та недоліки. Виявлення сильних та слабких сторін кожного додатку, а також можливостей для покращення.

Цей комплексний аналіз дозволяє виявити найкращі практики та підходи, які можуть бути використані при розробці власного мобільного

фітнес-додатку. Він допоможе уникнути типових помилок, оптимізувати функціональність та створити додаток, який буде відповідати потребам та очікуванням користувачів. Крім того, аналіз конкурентів дозволяє визначити потенційні ніші та можливості для інновацій, що допоможе створити унікальний та конкурентоспроможний продукт.

З урахуванням широкого спектру існуючих мобільних фітнес-додатків та різноманіття їх функціоналу, важливо ретельно розглянути основні варіанти реалізації власної ідеї.

Google Fit – це безкоштовна платформа для відстеження фізичної активності та здоров'я, розроблена Google (рис. 2.1). Вона доступна як мобільний додаток для Android та iOS, а також інтегрована в деякі смарт-годинники на Wear OS [21].

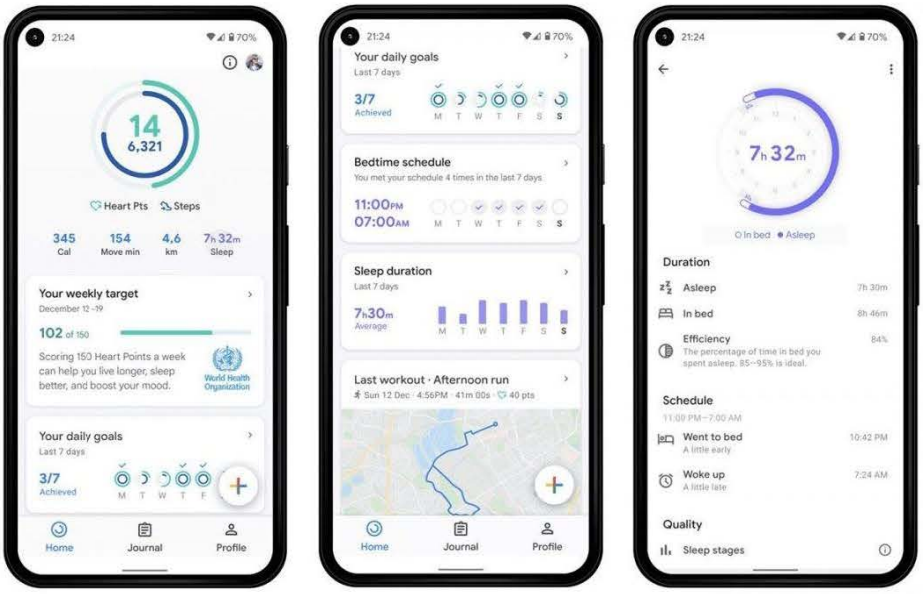


Рисунок 2.1 – Мобільний додаток Google Fit

Google Fit є безкоштовним додатком, який не має прямої монетизації у вигляді платної підписки або реклами. Однак, Google використовує його в рамках своєї загальної стратегії розвитку екосистеми та збору даних.

Цільова аудиторія Google Fit охоплює широкий спектр користувачів, які прагнуть вести активний та здоровий спосіб життя. Серед них можна виділити такі основні групи [22]:

1. Початківці: люди, які тільки починають свій шлях до здорового способу життя та фізичної активності. Google Fit приваблює їх своєю простотою, інтуїтивним інтерфейсом та базовими функціями для відстеження активності.

2. Користувачі Android та Wear OS: оскільки Google Fit тісно інтегрований з екосистемою Google, він особливо привабливий для користувачів Android-смартфонів та смарт-годинників на Wear OS. Ці користувачі цінують зручність синхронізації даних між пристроями та використання інших сервісів Google.

3. Люди, які шукають безкоштовний додаток: Google Fit є повністю безкоштовним, що робить його привабливим для тих, хто не хоче платити за підписку на фітнес-додатки.

4. Користувачі, яким потрібні базові функції: Google Fit пропонує основні функції для відстеження активності, такі як кроки, калорії та дистанція. Це підходить тим, хто не потребує розширених функцій, таких як детальний аналіз тренувань або персоналізовані плани.

5. Люди, які цінують простоту та зручність: Google Fit має простий та інтуїтивно зрозумілий інтерфейс, що робить його легким у використанні навіть для тих, хто не має досвіду використання фітнес-додатків.

Основні функції:

- Відстеження активності. Автоматично відстежує кроки, час активності, дистанцію та спалені калорії протягом дня. Також можна вручну додавати різні види активності, такі як біг, плавання, йога тощо.
- Цілі активності. Допомагає встановити та відстежувати щоденні цілі щодо кроків, часу активності та спалених калорій.
- Серцеві бали. Заохочує до більш інтенсивної активності, нараховуючи бали за кожну хвилину помірної або інтенсивної активності.

- Персоналізовані поради. Надає рекомендації щодо фізичної активності та здоров'я на основі ваших даних та цілей.

Переваги:

1. Простота та зручність. Має інтуїтивно зрозумілий інтерфейс та легко налаштовується.
2. Безкоштовність. Всі основні функції доступні безкоштовно.
3. Інтеграція з Google-сервісами. Легко синхронізується з іншими додатками Google, такими як Google Calendar та Google Assistant.
4. Підтримка Wear OS. Працює на смарт-годинниках з Wear OS, що дозволяє відстежувати активність без телефону.

Недоліки:

1. Обмежена функціональність. Немає деяких функцій, які є в інших фітнес-додатках, таких як відстеження сну, моніторинг харчування або детальний аналіз тренувань.
2. Менше персоналізації. Порівняно з іншими додатками, пропонує менше можливостей для персоналізації тренувань та цілей.
3. Залежність від телефону. Для повного функціоналу потрібен смартфон.

Google Fit є закритою платформою, тому Google не розголошує повний технологічний стек, який використовується для його розробки та підтримки [23]. Однак, на основі доступної інформації та аналізу, можна зробити деякі припущення щодо технологій, які можуть бути використані:

Мобільні додатки:

- Android. Java або Kotlin (основні мови програмування для Android).
- iOS. Swift або Objective-C (основні мови програмування для iOS).

Бекенд:

- Google Cloud Platform. Ймовірно, використовується для зберігання даних, обробки запитів та інших серверних операцій.

- Firebase. Може бути використаний для синхронізації даних між пристроями, повідомлень та інших функцій реального часу.
- Мови програмування. Можливо, використовуються такі мови, як Python, Go або Java для розробки серверної логіки.

Цей хороший вибір для тих, хто шукає простий та зручний додаток для відстеження базової активності та досягнення загальних цілей щодо фізичної активності. Він особливо підійде користувачам Android та Wear OS, які цінують інтеграцію з екосистемою Google. Однак, якщо вам потрібні більш розширені функції та персоналізація, варто розглянути інші варіанти.

Runkeeper (ASICS Runkeeper) – це мобільний додаток для відстеження та аналізу бігових тренувань, а також інших видів активності, таких як ходьба, їзда на велосипеді тощо (рис. 2.2)[24].


Рисунок 2.2 – Мобільний додаток Runkeeper

Runkeeper використовує модель Freemium, що означає, що базові функції додатку доступні безкоштовно, але деякі преміум-функції вимагають платної підписки [25].

Цільова аудиторія Runkeeper складається з активних людей, які займаються бігом, ходьбою, їздою на велосипеді або іншими видами спорту на відкритому повітрі [26]. Більш конкретно, можна виділити такі групи:

1. Бігуни-аматори. Це основна цільова аудиторія Runkeeper. Додаток пропонує їм зручний інструмент для відстеження своїх пробіжок, встановлення цілей, відстеження прогресу та мотивації.

2. Люди, які готуються до змагань. Runkeeper пропонує персоналізовані плани тренувань та аудіо підказки, що робить його корисним інструментом для тих, хто готується до марафонів, півмарафонів або інших змагань.

3. Любителі активного відпочинку. Runkeeper підходить не тільки для бігунів, але й для тих, хто займається ходьбою, їздою на велосипеді, лижами та іншими видами спорту на свіжому повітрі.

4. Користувачі, які шукають соціальної взаємодії. Runkeeper має активну спільноту користувачів, де можна ділитися своїми досягненнями, брати участь у викликах та отримувати підтримку від інших спортсменів.

5. Користувачі, які цінують простоту та зручність. Runkeeper має інтуїтивно зрозумілий інтерфейс та легко налаштовується, що робить його доступним для користувачів з різним рівнем технічної підготовки.

Основні функції:

- Відстеження активності. Використовує GPS для відстеження маршруту, дистанції, темпу, часу та спалених калорій під час тренувань.
- Аудіо підказки: надає голосові підказки про темп, дистанцію та інші параметри під час тренування.
- Персоналізовані плани тренувань. Пропонує адаптивні плани тренувань для досягнення різних цілей, таких як підготовка до забігу, покращення швидкості або просто підтримання форми.
- Змагання та виклики. Дозволяє брати участь у віртуальних змаганнях та викликах з іншими користувачами.

- Інтеграція з іншими додатками та пристроями. Сумісний з різними фітнес-трекерами та смарт-годинниками, а також інтегрується з іншими додатками, такими як MyFitnessPal та Spotify.

Переваги:

1. Простота використання. Має інтуїтивно зрозумілий інтерфейс та легко налаштовується.
2. Широкий функціонал. Пропонує безліч функцій для відстеження та аналізу тренувань.
3. Персоналізація. Дозволяє створювати власні маршрути та налаштовувати аудіо підказки.
4. Соціальні функції. Дає змогу ділитися своїми досягненнями з друзями та брати участь у змаганнях.

Недоліки:

1. Обмеження безкоштовної версії. Деякі функції, такі як детальний аналіз тренувань та персоналізовані плани, доступні лише у платній підписці Runkeeper Go.
2. Точність GPS. Іноді можуть виникати проблеми з точністю відстеження GPS, особливо у приміщеннях або при слабкому сигналі.
3. Реклама. Безкоштовна версія містить рекламу.

Мобільні додатки:

- Android. Java або Kotlin (основні мови програмування для Android).
- iOS. Swift або Objective-C (основні мови програмування для iOS).

Фреймворки: React Native або Flutter.

Карти та GPS: Google Maps API, Mapbox або Apple Maps (для відображення карт та відстеження GPS).

Аналітика: Firebase Analytics, Google Analytics або Flurry.

Бекенд:

- Хмарні платформи. Amazon Web Services (AWS), Google Cloud Platform (GCP) або Microsoft Azure (для хостингу та інфраструктури).

- Бази даних. PostgreSQL, MySQL або MongoDB (для зберігання даних користувачів та тренувань).
- Мови програмування. Node.js, Python, Ruby або Java (для розробки серверної логіки).
- Фреймворки: Express.js, Django, Ruby on Rails або Spring (для розробки веб-додатків та API).

Інші технології:

- Машинне навчання. Може використовуватися для аналізу даних тренувань та надання персоналізованих рекомендацій.
- Обробка природної мови (NLP). Може бути використана для розуміння голосових команд та надання аудіо підказок.

Runkeeper належить ASICS, тому вони можуть використовувати деякі власні технології та інструменти [27].

Runkeeper – це чудовий додаток для бігунів та інших спортсменів, які хочуть відстежувати свої тренування та прогрес. Він пропонує широкий спектр функцій, простий у використанні та має активну спільноту користувачів. Однак, деякі розширені функції доступні лише у платній версії, а точність GPS може бути не завжди ідеальною.

Centr – це мобільний додаток для фітнесу та здорового способу життя, розроблений командою знаменитого актора Кріса Хемсворта. Додаток пропонує комплексний підхід до здоров'я, поєднуючи різноманітні тренування, плани харчування та поради щодо ментального благополуччя. Centr створено для людей з будь-яким рівнем підготовки, і він включає в себе тренування з вагою, кардіо, йогу, пілатес, а також інші типи фізичних вправ. Плани харчування розроблені з урахуванням різних дієтичних потреб та вподобань, включаючи рецепти, які легко приготувати вдома. Крім того, додаток пропонує медитації, техніки релаксації та інші методи для зниження стресу та покращення ментального здоров'я. Centr також надає можливість користувачам спілкуватися з тренерами та дієтологами, отримуючи персональні рекомендації та підтримку. Цей інтегрований підхід допомагає

користувачам досягати своїх цілей у фітнесі та підтримувати баланс між фізичним та ментальним здоров'ям. (рис. 2.3) [28].

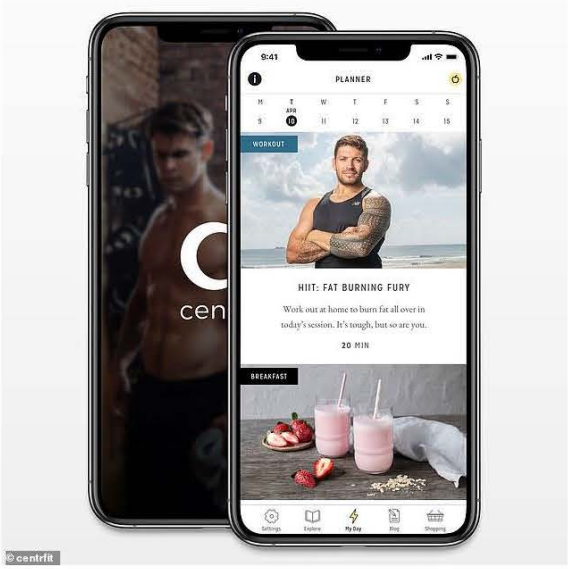


Рисунок 2.3 – Мобільний додаток Centr

Centr використовує бізнес-модель, що базується на передплаті (subscription-based model). Це означає, що користувачі отримують доступ до контенту та функцій додатку після оплати регулярної підписки (місячної або річної) [29].

Цільова аудиторія Centr досить широка, але можна виділити кілька основних груп [30]:

1. Прихильники здорового способу життя. Люди, які прагнуть покращити своє фізичне та ментальне здоров'я, шукають комплексний підхід до тренувань, харчування та відновлення.

2. Користувачі з середнім та високим рівнем доходу. Centr пропонує преміум-контент за платну підписку, тому його цільова аудиторія складається з людей, які готові інвестувати у своє здоров'я та добробут.

3. Користувачі, які шукають зручність та персоналізацію. Додаток пропонує індивідуальні плани тренувань та харчування, що підлаштовуються

під потреби та цілі користувача, а також зручний доступ до контенту з будь-якого місця та у будь-який час.

4. Люди, які шукають мотивацію та підтримку. Centr має активну спільноту користувачів, де можна знайти підтримку, поради та мотивацію для досягнення своїх цілей.

5. Початківці та досвідчені спортсмени. Додаток пропонує тренування різної складності, тому він підходить як для новачків, так і для тих, хто вже має досвід занять спортом.

Основні функції:

- Тренування. Пропонує різноманітні тренування для різних рівнів підготовки та цілей, включаючи силові, кардіо, функціональні, HIIT, бокс, йогу і пілатес. Всі тренування розроблені професійними тренерами та включають відеоінструкції.
- Плани харчування. Надає персоналізовані плани харчування з урахуванням ваших цілей, уподобань та дієтичних обмежень. Рецепти розроблені професійними дієтологами та включають різноманітні страви на кожен день.
- Медитації та сон. Пропонує медитації та практики для релаксації, зняття стресу та покращення сну.
- Експертні поради. Надає доступ до статей, відео та порад від експертів у сфері фітнесу, харчування та ментального здоров'я.
- Спільнота. Дозволяє спілкуватися з іншими користувачами, ділитися своїми досягненнями та отримувати підтримку.

Переваги:

1. Комплексний підхід. Пропонує комплексний підхід до здоров'я, охоплюючи фізичну активність, харчування та ментальне благополуччя.

2. Якісний контент. Тренування, плани харчування та медитації розроблені професіоналами та мають високу якість.

3. Персоналізація. Дозволяє налаштувати плани тренувань та харчування під свої потреби та цілі.

4. Мотивація. Пропонує різноманітні мотиваційні елементи, такі як відстеження прогресу, нагороди та спільнота підтримки.

Недоліки:

1. Вартість. Додаток платний, з щомісячною або щорічною підпискою.

2. Не підходить для всіх. Деякі тренування можуть бути занадто складними для початківців, а плани харчування можуть не підходити людям з певними дієтичними обмеженнями.

3. Обмежена кількість контенту. Хоча додаток регулярно оновлюється, кількість контенту може бути обмеженою порівняно з іншими додатками.

Для мобільних додатків використані ті самі технології як у минулих додатках [31].

Бекенд:

- Хмарні платформи. Amazon Web Services (AWS), Google Cloud Platform (GCP) або Microsoft Azure (для хостингу, зберігання даних та обчислень).
- Бази даних. PostgreSQL, MySQL або NoSQL бази даних (для зберігання даних користувачів, тренувань, планів харчування тощо).
- Мови програмування. Node.js, Python, Ruby, Java або Go (для розробки серверної логіки та API).
- Фреймворки. Express.js, Django, Ruby on Rails, Spring Boot або інші (для розробки веб-додатків та API).

Інші технології:

- Content Delivery Network (CDN). Для швидкої доставки медіа контенту (відео, зображення).
- Системи аналітики. Google Analytics, Firebase Analytics або Mixpanel (для збору даних про використання додатку та поведінку користувачів).

- Push-повідомлення. Firebase Cloud Messaging (FCM) або Apple Push Notification Service (APNs) (для надсилання сповіщень користувачам).
- Інструменти для A/B тестування. Optimizely, Firebase A/B Testing або інші (для експериментів та оптимізації додатку).

Це відмінний додаток для тих, хто шукає комплексний підхід до здоров'я та готовий інвестувати у якісний контент та персоналізовані рекомендації. Він підійде людям з різним рівнем підготовки, але може бути не найкращим вибором для початківців або людей з обмеженим бюджетом.

## 2.2 Огляд технологій та платформ для розробки мобільних додатків

Сучасний ринок мобільних додатків є надзвичайно динамічним та конкурентним середовищем, де успіх продукту залежить від багатьох факторів, включаючи його функціональність, зручність використання, продуктивність та здатність задовольнити потреби цільової аудиторії. Вибір правильних технологій та платформ для розробки є критичним кроком на шляху до створення конкурентоспроможного та інноваційного фітнес-додатку.

У цій частині ми заглибимося у світ сучасних інструментів та підходів до розробки мобільних додатків, зосереджуючись на тих, що є найбільш релевантними для створення фітнес-додатку. Ми проведемо детальний огляд нативних платформ Android та iOS, які дозволяють досягти максимальної продуктивності та інтеграції з апаратними можливостями пристроїв. Також розглянемо кросплатформні рішення, такі як React Native та Flutter, які пропонують ефективний спосіб розробки для обох платформ одночасно, скорочуючи час та ресурси, необхідні для створення додатку.

Крім того, ми проаналізуємо різні архітектурні підходи до розробки, такі як MVP (Model-View-Presenter), MVVM (Model-View-ViewModel) та Clean Architecture, які допомагають створювати структурований, легко підтримуваний та масштабований код. Також розглянемо важливі аспекти,

пов'язані з безпекою даних, оптимізацією продуктивності та забезпеченням сумісності з різними версіями операційних систем.

Android – це мобільна операційна система, заснована на ядрі Linux та іншому відкритому програмному забезпеченні. Вона розробляється переважно компанією Google і використовується на широкому спектрі пристроїв, включаючи смартфони, планшети, телевізори та інші [32].

Архітектура та компоненти (рис. 2.4) [33]:

1. Ядро Linux. Фундамент Android, що забезпечує управління пам'яттю, процесами, пристроями та безпекою. Це дозволяє Android використовувати переваги ключових функцій безпеки Linux та забезпечує ефективну взаємодію з апаратним забезпеченням пристроїв.

2. Апаратні абстракції (HAL). Шар, що відокремлює апаратне забезпечення від вищого рівня програмного забезпечення. Це дозволяє Android працювати на різних пристроях з різними конфігураціями апаратного забезпечення.

3. Бібліотеки. Набір готових компонентів для виконання різноманітних завдань, таких як робота з графікою (OpenGL ES, Vulkan), аудіо та відео (Media Framework), базами даних (SQLite), веб-контентом (WebKit) та іншими.

4. Android Runtime (ART). Середовище виконання для Android-додатків, що використовує компіляцію "на льоту" (JIT) та попередню компіляцію (AOT) для підвищення продуктивності та ефективності використання ресурсів. ART замінив Dalvik у версії Android 5.0 (Lollipop).

5. Native C/C++ Libraries – це набір бібліотек, написаних на мовах програмування C та C++, які надають доступ до низькорівневих функцій системи, таких як графіка, аудіо, відео, криптографія та інші. Ці бібліотеки використовуються для забезпечення високої продуктивності додатків, оскільки вони працюють ближче до апаратного забезпечення пристрою та операційної системи.

6. Фреймворк додатків. Набір API та компонентів, які дозволяють розробникам створювати додатки для Android. Фреймворк включає Activity Manager, Window Manager, Content Providers, View System, Notification Manager, Package Manager та інші.

7. Системні додатки – це набір стандартних додатків, що входять до складу операційної системи Android, забезпечуючи базову функціональність пристрою. До таких додатків належать телефон, контакти, повідомлення, браузер, камера та інші. Додаток телефону дозволяє здійснювати та приймати дзвінки, зберігати контакти, а також керувати журналом викликів. Додаток контактів забезпечує зручне зберігання та організацію контактної інформації, дозволяючи додавати, редагувати та видаляти контакти.

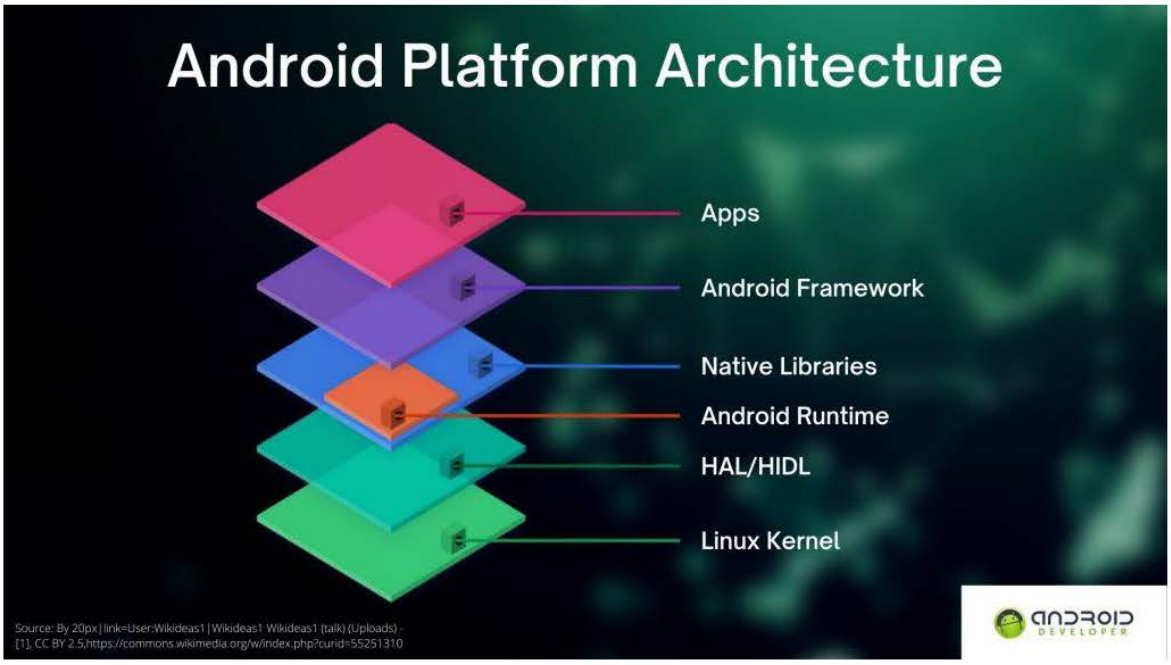


Рисунок 2.4 – Архітектура Android

Переваги Android:

- Відкритість. Android є відкритою платформою, що дозволяє розробникам вільно створювати та розповсюджувати додатки.

- Широкий вибір пристроїв. Величезна кількість пристроїв різних виробників працюють на Android, забезпечуючи широкий вибір для користувачів.
- Гнучкість. Android можна налаштувати під свої потреби, використовуючи різні теми, віджети та додатки.
- Великий магазин додатків. Google Play Store пропонує мільйони додатків на будь-який смак.
- Спільнота. Велика та активна спільнота розробників та користувачів, які допомагають один одному та діляться досвідом.

Недоліки Android:

- Фрагментація. Існує багато різних версій Android, що ускладнює розробку та тестування додатків.
- Безпека. Android більш вразливий до шкідливого програмного забезпечення, ніж деякі інші мобільні платформи, хоча Google активно працює над покращенням безпеки.
- Оновлення. Деякі виробники пристроїв повільно випускають оновлення для своїх смартфонів та планшетів.

Для розробки Android-додатків використовується мова програмування Java або Kotlin, а також інструменти Android Studio та Android SDK. Процес розробки включає створення інтерфейсу користувача, написання коду, тестування та публікацію додатку в Google Play Store [34].

Android є потужною та гнучкою платформою, яка пропонує безліч можливостей для розробників та користувачів. Завдяки своїй відкритості та широкому вибору пристроїв, Android залишається однією з найпопулярніших мобільних операційних систем у світі.

iOS – це мобільна операційна система, розроблена компанією Apple Inc. Вона є основою для таких популярних пристроїв, як iPhone, iPad та iPod Touch. iOS відома своїм інтуїтивно зрозумілим інтерфейсом, високою продуктивністю та безпекою [35].

Архітектура та компоненти (рис. 2.5) [36]:

1. Ядро Darwin. Основа iOS, що базується на ядрі XNU, яке поєднує в собі елементи Mach та BSD. Ядро Darwin забезпечує управління пам'яттю, процесами, файловою системою та мережевими функціями.

2. Core OS. Шар, що надає низькорівневі сервіси, такі як багатозадачність, управління пам'яттю, мережеві протоколи, безпека та інші.

3. Core Services. Набір фреймворків та бібліотек, що надають доступ до різноманітних функцій системи, таких як геолокація, контакти, календар, iCloud, мультимедіа, машинне навчання та багато іншого.

4. Media Layer. Шар, відповідальний за обробку графіки, аудіо та відео. Він включає такі фреймворки, як Core Graphics, Core Animation, Core Audio та Core Video.

5. Cocoa Touch. Шар, що надає інструменти для створення інтерфейсу користувача, обробки дотиків, управління жестами та інші функції, специфічні для сенсорних пристроїв.

6. Додатки. Верхній шар, що складається з вбудованих додатків Apple та додатків, встановлених користувачем з App Store.

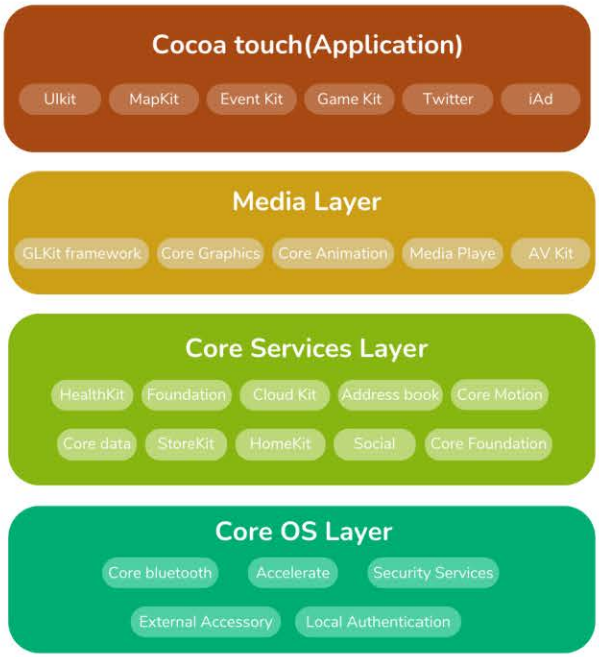


Рисунок 2.5 – Зображення архітектури iOS

Переваги iOS:

- Висока продуктивність. iOS оптимізована для роботи на апаратному забезпеченні Apple, що забезпечує високу швидкість та плавність роботи додатків.
- Безпека. iOS відома своїм високим рівнем безпеки, завдяки таким функціям, як шифрування даних, захист від шкідливого програмного забезпечення та обмеження на доступ до системних ресурсів.
- Інтуїтивний інтерфейс. iOS має простий та зрозумілий інтерфейс, який легко освоїти навіть новачкам.
- Якісні додатки. App Store пропонує великий вибір високоякісних додатків, які проходять суворий контроль якості перед публікацією.
- Екосистема. iOS тісно інтегрована з іншими продуктами та сервісами Apple, такими як iCloud, Apple Music, Apple Pay та іншими.

Недоліки iOS:

- Закритість. iOS є закритою платформою, що обмежує можливості розробників та користувачів у налаштуванні системи.
- Висока вартість пристроїв. Пристрої Apple, на яких працює iOS, зазвичай дорожчі за аналоги на Android.
- Обмежений вибір пристроїв. iOS доступна лише на обмеженій кількості пристроїв, що виробляються Apple.

Для розробки iOS-додатків використовується мова програмування Swift або Objective-C, а також інструменти Xcode та iOS SDK. Процес розробки включає створення інтерфейсу користувача, написання коду, тестування та публікацію додатку в App Store [37].

iOS є потужною та зручною операційною системою, яка пропонує високу продуктивність, безпеку та інтуїтивний інтерфейс. Вона є відмінним вибором для користувачів, які цінують якість, надійність та інтеграцію з екосистемою Apple.

React Native – це фреймворк з відкритим кодом, розроблений компанією Meta (раніше Facebook), який дозволяє створювати нативні мобільні додатки для платформ iOS та Android, використовуючи JavaScript та React [6].

React Native використовує концепцію "мостів" (bridges), які дозволяють JavaScript-коду взаємодіяти з нативними компонентами iOS та Android. Це означає, що ви можете писати код на JavaScript, а React Native перетворює його на нативні компоненти, що забезпечує високу продуктивність та нативний вигляд додатку [38].

Основні компоненти:

1. JavaScript. Основна мова програмування для написання бізнес-логіки та компонентів інтерфейсу користувача.

2. React. Бібліотека JavaScript для створення компонентів інтерфейсу користувача, які можна використовувати повторно та комбінувати для побудови складних інтерфейсів.

3. Нативні компоненти. Набір готових компонентів для iOS та Android, таких як кнопки, текстові поля, зображення та інші елементи інтерфейсу. React Native надає обгортки (wrappers) для цих компонентів, дозволяючи використовувати їх у JavaScript-коді.

4. Мости (Bridges). Механізм, що забезпечує зв'язок між JavaScript-кодом та нативними компонентами. Він дозволяє передавати дані та викликати методи нативних компонентів з JavaScript, і навпаки.

Переваги React Native:

- Кросплатформність. Один код для iOS та Android, що значно скорочує час та витрати на розробку.
- Швидка розробка. Завдяки використанню JavaScript та React, розробка додатків на React Native відбувається швидше, ніж при використанні нативних інструментів.
- Живе перезавантаження (Hot Reloading). Можливість бачити зміни в додатку миттєво, без необхідності перекомпіляції.

- Велика спільнота. React Native має велику та активну спільноту розробників, які створюють бібліотеки, інструменти та навчальні матеріали.

Недоліки React Native:

- Продуктивність. Хоча React Native забезпечує досить високу продуктивність, вона може бути нижчою, ніж у нативних додатків, особливо для складних анімацій або обчислень.
- Нативні функції. Для доступу до деяких специфічних функцій платформ, таких як ARKit (iOS) або деякі апаратні можливості, може знадобитися написання нативного коду.

React Native є потужним інструментом для розробки кросплатформенних мобільних додатків. Він дозволяє швидко створювати додатки з нативним виглядом та відчуттям, використовуючи знайомі технології JavaScript та React. Проте, слід враховувати деякі обмеження щодо продуктивності та доступу до нативних функцій.

Flutter – це фреймворк з відкритим кодом, розроблений Google, для створення високопродуктивних, візуально привабливих та кросплатформенних мобільних додатків для iOS та Android, а також веб- та десктопних застосунків [7].

На відміну від React Native, Flutter не використовує мости для зв'язку з нативними компонентами. Замість цього, він використовує власний високопродуктивний рушій рендерингу Skia, який малює інтерфейс користувача безпосередньо на полотні пристрою. Це дозволяє Flutter досягати швидкості 60 кадрів в секунду та забезпечує плавну анімацію та швидку реакцію інтерфейсу [39].

Основні компоненти:

1. Dart. Мова програмування, на якій пишуться Flutter-додатки. Dart є об'єктно-орієнтованою мовою з C-подібним синтаксисом, що підтримує як статичну, так і динамічну типізацію. Вона має швидкий компілятор та віртуальну машину, що забезпечує високу продуктивність додатків.

2. Віджети. Базові будівельні блоки інтерфейсу користувача у Flutter. Віджети можуть бути як простими (текст, зображення, кнопки), так і складними (списки, таблиці, форми). Flutter надає велику бібліотеку готових віджетів, які можна комбінувати та налаштовувати для створення будь-якого інтерфейсу.

3. Рушій рендерингу Skia. Високопродуктивний графічний рушій з відкритим кодом, який використовується для малювання віджетів на екрані пристрою. Skia забезпечує швидку та плавну роботу інтерфейсу, навіть на слабких пристроях.

Переваги Flutter:

- Швидкість та продуктивність. Завдяки власному рушію рендерингу Skia, Flutter-додатки працюють дуже швидко та плавно.
- Кросплатформність. Один код для iOS та Android, що значно скорочує час та витрати на розробку.
- Гаряче перезавантаження (Hot Reload). Миттєве відображення змін у додатку під час розробки, що значно пришвидшує процес розробки.
- Виразний та гнучкий інтерфейс. Flutter надає велику кількість настроюваних віджетів та інструментів для створення унікального та привабливого інтерфейсу користувача.
- Відмінна документація та підтримка. Flutter має докладну документацію та велику спільноту розробників, які завжди готові допомогти.

Недоліки Flutter:

- Відносно молодий фреймворк. Flutter є відносно новою технологією, і можуть виникати деякі проблеми та обмеження.
- Розмір додатків. Flutter-додатки можуть бути трохи більшими за розміром, ніж нативні додатки.
- Обмежена підтримка певних функцій. Flutter може мати обмежену підтримку деяких специфічних функцій, таких як 3D Touch на iOS або складні анімації на Android.

Flutter є потужним та перспективним фреймворком для розробки кросплатформенних мобільних додатків. Він пропонує високу продуктивність, чудовий інтерфейс користувача та швидкий процес розробки. Якщо ви шукаєте спосіб створити красивий, швидкий та сучасний мобільний додаток для iOS та Android, Flutter може бути відмінним вибором.

Архітектурні підходи до розробки мобільних додатків відіграють важливу роль у створенні структурованого, масштабованого та легко підтримуваного коду. Серед найпопулярніших підходів можна виділити MVP (Model-View-Presenter), MVVM (Model-View-ViewModel) та Clean Architecture. Кожен з них має свої переваги та недоліки, і вибір конкретного підходу залежить від специфіки проекту та уподобань команди розробників.

MVP пропонує чіткий поділ відповідальності між трьома компонентами: Model (дані), View (інтерфейс користувача) та Presenter (логіка). Presenter виступає посередником між Model та View, обробляючи дані та оновлюючи інтерфейс. Цей підхід забезпечує гарну тестуваність та легкість підтримки коду, але може призвести до збільшення кількості класів та складності взаємодії між ними [40].

MVVM є розширенням MVP, де ViewModel відповідає за підготовку даних для відображення у View. ViewModel може містити логіку для перетворення даних та обробки подій користувача. Цей підхід спрощує зв'язок між Model та View, але може призвести до збільшення складності ViewModel та ускладнити тестування [41].

Clean Architecture пропонує більш абстрактний підхід, розділяючи додаток на незалежні шари: Entities (бізнес-логіка), Use Cases (випадки використання), Interface Adapters (адаптери інтерфейсу) та Frameworks & Drivers (фреймворки та драйвери). Цей підхід забезпечує високу гнучкість, незалежність від фреймворків та легкість тестування, але може бути складним для розуміння та реалізації, особливо для невеликих проектів [42].

Вибір архітектурного підходу залежить від багатьох факторів, таких як розмір та складність проекту, досвід команди розробників, вимоги до

тестування та підтримки. Важливо враховувати переваги та недоліки кожного підходу, щоб обрати той, який найкраще відповідає потребам проекту та забезпечує його успішну реалізацію.

## 2.3 Вибір технологій та методів розв'язання поставленої задачі

Для успішної реалізації мобільного фітнес-додатку було обрано сучасний та ефективний стек технологій, що дозволяє створити високоякісний продукт, який відповідатиме потребам користувачів. TypeScript, як підмножина JavaScript, забезпечує статичну типізацію, що покращує читабельність коду, спрощує рефакторинг та виявляє помилки на етапі компіляції, що особливо важливо для фітнес-додатку, де точність обчислень та надійність логіки є критичними [43]. React Native, як фреймворк для розробки кросплатформенних мобільних додатків. Realm DB, мобільна база даних, оптимізована для роботи з мобільними пристроями, забезпечує ефективне зберігання та синхронізацію даних про тренування, активність користувача, плани харчування та інші важливі показники [44]. Reanimated, бібліотека для React Native, дозволяє створювати плавні та високопродуктивні анімації, що підвищують привабливість додатку та візуалізують прогрес користувача [45]. Zustand, бібліотека для управління станом у React-додатках, забезпечує централізоване сховище даних та зручні інструменти для оновлення стану, що спрощує роботу з даними користувача та налаштуваннями додатку [46]. Обраний стек технологій дозволяє створити сучасний, функціональний та зручний фітнес-додаток, який буде відповідати потребам користувачів та забезпечує високу продуктивність та надійність.

Для ефективної розробки та управління проектом було обрано ряд інструментів, які забезпечують продуктивність, зручність роботи та надійність процесу. WebStorm, як потужне інтегроване середовище розробки (IDE), спеціально розроблене для JavaScript та TypeScript, надає розробникам широкий спектр функцій, включаючи інтелектуальне автодоповнення коду,

налагодження, рефакторинг, вбудовані інструменти тестування та підтримку популярних фреймворків [47]. Це значно пришвидшує та спрощує процес розробки, дозволяючи зосередитися на створенні якісного продукту.

Realm Studio, візуальний інструмент для роботи з базою даних Realm, є незамінним помічником у процесі розробки. Він дозволяє зручно переглядати структуру бази даних, редагувати дані, створювати запити та відстежувати зміни в реальному часі. Це значно полегшує процес розробки та налагодження функціоналу, пов'язаного з даними, забезпечуючи швидкий доступ до інформації та можливість експериментувати з різними варіантами зберігання та обробки даних [48].

GitHub, провідна платформа для хостингу коду та управління версіями, забезпечує надійне зберігання коду проекту, зручну спільну роботу над ним та відстеження змін. Завдяки GitHub розробники можуть легко співпрацювати, переглядати історію змін, створювати гілки для експериментів та об'єднувати код, що є важливим для командної розробки та забезпечення цілісності проекту [49].

GitHub Copilot Chat, інноваційний інструмент на основі штучного інтелекту, надає розробникам підказки, пропонує варіанти коду та навіть автоматично генерує код на основі контексту та опису завдання. Це допомагає розробникам швидше та ефективніше вирішувати завдання програмування, підвищуючи продуктивність та якість коду, особливо при роботі зі складними або повторюваними задачами [50].

У цьому розділі було проведено огляд ключових технологій та платформ, що використовуються для розробки мобільних додатків, з акцентом на їх застосуванні у створенні фітнес-додатку. Були розглянуті як нативні платформи Android та iOS, так і кросплатформні рішення React Native та Flutter, кожне з яких має свої переваги та недоліки.

Детальний аналіз платформ Android та iOS показав, що обидві вони пропонують потужні інструменти та широкі можливості для розробки, але мають відмінності у підходах до дизайну, розробки та дистрибуції додатків.

React Native та Flutter, як кросплатформні рішення, дозволяють скоротити час та витрати на розробку, але можуть мати обмеження у продуктивності та доступі до деяких нативних функцій.

Вибір конкретного стеку технологій для розробки фітнес-додатку залежить від багатьох факторів, таких як цільова аудиторія, бюджет, вимоги до функціональності та досвід команди розробників. У цьому проекті було обрано TypeScript, React Native, Realm DB, Reanimated та Zustand, як оптимальне поєднання технологій, що забезпечує високу продуктивність, зручність розробки та широкий функціонал для створення сучасного фітнес-додатку.

Крім того, було розглянуто ряд інструментів, які сприяють ефективній розробці та управлінню проектом, таких як WebStorm, Realm Studio, GitHub та GitHub Copilot Chat. Ці інструменти забезпечують продуктивність, зручність роботи та надійність процесу розробки, дозволяючи команді зосередитися на створенні якісного продукту.

# РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

## 3.1 Проектування архітектури та бази даних мобільного додатку

Створення ефективного та масштабованого мобільного фітнес-додатку неможливе без ретельного проектування його архітектури та структури бази даних. Цей етап є ключовим для забезпечення стабільності, продуктивності та зручності підтримки додатку у майбутньому, а також для забезпечення безперебійної роботи та позитивного користувацького досвіду. У цьому розділі ми зосередимося на створенні архітектури мобільного фітнес-додатку, яка буде забезпечувати його модульність, гнучкість та легкість розширення, що є критично важливим у динамічному середовищі мобільних технологій. Крім того, ми приділимо особливу увагу структурі бази даних, яка є основою для зберігання та обробки даних.

React Native, як фреймворк, не нав'язує конкретної архітектури, надаючи розробникам свободу вибору. Проте, є деякі поширені підходи, які часто використовуються при розробці мобільних додатків на React Native. Основним підходом є компонентна архітектура яка може бути подана у вигляді модульної архітектури додатку.

Компонентна архітектура в React Native – це підхід до розробки, де інтерфейс користувача розбивається на невеликі, незалежні та багаторазові компоненти. Кожен компонент відповідає за певну частину інтерфейсу та має свою власну логіку, стан та відображення. Це підвид MVVM архітектури який дозволяє відокремити код призначений для інтерфейсу користувача від решти логіки [51]. Наприклад логіки взаємодії із базою даних. Приклад компонентної архітектури наведено на (рис. 3.1). Використання компонентної архітектури вчить надає та покращує навички застосування методів компонентної розробки програмного забезпечення.

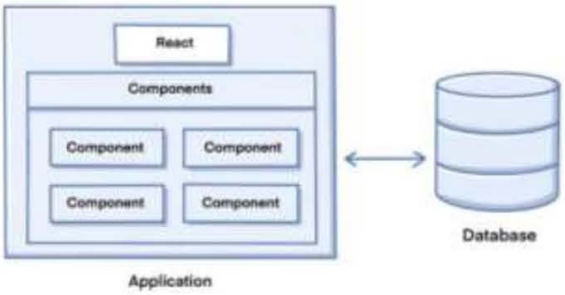


Рисунок 3.1 – Зображення компонентної архітектури

Для того щоб розділити наші компоненти за своїм призначенням, ми розділимо їх на окремі модулі, які будуть виконувати певні задачі застосунку.

На приклад у додатку будуть такі елементи як навігація (рис. 3.2) чи календар (рис. 3.3), які будуть розділені на окремі модулі.

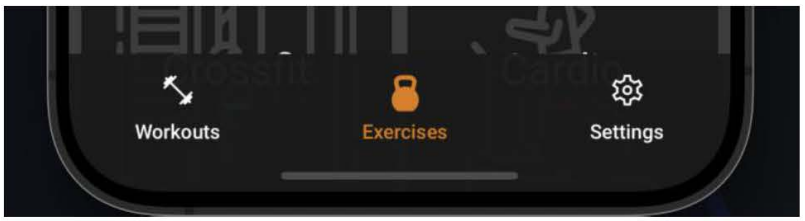


Рисунок 3.2 – Навігація мобільного додатку

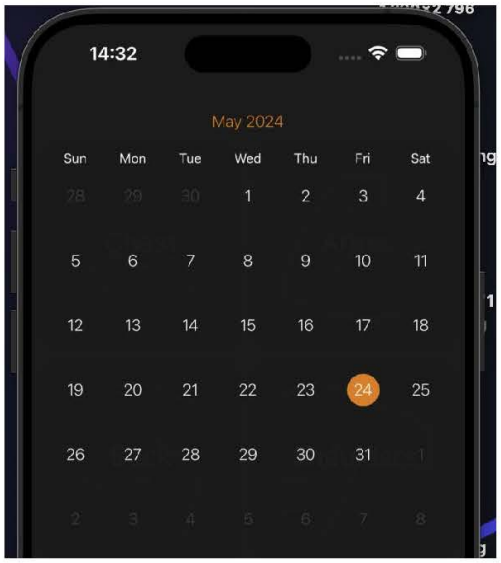


Рисунок 3.3 – Календар мобільного додатку

Тобто кожен модуль це окремий набір компонентів, які виконують певну задачу. Таки підхід полегшує розробку та подальше розширення додатку за допомогою інкапсуляції кожного модуля.Таким чином ми можемо сказати що модульна архітектура додатку – це підхід до розробки програмного забезпечення, де додаток розбивається на окремі, незалежні модулі, кожен з яких відповідає за певну функціональність або частину системи. Ці модулі можуть бути розроблені, протестовані та розгорнуті незалежно один від одного.

Для того щоб модулі могли взаємодіяти із даними які ми можемо використовувати в інших компонентах ми будемо використовувати глобальний стан додатку (Global state).

Глобальний стан додатку – це механізм, який дозволяє зберігати та керувати даними, доступними з будь-якого компонента вашого додатку, незалежно від їх ієрархії. Це означає, що ви можете централізовано зберігати інформацію, таку як дані користувача, налаштування теми, стан автентифікації тощо, і легко отримувати до неї доступ з будь-якої частини вашого додатку [52] .

Для реалізації ми будемо використовувати бібліотеку Zustand, яка дозволяє легко створити централізоване сховище додатку.

У сховищі глобального стану ми будемо зберігати дані про тему нашого застосунку для можливості змінювати кольори візуальних компонентів відповідно до обраної теми. Також будемо зберігати дані про дату, для коректної взаємодії інших модулів із календарем.

Кожен модуль та компонент використовуватиме окремі UI (User interface) компоненти, які будуть незалежні від будь-яких модулів. До таких компонентів належать:

- Кнопки.
- Текстові поля.
- Модальні вікна.
- Радіо-кнопки (Radio button).

Із цих компонентів будується візуальна частина всього застосунку.

Тобто загальна архітектурна ієрархія додатку буде виглядати таким чином (рис. 3.4).

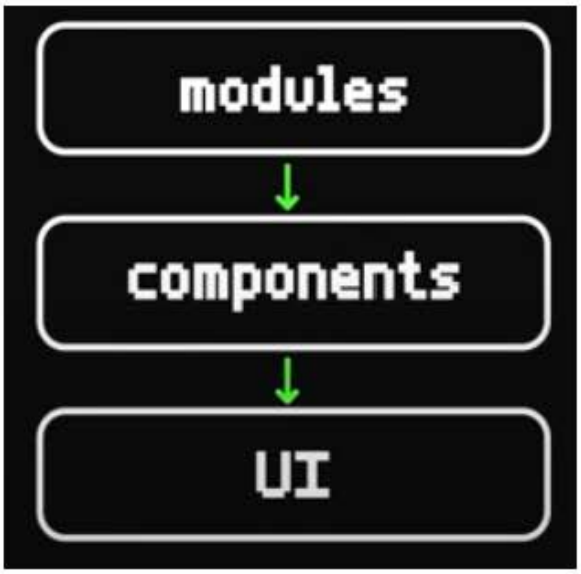


Рисунок 3.4 – Архітектурна ієрархія додатку

Для забезпечення гнучкості та масштабованості мобільного фітнес-додатку, важливо дотримуватися принципів модульної архітектури та відокремити взаємодію з базою даних від інших модулів. Це дозволить створити абстракцію над базою даних, що зробить її легкою для заміни або модифікації у майбутньому, без необхідності вносити зміни в інші частини коду. Такий підхід забезпечує легкість підтримки та розширення функціоналу додатку, а також спростить тестування та налагодження.

Використання патерну Repository для взаємодії з базою даних дозволяє приховати деталі реалізації та забезпечити єдиний інтерфейс для роботи з даними. Це зробить код більш читабельним та зрозумілим, а також дозволить легко замінити Realm DB на іншу базу даних у разі потреби, без необхідності змінювати логіку роботи з даними в інших модулях. Крім того, такий підхід сприятиме дотриманню принципів SOLID, що є важливим для створення якісного та надійного програмного забезпечення [53].

Виходячи із цього загальна архітектура мобільного додатку буде виглядати таким чином (рис. 3.5).

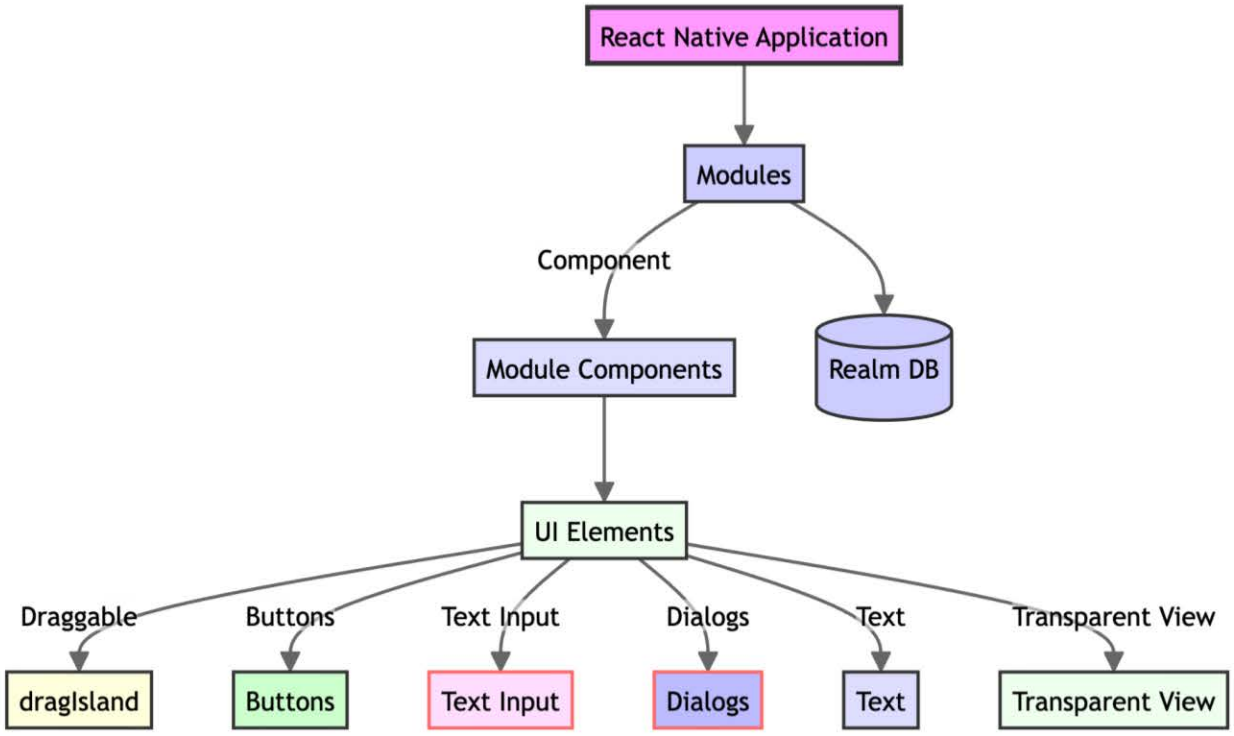


Рисунок 3.5 – Архітектура мобільного додатку

Створення архітектури надає можливість застосування на практиці ефективних підходів щодо проектування програмного забезпечення та застосовувати на практиці фундаментальні концепції, парадигми і основні принципи функціонування мовних, інструментальних і обчислювальних засобів інженерії програмного забезпечення.

Для проектування бази даних у мобільному фітнес-додатку з використанням Realm DB необхідно врахувати низку важливих факторів, що забезпечують ефективність, масштабованість та зручність роботи з даними.

Realm DB – це NoSQL база даних, що працює безпосередньо на пристрої користувача. Вона відрізняється від традиційних реляційних баз даних своєю структурою та підходами до роботи з даними. Тому при проектуванні необхідно враховувати її особливості:

- Об'єктно-орієнтований підхід. Realm DB працює з об'єктами, а не з таблицями, що дозволяє створювати гнучкі та інтуїтивно зрозумілі моделі даних.
- Підтримка відносин. Realm DB дозволяє створювати зв'язки між об'єктами, що забезпечує цілісність даних та спрощує роботу з ними.
- Реактивність. Realm DB автоматично оновлює дані у всіх частинах додатку при їх зміні, що дозволяє створювати динамічні та інтерактивні інтерфейси.

Фактори, що впливають на проектування:

- Типи даних. Необхідно визначити, які типи даних будуть зберігатися у базі (наприклад, інформація про користувача, тренування, плани харчування, результати тощо).
- Структура даних. Слід розробити логічну структуру даних, враховуючи зв'язки між об'єктами та вимоги до швидкості доступу до даних.
- Обсяг даних. Важливо оцінити очікуваний обсяг даних, щоб забезпечити достатню продуктивність та масштабованість бази даних.
- Вимоги до синхронізації. Якщо додаток потребує синхронізації даних між пристроями або з хмарним сервісом, необхідно врахувати це при проектуванні бази даних.
- Безпека. Особливу увагу слід приділити безпеці даних, враховуючи, що Realm DB зберігає дані на пристрої користувача.

Відповідно до цих факторів база даних буде виглядати наступним чином (рис. 3.6).

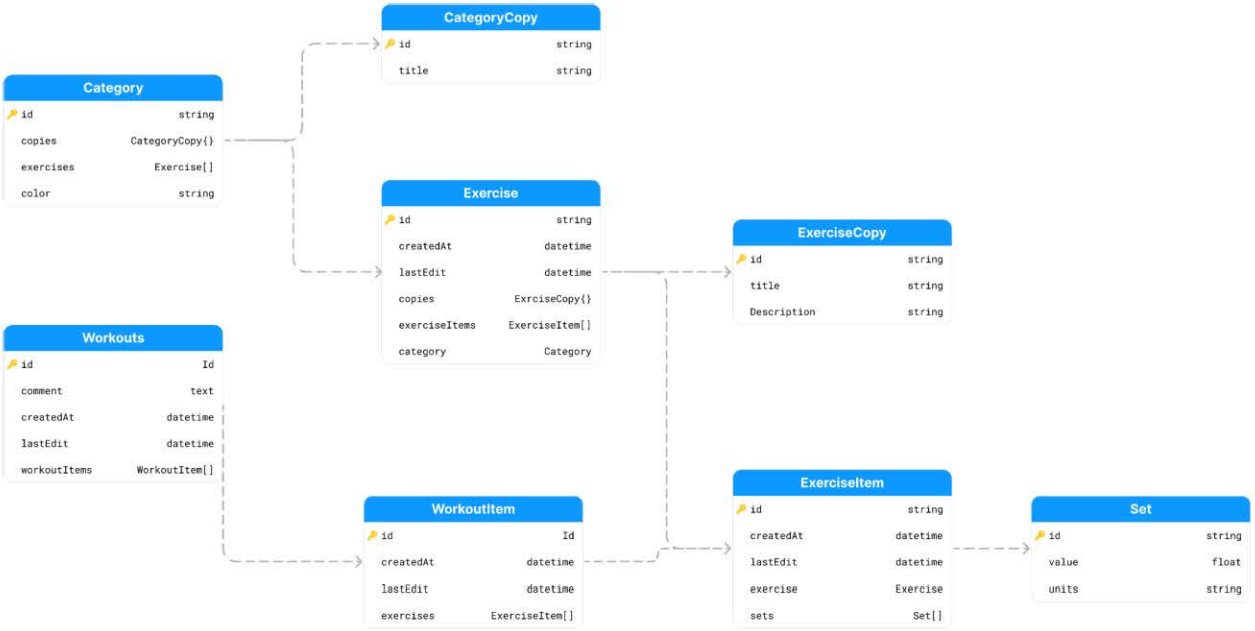


Рисунок 3.6 – Загальна структура бази даних мобільного додатку

Структура зображена у вигляді схем бази даних які будуть описані у вигляді коду.

База даних, представлена на діаграмі, призначена для зберігання та організації даних фітнес-додатку. Вона складається з кількох взаємопов'язаних сутностей, що дозволяють зберігати інформацію про категорії вправ, самі вправи, їх варіації, а також тренування користувачів.

Кожна категорія вправ (Category) має назву та колір для візуального розрізнення в інтерфейсі додатку. Крім того, сутність CategoryCopy дозволяє зберігати переклади назви категорії різними мовами.

Вправи (Exercise), кожна вправа має назву, опис та може містити кілька кроків або варіантів виконання (ExerciseItem). Також передбачено можливість зберігати переклади назви вправи різними мовами (ExerciseCopy).

Тренування (Workout) складаються з набору вправ (WorkoutItem), кожна з яких може мати кілька підходів (Set). Для кожного підходу зберігається інформація про вагу, кількість повторень та одиниці вимірювання. Крім того, тренування можуть мати коментарі та інформацію про час створення та останньої зміни.

Така структура бази даних забезпечує гнучкість та масштабованість, дозволяючи легко додавати нові категорії, вправи та варіації виконання. Зв'язки між сутностями забезпечують цілісність даних та дозволяють легко отримувати необхідну інформацію для відображення в додатку та аналізу тренувань користувачів.

## 3.2 Розробка додатку з використанням обраних технологій

У попередніх розділах було проведено детальний аналіз існуючих мобільних фітнес-додатків, що дозволило виявити ключові тенденції, переваги та недоліки конкурентних рішень. На основі цього аналізу, а також враховуючи потреби цільової аудиторії та сучасні вимоги до мобільних додатків, було сформовано чіткий перелік функціональних та нефункціональних вимог до розроблюваного фітнес-додатку. У цьому розділі буде детально описано процес розробки мобільного фітнес-додатку з використанням обраних інструментів та підходів.

Для початку нам потрібно встановити необхідні інструменти та ініціалізувати проект.

Ініціалізація React Native застосунку - це процес створення нового проекту та налаштування його базової структури, щоб почати розробку [54]. Зазвичай, цей процес включає в себе такі кроки:

1. Встановлення необхідних інструментів:
2. Node.js та npm (або yarn) – менеджери пакетів JavaScript, необхідні для встановлення та управління залежностями проекту.
3. React Native CLI (Command Line Interface) – інструмент командного рядка для створення та управління React Native проектами.
4. Android Studio та/або Xcode – середовища розробки для платформ Android та iOS відповідно.

Створення нового проекту:

1. Відкрити термінал або командний рядок.

2. Перейти до директорії, де повинен знаходитися проект.

3. Виконати команду npx create-react-native-app <назва_проекту>. React Native CLI автоматично створить новий проект з усіма необхідними файлами та папками.

Запуск проекту:

1. Виконати команду npx react-native run-android для запуску на Android-емуляторі або підключеному пристрої.

2. Виконати команду npx react-native run-ios для запуску на iOS-симуляторі або підключеному пристрої.

Також для початку розробки потрібно запустити редактор коду або середу розробки, у нашому випадку це буде Webstorm. Кінцевий вигляд сетапу для розробки зображено на (рис. 3.7).

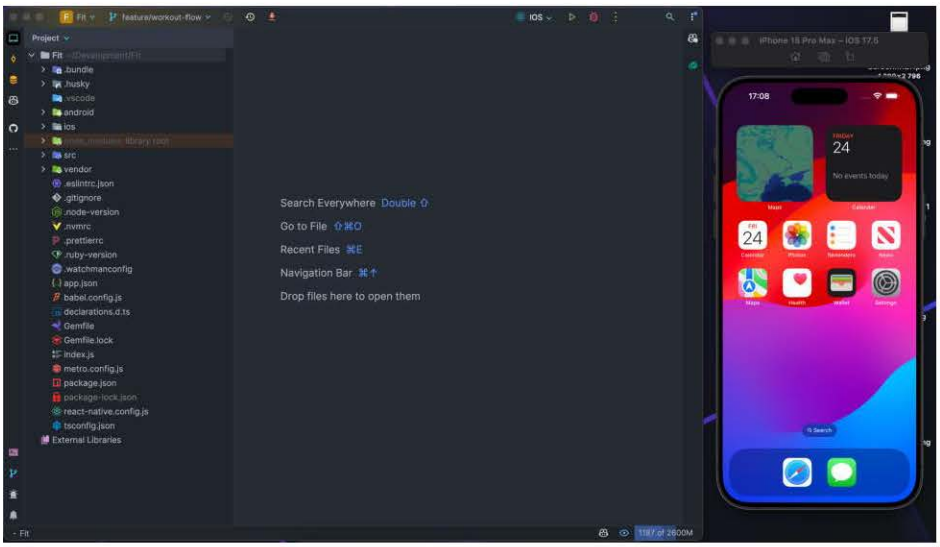


Рисунок 3.7 – Сетап для розробки мобільного додатку

Після ініціалізації проекта потрібно встановити необхідно бібліотеки та залежності, за допомогою команди npm install, деякі бібліотеки потребують змін у нативному коді, тобто зміну конфігурацій у папках ios або android.

Тепер потрібно створити структуру проекту відповідно до вимог.

- assets. Папка для статичних ресурсів (зображення, шрифти тощо).
- components. Папка для компонентів інтерфейсу користувача.

- db. Папка для файлів, пов'язаних з базою даних.
- hooks. Папка для власних React хуків.
- locales. Папка для файлів локалізації.
- modules. Папка для модулів, що об'єднують логіку та компоненти.
- store. Папка для файлів, пов'язаних з глобальним станом додатку
- styles. Папка для файлів стилів.
- types. Папка для файлів TypeScript типів.
- ui. Папка для компонентів інтерфейсу.
- utils. Папка для допоміжних функцій та утиліт.
- App.tsx. Головний компонент додатку.
- asyncStorageValues.ts. Файл для роботи з локальним сховищем даних AsyncStorage.
- constants.ts. Файл для оголошення констант.
- emitter.ts. Файл для роботи з подіями (EventEmitter).

Коли налаштовано інфраструктуру додатку, можно починати розробку компонентів.

Створення компонентів у React Native починається з чіткого визначення їхньої мети та структури. Потрібно розуміти, яку функціональність компонент виконуватиме, які дані (props) він отримуватиме та як він буде взаємодіяти з іншими елементами інтерфейсу.

Далі створюється файл компонента у відповідній папці проекту, дотримуючись прийнятих конвенцій іменування. У цьому файлі пишеться код компонента, імпортуючи необхідні модулі та компоненти з React Native. Оголошується функціональний або класовий компонент, описується його логіка обробляються дані та події, оновлюється стан (якщо потрібно) та повертається JSX-код, що описує структуру компонента.

Стилі компонента зазвичай створюються окремо за допомогою StyleSheet.create, а потім застосовуються до елементів компонента через властивість style.

Після створення компонент можна використовувати в інших компонентах або у головному компоненті додатку, передаючи йому необхідні дані через props.

Використання TypeScript для визначення типів даних та інтерфейсів компонентів є важливим кроком у розробці додатків на React Native. TypeScript дозволяє розробникам точно визначати типи властивостей (props) та стану (state) компонентів, що значно підвищує надійність коду. Завдяки статичній типізації можна виявити багато помилок ще на етапі компіляції, що зменшує кількість багів у коді та спрощує процес відлагодження. TypeScript також покращує автозаповнення та документацію в середовищі розробки, роблячи код зрозумілішим та легшим для підтримки.

Створення компонентів у React Native – це ітеративний процес, який вимагає постійного вдосконалення та рефакторингу коду. Цей процес включає декілька ключових етапів, починаючи від початкового проектування та розробки до тестування та оптимізації. Перш за все, важливо ретельно планувати структуру компонентів, враховуючи їхню функціональність та взаємодію з іншими частинами додатку.

Зміни, що вносяться в коді React Native додатку, зазвичай відображаються в реальному часі на запущеному емуляторі або фізичному пристрої завдяки функції "Hot Reloading" або "Fast Refresh". Ці механізми дозволяють оновлювати інтерфейс користувача без повної перезавантаження додатку, що значно прискорює розробку та покращує досвід розробника. Завдяки цьому, можна швидко експериментувати з різними варіантами коду та стилів, бачачи результат змін миттєво, що робить процес розробки більш інтерактивним та продуктивним. Приклад наведено на (рис. 3.8).

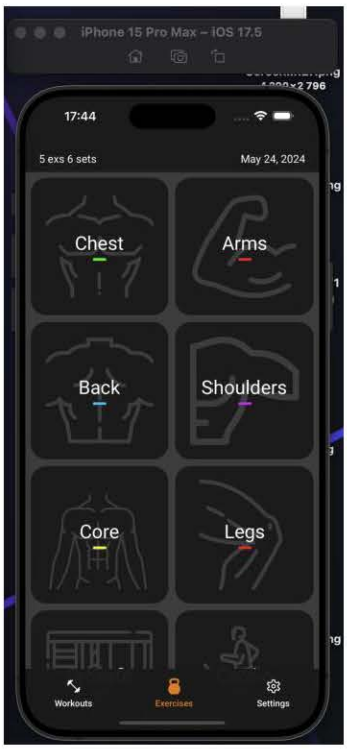


Рисунок 3.8 – Емулятор мобільного пристрою із запущеним додатком.

Розглянемо створення основних складових додатку. Однією із головних складових є навігація застосунку, її реалізовано за допомогою бібліотеки react-native-navigation.

React Navigation – це потужна бібліотека, яка є стандартом де-факто для реалізації навігації у React Native додатках. Вона дозволяє створювати різноманітні типи навігаційних структур, такі як стекова навігація, навігація вкладками, drawer-навігація [55].

Для додатку використано навігацію вкладками, це популярний патерн у мобільних додатках, де набір вкладок розташований у нижній частині екрана. Ця бібліотека надає вже готові варіанти реалізації, але вони не підходять оскільки потрібно зробити навігацію анімованою, тому було вирішено зробити кастомне рішення, за допомогою бібліотеки Reanimated. У результаті ми маємо готовий модуль який можно модіфикувати у майбутньому, структуру модуля наведено на (рис. 3.9).

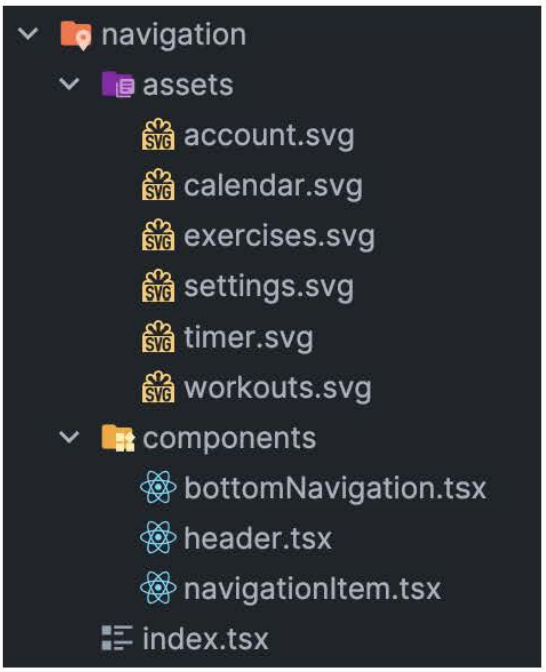


Рисунок 3.8 – Структура модуля навігації додатку.

У результаті ми маємо навігацію яка при зміні екрану відображає анімовану зміну індикатора сторінки, при натисканні на елемент навігації, іканка “підскакує” відносно назви екрану.

Іншою важливою складовою додатку є модуль TopDrawer (Верхня панель), який відкривається за допомогою жеста змахування вниз. В цьому модулі розміщено календар, який користувач відкриває для того щоб змінити дату тренування.

Цей модуль також було реалізовано за допомогою бібліотеки Reanimated, та React Native Gesture Handler.

React Native Gesture Handler – це потужна бібліотека, яка розширює можливості React Native у сфері обробки жестів та дотиків користувачів. Вона надає доступ до нативних API обробки жестів на платформах Android та iOS, забезпечуючи плавність, точність та високу продуктивність.

Для початку потрібно визначити математичні константи для коректної роботи анімації та жестів.

Визначено параметри анімації, такі як висота (DRAWER_HEIGHT), позиції закритого та відкритого стану (DRAWER_CLOSED_POSITION,

DRAWER_OPENED_POSITION), швидкість жестів для швидкого відкриття та точка перемикання стану (QUICK_OPEN_VELOCITY, OPEN_POINT).

Реалізація алгоритму відповідно до наведених констант:

- Встановити початкове зміщення offset у закриту позицію DRAWER_CLOSED_POSITION.
- Встановити початковий стан isOpened у false.
- Змінювати зміщення offset відповідно до руху пальця по вертикалі.
- Якщо швидкість руху перевищує QUICK_OPEN_VELOCITY вгору, відкрити панель.
- Якщо швидкість руху перевищує QUICK_OPEN_VELOCITY вниз, закрити панель.
- Якщо абсолютне значення зміщення менше за OPEN_POINT, відкрити панель.
- В іншому випадку, закрити.
- При тапі на фон, що перекриває екран закрити панель.

Зображення структури модуля на (рис. 3.10).

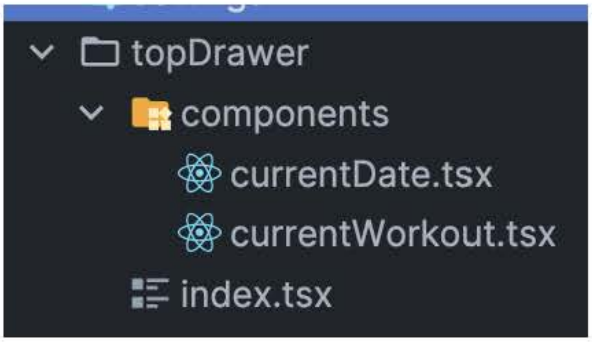


Рисунок 3.10 – Структура модуля TopDrawer.

У результаті бачимо плавну анімацію відкриття на (рис. 3.11).

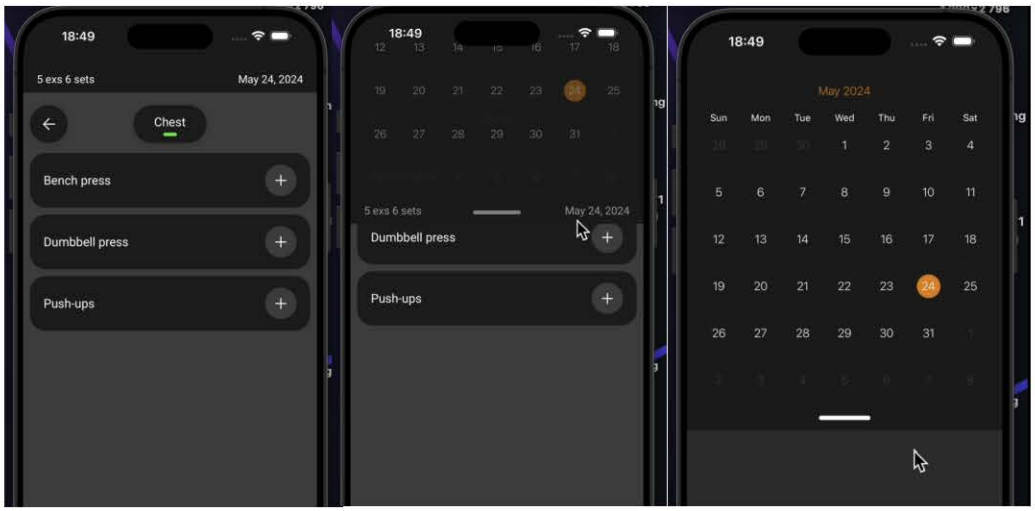


Рисунок 3.11 – Етапи анімації відкриття TopDrawer.

Наступною важливою частиною є модуль налаштувань. Це той модуль що може конфігурувати додаток відповідно до вимог. На даному етапі розробки це зміна теми та мови додатку.

Для збереження конфігурації використано бібліотеку AsyncStorage.

AsyncStorage – це проста, не синхронізована система зберігання ключ-значення, вбудована у React Native, яка забезпечує ефективне збереження даних на локальному пристрої користувача. Вона дозволяє зберігати невеликі обсяги даних, таких як налаштування, кеш або токени авторизації, що є критично важливими для забезпечення персоналізованого користувацького досвіду та підтримки сесій.

Однією з ключових особливостей AsyncStorage є його асинхронність, що означає, що операції читання та запису даних виконуються у фоновому режимі. Це дозволяє уникнути блокування основного потоку виконання, що забезпечує плавну роботу інтерфейсу користувача. Користувачі можуть продовжувати взаємодіяти з додатком без затримок, навіть коли відбуваються операції з даними [56].

Компонент ThemeSwitch (Перемикач теми) дозволяє користувачеві змінювати тему оформлення додатку. Він відображає кнопку налаштування,

при натисканні на яку відкривається діалогове вікно з варіантами тем: системна, світла та темна.

Вибрана тема зберігається у глобальному сховищі стану, а також у локальному сховищі AsyncStorage, щоб зберегти налаштування користувача між сесіями.

При зміні теми, компонент оновлює глобальний стан, що автоматично призводить до ре-рендерингу всіх компонентів, які використовують цей стан. Таким чином, зміна теми відображається миттєво у всьому додатку.

Використання AsyncStorage дозволяє зберегти вибір користувача навіть після закриття та повторного відкриття додатку. При ініціалізації компонента, він зчитує збережену тему з AsyncStorage та встановлює її як поточну.

Компонент LanguageSwitch (Перемикач мови) надає користувачеві можливість змінювати мову інтерфейсу додатку. Він відображає кнопку налаштування, при натисканні на яку відкривається діалогове вікно з варіантами мов: системна, англійська та українська.

Вибрана мова зберігається у локальному сховищі AsyncStorage, щоб зберегти налаштування користувача між сесіями. Також мова оновлюється у бібліотеці i18next, що відповідає за інтернаціоналізацію додатку.

При зміні мови, компонент зберігає вибір користувача в AsyncStorage, оновлює поточну мову в i18next, що призводить до автоматичного перекладу тексту в інтерфейсі на вибрану мову.

Використання AsyncStorage дозволяє запам'ятати вибір користувача навіть після закриття та повторного відкриття додатку. При ініціалізації компонента, він зчитує збережену мову з AsyncStorage та встановлює її як поточну.

Структура модуля виглядає наступним чином (рис. 3.12)

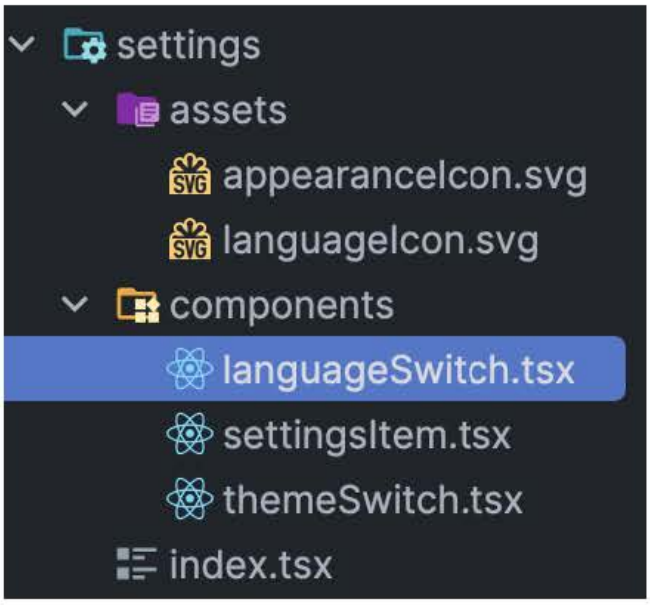


Рисунок 3.12 – Структура модуля налаштувань

Розглянемо модуль бази даних. Оскільки ми використовуємо RealmDB, то робота із нею може відрізнятися. Для початку необхідно описати схеми бази даних згідно спроектованої системи.

Створення схеми в Realm для React Native додатку передбачає визначення структури даних, які будуть зберігатися в базі. Це робиться шляхом створення класів моделей, які успадковуються від Realm.Object [57].

Кожен клас моделі представляє окрему таблицю в базі даних, а властивості класу – стовпці цієї таблиці. Для кожної властивості необхідно вказати її ім'я та тип даних. Realm підтримує різноманітні типи даних, такі як string, int, bool, date, float, double, data, list, linkingObjects та інші.

Після визначення моделей їх необхідно передати до конфігурації Realm при відкритті бази даних.

Приклад схеми наведено на (рис. 3.13).

```ts
import { BSON, Realm } from "realm";

import type { ExerciseItem } from "./exerciseItem.ts";

export class Set extends Realm.Object<Set> {  Show usages  hipetech
  _id!: BSON.ObjectId;
  value!: number;
  units!: string;
  createdAt!: Date;
  lastEdit!: Date;
  exerciseItem!: ExerciseItem;

  static schema: Realm.ObjectSchema = {
    name: "Set",
    primaryKey: "_id",
    properties: {
      _id: "objectId",
      value: "float",
      units: "string",
      createdAt: "date",
      lastEdit: "date",

      exerciseItem: {
        type: "linkingObjects",
        objectType: "ExerciseItem",
        property: "sets",
      },
    },
  };
}
```

Рисунок 3.13 – Приклад схеми Realm DB

Після створення схем необхідно реалізувати бандлінг бази даних. Бандлінг Realm в React Native проекті – це процес вбудовування схеми бази даних у додаток, щоб вона була доступна під час виконання на пристрої користувача. Це необхідно, оскільки під час розробки схема визначається за допомогою JavaScript класів, які недоступні на пристрої.

Бандлінг дозволяє Realm оптимізувати структуру бази даних та запити до неї, що покращує продуктивність додатку. Зазвичай цей процес відбувається автоматично під час збірки проекту, але у деяких випадках може знадобитися ручний бандлінг за допомогою інструментів командного рядка Realm.

Під час бандлінгу схема перетворюється на спеціальний формат, який Realm розуміє (зазвичай JSON-файл). Цей файл потім вбудовується в пакет додатку. Під час запуску додатка Realm зчитує вбудовану схему та

використовує її для створення та керування базою даних на пристрої користувача [58]. Структуру модуля наведено на (рис. 3.14)

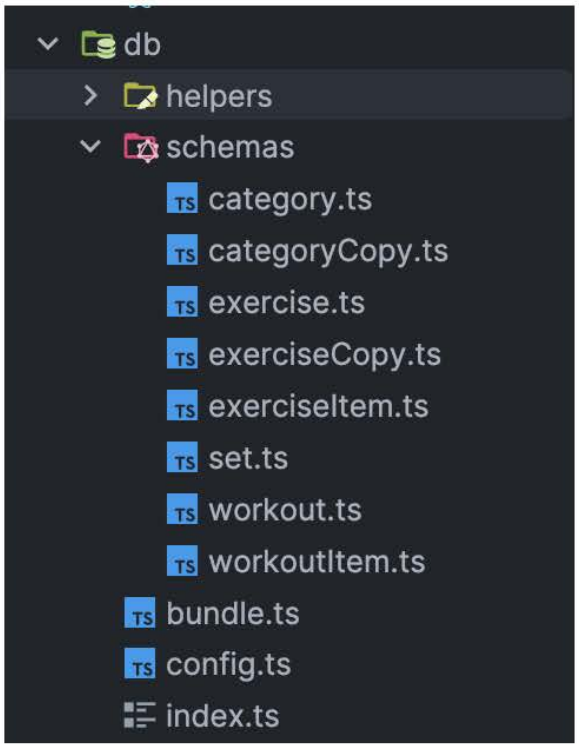


Рисунок 3.14 – Структура модуля бази даних

Основна логіка взаємодії реалізована за допомогою вбудованих у бібліотеку Realm хуків та функцій помічників (helpers), які створюють необхідну абстракцію на операціями з базою даних (створення, видалення, модифікація). Та грають головну роль у реалізації бізнес логіки застосунку.

## 3.3 Тестування та оцінка ефективності додатку

У цьому розділі розглядається процес тестування та оцінки розробленого мобільного фітнес-додатку.

Для того щоб коректно протестувати додаток, потрібно відтворити основний сценарій використання [59].

Для початку відкриємо застосунок, стартовий екран зображено на (рис. 3.15).

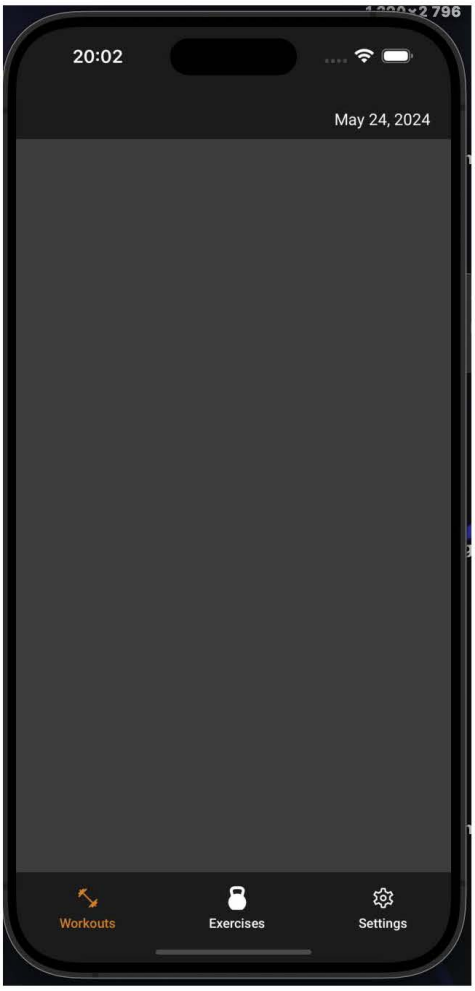


Рисунок 3.15 – Головний екран мобільного застосунку

Ми бачимо що головний екран коректно відображається, спробуємо додати вправу до нашого тренування. Для цього нам потрібно перейти за допомогою навігаційної панель на екран із вправами та натиснути + (рис. 3.16).

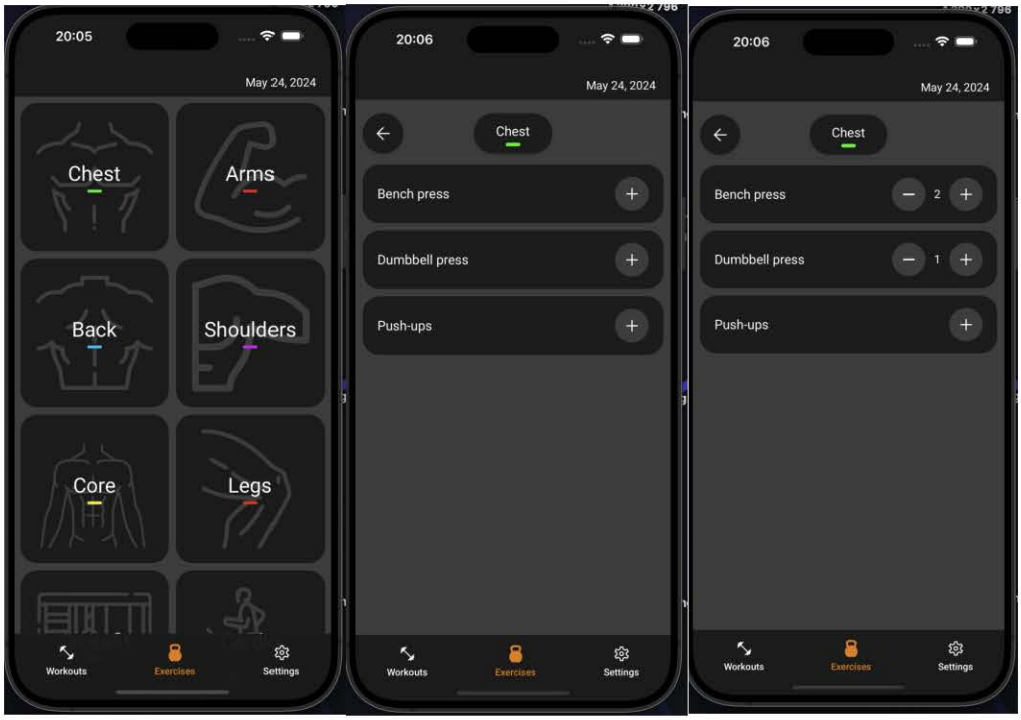


Рисунок 3.16 – Головний екран мобільного застосунку

Вправи коректно додаються та видаляються, на (рис. 3.17), видно як вправи відображаються. Та за допомогою перетягування ми можемо змінити їх порядок.

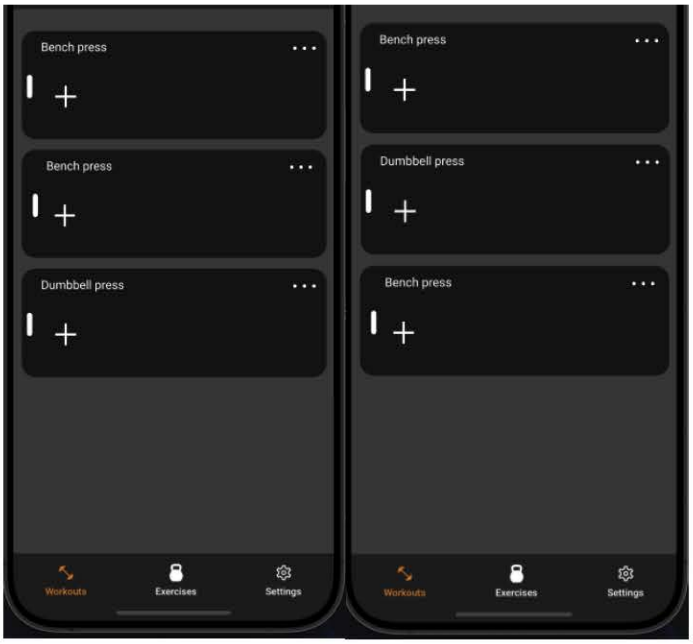


Рисунок 3.17 – Головний екран із доданими вправами

Натиснувши кнопку + на картці вправи, відкриється модальне вікно додавання подходу, де потрібно ввести необхідну вагу та кількість повторів (рис. 3.18).

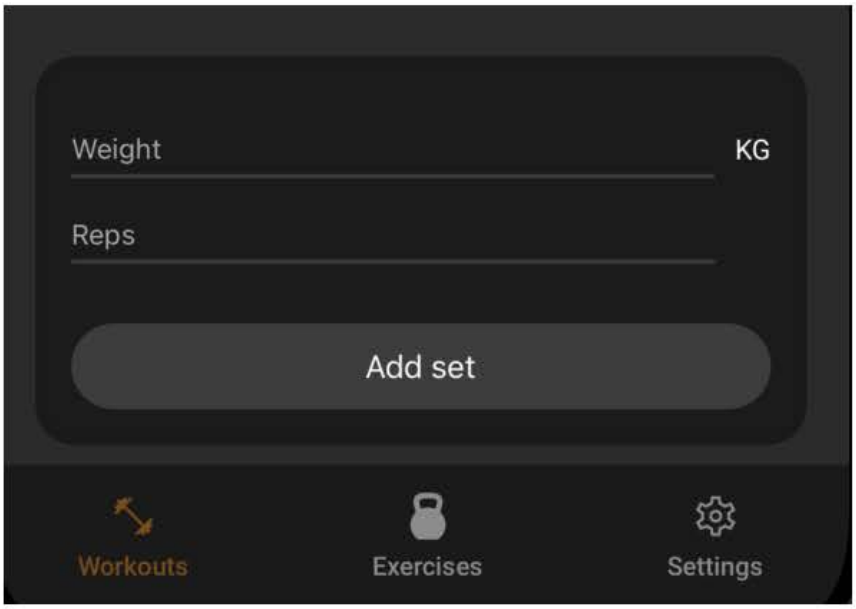


Рисунок 3.18 – Модальне вікно додавання підходу

Після додавання підходу, ми бачимо на головному доданий сет (рис. 3.19)

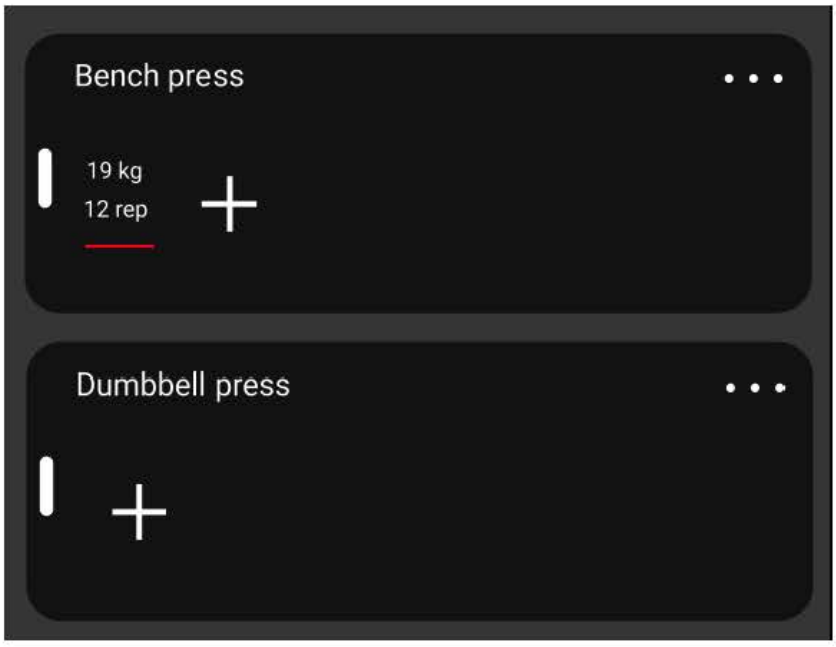


Рисунок 3.19 – Картка вправи з доданим підходом

Тепер протестуємо логіку зміни дати тренування, та зміни місяця, для цього потрібно змахнути вниз верхню панель, натиснути на дату в календарі, змахуючим жестом поміняти місяц (рис. 3.20)


Рисунок 3.20 – Взаємодія із календарем

Протестуємо модуль налаштувань, для цього нам потрібно переключити екран в панелі навігації (рис. 3.21).


Рисунок 3.21 – Екран налаштувань

Змінимо тему оформлення додатку, натиснувши на відповідний пункт у меню налаштувань. Це призведе до відкриття модального вікна, де буде

представлено кілька варіантів тем: системна (світла або темна, залежно від налаштувань пристрою), світла та темна. Користувач може вибрати бажану тему, після чого інтерфейс додатку миттєво зміниться відповідно до обраного стилю. (рис. 3.22).

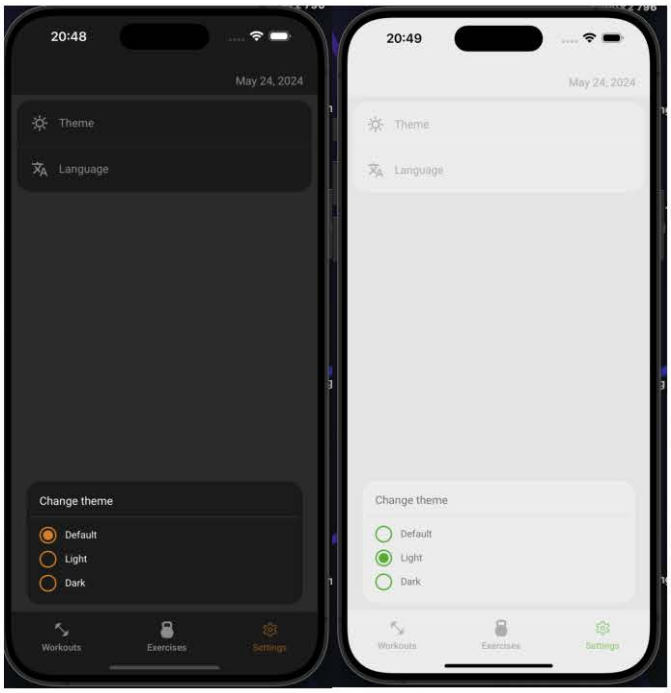


Рисунок 3.22 – Зміна теми застосунку

Змінимо поточну мову інтерфейсу додатку, обравши відповідний пункт у меню налаштувань. При натисканні на цей пункт відкриється модальне вікно, де будуть представлені доступні мови, такі як українська та англійська. Користувач зможе вибрати бажану мову, і після підтвердження вибору весь текст та елементи інтерфейсу додатку автоматично перекладуться на обрану мову, забезпечуючи більш персоналізований та зручний досвід користування (рис. 3.23).

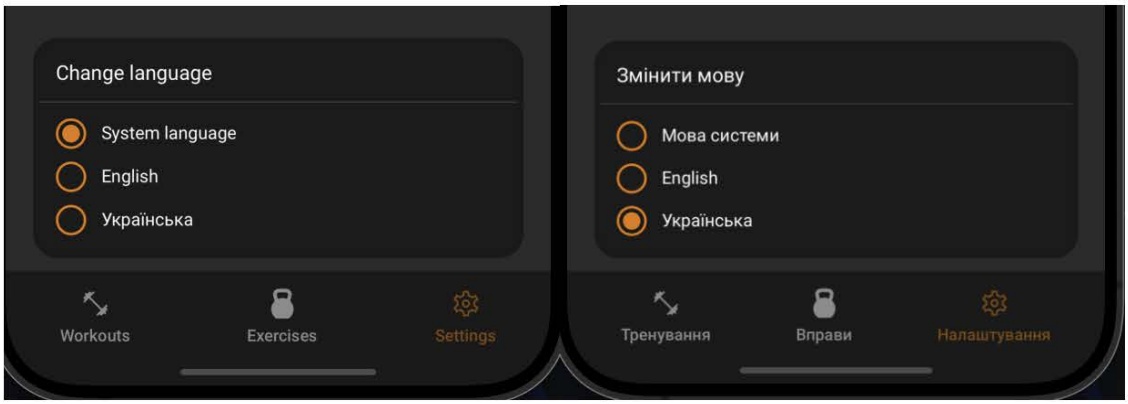


Рисунок 3.23 – Зміна мови застосунку

У цьому розділі було детально описано процес розробки мобільного фітнес-додатку з використанням React Native та суміжних технологій. Було розглянуто ключові аспекти проектування архітектури та структури бази даних, включаючи вибір компонентного підходу, використання глобального стану та патерну Repository для взаємодії з базою даних Realm.

Проведене тестування мобільного фітнес-додатку дозволяє оцінити його функціональність, зручність використання та продуктивність. Результати тестування та оцінки ефективності підтверджують, що розроблений мобільний фітнес-додаток є корисним та зручним інструментом для користувачів, які прагнуть підтримувати здоровий спосіб життя. Врахування отриманих рекомендацій та пропозицій дозволить покращити додаток та зробити його ще більш привабливим для цільової аудиторії.

Під час тестування виявлено декілька варіантів покращення мобільного застосунку:

Розширення функцій: Додавання нових видів тренувань, планів харчування, можливість відстежувати інші показники здоров'я (сон, вага, водний баланс тощо).

Персоналізація: Впровадження алгоритмів, які підбирають тренування та плани харчування індивідуально для кожного користувача, враховуючи його цілі, рівень підготовки та особливості організму.

Соціальні функції: Надати можливість користувачам ділитися своїми досягненнями, змагатися з друзями, приєднуватися до спільнот за інтересами.

Інтеграція з носимими пристроями: Можливість синхронізації даних з фітнес-трекерами та смарт-годинниками.

Геймифікація: Впровадження ігрових елементів, такі як бейджі, нагороди, рівні, щоб зробити тренування більш цікавими та мотивуючими.

Детально описано процес створення основних модулів додатку, таких як навігація, верхня панель з календарем та модуль налаштувань. Особливу увагу приділено використанню анімацій та жестів для створення інтуїтивного та привабливого інтерфейсу користувача.

Таким чином, у цьому розділі було продемонстровано успішну реалізацію мобільного фітнес-додатку з використанням сучасних технологій та підходів до розробки. Отримані результати підтверджують, що додаток є корисним та зручним інструментом для користувачів, які прагнуть вести здоровий спосіб життя, а також має потенціал для подальшого розвитку та вдосконалення.

Створення додатку покращило навички аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки. Проведене тестування допомогло застосувати на практиці інструментальні програмні засоби доменного аналізу, проектування, тестування, візуалізації, вимірювань та документування програмного забезпечення та мотивовано обирати мови програмування та технології розробки для розв’язання завдань створення і супроводження програмного забезпечення.

# ВИСНОВКИ

У ході виконання дипломної роботи розроблено мобільний фітнес-додаток, який надає користувачам можливості для створення та відстеження персоналізованих програм тренувань. Було проведено ґрунтовний аналіз предметної області, включаючи дослідження ринку мобільних фітнес-додатків, проведено огляд технологій та платформ для розробки, а також зроблено аналіз існуючих рішень, таких як Google Fit, Runkeeper та Centr. Кожен з цих додатків має свої переваги та недоліки, що було враховано при розробці власного продукту.

Дослідження ринку показало, що існує великий попит на мобільні фітнес-додатки, зумовлений зростаючим інтересом до здорового способу життя та зручністю використання мобільних технологій. Було виявлено, що користувачі очікують від таких додатків персоналізованого підходу, широкого функціоналу, зручного інтерфейсу та інтеграції з іншими сервісами та пристроями.

На основі отриманих даних було обрано оптимальний стек технологій для розробки фітнес-додатку. TypeScript було обрано як основну мову програмування завдяки його перевагам у забезпеченні статичної типізації та покращенні читабельності коду. React Native було обрано як фреймворк для розробки кросплатформенних мобільних додатків, що дозволяє скоротити час та витрати на розробку, забезпечуючи при цьому нативний вигляд та функціональність додатку. Realm DB було обрано як мобільну базу даних, оптимізовану для роботи з мобільними пристроями, що забезпечує ефективне зберігання та синхронізацію даних. Reanimated було використано для створення плавних та високопродуктивних анімацій, що підвищують привабливість додатку та візуалізують прогрес користувача. Zustand було обрано як бібліотеку для управління станом у React-додатках, що забезпечує централізоване сховище даних та зручні інструменти для оновлення стану.

У процесі розробки було створено архітектуру додатку, що базується на компонентному підході, використанні глобального стану та патерну Repository для взаємодії з базою даних. Це забезпечує модульність, гнучкість та легкість підтримки додатку. Особливу увагу було приділено розробці модуля бази даних з використанням Realm DB. Було визначено схему бази даних, що включає сутності для зберігання інформації про категорії вправ, самі вправи, їх варіації, а також тренування користувачів. Було реалізовано механізм бандлінгу бази даних, що дозволяє вбудувати схему в додаток для оптимізації роботи з даними. Для взаємодії з базою даних було використано вбудовані хуки та функції-помічники Realm, що забезпечує абстракцію над базою даних та спрощує її заміну або модифікацію у майбутньому.

Проведене тестування підтвердило працездатність та ефективність розробленого додатку. Було перевірено коректність роботи основних функцій, таких як додавання та видалення вправ, зміна дати тренування, зміна теми та мови інтерфейсу. Результати тестування показали, що додаток є стабільним, зручним у використанні та відповідає вимогам цільової аудиторії.

У ході виконання дипломної роботи було отримано такі результати:

- Створення повнофункціонального мобільного фітнес-додатку. Додаток надає користувачам можливість створювати та відстежувати персоналізовані програми тренувань, що включають різноманітні вправи та підходи. Він також дозволяє змінювати дату тренування, налаштовувати тему оформлення та мову інтерфейсу.
- Використання сучасних технологій та підходів. У розробці було застосовано актуальні технології та інструменти, такі як TypeScript, React Native, Realm DB, Reanimated та Zustand, що забезпечує високу якість та продуктивність додатку.
- Застосування принципів модульної архітектури. Архітектура додатку побудована на основі модулів, що забезпечує його гнучкість, масштабованість та легкість підтримки.

- Розробка зручного та інтуїтивного інтерфейсу. Інтерфейс користувача розроблено з урахуванням потреб цільової аудиторії, з використанням анімацій та жестів для підвищення зручності та привабливості.
- Успішне тестування та оцінка ефективності. Проведене тестування підтвердило коректність роботи додатку та його відповідність вимогам.

Під час тестування було виявлено потенційні напрямки для подальшого розвитку та вдосконалення додатку. Зокрема, можна розширити функціонал, додавши нові види тренувань, плани харчування та інші показники здоров'я, впровадити алгоритми персоналізації, додати соціальні функції та інтеграцію з носимими пристроями. Також можна розглянути можливість гейміфікації для підвищення мотивації користувачів.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційна система аутентифікації пристроїв інтернету речей. URL: https://ela.kpi.ua/server/api/core/bitstreams/8eb511c4-0977-4349-8363-1d3ed14b6c57/content (дата звернення: 16.05.2024).

2. Swift. URL: https://www.swift.org (дата звернення: 16.05.2024).

3. Kotlin. URL: https://kotlinlang.org (дата звернення: 16.05.2024).

4. JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 16.05.2024).

5. Dart. URL: https://dart.dev (дата звернення: 16.05.2024).

6. React Native. URL: https://reactnative.dev (дата звернення: 16.05.2024).

7. Flutter. URL: https://flutter.dev (дата звернення: 16.05.2024).

8. Xamarin. URL: https://dotnet.microsoft.com/en-us/apps/xamarin (дата звернення: 16.05.2024).

9. Ionic. URL: https://ionicframework.com (дата звернення: 16.05.2024).

10. Retrofit. A type-safe HTTP client for Android and Java. URL: https://square.github.io/retrofit (дата звернення: 16.05.2024).

11. Alamofire. HTTP networking library written in Swift. URL: https://github.com/Alamofire/Alamofire (дата звернення: 16.05.2024).

12. Glide. Media management and image loading framework for Android. URL: https://github.com/bumptech/glide (дата звернення: 16.05.2024).

13. SDWebImage. Async image downloader with cache support. URL: https://github.com/SDWebImage/SDWebImage (дата звернення: 16.05.2024).

14. Firebase. URL: https://firebase.google.com (дата звернення: 16.05.2024).

15. Android Studio. URL: https://www.android.com (дата звернення: 17.05.2024).

16. Xcode 15. URL: https://developer.apple.com/xcode (дата звернення: 17.05.2024).

17. Git. Version control. URL: https://git-scm.com (дата звернення: 17.05.2024).

18. Figma. URL: https://www.figma.com (дата звернення: 17.05.2024).

19. Postman. Http client. URL: https://www.postman.com (дата звернення: 17.05.2024)

20. Jenkins. URL: https://www.jenkins.io (дата звернення: 17.05.2024).

21. Google Fit. URL: https://www.google.com/fit (дата звернення: 18.05.2024).

22. Google Fit: Marketing Strategies for Health Apps: Showcasing Google Fit Features. URL: https://fastercapital.com/content/Google-Fit--Marketing-Strategies-for-Health-Apps--Showcasing-Google-Fit-Features.html (дата звернення: 18.05.2024).

23. Google Fit Platform overview. URL: https://developers.google.com/fit/overview (дата звернення: 19.05.2024).

24. ASICS Runkeeper. URL: https://runkeeper.com/cms (дата звернення: 20.05.2024).

25. Runkeeper Subscription. URL: https://runkeeper.com/go (дата звернення: 20.05.2024).

26. Runkeeper Traffic Analytics and Audience Report. URL: https://www.similarweb.com/website/runkeeper.com/#overview (дата звернення: 21.05.2024).

27. Runkeeper - Tech Stack, Apps, Patents & Trademarks. URL: https://www.crunchbase.com/organization/fitnesskeeper/technology (дата звернення: 21.05.2024).

28. Centr | Fitness App & Wellness Program Inspired by Chris Hemsworth. URL: https://centr.com (дата звернення: 22.05.2024).

29. Centr pricing. URL: https://centr.com/centrpower (дата звернення: 22.05.2024).

30. Centr Traffic Analytics, Ranking & Audience. URL: https://www.similarweb.com/website/centr.com/#overview (дата звернення: 22.05.2024).

31. Centr - Tech Stack, Apps, Patents & Trademarks. URL: https://www.crunchbase.com/organization/altus-fire-and-life-safety/technology (дата звернення: 22.05.2024).

32. Android 14. URL: https://www.android.com/android-14. (дата звернення: 22.05.2024).

33. Android Architect. URL: https://developer.android.com/topic/architecture (дата звернення: 22.05.2024).

34. Kotlin and Android | Android Developers https://developer.android.com/kotlin (дата звернення: 23.05.2024).

35. IOS 17. URL: https://www.apple.com/ios (дата звернення: 22.05.2024).

36. iOS Architecture. URL: https://intellipaat.com/blog/tutorial/ios-tutorial/ios-architecture (дата звернення: 23.05.2024).

37. App Dev Tutorials. URL: https://developer.apple.com/tutorials/app-dev-training/ (дата звернення: 23.05.2024).

38. Communication between native and React Native. URL: https://reactnative.dev/docs/communication-ios (дата звернення: 23.05.2024).

39. Skia: The 2D Graphics Library. URL: https://skia.org (дата звернення: 23.05.2024).

40. MVP (Model View Presenter) Architecture Pattern. URL: https://www.geeksforgeeks.org/mvp-model-view-presenter-architecture-pattern-in-android-with-example/ (дата звернення: 23.05.2024).

41. What Is MVVM Architecture? URL: https://builtin.com/software-engineering-perspectives/mvvm-architecture (дата звернення: 23.05.2024).

42. Clean Architecture for mobile. URL: https://proandroiddev.com/clean-architecture-for-mobile-to-be-or-not-to-be-2ffc8d46e402 (дата звернення: 23.05.2024).

43. Typescript. URL: https://www.typescriptlang.org (дата звернення: 23.05.2024).

44. Realm DB. URL: https://www.mongodb.com/developer/products/realm/ (дата звернення: 24.05.2024).

45. React Native Reanimated. URL: https://docs.swmansion.com/react-native-reanimated/ (дата звернення: 24.05.2024).

46. Zustand. URL: https://github.com/pmndrs/zustand (дата звернення: 24.05.2024).

47. WebStorm The JavaScript and TypeScript IDE. URL: https://www.jetbrains.com/webstorm/ (дата звернення: 24.05.2024).

48. Realm Studio. URL: https://www.mongodb.com/docs/atlas/device-sdks/studio/ (дата звернення: 24.05.2024).

49. Github. URL: https://github.com/about (дата звернення: 24.05.2024).

50. Github Copilot. The world's most widely adopted AI developer tool. URL: https://github.com/features/copilot (дата звернення: 24.05.2024).

51. What is Component-Based Architecture? URL: https://www.mendix.com/blog/what-is-component-based-architecture (дата звернення: 24.05.2024).

52. Understanding and Properly Using React Global State. URL: https://clerk.com/blog/understanding-and-properly-using-react-global-state (дата звернення: 24.05.2024).

53. What is Repository Pattern? URL: https://www.linkedin.com/pulse/what-repository-pattern-alper-sara (дата звернення: 24.05.2024).

54. Introducing Create React Native App. URL: https://reactnative.dev/blog/2017/03/13/introducing-create-react-native-app (дата звернення: 24.05.2024).

55. Getting started | React Navigation. URL: https://reactnavigation.org/docs/getting-started (дата звернення: 24.05.2024).

56. AsyncStorage. URL: https://reactnative.dev/docs/asyncstorage (дата звернення: 24.05.2024).

57. Define a Realm Object Model. URL: https://www.mongodb.com/docs/atlas/device-sdks/sdk/react-native/model-data/define-a-realm-object-model/ (дата звернення: 24.05.2024).

58. Bundle a Realm File. URL: https://www.mongodb.com/docs/atlas/device-sdks/sdk/react-native/realm-files/bundle/ (дата звернення: 24.05.2024).

59. Андрєєв Р. Т. Дослідження технологій створення кросплатформенних мобільних застосунків. Запоріжжя, 2020. 49с. URL: https://dspace.znu.edu.ua/xmlui/bitstream/handle/12345/3786/Андрєєв.pdf (дата звернення: 25.05.2024)