ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра комп'ютерних наук

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «Розробка кросс-платформного додатку типу планер-трекер на основі фреймворку Flutter»

Виконав: студент 4 курсу, групи КНД–42
спеціальності 122 Комп'ютерні науки
Дубовицький Д.С.
(прізвище та ініціали)

Керівник Іщеряков С.М.
(прізвище та ініціали)

Рецензент_____
(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій
Кафедра Комп'ютерних наук
Ступінь вищої освіти «Бакалавр»
Спеціальність 122 Комп‘ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп’ютерних наук

В.В. Вишнівський
“ ___ ” _____ 2021 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Дубовицький Денис Сергійович
(прізвище, ім‘я, по батькові)

1.Тема роботи: «Розробка кросс-платформного додатку типу планер-трекер на основі фреймворку Flutter»
Керівник роботи кандидат технічних наук, доцент Іщеряков С.М.

(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від 12.03.2021 року №65.

2. Строк подання студентом роботи 30.05.2021р.

3. Вхідні дані до роботи:

Науково-технічна література з питань, що пов‘язані з алгоритмами штучного інтелекту

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

4.1 Аналіз кросс-платформних технологій
4.2 Аналіз аналогічних додатків
4.3 Розробка дизайну
4.4. Розробка кросс-платформного додатку на основі Flutter

5. Перелік графічного матеріалу

1. Тема
2. Мета, об’єкт і предмет дослідження
3. Актуальність дослідження
4. Технологія кросс-платформної розробки
5. Огляд існуючих технологій
6. Технологія Flutter
7. Аналіз аналогічних додатків
8. Розробка дизайну
9. Аналіз архітектур Flutter
10. Розробка додатку
11. Презентація
12. Висновки

7. Дата видачі завдання 10.02.2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз кросс-платформної технології	10.03 – 12.03	
2	Аналіз технології Flutter	13.03 – 25.02	
3	Аналіз аналогічних додатків	25.03 – 26.03	
4	Розробка дизайну додатку	26.03 – 01.04	
5	Аналіз архітектур	01.04 – 07.04	
6	Розробка додатку	07.04 – 07.05	
7	Вступ, висновки, реферат	30.05 – 07.05	
8	Основні розділи	07.05 – 15.05	
9	Попередній захист роботи	25.05	

Студент _____ Дубовицький Д.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Іщеряков С.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 60 с., 24 рис., 1 табл., 4 дод., 18 джерел.
КРОСПЛАТФОРМНИЙ, FLUTTER, ANDROID, IOS, FRAMEWORK, МОБІЛЬНИЙ, АРХІТЕКТУРА, MOBX, ІНТЕРФЕЙС, КЕРУВАННЯ СТАНОМ ПРОГРАМИ

Об`єкт дослідження – технології розробки кросс–платформних мобільних додатків

Предмет дослідження – архітектура та основні елементи крос-платформних технологій, проэктування кросс–платфорного мобільного додатку на основі фреймворка “Flutter”

Мета роботи – Розробка кросс–платформного додатку типу планер–трекер

Методи дослідження – аналіз та порівняння крос-платформних архітектур, системи керування станом додатку, реалізація інтерфейсу у технології Flutter

Визначено найбільш єфективна технологія Fluter.

Здійснено та проведено аналіз сучасних мобільних операційних систем, проведений аналіз способів створення додатків для зазначених систем.

Розглянуто особливості роботи кросс-платформних рішень

На основі результатів виконаних досліджень розроблено кросс-платформний додаток типу планер-трекера.

Упровадження розробленого додатку дозволяє більш ефектино здійснювати спортивні тренування як початковим культуристам, так і професійним спортсменам.

ЗМІСТ

ВСТУП

Ні для кого не є секретом, що зараз наше життя оточене різноманітними гаджетами – ноутбуки, годинники, смартфони тощо. У повсякденному житті ми міксуємо їх, використовуючи ті чи інші технічні прилади при виконанні найбільш відповідних до них завдань. Оскільки кожен прилад працює на своїй платформі(варіанти), то існує необхідність підлаштування вже існуючих додатків під використання на декількох платформах та створення нових кросс–платформних додатків. Тема моєї дипломної роботи є досить актуальною для нашого життя, оскільки з кожним роком ми спостерігаємо на все більший розвиток технологій, і потребність в кросс–платформних додатків буде тільки зростати.

Проте не зважаючі на технологічний прогресс, людство також дуже зацікавлене такою стороною свого життя, як спорт. Спортивними змаганнями цікавляться з малих літ до самої старості, сім'ї збираються разом, щоб подивитися матч з футболу, по вулицях і в парках ми бачимо людей на ранкових і вечірніх пробіжках, на кожному кроці стоять спортзали, фітнес клуби та басейни, щорічно десятки компаній витрачають шалені гроші на модернізацію та розробку нових тренажерів. Оскільки не кожна людина може собі дозволити займатися з тренером, для поліпшення результату багато людей ведуть щоденники, у яких відмічають, які саме вправи вони робили, на яку групу м'язів, скільки і яких саме вправ вони робитимуть завтра. Тож додатки типу плнер–трекеров завжди будуть залішатися актуальними, не лише тому, що стосуються нашої невід'ємної сфери життя, а й тому, що круг споживачів осягає всі покоління та любу стать.

Тому обираючи тип та тематику для мого диплому, я обрав саме розробку кросс–платформного додатку, оскільки як я казав раніше, вважаю, що з розвитком технологій необхідність кросс–платформних додатків буде тільки зростати, та обираючи тематику зупинився на планер–трекеру, тому що спортивна сторона життя

людини та необхідність її планування, аналізу та корегування буде актуальна завжди. Підриємства які націлені на створення мобільних додатків для широкого кола користувачів, питання як швидкість розробки додатку та його вартість є ключовими для ведення бізнесу. Також, не менш важливими є швидкість роботи самого додатка та єфективність використовування пам’яті мобільного телефону.

Таким чином об’єктом дослідження є технології розробки кросс–платформних мобільних додатків

Предметом дослідження є архітектура та основні елементи крос-платформних технологій, проэктування кросс–платформного мобільного додатку на основі фреймворка “Flutter”

Метою роботи є створення крос-платформного додатку на основі прооведеного аналізу.

Завданнями дослідження є:

- аналіз та порівняння сучасних фреймврків створення кросс-платформних додатків;
- розробка дизайну на основі аналізу аналогічних рішень;
- аналіз існуючі рішення архітектур мобільного додатку;
- розробка додатку на основі фреймворка Flutter та архітектури MobX.

1 ІСНУЮЧІ ТЕХНОЛОГІЇ КРОСС–ПЛАТФОРМНОЇ РОЗРОБКИ

1.1 Актуальність технології розробки кросс–платформних додатків

У сучасному світі, кількість мобільних додатків зростає кожну годину, тож швидкість потрапляння продукту на ринок становить великий відсоток від подальшого успіху проекту, тому чим раніше створюється додаток, тим більша вірогідність, що конкурентноздатні проекти привернуть увагу потенційних споживачів та СМІ.Також разом з кількістю додатків зростає їх складність, а отже з нею врази зростає і вартість розробку додатку.

Щоб охопити найбільшу кількість споживачів задля беззаперечного успіху, розрбники орієнтують свої додатки під роботу на двух домінуючих платформах: Android та iOS – основні мобільні платформи, які охоплюють 90% відсотків ринку. Ці платформи мають різні корені у своєму походженні, різні мови програмування та різний підхід у розробці додатків.

iOS – пропрієтарна системою компанії Apple, розробка додатків ведеться на мовах програмування Swift або Objective–C виключно за допомогою пропрієтарної середи розробки xCode та інструментів розробника, які доступні тільки на приладах Apple [13]. Встановлюється iOS тільки на пристрої компанії Apple, додатки для системи встановлюються виключно з магазина додатків App Store [14].

Android – платформа з відкритим кодом, розроблена Open Handset Alliance (OHA) під керівництвом Google [15] . Розробка додатків ведеться за допомогою мов програмування Java[17] або Kotlin[16], інструменти розробника знаходяться у відкритому доступі і працюють на MacOS, Linux та Windows[18]. На сьогоднішній день Anroid працює на телефонах, планшетах, відеорегістраторах, телевізорах,

годинниках і є вільною операційною системою. Додатки можуть бути встановлені із пам’яті телефону, карти пам’яті, або сторонніх магазинів додатків.

Обидві платформи мають різний підхід щодо компіляції додатків. Додатки операційної системи Андроїд, які написані на мовах програмування Java або Kotlin компілюються у проміжний байт–код, який вже потім компілюється кінцевим пристроєм у машинний код придатний до виконання під час встановлення додатку [1].

Додатки написані для операційної системи iOS заздалегідь компілюються у машинний код процесора пристроїв, на яких планується виконання і розповсюджується у вигляді вже скомпільованого додатку.

Беручи до уваги цю інформацію, робимо висновок, що платформи кардинально відрізняються як за філософією, так і способами розповсюдження, засобами компіляції, мовами програмування, які використовуються для створення додатків для цих систем.

Для розробки додатку, який працював би на платформах iOS та Android, при нативному підході до розробки мобільних додатків, дві комманди розробників із кваліфікаціями відповідно до платформи створюють два додатки та дві кодові бази.

Тож для більш ефективного використання людських та фінансових ресурсів була створена кросс–платформна технологія, яка дозволяє зменшити кошти та час на розробку та тестування двух додатків.

При кросс–платформному підході до створення додатків пишется лише одна кодова база для обох платформ та потрібня лише одна комманда розробників, також зменьшується час на тестування, оскільки на обох платформах працює один додаток, який потрібно протестувати лише один раз.

На данний момент існує багато кросс–платформних рішень, найбільш популярні та перспективні серед них, які займають більше 80%ринку кросс–платформних додатків – React Native, Flutter, Ionic (рис.–1).

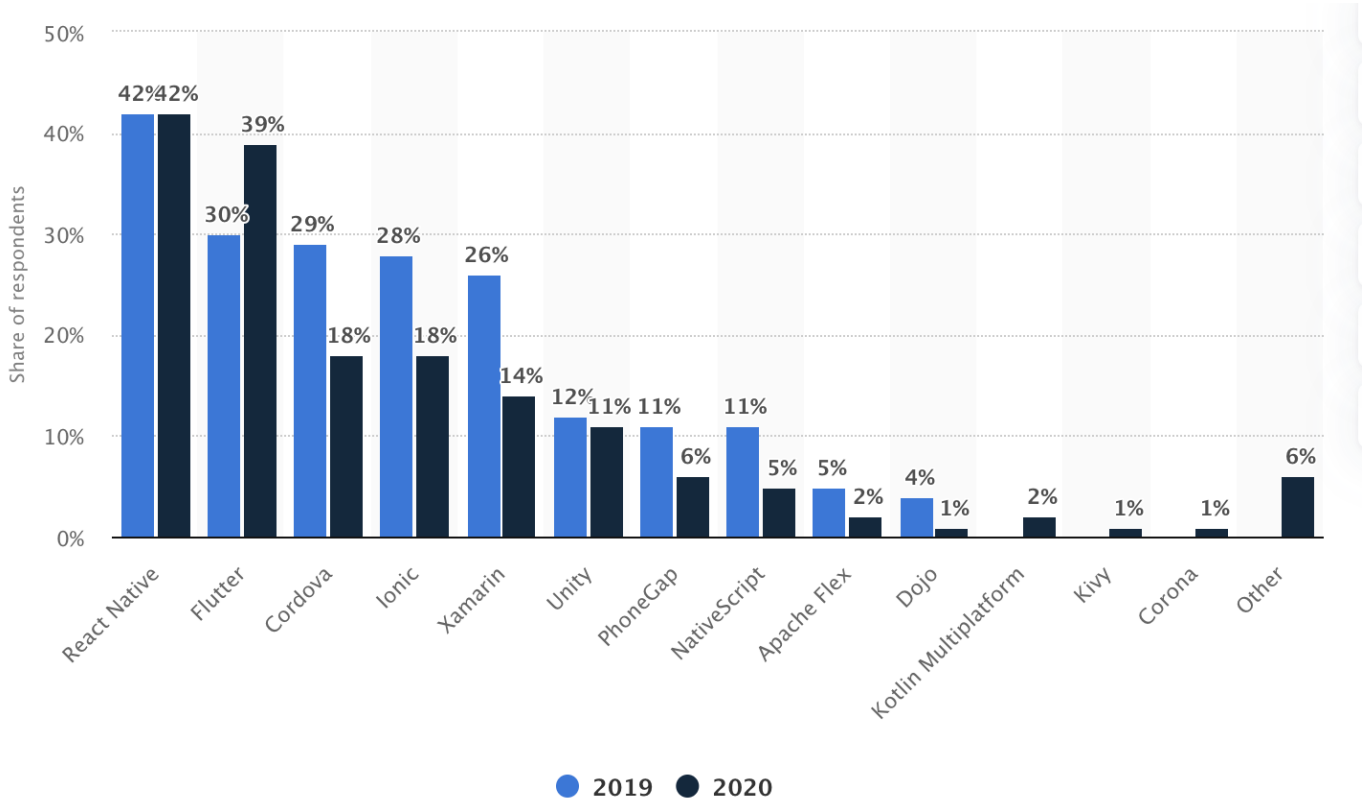


Рисунок 1.1 – Відсоток розробників, які використовували фреймворк

1.2 Технологія React–native

React Native – кроссплатформна технологія з відкритим кодом, представлена компанією фейсбук у 2015 році [2].

Розробка додатків за допомогою технології відбувається на мові програмування JavaScript.

Основними компонентами архітектури є (рис. 1.2):

– Нативні модулі – набір функцій, написаних для Android та iOS окремо для на рідних для платформи мовах програмування – Java для Android та Swift / Objective–C для iOS. Ці функції є доступними для виконання через React–Native Bridge, і таким чином реалізують доступ до можливостей платформи.

- React Native Bridge – міст, написаний на C ++ або Java, який відповідає за зв’язок між Нативним потоком виконання та JavaScript потоком. Використовує пропрієтарний протокол для коммунікації з Нативним та JavaScript потоками.
- Javascript VM – Віртуальна JavaScript машина, яка виконує JavaScript код додатку. Для виконання JavaScript використовується JavaScriptCore – рушій JavaScript з відкритим сирцевим кодом. В iOS JavaScriptCore надається системою iOS, для Android версії додатку JavaScriptCore поставляється у вигляді програмної бібліотеки.

Основними перевагами React Native є:

- мова програмування JavaScript, найпопулярніша на даний момент мова програмування;
- можливість використання графічних елементів системи, на якій працює додаток за допомогою React Native Bridge;
- великий набір вже готових модулей до використання.

Недоліками є:

- використовує мову програмування з динамічною типізацією, робота з кодовую базою такого типу є важчою, ніж робота з типізованими мовами програмування;
- великий розмір додатків. Разом з додатком постачається сама технологія React Native, та у випадку Android додатків також постачається JavaScriptCore;
- низька швидкість роботи. Комманди додатку спочатку інтерпритуються Java Script рушієм, після чого RN Bridge повідомляє нативні модулі про те, що треба оновити інтерфейс користувача, що призводить до нижчої продуктивності роботи додатку[3].

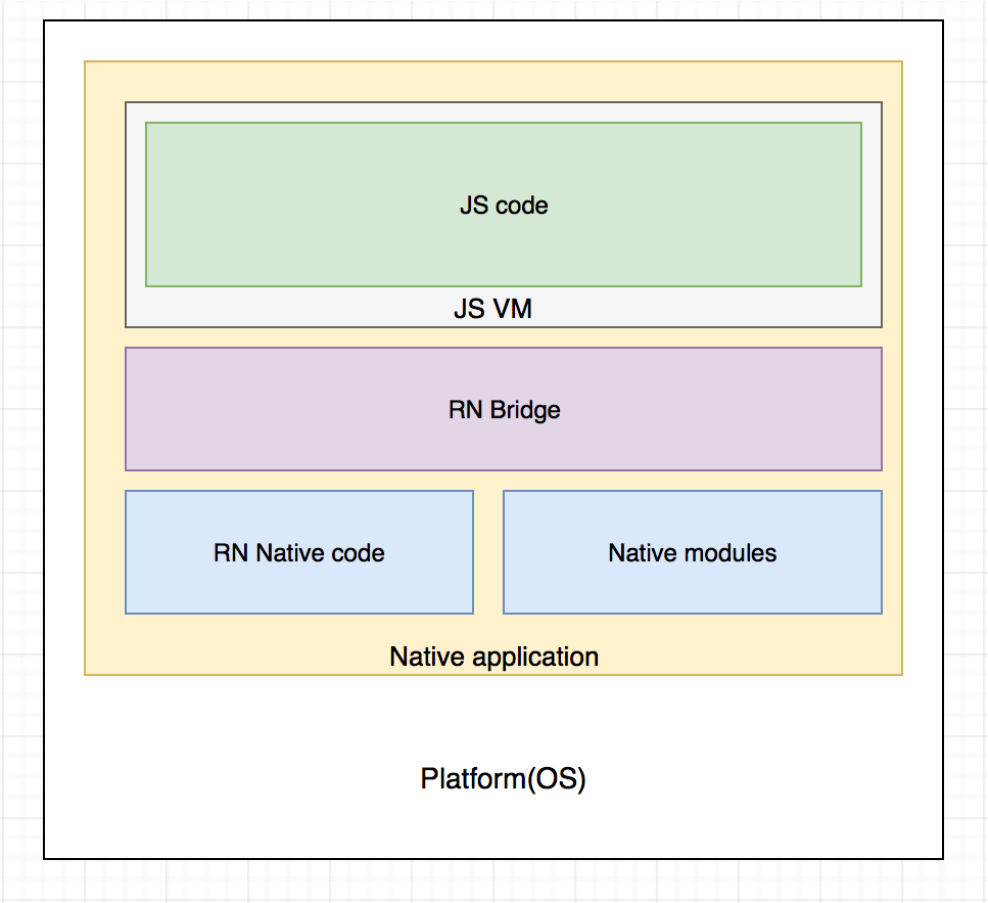


Рисунок 1.2 – Архітектура технології React Native

1.3 Технологія Cordova

Cordova бере свій початок у 2008 році, коли на події iPhoneDevCamp декілька інженеріві з компанії Nitobi експериментували з можливостями елементу інтерфейсу WebView виступати у якості оболонки для запуску web–додатків у середовищі мобільного додатку, та пакуванням веб-сторінок разом з додатком. Єксперимент вийшов успішним і його результати використали для створення фреймворку PhoneGap.

У 2011 році компанія Adobe купила Nitobi і PhoneGap було передано у власність до Apache Foundation, де проєкт був перейменований у Apache Cordova. На основі фреймворка Cordova працює Ionic.

Для розробки додатків на фреймворку Cordova використовуються звичайні методи розробки веб сторінок: HTML, CSS, JavaScript.

Архітектура Cordova складається з (рис. 1.3):

– Web App визначається як основна частина, де знаходиться код програми. Це просто веб–сторінка, яка створюється за допомогою HTML, CSS та JavaScript. За замовчуванням локальний файл, тобто index.html, використовується для посилання на CSS, JavaScript, медіа–файли та інші ресурси, необхідні для запуску програми. Додаток в основному виконується в WebView в межах мобільного додатку–оболонки, який розповсюджується в магазинах програм.
– Web View, що забезпечує відображення інтерфейс користувача програми Cordova, а також може бути компонентом для деяких платформ у більших гібридних додатках. Ці програми поєднують WebView із власними компонентами програми.

– Plugins, що визначаються як невід’ємна частина екосистеми Кордови, яка забезпечує інтерфейс для Кордови та власних компонентів для взаємодії між собою. Він також надає інтерфейс для прив'язки стандартних API пристроїв. Це дозволяє вам викликати власний код з JavaScript.

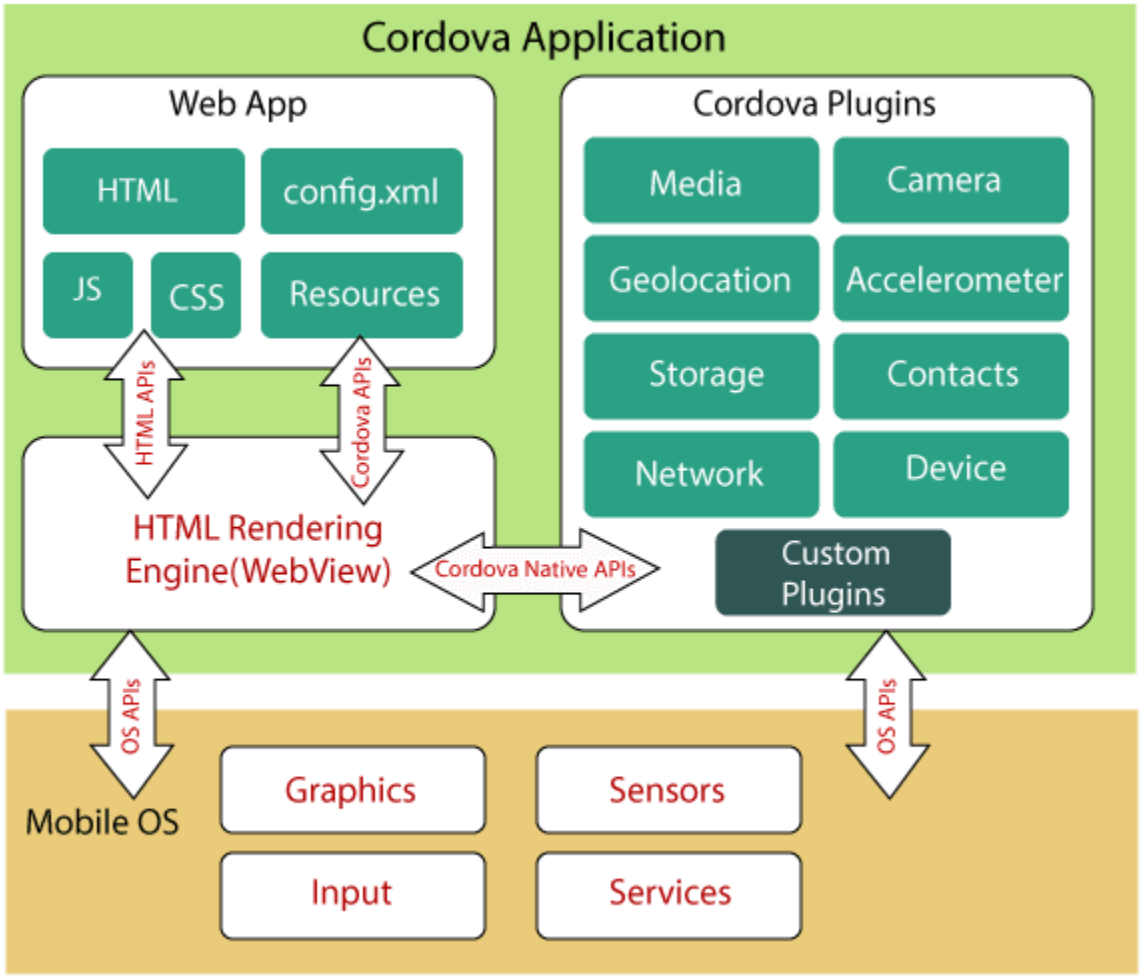


Рисунок 1.3 – Архітектура технології Cordova

В основному проект Apache Cordova визначає набір основних плагінів, які називаються Core Plugins. Ці плагіни забезпечують доступ до можливостей пристрою в додатку, таких як камера, акумулятор, контакти тощо.

Використовуючи ці плагіни, розробники можуть легко пов’язати код JavaScript із власним кодом, який працює у серверній системі. На додаток до цих основних плагінів доступні кілька сторонніх плагінів, які надають додаткові прив'язки до функцій, які не обов'язково доступні на всіх платформах.

При запуску програми, по–перше, Apache Cordova завантажує сторінку за замовчуванням (як правило, index.html) у WebView. WebView дозволяє

користувачеві взаємодіяти з програмою, вводити дані в поля введення, натискати кнопки та переглядати результати в WebView.

Для доступу до власних функцій мобільного, таких як аудіо чи камера, Cordova пропонує пакет API–інтерфейсів JavaScript, які розробники можуть використовувати з їх коду JavaScript. Виклики API–інтерфейсів Cordova JavaScript перетворюються на виклики API власного пристрою за допомогою спеціального моста. З API мобільного пристрою можна взаємодіяти за допомогою плагінів Apache Cordova [4].

Переваги використання технології Cordova:

- Швидка розробка, яка відбувається за допомгою звичайних web інструментів – HTML, CSS, JavaScript;
- Можливість конвертувати вже готові веб сторінки в cordova додатки.

Недоліки використання технології Cordova:

- Повільна робота інтерфейсу додатку;
- Немає доступу до елементів системи;
- Необхідно писати плагіни для доступу до функцій системи.

1.4 Технологія Flutter

Flutter – це фреймворк з відкритим кодом, створений Google. Він використовується для розробки кросплатформенних додатків для Android, iOS, Linux, Mac, Windows, Google Fuchsia та web сторінок використовуючи єдину кодову базу.

Інтерфейси користувача створені за допомогою Flutter будуються, викладаються, компонуються та фарбуються самим Flutter. Механізм отримання текстури та участі у життєвому циклі програми базової операційної системи неминуче змінюється залежно від особливостей цієї платформи. Flutter engine є платформно–агностичним, представляючи стабільний ABI (Application Binary Interface), який забезпечує platform embedder[12] способом налаштування та використання Flutter. Таким чином, Flutter може бути пристосований для

використання на будь якій платформі, для цього необхідно лише створити platform–embedder та імплементувати ABI рушія Flutter.

Вперше представлений у Квітні 2017 році [5], а перша стабільна версія вийшла 4 грудня 2018 року [6].

Розробка ведеться на мові програмування Dart, а архітектура фреймворку складається з (рис. 1.4):

- Framework, написаний на мові програмування Dart, забезпечує сучасну реактивну архітектуру. Він складається з набіу бібліотек платформи, макетування та основоположних бібліотек. Фреймворк відповідає за надання дерева віджетів, обчислення змін у дереві віджеті, збереження стану додатку, роботу з мережею, розпізнавання жестів та хореографію анімації;
- Engine написаний на C ++ і відповідає відповідає за малювання та растеризацію інтерфейсу (через Skia), розміщення тексту, файлові та мережеві вводи–виводи, підтримку спеціальних можливостей (Accessibility), архітектуру плагінів, а також середовище виконання та компіляції Dart;
- Embedder, власний додаток для ОС, який розміщує весь вміст Flutter і виконує роль клею між головною операційною системою та Flutter. Коли ви запускаєте програму Flutter, вбудовувач забезпечує точку входу, ініціалізує механізм Flutter, отримує потоки для інтерфейсу та растрування та створює текстуру, до якої Flutter може писати. Embedder також відповідає за життєвий цикл програми, включаючи жести введення (такі як миша, клавіатура, дотик), розмір вікна, управління потоками та повідомлення платформи.

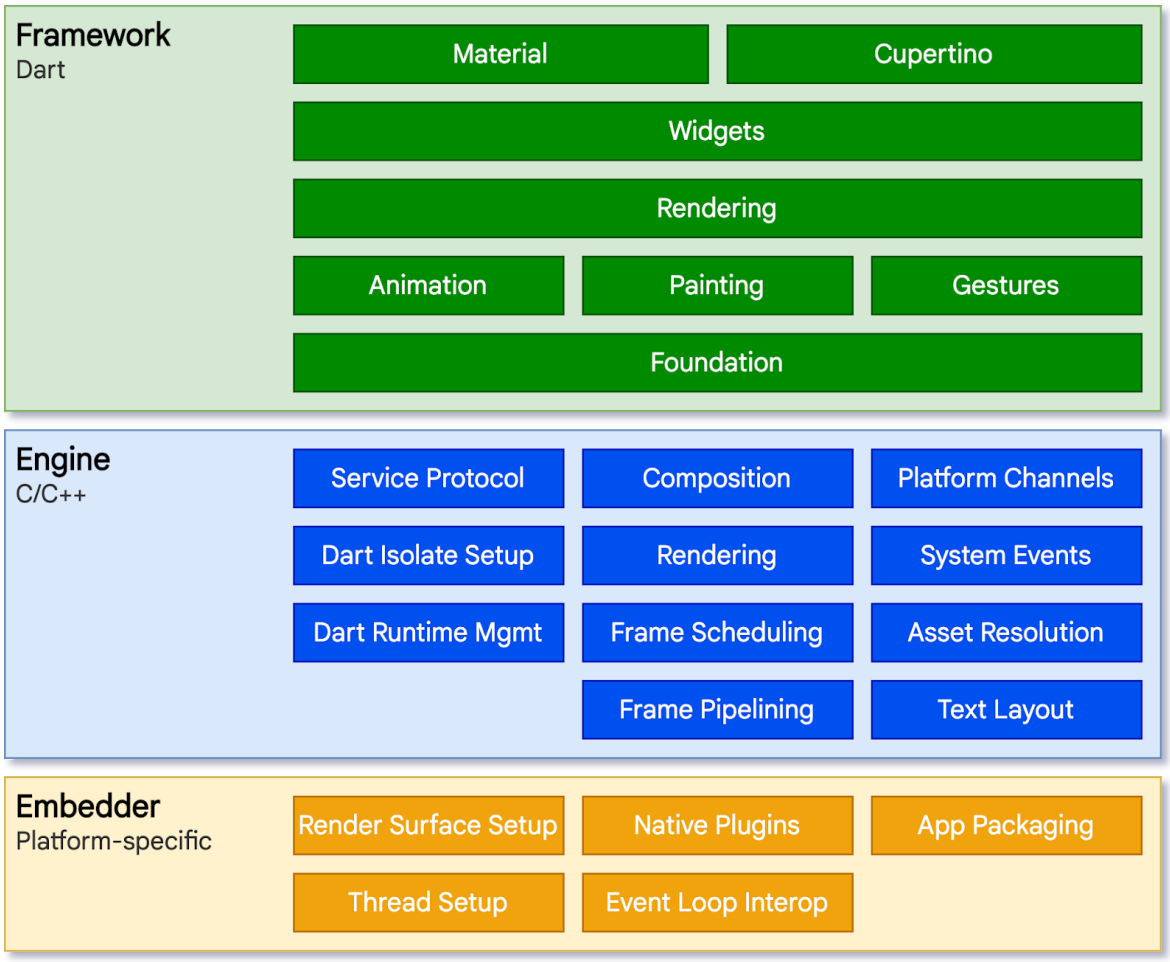


Рисунок 1.4 – Архітектура Flutter

Перевагами Flutter є:

– велика швидкість роботи;
– інтерфейс незалежний від системи;
– типізована мова програмування;
– велика кількість платформ.

Недоліками є:

– непопулярна мова Dart;
– неможливість використання графічних елементів системи;
– більший розмір додатку ніж нативний додаток.

1.5 Обгрунтування вибору фреймворку Flutter

Оскільки Flutter використовує мову програмування Dart, на відміну від Javascript, що використовується у React Native та Cordova, PhoneGap, Ionic, можливість масштабування додатків на Flutter значно збільшується через використання статично типізованої мови.

Використання власного двигуну Skia написанного на C/C++ для макетування, малювання, композиції та растеризації робить додатки Flutter таким же швидким як і додатки, написані на рідних для платформи мовах. Можливість реалізації будь–яких дизайнів за рахунок того, що Flutter використовує власні віджети замість віджетів, предоставленных системою.

За рахунок універсальності двигуна Flutter, ця технологія підтримує створення додатків для Android, iOS, Windows, MacOS, Linux та Web.

Таблиця 1.1 – порівняння фреймворків

Фреймворк	Мова Програмування	Технологія створення Інтерфейсу	Підримуємі платформи
React-Native	JavaScript	Елементи системи через React Bridge	Android, iOS
Cordova, PhoneGap	JavaScript	Використання WebView	Android, iOS
Flutter	Dart	Двигун Skia	Android, iOS, Windows, MacOS, Linux, Web

Через велику швидкість роботи, статично типізований язик програмування, можливість створення дизайну незалежно від системи на якій працює додаток та велику кількість підтримуємих платформ, Flutter було обрано для розробки кросс–платформного додатку.

Оскільки термін розробки аналогічного додатка становить 3 місяці, при середній зарплаті iOS-фахівця в 64000 в гривень на місяць, і середній зарплаті Android-фахівця 65000 гривень, загальна вартість розробки становитиме (65000 + 64000) * 3 = 387000 гривень.

При використанні крос-платформного рішення, нам знадобиться всього один розробник. З огляду на середню зарплату крос-платформного розробника в 67000 гривень, розробка цієї програми загалом буде коштувати 67000 * 3 = 201000 гривень, що на 0.48% дешевше ніж вартість розробки аналогічного додатку без використання крос-платформних технологій.

2 РОБОТА З АНАЛОГАМИ ТА ЇХ АНАЛІЗ

2.1 Аналіз ринку аналогічних додатків

Задля кращого розуміння продукту та послідовного та логічного проектування необхідно ознайомитись с аналогами, які являються кросс-платформними додатками та відповідають тематиці, обраної мною для розробки додатку дипломної роботи.

Оскільки із-за зручності кросс-платформні додатки є досить актуальними, а тема спорта була и буде завжди важливою складовою нашого життя, з пошуком аналогів не було проблем.

Я розглядав такі аналогічні додатки, як Fitbod, Futness point, iFit, GymApp, Strong, Тренування вдома, Nike Training Club, HitFit, BetterMe, Jefit, Workout.

Ознайомлюючись з цими додатками, я виокремив три додатки, які являються кросс-платформними додатками, відповідають моїй тематиці та найбільш схожі на додаток, який я хочу створити для дипломної роботи, а саме:

- Strong;
- Jefit;
- Workout;

Далі хочу більш детальніше їх розглянути, виокремити їх переваги та недоліки з моєї точки зору, провести глибокий аналіз кожного додатку та розглянути декілька їх екранів.

2.2 Додаток Strong

2.2.1. Огляд та аналіз додатку

Додаток з'явився на ринку Google Play у 2017 році та доданий до App Store. На даний момент продукт має більш ніж 1 млн установок в Google Play, що безпосередньо вказує на його необхідність серед споживачів. Оскільки додаток Strong був розроблений для завчасного планування тренувань, відстежування прогресу за рахунок передових функцій та наочного ознайомлення зі статистикой тренувань через графіки, а отже, і прогрессом занять, основною частиною споживачів виступають саме досвідчені культуристи, тренери, та просто спортивні люди, які хочуть не тільки покращити свої результати, але й відстежувати прогресс від систематичних занять.

Задля більшої зручності споживачів та для більш сильної фінансової віддачі від додатку, розробники також пропонують платну версію продукту, у якій споживачі можуть ознайомитись з більшим набором статистики та додати більше планових тренувань. Це безпосередньо також сприяє більшому відгуку від цільової аудиторії у вигляді постійного примножування нових споживачів.

Ознайомившись та проаналізувавши відгуки користувачів, приходжу до висновку, що більшість з них вельми позитивні, і користувачі при користуванням додатком більше декількох років зазнають, що це один із найкращих помічників під час їх щоденних тренувань. Проте, деякі споживачі визнають і недоліки додатку, такі як надлишок спливаючих вікон та трохи громазкий інтерфейс. Також деякі визначають, що ціна за додаток трохи завищена, проте, це вже вибір і точка зору кожної окремої людини.

Шукаючи та обираючи, які саме додатки розглядати за аналоги та аналізувати структуру, інтерфейс, веб-дизайн, недоліки та плюси, окупованість та популярність на ринку, одним із додатків був обраний саме Strong, оскільки в першу чергу він має ту ж саму цільову аудиторію та тематику, що й додаток, який був розроблений мною

для диплому. Також я вибрав для аналізу саме додаток Strong, оскільки вцілому мені сподобався лаконічний інтерфейс та велике різноманіття графіків.

Далі хочу більш детально розглянути додаток та його сильні та слабкі сторони, щоб на основі проведеного глибоко аналізу конкурента створити більш актуальний та конкурентноздатний додаток.

Сильні сторони додатку, які я виокремив після детального аналізу додатку:

- крім основного вигляду продукту, яким споживачі активно користуються на смартфонах, розробка має комплементарний додаток для ЕПЛ ВОТЧ, що полегшує використання у спортивних залах, фітнес центрах, на вулиці тощо, що в свою чергу сприяє більшої популяризації додатку серед користувачів;
- оскільки однією із основною функцією додатку є збір інформації та створення на її основі статистики тренувань, непереборним плюсом є надзвичайна кількість графіків, на відміну від інших додатків цього сегменту.
- надзвичайно велика бібліотека самих різноманітних вправ, їх видів, всіх типів для тих чи інших груп м'язів є одним із найбільших плюсів додатку, оскільки крім величезного вибору самих вправ, користувачі можуть ознайомитись з детальним проілюстрованим описом виконання тої чи іншої вправи, що допомагає більш ефективно виконувати тренування. Більш того, крім текстового опису, користувачі можуть ознайомитись з проанімованим варіантом виконання вправ, що допомагає не тільки зрозуміти, як потрібно виконувати вправу, але й наочно ознайомитись з правильною технікою виконання, щоб раптом не нашкодити собі;
- на мій погляд найбільш вдале виконання інтерфейсної частини, а саме калькулятору спортивних дисків для штанг, оскільки вирішення інтерфейсу є не тільки інтуїтивно зрозумілим для споживачів, але й візуально привабливим, чого не вистачає іншим додаткам цього сегменту.

Проте як казав Антуан де Сент–Екзюпери: “Нема в світі досконалості”, тому привожу список найяскравіших недоліків, які я виокремив після детального аналізу додатку:

- на жаль, відсутня можливість користування додатком без реєстрування свого аккаунту, про що нам інформується на першій сторінці після скачування та відкриття додатку. Звичайно, в наш час при використанні різноманітних додатків реєстрування свого облікового запису є звичайною практикою, проте на мою думку більш доречним є залишити вибір за споживачем та дати змогу користуватися продуктом без реєстрації. Зокрема, відсутність змоги користуватись додатком без реєстрації свого облікового запису в багатьох випадках відштовхує споживачів від бажання спробувати використовувати додаток, що впливає на негативну статистику користуванням додатку та зменшує кількість потенційних користувачів;
- відсутня можливість створювати план тренувань на тиждень вперед та притримуватись його, що кожний раз забирає більше часу обрання необхідного плану тренування. На мою думку, це можно реалізувати більш зручно, створивши можливість автоматичного обрання додатком тренування в залежності від дня тижня, та користуючись даними плану тренувань, нагадувати та мотивувати займатись користувача тим чи іншим сповіщенням;
- відсутність запросу додатком у споживача рівня втомлюваності після тренування, що не дає змогу створювати статистику стомлюваності від тих чи інших вправ та планів тренувань і створювати статистику оптимальних тренувань;
- на мій погляд, інтерфейс додатку занадто перевантаженний, більш спрощений дизайн був би не тільки лаконічнішим, а отже і привабливішим, що привертало би більше увагу споживачів, але й більш зручнішим для тренувань у спортзалі, оскільки перевантаженний дизайн супроводжують маленькі кнопки, натискання яких під час занять у спортзалі викликає дискомфорт.

2.2.2 Екран редагування плану тренувань

Екран планування тренувань додатки Strong складається з таблиці з вже створених тренувань, надаючи користувачу можливість обирати будь-яке тренування і не прив'язувати його до певного дня тижня (рис. 2.2).

Перевагою цього підходу є той факт, що користувач може експериментувати з різними тренуваннями в різні дні тижня.

Негативною стороною даного підходу є відсутність конкретного плану тренувань на тиждень, який більше підходить початківцям та не непрофесійним спортсменам.

Зовнішній вигляд екрану редагування плану тренувань виконан доволі лаконічно, порівнюючи з іншими додатками, дизайн заснований на білому тлі, а неагресивна кольорова гамма, яка складається з відтінків синього і сірих кольорів не відволікає користувачів та допомагає зосередитись, має заспокійливий ефект та не викликає роздратування.

Загалом дизайн еркану розроблений таким чином, щоб навіть при першому ознайомленні користувачу було інтуїтивно зрозумілий функціонал та логіка елементів. Як я казав раніше, единим недоліком у дизайні інтерфейсі можу виокремити дещо, все ж таки, перевантаженний інтерфейс та замалий розмір деяких кнопок.

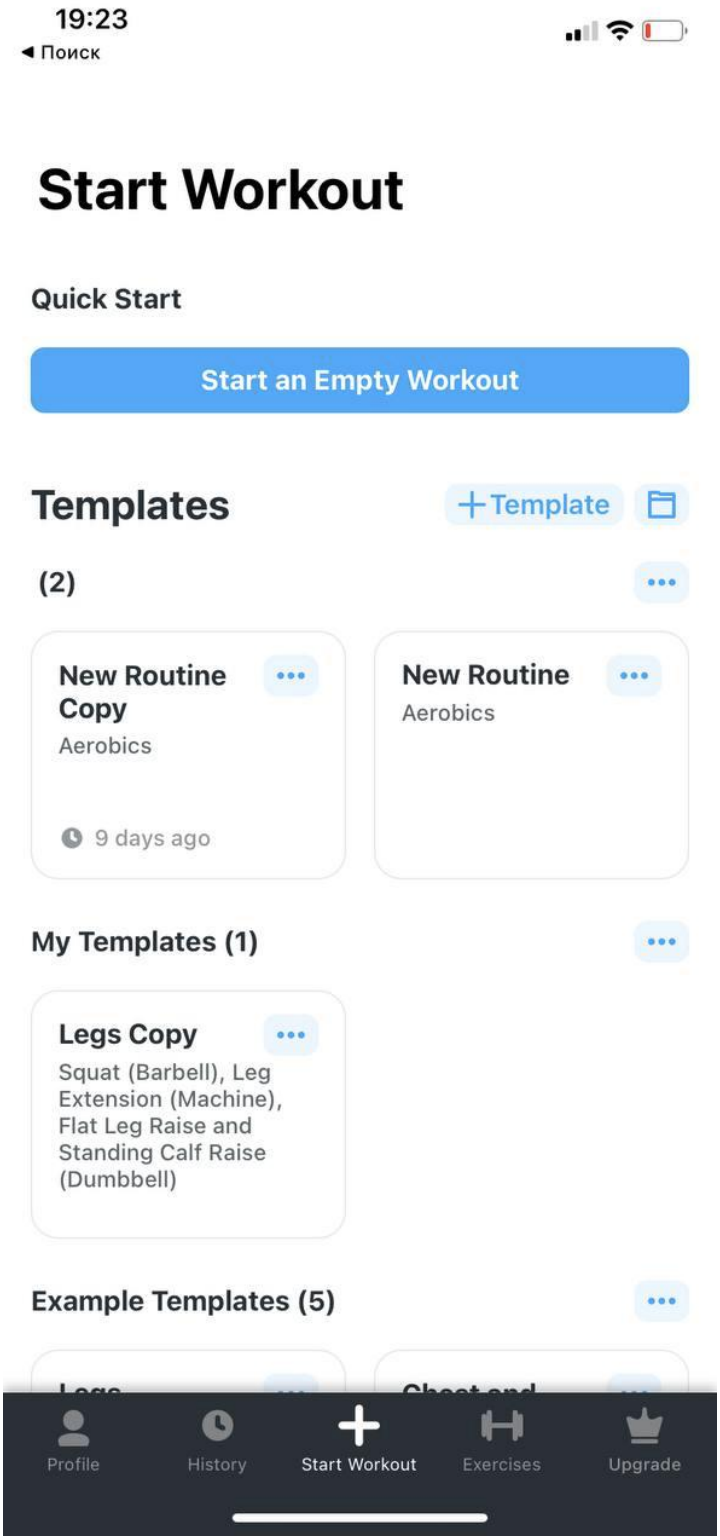


Рисунок 2.2 – Екран редагування плану тренувань додатку Strong

2.2.3 Екран планування тренувань

Ознайомившись з екраном плануваняя тренувань додатку Strong можно відмітити такий же підхід до дизайну, як і у екрану редагування плану тренувань, зокрема простий та інтуітивний інтерфейс та неагресивна кольорова гамма, яка складається з відтінків синього і сірих кольорів на білому тлі (рис. 2.3).

До переваг дизайну можна віднести наступне:

- можливість побачити список підходів для кожної вправи;
- можливість окремо запланувати вагу і кількість повторень для кожної вправи;
- можливість змінити порядок вправ за допомогою натискання і перетягування в списку на потрібну позицію.

Серед недоліків дизайну можна відзначити:

- вправи і підходи одночасно представлені на одному екрані, що, при великій кількості вправ, робить екран довгим і ускладнює можливість користувача розпізнати кокнретне тренуваняя з довгого списку назви тренувань;
- поля введення ваги і кількості повторень є недостатньо великими для комфортного використання на сенсорному екрані мобільного телефону, що ускладнює використання додатку у спротзалі та робить його більш дискомфортним.

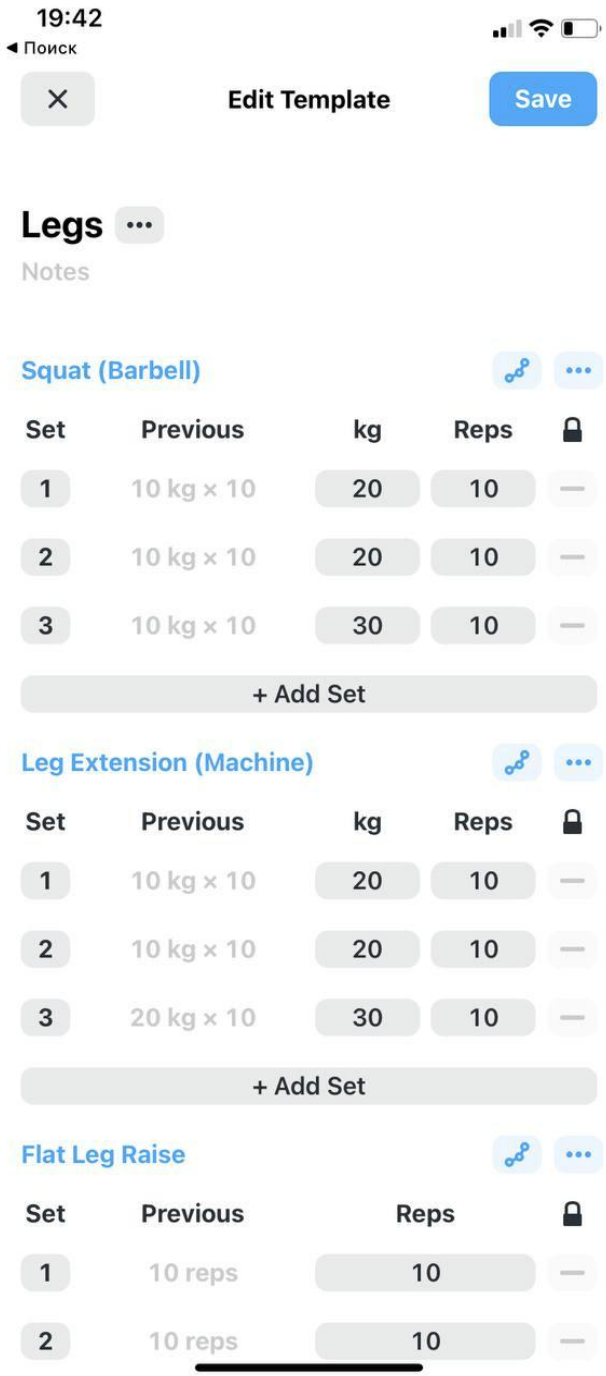


Рисунок 2.3 - Екран планування тренувань додатку Strong

2.2.4 Екран тренування

Екран тренування є однією із самий важливих і часто використовуваних частин додатку, якість якого може стати вирішальний факторог того, буде користувач застосовувати додаток, чи ні.

Додаток Strong в своєму підході використовує список всіх тренувань і підходів до них, і кнопку-іконку для збереження підходу (рис. 2.4).

Плюсом даного методу є той факт, що користувач одночасно може бачити всі вправи та підходи, а також наочно ознайомитись з прогресом в тренуванні.

Негативною стороною такого методу є те, що користувачеві необхідно знаходити свою поточне вправу і підходи до неї кожного разу, коли необхідно зробити новий запис. Так само, кнопки збереження запису невеликого розміру і завжди знаходиться на різній висоті. Оскільки це найчастіше використовувана функція, більш правильним вирішенням було б виділити її і зробити більше, щоб по-перше було інтуїтивно більш зрозуміле її місцезнаходження та для більшого комфорту у використанні цього екрану.

Також ми можемо спостерігати зміну у вирішення кольорової гамми екрану – на відміну від минулих екранів, де ми спостерігали лише відтінки синього та сірого на білому тлі, розробники додали для цього екрану варіацію відтінків зеленого, зокрема для кнопки збереження запису, про яку йшлося раніше. На мій погляд, це було вдалим рішенням, оскільки допомогає зосередитися на важливих частинах інтерфейсу та акцентує на них увагу користувача.

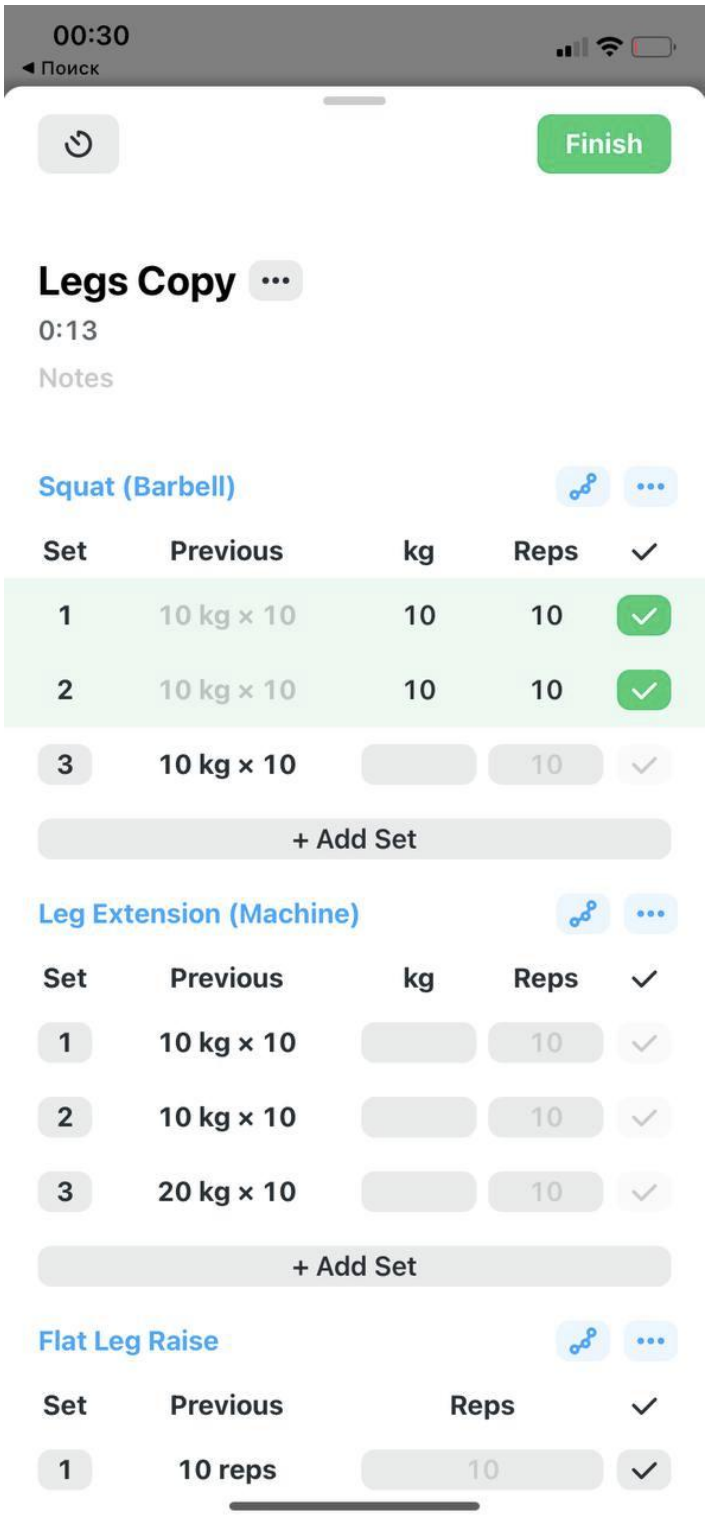


Рисунок 2.4 - Екран тренування додатку Strong

2.3 Додаток JeFit

2.3.1 Огляд та аналіз додатку

Розробники випустили додаток JeFit у 2010 році, за цей час на платформі Google Play вони мають понад 5 мільйонів завантажень.

Додаток підходить як для початкових спортсменів, так і для досвідчених культуристів, що безпосередньо впливає на збільшення кількості потенційних споживачів. Додаток має як безкоштовну версію, так і платну, яка коштує 39.99$ на рік.

Оглянувши та детально проаналізувавши цей додаток, я можу виділити наступні переваги та недоліки.

Перевеги:

- у додатку вже є готові плани тренувань, тому користувачі можуть не витрачати свій час, налаштовуючи базові тренування, оскільки вони вже є;
- до кожної вправи є не тільки текстовення пояснення техніки та правильного виконання вправи, але й наочний показ виконання вправи у вігляді анімації, що допомагає споживачам краще та єфективніше виконувати тренування, та економить час на пошуки правильної техніки виконання з інших джерел;
- користувачі мають можливість завантажити у додаток свої реальні фотографії після тренувань, що дає змогу наочно відслідковувати свій прогрес та мотивує продовжувати тренування.

До недоліків додатку можу виокремити наступне :

- відсутність синхронізації з додатками Apple Health і Google Fit, що обмежує споживачів та знижує ефективність від тренувань, оскільки данні додатки є популярними додатками, які відстежують стан здоровья, нагрузку споживачів та калорії;

- невелика кількість графіків та статистики, що обмежує наочне ознайомлення споживачів з прогрессом виконаня тренувань та їх аналіз для подальшого поліпшення та редагування;
- відсутність темної теми додатку, що обмежує споживачів налаштувати зовнішній вигляд під себе та створює дискомфорт у разі використання додатку у малонет темной темы;
- дизайн доволі низькоконтрастний, що повищує нагрузку на очі, викликає дискомфорт. Також із-за цього користувачі витрачають більше часу на зосередженні уваги на потрібних елементах та читання необхідних текстових елементах;
- під час синхронізації додатку з сервером, синхронизация с сервером, додаток перериваєтся, що тех впливає на негативний досвід використання додатку споживачами, оскільки це заважає споживачам нормально користуватися додатком у будь-який момент;
- присутність елементу соціальної мережі, що також відволікає увагу від важливих частин інтерфейсу та додає дискомфорту під час використання додатку;
- використання додатку можливо лише при вході зі свого аккаунта, який необхідно реєструвати, що може сприяти відмовленню використовувати додаток у потенційних споживачів.

2.3.2 Екран редагування плану тренувань

Екран планування тренувань додатку JeFit складається зі списку із запланованих тренувань на кожен обраний день тижня (рис. 2.5).

Використання фіксованого розкладу занять є одним із накращих рішень для споживачів, які являються початковими спортменами, оскільки дозволяє дотримуватися одного плану тренувань та вибудовувати звички щодо занять у конкретні дні тижня.

Проте на мій погляд, це рішення не підходить для досвідчених спорстменів, оскільки при такому підході доводиться слідувати конкретному плану тренувань або вносити зміни та коригувати конкретний план тренувань, що не дозволяє змішувати плани тренуваннь в різні дні тижня.

Щодо дизайну інтерфейсу, на мій погляд його вирішення виглядає достатньо органічним. У більшій частині елементі виконані у сірій та синій гаммі, що не дратує сприйняття користувачів. План на кожен день тижня акцентується кольоровим елементом з лівої сторони, що дозволяю асоціювати конкретний план з конкретним кольоров, а це сприяє більш швидкому считуванню та пошуку інформації та збільшення комфорту використання екрану. Усі елементи виконані у комфортних розмірах з точки зору ергономіки та зрічності використання, що також поліпшує позитивний досвід використання додатку у споживачів.

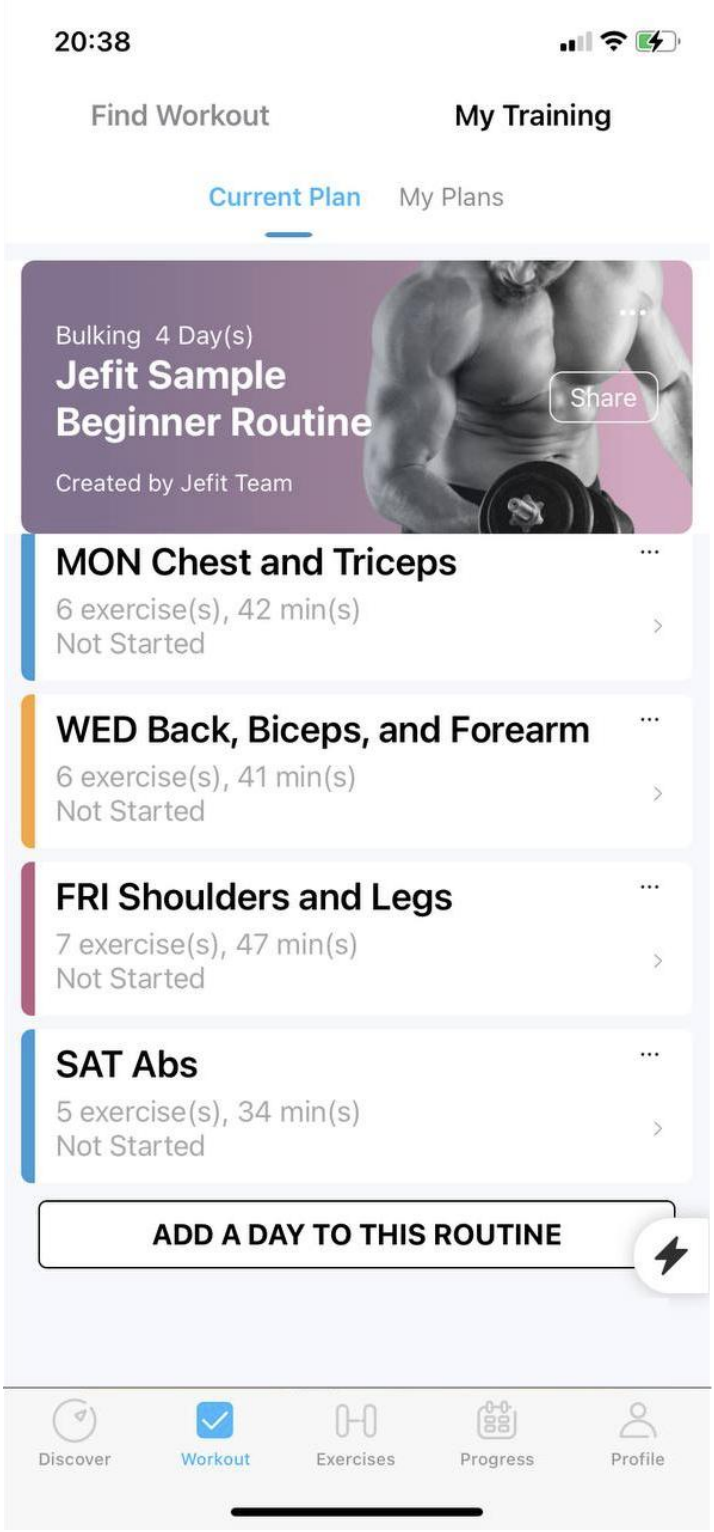


Рисунок 2.5 - Екран редагування плану тренувань додатку JeFit

2.3.3 Екран планування тренувань

Ознайомившись та проаналізувавши екран планування тренуваннь додатку, можна відзначити, що він виконаний за допомогою відтінків синіх і сірих кольорів, і складається зі списку запланованих вправ (рис. 2.6).

Кожен елемент списку вправ включає в себе назву вправи, зображення виконання вправи, а також поля введення для кількості підходів, кількості повторень, відпочинку та інтервалу.

Плюсами даного підходу є можливість бачити одночасно всі вправи в данному тренуванні, можливість вказати час відпочинку після підходу і інтервал.

Проте також тут є недолік, а саме - не можна вказати вагу і відпочинок для кожного підходу індивідуально. Також вводити кількість повторень для кожного підходу в одному полі введення може ускладнити процес планування прави для споживачів.

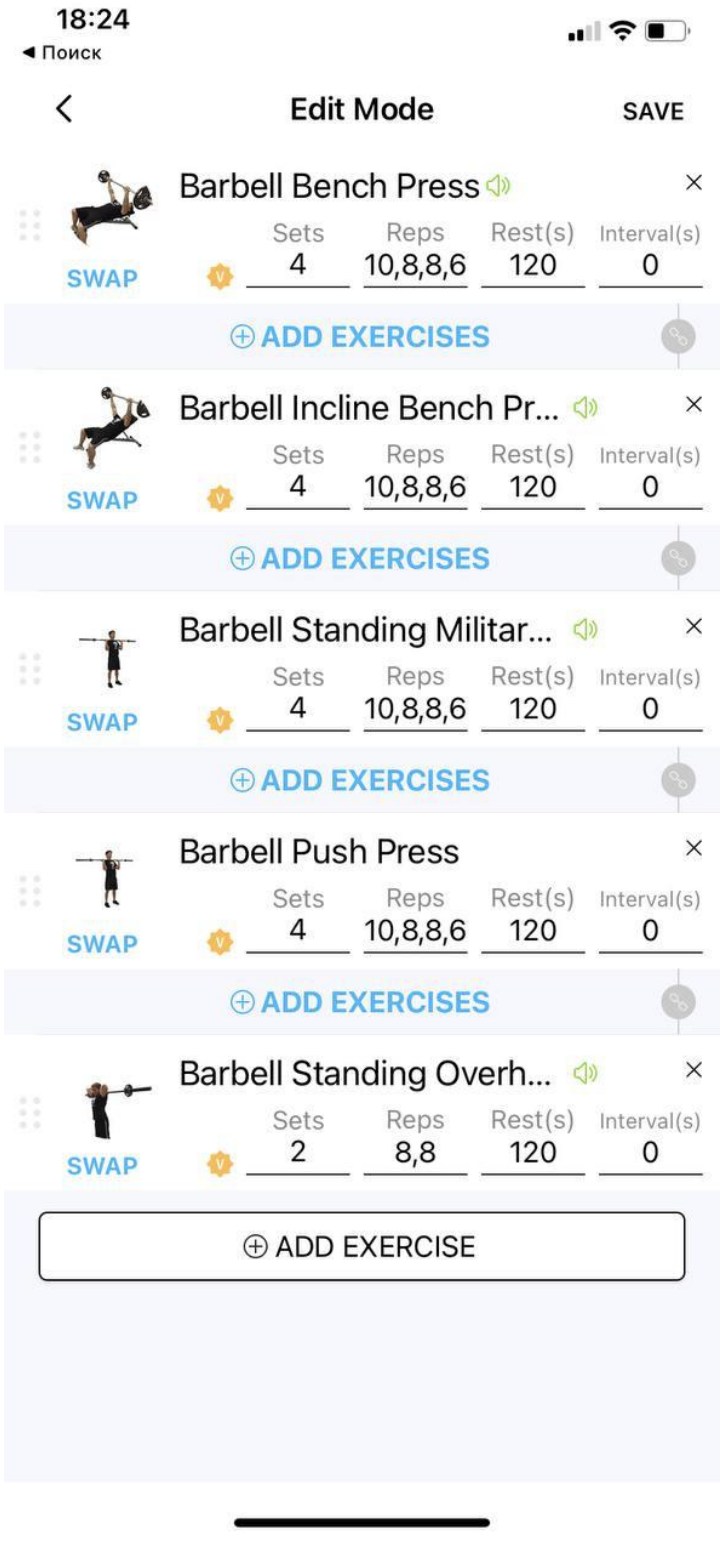


Рисунок 2.6 - Екран планування тренуваннь додатку JeFit

2.3.4 Екран тренування

Додаток jeFit використовує інший підхід, в якому вправи представлені у вигляді каруселі, а підходи у вигляді списку. Для збереження даних використовується одна кнопка -"залогувати підхід" знизу екрана, а індикатор поточного підходу змінює свою позицію під час тренування (рис. 2.7).

Даний підхід зручний тим, що кнопка "залогувати підхід" завжди знаходиться в одному місці і має досить великий розмір для зручного користування.

З мінусів нацьому екрані можна відзначити низький контраст тексту на елементах списку.

Загалом, разглядаючи зовнішній вигляд дизайну, відмічаємо, що на ньому елементи також загалом у сірій та синій гаммі кольорів. Акцентним зеленим кольором привартається увага до вже виконаних підходів, що сприяє розумінню логіки екрану споживачами.

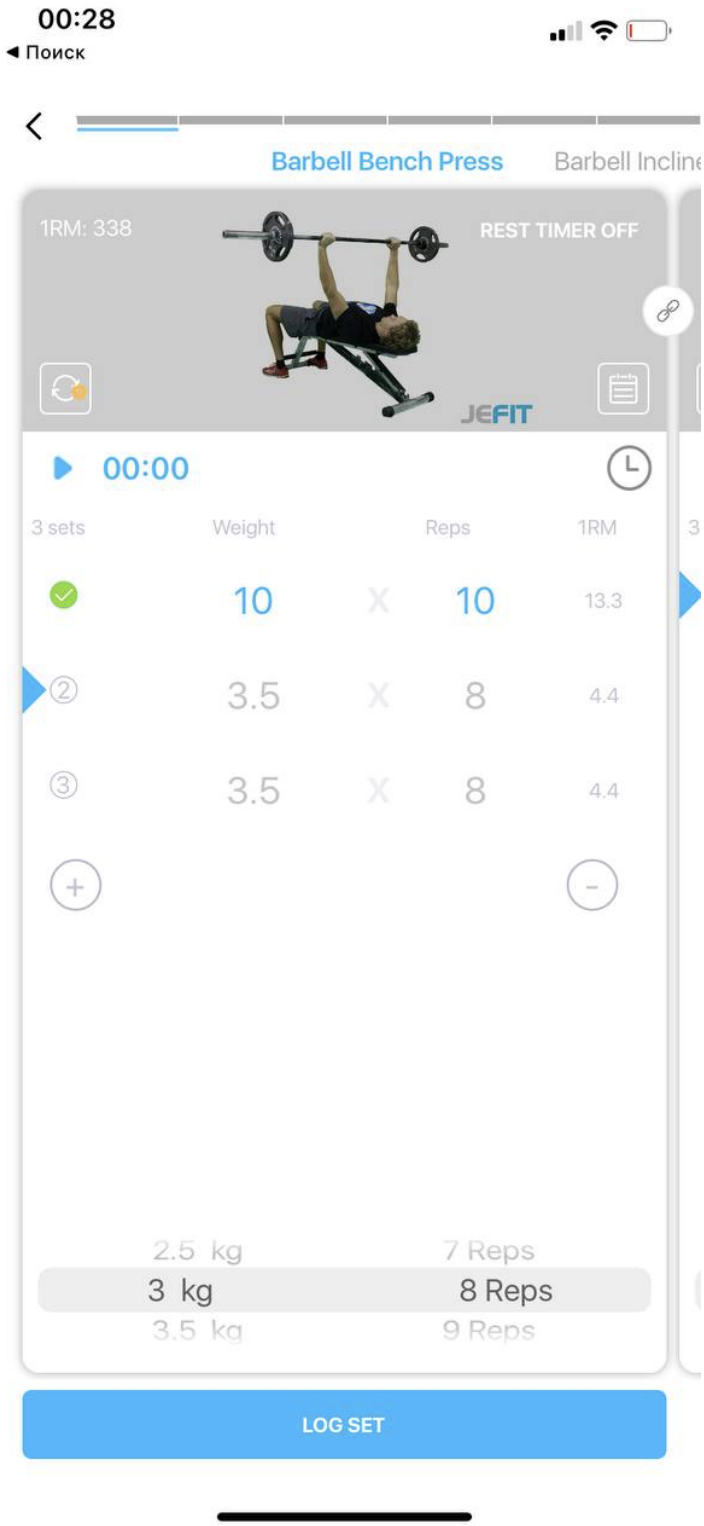


Рисунок 2.7 - Екран тренування додатку JeFit

2.4 Додаток Workout

2.4.1 Огляд та аналіз додатку

Додаток Workout розробила компанія Fitness22, яка спеціалізується на додатках для фітнеса, здоров'я та спорту. Додаток доданий до платформи Google Play у 2016 році, за цей час додаток був завантажений 100 000 на платформі Google Play.

На відміну від додатків, згаданих та розглянутих мною раніше, цей додаток спрямований виключно на початківців, що безпосередньо зменьшує кількість потенційних споживачів. Додаток сам створює план тренування та ваги, тому більше відіграє роль цифрового тренера, аніж помічника у тренуваннях.

Проаналізувавши додаток, можу виокремити наступні переваги та недоліки.

Перевеги:

- оскільки додаток розроблений для початківців, то в ньому вже є велика кількість готових планів тренувань, що облегшує у додатку вже є готові плани тренувань, тому користувачі можуть не витрачати свій час на пошук та налаштування базових тренувань, оскільки вони вже є;
- в залежності від своїх особистих цілей та характеристик, таких як ріст, вага, ціль схуднути або накачати конкретні м'язи, додаток створює та налаштовує індивідуальний план тренувань для кожного окремого споживача, використовуючи та аналізуючи введені користувачем данні;
- на відміну від попереднього розглянутого мною додатку, цей додаток для використання не потребує входу та створення свого аккаунту, що полегшує використання та збільшує кількість потенційних користувачів;
- у додатку є така функція, як нагадування про наступне тренування, що допомогає більш ефективніше використовувати додаток;
- також у додатку є велика кількість різноманітних статей про правильне ведення тренувань та бодібілдінг, що допомагає початківцям більш зануритись у тренування не лише з точки зору тренувань, але й з точки зору

проінформованності та теоретичного знання правильної техніки виконання тих чи інших вправ.

До мінусів можу відвести:

- оскільки додаток розроблений для початківців, то в ньому вже є велика кількість готових планів тренувань, що облегшує у додатку вже є готові плани тренувань, тому користувачі можуть не витрачати свій час на пошук та налаштування базових тренувань, оскільки вони вже є;
- відсутність графіків та статистики на операційній системі Android, що обмежує можливість використання додатку для споживачів, які використовують згадану операційну систему;
- відсутність синхронізації з додатками Apple Health і Google Fit, що обмежує споживачів та знижує ефективність від тренувань, оскільки данні додатки є популярними додатками, які відстежують стан здоровья, нагрузку споживачів та калорії;
- не зважаючи на те, що відсутність необхідності створювати свій аккаунт я відмічаю як плюс, оскільки це заощаджує час та збільшує кількість потенційних користувачів, це вирішення має свої мінуси, а саме оскільки нема синхронізації з акаунтом, то всі данні користувача при втраті смартфону також будуть втрачені.

2.4.2 Екран редагування плану тренувань

Додаток Workout не використовує в своєму підході до планування тренувань список. Замість цього, використовується елемент карусель і кнопка "додати тренування" (рис. 2.7).

Такий підхід дозволяє не використовувати додаткових екранів для створення плану тренувань, і редагування самих тренувань. Але в той же час, для того щоб прочитати список тренувань в в плані, необхідно відкрити план і перегорнути всю карусель, що займає більше часу, аніж перегляд списку і негативно позначається на враженні від використання програми.

Аналізуючи кольорове вирішення дизайну інтерфейсу додатку, бачимо, що тут також використовується синя та сіра гамма кольорів, що не викликає роздратування та позитивно позначається на враженні від використання програми.

Виходячи з аналізу альтернатив, можна зробити висновок, що найбільш ефективним способом представлення плану тренувань буде список, так як необхідно щоб з погляду на поточне тренування користувач міг візуально оцінити вільні і зайняті дні в даній програмі. Також, для планування вправ для вільного дня, користувачеві досить просто натиснути на нього, після чого включиться екран планування тренування.

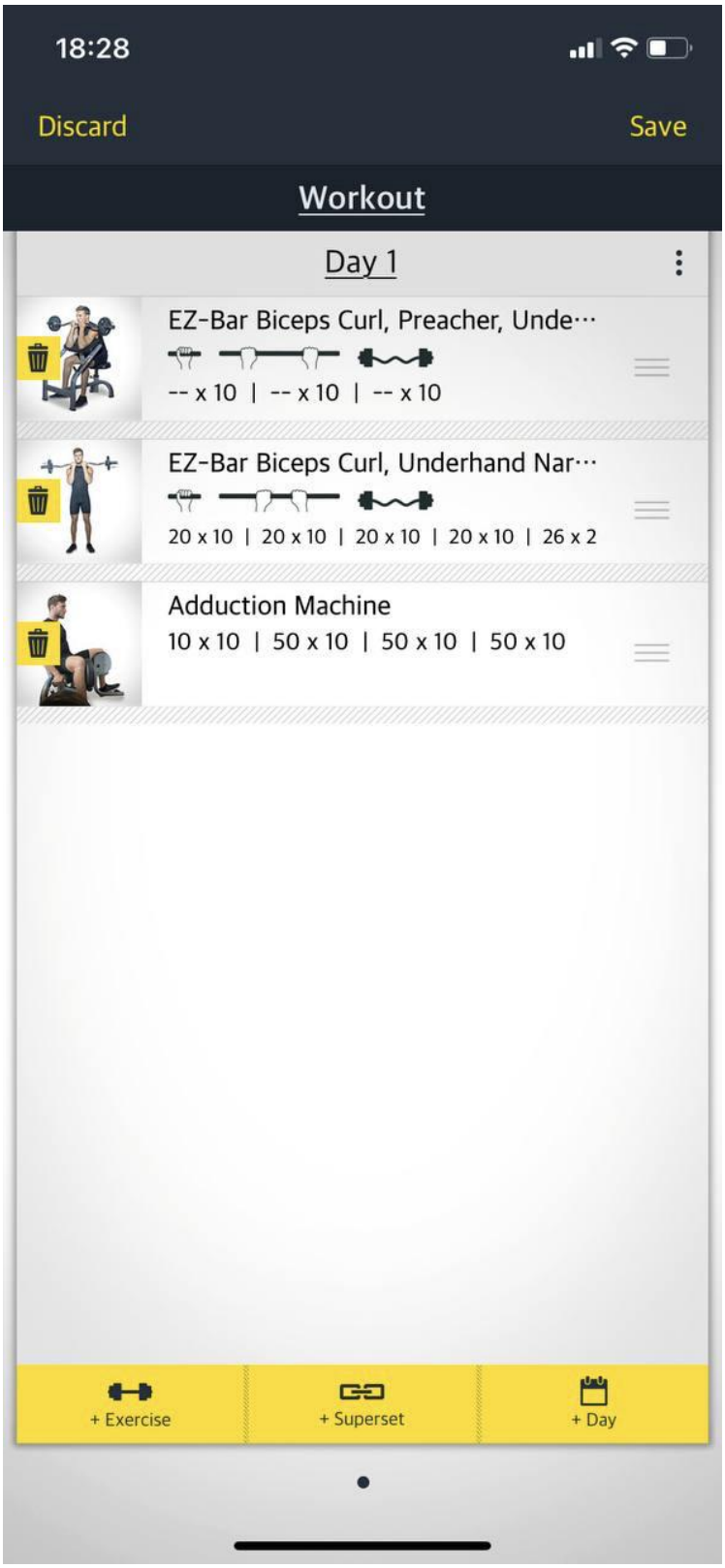


Рисунок 2.7 - Екран редагування плану тренувань додатку JeFit

2.4.3 Екран планування тренувань

Додаток Workout, в свою чергу, використовує інший підхід в дизайні свого екрану. Тут використовується гамма жовтих і чорних кольорів, а сам дизайн складається зі списку вправ і інформації про тип, кількості підходів і вагу (рис. 2.7).

Для того, щоб змінити кількість підходів або їх вагу, необхідно перейти, натиснути на вправу і перейти на екран редагування підходів.

Плюсом такого підходу є те, що на екрані одночасно немає безлічі полів введення, і при плануванні підходів користувач сфокусований лише на одній вправі.

Мінусом є той факт, що додатковий перехід на екран планування підходів і назад займає у користувача багато часу в разі, якщо потрібно спланувати багато вправ.

Також до мінусів можна віднести колірну гамму додатку, контрастний жовтий колір може служити додатковим фактом роздратування очей при використанні додатку та може негативно відображатися на враженні від використання додатку, що в свою чергу може зменьшити кількість потенційних споживачів додатку.

2.4.4 Екран тренування

У свою чергу, додаток Workout використовує фіксовані екрани зі списком підходів для відображення підходу. У кожного елемента списку є кнопка збереження запису і переходу до наступного підходу (рис. 2.9).

Такий підхід є незручним, оскільки для вибору іншої вправи необхідно переходити на екран вибору вправ. Також, оскільки положення кнопки збереження підходу постійно змінюється, користувачеві кожен раз необхідно натискати в різне місце, що призводить до негативного досвіду використання додатку споживачами.

Ознайомившись з дизайном екрану, можу відмітити, що тут також використовуеться гамма жовтих та чорних відтінків, що через свій надзвичайно

великий контраст може втомлювати очі под час довготривалого використання додатку, що також збільшує кількість негативного досвіду використання додатку.

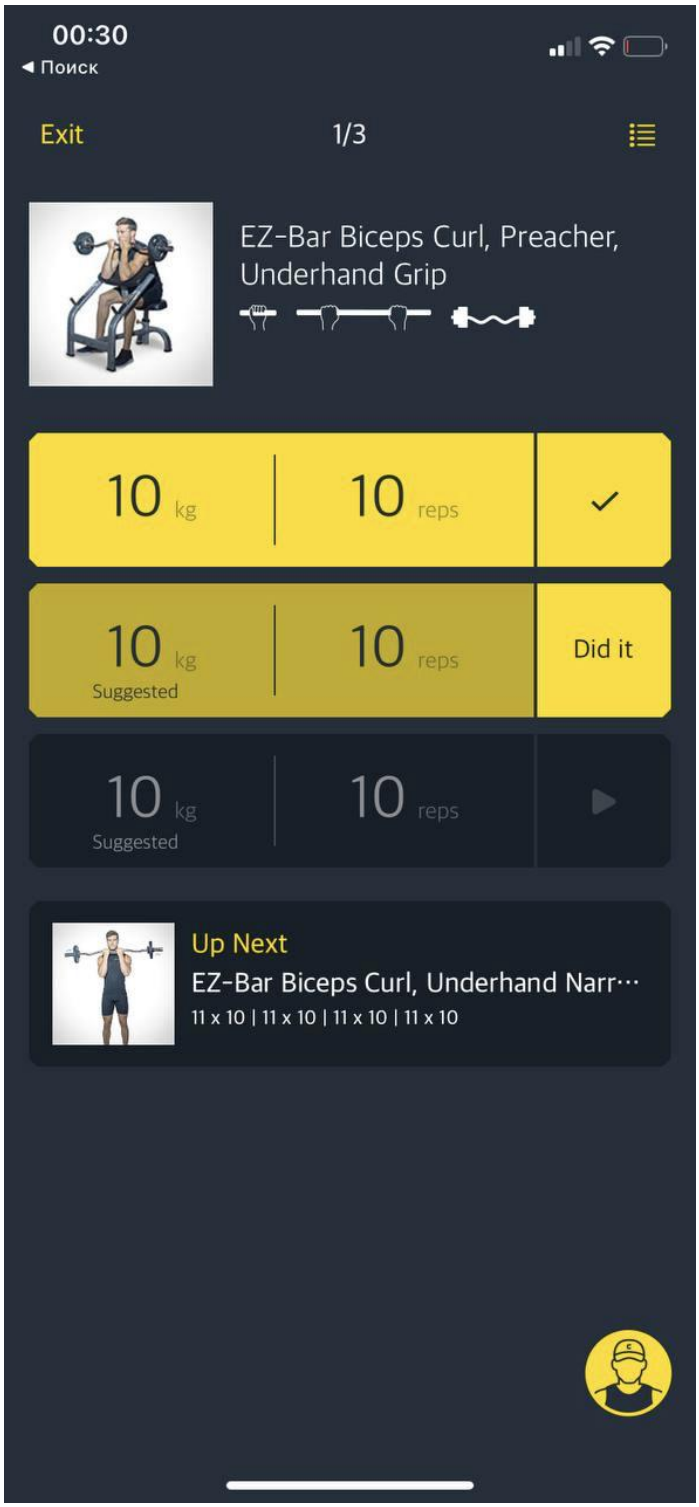


Рисунок 2.9 - Екран тренування додатку Workout

3 РОЗРОБКА ДОДАТКУ НА ОСНОВІ ФРЕЙМВОРКА FLUTTER

3.1 Технічне завдання

Провівши глубокий аналіз додатків-конкурентів, виокремивши недоліки до переваги кожного додатку, проаналізувавши додатку з точки зору комфортності використання та логіки додатку, я можу сформувати основне технічне завдання для додатку:

- кросс-платформний додаток на фреймворке Flutter по типу планера з тематикою для спортсменів-культуристів;
- здатність планувати свої тренування задля збільшення кількості потенційних споживачів;
- таймер відпочинку задля запобігання перевантаження споживачів та для більш комфортного використання додатку;
- можливість синхронізувати данні користувачів з акаунтом задля запобігання втрачення цих данних;
- велика бібліотека вправ задля збільшення варіацій планування тренування та для більш професійного та комфортного використання додатку;
- записисування виконаних тренувань для більш глибокого розуміння проведених тренувань.

У майбутньому додаток також буде доповнюватись наступники пунктами:

- кросс-платформний додаток на фреймворке Flutter по типу планера з тематикою для спортсменів-культуристів;
- велика кількість статистичних данних задня наочного відображення та аналізу прогрясу тренувань та збільшення рівня комфортності використання додатку, та збільшення кількості потенційних споживачів;

- оповіщення про майбутнє тренування, для завчасного настрою споживачів на тренування та планування свого часа з урахуванням тренування;
- оптимізація додатку під Apple Watch задля збільшення кількості потенційних споживачів додатку;
- опис правильної техніки виконання вправ тренування задля більш професійного проведення тренування споживачами та покращення позитивного досвіду від використання додатку;
- використання не лише текстового опису правильної техніки виконання вправ тренування, але й графічні пояснення, з окрема у вигляді анімації, для наочного ознайомлення та розуміння провильної техніки виконання вправ тренування та заощадження часу споживачів на пошуки графічних пояснень техніки виконання вправ з інших джерел;
- надання можливості обрати готові плани тренувань із існуючих у додатку задля більшого комфорту використання додатку початковими спортсменами, які ще не дуже освідчені у всіх необхідних складових кожного окремого типу тренувань;
- синхронізація з такими додатками, як Apple Health і Google Fit, оскільки вони є популярними додатками, які відстежують стан здоровья, нагрузку споживачів та калорії, що облегшує споживачам відстежування свого стану та покращує позитивний досвід від використання додатку.

3.2 Розробка дизайну додатку на основі проведеного аналізу

3.2.1 Екран редагування плану тренувань

Провівши аналіз альтернативних додатків та виокремивши переваги та недоліки кожного, можна перейти до дизайну аналогічного екрану свого додатку (рис. 3.1).

Колірна гамма синьо-сірих кольорів є найбільш популярною серед додатків даного типу, тому для дизайну додатки була обрана саме вона. Найголовнішим на екрані є відображення вправ в данному тренуванні. Представлення у вигляді списку є доволі популярним рішенням, яке зрозуміле користувачами і дозволяє ефективно відобразити велику кількість вправ на екрані, бачити їх порядок, а також змінювати порядок за допомогою натискання і перетягування елементів списку.

Для відображення підходів до вправ, я вирішив використовувати розкриваючі елементи списку, таким чином, при натисканні на вправу, воно розкривається і показує список підходів, запланованих для нього. За допомогою даного рішення ми можемо досягти двох цілей: дозволити користувачеві бачити одночасно всі заплановані вправи, і не витрачати час на перехід між екранами для редагування підходів.

Кожен підхід складається з полів для введення ваги, кількості повторень, і вибору відпочинку після підходу.

Для додавання нових вправ використовується концепт представлений в Material Design "Floating Action Button", синя кнопка внизу праворуч на екрані, що привертає до себе увагу контрастним синім кольором та розташуванням.

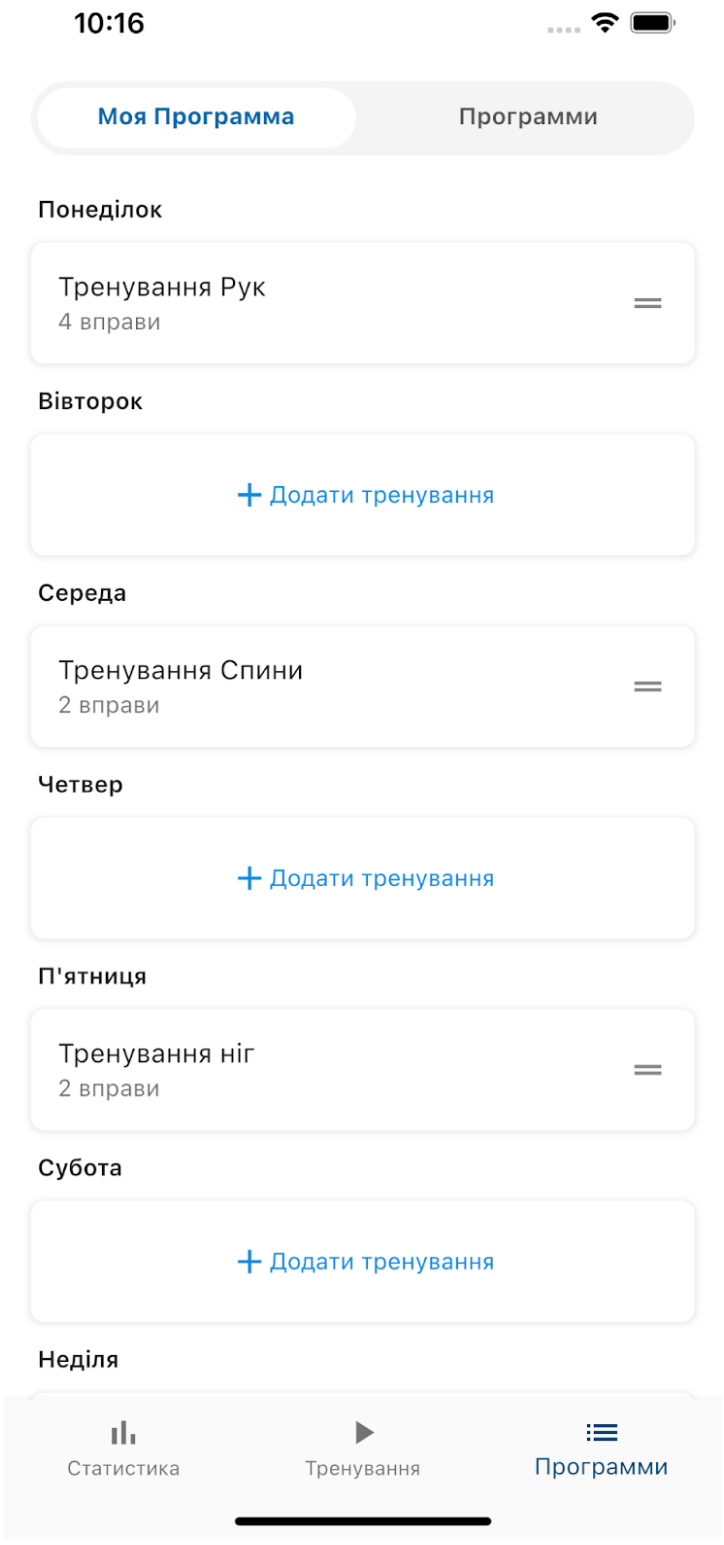


Рисунок 3.1 – Екран редагування плану тренувань мого додатку

3.2.2 Екран планування тренувань

Також, одна з найчастіших використовуваних частин додатку – це вибір тренування з поточного плану тренувань на тиждень і початок запису тренування (рис. 3.2).

На цьому етапі важливо скоротити кількість кроків, які необхідно зробити користувачеві для того, щоб знайти потрібне тренування і почати його запис.

Отже, щоб збільшити позитивний досвід від використання додатку споживачем, задля зменшення кількості кроків у пошуку потрібного тренування, екран планування та вибору тренування буде першим, що користувач бачить при включенні додатку.

Список вправ, що входять до цього тренування видно на нижній панелі, яку можна витягнути і почати запис за допомогою кнопки "почати тренування".

Також мною був обран спосіб вибору тренування за допомогою елемента "карусель", яка автоматично перевертається до тренування для поточного дня тижня, оскільки після проведенного мною глибоко аналізу додатків-конкурентів я прийшов до висновку, що це одне із найбільш вдалих та зрозумілих користувачам рішення.

З приводу кольорового вирішення дизайну інтерфейсу екрану, я продовжую використання гамми синіх та сірих відтінків, оскільки як я згадував раніше, на мою думку це є одним із найвдалих рішень цього питання.

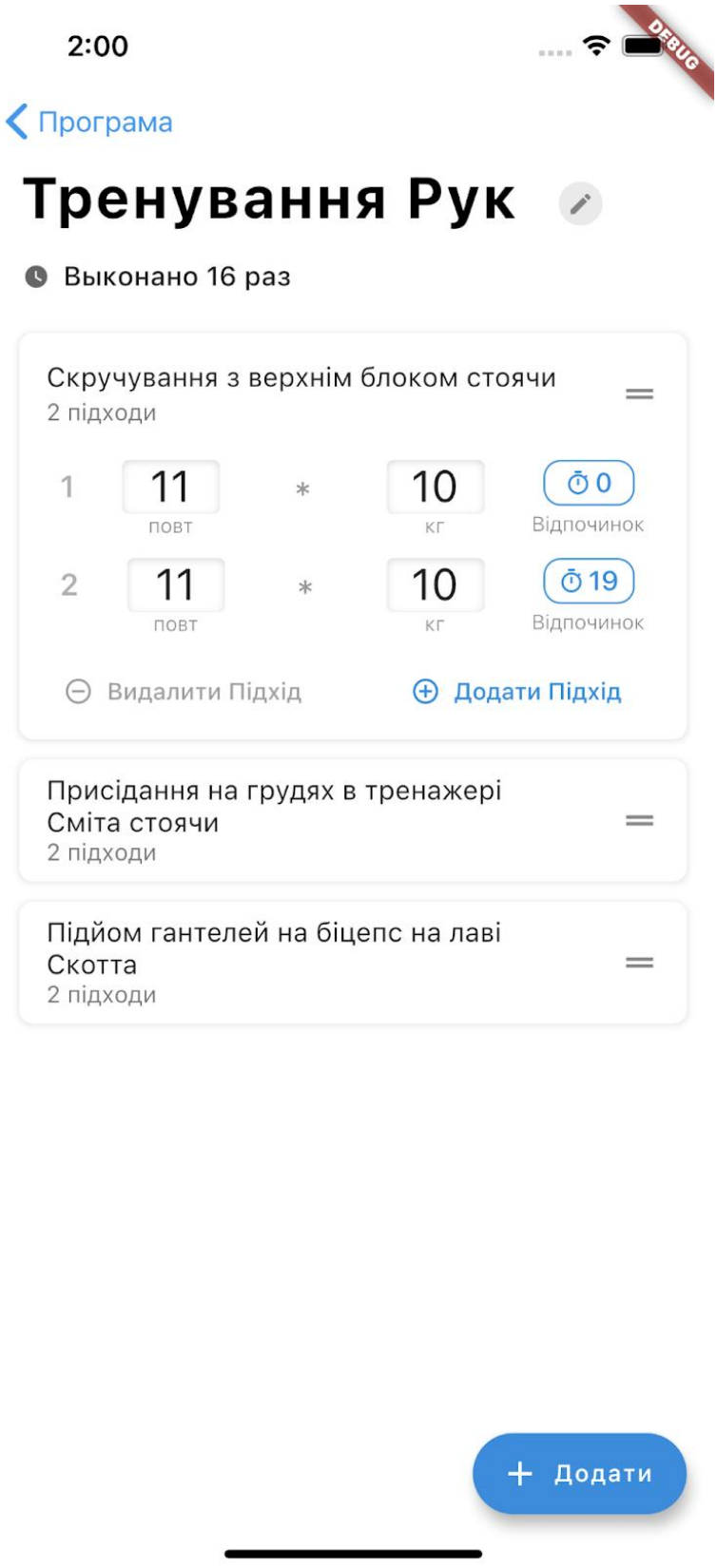


Рисунок 3.2 - Экран планування тренувань мого додатку

3.2.3 Екран тренування

Після того як тренування обране і користувач перейшов до режим запису вправ, на екрані з’являються підходи, а також вага і кількість повторень кожного підходу (рис.3.3).

При натисканні на "зберегти підхід", створюється запис в базі даних, на основі якої додаток будує графіки прогресу і дозволяє користувачеві відслідковувати, кількість і хід своїх тренувань.

Важливим також є таймер відпочинку після збереження кожного підходу, оскільки часу на відновлення повинно бути достатньо, щоб відновити сили, але не надто багато, щоб спортсмен не втратив тонус та бажання прожовжувати тренування.

Для цього екрану я вибрав не тільки відтінки синіх та сірого кольорів, але й ввів зелений акцентуючий колір, щоб привернути увагу користувачів до данної кнопки, яка є дуже важливою для створення бази даних та створення подальшого графіку тренувань.

Рисунок 3.3 - Экран тренування мого додатку

3.3 Аналіз існуючих архітектур.Вибір архітектури додатку

Для розробки мобільного додатку ключовим моментом є вибір способу управління даними, архітектура. Для додатків на основі фреймворку Flutter існує кілька популярних рішень, ознайомлення та аналіз з якими я приводжу нижче.

3.3.1 BLoC

BloC архітектура (рис. 3.4), розгортка абривіатури - Business Logical Component.

Ключовим об'єктом в BLoC архітектурі є BLoC - компонент.

BLoC компонент обробляє передані йому події й оновлює додатки відповідно до отриманих події. Після обновлення стану програми, BLoC - компонент оповіщає інтерфейс про те, що стан оновилося, після чого інтерфейс оновлюється відповідно до нового стану додатку [11].

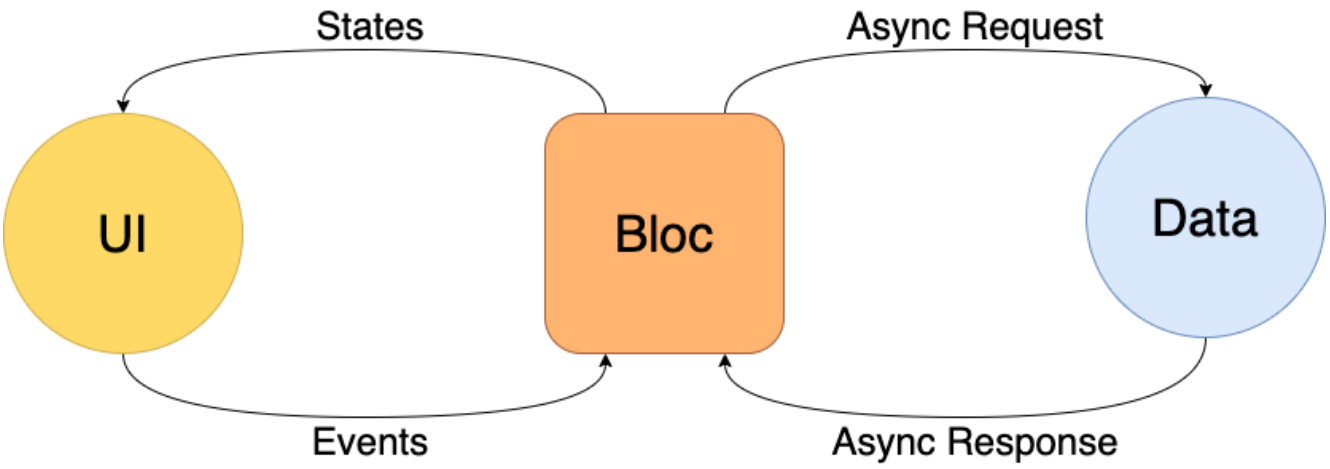


Рисунок 3.4 - BloC архітектура

У Flutter даний концепт реалізовано за допомогою Stream і Sink.

Sink використовується для введення даних і подій в BLoC - компонент, з огляду на нові дані BLoC оновлює стан програми та сповіщає інтерфейс за допомогою елемента Stream.

Перевагами BloC є можливість перевикористати BLoC - компоненти і їх частини, а також простота тестування.

Недоліком є те, що для кожного окремого параметра стану необхідно створювати Stream і Sink, що збільшує кількість необхідно коду для реалізації архітектури, а також ускладнює підтримку програми.

3.3.2 Redux

Основою архітектури Redux є те, що стан додатки змінюється тільки в одному місці. Також, при кожній зміні стану його необхідно створити знову, щоб уникнути мутації стану [10].

У Redux архітектурі, при подію інтерфейсу створюється Action, який передається потім в Reducer, який створює новий стан і записує його в Store. Після того, як в Store стан змінюється, інтерфейс отримує повідомлення про новий стан і оновлює презентацію відповідно (рис. 3.5).

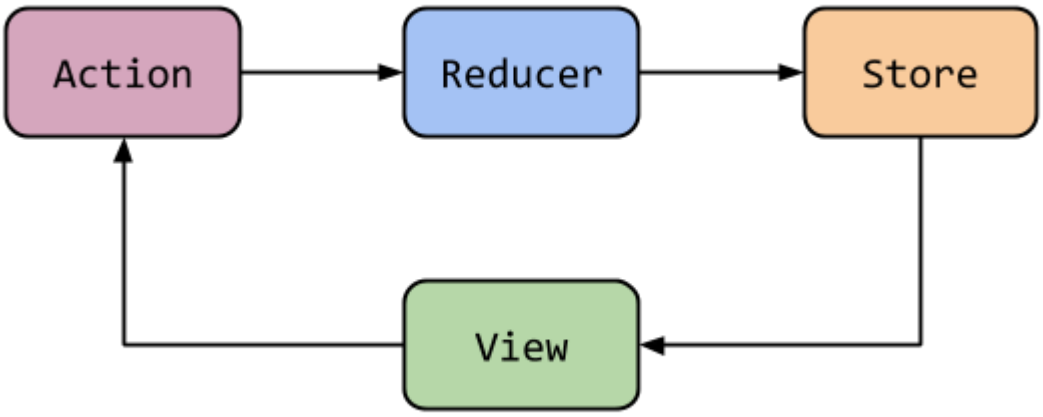


Рисунок 3.5 - Redux архітектура

Перевагами даного підходу є легкість тестування і пошуку помилок програми. Але при цьому, потрібна велика кількість коду для реалізації цієї архітектури. Також, даний підхід значно ускладнюється при роботі з асинхронними операціями.

3.3.3 MobX

В архітектурі MobX стан змінюється за допомогою Action, який створюється при взаємодії з елементами інтерфейсу, на що реагує Computed і Reaction.
Computed створює свій стан, виходячи з оновлень базового стану програми.
Після того, як стан програми оновлено і Computed [8] оновив свій внутрішній стан, працює Reaction [9], який оновлює інтерфейс користувача (рис. 3.6).

MobX є ефективною архітектурою для роботи зі станом мобільного додатка, в той же час вимагаючи мінімальну кількість коду. Через ці якості він був обраний мною для даного проекту дипломної роботи.

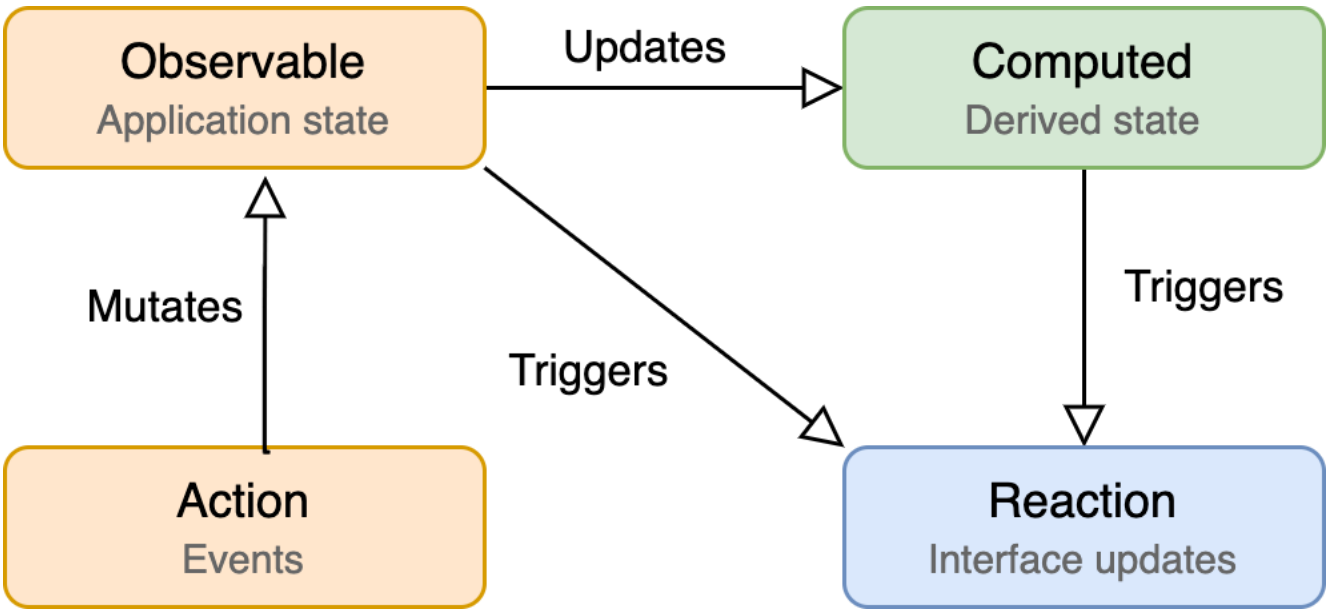


Рисунок 3.6 - MobX архітектура

3.4 Проектування екрану планування програми тренувань

Для побудови екрану планування програми тренувань використовується дерево віджетів (Додаток А). Оскільки екран тренувань складається зі списку, елемент ReorderableList буде основним компонентом будують цю частину інтерфейсу. Для синхронізації інтерфейсу і внутрішнього стан програми, використовується елемент Observer. Observer отримує повідомлення про те, що стан програми змінено і оновлює інтерфейс до відповідності з ним. Алгоритм роботи даного екрану відображений на блок схемою (Рис 3.7).

Рисунок 3.7 – Блок схема екрану планування програми тренувань

Стан програми складається з масиву створених тренувань, та методів їх модифікації onReorder, addRoutine, UpdateRoutine, DeleteRoutine. За допомогою цих методів та

можливості через елемент Observer отримувати нові дані після кожного оновлення реалізовані базові операції зміни данних CRUD – Create, Read, Update, Delete (Рис 3.8).

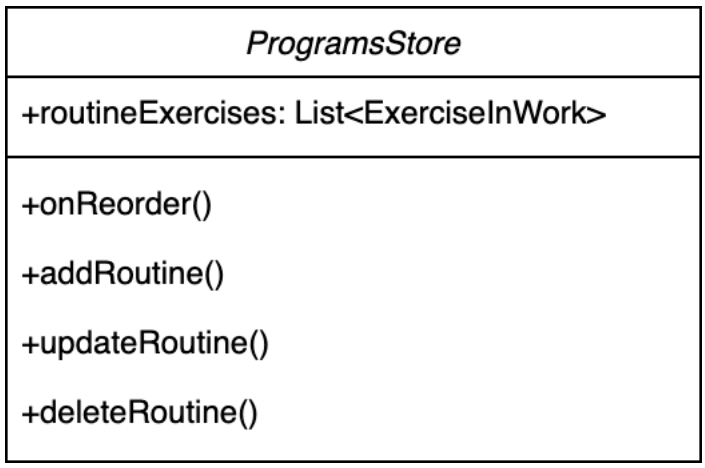


Рисунок 3.8 – Інтерфейс стану екрану планування програми тренувань

3.5 Проектування екрану планування тренування

Екран редагування тренування складається зі списку, який використовує різнотипні елементи. Для використання різнотипних елементів списку використовується елемент CustomScrollView. CustomScrollView дозволяє використовувати різнотипні елементи, при цьому перевикористовувати однотипні елементи для підвищення продуктивності списку.

Базовим елементом списку є Sliver, який призначений для відображення елементів прокрутки на екрані, а також для перевикористання елемента з іншими даними в разі довгих списків з однотипними елементам. Для відображення данного екрану використовуються дерево віджетів(Додаток Б), яке одночасно виконує декілька функцій: малювання інтерфейсу, пасування подій інтерфейсу до контролера стану програми, реагування на зміни у стані данних інтерфейсу та оновлення його згідно цим данним. На рисунку 3.9 зображена блок-схема логіки роботи даного екрану.

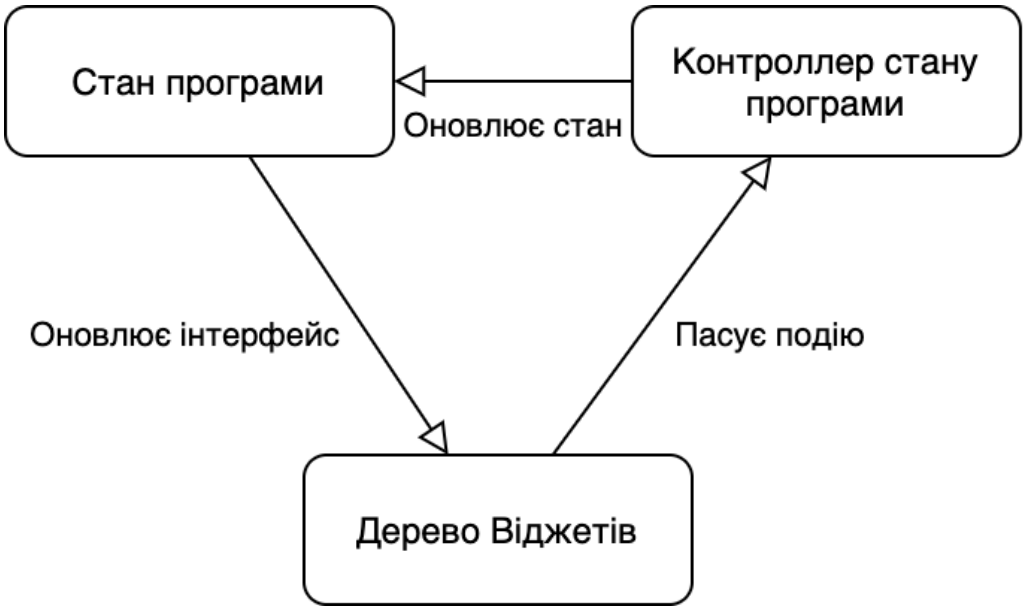


Рисунок 3.9 – Блок-схема логіки роботи екрану редагування тренування

Оскільки даний екран контролює не лише дані про вправи, які необхідно виконати, а й інформацію про підходи до них, контролер стану програми має реалізувати одночастно два базох інтерфейса модифікації даних – один для вправ і інший для підходів до них. На рис 3.10 відображений інтерфейс комунікації зі станом данного екрану.

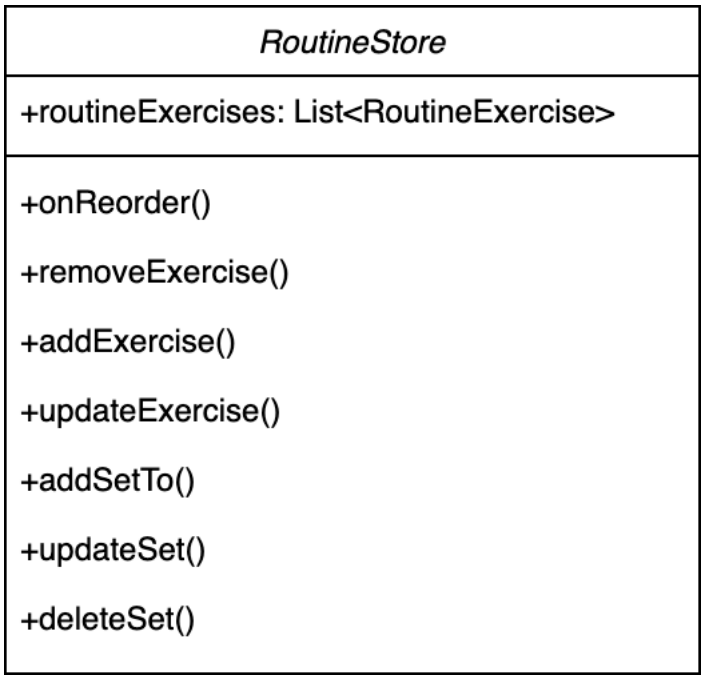


Рисунок 3.10 – Інтерфейс комунікації зі станом данного екрану

3.6 Проектування екрану запису тренування

Екран запису тренування є найбільш важливим для додатку, оскільки саме він є лицем та буде представляти найбільш використовуваний функціонал - створення записів тренування, на основі яких в подальшому можливо буде провести аналіз прогресу користувача.

На відміну від інших екранів, головним віджетом стане карусель, яку представляє віджет PageView. Карусель вміщує в собі вправи, які користувач має виконувати крок за кромом, дерево віджетів данного екрану приведено в Додатку В. Також, кожен елемент каруселі використвує лист з підходів, які необхідно виконати в данній вправі, дерево віджетів в Додатку Г. Життєвий цикл данного екрану такий самий, як і на екрані редагування тренування (рис. 3.11).

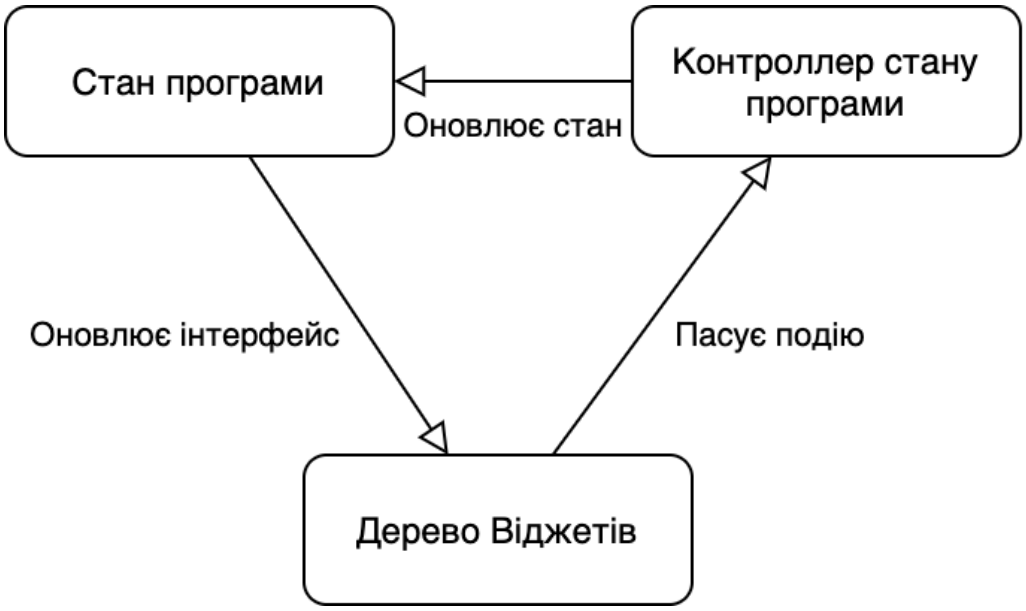


Рисунок 3.11 – Блок-схема логіки роботи екрану запису тренування

Особливістью дизайну данного екрану є те, що користувач не повинен приймати рішень та редагувати існуючі данні, а лише виконувати вправи та підходи, на які вказує інтерфейс, а також відпочивати впродовж певного проміжку часу, який відмірюється додатком.

Інтерфейс контролеру стану програми складається з збереженних змінних стану currentRoutine, який відповідає за те, яке тренуання вибране на даний момент, currentExerciseIndex, який зберігає індекс вправи, яку на данний момент має виконати користувач, а також routineExercises, що є списком вправ, які користуач має виконати в поточному тренуванні.

Інтерфейс контроллеру стану складається з таких методів - startWorkout, який переводить додаток в режим запису тренування, moveToExercise що змінює поточну вправу, setCompleted, що зберігає дані про виконаний підхід та рухає індикатор поточного підходу на наступний, та finishWorkout, що завершує режим запису тренування. Інтерфейс стану та його контролеру відображено на рисунку 3.12.

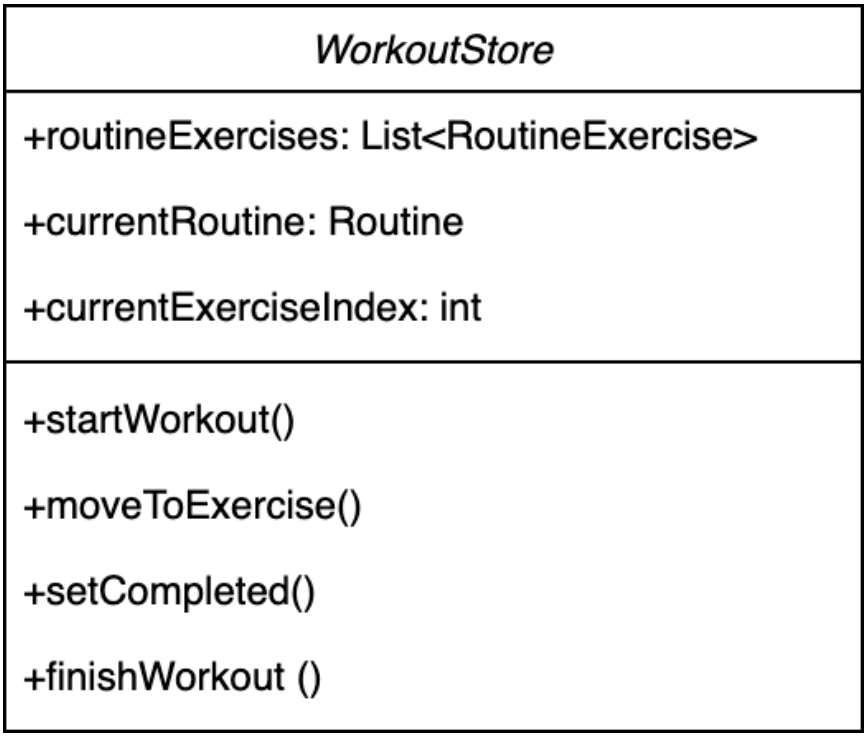


Рисунок 3.12 – Інтерфейс станів та контролерів відповідно

ВИСНОВКИ

Дослідивши сьогоденний ринок мобільних додатків, та проаналізувавши технології їх розробки, мною була обрана тематика спортивних тренувань, адже ця сторона життя для кожного з нас завжди буде актуально.У ході виконання дипломного проекту мною було був розроблений кросс-платформний додаток типу планер-трекеру на основі фреймворку «Flutter» та архітектури MobX. У ході виконання дипломного проекту, на кожному етапі розробки додатку переді мною було поставлено важливе завдання, які я виконав, використовуючи теоретичні знання, практичні навички та вміння, здобуті під час навчання в університеті на спеціальності 122 «Комп'ютерні науки».

Першим та важливим для подальшого розуміння процесів розробки, логіки інтерфейсу, потреб користувача та зовнішнього вигляду додатку було завдання проаналізувати та порівняти сучасні фреймврки, які використовують для створення кросс-платформних додатків та проаналізувати аналогічні кросс-платформні додатки-конкуренти.

Проаналізувавши та порівнявши технології React-native, Cordova та Flutter, я прийшов до висновку, що використання фреймворку Flutter буде найбільш доцільним для вирішення наступних завдань, що будуть стоять переді мною, адже фреймворк Flutter використовує власні віджети, що дає можливість для реалізації будь-якого дизайну, та має універсальний двигун, що підтримує створення додатків на Android, iOS, Windows, MacOS, Linux та Web.

Проаналізувавши та порівнявши такі додатки, як Strong, JeFit, Workout, розглянувши їх переваги та недоліки, екрани редагування плану тренувань, екрани планування тренувань та екрани тренувань кожного з них, я прийшов до розуміння концепту власного додатку та розробив власний дизайн інтерфейсу свого додатку на основі висновків, які я зробив під час аналізу інтерфейсу кожного додатку.

Проаналізувавши існуючі рішення архітектур мобільного додатку, такі як BLoC, Redux та MobX, я прийшов до висновку, MobX є ефективною архітектурою для роботи зі станом мобільного додатка, в той же час вимагаючи мінімальну кількость коду. Через ці якості він був обраний мною для даного проекту дипломної роботи.

Оперуючі данними, які я отримав під час глибоко аналізу та дослідження сучасних фреймворків, аналізу додатків-конкурентів та існуючих рішень архітектур мобільних додатків, я розробив додаток на основі фреймворку Flutter та архітектури MobX на тематику спортивних тренувань.

ПЕРЕЛІК ПОСИЛАНЬ

1 Android Runtime (ART) и Dalvik [Електронний ресурс] URL:

https://source.android.com/devices/tech/dalvik

2 React Native: Bringing modern web techniques to mobile - Facebook Engineering [Електронний ресурс] URL:

https://engineering.fb.com/2015/03/26/android/react–native–bringing–modern–web–techniques–to–mobile/

3 React Native Internals [Електронний ресурс] URL:

https://www.reactnative.guide/3–react–native–internals/3.1–react–native–internals.html

4 What is Apache Cordova? [Електронний ресурс] URL:
https://ionic.io/resources/articles/what-is-apache-cordova

5 v0.0.6: Rev alpha branch version to 0.0.6, flutter 0.0.26 (#10010) [Електронний ресурс] URL:

https://github.com/flutter/flutter/releases/tag/v0.0.6

6 Flutter 1.0: Google’s Portable UI Toolkit [Електронний ресурс] URL:

https://developers.googleblog.com/2018/12/flutter-10-googles-portable-ui-toolkit.html

7 MobX [Електронний ресурс] URL:

https://mobx.js.org/README.html

8 Deriving information with computeds [Електронний ресурс] URL:

https://mobx.js.org/computeds.html

9 Running side effects with reactions { } [Електронний ресурс] URL:

https://mobx.js.org/reactions.html

10 Redux Essentials, Part 1: Redux Overview and Concepts# [Електронний ресурс] URL:

https://redux.js.org/tutorials/essentials/part-1-overview-concepts

11 Architecture [Електронний ресурс] URL:

https://bloclibrary.dev/#/architecture

12 Flutter architectural overview [Електронний ресурс] URL:
https://flutter.dev/docs/resources/architectural-overview

13 Benefits of native ios app development [Електронний ресурс] URL:

https://light-it.net/blog/benefits-of-native-i-os-app-development/

14 Android and iOS [Електронний ресурс] URL:

https://home.ubalt.edu/abento/315/android-ios/index.html

15 What is the Android Operating System? [Електронний ресурс] URL:

https://www.ictea.com/cs/index.php?rp=%2Fknowledgebase%2F8974%2FiQue-es-el-Sistema-Operativo-Android.html&language=english

16 Kotlin is now Google's preferred language for Android app development [Електронний ресурс] URL: https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/

17 Build Your First Android App in Java [Електронний ресурс] URL: https://developer.android.com/codelabs/build-your-first-android-app#0

18 SDK Platform Tools release notes [Електронний ресурс] URL: https://developer.android.com/studio/releases/platform-tools

Додаток А

Фрагмент коду екрану редагування плану тренувань:

```
Container(
  child: ReorderableList(
    onReorder: this._reorderCallback,
    child: Observer(
      builder: (_) => ListView.builder(
        controller: controller,
        itemBuilder: (context, index) => _programsStore
                .routines[index].isEmpty
            ? EmptyDay(
                () =>
                    goToRoute(_programsStore.routines[index]),
                _programsStore.routines[index])
            : ProgramItem(
                onTap: () =>
                    goToRoute(_programsStore.routines[index]),
                item: _programsStore.routines[index]),
        itemCount: _programsStore.routines.length,
      ),
    ),
  ),
)
```

Додаток Б

Фрагмент коду екрану планування тренувань:

```
ReorderableList(
  onReorder: this._reorderCallback,
  child: CustomScrollView(
    controller: controller,
    slivers: <Widget>[
      CupertinoSliverNavigationBar(
        previousPageTitle: "Програма",
        backgroundColor: Colors.white,
        border: null,
        largeTitle: Row(
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            Flexible(
              child: Observer(
                builder: (_) => AutoSizeText(
                  this._routineStore.routine.name.replaceAll("",
"\u{200B}"),
                  minFontSize: 25,
                  softWrap: false,
                  overflow: TextOverflow.ellipsis,
                  maxLines: 1,
                ),
              ),
            ),
            EditButton(
                ()=> openEditModal()
            )
          ],
        ),
```

```
      ),
      SliverToBoxAdapter(
        child: Container(
          margin: EdgeInsets.only(left: 10, top: 8, bottom: 18),
          child: Row(
            children: [
              EditExerciseStatsStrip()
            ],
          ),
        ),
      ),
      SliverPadding(
        padding: EdgeInsets.only(
            bottom: MediaQuery.of(context).padding.bottom),
        sliver: SliverAnimatedList(
          key: animatedListKey,
          initialItemCount: _routineStore.routineExercises.length,
          itemBuilder:
              (BuildContext context, int index, Animation animation) {
            print(_routineStore.routineExercises.length);
            return ReorderableListItem(
              onRemove: () => removeItem(
                  context, _routineStore.routineExercises[index]),
              data: _routineStore.routineExercises[index],
              // first and last attributes affect border drawn during
dragging
              isFirst: index == 0,
              position: index,
              isLast: index == _routineStore.routineExercises.length -
1,
              draggingMode: _draggingMode,
              animation: animation,
            );
```

```
            },
          ),
        ),
      ],
    ),
)
```

Додаток В

Дерево віджетів каруселі вибору вправ:

```
return Observer(
  builder: (_) =>
  new PageView.builder(
    controller: pc,
    physics: _workoutStore.workoutActive ? PageScrollPhysics() :
NeverScrollableScrollPhysics(),
    itemCount: _workoutStore.routineExercises.length + 2,
    onPageChanged: (position) =>
_workoutStore.moveToExercise(position),
    itemBuilder: (context, index) {
      if (_workoutStore.currentRoutine != null) {
        if(index == 0){
          return AnimatedWorkoutOverview(overview, prevOverview,
this.widget.carouselScrollNotifier, this._workoutStore, direction);
        } else if (index == _workoutStore.routineExercises.length+1){
          return ResultCard();
        } else {
          return PlayExerciseCard(_workoutStore.routineExercises[index
- 1]);
        }
      } else {
        return Container(child: Text("Завантаження..."),);
      }
    },
  ),
```

Додаток Г

Дерево віджетів списку підходів вправи:

```
SingleChildScrollView(м
  child: Stack(
    children: [
      CompletedBackground(widget: this.widget),
      Padding(
        padding: EdgeInsets.only(top: 6),
        child: ExerciseSetsList(widget: this.widget),
      ),
      if (!this.widget.exerciseInWork.isCompleted())
        AnimatedPositioned(
          key: ValueKey(this.widget.exerciseInWork.exerciseTimestamp),
          top: this.widget.exerciseInWork.activeSetIndex() * 58.0 - 4,
          duration: Duration(milliseconds: 300),
          child: CurrentSetIndicator(),
        )
    ],
  ),
)
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

ДИПЛОМНА РОБОТА

НА СТУПІНЬ ВИЩОЇ ОСВІТИ БАКАЛАВР

із спеціальності 122 комп'ютерні науки

РОЗРОБКА КРОСС–ПЛАТФОРМНОГО ДОДАТКУ ТИПУ ПЛАНЕР–ТРЕКЕР НА ОСНОВІ ФРЕЙМВОРКУ FLUTTER

Виконав: студент 4 курсу, групи КНД-42 Дубовицький Д.С

Керівник: кандидат технічних наук, доцент Іщеряков С.М.

Київ - 2021

Мета роботи: Розробка кросс–платформного додатку типу планер–трекер

Об'єкт дослідження: технології розробки кросс–платформних мобільних додатків

Предмет дослідження: архітектура та основні елементи крос-платформних технологій, проэктування кросс–платформного мобільного додатку на основі фреймворку "Flutter"

Актуальність роботи

- Зростаючий інтерес до спортивного образу життя
- Підвищення єфективності тренувань
- Доступність на популярних мобільних платформах
- Швидкість розробки та внесення змін
- Вартість розробки

Кросс-платформні технології

- Єдина кодова база
- Швидка розробка
- Низька вартість розробки
- Швидкість розробки та внесення змін
- Зростаюча кількість уніфікованих готових рішень

Огляд існуючих технологій

React Native

+ Використовує елементи системи
+ Популярна мова Javascript
+ Велика кількість створених бібліотек

- Нижча швидкість роботи
- Динамічна мова програмування

Cordova

+ Використовує звичайні HTML, CSS, JS
+ Можливо використовувати вже створені вебсайти
+ Велика кількість створених бібліотек

- Повільна швидкість роботи
- Неможливість використовувати елементи системи

Flutter

+ Нативний двигун рендеру SKIA Велика швидкість роботи
+ Типізована мова програмування
+ Велика кількість підтримуємих платформ

- Неможливість використовувати елементи системи

Технологія Flutter

Під час розробки використовує Dart VM та Just In Time компіляцію, що дозволяє змінювати код додатку без рекомпіляції всього додатку та навіть його перезапуску

Ahead Of Time компіляція в машинний код або LLVM для роботи на пристроях користувачів дозволяє велику швидкість роботи додатку

Типізована мова програмування гарантує вищу стабільність додатку та швидкість розробки

Велика кількість підтримуємих платформ Android, iOS, Linux, Windows, MacOS, Web.

Аналіз Аналогічних додатків

React Native

+ Велика кількість статистики
+ Додаток Apple Watch
+ Темна тема

- Обов’язкова реєстрація
- Відсутність сталої програми тренувань

JeFit

+ Підтримує програми тренувань
+ Додаток Apple Watch
+ Велика бібліотека тренувань

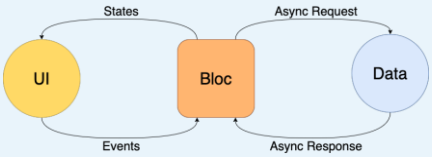
- Обов’язкова реєстрація
- Складний інтерфейс

Workout

+ Підтримує програми тренувань
+ Має персональні програми

- Не має статистики
- Важкий в навігації інтерфейс

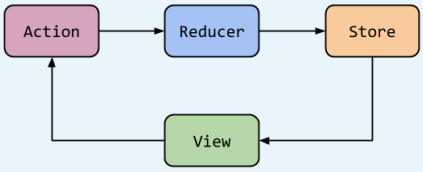
Аналіз Архітектур Flutter



BLoC

+ Велика популярність
+ Просте тестування

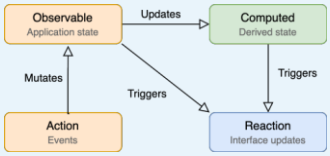
- Складний для сприйняття
- Потребує багато коду



Redux

+ Легкість тестування
+ Легкість пошуку помилок

- Потребує багато файлів та коду
- Погана підтримка асинхронних функцій

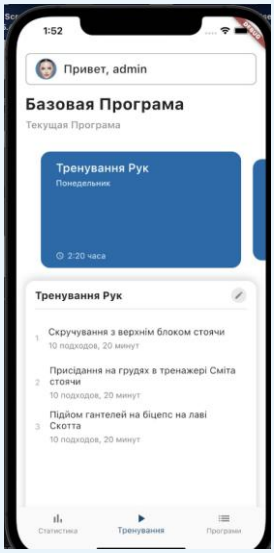


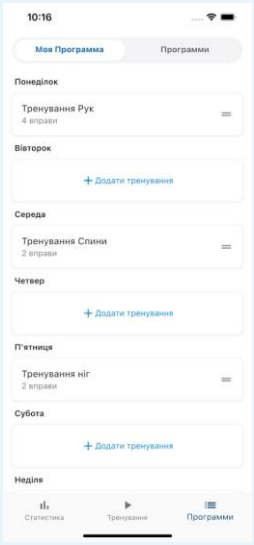
MobX

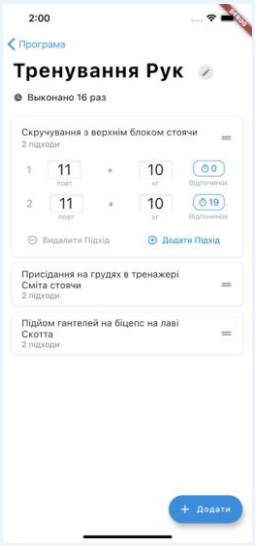
+ Потребує мало коду
+ Просте тестування

- Полягається на кодогенерацію

Розробка дизайну







Розробка додатку

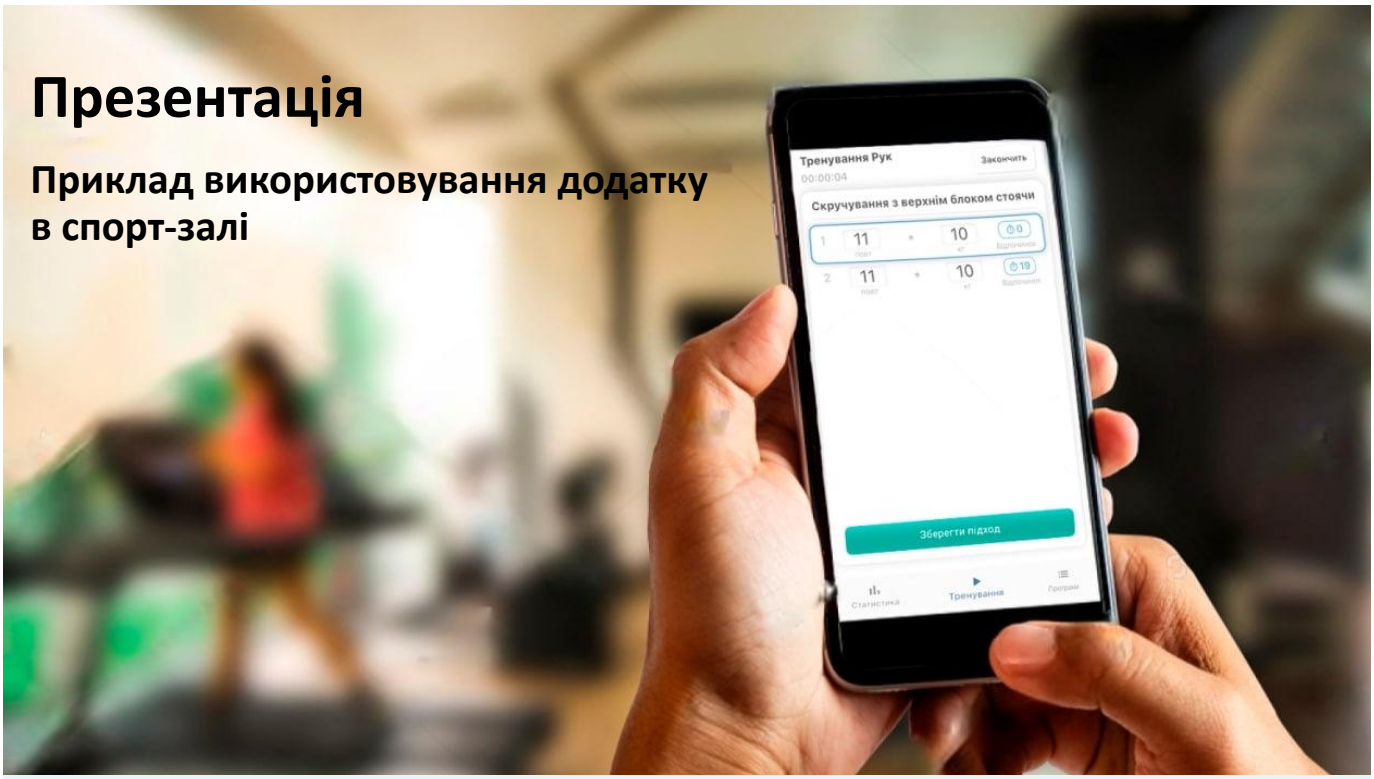
Логіка роботи зі станом програми

Фрагмент дерева віджетів

```
child: ReorderableList(
  onReorder: this._reorderCallback,
  child: Observer(
    builder: (_) => ListView.builder(
      controller: controller,
      itemBuilder: (context, index) => _programsStore
          .routines[index].isEmpty
          ? EmptyDay(
              () =>
                  goToRoute(_programsStore.routines[index]),
              _programsStore.routines[index])
          : ProgramItem(
              onTap: () =>
                  goToRoute(_programsStore.routines[index]),
              item: _programsStore.routines[index]),
      itemCount: _programsStore.routines.length,
    ),
  ),
),
```

Презентація

Приклад використовування додатку в спорт-залі



Висновок

- **Flutter високопродуктивний сучасний фреймворк**
- **DartVM та JIT копіляція задають умови швидкої розробки**
- **MobX архітектура здатна єфективно контролювати стан додатку**
- **У рамках цієй роботи був створений додаток-помічник**