Міністерство освіти і науки України

Національний університет «Києво-Могилянська Академія»

Кафедра мультимедійних систем

КУРСОВА РОБОТА

за спеціальністю «Комп’ютерні науки» 122

на тему: розробка мобільного додатку на основі ОС Android

Науковий керівник:

ст. викладач Борозенний С. О.

Виконав:

студент 3-го курсу Підлісний М. В.

Київ – 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних технологій факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри мультимедійних систем

доцент, к.ф.-м.н

_____О.П. Жежерун

„____”_____2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту _____ факультету _____ ____3-го_____курсу

ТЕМА _____

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

1 Аналіз предметної області

2 Операційна система Android

3 Проєктування архітектури програми

4 Google Maps SDK

5 Розробка програми

Висновки

Список літератури

Додатки (за необхідністю)

Дата видачі „___” _____ 2021 р. Керівник _____

Завдання отримав _____

Календарний план виконання роботи

Тема: Розробка мобільного додатку на основі ОС Android

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу.	02.11.2020	
2.	Огляд технічної літератури за темою роботи.	15.11.2020	
3.	Аналіз архітектури Android-програм	25.11.2020	
3.	Проектування архітектури додатку	Грудень 2020	
4.	Програмування додатку	Лютий-березень 2021	
5.	Написання текстової частини роботи.	10.04.2021	
6.	Створення слайдів для доповіді та написання доповіді.	14.04.2021	
7.	Захист курсової роботи	24.05.2021	

Студент _____

Керівник _____

“_____”_____

ЗМІСТ

Анотація **5**

Вступ **6**

1. Аналіз предметної області **7**

1.1 Характеристика предметної області *7*

1.2 Функціональні вимоги *9*

2. Операційна система Android **10**

2.1 Загальні відомості *10*

2.2 Основна мова програмування *11*

3. Проєктування **13**

3.1 Архітектура програмної системи *13*

3.2 Архітектура додатку *14*

4. Google Maps SDK **18**

4.1 Загальні відомості та обґрунтування використання *18*

4.2 Взаємодія Google Maps SDK з Android-застосунком *18*

5. Структура Android-додатку **20**

5.1 Інтерфейс користувача *20*

5.2 Робота додатку на задньому плані *28*

Висновки **29**

Список літератури **30**

Додаток А (обов'язковий). Лістінг коду Android-застосунку, що реалізує основні функції з роботою з картами в застосунку **31**

Додаток Б (обов'язковий). Лістінг коду Android-застосунку, що реалізує додавання маршруту користувача на карті **33**

Анотація

Курсова робота присвячена дослідженню проєктування та розробки мобільних застосунків на ОС Android з використанням сучасних стилів, бібліотек та технологій. Отриманим результатом роботи є створений на архітектурному шаблоні MVVM та принципами Чистої архітектури мобільний фітнес додаток для відстежування пробіжок та перегляду вправ.

Основні функції системи: відстежування та перегляд пробіжок, перегляд статистики пробіжок та перегляд вправ.

Ключові слова: Android, Kotlin, Kotlin Coroutines, Google Maps SDK, Retrofit2, Room, Dagger2(Hilt), Glide, Moshi, Background Service, WorkManager, MVVM, Architectural Components, Navigation Components, фітнес додаток.

Вступ

З розвитком IT індустрій у світ з’явилися мобільні пристрої, які стали дуже зручним портативним гаджетом полюбленим усіма. Сьогодні складно знайти людину, яка б не мала у себе в кишені смартфон. Більш того, тепер не потрібно мати безліч різних пристроїв заточених лише під один функціонал. Сучасний смартфон слугує користувачу як камера, медіа та аудіо програвач, швидкий інструмент для пошуку різноманітного виду інформації, будильник, калькулятор та багато іншого. Тож не дивно, що зараз ринок смартфонів та їх програмного забезпечення росте та розвивається[1]. Разом з цим розвиваються й операційні системи телефонів, в яких вже тривалий час виступає лідером ОС Android.

В наслідок популярності ОС Android існує безліч різноманітних застосунків, розміщених в магазині Play Market. Проте це не означає, що в точності всі вони ідеально розроблені та бездоганно слугують потребам своїх користувачів. Сьогодні, попри кількість застосунків, доволі складно знайти програму, яка була б використовувала сучасні технології та відповідала усім вимогам користувачів.

Мета цієї роботи полягає в тому, щоб з’ясувати та ознайомитися з найпоширенішими сучасними технологіями, принципами та підходами до розробки мобільного додатку на ОС Android.

У результаті виконання дослідження створено мобільний фітнес застосунок для відстежування пробіжок та перегляду вправ, які завантажуються з власно написаного сервера.

1. Аналіз предметної області

1.1 Характеристика предметної області

Тема цієї курсової роботи – дослідження розробки мобільних додатків на ОС Android. В якості результату роботи було створено фітнес застосунок. Тема застосунку була вибрана не випадково, адже на цей час додатки пов’язані зі здоров’ям та спортом користуються великою популярністю. Ця тема особливо актуальна під час карантину, коли всі спортзали зачинені та люди все більше починають тренуватися в себе вдома чи виходити на невеликі пробіжки для підтримання здорового способу життя та просто щоб подихати свіжим повітрям після тяжкого робочого дня.

На перший погляд, може здаватися, що для простого тренування вдома багато не потрібно, а для пробіжки й взагалі необхідність в додатковому застосунку може бути зовсім малою. Але це не зовсім так. Наприклад в нас з’явиться потреба у відстежуванні нашої статистики тренувань чи навіть просто ми захочемо дізнатися результати завершеної пробіжки: захочемо дізнатися скільки ми витратили калорій, яка була середня швидкість, час та дистанція, яку ми подолали. Ми відразу зіткнемося з проблемою у виборі найкращого підходу для розв'язання цієї задачі.

Перше, що приходить на думку, найпростіший варіант – це просто записувати результати нашого тренування в якийсь записник чи зошит. Але ця ідея далека від ідеалу, бо нам потрібно буде весь час додатково з собою брати зошит, тримати всю інформацію по результатах в голові та більш того майже неможливо порахувати витрачені калорії та середню швидкість, які в багатьох випадках займають одні з ключових цілей й по якім можна відстежувати успіх та розвиток.

На мою думку, мобільний додаток є найкращим вибором для розв'язання цієї задачі, оскільки люди сьогодні мають свій телефон завжди при собі й тому

використання застосунку буде не тільки зручним, а й також не буде вимагати брати додаткових речей.

Реалізований в ході написання цієї роботи застосунок має стати чудовим прикладом для розв'язання цієї проблеми. Користувацький інтерфейс реалізований дружню та не вимагає значних зусиль у пошуках функціоналу та його використанні. Також для зручності користувача додаток підтримує як портретний, так і альбомний режим екрану й зберігає всі дані після повороту. У користувача є можливість відстежувати свою пробіжку, яку він може зупиняти та продовжувати після паузи. Цей функціонал також підтримується при закритому вікні програми. Користувачу не потрібно весь час мати відкритий додаток на телефоні, відстеження може працювати й при звернутій програмі чи взагалі в стані заблокованого телефона. Після початку відстежування пробіжки на панелі сповіщень з’являється зручне вікно з якого можна повернутися в додаток, зупинити відстежування або продовжити його після зупинки. Також є можливість скасувати відстеження якщо воно вже не потрібне та користувач не бажає, щоб пробіжка була збережена. Користувач може переглядати результати по всім пробіжкам у вигляді списку з прокруткою, де він також може видаляти окремі пробіжки. Додатково є можливість переглянути статистику збережених пробіжок у вигляді графіку. Крім цього програма показує тренування на день у вигляді зображень вправ з їх назвою списком з прокруткою, які отримуються з сервера. У користувача є можливість переглянути детальну інформацію вправи та її опис. До того ж тренування кешуються локально на пристрої користувача, що дає можливість переглядати їх без доступу до інтернету. Разом з тим реалізовано функціонал оновлення даних тренування при виключеному телефоні один раз на день коли пристрій під’єднаний до бездротового інтернету та до зарядного пристрою.

1.2 Функціональні вимоги

При розробці застосунку був створений такий перелік функціональних вимог:

a) Підтримка вертикальної та горизонтальної орієнтації екрану
b) Можливість відстежувати пробіжку
c) Можливість зупиняти та продовжувати відстежування
d) Можливість відстежування пробіжки зі згорнутою програмою чи заблокованим телефоном
e) Можливість зупинки та продовження відстежування пробіжки через вікно на панелі сповіщень
f) Можливість переходу до програми з панелі сповіщень
g) Можливість перегляду здійснених пробіжок
h) Можливість видалення пробіжки зі збереженого списку
i) Сортування пробіжок по даті, часу, дистанції, середній швидкості та витраченим калоріям
j) Наявність графіку по збереженим пробіжкам та сумарної статистики
k) Можливість перегляду списку вправ та детальної інформації по кожній
l) Наявність кешування вправ в локальному сховищі пристрою
m) Можливість доступу до програми без підключення до інтернету
n) Оновлення вправ один раз на день при підключенні до бездротового інтернету та зарядного пристрою

2. Операційна система Android

2.1 Загальні відомості

На сьогодні найпопулярнішими мобільними платформами є IOS – операційна система від компанії Apple та Android – операційна система розроблена компанією Google. Оскільки тема даної курсової роботи це розробка для ОС Android, то варто зупинитися на ній більш детально.

Android – це мобільна операційна система розроблена на базі ядра Linux у 2008 році. Ця ОС є повністю безплатною та являє собою систему з відкритим кодом. Android стрімко розвивається, добре підтримується та кожного року компанія Google випускає оновлену версію з розв'язання проблем та запровадженнями нового функціоналу. Разом з ОС Android Google пропонує зручне використання їх сервісів під назвою Google Play Services, які разом формулюють доволі гнучку екосистему. До того ж існує різноманіття операційних систем, які були розроблені беручи за основу ОС Android. Тому насамперед цим ОС Android завдячує свою популярність на ринку[2] та таким чином на момент дослідження має близько 3 мільйонів додатків у магазині Google Play Market[3]. Проте не всі операційні системи реалізовані на базі Android є тільки орієнтованими на смартфони та планшети, багато з них підтримують безліч пристроїв: від розумного холодильна до бортового комп’ютера в машині. Найпопулярнішими реалізаціями є Android TV – платформа для “розумних” телевізорів та Watch OS – операційна система для “розумних” годинників.

Для мобільних розробників додатків для ОС Android на відмінну від багатьох інших операційних систем надається велика свобода дій у створенні програм. Це обумовлено насамперед відкритістю коду системи та як наслідок широкою підтримкою бібліотек та різноманітних плагінів від розробників третіх сторін.

2.2 Основна мова програмування

Напевно одні з перших питань які виникають у розробників, які не знайомі з мобільною розробкою, але мають потребу написати мобільний застосунок для ОС Android – це яку мову та середовище розробки варто вибрати. Й тут насамперед варто розглянути дві основні мови програмування: Java та Kotlin.

Java – це об'єктно-орієнтована мова програмування, власником якої є компанія Oracle. Написаний код на Java компілюється в байт код, що дозволяє мові бути платформо незалежною, що слугує однією з багатьох переваг мови. Мова Java вважається досить легкою для розуміння, проте й іноді може викликати певні труднощі у початківців у її опануванні. До нещодавно Java була мовою номер один в розробці мобільних застосунків для ОС Android. При чому, необхідно зазначити, що Java використовувалася для побудови ОС Android, зокрема інтерфейс та більшість стандартних бібліотек написані мовою Java. Попри те, що Google оголосила Java мовою номер два для розробки для ОС Android, на цей час існує велика кількість різноманітних бібліотек, плагінів та готових рішень написаних мовою Java для мобільних додатків. Разом з цим існує спільнота розробників колосального розміру, яка вважається однієї з найбільших серед усіх інших мов. Це передбачено широким використанням мови Java не тільки у сфері мобільних застосунків, а й далеко поза межами її.

Kotlin – це мультипарадигмена мова програмування, розроблена компанією JetBrains у 2011 році. Ця мова працює поверх віртуальної машини Java й наслідком чого код, написаний на Kotlin так само як і з мовою Java компілюється в байт код. Ця особливість надає мові її головну перевагу у використані. Ця перевага закладається в тому, що оскільки мови Java та Kotlin обидві компілюються в однаковий байт код, то це дає змогу використовувати ці дві мови одночасно в одному проєкт викликаючи Java код як з Kotlin файлу,

так і навпаки. У такий спосіб всі старі бібліотеки та частини готових застосунків написані з використанням мови Java підтримуються та можуть бути застосовані без потреби переписування при створені програми за допомогою мови Kotlin. Також мова є лаконічною з великою кількістю різноманітного додаткового функціонала та вважається доволі легкою для початківців. Таким чином, беручи до уваги всі особливості та переваги мови Kotlin, Google оголосила мову Kotlin номер один для розробки мобільних додатків для ОС Android у 2019 році[4]. Разом з тим необхідно зазначити, що нові версії стандартних бібліотек ОС Android розробляються мовою Kotlin.

Основні переваги мови Kotlin над мовою Java:

- Підтримка механізму Null Safety
- Можливості підтримки функціонального програмування
- Написаний код є більш компактний та значно легший для читання
- Ширша стандартна бібліотека
- Механізм делегатів
- Механізм “розширюючих” функцій
- Механізм “взаємодії” з мовою Java

Беручи до уваги всі ці переваги мови Kotlin було вирішено вибрати саме її в якості основної мови для створення мобільного застосунку.

3. Проєктування

3.1 Архітектура програмної системи

Значна частина з відведеного на дослідження часу, була витрачена на проєктування архітектури та зокрема на пошук найкращої, а точніше тієї, яка буде найбільше підходити для конкретної задачі. Адже етап проєктування є одним з найважливіших етапів життєвого циклу створення застосунків та вимагає уважного продумування всіх дрібниць. Після детально проробленого аналізу функціональних вимог, було визначено, що програмна система буде складатися з трьох елементів (рисунок 3.1.2). Оскільки в додатку буде можливість переглядати різні тренування, які система буде пропонувати кожного дня, то буде створена зовнішня база даних в якій, безпосередньо будуть зберігатися тренування. Для взаємодії додатка та зовнішньої бази даних буде створено окремий веб-сервер у вигляді API, який буде вилучати данні з бази даних та відправляти їх до застосунку через HTTP протокол використовуючи архітектурний стиль REST. Разом з тим, оскільки в додатку буде реалізовано функціонал відстежування пробіжок, то для здобуття інформації про данні місцезнаходження користувача та для можливості демонстрації руху на карті буде використано Google Maps SDK.

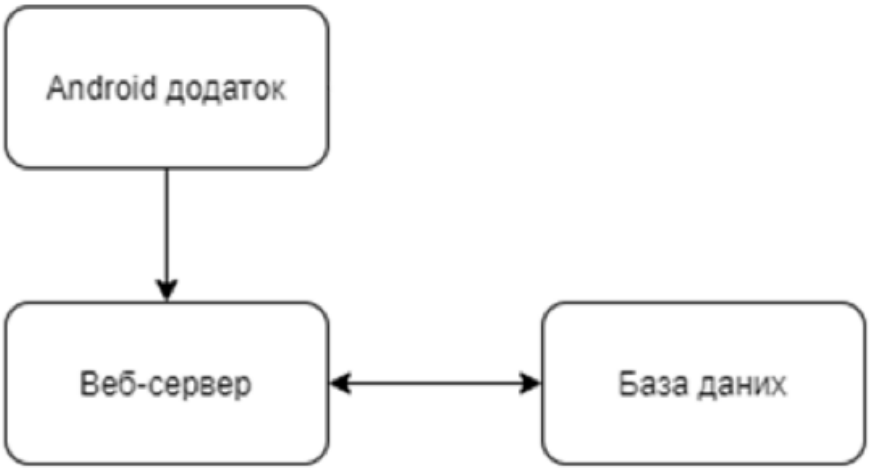

Рисунок 3.1.2 – Загальна схема архітектури програмної системи

3.2 Архітектура додатку

Після ретельного дослідження було визначено, що для розробки архітектури буде використано шаблон проєктування архітектури. Шаблон проєктування архітектури містить визначені правила та опис зв’язків між елементами програми. Зручність використання шаблону полягає в його гнучкості та більш того, шаблон не вважається закінченим набором правил, існує можливість легко розширювати його спираючись на конкретну задачу.

На момент дослідження найпопулярнішим шаблоном для розробки програм для ОС Android є шаблон проєктування архітектури MVVM (Model-View-ViewModel). MVVM ідеально підходить для розробки додатків з графічним інтерфейсом, адже він полегшує його відокремлення від бізнес-логіки, наслідком чого під час розробки можна легко абстрагуватися розробляючи лише один компонент без потреби змінювати інші. Даний шаблон визначає три основних компонентів: View, ViewModel та Model (рисунок 3.2.1).

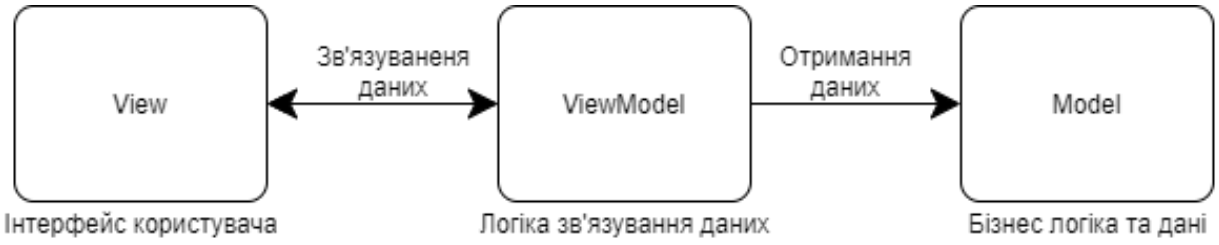

Рисунок 3.2.1 – Схема структури шаблону MVVM

View – як і в багатьох схожих шаблонах проектування архітектури застосунків, відповідає за графічний інтерфейс та також отримує та передає взаємодію користувача з додатком до ViewModel й як наслідок відображає дані, які були отримані з ViewModel. Компонент View також часто можна побачити під назвою UI Controller. Model також має ідентичний функціонал як в схожих шаблонах проектування, наприклад в MVC або MVP. Основна задача

Model – це робота з даними застосунку, оновлення їх та передача у ViewModel. Й остання частина архітектури це ViewModel. Як вже було вище зазначено в компоненті ViewModel передаються дані з Model та конвертуються в значення, які будуть передані в View для їх відображення, тобто ViewModel слугує таким наче “мостом” між Model та View. Разом з тим завдяки ViewModel в MVVM використовується механізм зв’язування даних, це означає що на кожну взаємодію користувача, на введення даних, натискання кнопки, тощо, буде викликана певна функція та використана певна логіка з ViewModel. У такий спосіб завдяки розбиванню архітектури на ці три компоненти, які мають різні обов’язки, буде використано принцип “Розділення відповідальності”. У такий спосіб, є можливість легшого тестування застосунку та як вже було зазначено легшого механізму додавання нового функціоналу та масштабування програми.

Вище була зазначена загальна структура шаблону проектування архітектури MVVM, проте певні механізми роботи мобільних додатків вимагають додаткових елементів архітектури для кращої розробки без потреби дублювання коду та пошуку обхідних шляхів для налагодження процесу, для прикладу для підтримки механізму повороту екрану. З цієї причини, під час розробки архітектури для мобільних додатків на ОС Android рекомендується використовувати розширений шаблон MVVM, створений та опублікований у відкритому доступі компанією Google[5] (рисунок 3.2.2). На ньому визначені зв’язки та залежності між елементами програми.

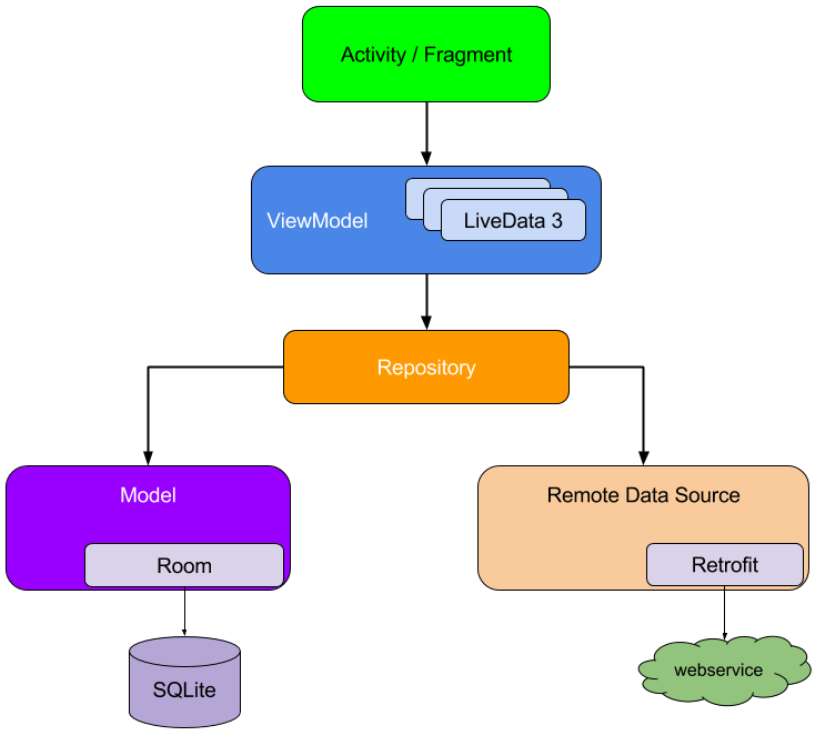

Рисунок 3.2.2 – Схема розширеної структури MVVM компанії Google

Як і в загальній структурі шаблону MVVM, розширений шаблон містить компоненти Model та ViewModel, а View представлений у вигляді Activity чи Fragment класів. Також як окрема частина застосунку виділений зовнішній ресурс даних, як правило це дані які отримуються за допомогою HTTP протоколу з серверу чи стороннього API. Разом з тим використано шаблон проектування Repository. Так як в шаблоні присутні два ресурси даних, Repository допомагає абстрагуватися від шару даних та використовується як єдиний централізований ресурс даних. В додатку також будуть використані класи LiveData, які є класами для зберігання даних з механізмом шаблону Observer. На класи LiveData можна “підписуватися” та стежити за змінами даних під час життєвого циклу додатку. Така особливість класів LiveData вирішує проблему з втратою даних після повороту екрану пристрою, адже тоді вікно програми має створитися по-новому.

Здійснений аналіз розширеного шаблону проектування MVVM, дає змогу дійти до висновку, що використання даного шаблону має низку переваг та у результаті буде застосований для формування архітектури програми. Проте, так як визначено, що застосунок буде підтримувати декілька різних екранів, варто також розглянути архітектуру їх формування. У зв'язку з специфікою розробки мобільних застосунків існує безліч підходів до вирішення цієї задачі, але найпоширенішими є побудова всіх екранів інтерфейсу користувача за допомогою створення активності (англ. Activity) для кожного екрану (Multi-Activity Architecture) та створення єдиної активності у вигляді єдиного вікна програми та фрагментів (англ. Fragment), які будуть слугувати наповненням його (Single-Activity Architecture). Використання Single-Activity Architecture має низку переваг[6], а саме гнучкість у використанні, легший механізм передачі даних та зручніший механізм навігації за допомогою бібліотеки Navigation, представленою компанією Google. На рисунку 3.2.3 можна побачити кінцеву схему архітектури застосунку.

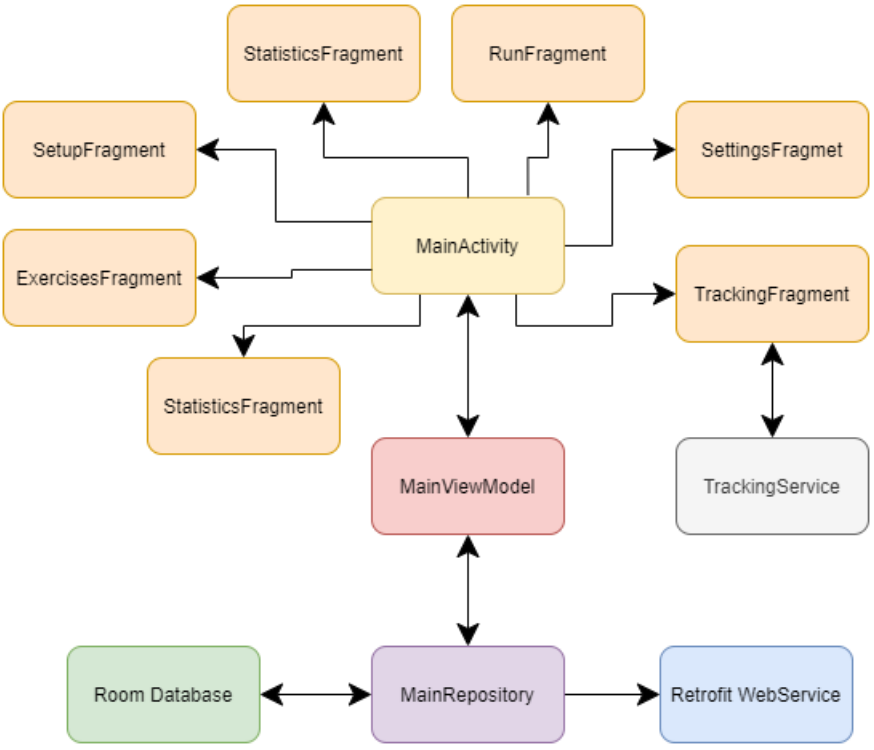

Рисунок 3.2.3 – Схема архітектури додатку

4. Google Maps SDK

4.1 Загальні відомості та обґрунтування використання

У зв’язку з тим, що однією з основних функціональних можливостей створеного застосунку є відстежування пробіжки, то не можна обійтися без можливості візуального відображення як процесу, так і результату тренування. Для вирішення даної задачі було прийнято рішення використати карти на яких буде відображатися здійснений маршрут пробіжки. Після детального дослідження можливих інструментів для інтеграції карти в застосунок, було вибрано Google Maps SDK.

Google Maps SDK – це набір із засобів розробки, який дозволяє легко інтегрувати сервіс “Google Карти” до програми. На момент написання цієї роботи існують окремі версії як для ОС Android так і для ОС iOS. Google Maps SDK є частиною продукту Google Maps Platform API, є найпоширенішим інструментом в своїй категорії та вважається одним з найкращих зі всіх схожих[7]. Використовуючи Google Maps SDK у розробника доступний широкий спектр можливостей для організування карт у своєму додатку, зокрема можливість додавання ламаних ліній та відображення їх на карті.

4.2 Взаємодія Google Maps SDK з Android-застосунком

Для інтеграції Google Maps SDK до існуючого застосунку на ОС Android потрібно виконати наступні кроки:

a) Додати gradle залежності
b) Отримати ключ від API
c) Додати мета-дату та дозволи на використання розташування пристрою до маніфесту застосунку
d) Додати карти в класі фрагменту чи активності

Розглянемо пункт d більш детально. Відображення карт в застосунку реалізовується за допомогою елемента MapView. Після додавання цього елемента в XML код фрагмента, потрібно визначити та ініціалізувати клас GoogleMap. Наведений код в додатках А та Б детально демонструє основні функції, які було використано для роботи з картами. У додатку А висвітлено такі функції як масштабування камери карти, рух карти по координатам пристрою користувача, ініціалізація карт, збереження скріншоту пробіжки. Разом з тим у додатку Б продемонстровано дві функції для додавання ламаних ліній на карті, які зрештою формують маршрут пробіжки.

5. Структура Android-додатку

5.1 Інтерфейс користувача

Як вже було зазначено існує мільйони різноманітних додатків й дуже часто на одну тему можна знайти безліч схожих застосунків, але не викликає жодних сумнів, що одні є більш популярні, інші менш, а деякі взагалі майже невідомі. Як правило, в популярності застосунку одну з найважливіших ролей відіграє унікальний інтерфейс користувача. Часто можна побачити ситуацію, коли в двох додатків з ідентичним функціоналом сильно відрізняється їх популярність й здебільшого це обумовлено саме різницею в дизайні інтерфейсів застосунків. У результаті гарний та зручний дизайн додатку, який був добре продуманий беручи до уваги всі потреби користувачів є причиною для інсталяції й в подальшому бажані ще не раз відкривати його.

Таким чином, тримаючи в голові важливість зручності та простоти дизайну й був створений інтерфейс користувача для фітнес додатку. Під час роботи над дизайном були використані переважно стандартні елементи інтерфейсу Android та елементи інтерфейсу з бібліотеки Material Design.

При першому запуску програми користувача вітає вікно з запитом на надання доступу до відстежування місця розташування пристрою та його використання (рисунок 5.1.1).

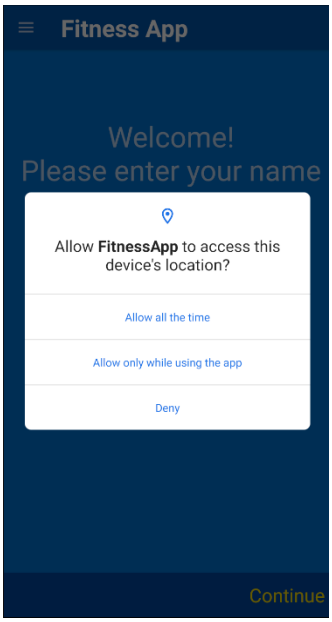

Рисунок 5.1.1 – Екран з запитом доступу до розташування пристрою

Після дозволу на отримання інформації щодо місцеположення пристрою користувач побачить вікно початкового налаштування (рисунок 5.1.2). В ньому користувач повинен ввести своє ім’я, вагу та натиснути на кнопку “Continue” для переходу на головний екран.

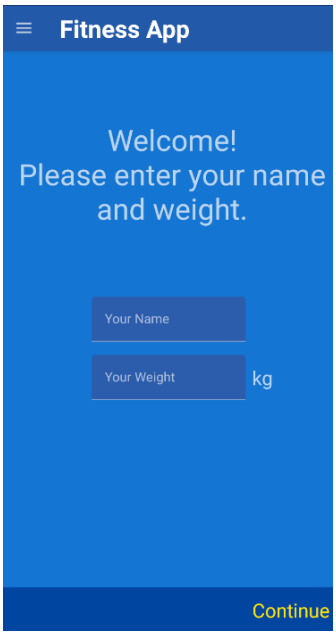

Рисунок 5.1.2 – Екран початкового налаштування

Головний екран – це екран перегляду пробіжок (рисунок 5.1.3). На ньому розташований список всіх здійснених збережених пробіжок у вигляді вертикального списку. Список пробіжок зроблений за допомогою компонента RecyclerView та кожний компонент списку складається із збереженого зображення маршруту відстеженої пробіжки та її даних: дата, час, дистанція, середня швидкість та витрачені калорії. Разом з тим є можливість видалення пробіжки за допомогою “довгого” натискання на елемент списку. Також присутній механізм сортування по всім даним пробіжки. По замовченню всі збереженні пробіжки в списку сортуються по їх даті, а для сортування по іншим даним потрібно вибрати бажаний критерій в випадаючому списку. Разом з тим на екрані зображена кнопка для початку відстежування пробіжки. Ця кнопка оформлена у вигляді Float button з бібліотеки Material design.

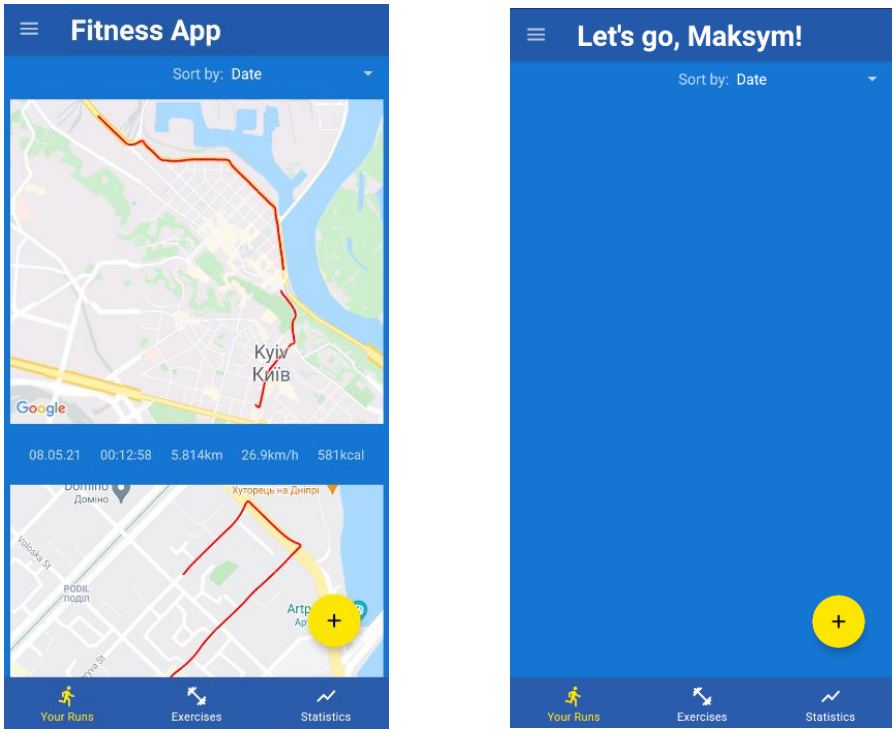

Рисунок 5.1.3 – Екран перегляду пробіжок (з та без пробіжок)

Додатково на вікнах програми зображено елемент BottomNavigationView та елемент DrawerView (рисунок 5.1.4) для легшої навігації по застосунку.

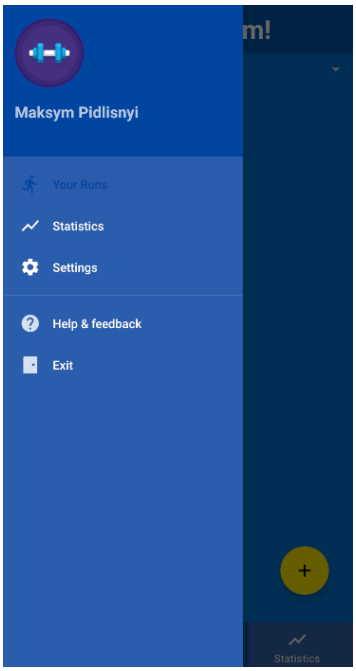

Рисунок 5.1.4 – Екран DrawerView

Після натискання на кнопку “розпочати пробіжку”, відкривається вікно відстежування пробіжки (рисунок 5.1.5). На ньому зображена карта, секундомір та кнопка розпочати або якщо відстежування вже розпочалося, то зупинити відстежування.

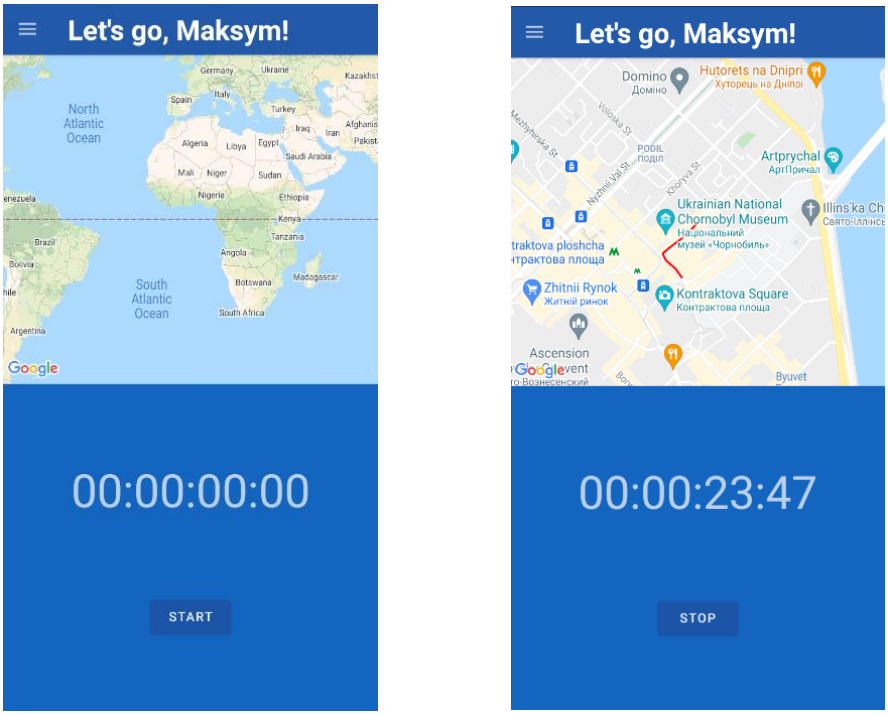

Рисунок 5.1.5 – Екран відстежування пробіжки

Після натискання на кнопку “Start” розпочинається відстежування, кнопка “Start” зникає, а на її місці з’являється кнопка “Stop”, секундомір починає йти та вигляд карти масштабується до розміру зручного для відображення маршруту користувача. Додатково вигляд карти рухається плавно з рухом користувача та на карті відображається червона лінія маршруту, яка будується за допомогою координат пристрою. Після зупинки відстежування з’являються кнопки “Resume” та “Finish run” й додатково наявний механізм скасування пробіжки при натисканні на кнопку “X” в верхньому правому кутку (рисунок 5.1.6).

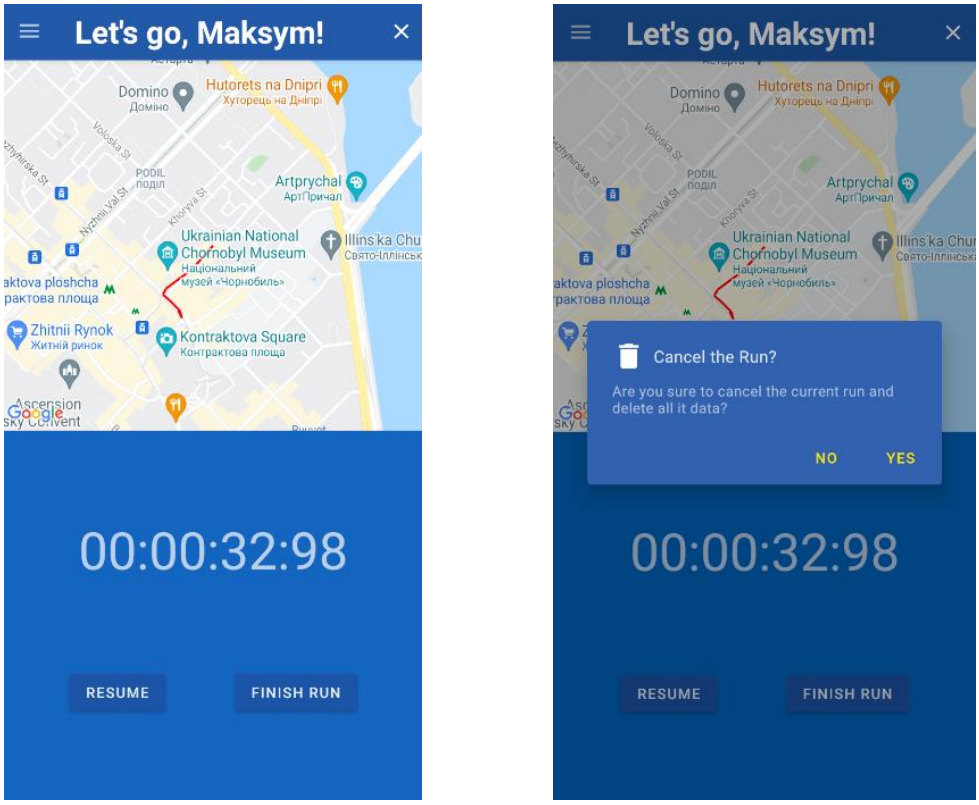

Рисунок 5.1.6 – Екран відстежування пробіжки після зупинки

Натиснувши на кнопку “Resume” відстежування продовжується та лінія маршруту починає відображатися з поточного місцезнаходження пристрою користувача. Разом з тим при натисканні на кнопку “Finish run” дані по пробіжці будуть збережені та екран перегляду пробіжок з оновленим списком буде відкритий.

Варто зазначити, що екран відстежування пробіжки також підтримує горизонтальну орієнтацію пристрою та зберігає всі данні після повороту екрану (рисунок 5.1.7).

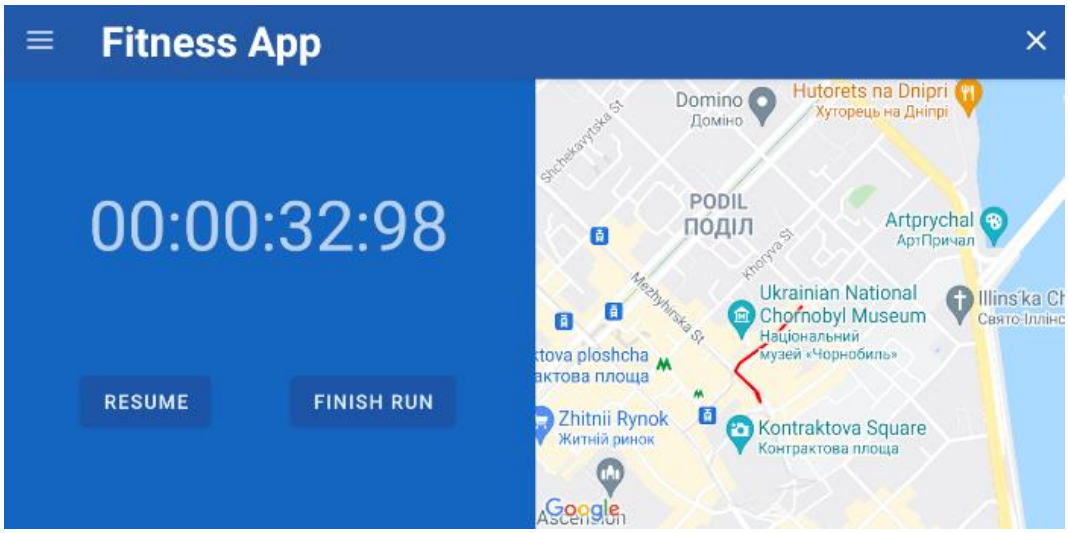

Рисунок 5.1.7 - Екран відстежування у горизонтальній орієнтації

Екран перегляду тренування (рисунок 5.1.8) реалізований у подібній структурі до екрану перегляду пробіжок. Цей екран містить вертикальний список вправ з можливістю прокрутки та заголовком секції. Відображення списку вправ реалізовано за допомогою елемента RecyclerView. Кожний елемент списку складається з фотографії та назви вправи. Для відображення та оптимізації зображень використано бібліотеку Glide. Разом з тим реалізований механізм показу анімованого зображення завантаження для фотографій при ситуації коли вони ще не були виведені на екран. Всі фотографії та дані по вправам були взятті з бібліотеки вправ веб-сайту ACE Fitness[8] з відкритим доступом.

Рисунок 5.1.8 – Екран перегляду тренування

При натиску на зображення вправи буде відкривати нове вікно з детальною інформацією (рисунок 5.1.9). На цьому екрані зображена назва вправи, група м’язів на які вона орієнтована, необхідне обладнання, складність виконання вправи, два зображення та опис вправи.

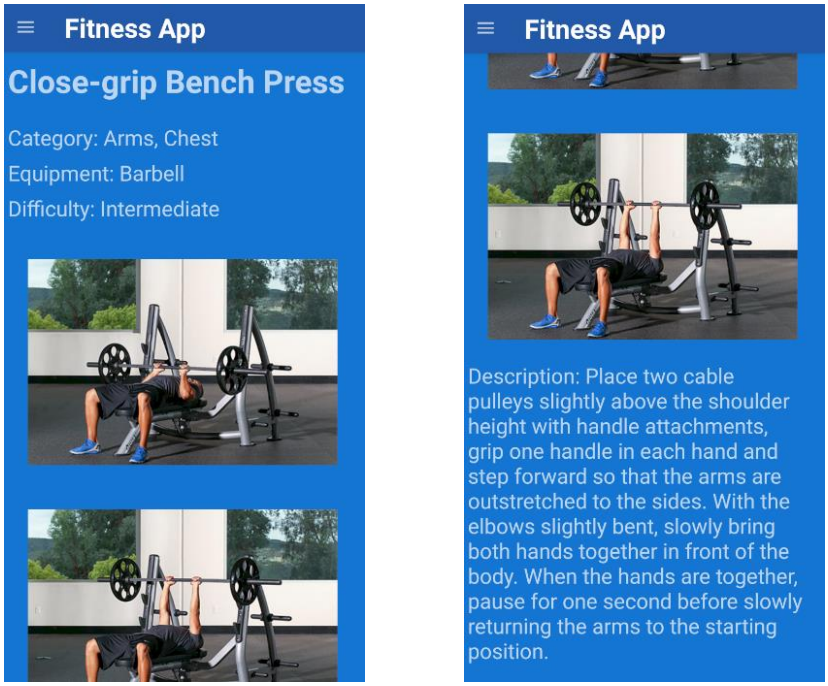

Рисунок 5.1.9 – Екран з детальною інформацією по вправі

Екран перегляду статистики по пробіжкам розділений на дві умовні частини. Перша частина відображає сумарні данні по збереженим пробіжкам, а друга відображає графік відповідно їх середньої швидкості, який реалізований за допомогою елемента BarChart. При натиску відображається повна інформація по пробіжці

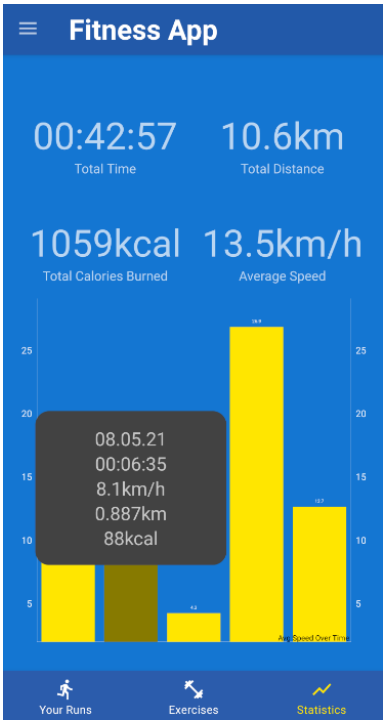

Рисунок 5.1.10 – Екран перегляду статистики по пробіжкам

Екран статистики також підтримує горизонтальну орієнтацію (рисунок 5.1.11).

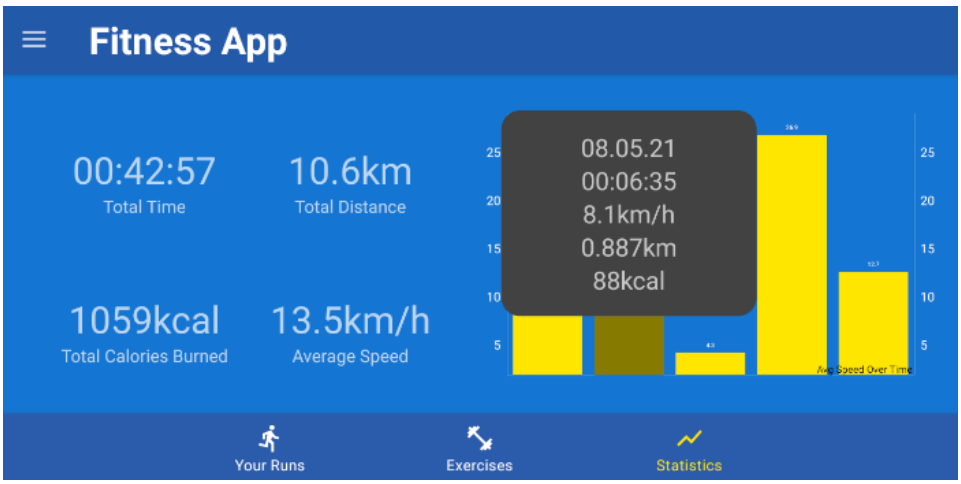

Рисунок 5.1.11 – Екран статистики в горизонтальній орієнтації

5.2 Робота додатку на задньому плані

Робота додатку на задньому плані рівнозначно важлива як і його зручний інтерфейс. Хоча й користувач не бачить та може не знати, які механізми виконуються поза межами вікна його застосунку, результати роботи цих механізмів однозначно покращують роботу з ним. Таким чином під час дослідження було розглянуто два найпоширеніші методи роботи додатку на задньому плані використовуючи сервіс[9] та бібліотеки WorkManager[10].

Сервіс – це один із компонентів програми, який виконує роль певного процесу операцій у фоновому режимі. Розрізняють три види сервісів: переднього плану (коли операції сервісу помітні користувачу), заднього плану (операції не помітні) та прив’язаного типу (сервіс зв’язаний з іншою частиною програми). В застосунку за допомогою сервісу було реалізовано: секундомір та процес оновлювання даних місця знаходження пристрою. Разом з тим, використовуючи сервіс переднього плану було створено сповіщення в панелі повідомлень для відстеження пробіжки та відображення по ній інформації зі згорнутою програмою чи заблокованим пристроєм. В додатку В продемонстровано основні функції сервісу, які були використані в додатку.

WorkManager – це бібліотека яка використовується для реалізування відкладених операцій у майбутньому. WorkManager також є частиной рекомендованого набору бібліотек Jetpack. У бібліотеки WorkManager існує підтримка двох можливих процесів виконання: одноразового та періодичного. Також можливо створювати певну чергу на виконання та процеси будуть виконані після того як затверджені обмеження будуть дотриманні. Використовуючи бібліотеку WorkManager було реалізовано оновлення даних вправ в сховищі для кешування їх на пристрої. Оновлення відбується один раз на день, коли смартфон під’єднано до безпровідного інтернету та зарядного пристрою.

Висновки

В результаті виконаного дослідження було розглянуто найпоширеніші сучасні технології, принципи й підходи до проектування та розробки мобільних додатків на ОС Android.

Разом з тим було створено фітнес додаток для відстежування пробіжок та перегляду тренувань. В створеному додатку було реалізовано всі функціональні вимоги, які були поставлені на початку дослідження та додатково було створено веб-сервер на мові JavaScript, який виконував роль API для застосунку, завантажуючи вправи з сервісу бази даних MongoDB Atlas. Також варто зазначити, що сервер був завантажений на сервіс Heroku.

Додаток був розроблений з думкою про його зручність та можливі сценарії використання користувачем. Таким чином було реалізовано кешування вправ на пристрої, що дає можливість переглядати їх без доступу до інтернету й оновлювати їх один раз на день. Разом з тим було реалізовано можливість швидкого керування та перегляду відстежування пробіжки без потреби мати застосунок у відкритому вигляді.

Як можливе покращення системи є реалізація механізму створення індивідуальних тренувань та розширення їх видів.

Список літератури

1. Smartphones – Statistics & Facts [Електронний ресурс]. – 2021 – https://www.statista.com/topics/840/smartphones/#dossierSummary
2. Mobile Operating System Market Share Worldwide [Електронний ресурс]. Режим доступу: https://gs.statcounter.com/os-market-share/mobile/worldwide
3. Number of Android apps on Google Play [Електронний ресурс]. Режим доступу: https://www.appbrain.com/stats/number-of-android-apps
4. Android's Kotlin-first approach [Електронний ресурс]. Режим доступу: https://developer.android.com/kotlin/first
5. Recommended app architecture [Електронний ресурс]. Режим доступу: https://developer.android.com/jetpack/guide#recommended-app-arch
6. Reasons to use Android Single-Activity Architecture with Navigation Component [Електронний ресурс]. Режим доступу: https://oozou.com/blog/reasons-to-use-android-single-activity-architecture-with-navigation-component-36
7. Best Map API for Location-Based Services [Електронний ресурс]. Режим доступу: https://www.mobindustry.net/best-map-api-for-location-based-services-mapbox-vs-google-maps-vs-openstreetmap/
8. Exercise Database & Library [Електронний ресурс]. Режим доступу: https://www.acefitness.org/education-and-resources/lifestyle/exercise-library/
9. Services overview [Електронний ресурс]. Режим доступу: https://developer.android.com/guide/components/services
10. WorkManager [Електронний ресурс]. Режим доступу: https://developer.android.com/jetpack/androidx/releases/work

Додаток А (обов’язковий). Лістінг коду Android-застосунку, що реалізує основні функції з роботою з картами в застосунку

```
private fun zoomTrackAndSave() {
   val bounds = LatLngBounds.Builder()
   for (line in points) {
      for (point in line) {
         bounds.include(point)
      }
   }
   val height = binding.map.height
   val width = binding.map.width
   map?.moveCamera(CameraUpdateFactory
         .newLatLngBounds(
               bounds.build(),
               width,
               height,
               (height * 0.05f).toInt()))
}
private fun navigateCameraToUser() {
   if (points.last().isNotEmpty() && points.isNotEmpty()) {

map?.animateCamera(CameraUpdateFactory.newLatLngZoom(points.last().last(),
Constants.MAP_ZOOM))
   }
}
private fun finishRun() {
   map?.snapshot { img ->
      var overallDistance = 0
      for (line in points) {
         overallDistance  += Helper.calculatePolylineLength(line).toInt()
      }
      val timestamp = Calendar.getInstance().timeInMillis
      val caloriesBurned = ((overallDistance * weight) / 1000).toInt()
      val avgSpeed = round((overallDistance / 1000) /
            ((timeInMillis) /10000 / 60 / 60)
      var runEntity =
            Run(img, timestamp, avgSpeed, overallDistance, timeInMillis,
caloriesBurned)
```

```
        if (run.img != null) {
            viewModel.insertRun(run)
            Snackbar.make(
                    requireActivity().findViewById(R.id.rootView),
                    "Run was saved successfully.",
                    Snackbar.LENGTH_SHORT
            ).show()
        }

        stopRun()}
}
binding.mapView.getMapAsync {
    map = it
    addPolylines()
}
```

Додаток Б (обов’язковий). Лістінг коду Android-застосунку, що реалізує додавання маршруту користувача на карті

```
private fun addLastPolyline() {
    if (points.last().size > 1 && points.isNotEmpty()) {
        val lastLatLng = points.last().last()
        var preLastLatLng = points.last()[points.last().size - 2]
        val polylineSpecs = PolylineOptions()
            .width(POLYLINE_WIDTH)
            .color(POLYLINE_COLOR).add(lastLatLng)
            .add(preLastLatLng)

        map?.addPolyline(polylineSpecs) }
}

private fun addPolylines() {
    for (polyline in points) {

        val polylineSpecs = PolylineOptions()
                .width(POLYLINE_WIDTH)
            .color(POLYLINE_COLOR)
                .addAll(polyline)
        map?.addPolyline(polylineSpecs) }
}
```