**ТЕХНОЛОГИЧНО УЧИЛИЩЕ “ЕЛЕКТРОННИ СИСТЕМИ”**

**към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**ДИПЛОМНА РАБОТА**

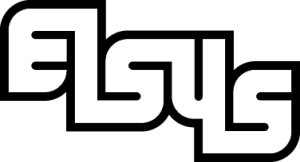
Тема: Android фитнес приложение

Дипломант: Научен ръководител:

*Ивайло Георгиев Иван Игнатов*

СОФИЯ

2020

**TЕХНОЛОГИЧНО УЧИЛИЩЕ “ЕЛЕКТРОННИ СИСТЕМИ”**

**към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

Дата на заданието: 15.11.2019 г. Утвърждавам:..............................

Дата на предаване: 15.02.2020 г. /проф. д-р инж. Т. Василева/

**ЗАДАНИЕ**

**за дипломна работа**

на ученика Ивайло Николаев Георгиев 12 А клас

1.Тема: Мобилно фитнес приложение

2.Изисквания:

2.1 Поддръжка на различни акаунти

2.2 Упражнения с характеристики и история със серии и повторения

2.3 Крачкометър

2.4 База данни с храни

2.5 Задаване на желани резултати и изчисляване на нужните за постигането им калории и макроси, както и предложения за тренировки

2.6 Списък с приятели и feed от техните дейности

2.7 Записване на тегло, упражнения и храни

2.8 Сканиране на баркод на стока

3.Съдържание 3.1 Обзор

3.2 Същинска част

3.3 Приложение

Дипломант :...........................................

Ръководител:..........................................

/ Иван Игнатов/

Директор:................................................

/доц. д-р инж. Ст. Стефанова /

**Отзив на научен ръководител**

Реализирайки в последователни стъпки заданието си, дипломантът успешно създаде Android приложение, отговарящо на съвременните изисквания и тенденции в сферата на Android продуктите. Облягайки се на опита и уменията натрупани по време на учебния процес, дипломантът адекватно решаваше възникващите пред него проблеми, като оформянето на привлекателен, но същевременно удобен потребителски интерфейс и оптимизирането на програмния код, превръщайки го в по-надежден. В процеса на разработка на дипломната си работа проличаха професионалното му любопитство и желание за усвояване на нови умения и технологии, резултат от които е избрал да използва Firebase за базата данни на приложението. Разработвайки дипломната си работа, Ивайло бе систематичен и изпълнителен и макар иновативните технологии, които използваше, не загуби фокус и подходът му си остана практичен. Дипломантът успешно разви уменията, придобити по време на обучението си, които несъмнено ще открият приложение в професионалната сфера.

Бих му препоръчал при проектиране на приложения, да обърне особено внимание на решения за ситуации, от които нормалното функциониране на приложението му зависи. Съвремието ни изисква все по-дълбоко обезпечаване на работата на софтуера, дори в кризисни ситуации.

С оглед на използваните технологии, препоръчвам за рецензент Илиян Спиров.

Ръководител: ..........................................

/ Иван Игнатов/

# Увод

Съвременното ежедневие носи със себе си забързан начин на живот, в който се налага лавиране между семейство, работа, задължения, както и да се усвояват нови умения. Технологичния прогрес предоставя база, с която да се изградат решения за справяне с многобройните задължения. Тук се засяга създаването на фитнес приложение, като необходим помощник в спортните начинания. То ще улесни всекидневието (включително това на спортистите) и ще мотивира онези, които желаят промяна в начина си на живот.

Хората, често правят първите си стъпки с голяма доза несигурност, навлизайки в сферата на спортната дейност. Веднъж, превърнали спорта в неотменна част от ежедневието си, той набира инерция, заемайки все повече от времето и енергията им.

Така се зароди идеята за създаването на мобилно фитнес приложение, което да се превърне в незаменим асистент в следенето на десетките показатели, които всеки спортист, бил той аматьор или професионално занимаващ се – следи. Мобилното приложение ще включва в себе си функции, като:

* Крачкомер;
* Фитнес упражнения – с всички техни движения;
* Изчисляване на прием на калории в зависимост от зададена от потребителя цел.

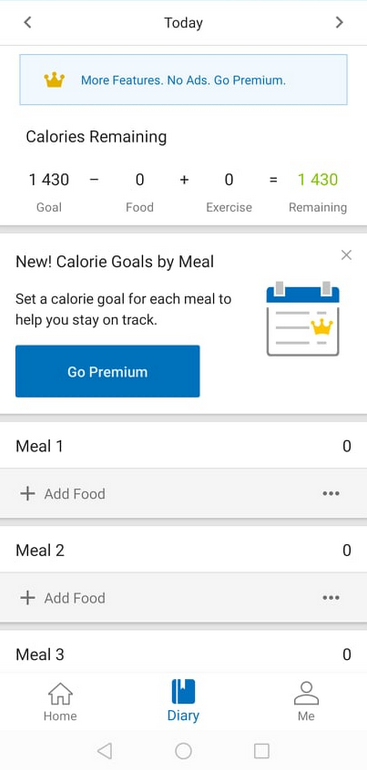
# ПЪРВА ГЛАВА

# Обзор на съществуващите решения. Преглед на технологии подходящи за изпълнение на дипломната работа.

## 1.1. Съществуващи приложения

### 1.1.1. MyFitnessPal

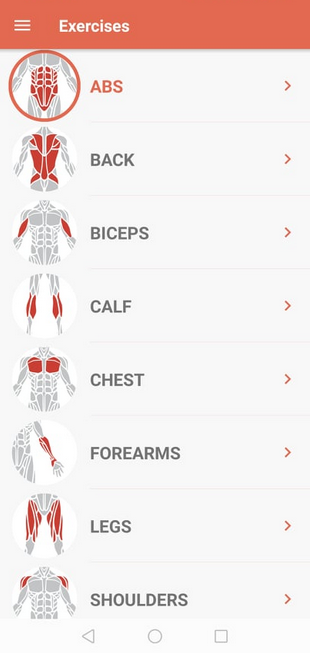
MyFitnessPal е фитнес приложение, с което потребителите да проследяват приема на дневното количество калории (Фигура 1.1). Разработена е както десктоп версия за Windows, така и мобилни версии за Android и IOS. Приложението предоставя интуитивен потребителски интерфейс (UI), позволяващ бързото му усвояване от разнородни възрастови групи. Приложението е подходящо и за тези, които се интересуват от спазването на хранителен режим. Конкурентното предимство на MyFitnessPal са функциите му за изчисляване и следене на прием и разход на калории, но не се изчерпват до там. Той включва и крачкометър, рецепти за диетични хранения (които разработчиците, периодично допълват, редактират или премахват), списък с приятели, както и набор от статии посветени на здравословния начин на живот. Сериозен пропуск в приложението е липсата на опция за създаване на личен режим/програма за тренировки; информационно табло(dashboard), което да визуализира прогреса, както и допълнителна информация, предоставена по привлекателен за потребителя начин (чрез различни динамични фигури и скали).



*Фигура 1.1*

### 1.1.2. Fitness & Bodybuilding

Fitness & Bodybuilding е приложение от групата на MyFitnessPal, но е насочено към техниката на упражненията и създадено, по-скоро за бодибилдинг средите. Ето защо приложението е разработено за Аndroid и IOS и липсва десктоп версия – тъй като се очаква, че потребителя ще го използва по време на тренировка. По същата причина не са развити функции, като калориен калкулатори или съвети за хранителен режим. Особено преимущество на работата с това приложение е улеснения интерфейс, както може да се види на Фигура 1.2, подходящ и за потребители без сериозен опит във фитнес среда. Приложението предлага и набор от препоръчителни тренировки според нивото на ползващия го, като в същото време е напълно безплатна. Освен изчисленията на калории, очевидна е и липсата на крачкометър.



*Фигура 1.2*

## 1.2. Технологии и средства за развой

### 1.2.1. Android Studio

Android Studio е официалната среда за разработка (IDE - Integrated Development Enviroment) за операционната система на Google – Android, изградена на основата на JetBrains' IntelliJ IDEA софтуера. Android Studio включва както основните инструменти за работа под IntelliJ, така и строго специфични елементи и функции, необходими в развоя на Android приложения. Сред тях са:

* Gradle базирана изграждаща система;
* Емулатор, включващ в себе си множество инструменти и функции, работещ изключително бързо.

Android Studio разполага с версии за множество операционни системи - Windows след версия 7, macOS, както и Linux дистрибуции (базирани основно на Debian и Ubuntu). Успешно замества други среди за разработка, като например [Eclipse Android Development Tools](https://en.wikipedia.org/wiki/Eclipse_(software)#_blank) (ADТ), тъй като предлага богат набор от инструменти, засягащи front-end и back-end.[1]

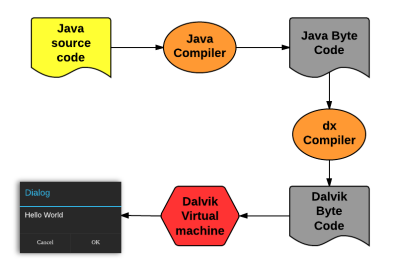
### 1.2.2. IntelliJ IDEA

IntelliJ IDEA е интегрирана среда за разработка (IDE) за компютърен софтуер, създадена от JetBrains. Достъпна както под [Apache 2 Licensed](https://bg.wikipedia.org/w/index.php?title=Apache_2_Licensed&action=edit&redlink=1) community edition, такаи под платен лиценз - commercial edition. И двете версии на средата позволяват разработка на приложения. Платената версия е насочена към професионалните среди и предоставя значително повече инструменти и гъвкавост. Възходът на IntelliJ започва през 2010 година, когато получава най-висок резултат от четирите най-използвани инструменти за програмиране в Java, а именно - [Eclipce](https://bg.wikipedia.org/w/index.php?title=Eclipce&action=edit&redlink=1), IntelliJ IDEA, [NetBeans](https://bg.wikipedia.org/wiki/NetBeans) и [JDeveloper](https://bg.wikipedia.org/w/index.php?title=JDeveloper&action=edit&redlink=1). [Интегрираната среда за разработка](https://bg.wikipedia.org/wiki/Интегрирана_среда_за_разработка) подпомага разработчиците, чрез множество инструменти, подсигуряващи завършеността и кохерентността на програмния код. Някой от инструментите са: анализ на контекста на кода, улеснена навигация в кода, избистрена структура, позволяваща създаване на разнородни структури, като класове и декларации, както и система предлагаща поправки при синтактични грешки или други несъответствия.

### 1.2.3. Basic4Android

Basic4Android е RAD(Rapid Application Development) инструмент, използван далеч по-рядко, тъй като е разработен на основата на Visual Basic, от където и произлиза името на тази среда – Basic For Android.

Средата позволява ненадмината бързина в създаването на програмен код, както и в оформянето на потребителския интерфейс, но бидейки базиран на Visual Basic, не е популярен сред разработчиците от java средите, заради синтактичните разлики между двата езика. Сериозен пропуск в тази среда е липсата на унаследяване на класове, както и рудиментарната форма на класове. Липса, която може да бъде заобиколена, но с цената на сериозни усилия и знания от страна на програмиращия. В заключение, тази среда е подходяща за бързо създаване на Android приложения, без да ограничава потребителя от необходимите му функции, но изисква по-дълбоки общи познания от използващите го. Средата разполага и с визуален дизайнер за създаване на потребителския интерфейс (UI), който обаче има своите ограничения, поради което в професионалните среди за дизайна се използва изцяло текстовата среда на B4A. Любопитен е процеса, в който средата създава кода - езикът, на който се програмира, е подобен на Visual Basic и Visual Basic.Net, но програмата го „превежда“ на java, след което тя на свой ред бива преработена в машинен код за Dalvik Virtual Machine (Фигура 1.3).



*Фигура 1.3 илюстрира комплексния цикъл през, който преминава машинния код, за да бъде обработен от микропроцесора на Android устройството.*

### 1.2.4. Java

Java е [обектно-ориентиран](https://bg.wikipedia.org/wiki/Обектно-ориентирано_програмиране) [език за програмиране](https://bg.wikipedia.org/wiki/Език_за_програмиране). Изходният код, създаден в Java, не се компилира до машинен код за определен [микропроцесор](https://bg.wikipedia.org/wiki/Микропроцесор), а до междинен език - така наречения [байткод](https://bg.wikipedia.org/w/index.php?title=Байткод&action=edit&redlink=1). Байткодът не се изпълнява директно от микропроцесора, а от негов аналог – [виртуален процесор](https://bg.wikipedia.org/wiki/Виртуална_машина), наречен [Java Virtual Machine](https://bg.wikipedia.org/wiki/Java_Virtual_Machine) (JVM). Коренът на тази така усложнена система е архитектурата на мобилните устройства използващи Android ОС. Те използват ARM микропроцесори (Advanced RISC Machine), които използват различен тип инструкции. Употребата на ARM процесори за мобилните устройства е продиктувана основно от по-ниската им енергийна консумация, докато същевременно производителността им е достатъчно голяма. Ето защо, наред с изящната система за енергиен мениджмънт, мобилните устройства минимизират натоварванията върху процесора, чрез различен тип цикъл в живота на програмите, които изпълняват. Ако не се осъществи завръщане към приложението във времевата рамка на Android ОС, жизнения цикъл на приложението се прекратява, освобождавайки ценни ресурси и спестявайки енергия. Тази структура е изключително важна в контекста на програмния език Java, тъй като се използва в JVM, която е известна със затрудненията си да изчиства рам паметта си. Което далеч не превръща Java в неподходящ за разработка на мобилни приложения език - напротив, сред основните му предимства са:

* лесната преносимост между разнородни платформи (софтуер или хардуер);
* освобождаване на паметта от обекти, които не се използват (Garbage collector), което има косвен ефект и върху енергийната консумация;
* първоначално заделяне на граница за консумация на оперативната памет (memory heap);
* Относителна степен на сигурност поради факта, че няма директен достъп до паметта.

### 1.2.5. Kotlin

Kotlin е крос-платформен програмен език за общо предназначение, който е използван основно в Android Studio средата. Още със създаването си е структуриран, предвиждайки тясното взаимодействие с Java и JVM, като в отделни ситуации може да работи в нееднородна среда с Java. Kotlin основно работи с JVM, но не се ограничава до нея и може да се компилира до JavaScript или native код. Най-голяма популярност има в изработването на Android приложения, като към момента е основния език за разработка под Android Studio, измествайки от тази позиция Java, който макар и поддържан, вече е второстепенен език. Според създателите на езика, стремежът им е бил насочен към улесняване на работния процес, без това да е за сметка на функционалността на езика. По този начин се подобрява и ускорява процеса на разработка на мобилни приложения.

### 1.2.6. Firebase

Firebase е платформа, закупена от Google, представляваща инфраструктура за мобилни и web приложения. Инструментите, които предоставя са:

* Инструменти за анализ (по потребители, възрастови групи, културни групи и потребление във времеви отрязъци);
* Бази от данни, достъпни на сървъри на google;
* Cloud сървъри за съхранение на данни;
* Hosting услуги.

# Втора Глава

# Функционални изисквания и проектиране на фитнес приложението.

## 2.1. Функционални изисквания към фитнес приложението.

* Поддръжка на потребителски регистрации (акаунти) и индивидуализиран профил за всеки потребител, включващ име, пол, възраст и физически характеристики на потребителите, като тегло и височина;
* Упражнения с детайлна информация за тях, хронология на практикуваните упражнения, включваща сериите и повторенията, направени от потребителя;
* Крачкометър - използващ вградените в мобилния хардуер сензори и технологии в Android ОС за правилно и прецизно отчитане на информацията;
* База от данни съдържаща в себе си храни и напитки, техни характеристики, като калоричност, макроси и грамаж на разфасовка;
* Задаване на потребителски цели и изчисляване на нужните за постигането им калории и макроси.
* Списък с приятели - създаване на своеобразна социална мрежа, в която ще бъде включена и функция с потребителски дневник (feed/log).
* Проследяване на потребителско тегло, съхраняване на данни за упражнения и храни, които в последствие ще намерят директно приложение в изчисленията.
* Сканиране на баркод на стока - което ще улесни проверка за калорийната стойност на приеманите храни, доколкото храните са записани в базата данни.

## 2.2. Съображения за избор на програмни средства и развойна среда.

### 2.2.1. IDE – Android Studio

Сред най-големите позитиви на Android Studio са богатата документация и широката база от потребители. Средата се комбинира успешно с друг продукт на Google, а именно с Firebase и разполага с богат набор от собствени инструменти, като графичен редактор на xml файлове, система за създване на apk инсталационни файлове и други.

### 2.2.2. Firebase

Firebase е безплатна и лесна за употреба платформа, която предоставя нужните за дипломната работа функции. Разполага с база данни в реално време, инструменти за анализ на потребителите(analytics) и завършена система за вход и изход (authentication). Базата данни пази данните синхронно между потребителските приложения, използвайки обекти, които засичат събития в реално време (event listener).

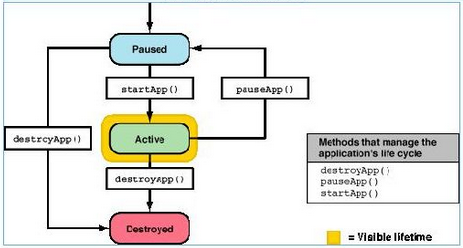
### 2.2.3. Език за програмиране – Java

Основните езици използвани в Android studio са два - Java и Kotlin. Java е по-добре познат на общността около програмирането за мобилни приложения, което обуславя по-улеснения процес по намиране на решения на проблеми в сайтове като github и stackoverflow. От друга страна Kotlin е разработен, като крос език, което означава, че в бъдеще може да се комбинира новия официален език за Android studio с Java.

## 2.3. Принцип на софтуерния модел

### 2.3.1. Структура на приложението

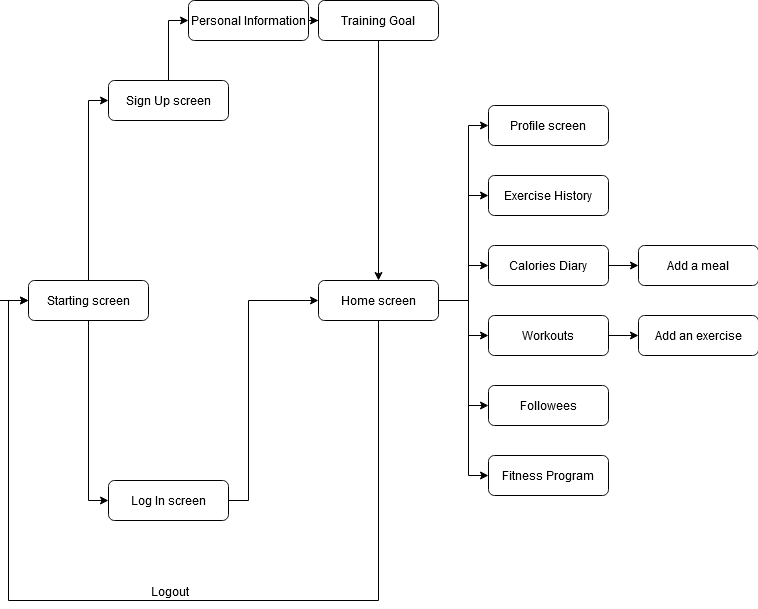
Основната структурна единица на Android приложенията е т.нар. Activity. Най-общо казано то е един екран от приложението, с който потребителят може да си взаимодейства. Всяко Activity си има свой xml файл, в който се описват параметрите и местоположението на използваните изгледи в съответния екран. Всяко Activity прекарва по-голямата част от жизнения си цикъл сменяйки 2 състояния – „активно”, когато то работи и „в пауза”, когато телефонът се заключи или бъде изключен. Дори тогава, обаче, съответното Activity не е напълно неактивно, в смисъл, че то може да си назначава работа, да извършва дадени операции, които не се нуждаят от интерфейс (Фигура 2.1).



*Фигура 2.1*

Приложението съдържа много различни екрани, които са свързани помежду си (Фигура 2.2). Комуникира си с база данни на Firebase, чиято структура се концентрира около потребителя и неговите дейности в приложението.

Приложението има един главен екран (Home page), през който може да се стига до всеки един от останалите прозорци, като преди това трябва да се мине през регистрация или вписване.



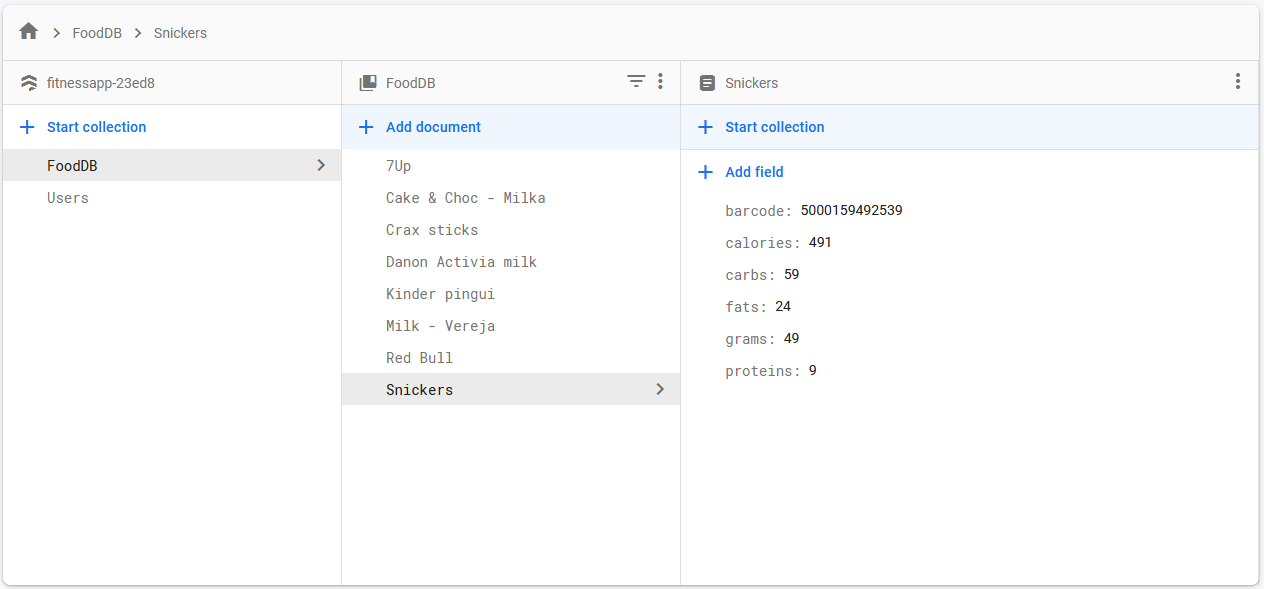
*Фигура 2.2*

Firebase е нерелационна (NoSQL) база данни. Този тип бази данни са подходящи за съхранение на голямо количество данни. При релационните(SQL базираните) бази данни има различни ограничения, които нерелационните бази данни разрешават, като:

* лесна мащабируемост върху клъстери от сървъри;
* поддръжка на различни типове данни;
* позволяване на разработка с гъвкави методологии.

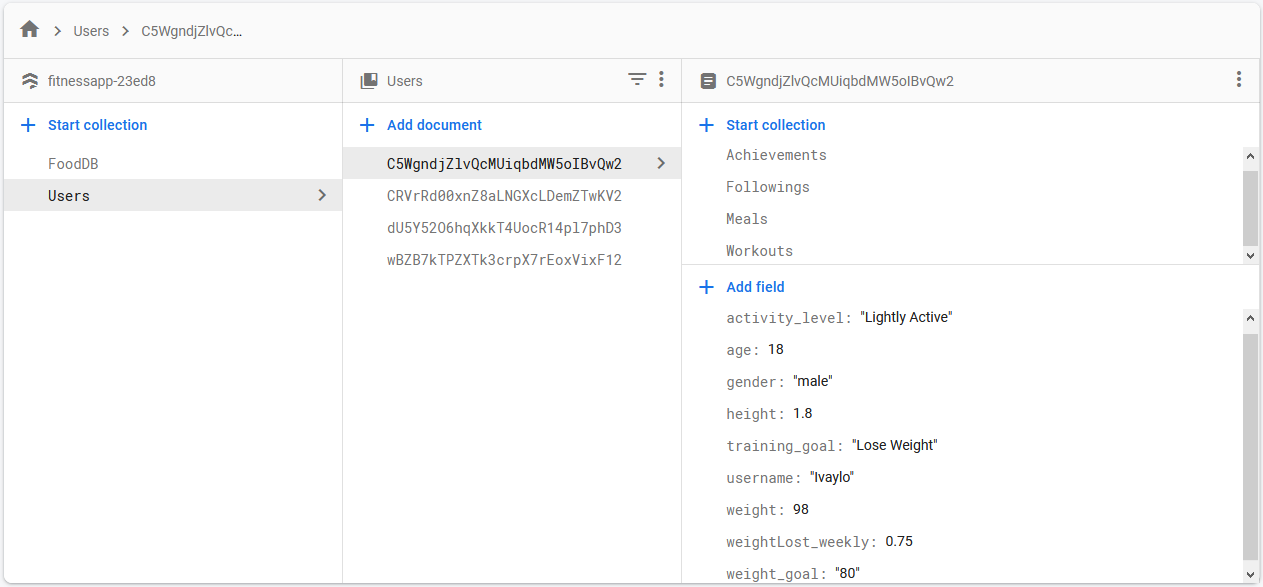
При SQL базираните бази данни, данните се съхраняват в таблици, подредени в редове и колони. Докато в NoSQL базираните бази данни, данните се структурират в JSON-подобни документи. Модела на данните е динамичен и е възможно да се добавят нови полета с данни, без това да налага преработка на схемата/структурата на базата данни.[2] [3] [4]

На Фигура 2.3 може да се види базата данни, която представлява две главни колекции: потребители (Users) и храни (FoodDB). В колекцията за храни има документи с пореден номер - името на продукта. Във всеки един от тези документи се съдържа информацията за съответния продукт: баркод, калории, въглехидрат, протеини, мазнини и грамаж.



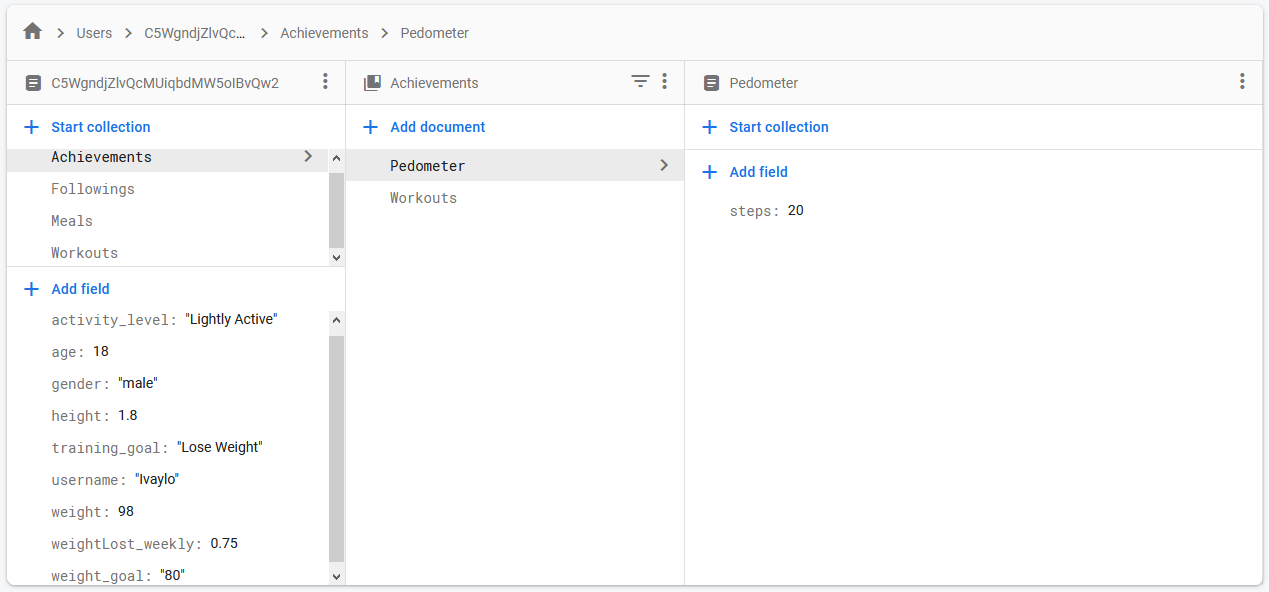
*Фигура 2.3*

В колекцията за потребители се съдържат документи с потребителския пореден номер. Във всеки документ има информация за потребителя: име, години, пол, физически характеристики и тренировъчни цели. Има също така и четири колекции: постижения, последователи, хранения и тренировки (Фигура 2.4).



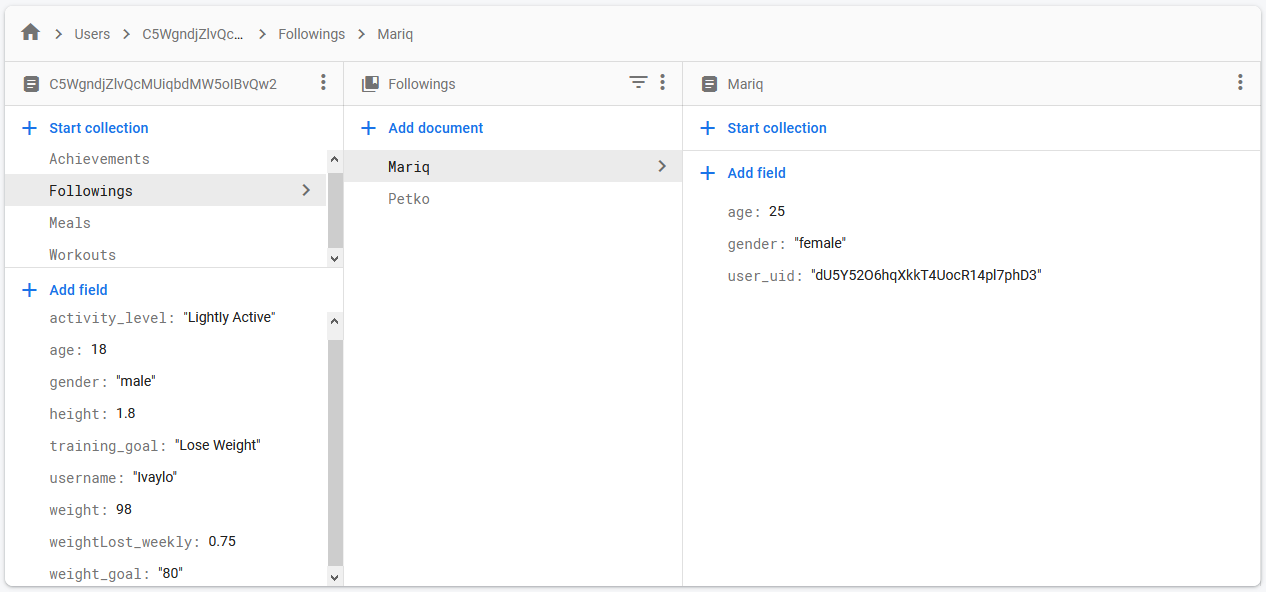
*Фигура 2.4*

Във Фигура 2.5 се вижда колекцията за постижения, в която има видовете постижения – такива, които са постигнати чрез крачкометъра (Pedometer) или такива постигнати с правенето на упражнения.



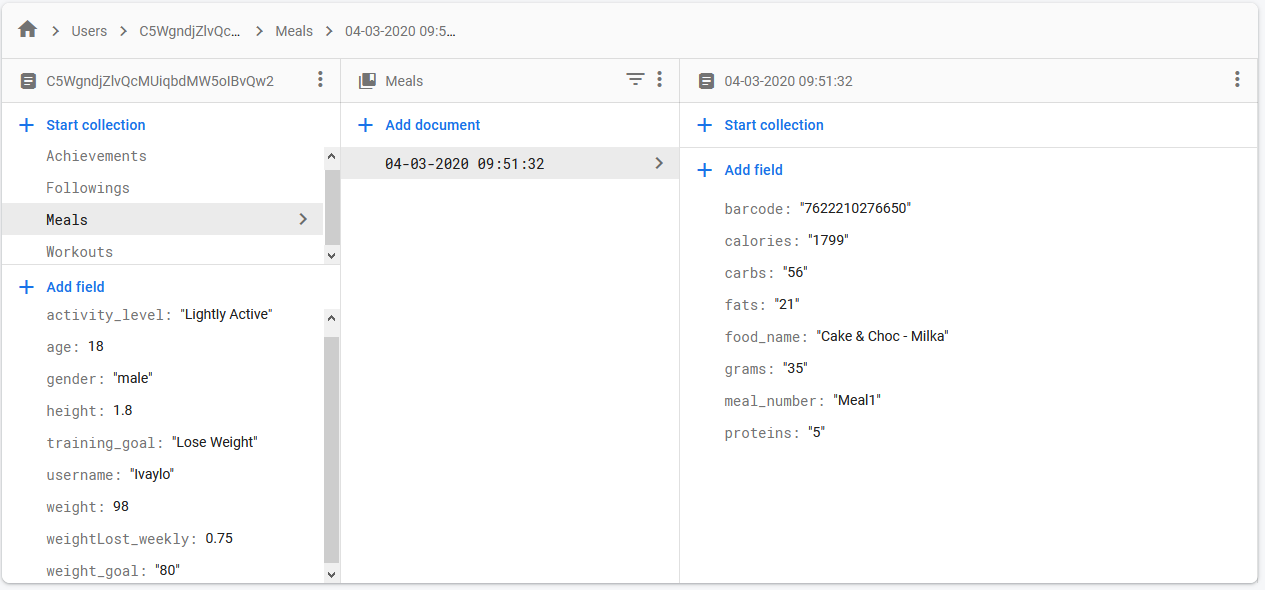
*Фигура 2.5*

В колекцията за последователи, показана на Фигура 2.6, има имената на хората, които потребителя следва, техните години, пол и пореден потребителски номер(uid).



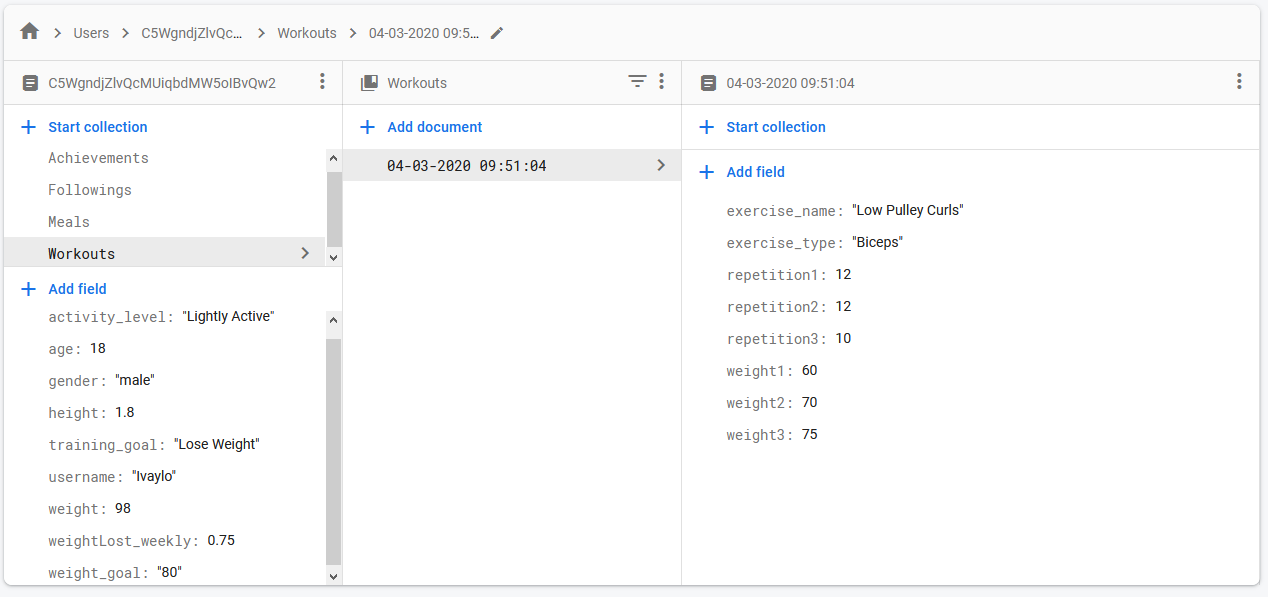
*Фигура 2.6*

В колекцията за хранения има датата на добавянето на продукта, характеристиките му, името на продукта, номера на хранене и макросите на продукта(Фигура 2.7).



*Фигура 2.7*

В колекцията за тренировки има датата на извършване на тренировката, тип и наименование на упражнението, повторение и тежест за всяка серия (Фигура 2.8).



*Фигура 2.8*

# ТРЕТА ГЛАВА

# Програмна реализация на мобилно фитнес приложение

## 3.1. Създаване на първоначалния екран

При стартиране на приложението Android ОС създава първоначално своя основен процес – Main Activity. В него са позиционирани единствено два бутона – бутон за регистрация и за вход – основни стъпки, с които се подсигурява на приложението потребителски данни, без които иначе приложението не може да продължи зареждането си. Бутоните са свързани логически с програмния код, чрез традиционния ClickListener, който се изпълнява след засечено отпускане на бутона. В методите се създава променлива от тип Intent, която е абстрактно описание на действие за преминаване от един процес (activity) в друг. Като термина процес в смисъла на activity не е задължително да бъде един единствен за приложението, какъвто е класическия пример с десктоп програмите. Напротив, Android приложенията, често използват няколко процеса, а употребата на повече от един е промотирана от създателите на Android платформата. До известна степен, използването на различни процеси само за едно приложение, допринася за по-лесния контрол на енергийна консумация, комбиниране на процеси между приложения (в случай, когато разработчика го е предвидил), но и инструментариум за изолиране на процесите един от друг, с цел по-висока сигурност. Впоследствие се инициализира променливата mauth, от тип FirebaseAuth, която ще се използва, като основа за вход и изход на потребителя към базата данни. Както е видно от приложения по-долу код във Фигура 3.1, FireBase модула влиза в действие още от самото начало при вход или ако потребителя използва приложението за пръв път – регистрация. В контекста на горното обяснение засягащо процесите в Android и цикъла им на живот – видно е, че бутоните извикват отделни процеси - „SignUpActivity“, „SignInActivity“, т.е. още от самото начало, независимо от посоката, в която ще поеме потребителя, приложението вече разполага с два процеса - „Main Activity“ и „SignUpActivity“ или „SignInActivity“ в зависимост от ситуацията.



*Фигура 3.1*

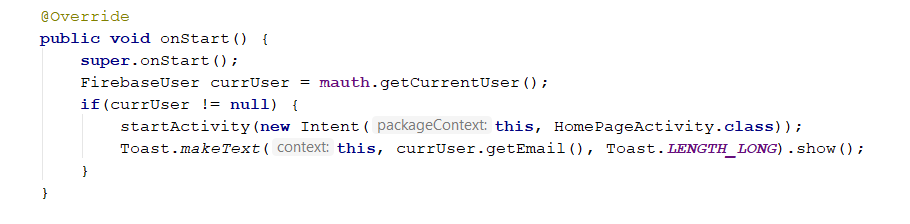
### 3.1.1 Структуроопределящ елемент на приложението.

В манифеста на приложението се дефинира име, икона, схема на дизайна, както и дефиниране на всички процеси в него. Екрана е разделен на две части – в лявата част е кода (в xml формат), а в дясно е визуализиран екрана на приложението. Дизайна на приложението обикновено е кохерентен и универсален за всички екрани в приложението, но разработчика, не е ограничен технически, и при нужда би могъл да зададе различни дизайни за всеки един от екраните.[1]

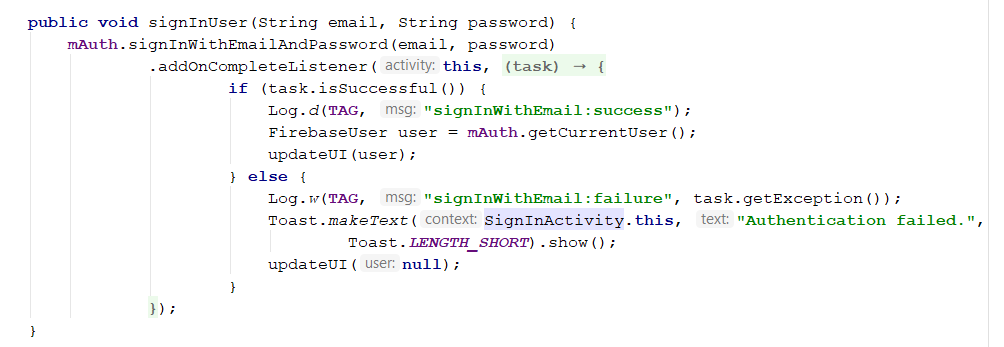
Името на приложението може да бъде дефинирано и в ресурсите на приложението (а именно strings), където чрез id ключове, да се зададе име на съответния ключ. Подобна функционалност е особено полезна, за приложения, преведени на няколко езика.

## 3.2. Поддръжка на различни акаунти и начин на регистриране.

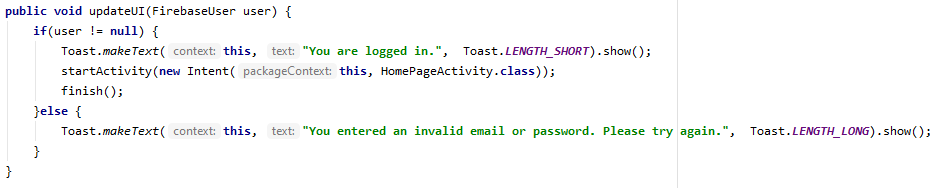
Впоследствие се използва модула Firebase и в процеса на вход приложението търси потребителското id, чрез което впоследствие се изпраща заявка към базата данни, за да извлече нужните данни за калкулации. Самият Firebase е лесно достъпен, тъй като в Android Studio са интегрирани и предвидени връзките между средата за разработка и Firebase модула, т.е. от разработчика не се изисква да намери начин да извиква тези функции – това вече е сторено за него (Фигура 3.2).

*Фигура 3.2*

Видно от Фигура 3.3 функционалността на модула FireBase е свързана с Android Studio, така че на разработчика му остава, само да провери за съответствие между потребителското име попълнено в полето за вход и паролата към него в базата данни на Firebase. Сценарий за неуспешен вход (при липса на съвпадения в базата) е предвиден. Целият процес на влизане протича в едно съвсем обикновено условие „ако“ (if). Макар съществуващия инструментариум, обаче, не е отнета гъвкавостта и при нужда може да се комбинират или измислят вариации на тази логика. В случая, тя пасва идеално с целите на това приложение.

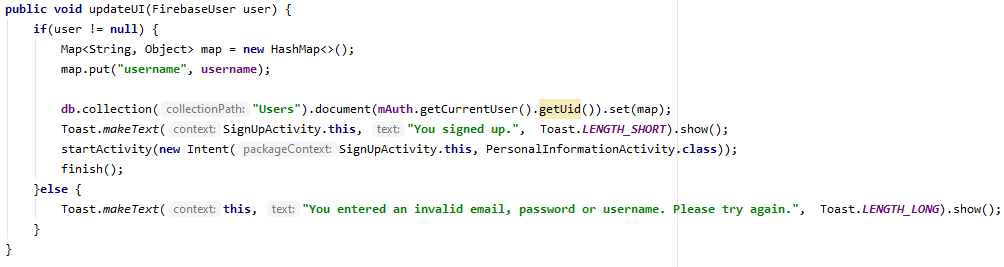
*Фигура 3.3*

Самият Firebase модул е предвидил и справяне с потребителските сесии, т.е. той не просто превежда приложението през целия цикъл на вход и изход от приложението, но предвижда и конфликтни ситуации, например при опит за влизане в системата, когато потребителят вече е вътре. От Фигура 3.4 е видно преминаването от процес (activity) в друг процес, събитие, за което потребителя остава изцяло в неведение, поради флуидното функциониране на приложението.



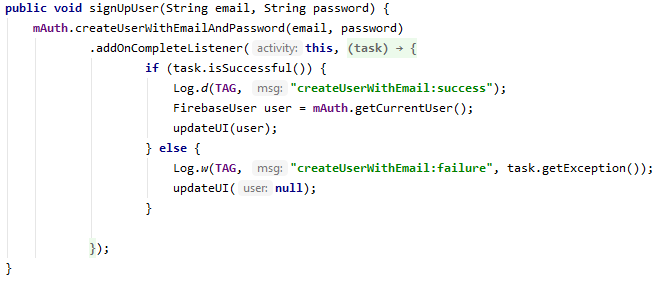
*Фигура 3.4*

След успешен вход, потребителя бива насочен към потребителския панел, в който трябва да попълни лични данни или физически характеристики, сред тях са – име, полова принадлежност, височина, тегло. Данните се използват впоследствие за изчисления и статистика, за самия потребител, който да наблюдава собствения си прогрес в спортните си занимания (Фигура 3.5).



*Фигура 3.5*

На Фигура 3.6 е показано, че успешната или неуспешната регистрация се обработва отново от Firebase модула. В случай на успешна регистрация, данните на новия потребител се записват в базата данни на Firebase .



*Фигура 3.6*

Кодът във Фигура 3.7 се грижи за попълване на личните данни на потребителя , които впоследствие ще се използват за изчисления и като условия, с които се разбира дали потребителя постига предварително зададените от него цели.



*Фигура 3.7*

Информацията за целите на потребителя и въвеждането на данни е максимално опростена, каквато е и ситуацията със структурата на базата данни – тази архитектура цели по-прости, а от там и бързи заявки към базата данни, тъй като скоростта на работа е критична в сферата на мобилните приложения. На Фигура 3.8 може да се види, че модулът Firebase си комуникира не с основния процес (Main Activity), а с конкретния процес – визуализиран от екрана с информацията и целите на потребителя.



*Фигура 3.8*

## 3.3. Упражнения с характеристики и история със серии и повторения.

Във Фигура 3.9 се наблюдават процесите, които се извършватр след като едно уплажнение бъде завършено. Извършват се проверки на въведената информация, след което се записват в Firebase базата данни. След това се добавя, към общия брой на извършени от потребителя упражнения, току-що направеното и после това число се записва в личните постижения на потребителя.



*Фигура 3.9*

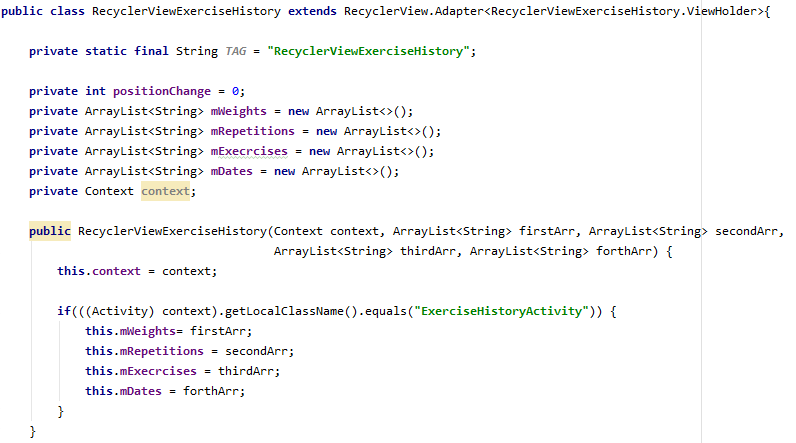
На Фигура 3.10 се разглежда извличането на информация за екрана с хронология на упражненията на потребителя. Данните се валидират, за да подсигурим коректното действие на приложението. Структурата на упражненията има два параметъра – маса, с която се извършва упражнението, и броя повторения.

Инфорамацията се запазва в четири различни листа(ArrayList), след което се създава обект от тип RecyclerView и адаптер, които ще послужи за визуализирането на информацията в листовете на съответното recycler view.

*Фигура 3.10*

RecyclerView представлява по-гъвкава версия на изгледа – ListView. В RecyclerView модела работят заедно наколко различни компонента, за да покажат инфорамцията. Самият RecyclerView контейнер се добавя в желания layout. Вече съответния RecyclerView контейнер се пълни с изгледи, осигурени от layout мениджъра.

Във Фигура 3.11 се имплементира RecyclerView адаптер, който е специфично направен да приема нужните параметри за изобразяването на упражненията и техните характеристики.

  
*Фигура 3.11*

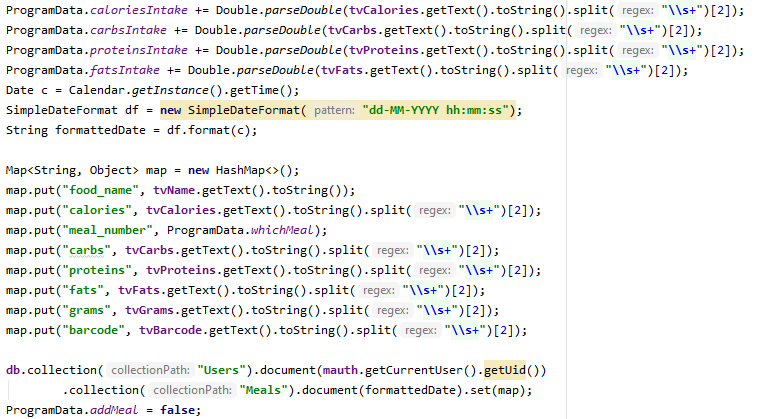
## 3.4. Добавяне на храни, изчисление на калории и макроси

При отварянето на дневника за калории се извиква метода calculateCalories, който взима стойностите от базата данни, записани при регистрацията и изчислява калориите. Във Фигура 3.12 са показани част от изчисленията. След което резултата се записват в text view, заедно с приетото количество калории и оставащите за деня.

*Фигура 3.12*

Функцията calculateCalories се извиква при главния екран и при дневника за калории, където метода изчислява и макросите (въглехидрати, мазнини и протеини).

Дневника за калории дава опцията да се добавя приетата храна през деня. Добавената храна се визуализира в съответното recycler view на едно от трите хранения. След като се натисне бутона за добавяне на храна, се отваря екран, в който се съдържа базата данни с храни. Там има опция да се добави храна чрез сканиране на баркод или чрез избиране на желаната храна. При сканирането на баркод се взима сканирания баркод и се проверява дали в базата данни той съществува. Ако съществува се отваря прозорец с бутон за добавяне на продукта и характеристиките му (калории, макроси, грамаж и баркод). На Фигура 3.13 се показва добавяне му, при което се взимат калориите и макросите от базата данни. След това те се добавят към глобалните променливи, които отговарят за общо консумираните калории и макроси.



*Фигура 3.13*

При връщане в дневника за калории се извършва операцията onRestart, в която отново се извиква метода calculateCalories, който обновява калориите и макросите.

## 3.5. Списък с последвани хора и feed от техните дейности

На Фигура 3.14 е показано как се добавя човек за следване. Въведения string се сравнява с потребителското име на всички регистрирани хора в приложението. Ако съвпадне се взимат годините, пола и потребителския му номер и се вкарват в колекцията followings на текущия потребител. На list view се визуализират имената на всички следвани хора.



*Фигура 3.14*

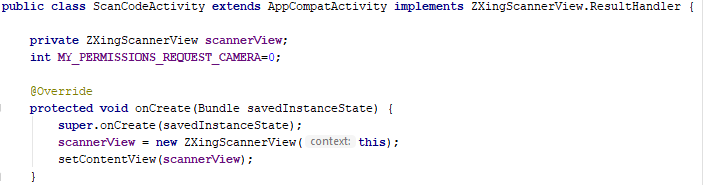
Когато един потребител надхвърли определен брой крачки или упражнения, в базата данни се добавя броят им в съответното подразделение. Ако някой от следваните хора има постижения те ще се изобразят, на вертикалното recycle view(Фигура 3.15).



*Фигура 3.15*

## 3.6 Сканиране на баркод на стока

Функцията за сканиране на баркод на продукти, използва съществуващата библиотека – ZxingScanner (Zebra Crossing) (Фигура 3.16). Тя използва една от функционалностите на камерата - onPreviewFrame. Този метод връща YUV(система за цветно кодиране) данни от снимката взета при сканирането. YUV съдържа три параметъра – яркост(Y), синя проекция(U) и червена проекция(V). В зависимост от хардуера и настройките на камерата между 15-30 FPS(кадри за секунда) метода onPreviewFrame връща YUV данните. След това се махат байтовете, които са извън сканираната зона и цветовете(UV). Преработват се останалите данни като се проследява контраста на яркостта(Y), за да се намерят черните линии и след това се взима широчината им.[5]

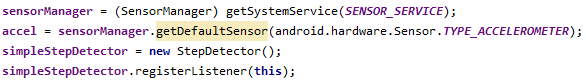


*Фигура 3.16*

Скенера след това връща сканирания баркод като string, който програмата обработва и сравнява с вече съществуващата база данни от продукти.

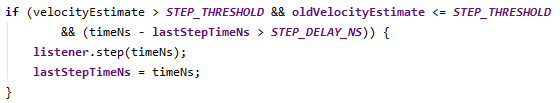
## 3.7 Имплементация на крачкометъра

Във Фигура 3.17 е показана имплементацията на крачкометъра. Той се състои от интерфейса StepListener, който следи в реално време за извършването на стъпка. Класът Sensor чрез алгоритми филтрира стойностите, които са близко до това да се нарекат стъпка. И класът StepDetector определя дали резултатите филтрирани от сензора се зачитат за стъпка.



*Фигура 3.17*

В класа StepDetector има променлива, която определя скоростта, която е нужна за да се зачете стъпка. Когато тя е надвишена и времето между миналата и текущата стъпка е по-голямо от определеното минимално, тогава се извиква метода step, който отчита една стъпка и я добавя към общия брой (Фигура 3.18).



Фигура 3.18

# ЧЕТВЪРТА ГЛАВА

# Ръководство за потребителя

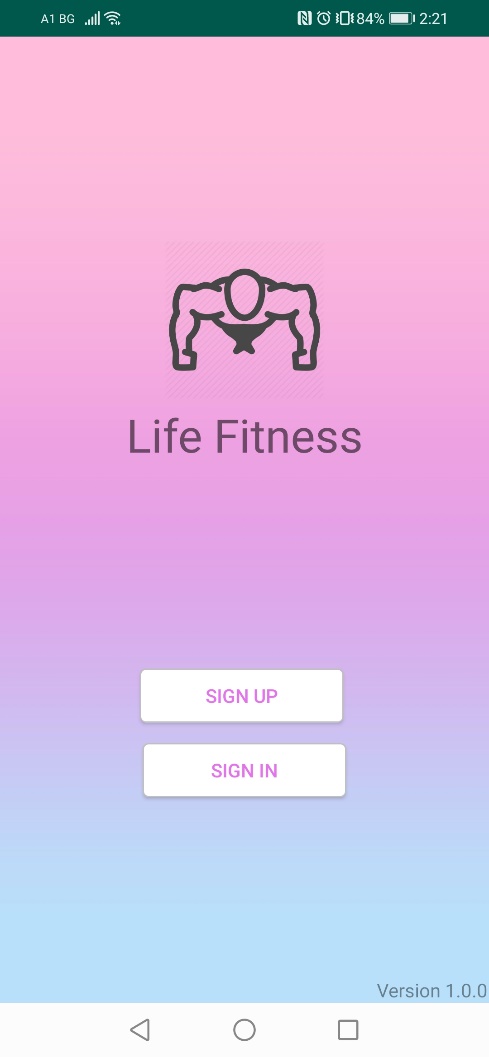
## 4.1 Изисквания за стартиране на приложението

Изисква се наличието на интернет и Android устройството с минимална версия 9.0 Pie (Api level 28). Това са двете изисквания за нормална работа с приложението.

## 4.2. Инструкции за ползване

### 4.2.1. Начален екран

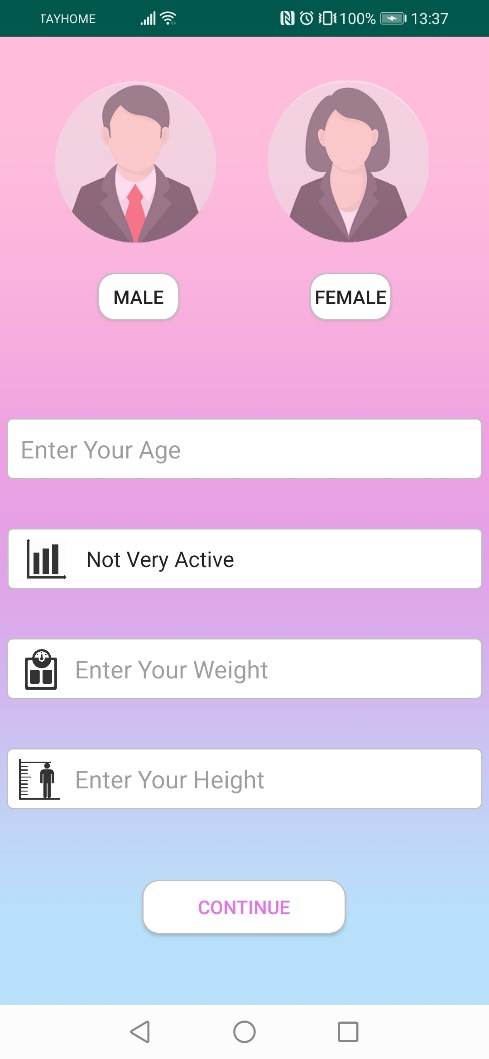
При стартиране на приложението потребителя ще види началния екран с логото на приложението и два бутона - единия за регистрация, а другия за влизане в приложението (Фигура 4.1).



*Фигура 4.1*

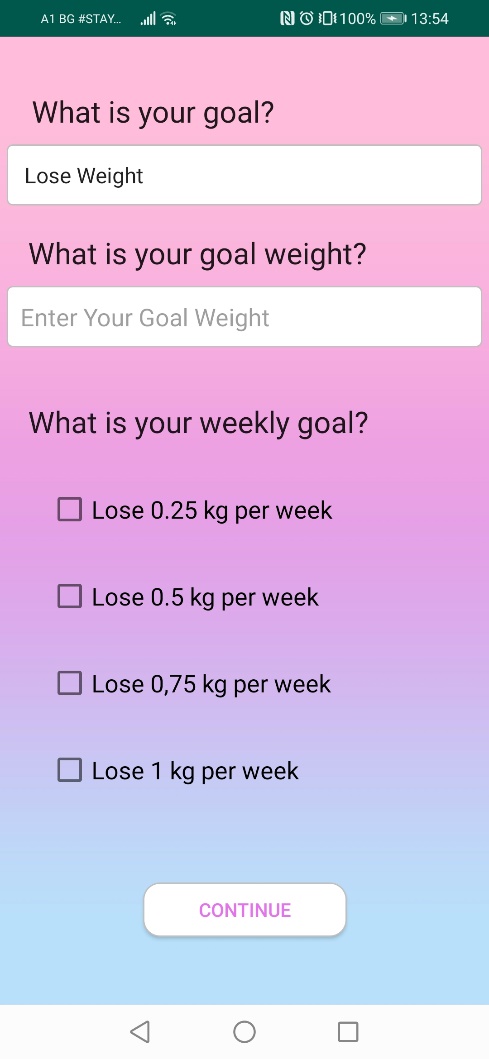
### 4.2.2. Екран за регистрация

При натискане на бутона за регистриране, приложението отваря екран за регистрация. Там се изисква да се въведат парола, имейл и потребителско име. При успешна регистрация на потребителя се визуализира екран подобен на Фигура 4.2. Този панел изисква въвеждането на лична информация, нужна за изчислението на необходимите калории. Избира се пол с натискане на бутона под съответния портрет. При натискане на второто поле се отваря падащо меню, от което се избира нивото на активност.



*Фигура 4.2*

След като е въведена информацията при натискане на бутона „Next“ се отваря следващият екран, в който се въвежда главната информация за изчисление на калории. На Фигура 4.3 се вижда структурата на екрана и какво трябва да бъде въведено. В първото поле се определя целта – дали ще се свалят килограми, качват или поддържат. В зависимост от избора, последващата информация ще се промени. Ако потребителя е избрал да сваля килограми до първата отметка ще пише „Lose 0.25 kg per week” , но ако избере да качва, до не ще пише „Gain 0.25 kg per week”.



*Фигура 4.3*

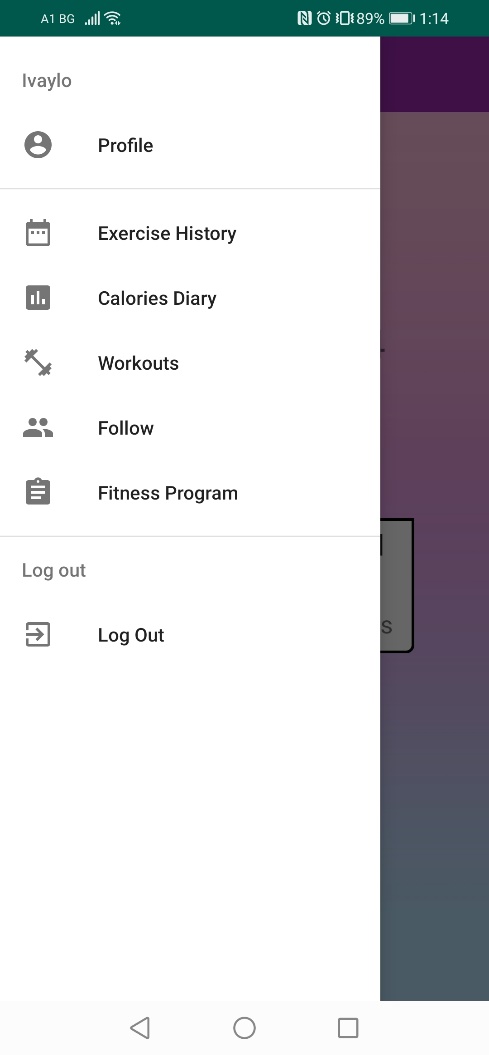
### 4.2.3 Главен екран

След регистрация или влизане с вече съществуващ акаунт, се отваря главния екран (Фигура 4.4), в който има крачкометър, feed от последваните хора и вече изчислените калории.



*Фигура 4.4*

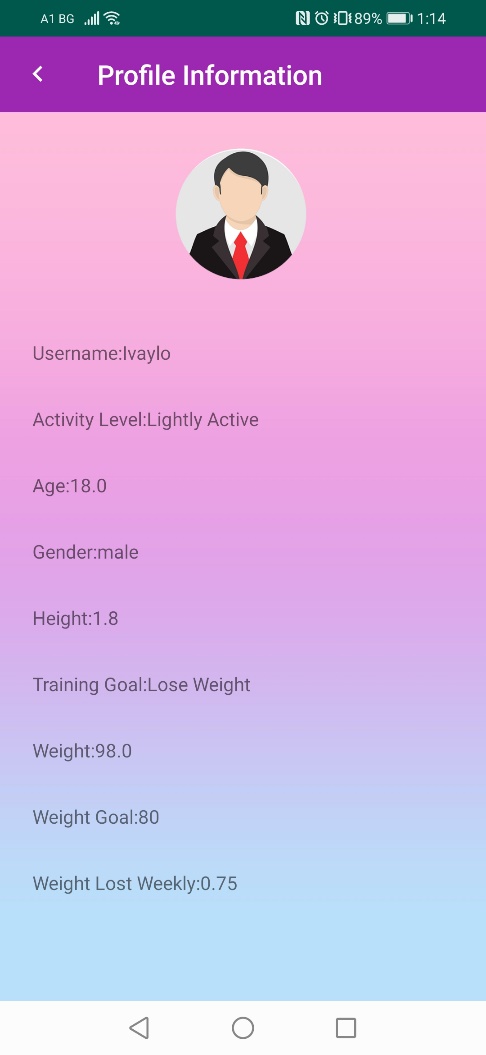
При натискане на бутона в горния ляв ъгъл се отваря падащо меню, като във Фигура 4.5, в което има бутони, които водят до екрани за: разглеждане на профила, история за проведени упражнения, дневник за калории, тренировки, следване, фитнес програма, както и за излизане от профила.



*Фигура 4.5*

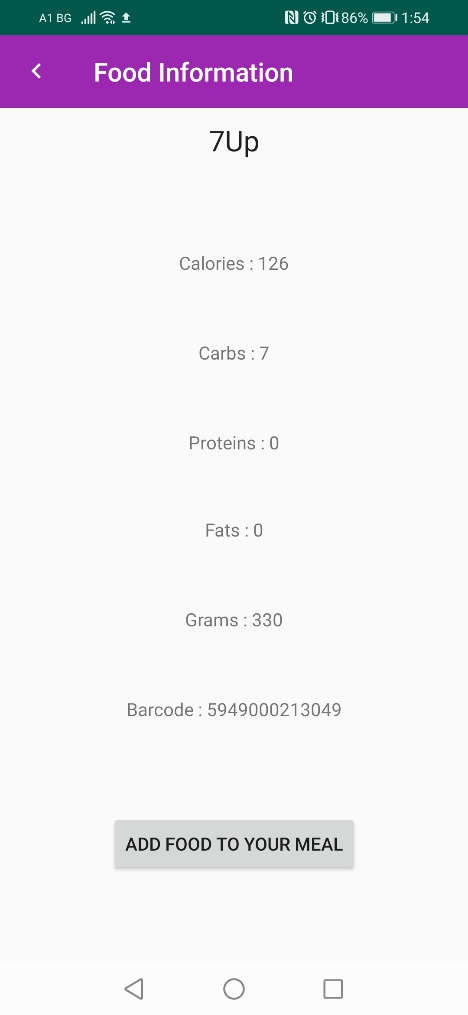
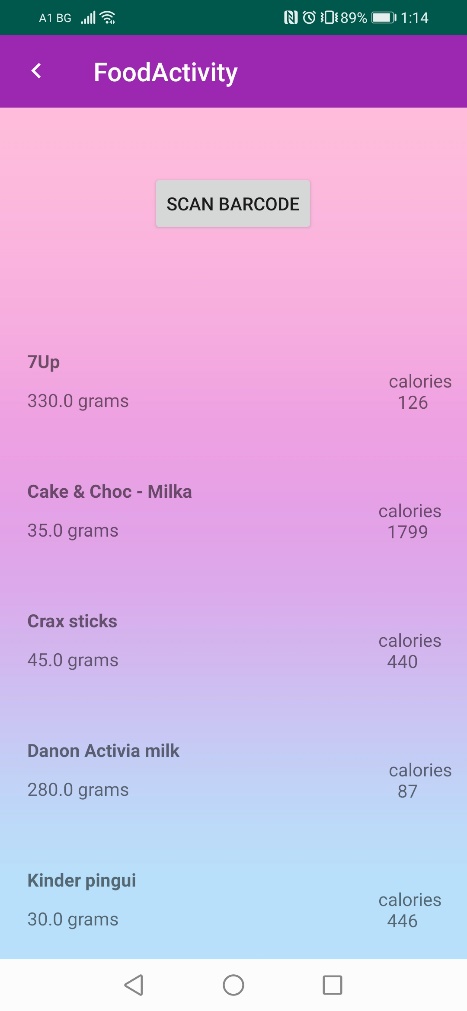
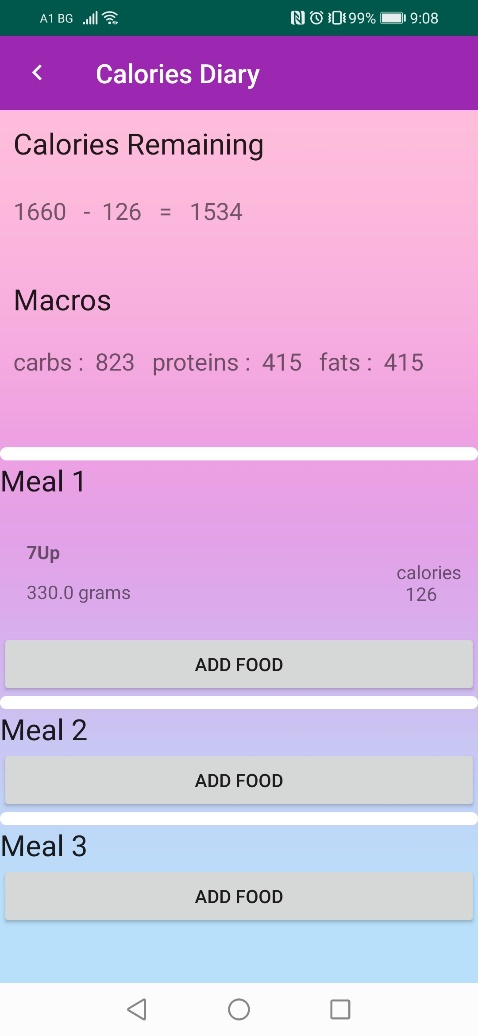
### 4.2.4 Основни функционалности на приложението

При натискане на първия бутон се отваря екран, подобен на Фигура 4.6, на който е цялата информация на потребителя.



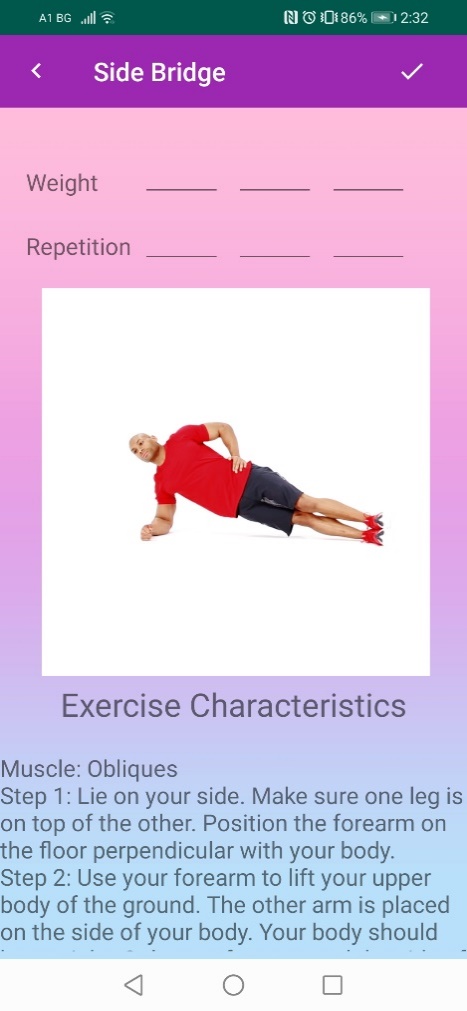
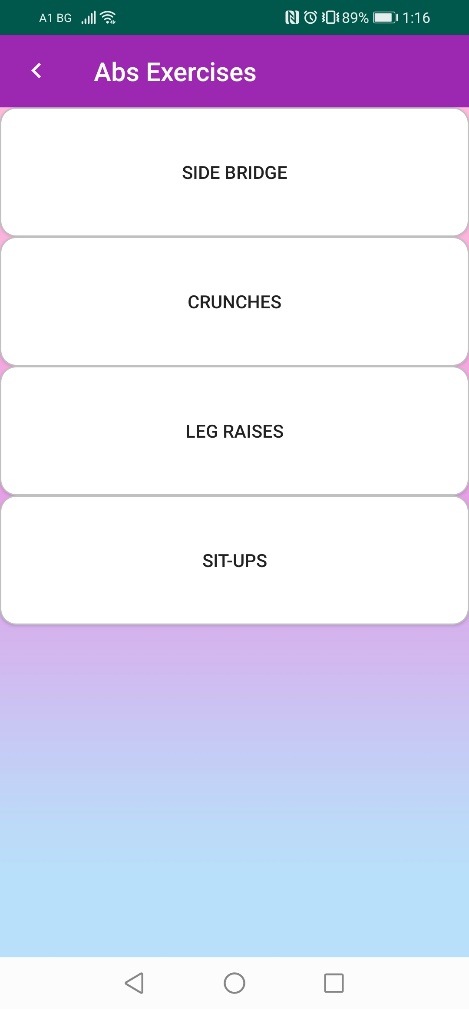
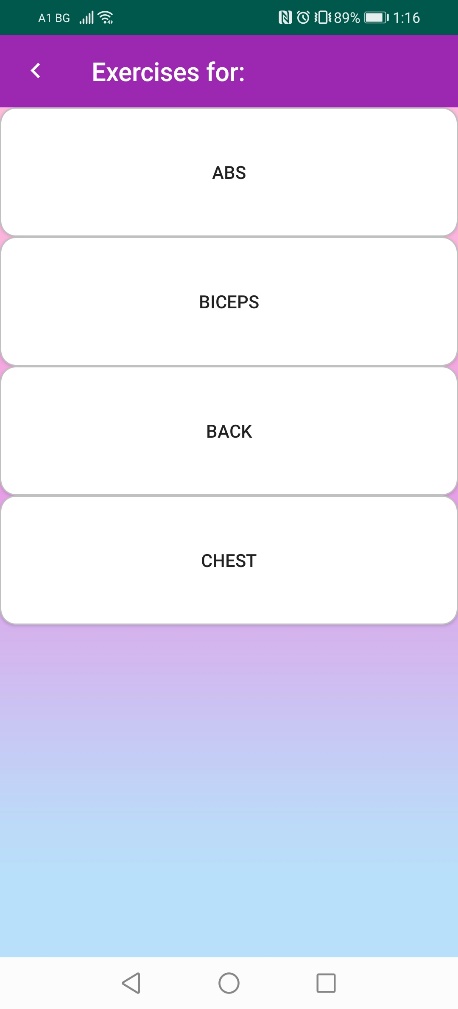
*Фигура 4.6*

При натискане на бутона calories diary се отваря екран, подобен на показания във Фигура 4.7, на който са изобразени калориите и макросите. Също така има три различни буна за всяко от храненията. При натискането на който и да е от тях, се отваря екран за добавяне на храна (Фигура 4.8). Там са подредени всички храни в базата данни на приложението. При избирането на храна се отваря екран от Фигура 4.9, в който пише информацията за продукта (баркод, калории, макроси и грамаж). Има и бутон, който добавя продукта във вече избраното за деня хранене.



*Фигура 4.7 Фигура 4.8 Фигура 4.9*

При натискане на бутона за тренировки се визуализира екран, като този във Фигура 4.10, на който има видове упражнения (за гърди, гръб, корем и т.н.). При избиране на някое от тях се отваря екрана от Фигура 4.11, в който има упражненията от съответния вид. След като се натисне върху дадено упражнение се показва екран със снимка и подробно описание за начина на изпълнение на упражнението. Записва се информация с каква тежест и колко повторения от серията са направени в двете полета над илюстрацията. (Фигура 4.12).



*Фигура 4.10 Фигура 4.11 Фигура 4.12*

При въвеждането им и натискането на бутона, в горния десен ъгъл, информацията за повторения и тежест се запазва. След което, ако се натисне бутона за история на упражненията, който се намира в главния екран, ще се покаже записаната тренировка, както е изобразено във Фигура 4.13.



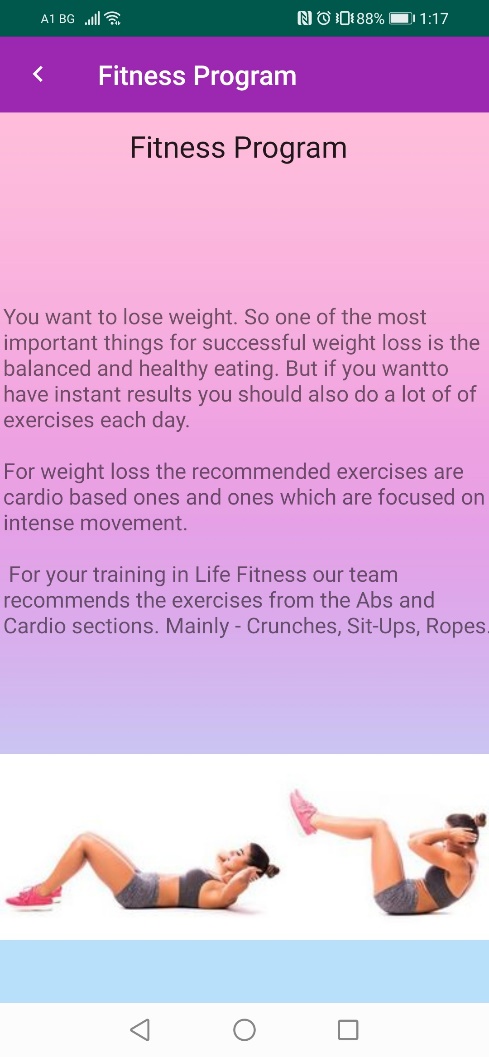
*Фигура 4.13*

При натискане на бутона Follow се отваря екран, в който може да се добавят хора, които потребителя иска да следва и да получава информация за техния прогрес през деня, изобразявайки го на главния екран. За да се добави такъв човек, трябва да се въведе неговото потребителско име. Ако има регистрация с това име, то ще се добави в долното поле. Ако се натисне името ще излезе друг екран с неговия профил (Фигура 4.14).



*Фигура 4.14*

При натискане на последния бутон – Fitness Program, ще се отвори екран, подобен на показания във Фигура 4.15. Визуализира се програма и съвети, дадени от специалисти в сферата, за видове упражнения и начин на хранене, които са в зависимост от зададената цел за използване на приложението (сваляне, качване или поддържане на килограми).



*Фигура 4.15*

# Заключение

В заключение може да се каже, че фитнес приложението би било полезно за всички, които се занимават или ще започнат да се занимават с физическа дейност . Това е първото приложение, което правя на Android Studio, но макар трудностите в началото, сега мога да твърдя, че съм се запознал в голяма част със средата и с нейните функционалности. Изискванията в по-голямата си част са спазени.

Бъдещото развитие се състои в това да се подобри интерфейса на приложението както и да се добавят различни анимации, текстури и други. Също може да се създаде offline mode, който да не се нуждае от Firebase, за да зареди потребителската информация. Друго нещо, което може да направи работата с приложението по приятна и комуникативна, е да се добави секция за постове и коментари на различни потребители.

# Използвана литература

1. Android developer guide.
2. Firebase documentation.
3. <https://www.359gsm.com/android_otvytre/>
4. <https://help.superhosting.bg/sql-nosql-databases.html>
5. <https://taniaanand.wordpress.com/2018/01/28/scanner-code-library-for-android-zebra-crossing/>

# Съдържание

[Увод 1](#_Toc35880363)

[ПЪРВА ГЛАВА 2](#_Toc35880364)

[Обзор на съществуващите решения. Преглед на технологии подходящи за изпълнение на дипломната работа. 2](#_Toc35880365)

[1.1. Съществуващи приложения 2](#_Toc35880366)

[1.1.1. MyFitnessPal 2](#_Toc35880367)

[1.1.2. Fitness & Bodybuilding 3](#_Toc35880368)

[1.2. Технологии и средства за развой 4](#_Toc35880369)

[1.2.1. Android Studio 4](#_Toc35880370)

[1.2.2. IntelliJ IDEA 5](#_Toc35880371)

[1.2.3. Basic4Android 6](#_Toc35880372)

[1.2.4. Java 7](#_Toc35880373)

[1.2.5. Kotlin 8](#_Toc35880374)

[1.2.6. Firebase 9](#_Toc35880375)

[Втора Глава 10](#_Toc35880376)

[Функционални изисквания и проектиране на фитнес приложението. 10](#_Toc35880377)

[2.1. Функционални изисквания към фитнес приложението. 10](#_Toc35880378)

[2.2. Съображения за избор на програмни средства и развойна среда. 11](#_Toc35880379)

[2.2.1. IDE – Android Studio 11](#_Toc35880380)

[2.2.2. Firebase 11](#_Toc35880381)

[2.2.3. Език за програмиране – Java 12](#_Toc35880382)

[2.3. Принцип на софтуерния модел 12](#_Toc35880383)

[2.3.1. Структура на приложението 12](#_Toc35880384)

[ТРЕТА ГЛАВА 19](#_Toc35880385)

[Програмна реализация на мобилно фитнес приложение 19](#_Toc35880386)

[3.1. Създаване на първоначалния екран 19](#_Toc35880387)

[3.1.1 Структуроопределящ елемент на приложението. 20](#_Toc35880388)

[3.2. Поддръжка на различни акаунти и начин на регистриране. 21](#_Toc35880389)

[3.3. Упражнения с характеристики и история със серии и повторения. 26](#_Toc35880390)

[3.4. Добавяне на храни, изчисление на калории и макроси 29](#_Toc35880391)

[3.5. Списък с последвани хора и feed от техните дейности 31](#_Toc35880392)

[3.6 Сканиране на баркод на стока 33](#_Toc35880393)

[3.7 Имплементация на крачкометъра 34](#_Toc35880394)

[ЧЕТВЪРТА ГЛАВА 35](#_Toc35880395)

[Ръководство за потребителя 35](#_Toc35880396)

[4.1 Изисквания за стартиране на приложението 35](#_Toc35880397)

[4.2. Инструкции за ползване 36](#_Toc35880398)

[4.2.1. Начален екран 36](#_Toc35880399)

[4.2.2. Екран за регистрация 37](#_Toc35880400)

[4.2.3 Главен екран 39](#_Toc35880401)

[4.2.4 Основни функционалности на приложението 41](#_Toc35880402)

[Заключение 47](#_Toc35880403)

[Използвана литература 48](#_Toc35880404)

[Съдържание 49](#_Toc35880405)