

**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ  
Факултет по Компютърни системи и Технологии**

**ДИПЛОМНА РАБОТА**

*Тема: Приложение за GPS проследяване*

Дипломант: **Даниел Георгиев Донков**Фак. No.: **121217035**Научен ръководител: **проф. Георги Попов**

**София, 2021**

**Съдържание**

**1. Увод ................................................................................3** *1.1. Цели и задачи ...........................................................................4*

**2. Обзор на съществуващите приложения.........................4**

**3. Проектиране на приложението .....................................5** *3.1. Увод в използваните технологии ..........................................5  
 3.1.1. Firebase .........................................................................5  
 3.1.2. Android Studio ...............................................................6  
 3.1.3. Java ................................................................................7*

*3.2. Изготвяне и подредба на фрагментитие в activities…….....8  
 3.2.1. Sign Up Activity ……………………………….............................8  
 3.2.2. Login Activity ………………..…………………….........................8*

*3.2.3. Main Activity ……………………..………………..........................9*

*3.2.4. Maps Activity ……………………………………...........................9*

*3.3. XML source code на всички activities………………………………......10  
 3.3.1. Sign Up Activity ………………………………...........................10  
 3.3.2. Login Activity ………………..…………………….......................12*

*3.3.3. Main Activity ……………………..………………........................13*

*3.3.4. Maps Activity …………………………………….........................14*

*3.4 Build gradle ………………………………………..……………………………......17*

*3.4.1. Project ниво ……………………………………...........................17  
 3.4.2. Module ниво ………….……..…………………….......................18*

**4. Реализация ...................................................................19**  
 *4.1. Sign Up Activity class ……………………………..……………………….......19  
 4.2. Login Activity class ……………..……………......................................23  
 4.3. Main Activity class ………………………………...................................26  
 4.4. Maps Activity class …………………….…………….............................33*

*4.5. Foreground Service class ……………………………............................38*

*4.6. RxUpdate Worker class ………………………..……............................39*

**5. Заключения ..................................................................43**

**6. Източници .....................................................................43**

**1. Увод**

Във все така разрастващият се технологичен свят, като се взима на предвид и обстановката с все още продължаващата пандемия, хората по цял свят са все повече и повече активни в различни по вид социални мрежи като в това число най - голям процент са тийнейджърите. Нараства и броят на родители които се притесняват от това, каква информация и с кого децата им споделят в тези така наречени социални мрежи. Дори и информацията да не споделена има доста на брой начини с които тя да бъде взета. Също така има и голям брой на безследно изчезнали деца не само в България, но и по света. Затова хората се нуждаят от предпазни мерки. Тази дипломна работа предоставя решение на този проблем. Днешните смарт устройства имат много на брой функции, стига да им е предоставена интернет връзка, а локацията на едно дете по всяко време на денонощието е една доста важна информация за неговият родител.

***1.1. Цели и задачи***

Целта на тази дипломна работа е изработката на приложение за следене на локацията на смартфон. Като това да става дори приложението да не се използва. Избраната среда за програмиране позволява да се инсталира на голям процент Android устройства, които са над или равни на версията на Android – Oreo.

Процесът включва интегриране на cloud база данни Firebase Firestore, програмиране и интегриране на user interface система за регистриране и автентикация на потребителя която включва и имейл верификация - Firebase Authentication system. Също така и реализацията, а именно писане на код и неговото логическо свързване.

Използваният софтуер за приложението е Android Studio , а избраният език за програмиране е Java.

**2. Обзор на съществуващите приложения**

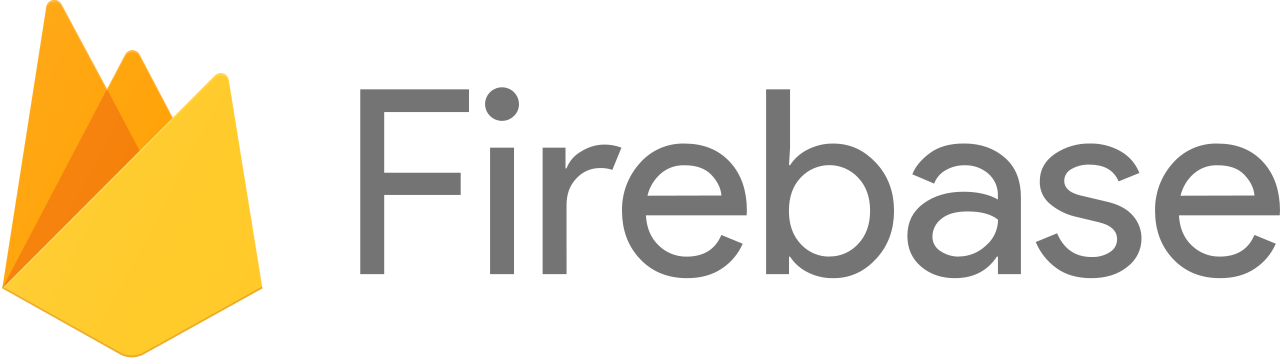
След обзор на съществуващите приложения в Google Play Store, които интегрират система за споделяне/следене на локация се наблюдава използване на предварителен user login / user sign up начин за автентикация на настоящия използващ приложението потребител.

Повечето приложения от такъв тип могат да споделят локация само ако тя бъде изискана от отсрещния потребител или с допълнителни настройки. В този проект е необходимо да се постигне качване на локацията в базата данни постоянно.

**3. Проектиране на приложението**

***3.1 Увод в използваните технологии***

***3.1.1 Firebase***

**Фиг.1 Лого на платформата Firebase**

Firebase на Фиг. 1 произведенa от Google е онлайн платформа за създаване на мобилни приложения. Tя се интегрира в мобилното приложение и по този начин се използват нейните функционалности.

Някой от които са онлайн сървърна автентикационна система за регистриране и влизане в приложението, както и cloud база данни които толкова общо са написани, че да може свободно да се модифицира от потребителя, така че да отговаря на неговите нужди.

Firebase authentication има няколко на брой избора от различни по вид начини за логване и регистриране на потребителя в приложението, които лесно могат да се интегрират в избраното приложение само с няколко допълнителни настройки. A пък Firebase Firestore е много гъвкава и лесно мащабируема база данни за мобилни, уеб и сървърни разработки която е с интегрирана cloud технология.

***3.1.2 Android Studio***

**Фиг. 2 Лого на средата за разработка Android Studio**

Android Studio е официалната интегрирана среда за разработка (IDE -

Integrated Development Environment) за мобилната операционна система Android на Google, изградена върху софтуера IntelliJ IDEA на JetBrains. Тя поддържа езиците Java, C ++, Kotlin и „всички езикови функции на Java 7, както и подмножество от езикови функции на Java 8, които варират в зависимост от версията на платформата.

Android SDK (Software development kit) той притежава разообразие от инструменти за разработка включвайки софтуерни библиотеки, графични библиотеки, документация, емулатор, интегрирана среда за разработка, дебъгер и много други.

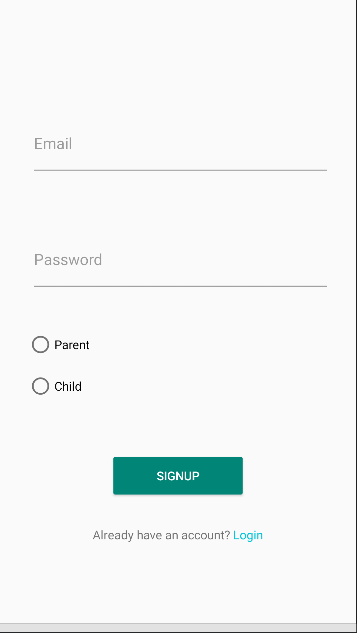
***3.1.3 Java***



**Фиг.3 Лого на програмен език Java**

Java e програмен език с общо предназначение, той е обектно ориентиран и е проектиран да има колкото се може по-малко зависимости за реализация. Tой е избран като технология за изграждане на приложения, използващи управляван код, който може да се изпълнява на мобилни устройства. Tой работи във виртуална машина, така че няма нужда да се прекомпилира за всеки телефон и е лесен за защита и има голям брой инструменти за разработка на Java. Както и като доста известен език според повечето потребители на уеб базираните услуги и форуми, ми помага да намеря значително много помощни ресурси в интернет.

***3.2. Изготвяне и подредба на фрагментите в activities***

***3.2.1. Sign Up Activity***

Вземайки на предвид това че приложението се нуждае от система за регистриране се създава първия екран от приложението Sign Up Activity Фиг.4 , като на него се поставят 5 елемента в Design View.

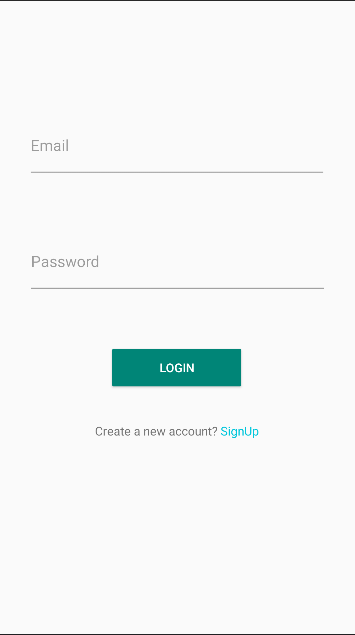
2 елемeнта TextInputEditText за Email и Password

1 елемент Radio Button Container който съдържа 2 Radio Button-a за userType

1 Button елемент SignUp

1 TextView clickable eлемент , който води

**Фиг.4 Sign Up Activity** към Login Activity

***3.2.2. Login Activity***

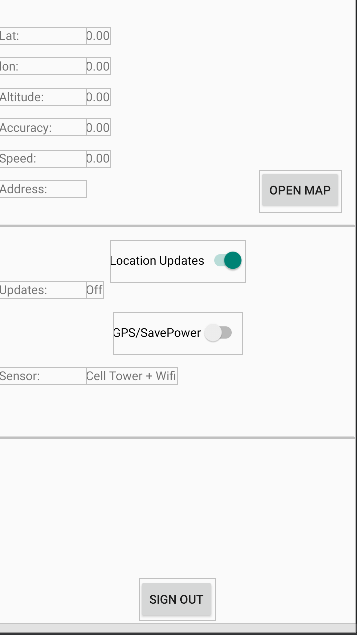
Второто activity от приложението е Фиг.5 Login Activity или Sign In Activity което съдържа:

2 елемeнта TextInputEditText за Email и Password

1 Button елемент за автентикация на потребителя: Login

1 ТеxtView clickable елемент, който води към Sign Up Activity

**Фиг.5 Login Activity**

***3.2.3. Main Activity***

Третото activity на приложението е същинското Activity (Маin Activity Фиг.6) в което ще се поместят по-голяма част от главните функции.

В него се разполагат доста на брой елементи които са:

16 TextView елемента половината от които служат за информация на по динамичната част от която се помества до тях.

2 Button eлемента : Sign Out и Open Map

единия от които води до следващия екран, а **Фиг.6 Маin Activity**

именно Maps Activity Фиг.7.

Другият служи за излизане на потребителя от запазения акаунт.

2 RadioButton елемента, единият от които се използва за пускане/спиране на ъпдейтите на локацията (Location Updates), другият за сменяне на метода по който се взема информацията за локацията ( GPS/Save Power).

***3.2.4. Maps Activity***

Последният екран (Аctivity) е Фиг.7 Maps Activity, като той съдържа само два елемента:

Google Map Fragment който покрива целия екран .

Button елемент който е върху Google Map Fragment-a ADD FRIEND, който служи за добавяне на приятел от базата данни за следене на локацията.

**Фиг.7 Maps Activity** 

***3.3. XML source code на всички activities***

***3.3.1. Sign Up Activity***

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".LoginActivity">  
  
 <Button  
 android:id="@+id/signUp"  
 style="@style/Widget.AppCompat.Button.Colored"  
 android:layout\_width="157dp"  
 android:layout\_height="55dp"  
 android:layout\_marginStart="30dp"  
 android:layout\_marginTop="35dp"  
 android:layout\_marginEnd="30dp"  
 android:text="@string/signup"  
 android:textColorHint="@color/colorPrimaryDark"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/radioGroup" />  
  
 <TextView  
 android:id="@+id/loginText"  
 android:layout\_width="287dp"  
 android:layout\_height="48dp"  
 android:layout\_marginStart="30dp"  
 android:layout\_marginTop="5dp"  
 android:layout\_marginEnd="30dp"  
 android:clickable="true"  
 android:gravity="center"  
 android:text="@string/loginhere"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.497"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/signUp"  
 app:layout\_constraintVertical\_bias="0.118" />  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/tv\_password"  
 android:layout\_width="346dp"  
 android:layout\_height="78dp"  
 android:layout\_marginStart="30dp"  
 android:layout\_marginTop="56dp"  
 android:layout\_marginEnd="30dp"  
 android:hint="@string/password"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="1.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_email">  
  
 </com.google.android.material.textfield.TextInputEditText>  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/tv\_email"  
 android:layout\_width="346dp"  
 android:layout\_height="78dp"  
 android:layout\_marginStart="30dp"  
 android:layout\_marginTop="128dp"  
 android:layout\_marginEnd="30dp"  
 android:hint="@string/email"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="1.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent">  
  
 </com.google.android.material.textfield.TextInputEditText>  
  
 <RadioGroup  
 android:id="@+id/radioGroup"  
 android:layout\_width="353dp"  
 android:layout\_height="113dp"  
 android:layout\_marginStart="5dp"  
 android:layout\_marginTop="35dp"  
 android:layout\_marginEnd="5dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.538"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_password">  
  
 <RadioButton  
 android:id="@+id/radioButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:minHeight="48dp"  
 android:text="Parent" />  
  
 <RadioButton  
 android:id="@+id/radioButton2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:minHeight="48dp"  
 android:text="Child" />  
 </RadioGroup>  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

***3.3.2. Login Activity***

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".LoginActivity">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/tv\_username"  
 android:layout\_width="346dp"  
 android:layout\_height="78dp"  
 android:layout\_marginLeft="30dp"  
 android:layout\_marginTop="128dp"  
 android:layout\_marginRight="30dp"  
 android:hint="@string/email"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.507"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent">  
  
 </com.google.android.material.textfield.TextInputEditText>  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/tv\_password"  
 android:layout\_width="346dp"  
 android:layout\_height="78dp"  
 android:layout\_marginLeft="30dp"  
 android:layout\_marginTop="56dp"  
 android:layout\_marginRight="30dp"  
 android:hint="@string/password"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.538"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_username" />  
  
 <Button  
 android:id="@+id/buttonLogin"  
 style="@style/Widget.AppCompat.Button.Colored"  
 android:layout\_width="157dp"  
 android:layout\_height="55dp"  
 android:layout\_marginStart="30dp"  
 android:layout\_marginTop="56dp"  
 android:layout\_marginEnd="30dp"  
 android:text="@string/login"  
 android:textColorHint="@color/colorPrimaryDark"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.498"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_password" />  
  
 <TextView  
 android:id="@+id/signuphere"  
 android:layout\_width="261dp"  
 android:layout\_height="48dp"  
 android:layout\_marginStart="30dp"  
 android:layout\_marginTop="5dp"  
 android:layout\_marginEnd="30dp"  
 android:clickable="true"  
 android:gravity="center"  
 android:text="@string/signuphere"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.497"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/buttonLogin"  
 app:layout\_constraintVertical\_bias="0.074" />  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

***3.3.2. Main Activity***

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <View  
 android:id="@+id/divider2"  
 android:layout\_width="409dp"  
 android:layout\_height="1dp"  
 android:layout\_marginTop="60dp"  
 android:background="?android:attr/listDivider"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_sensor" />  
  
 <TextView  
 android:id="@+id/tv\_labellat"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="32dp"  
 android:text="Lat:"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/tv\_lat"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="32dp"  
 android:text="0.00"  
 app:layout\_constraintStart\_toEndOf="@+id/tv\_labellat"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/tv\_labellon"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="lon:"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_labellat" />  
  
 <TextView  
 android:id="@+id/tv\_lon"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="0.00"  
 app:layout\_constraintStart\_toEndOf="@+id/tv\_labellon"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_lat" />  
  
 <TextView  
 android:id="@+id/tv\_labelaltitude"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="Altitude:"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_labellon" />  
  
 <TextView  
 android:id="@+id/tv\_altitude"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="0.00"  
 app:layout\_constraintStart\_toEndOf="@+id/tv\_labelaltitude"  
 app:layout\_constraintTop\_toTopOf="@+id/tv\_labelaltitude" />  
  
 <TextView  
 android:id="@+id/tv\_labelaccuracy"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="Accuracy:"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_labelaltitude" />  
  
 <TextView  
 android:id="@+id/tv\_accuracy"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="0.00"  
 app:layout\_constraintStart\_toEndOf="@+id/tv\_labelaccuracy"  
 app:layout\_constraintTop\_toTopOf="@+id/tv\_labelaccuracy" />  
  
 <TextView  
 android:id="@+id/tv\_labelspeed"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="Speed:"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_labelaccuracy" />  
  
 <TextView  
 android:id="@+id/tv\_speed"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="0.00"  
 app:layout\_constraintStart\_toEndOf="@+id/tv\_labelspeed"  
 app:layout\_constraintTop\_toTopOf="@+id/tv\_labelspeed" />  
  
 <TextView  
 android:id="@+id/tv\_labelsensor"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="Sensor:"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/sw\_gps" />  
  
 <TextView  
 android:id="@+id/tv\_sensor"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Cell Tower + Wifi"  
 app:layout\_constraintStart\_toEndOf="@+id/tv\_labelsensor"  
 app:layout\_constraintTop\_toTopOf="@+id/tv\_labelsensor" />  
  
 <TextView  
 android:id="@+id/tv\_labelupdates"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:text="Updates:"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/sw\_locationsupdates" />  
  
 <TextView  
 android:id="@+id/tv\_updates"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Off"  
 app:layout\_constraintStart\_toEndOf="@+id/tv\_labelupdates"  
 app:layout\_constraintTop\_toTopOf="@+id/tv\_labelupdates" />  
  
 <Switch  
 android:id="@+id/sw\_locationsupdates"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:checked="true"  
 android:minHeight="48dp"  
 android:text="Location Updates"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/divider" />  
  
 <Switch  
 android:id="@+id/sw\_gps"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:minHeight="48dp"  
 android:text="GPS/SavePower"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_labelupdates" />  
  
 <TextView  
 android:id="@+id/tv\_address"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintStart\_toEndOf="@+id/tv\_lbladdress"  
 app:layout\_constraintTop\_toTopOf="@+id/tv\_lbladdress" />  
  
 <TextView  
 android:id="@+id/tv\_lbladdress"  
 android:layout\_width="100dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="Address:"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_labelspeed" />  
  
 <View  
 android:id="@+id/divider"  
 android:layout\_width="409dp"  
 android:layout\_height="1dp"  
 android:layout\_marginTop="32dp"  
 android:background="?android:attr/listDivider"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_address" />  
  
 <Button  
 android:id="@+id/showMap"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginEnd="16dp"  
 android:layout\_marginBottom="16dp"  
 android:text="Open Map"  
 app:layout\_constraintBottom\_toTopOf="@+id/divider"  
 app:layout\_constraintEnd\_toEndOf="parent" />  
  
 <Button  
 android:id="@+id/signOut"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="50dp"  
 android:layout\_marginTop="50dp"  
 android:layout\_marginEnd="50dp"  
 android:layout\_marginBottom="10dp"  
 android:text="Sign Out"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.497"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/divider2"  
 app:layout\_constraintVertical\_bias="0.964" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

**3.3.4. Maps Activity**

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:orientation="vertical">  
  
 <fragment  
 android:id="@+id/map"  
 class="com.google.android.gms.maps.SupportMapFragment"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_alignParentStart="true"  
 android:layout\_alignParentTop="true"  
 tools:context=".MapsActivity" />  
  
 <Button  
 android:id="@+id/addButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentStart="true"  
 android:layout\_alignParentEnd="true"  
 android:layout\_alignParentBottom="true"  
 android:layout\_marginStart="165dp"  
 android:layout\_marginEnd="165dp"  
 android:layout\_marginBottom="30dp"  
 android:background="#359c5e"  
 android:padding="8dp"  
 android:text="@string/addText"  
 android:textColor="#212121"  
 tools:text="add friend" />  
  
</RelativeLayout>

**3.4. Build Gradle**

**3.4.1. Module ниво**

apply plugin: 'com.android.application'  
apply plugin: 'com.android.application'  
apply plugin: 'com.google.gms.google-services'  
  
android **{** compileSdkVersion 30  
 buildToolsVersion "30.0.0"  
  
 compileOptions **{** sourceCompatibility JavaVersion.*VERSION\_1\_8* targetCompatibility JavaVersion.*VERSION\_1\_8* **}** defaultConfig **{** applicationId "com.example.diploma"  
 minSdkVersion 19  
 targetSdkVersion 30  
 versionCode 1  
 versionName "1.0"  
  
 multiDexEnabled true  
  
 testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
 **}** buildTypes **{** release **{** minifyEnabled false  
 proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  
 **}  
 }  
}**dependencies **{** implementation platform('com.google.firebase:firebase-bom:28.4.0')  
 implementation 'com.google.firebase:firebase-analytics'  
 implementation 'com.google.firebase:firebase-firestore'  
 implementation fileTree(dir: "libs", include: ["\*.jar"])  
 implementation 'androidx.appcompat:appcompat:1.3.1'  
 implementation platform('com.google.firebase:firebase-bom:28.4.0')  
 implementation 'com.google.firebase:firebase-auth'  
 implementation 'com.google.android.gms:play-services-auth:19.2.0'  
 implementation 'androidx.constraintlayout:constraintlayout:2.1.0'  
 implementation 'com.google.android.gms:play-services-maps:17.0.1'  
 testImplementation 'junit:junit:4.12'  
 androidTestImplementation 'androidx.test.ext:junit:1.1.3'  
 androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'  
 implementation 'com.google.android.gms:play-services-location:18.0.0'  
 implementation "androidx.fragment:fragment:1.4.0-alpha08"  
 implementation 'com.google.android.material:material:1.5.0-alpha02'  
 implementation 'com.github.VishnuSivadasVS:Advanced-HttpURLConnection:1.2'  
 implementation 'com.google.firebase:firebase-dynamic-links'  
 implementation 'com.google.firebase:firebase-analytics'  
 implementation "androidx.work:work-runtime:2.5.0"  
**}**

**3.4.2. Project ниво**

// Top-level build file where you can add configuration options common to all sub-projects/modules.  
buildscript {  
  
 repositories {  
 google()  
 jcenter()  
 }  
  
 dependencies {  
 classpath "com.android.tools.build:gradle:4.0.2"  
 classpath 'com.google.gms:google-services:4.3.10'  
 // NOTE: Do not place your application dependencies here; they belong  
 // in the individual module build.gradle files  
 }  
}  
  
allprojects {  
 repositories {  
 google()  
 jcenter()  
 maven { url 'https://jitpack.io' }  
 }  
}  
  
task clean(type: Delete) {  
 delete rootProject.buildDir  
}

**4. Реализация**  
 ***4.1. Sign Up Activity class***

public class SignUpActivity extends Activity {  
  
 private static final String *TAG* = "EmailPassword";  
   
 private FirebaseAuth mAuth;  
 FirebaseFirestore db = FirebaseFirestore.*getInstance*();  
  
 CollectionReference users = db.collection("users");  
 Map<String, Object> dbUser;  
 Button signUpButton;  
 RadioButton rbParent, rbChild;  
 TextInputEditText email, password;  
 TextView loginhere;  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_sign\_up*);  
   
 mAuth = FirebaseAuth.*getInstance*();  
 dbUser = new HashMap<>();  
  
 email = findViewById(R.id.*tv\_email*);  
 password = findViewById(R.id.*tv\_password*);  
 signUpButton = findViewById(R.id.*signUp*);  
 loginhere = findViewById(R.id.*loginText*);  
 rbChild = (RadioButton) findViewById(R.id.*radioButton2*);  
 rbParent = (RadioButton) findViewById(R.id.*radioButton*);  
  
 signUpButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick ( View view ) {  
 if(signUpButton.isPressed()) {  
 if (!email.getText().toString().equals("") && !password.getText().toString().equals("")) {  
 if (email.getText().toString().contains("@") && email.getText().toString().contains(".")) {  
  
 if (rbParent.isChecked()) {  
 createAccount(email.getText().toString() , password.getText().toString(), "parent");  
 }  
 else if (rbChild.isChecked()) {  
 createAccount(email.getText().toString() , password.getText().toString(), "child");  
 }  
 else {  
 Toast.*makeText*(getApplicationContext(),"Please specify user type!",Toast.*LENGTH\_SHORT*).show();  
 }  
  
 } else {  
 Toast.*makeText*(getApplicationContext() , "Email is not a valid email!", Toast.*LENGTH\_SHORT*).show();  
  
 }  
 } else {  
 Toast.*makeText*(getApplicationContext() , "All fields are required for registration!" , Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 }  
 });  
  
 loginhere.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick ( View view ) {  
 if (loginhere.isPressed()){  
  
 Intent intent = new Intent(SignUpActivity.this,LoginActivity.class);  
 startActivity(intent);  
  
 }  
  
 }  
 });  
  
 }  
  
 private void createAccount(String email, String password, String userType) {  
 mAuth.createUserWithEmailAndPassword(email, password)  
 .addOnCompleteListener(SignUpActivity.this, new OnCompleteListener<AuthResult>() {  
 @Override  
 public void onComplete(@NonNull Task<AuthResult> task) {  
 if (task.isSuccessful()) {  
   
 Log.*d*(*TAG*, "createUserWithEmail:success");  
 FirebaseUser user = mAuth.getCurrentUser();  
  
  
 dbUser.put("userType",userType);  
 db.collection("users").document(user.getEmail())  
 .set(dbUser)  
 .addOnSuccessListener(new OnSuccessListener<Void>() {  
 @Override  
 public void onSuccess(Void aVoid) {  
 Log.*d*(*TAG*, "DocumentSnapshot successfully written!");  
  
 }  
 })  
 .addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
 Log.*w*(*TAG*, "Error writing document", e);  
  
 }  
 });  
 sendEmailVerification(user);  
  
  
  
  
 } else {  
  
 Log.*w*(*TAG*, "createUserWithEmail:failure", task.getException());  
 Toast.*makeText*(SignUpActivity.this, "Authentication failed.",  
 Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
   
 }  
  
 public void sendEmailVerification(FirebaseUser user) {  
  
  
 user.sendEmailVerification()  
 .addOnCompleteListener(new OnCompleteListener<Void>() {  
 @Override  
 public void onComplete(@NonNull Task<Void> task) {  
 if (task.isSuccessful()) {  
 Log.*d*(*TAG*, "Email sent.");  
 Toast.*makeText*(SignUpActivity.this, "Verification email sent!",  
 Toast.*LENGTH\_SHORT*).show();  
 FirebaseAuth.*getInstance*().signOut();  
 Intent intent = new Intent(getApplicationContext(), LoginActivity.class);  
 startActivity(intent);  
 }  
 }  
 });  
 }  
  
}

Логиката на Sign Up class-а e да се вземат въведените от потребителя имейл, парола, както и вида на потребителя, чрез натискане на бутона Sign Up, затова се инициализира signupButton обект от тип бутон той се свързва с бутона Sign Up чрез findViewById(R.id.signup).

След което се инициализира OnClickListener за този бутон и при натискане (if(signupButton.isClicked)) от потребителя се проверява въведената информация, а именно имейл и парола полетата дали не са празни, дали имейл полето съдържа символите „.“ , „@“ и дали един от радио бутоните за вида на потребителя е натиснат. При правилно въведена информация се създава акаунт и се изпраща верификационно писмо с функцийте createAccount и sendEmailVerification. Това предотвратява влизането на потребителя докато не му бъде верифициран имейла. След успешно създаване на акаунта на потребителя се взима вида на потребителя в зависимост от това кой радио бутон е избран и се качва информацията в базата данни. След успешно верифициране потребителя може да влезе в приложението. Ако потребителя вече има акаунт може да отвори следващия прозорец Login Activity, това става чрез свързване на инициализирания TextView loginhere който се свързва чрез findViewById(R.id.loginText).

Отново използваме OnClickListener функцията върху loginhere, която при натискане текста създава нов Intent (намерение) и се извиква функцията startActivity(Intent).

***4.2. Login Activity class***

public class LoginActivity extends AppCompatActivity {  
  
 private FirebaseAuth mAuth = FirebaseAuth.getInstance();  
 FirebaseUser user = mAuth.getCurrentUser();  
 String userType;  
 TextInputEditText email, password;  
 Button loginButton;  
 TextView signuphere;  
  
 @Override  
 protected void onCreate ( Bundle savedInstanceState ) {  
  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_login);  
  
  
 loginButton = findViewById(R.id.buttonLogin);  
 signuphere = findViewById(R.id.signuphere);  
 email = findViewById(R.id.tv\_username);  
 password = findViewById(R.id.tv\_password);  
  
 loginButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick ( View view ) {  
 if(loginButton.isPressed()) {  
 if (!email.getText().toString().equals("") && !password.getText().toString().equals("")) {  
 if (email.getText().toString().contains("@") && email.getText().toString().contains(".")) {  
  
 signIn(email.getText().toString() , password.getText().toString());  
  
 } else {  
 Toast.makeText(getApplicationContext() , "Email is not a valid email!", Toast.LENGTH\_SHORT).show();  
  
 }  
 } else {  
 Toast.makeText(getApplicationContext() , "Fields are empty! Please fill in your user information!" , Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 }  
 });  
  
 signuphere.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick ( View view ) {  
 if (signuphere.isPressed()){  
  
 Intent intent = new Intent(LoginActivity.this,SignUpActivity.class);  
 startActivity(intent);  
  
 }  
  
 }  
 });  
  
 }  
  
  
  
 private void signIn(String email, String password) {  
 // [START sign\_in\_with\_email]  
 mAuth.signInWithEmailAndPassword(email, password)  
 .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {  
 @Override  
 public void onComplete(@NonNull Task<AuthResult> task) {  
 if (task.isSuccessful()) {  
 if(mAuth.getCurrentUser().isEmailVerified()) {  
   
 Log.d(TAG , "signInWithEmail:success");  
 FirebaseUser user = mAuth.getCurrentUser();  
 updateUI(user);  
 }else{  
 Toast.makeText(LoginActivity.this, "Please verify your email!",  
 Toast.LENGTH\_SHORT).show();  
 }  
 } else {  
   
 Log.w(TAG, "signInWithEmail:failure", task.getException());  
 Toast.makeText(LoginActivity.this, "Authentication failed.",  
 Toast.LENGTH\_SHORT).show();  
 }  
 }  
 });  
 }  
  
 private void reload(FirebaseUser user) {  
  
 if (user!=null) {  
 Intent intent = new Intent(this , MainActivity.class);  
 startActivity(intent);  
 Toast.makeText(LoginActivity.this , "Already logged in as " + user.getEmail().toString() , Toast.LENGTH\_SHORT).show();  
 }else {  
 Toast.makeText(LoginActivity.this , "Didn't log as " + user.getEmail().toString() , Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 private void updateUI(FirebaseUser user) {  
  
 FirebaseFirestore.getInstance().collection("users").document(user.getEmail()).get().addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 userType = (String) task.getResult().get("userType");  
 Intent intent = new Intent(getApplicationContext() , MainActivity.class);  
 intent.putExtra("userType" , userType);  
 startActivity(intent);  
 Toast.makeText(LoginActivity.this , "Logged in as " + mAuth.getCurrentUser().getEmail() , Toast.LENGTH\_SHORT).show();  
 }else {  
 Toast.makeText(LoginActivity.this , "Cannot log in as " + mAuth.getCurrentUser().getEmail() , Toast.LENGTH\_SHORT).show();  
 }  
 });  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.O)  
 @Override  
 protected void onStart () {  
 super.onStart();  
  
 if (user != null) {  
 if (user.isEmailVerified()) {  
 FirebaseFirestore.getInstance().collection("users").document(user.getEmail()).get().addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 userType = (String) task.getResult().get("userType");  
 Intent intent = new Intent(this , MainActivity.class);  
 intent.putExtra("userType" , userType);  
 startActivity(intent);  
 Toast.*makeText*(LoginActivity.this , "Logged in as " + mAuth.getCurrentUser().getEmail() , Toast.*LENGTH\_SHORT*).show();  
  
 } else {  
 Toast.*makeText*(LoginActivity.this , "Task wasn't successful!" , Toast.*LENGTH\_SHORT*).show();  
 Intent intent = new Intent(this , MainActivity.class);  
 startActivity(intent);  
 }  
 });  
  
 } else {  
 Toast.*makeText*(LoginActivity.this , "Please verify your email " + mAuth.getCurrentUser().getEmail() , Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 }  
  
}

В Login Activity има подобна подредба на елементите като в Sign Up Activity, отново има бутон на който се слага OnClickListener, а именно бутона loginButton който при натискане проверява въведената информация от потребителя(имейл,парола).

След това се проверява дали съществува такъв потребител в автентикационната система на Firebase, чрез инициализирания обект mAuth който се свързва с автентикационната система вградена в проекта посредством функцията FirebaseAuth.getInstance(). С функцията signUp(email,password) опитваме да намерим потребител в системата на Firebase с въведения от него имейл. При успешно намиране на такъв потребител се взима userType от базата данни за този потребител, чрез FirebaseFirestore.getInstance().getCollection(“users”).get(user.getEmail).get() при успешно намиране на такъв документ се извлича само информацията от него която е нужна и се записва в userType = (String) task.getResult().get(“userType”) след което се предава като информация на следващия клас чрез Intent: intent.putExtra("userType" , userType) и се стартира Main Activity класа.

***4.3. Main Activity class***

public class MainActivity extends AppCompatActivity {  
  
 FirebaseAuth mAuth = FirebaseAuth.*getInstance*();  
 FirebaseUser user = mAuth.getCurrentUser();  
 Map<String, Object> dbUser;  
 FirebaseFirestore db = FirebaseFirestore.*getInstance*();  
 LocalDateTime localDateTime;  
  
 private static final int *PERMISIONS\_FINE\_LOCATION* = 99;  
 TextView tv\_lat, tv\_lon, tv\_altitude, tv\_accuracy, tv\_speed, tv\_sensor, tv\_updates, tv\_address, tv\_labelupdates;  
 Switch sw\_gps, sw\_locationsupdates;  
 Button showMap,signOut;  
 LocationRequest locationRequest;  
 FusedLocationProviderClient fusedLocationProviderClient;  
 LocationCallback locationCallback;  
  
 double myLat = 0;  
 double myLng = 0;  
  
 @RequiresApi(api = Build.VERSION\_CODES.*O*)  
 @Override  
 protected void onCreate ( Bundle savedInstanceState ) {  
  
 dbUser = new HashMap<>();  
  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 tv\_lat = findViewById(R.id.*tv\_lat*);  
 tv\_labelupdates = findViewById(R.id.*tv\_labelupdates*);  
 tv\_lon = findViewById(R.id.*tv\_lon*);  
 tv\_altitude = findViewById(R.id.*tv\_altitude*);  
 tv\_accuracy = findViewById(R.id.*tv\_accuracy*);  
 tv\_speed = findViewById(R.id.*tv\_speed*);  
 tv\_sensor = findViewById(R.id.*tv\_sensor*);  
 tv\_updates = findViewById(R.id.*tv\_updates*);  
 tv\_address = findViewById(R.id.*tv\_address*);  
 sw\_gps = findViewById(R.id.*sw\_gps*);  
 sw\_locationsupdates = findViewById(R.id.*sw\_locationsupdates*);  
 showMap = findViewById(R.id.*showMap*);  
 signOut = findViewById(R.id.*signOut*);  
  
 locationRequest = LocationRequest.*create*();  
 locationRequest.setInterval(15000);  
 locationRequest.setFastestInterval(5000);  
 locationRequest.setPriority(LocationRequest.*PRIORITY\_BALANCED\_POWER\_ACCURACY*);  
  
 locationCallback = new LocationCallback() {  
 @Override  
 public void onLocationResult ( @NonNull LocationResult locationResult ) {  
 super.onLocationResult(locationResult);  
 Location location = locationResult.getLastLocation();  
 updateUIValues(location);  
 updateDBValues(location);  
 }  
 };  
  
 showMap.setOnClickListener(view -> {  
 if (showMap.isPressed()){  
 Intent intent = new Intent(MainActivity.this , MapsActivity.class);  
 updateGPS();  
 intent.putExtra("Latitude" , myLat);  
 intent.putExtra("Longitude" , myLng);  
 startActivity(intent);  
 }  
  
 });  
  
 signOut.setOnClickListener(view -> {  
 if (signOut.isPressed()){  
  
 Intent intent = new Intent(MainActivity.this,LoginActivity.class);  
  
 FirebaseAuth.*getInstance*().signOut();  
  
 startActivity(intent);  
  
 Toast.*makeText*(MainActivity.this,"Successfully signed out!",Toast.*LENGTH\_SHORT*).show();  
  
 }  
  
 });  
  
 sw\_gps.setOnClickListener(view -> {  
 if (sw\_gps.isChecked()) {  
 locationRequest.setPriority(LocationRequest.*PRIORITY\_HIGH\_ACCURACY*);  
 tv\_sensor.setText("Using GPS sensors");  
 } else {  
 locationRequest.setPriority(LocationRequest.*PRIORITY\_BALANCED\_POWER\_ACCURACY*);  
 tv\_sensor.setText("Using Towers + WIFI");  
 }  
 });  
  
 sw\_locationsupdates.setOnClickListener(view -> {  
 if (sw\_locationsupdates.isChecked()) {  
 startLocationUpdates();  
 } else {  
 stopLocationUpdates();  
 }  
 });  
  
 }  
  
 private void startLocationUpdates ( ) {  
 tv\_updates.setText("Location is being tracked");  
 if (ActivityCompat.*checkSelfPermission*(this , Manifest.permission.*ACCESS\_FINE\_LOCATION*) == PackageManager.*PERMISSION\_GRANTED*) {  
  
 fusedLocationProviderClient.requestLocationUpdates(locationRequest , locationCallback , null);  
 updateGPS();  
  
 }  
 else {  
 Toast.*makeText*(this, "This app requires permission for Location to be granted in order to work properly!", Toast.*LENGTH\_SHORT*).show();  
 finish();  
 }  
  
 }  
 private void stopLocationUpdates ( ) {  
 tv\_updates.setText("Location is not being tracked");  
 tv\_lat.setText("Not tracking location");  
 tv\_lon.setText("Not tracking location");  
 tv\_speed.setText("Not tracking location");  
 tv\_address.setText("Not tracking location");  
 tv\_accuracy.setText("Not tracking location");  
 tv\_altitude.setText("Not tracking location");  
 tv\_sensor.setText("Not tracking location");  
  
 fusedLocationProviderClient.removeLocationUpdates(locationCallback);  
 }  
  
 private void updateGPS () {  
  
 fusedLocationProviderClient = LocationServices.*getFusedLocationProviderClient*(MainActivity.this);  
  
 if (ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*) == PackageManager.*PERMISSION\_GRANTED*) {  
  
 fusedLocationProviderClient.getLastLocation().addOnSuccessListener(this , new OnSuccessListener<Location>() {  
  
 @RequiresApi(api = Build.VERSION\_CODES.*O*)  
 @Override  
 public void onSuccess ( Location location ) {  
 if(location!=null) {  
 updateUIValues(location);  
 if (isConnected()) {  
 updateDBValues(location);  
 }  
 }else{  
 Toast.*makeText*(getApplicationContext(),"Cannot get location!",Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
 }  
 else {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {  
 requestPermissions(new String [] {Manifest.permission.*ACCESS\_FINE\_LOCATION*}, *PERMISIONS\_FINE\_LOCATION*);  
 }  
  
 }  
}  
  
 @RequiresApi(api = Build.VERSION\_CODES.*O*)  
 private void updateUIValues ( Location location ) {  
  
 tv\_lat.setText(String.*valueOf*(location.getLatitude()));  
  
 tv\_lon.setText(String.*valueOf*(location.getLongitude()));  
  
 tv\_accuracy.setText(String.*valueOf*(location.getAccuracy()));  
  
 if (location.hasAltitude()){  
 tv\_altitude.setText(String.*valueOf*(location.getAltitude()));  
 }  
 else {  
 tv\_altitude.setText("Not available");  
 }  
  
 if (location.hasSpeed()){  
 tv\_speed.setText(String.*valueOf*(location.getSpeed()));  
 }  
 else {  
 tv\_speed.setText("Not available");  
 }  
  
 Geocoder geocoder = new Geocoder(MainActivity.this);  
  
 try {  
 List<Address> addresses = geocoder.getFromLocation(location.getLatitude(), location.getLongitude(), 1);  
 tv\_address.setText(addresses.get(0).getAddressLine(0));  
 }  
 catch (Exception e) {  
 tv\_address.setText("Unable to get street address");  
  
 }  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.*O*)  
 private void updateDBValues ( Location location ) {  
  
 myLat = location.getLatitude();  
 myLng = location.getLongitude();  
  
 dbUser.put("latitude", myLat);  
 dbUser.put("longitude", myLng);  
 dbUser.put("email", user.getEmail());  
 dbUser.put("time", localDateTime.*now*());  
  
 db.collection("users").document(user.getEmail())  
 .update(dbUser)  
 .addOnSuccessListener(aVoid -> Log.*d*(*TAG*, "DocumentSnapshot successfully written!"))  
 .addOnFailureListener(e -> Log.*w*(*TAG*, "Error writing document", e));  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.*O*)  
 @Override  
 protected void onStart ( ) {  
 super.onStart();  
  
 if (ActivityCompat.*checkSelfPermission*(this , Manifest.permission.*ACCESS\_BACKGROUND\_LOCATION*) == PackageManager.*PERMISSION\_GRANTED*) {  
 Intent intent = new Intent(getApplicationContext() , ForegroundService.class);  
 stopService(intent);  
  
 sw\_locationsupdates.setChecked(true);  
 sw\_gps.setChecked(true);  
 tv\_updates.setText("Location is being tracked");  
 tv\_sensor.setText("Using GPS sensors");  
 locationRequest.setPriority(LocationRequest.*PRIORITY\_HIGH\_ACCURACY*);  
  
 updateGPS();  
  
 Bundle extras = getIntent().getExtras();  
 String userType = "child";  
 if (extras != null) {  
 userType = extras.getString("userType");  
 if (userType != null) {  
 if (userType.equals("child")) {  
 showMap.setVisibility(View.*GONE*);  
 sw\_locationsupdates.setVisibility(View.*GONE*);  
 tv\_updates.setVisibility(View.*GONE*);  
 tv\_labelupdates.setVisibility(View.*GONE*);  
 }  
 }  
 }  
 }else {  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);  
 builder.setTitle("This app requires background location in order to work correctly!");  
  
  
 builder.setPositiveButton("OK" , ( dialog , which ) -> {  
 Intent intent = new Intent(android.provider.Settings.*ACTION\_LOCATION\_SOURCE\_SETTINGS*);  
 startActivity(intent);  
 });  
 builder.setNegativeButton("Cancel" , ( dialog , which ) -> {  
 dialog.cancel();  
 android.os.Process.*killProcess*(android.os.Process.*myPid*());  
 });  
  
 builder.show();  
 }  
 }  
 @RequiresApi(api = Build.VERSION\_CODES.*O*)  
 @Override  
 protected void onDestroy ( ) {  
 super.onDestroy();  
 Intent intent = new Intent(getApplicationContext(),ForegroundService.class);  
 startForegroundService(intent);  
 }  
  
 public boolean isConnected() {  
 boolean connected = false;  
 try {  
 ConnectivityManager cm = (ConnectivityManager)getApplicationContext().getSystemService(Context.*CONNECTIVITY\_SERVICE*);  
 NetworkInfo nInfo = cm.getActiveNetworkInfo();  
 connected = nInfo != null && nInfo.isAvailable() && nInfo.isConnected();  
 return connected;  
 } catch (Exception e) {  
 Log.*e*("Connectivity Exception", e.getMessage());  
 }  
 return connected;  
 }  
  
}

В Main Activity класа има няколко метода: onCreate(), onStart(),onDestroy(). В началото на класа се взима потребителя от FirebaseAuth системата чрез mAuth.getCurrentUser() и се записва в обекта user който е от тип FirebaseUser.

При инициализиране на класа се изпълнява метода onCreate() в който са инициализирани 4-ри OnClickListener за 4-те бутона: „Open Map“(Който бутон при натискане отваря Maps Activity и предава информация за локацията с intent.putExtra() latitude и longitude), „Sign Out“, „Location Updates“, „GPS/Save Power“ и locationCallback обект от тип LocationCallback който при onLocationResult() извиква ф/те updateUI(location) и updateDB(location).

След това се стартира onStart() метода който проверява дали е разрешено на приложението ACCESS\_BACKGROUND\_LOCATION което разрешение е да се използва винаги. Ако не е разрешено се инициализира Аlertdialog.builder който изкарва на екрана тип pop up съобщение на потребителя, че приложението се нуждае от това разрешение, ако потребителят натисне „ОК“ го отвежда до настройките на устройството където собственоръчно трябва да даде разрешение, ако натисне „Cancel” бутона, се изпълнява отново докато не се даде разрешение. При дадено разрешение се премахва ForegroundService.class ако има такъв активен чрез stopService(); и се взима типа на потребителя от Intent.extra, ако няма такъв userType се записва „child“ и се ограничават функционалностите на приложението. Ако потребителя е родител (parent) се показват бутоните: „Open Map“, „Location Updates“, като последните два могат да се използват и от двата типа потребители „Sign Out“, „GPS/Save Power“. Бутоните „Location Updates“ и „GPS/Save Power“ се превключват в “On” режим, като тяхната цел е да се стартира/спре ъпдейта на локацията и да се превключи от GPS локация, към локация използваща WIFI или телефонни кули чрез функциите startLocationUpdates(), stopLocationUpdates(), a за GPS/Save Power просто се сменя приоритета на locationRequest(PRIORITY\_HIGH\_ACCURACY, PRIORITY\_BALANCED\_POWER\_ACCURACY). Извиква се и функцията UpdateGPS(), която проверява за разрешение от потребителя и ако е дадено взима последната локация на устройството и я предава на функциите UpdateUIValues (Тя има за цел да обнови информацията на екрана в Main Activity, като се инициализира и Geocoder който показва адрес на който се намира самото устройството, ако такъв съществува) и UpdateDBValues (За която специално се проверява дали устройството е свързано към интернет чрез функцията isConnected() и ако не е функцията не се изпълнява. Самата функция взима Latitude и Longitude от подадената локация и обновява информацията за дадения потребител в базата данни като се използва качване на данни чрез хеш мап за дадения потребител). Накрая имаме метода onDestroy() който се активира когато приложението е затворено и не работи, той стартира класа ForegroundService.class.

***4.4. Maps Activity class***

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {  
  
 private GoogleMap mMap;  
 private String m\_Text = "";  
 Button addButton;  
  
 FirebaseFirestore db = FirebaseFirestore.*getInstance*();  
 FirebaseAuth mAuth = FirebaseAuth.*getInstance*();  
 DocumentReference document = null;  
 CollectionReference friendsMarkersCollection = db.collection("users/"+mAuth.getCurrentUser().getEmail()+"/friendsMarkers");  
 Map<String, Object> friendsMarkers = new HashMap<>();  
  
 @Override  
 protected void onCreate ( Bundle savedInstanceState ) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_maps*);  
 SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()  
 .findFragmentById(R.id.*map*);  
 mapFragment.getMapAsync(this);  
  
 }  
 @Override  
 public void onMapReady ( GoogleMap googleMap ) {  
  
 mMap = googleMap;  
 Bundle extras = getIntent().getExtras();  
  
 if (extras != null) {  
 double latitude = extras.getDouble("Latitude");  
 double longitude = extras.getDouble("Longitude");  
 LatLng marker = new LatLng(latitude , longitude);  
 mMap.addMarker(new MarkerOptions().position(marker).title("You are here"));  
 mMap.moveCamera(CameraUpdateFactory.*newLatLng*(marker));  
 }  
  
 addButton = findViewById(R.id.*addButton*);  
  
 addButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick ( View view ) {  
 AlertDialog.Builder builder = new AlertDialog.Builder(MapsActivity.this);  
 builder.setTitle("Enter email to trace");  
 final EditText input = new EditText(MapsActivity.this);  
 input.setInputType(InputType.*TYPE\_CLASS\_TEXT* | InputType.*TYPE\_CLASS\_TEXT*);  
 builder.setView(input);  
  
 builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 m\_Text = input.getText().toString();  
  
 FirebaseUser user = mAuth.getCurrentUser();  
 friendsMarkers.put("email",m\_Text);  
 friendsMarkersCollection.document(m\_Text).set(friendsMarkers);  
 Random random = new Random();  
 int randomNum = random.nextInt(10 - 0 + 1) + 0;  
 newMarker(m\_Text,randomNum);  
 }  
 });  
 builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 dialog.cancel();  
 }  
 });  
  
 builder.show();  
  
 }  
 });  
  
 FirebaseFirestore.*getInstance*()  
 .collection("users/" + mAuth.getCurrentUser().getEmail() + "/friendsMarkers")  
 .get()  
 .addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 List<DocumentSnapshot> myListOfDocuments = task.getResult().getDocuments();  
 for (DocumentSnapshot document:myListOfDocuments  
 ) {  
 Random random = new Random();  
 int randomNum = random.nextInt(10 - 0 + 1) + 0;  
 Toast.*makeText*(getApplicationContext() , "Automatically added marker for saved friend " + document.get("email").toString(), Toast.*LENGTH\_SHORT*).show();  
 newMarker(document.get("email").toString(),randomNum);  
  
 }  
 }  
 });  
  
 }  
  
 private void newMarker ( String string , int rand) {  
  
 BitmapDescriptor markerColor;  
  
 switch (rand) {  
 case 1:  
 markerColor = BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.*HUE\_AZURE*);  
 break;  
 case 2:  
 markerColor = BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.*HUE\_BLUE*);  
 break;  
 case 3:  
 markerColor = BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.*HUE\_CYAN*);  
 break;  
 case 4:  
 markerColor = BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.*HUE\_GREEN*);  
 break;  
 case 5:  
 markerColor = BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.*HUE\_MAGENTA*);  
 break;  
 case 6:  
 markerColor = BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.*HUE\_ORANGE*);  
 break;  
 case 7:  
 markerColor = BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.*HUE\_RED*);  
 break;  
 case 8:  
 markerColor = BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.*HUE\_ROSE*);  
 break;  
 case 9:  
 markerColor = BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.*HUE\_VIOLET*);  
 break;  
 case 10:  
 markerColor = BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.*HUE\_YELLOW*);  
 break;  
 default:  
 markerColor = BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.*HUE\_RED*);  
 break;  
 }  
  
 if (string!=null) {  
 if (string.contains("@")&&string.contains(".")) {  
 DocumentReference docRef = db.collection("users").document(string);  
 docRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {  
 @Override  
 public void onComplete ( @NonNull Task<DocumentSnapshot> task ) {  
 if (task.isSuccessful()) {  
 DocumentSnapshot document = task.getResult();  
  
 if (document.exists()) {  
 double latitude = document.getDouble("latitude");  
 double longitude = document.getDouble("longitude");  
 Map<String, Object> timestamp = (Map<String, Object>) document.get("time");  
 String minute;  
 if ((Long)timestamp.get("minute")<10){  
 minute = "0"+timestamp.get("minute").toString();  
 }else{  
 minute = timestamp.get("minute").toString();  
 }  
  
 String time =  
 timestamp.get("dayOfMonth").toString()+"."  
 +timestamp.get("monthValue").toString()  
 +"." + timestamp.get("year").toString()  
 +" " + timestamp.get("hour").toString()  
 +":"+ minute;  
  
 // Add a marker and move the camera  
 LatLng marker = new LatLng(latitude , longitude);  
 mMap.addMarker(new MarkerOptions().position(marker).title(document.getString("email") + " At: "+ time).icon(markerColor));  
 mMap.moveCamera(CameraUpdateFactory.*newLatLng*(marker));  
  
  
 } else {  
 Toast.*makeText*(MapsActivity.this , "No such email in database!" , Toast.*LENGTH\_SHORT*).show();  
 }  
  
 } else {  
 Log.*d*(*TAG* , "get failed with " , task.getException());  
 }  
 }  
 });  
 } else {  
 Toast.*makeText*(MapsActivity.this,"Enter valid email!", Toast.*LENGTH\_SHORT*).show();  
 }  
 }else {  
  
 Toast.*makeText*(MapsActivity.this,"Enter valid email!" + string, Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
}

Maps Activity класа има един метод onMapReady(), който при вече заредило activity се стартира и се инициализира onClickListener() на бутона “ADD FRIEND” който при натискане стартира AlertDialog.Builder който извежда на екрана текстово поле и изисква да се въведе имейл на друг потребител, при правилно въведен имейл се генерира случайно число от 1 до 10 и се стартира функцията newMarker(m\_Text, randomNum), която от своя страна, избира цвят за новия маркер чрез switch case за който всъщност използваме променливата randomNum, след това се добавя нов маркер за въведения имейл и се запаметява в базата данни за този потребител. Преди това се проверяват координатите на този потребител в базата данни чрез предоставения имейл и се задава избрания цвят на маркера. След повторно влизане в Maps Activity се добавят всички запазени приятели на потребителя. Чрез for each цикъл се обхождат всички документи в колекцията friendsMarkers на дадения потребител.

***4.5. Foreground Service class***

public class ForegroundService extends Service {  
 PeriodicWorkRequest.Builder dataSendBuilder;  
 FirebaseAuth mAuth = FirebaseAuth.*getInstance*();  
  
  
 @Override  
 public void onCreate() {  
 super.onCreate();  
 startBackgroundUpdate();  
 }  
  
 @Override  
 public int onStartCommand(Intent intent, int flags, int startId) {  
 createNotificationChannel();  
 Intent intent1 = new Intent(this, MainActivity.class);  
 PendingIntent pendingIntent = PendingIntent.*getActivity*(this,0,intent1,0);  
 Notification notification = new NotificationCompat.Builder(this,(mAuth.getCurrentUser().getEmail()))  
 .setContentTitle("diploma")  
 .setContentText("This app is using your location!")  
 .setSmallIcon(R.mipmap.*ic\_launcher*)  
 .setContentIntent(pendingIntent)  
 .build();  
  
 startForeground(1,notification);  
  
 return *START\_STICKY*;  
 }  
  
 public void startBackgroundUpdate () {  
 dataSendBuilder =  
 new PeriodicWorkRequest.Builder(RxUpdateWorker.class, 15,  
 TimeUnit.*SECONDS*);  
 PeriodicWorkRequest dataSendWork = dataSendBuilder.build();  
 WorkManager.*getInstance*(getApplicationContext()).enqueue(dataSendWork);  
 }  
  
 private void createNotificationChannel ( ) {  
  
 if(Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*){  
 NotificationChannel notificationChannel = new NotificationChannel(  
 (mAuth.getCurrentUser().getEmail()),"ForegroundNotification",NotificationManager.*IMPORTANCE\_DEFAULT*);  
 NotificationManager manager = (NotificationManager) getApplicationContext().getSystemService(Context.*NOTIFICATION\_SERVICE*);  
 manager.createNotificationChannel(notificationChannel);  
 }  
 }  
  
 @Override  
 public void onDestroy() {  
 super.onDestroy();  
 stopForeground(true);  
 stopSelf();  
 }  
  
 @Nullable  
 @Override  
 public IBinder onBind ( Intent intent ) {  
 return null;  
 }  
}

ForegroundService класа служи за извикване, както и за поддържане на класа RxUpdateWorker като се използва изпращането на нотификация и връща return START\_STICKY; което прави невъзможно нотификацията да бъде премахната. В onStartCommand() метода се инициализира нотификационен канал с функцията createNotificationChannel и нотификация в този канал чрез NotificationCompat.Builder която показва на потребителя, че приложението използва локацията на устройството. В onCreate() метода се стартира функцията startBackgroundUpdate() която от своя страна инициализира PeriodicWorkRequest.Builder на който му здаваме време през което да се извършва работата и му посочваме класа RxUpdateWorker и след това го стартираме. Накрая имаме OnDestroy() метода който се активира когато премахнем ForegroundService.class, това се случва като стартираме MainActivity.class или по конкретно в неговия метод onStart().

***4.6. Rx Update Worker class***

public class RxUpdateWorker extends Worker {  
 FusedLocationProviderClient fusedLocationProviderClient;  
 LocationRequest locationRequest;  
 private FirebaseAuth mAuth = FirebaseAuth.*getInstance*();  
 LocationCallback locationCallback;  
  
 public RxUpdateWorker(@NonNull Context context, @NonNull WorkerParameters params) {  
 super(context, params);  
 }  
  
 @SuppressLint("MissingPermission")  
 @RequiresApi(api = Build.VERSION\_CODES.*O*)  
 @Override  
  
 public Result doWork() {  
  
 locationRequest = LocationRequest.*create*();  
   
locationRequest.setPriority(LocationRequest.*PRIORITY\_BALANCED\_POWER\_ACCURACY*);  
  
 fusedLocationProviderClient = LocationServices.*getFusedLocationProviderClient*(getApplicationContext());  
 locationCallback = new LocationCallback() {  
 @Override  
 public void onLocationResult ( @NonNull LocationResult locationResult ) {  
 super.onLocationResult(locationResult);  
 Location location = locationResult.getLastLocation();  
 uploadLocation(location);  
 }  
 };  
  
 if (ActivityCompat.*checkSelfPermission*(getApplicationContext() , Manifest.permission.*ACCESS\_FINE\_LOCATION*) == PackageManager.*PERMISSION\_GRANTED*) {  
 fusedLocationProviderClient.requestLocationUpdates(locationRequest , locationCallback ,Looper.*getMainLooper*());  
 updateGPS();  
 }  
 else {  
 displayNotification();  
 }  
  
 return Result.*retry*();  
 }  
  
 @SuppressLint("RestrictedApi")  
 @RequiresApi(api = Build.VERSION\_CODES.*O*)  
 private void updateGPS () {  
  
 fusedLocationProviderClient = LocationServices.*getFusedLocationProviderClient*(getApplicationContext());  
  
 if (ActivityCompat.*checkSelfPermission*(getApplicationContext(), Manifest.permission.*ACCESS\_FINE\_LOCATION*) == PackageManager.*PERMISSION\_GRANTED*) {  
  
 fusedLocationProviderClient.getLastLocation().addOnSuccessListener(getBackgroundExecutor() , new OnSuccessListener<Location>() {  
  
 @RequiresApi(api = Build.VERSION\_CODES.*O*)  
 @Override  
 public void onSuccess ( Location location ) {  
 if (location != null) {  
 if (isConnected()) {  
 uploadLocation(location);  
 }  
 } else {  
 displayNotification();  
 }  
 }  
 });  
  
 }  
 else {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {  
 displayNotification();  
 }  
  
 }  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.*O*)  
 private void uploadLocation(Location location) {  
 Map<String, Object> dbUser = new HashMap<>();  
 mAuth = FirebaseAuth.*getInstance*();  
 FirebaseUser user = mAuth.getCurrentUser();  
 FirebaseFirestore db = FirebaseFirestore.*getInstance*();  
 CollectionReference users = db.collection("users");  
 LocalDateTime localDateTime = null;  
  
 double latitude = location.getLatitude();  
 double longitude = location.getLongitude();  
  
 dbUser.put("latitude", latitude);  
 dbUser.put("longitude", longitude);  
 dbUser.put("email", user.getEmail());  
 dbUser.put("time", localDateTime.*now*());  
 users.document(user.getEmail()).set(dbUser);  
  
 db.collection("users").document(user.getEmail())  
 .update(dbUser)  
 .addOnSuccessListener(new OnSuccessListener<Void>() {  
 @Override  
 public void onSuccess(Void aVoid) {  
 Log.*d*(*TAG*, "DocumentSnapshot successfully written!");  
 }  
 })  
 .addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
 Log.*w*(*TAG*, "Error writing document", e);  
 }  
 });  
  
 }  
 @RequiresApi(api = Build.VERSION\_CODES.*O*)  
 private void displayNotification () {  
 NotificationManager manager = (NotificationManager) getApplicationContext().getSystemService(Context.*NOTIFICATION\_SERVICE*);  
 NotificationChannel channel = new NotificationChannel(mAuth.getCurrentUser().getEmail(),"diploma",NotificationManager.*IMPORTANCE\_DEFAULT*);  
 manager.createNotificationChannel(channel);  
 NotificationCompat.Builder builder = new NotificationCompat.Builder(getApplicationContext(), mAuth.getCurrentUser().getEmail())  
 .setContentTitle("Error message")  
 .setContentText("Cannot update user data to db!").setSmallIcon(R.mipmap.*ic\_launcher*);  
 manager.notify(1,builder.build());  
 }  
  
 public boolean isConnected() {  
 boolean connected = false;  
 try {  
 ConnectivityManager cm = (ConnectivityManager)getApplicationContext().getSystemService(Context.*CONNECTIVITY\_SERVICE*);  
 NetworkInfo nInfo = cm.getActiveNetworkInfo();  
 connected = nInfo != null && nInfo.isAvailable() && nInfo.isConnected();  
 return connected;  
 } catch (Exception e) {  
 Log.*e*("Connectivity Exception", e.getMessage());  
 }  
 return connected;  
 }  
  
}

RxUpdateWorker класа е Background worker който всъщност е процес стартиран в задната част на приложението, неговата цел е постоянно да се праща информация на базата данни стига да има налична интернет връзка. Стартирайки класа се стартира и метода doWork(), в който метод е инициализирана заявка locationRequest от тип LocationRequest, както и locationCallback който onLocationResult() извиква функцията uploadLocation(location). Функцията uploadLocation както и в предишните класове така и тук ъпдейтва локацията на дадения потребител в базата данни чрез хеш мап (хеш таблица). В случай че не може да се качи информацията се показва нотификация на устройството която казва на потребителя че има проблем с качването на локацията, това става чрез displayNotification() функцията. UpdateGPS() функцията проверява дали приложението има позволение от потребителя да използва локацията на устройството и устройството има интернет връзка чрез функцията isConnected() и след това отново се извиква функцията uploadLocation(location).

**5. Заключения**

След представените резултати, може да се заключи, че целите, поставени в началото на дипломната работа, са постигнати. Изготвената дипломна работа е работеща програма, която се нуждае от реален екип и малко повече задълбочена разработка за да се постигне световно ниво, както и да се помисли за подобрения

Целите в близкото бъдеще за тази програма са да бъде изтествана в по-продължителен период от време, за да се намерят пропуски в кода или нови по лесни начини за реализация на функциите й както и да се добавят допълнителни функционалности.

**6. Източници**

1. Firebase Firestore & Firebase Authentication –

<https://firebase.google.com>

2. Android Studio -

<https://developer.android.com>

3. Java –

<https://www.java.com/en/>

4. Diploma GitHub link to project, apk and word files –

<https://github.com/DanielDonkov/Diploma.git>