



Android
Studio



פרויקט גמר בהנדסת תוכנה

חלופה – תכנון ותכנות מערכות בטלפונים ניידים
תחת מערכת הפעלה Android

שם האפליקציה/פרויקט: הטרמפיאדה – TheTrempiada

שם התלמיד: זיו נדל

תוכן עניינים

3.....	מבוא
3.....	הרקע לפרויקט
4.....	תהליך המחקר
7.....	אתגרים מרכזיים
7.....	הצגת פתרונות לבעיה
9.....	תרשים סכימטי ו-Activities (מדריך למפתח)
10.....	Main Menu (Navigation Drawer)
11.....	BaseActivity
13.....	MainActivity
17.....	LoginActivity & SignUpActivity
19.....	SettingsActivity & ProfileSettingsActivity
21.....	Menu Fragments
21.....	Profile Fragment
22.....	Contact Fragment
22.....	About Fragment
23.....	Exit Fragment
24.....	בסיס נתונים
27.....	פירוט מחלקות עזר ועצמים (מדריך למפתח)
32.....	מדריך למשתמש
42.....	רפלקציה
43.....	ביבליוגרפיה
44.....	נספחים

מבוא

הספר הזה מכיל פירוט נרחב של כל מה שקשור באפליקציה שתכננתי במסגרת מגמת הנדסת תוכנה. האפליקציה מיועדת לטלפונים ניידים (סמארטפונים) תחת מערכת ההפעלה Android.

הספר מתאר בעיקר פירוט טכני של מבנה ופעולת האפליקציה: תרשימים של המחלקות והחלונות באפליקציה, פירוט על שימוש בבסיסי נתונים ומחלקות, ושימוש בשירותים שונים המוצעים ע"י מערכת ההפעלה אנדרואיד. בנוסף בספר מצורפות הוראות הפעלה מפורטות לאפליקציה ורפלקציה אישית.

הרקע לפרויקט

שם הפרויקט: הטרמפיאדה (The Trempiada).

תיאור הפרויקט: האפליקציה שלי מציעה שירותים **טרמפים**: עוברי דרך יכולים להירשם בתור משתמשים שמציעים טרמפ או מבקשים טרמפ באפליקציה. האפליקציה מעדכנת בזמן אמת מציעי טרמפים על טרמפיסטים במסלול הנסיעה שלהם (שאותו הזינו המשתמשים מראש), ומנגד מעדכנת בזמן אמת טרמפיסטים על משתמשים שעשויים לעבור בדרך ולאסוף אותם, או על משתמשים שהסכימו לאסוף אותם.

באפליקציה יש פיצ'רים נוספים כגון תפריטי התאמה אישית, בניית פרופיל אישי ובעתיד אף דירוג של משתמשים על מנת לאמת את אמינותם.

האפליקציה חנימית לחלוטין, ולכן לא תהיה אופציה לנהגים לגבות תשלום תמורת השירות שלהם. בנוסף האפליקציה **לא** מציעה לנהגים לחרוג ממסלולם כדי לאסוף טרמפיסטים, אולם מציעה לטרמפיסטים לשנות את מיקומם (בטווח הגיוני) על מנת לתפוס טרמפ.

בעקבות מורכבות האפליקציה, אני כרגע מגדיר אותה בתור "אבטיפוס". רוב התכונות המתוכננות פועלות, אך לא כולן, ואני מתכנן להרחיב את האפליקציה בזמני הפנוי מעבר למגמת הנדסת תוכנה ובית הספר התיכון.

קהל היעד: האפליקציה מתאימה לכולם, נהגים וטרמפיסטים כאחד. הייעוד הוא שישתמשו באפליקציה טרמפיסטים, בני נוער, חיילים, אנשים ללא רכב פרטי ואנשים בכלל. מגבלת גיל ואימות לא נוספו עדיין לאפליקציה.

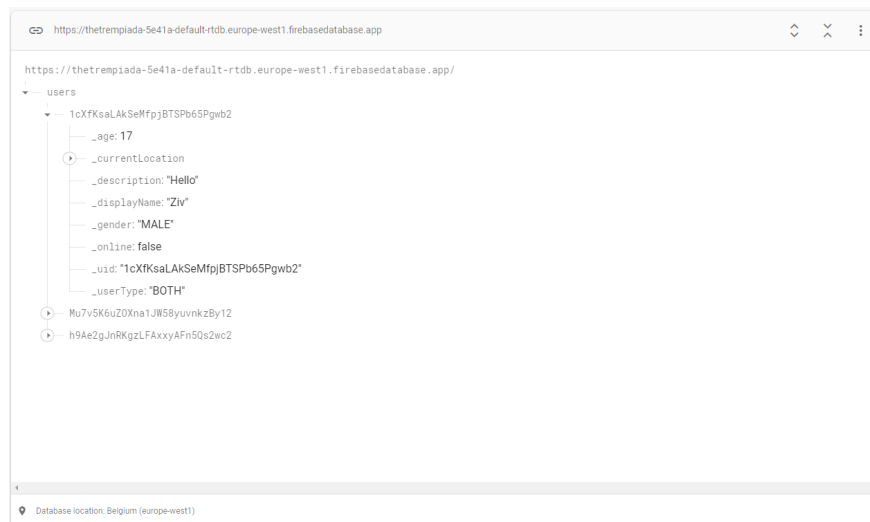
הסיבות לבחירת הנושא:

אני מתגורר ביישוב יחסית מרוחק, ולכן התחבורה ביישוב מהווה קושי משמעותי. תדירות התחבורה הציבורית נמוכה למדי, ולכן אני מסתמך הרבה על "טרמפים" עם אנשים זרים כדי להגיע ליישוב. הרהרתי בכך שבעידן טכנולוגי כמו שלנו, שבו יש פתרון טכנולוגי לכל בעיה ואתגר, עדיין לא קיים פתרון שימצה וישפר את האפשרות של טרמפים, וכי עדיין מסתמכים על הושטת יד בתחנת האוטובוס בתקווה שאדם יסכים לעצור. לכן, החלטתי לנסות להרים את הכפפה וליצור אפליקציה – פתרון טכנולוגי – שיאפשר תקשורת בין טרמפיסטים לנהגים המעוניינים לעזור לאנשים אחרים בחינם, ולתת להם טרמפ.

תהליך המחקר

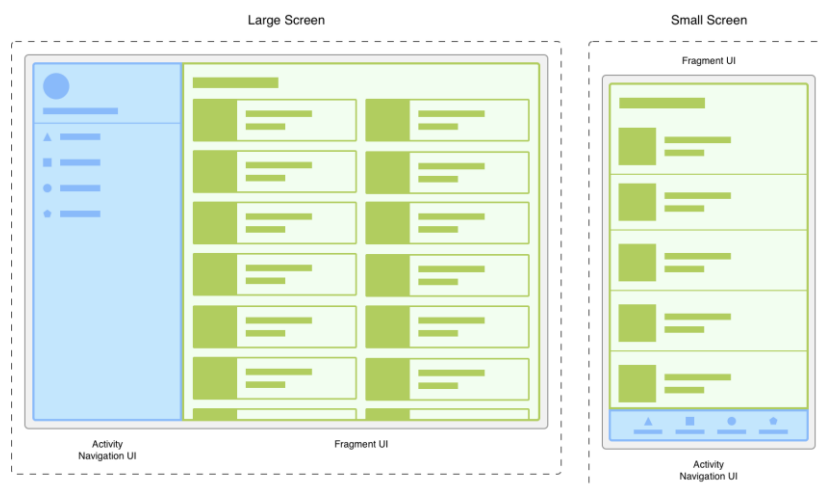
האפליקציה שלי מורכבת מאוד מבחינה טכנית, ולכן לצורך יישומה נזקקתי ללמוד נושאים רבים שמעבר לתוכנית הלימודים:

- Google Firebase Database: לצורך מימוש הפרויקט שלי ואחסון המידע השתמשתי ב-Firebase Database של גוגל. השתמשתי בשלושה שירותים שמספק פיירבייס. הראשון והמרכזי הוא Firebase Realtime Database. מדובר ב-DB מבוסס פורמט key-value לצורך אחסון נתונים, כאשר ה-value יכול להיות גם עצם, ושיטה זו הייתה לי נוחה מאוד. ה-Firebase Realtime Database מתעדכן בזמן אמת, ולכן בחרתי להשתמש בו לאחסון פרטי המשתמשים ופרטי האפליקציה שלי. השירות השני שהשתמשתי בו הוא Firebase Authentication, שאפשר לי ליצור את המשתמשים ולאפשר התחברות מאובטחת. לצורך אחסון תמונות הפרופיל של המשתמשים השתמשתי ב-Firebase Storage. בנוסף, פיירבייס מספק קונסולה אינטרנטית נוחה במיוחד למעקב אחר השינויים והערכים בדאטאבייס. פיירבייס מספר גם קובץ לניהול הרשאות על מנת לאבטח את מבנה הנתונים.



הקונסולה של Realtime Database - שיטת key-value

- Activity Life Cycle: מאחר והאפליקציה שלי פועלת בחלקית גם ברקע, ויש משמעות ליציאה מהאפליקציה לצורך ניתוק והשהייה של משתמשים, למדתי את הנושא של "מחזור החיים" של האקטיביטי באנדרואיד. מדובר בשימוש נרחב ועמוק יותר בפעולות כמו onCreate, onStop, onResume, onPause ונוספות.
- Fragment: פרגמנט באנדרואיד הוא חלק מהעיצוב של התוכנה, ניתן לקשר אליו קובץ XML לצורכי עיצוב וכן קובץ Java לצורך תכנות, ולפיכך הוא משמש בתפקיד דומה לאקטיביטי. פרגמנט חייב להיות "מולבש" על אקטיביטי או על פרגמנט אחר. שימוש בפרגמנטים אפשר לי להיות יותר גמיש עיצובית ולחסוך באקטיביטיז וברענונים מיותר של האפליקציה. הפרגמנט הבולט ביותר שהשתמשתי בו הוא המפה המוצגת ב-MainActivity.

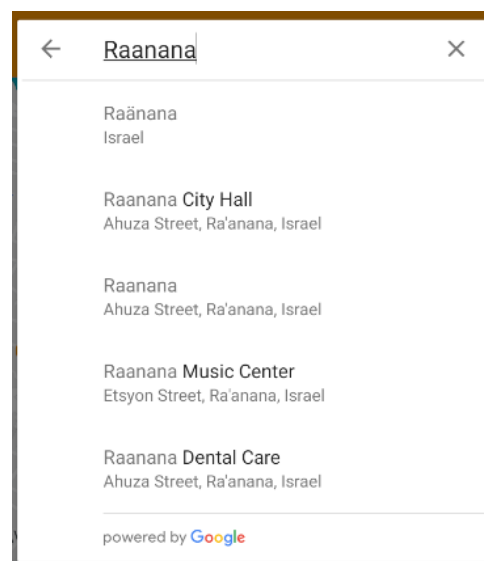


דוגמא לשימוש בפרגמנט בתוך אקטיביטי

- Enums: השתמשתי במחלקות מסוג enum, שאינן כלולות בתוכנית הלימודים, לצורך יצירת עצמים בעלי אופציות קבועות לערכים (לדוגמא gender).
- Material UI Components: לצורך עיצוב האפליקציה, למדתי על המחלקה Material UI המספקת עיצובים מוכרים ממערכת ההפעלה אנדרואיד. לקחתי ממנה עיצובי דיאלוגים, תפריטים, כפתורים, חלונות קופצים ועוד.
- Regular Expressions: ביטוי רגולרי (בקיצור: Regex), הוא רצף תווים שמגיד תבנית חיפוש. מדובר בשפה רגולרית שמוגדרת כשתי מחרוזות או יותר הכפופות לתקנות תחביר מסוימות. השתמשתי בביטויים אלו באפליקציה שלי בעיקר לצורך יצירת תבנית של שם משתמש וסיסמא, לדוגמא:

```
if(s.toString().matches("(?=.*[a-zA-Z])\\w{6,16}$"))
```

 לבדיקת תקינות של סיסמא.
- Google Places SDK: השתמשתי בשירות של גוגל הנקרא Google Places המאפשר לי לספק למשתמשים אפשרות לבצע חיפוש של מקומות ולבחור יעד בהתבסס על מאגר המידע של גוגל וגוגל מפות.



Google Places SDK - המחשה מתוך האפליקציה שלי

- Callbacks: Callbacks מאפשרות למפתח "להודיע" למחלקות שפעולה כלשהי סיימה ביצוע, וכך לעבור לביצוע דבר אחר או לאשר התחלת ביצוע פעולה מסוימת. משתמשים הרבה ב-Callbacks לצורך events כמו onClick ועוד. השתמשתי בפרויקט שלי ב-Callbacks שיצרתי לצורך דברים רבים, לדוגמא להודיע לתוכנית שהמשתמש נתן הרשאה לשימוש במיקום, כדי להציג את מיקומו הנוכחי, או להודיע לתוכנית שהמשתמש לא נגע במסך 5 דקות ולכן להגדיר את סטטוס המשתמש כ-Offline.

השתמשתי בשירותים נוספים, שחלקם כלולים באופן כלשהו בתוכנית הלימודים, כמו Google Maps SDK. מעבר למחקר לצורך יישום טכני של האפליקציה, סקרתי את המצב הקיים בשוק ומצאתי מספר פתרונות הדומים לאפליקציה שהצעתי, כגון "ווייז קארפול", אך רובם כללו תשלום לנהג שנותן את הטרמפ. אני יכול להעיד שבתור אדם שמשתמש בטרמפים רבות (מכיוון שאני גר ביישוב מרוחק מבית הספר), לא מצאתי באפליקציות הקיימות פתרון, ולא שמעתי על הרבה אנשים שמשתמשים בהם. אני שואף ליצור אפליקציה שמיישה, שתהווה פתרון ממשי, שאינו קיים כיום בשוק האפליקציות.

אני רציתי ליצור שירות חינמי, כמו שלא עולה כסף להרים את היד לבקשת טרמפ בתחנת האוטובוס. האפליקציות הקיימות בשוק לא נתמכו בישראל או שאינן היו פרקטיות, ולפיכך חסרות תועלת בשבילי ובשביל הסביבה שלי.

אתגרים מרכזיים

האתגרים המרכזיים בפרויקט שלי היו בעיקר בפן הטכני. היה קשה לייצר את הקישוריות בין המשתמשים וליצור פלטפורמת אונליין שיכולה לספק את המטרות שלה. בנוסף היה קשה להקפיד על אבטחת המשתמשים כדי לא ליצור תקלות, ומאחר שהאפליקציה שלי כה מורכבת, היה קשה במיוחד לנטר ולתקן כל בעיה. פעמים רבות מצאתי את עצמי יושב שעות רבות בניסיון לפתור בעיה, ושעות רבות רק כדי להבין איפה נמצאת הבעיה. בעיות כאלה מתעצמות כשמדובר באפליקציה כה מורכבת וגדולה, המשתמשת בכל כך הרבה שירותים ואפשרויות בזמנית.

כמו כן, הייתה לי שנה עמוסה ולכן היה לי קשה להקדיש מספיק זמן לסיום הפרויקט, שעקב היקפו דרש המון זמן. לכן האפליקציה שאני מגיש היא רק אב טיפוס ואינה מכילה את כל האפשרויות המתוכננות.

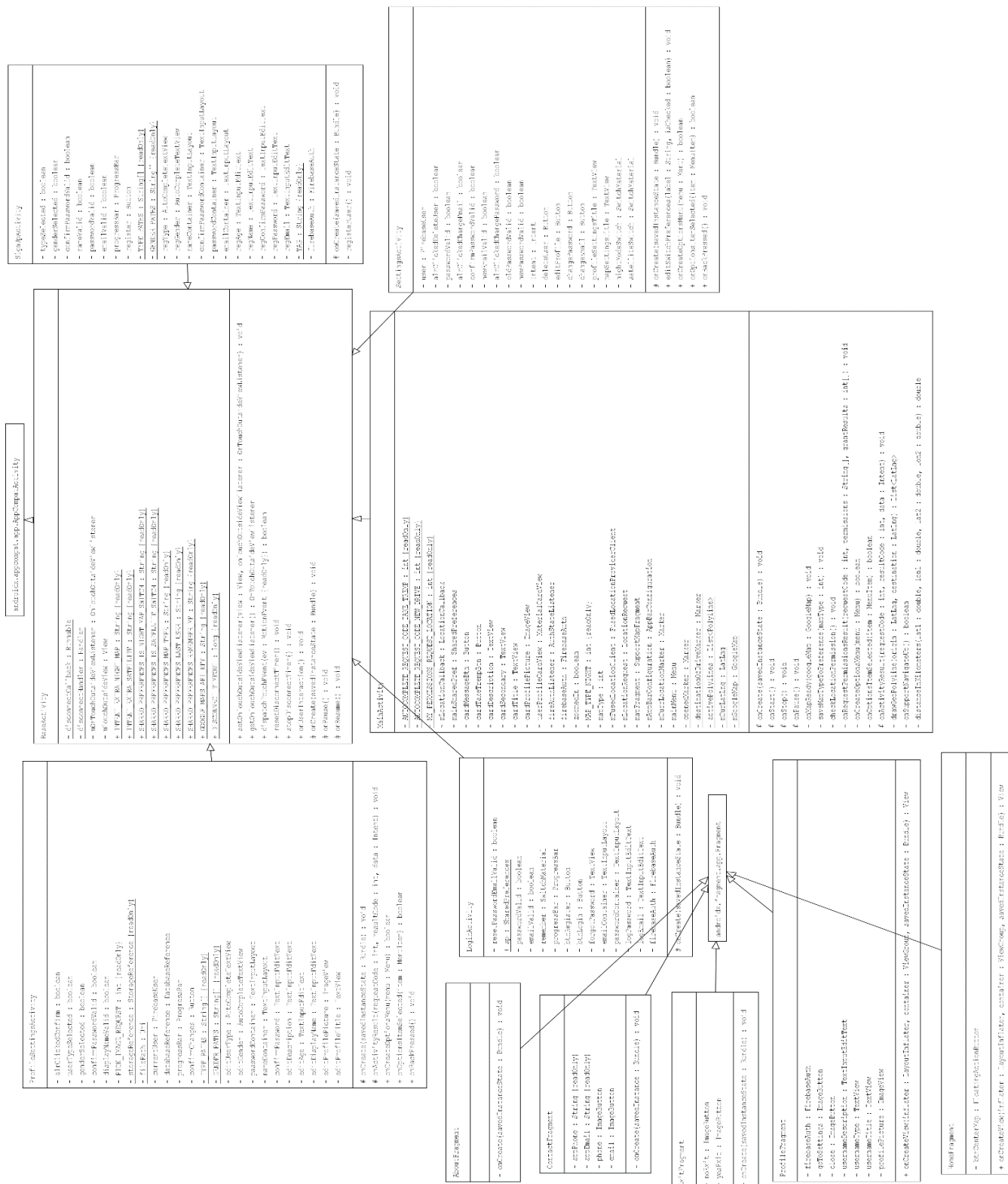
האתגר המרכזי ביותר בכל אפליקציה, לדעתי, הוא לספק את הצורך המקורי שלשמה היא תוכננה, ואני מנסה לספק את הצורך בחיבור של תרבות הטרמפים לעולם הטכנולוגי ולספק פתרון שיאפשר לאנשים לשפר את איכות החיים שלהם במקומות פריפריאליים בהם התחבורה הציבורית אינה מפותחת.

הצגת פתרונות לבעיה

כמו שציינתי, האפליקציה שלי מיועדת לספק פתרון לאתגרים ולבעיות שציינתי בסעיף הקודם, ע"י יצירת תקשורת בין הטרמפיסטים לנהגים באופן שיאפשר לנהגים לא לחרוג ממסלולם בשום צורה (כי אחרת הם אינם ירצו לקחת חלק בשירות זה), ובאופן שיאפשר לטרמפיסטים להגיע הכי קרוב לנקודות האיסוף ולייצר נקודת הורדה שקרובה ככל האפשר

לייעדם. כל התקשורת באפליקציה יוצרה על גבי Firebase שבהיותו שירות מבוסס, גדול ומתוחכם מסוגל לספק את כל צורכי התקשורת שהאפליקציה צריכה. היות והאפליקציה מורכבת, אני מתכנן לפתור את הבעיות שפיתוחה מעורר ע"י שימוש בכמה שיותר כלים מוכנים, לדוגמא שירותים של Google, כי אין סיבה להמציא את הגלגל מחדש כאשר הפתרון כבר קיים, וזה יחסוך עומס של קוד על האפליקציה ובעיות קריאות, וכן יאפשר לי להתמקד בחלקים שעלי לפתח בעצמי. השתמשתי בגרסת API 26 להרצת האפליקציה (התואמת לגרסת אנדרואיד 8.0 ומעלה), אך ניתן להריץ את האפליקציה גם בגרסאות נמוכות יותר (עד API 19). הרצת האפליקציה בשלבי התכנון והבנייה הייתה על אפולטור ומכשיר סלולרי כאחד. שפת האפליקציה היא אנגלית בשלב פיתוח זה.

(מדריך למפתח)



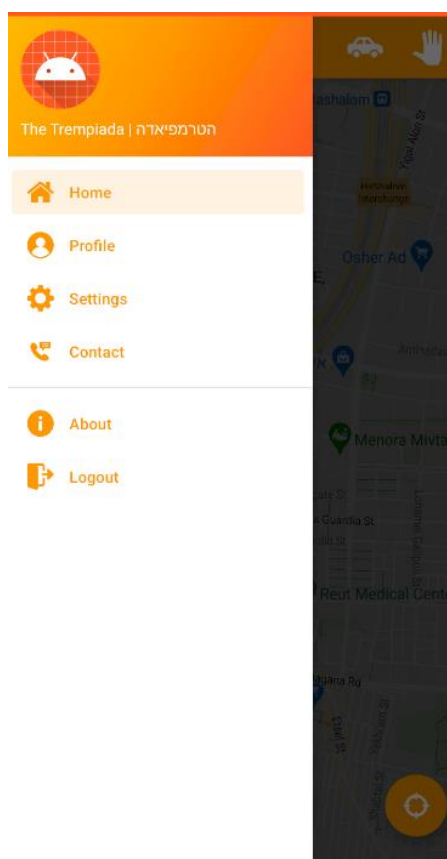
ניתן לראות סכמת UML הכוללת רק את האקטיביטיז (והפרגמנטים) של האפליקציה והקשרים ביניהם. בהמשך חלק זה של הספר אוסיף תרשימי ירושה וקשרים לכל אקטיביטי בנפרד.

ניתן ללחוץ על הקישור הבא לפתיחת תרשים ה-UML בגודל מלא:

<https://ibb.co/nMXqDZM>

בפרק זה יש תיאור מלא של כל האקטיביטיז והמסכים באפליקציה, קוד מלא של המחלקות ניתן לראות בפרק נספחים בספר זה.

Main Menu (Navigation Drawer)



ליצירת התפריט הראשי השתמשתי בטכנולוגיה שלא למדנו – Navigation Drawer. מדובר במרכיב UI המשמש לניווט באפליקציה בשיטה שונה ממעבר בין אקטיביטיז שונים באמצעות Intent. ישנם מספר קבצי XML האחראים על הפונקציונליות של התפריט. אחד אחראי על העיצוב, אחד הוא קובץ XML רגיל של פרטי menu שלמדנו בכיתה, ואחד משמש לניווט, כלומר ניתן להכניס בו הפניות לאקטיביטי או לפרגמנט, וע"י התאמה של האקטיביטי המתאים לכפתור המתאים ניתן לעבור לחלון זה ללא שימוש ב-Intent. כאשר פריט כלשהו

שאינו אקטיביטי לחוץ בתפריט, כפתור פתיחת התפריט המופיע בפינה הימנית העליונה הופך לכפתור חזור בכדי לחזור לתפריט הראשי.

```
<?xml version="1.0" encoding="utf-8"?>
<navigation 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:id="@+id/mobile_navigation"
    app:startDestination="@+id/nav_home">

    <fragment
        android:id="@+id/nav_home"
        android:name="com.example.thetrempiada.ui.home.HomeFragment"
        android:label="Home"
        tools:layout="@layout/fragment_home" />

    <fragment
        android:id="@+id/nav_about"
        android:name="com.example.thetrempiada.ui.about.AboutFragment"
        android:label="About"
        tools:layout="@layout/fragment_about"/>

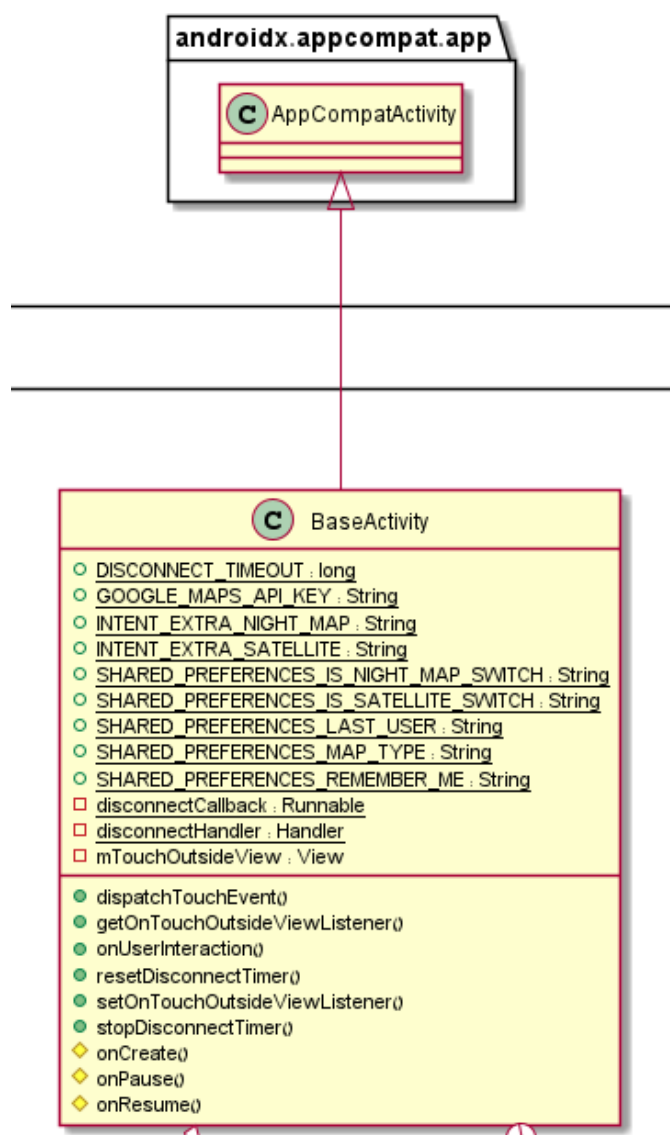
    <fragment
        android:id="@+id/nav_profile"
        android:name="com.example.thetrempiada.ui.profile.ProfileFragment"
        android:label="Profile"
        tools:layout="@layout/fragment_profile"/>

    <activity
        android:id="@+id/nav_settings"
        android:name="com.example.thetrempiada.ui.settings.SettingsActivity"
        android:label="Settings"
        tools:layout="@layout/activity_settings"/>
```

המחשה של קובץ ה-XML המשמש לניווט

בתפריט הראשי יש כפתור בית החוזר ל-MainActivity, כפתור למעבר ל-ProfileFragment, כפתור למעבר ל-SettingsActivity, כפתור הפותח את Contact Fragment, כפתור למעבר ל-AboutFragment וכפתור לפתיחת הפרגמנט המשמש להתנתקות. יש אופציה להוספת כפתורים נוספים במקרה הצורך.

BaseActivity

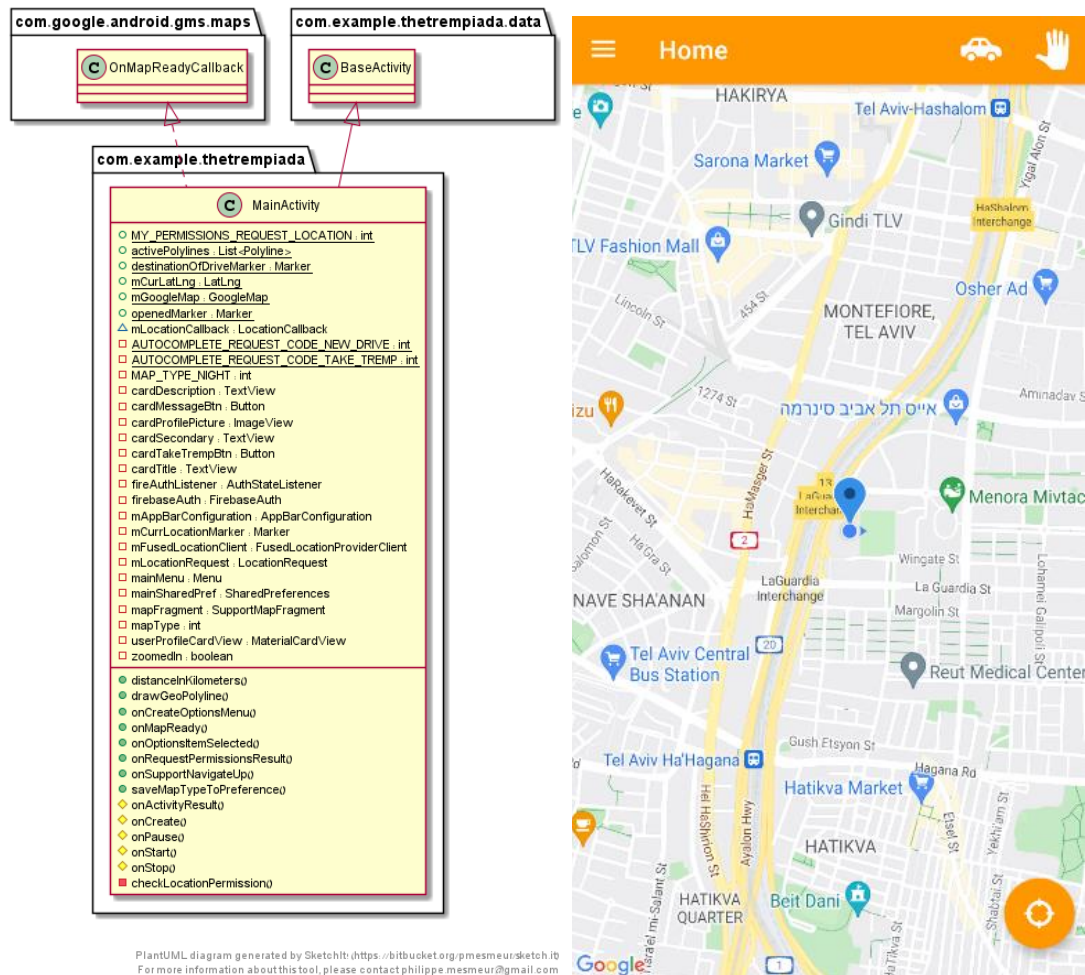


אקטיביטי זה אינה מוצגת ויזואלית למשתמש ואינה נגישה למשתמש דרך האפליקציה. היא משמשת בתור "אקטיביטי אם" לכל שאר האקטיביטיז, כי ישנן פעולות שהאפליקציה צריכה שיהיו בכל מקום באפליקציה (בכל האקטיביטיז), למשל פעולות שפועלות ברקע. כל האקטיביטיז באפליקציה יורשות את המחלקה הזו, והמחלקה הזו יורשת מ-AppCompatActivity, כדי שבאמצעות היררכיית המחלקות, כל האקטיביטיז בפרויקט יירשו גם הן מ-AppCompatActivity (המכילה את כל פעולות הבסיס ההכרחיות לתפקוד של אקטיביטי). BaseActivity מכילה גם קבועים (כמו שמפורט בתרשים ה-UML) הזמינים לכל האקטיביטיז באפליקציה. הפעולה מכילה בעיקר אתחולים של המשתמש שמתרחשים בכל כניסה לאקטיביטי (onCreate גלובלי), וכן ניתוקים (setOffline) של המשתמש בהתאם למצב האקטיביטי (onResume, onStop וכדומה). בתוך המחלקה עצמה הכנסתי

interface שאחראי לבדוק האם משתמש לחץ מחוץ ל-View מסוים, לדוגמה CardView. שאני משתמש בו ליצירת פרופיל משתמש ב- MainActivity כדי לסגור אותו כמו דיאלוג.

MainActivity

THETREMPIADA's Class Diagram

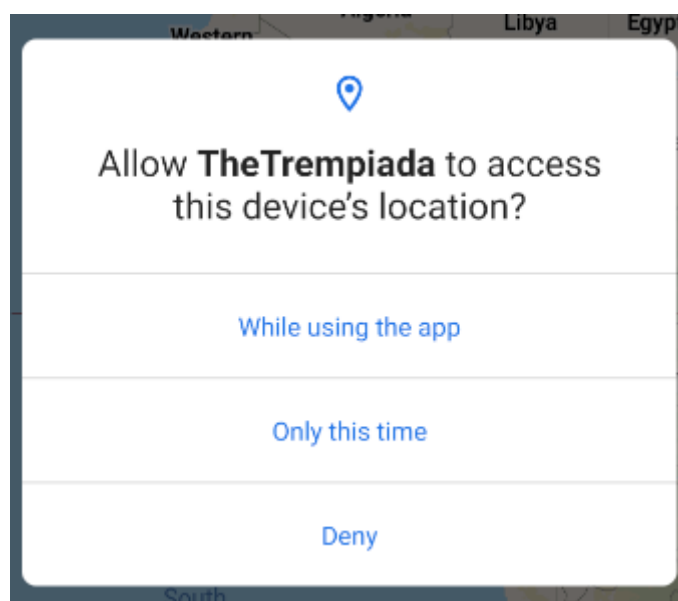


`MainActivity` היא האקטיביטי הראשית של הפרויקט, ולאחר התחברות \ הרשמה מועברים אליה מיידית. יש בה `Fragment` שמכיל את המפה, וכן כפתור מרחף (Floating Button). בנוסף יש בה כפתורים ליצירת מסלול נסיעה חדש (מכונית), ויצירת בקשה חדשה לטרמפ (כף יד), כפתורים אלו מופיעים בהתאם לסוג המשתמש. אם הוא נהג, יופיע כפתור המכונית בלבד, אם הוא טרמפיסט, יופיע כפתור היד בלבד, ואם הוא גם נהג וגם טרמפיסט, יופיעו שני הכפתורים. כפתורים אלו מופיעים רק כששירותי המיקום פעילים ולמשתמש יש מיקום זמין. בצד ימין למעלה יש כפתור לפתיחת תפריט הניתן לפתיחה מכל מקום באפליקציה.

באקטיבטי מוצגים פלטים רבים כגון Toasts, דיאלוגים (גם דיאלוגים לבקשת הרשאות כמו הרשאת מיקום), CardView להצגת פרופילים של משתמשים אחרים המוצגים על המפה, ועוד. אני משתמש באקטיבטי זה ובכל הפרויקט ברשימות (`ArrayList`) לשמירת ערכים ועצמים, וזה מבנה הנתונים הבסיסי היחיד שאני משתמש בו מכיוון שהוא מאוד נוח ומשלב את כל מבני הנתונים הישירים גם יחד (פונקציונליות של תור, מחסנית, מערך דינמי ואפשרויות מיון וקריאה נוחות).

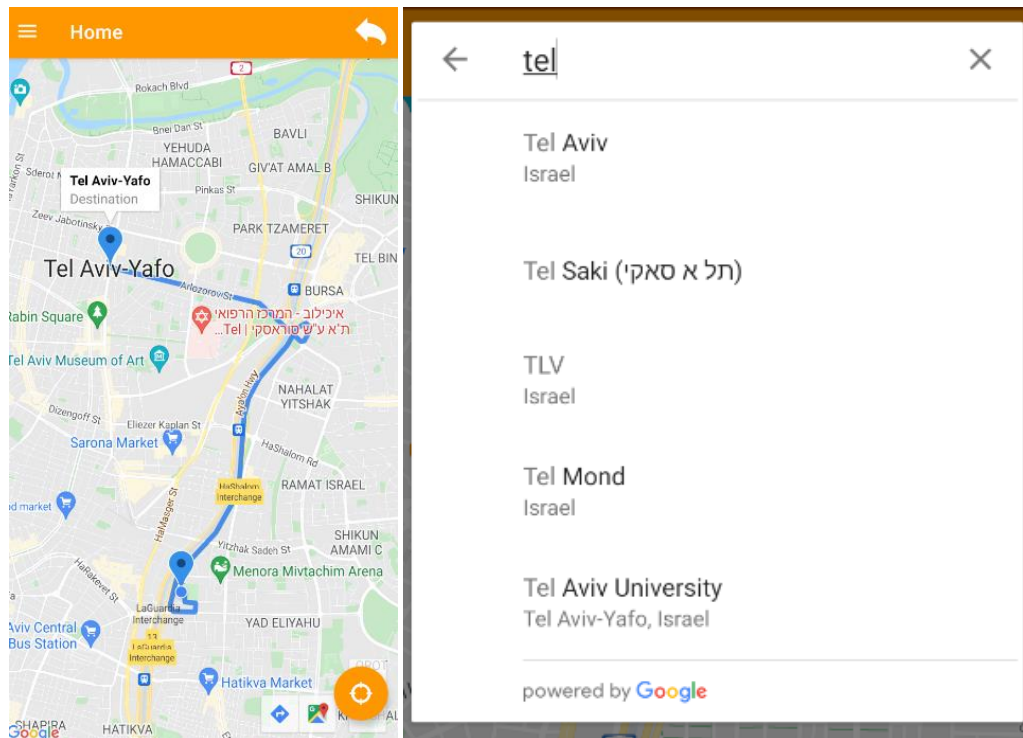
ניתן להגדיר את ה-MainActivity כמרכז הויזואלי של האפליקציה, וכמעט הכל קורה בה או עליה (באמצעות דיאלוגים ופרגמנטים). ב-MainActivity מוצגות כל התכונות של האפליקציה. אמחיש כעת את התכונות המרכזיות שניתן לבצע באפליקציה.

דיאלוגים, הרשאות והתראות למיניהם נצפים דרך MainActivity:



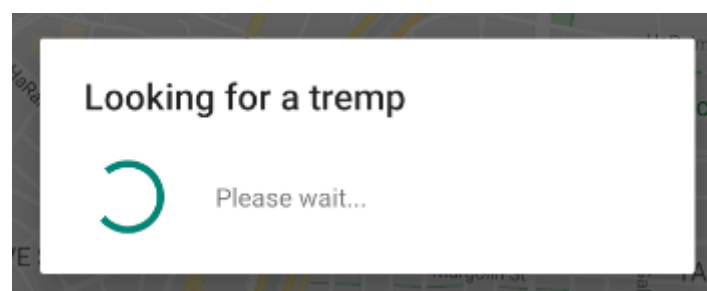
דיאלוג לבקשת הרשאה לשימוש במיקום

ניתן לחפש מקומות ולבצע תכנון מסלול באמצעות כפתורי המכונית וכף היד. לחיצה על כפתור המכונית מאפשרת לנהג לבחור יעד נסיעה ולתכנן מסלול.

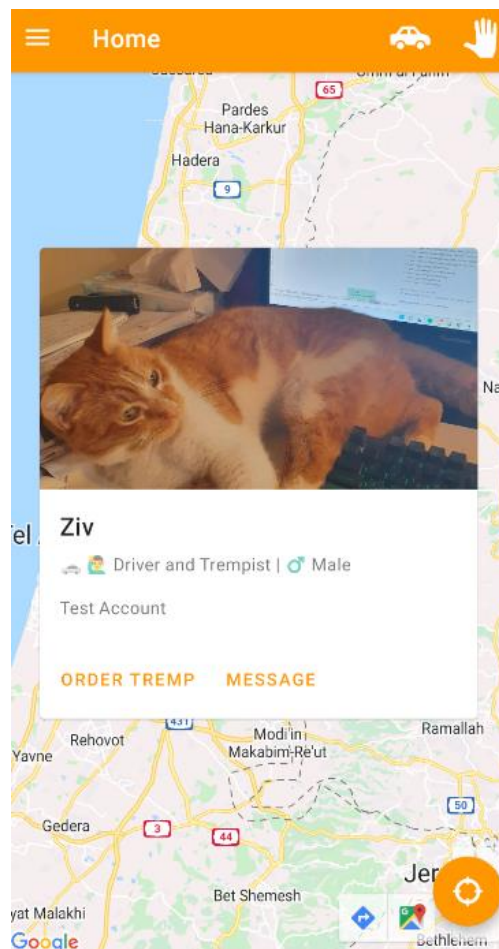


בעת הימצאות במצב ניווט (המצב המתואר בתמונה למעלה), ניתן ללחוץ על כפתור חזור לביטול הניווט וחזרה למסך הראשי.

לחיצה על כפתור כף היד פותח את אותו החלון של בחירת היעד, ולאחר מכן נפתח דיאלוג (מסוג LoadingDialog המובנה באנדרואיד) שמחפש נהגים זמינים באיזור שיכולים לקחת את הטרמפיסט לקרבת יעדו.



ניתן לצפות בפרופיל של משתמש אחר הנמצא בטווח של כמה קילומטרים מהמיקום הנוכחי של המשתמש באמצעות לחיצה על הסמן שמסמן את מיקומו:

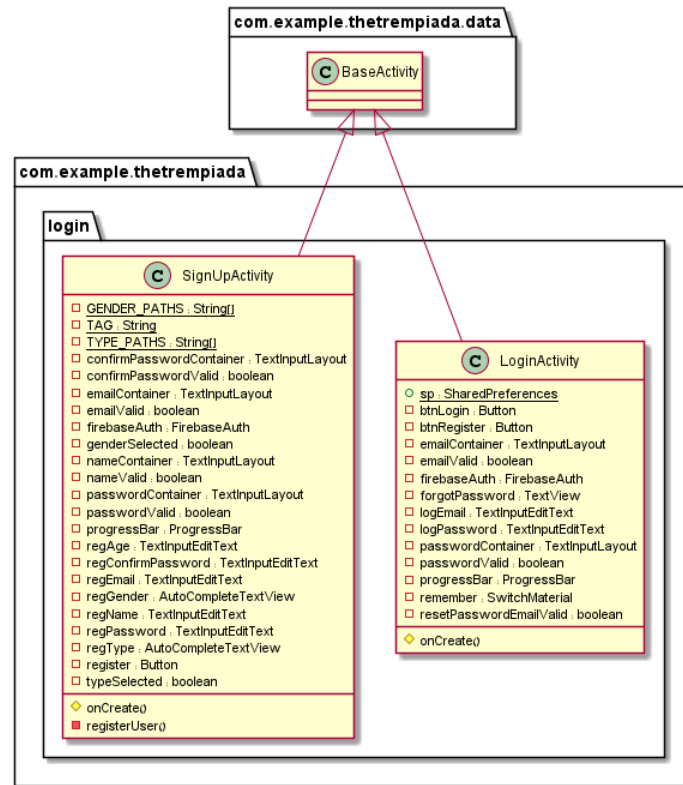


הכרטיס המוצג הוא מסוג CardView. הכפתורים בכרטיס אינם פעילים נכון לעכשיו, ובעתיד תתווסף אופציה לדירוג המשתמש דרך ממשק זה. לחיצה מחוץ לכרטיס סוגרת אותו.

ב-MainActivity השתמשתי בכמה פעולות ואלגוריתמים מעניינים שכתבתי, לדוגמה אלגוריתם שמקבל נ.צ. של מיקום נוכחי ונ.צ. של יעד ומשרטט על המפה מסלול (בהתאם לכבישים והנתיבים הקיימים) בקו צבעוני מנקודת ההתחלה אל היעד. כתבתי גם אלגוריתם המחשב מרחק בקילומטרים בין שתי נקודות המתחשב בכך שכדור"א אינו ישר. האלגוריתם מתבסס על רדיוס כדור הארץ ומשתמש בטריגונומטריה בנוסחה שנקראת Haversine Formula. ישנם עוד מספר אלגוריתמים ופעולות מתוחכמים שכתבתי וניתן לראות אותם בנספחים של ספר זה, בעיקר ב-MainActivity ו-SettingsActivity.

LoginActivity & SignUpActivity

LOGIN's Class Diagram



PlantUML diagram generated by Sketchit: <https://bitbucket.org/pmesmeur/sketchit>
For more information about this tool, please contact philippe.mesmeur@gmail.com

Sign Up

TheTrempiada



Display Name

Age Gender User Type

Email

Password

Confirm password

SIGN UP

Email

Password

Forgot password?

Remember me?

LOGIN

SIGN UP

SignUpActivity ו-LoginActivity משמשות להתחברות והרשמה בהתאמה. בשתייהן ישנם שדות EditText שעוצבו באמצעות MaterialUI שחוסמים קלט לא תקין (לדוגמא מייל בתבנית לא תקינה, או סיסמא עם מעט מדי תווים) באמצעות שימוש ב-Regex. רק כשהקלט תקין ניתן ללחוץ על כפתור Login או Sign Up. LoginActivity היא האקטיביטי הראשונה של האפליקציה, שנפתחת כאשר משתמש פותח את האפליקציה לראשונה, והיא מוגדרת ב-Manifest כתור "שער הכניסה" לאפליקציה. כשהמשתמש מחובר, האפליקציה מדלגת על LoginActivity בכניסה חוזרת באמצעות ערך הנשמר ב-SharedPreferences המצביע שהמשתמש מחובר. ב-LoginActivity יש Switch שמאפשר לקבוע אם המשתמש יישאר מחובר גם לאחר שהאפליקציה נסגרת ונפתחת מחדש (האפליקציה זוכרת את בחירתו באמצעות SharedPreferences). ב-LoginActivity יש TextView לחיצה "Forgot Password?" הפותח דיאלוג בלחיצה המאפשר הזנת כתובת מייל לקבלת מייל לשחזור סיסמא במידה ויש משתמש שהמייל המוזן מתאים לו בדאטאבייס.

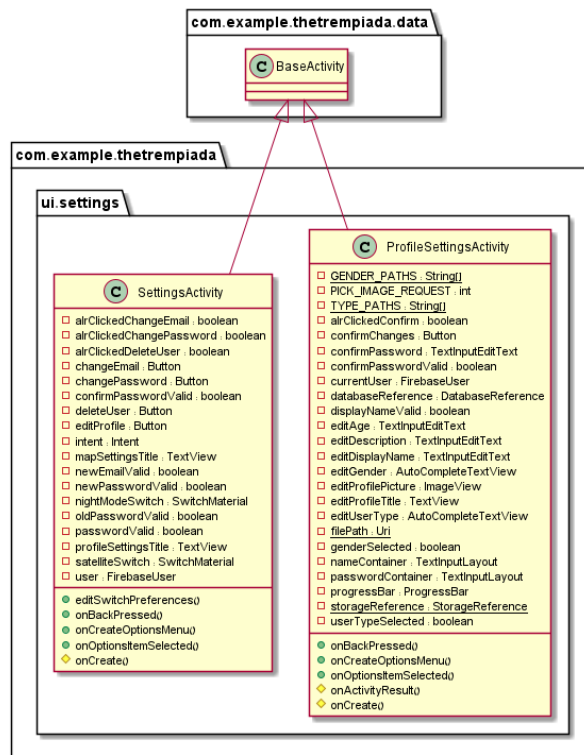
ב-SignUpActivity יש סוג שדה קלט נוסף, של רשימה (DropDownList הממומש באמצעות ArrayList ו-ListAdapter) המשמש בשדות "Gender" ו-"UserType" אשר מספר האפשרויות בהם מוגדר מראש.

הקישוריות בין האקטיביטיז מתבצעת באמצעות כפתור Sign Up ב-LoginActivity אשר עובר באמצעות Intent ל-SignUpActivity. כפתורי Login ו-SignUp ב-LoginActivity וב-SignUpActivity בהתאמה מעבירים את האפליקציה באמצעות Intent ל-MainActivity במידה והקלט תקין בהרשמה, וניתן ליצור משתמש בהצלחה, או שפרטי ההתחברות נכונים בכניסה.

The screenshot shows a registration form with three fields: Gender, Email, and Password. The Gender field is a dropdown menu with 'Male' and 'Female' options. The Email field contains '12345' and has a red error message: 'Not a valid email!'. The Password field contains '12345' and has a red error message: 'Password must be 6-16 numbers, letters or underscore, and contain at least 1 letter.' Each error message is accompanied by a red exclamation mark icon.

SettingsActivity & ProfileSettingsActivity

SETTINGS's Class Diagram



PlantUML diagram generated by SketchUML (<https://bitbucket.org/pmismeurs/sketch-uml>)
For more information about this tool, please contact philippe.mesmeur@gmail.com

Profile Settings

Edit Profile

Display Name

Age

Gender

User Type

Description

Password

CONFIRM

Settings

Map Settings

Night Mode

Satellite

Profile Settings

Change Email

Change Password

Edit Profile

Delete Account

SettingsActivity ו-ProfileSettingsActivity משמשות לשינוי הגדרות המשתמש והגדרות האפליקציה. ניתן להגיע ל-SettingsActivity באמצעות התפריט הראשי. בקטגוריית ה-Map Settings ניתן לשנות את תצוגת המפה לתצוגת לוויין או תצוגת לילה (שיצרת באמצעות קובץ JSON) בעזרת ה-Switch. האפליקציה זוכרת את בחירת עיצוב המפה באמצעות SharedPreferences ושומרת על המצב של הסוויצ'ים גם אחרי סגירת האקטיביטי ופתיחתה מחדש. בקטגוריה השנייה, Profile Settings, הכפתורים פותחים דיאלוגים\אקטיביטי. Change Email, Change Password, Delete Account פותחות דיאלוגים המאפשרות לבצע פעולות אלו ע"י אימות מחדש (Reauthentication) של זהות המשתמש באופן מאובטח ולאחר מכן לשלוח בקשה לדאטאבייס לביצוע הפעולה.

The image shows three side-by-side screenshots of a mobile application's settings section. Each screen has a title at the top in orange: 'Change Password', 'Change Email', and 'Delete Account'.
 - The 'Change Password' screen has two input fields: 'New Password' and 'Old Password', each with an eye icon for toggling visibility. At the bottom is a grey 'CONFIRM' button.
 - The 'Change Email' screen has two input fields: 'New Email' and 'Password', each with an eye icon. At the bottom is a grey 'CONFIRM' button.
 - The 'Delete Account' screen has a warning message 'This action cannot be undone!' in orange above a single 'Password' input field with an eye icon. At the bottom is a grey 'CONFIRM' button.

גם כאן כפתורי ה-Confirm נפתחים רק כאשר הפרטים שהוזנו תקינים מבחינה תחבירית (באמצעות Regex). ביצוע הפעולה מבוצע רק אם הפרטים שהוזנו נכונים.

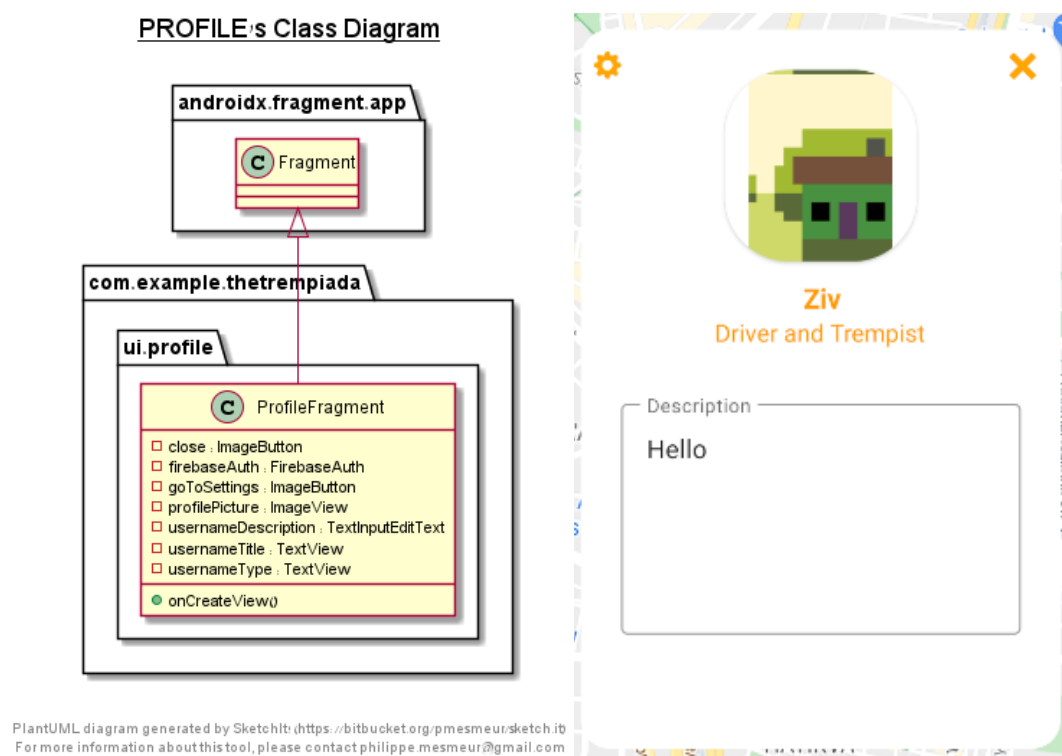
הכפתור EditProfile פותח באמצעות Intent את ProfileSettingsActivity, המאפשרת את עריכת שדות המשתמש (כולל העלאת תמונת פרופיל חדשה). ב-ProfileSettingsActivity יש קלטים של EditText, DropDownList וכן העלאת תמונה באמצעות Intent המעביר את המשתמש באמצעות action אל סייר הקבצים שלו לבחירת תמונה. האפליקציה מייצרת קישור URI זמני לצורך העלאת התמונה ל-Firebase Storage. הכפתור Confirm מתאפשר ללחיצה רק כאשר הוכנסה סיסמא תקינה ויש שינוי לפחות בשדה אחד. אם הסיסמא נכונה, הבקשה לשינוי פרטים נשלחת לדאטאבייס.

גם ב-SettingsActivity וגם ב-ProfileSettingsActivity יש כפתור חזור בפינה השמאלית עליונה של המסך המשתמש בפונקציה onBackPressed() מהמחלקה AppCompatActivity לחזרה למסך הקודם (האקטיביטי הקודם).

Menu Fragments

בתפריט הראשי ישנם כפתורים הפותחים Fragments על גבי המפה הפתוחה ב- MainActivity, פרגמנטים אלו מתפקדים כמו חלונות נפרדים באפליקציה אך הם אינם אקטיביטיז מטעמי עיצוב ונוחות, וחסכון במשאבים (האפליקציה מרעננת את עצמה הרבה במעבר באמצעות Intent בין אקטיביטיז, ויש פעולות שפועלות ברקע בכל האקטיביטיז ולכן רצוי לחסוך בהן).

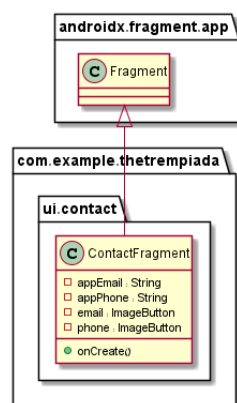
Profile Fragment



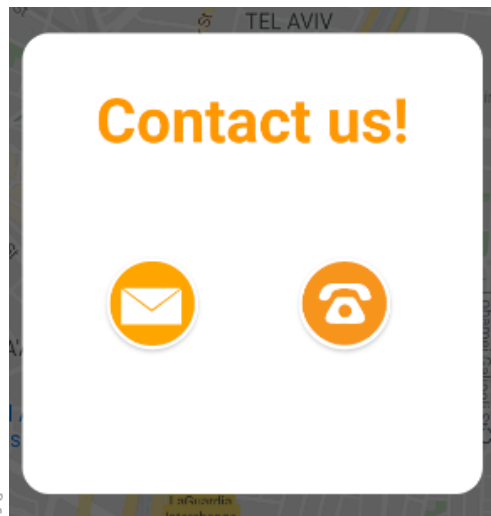
פרגמנט זה מציג את פרופיל המשתמש. הוא מציג את שם המשתמש (Display Name), תיאור המשתמש, סוג המשתמש (נהג, טרמפיסט או גם וגם), ואת תמונת המשתמש (ב- ImageView). תיאור המשתמש מוצג ב- EditText שאינו פעיל, כלומר לא ניתן לערוך אותו, לצורך העיצוב. בפינות הפרגמנט יש כפתור סגירה המשתמש ב- onBackPressed לצורך סגירת הפרגמנט וכפתור הגדרות המעביר בעזרת Intent את המשתמש ל- ProfileSettingsActivity.

Contact Fragment

CONTACT's Class Diagram



PlantUML diagram generated by Sketchit: <https://bitbucket.org/pmemeur/sketchit>
For more information about this tool, please contact philippe.mesmeur@gmail.com



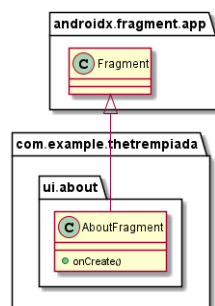
פרגמנט זה מכיל דיאלוג בלבד. בתוך הדיאלוג יש כותרת ב-TextView ושני כפתורים המאפשרים למשתמש ליצור קשר עם מפתח האפליקציה (אני) באמצעות מייל או שיחת טלפון. ההפניות לאפליקציית המיילים או לאפליקציית הטלפון מתבצעת באמצעות Intent שפועל עם Action. ל-Intent מוסיפים Extra המכיל את כתובת המייל \ מספר הטלפון. זה נראה כך:

```

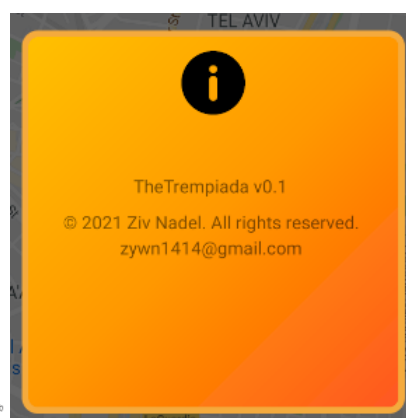
Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:" +
appPhone));
startActivity(intent);
  
```

About Fragment

ABOUT's Class Diagram



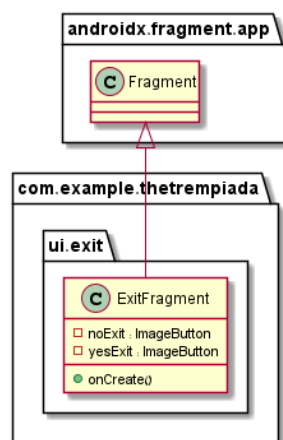
PlantUML diagram generated by Sketchit: <https://bitbucket.org/pmemeur/sketchit>
For more information about this tool, please contact philippe.mesmeur@gmail.com



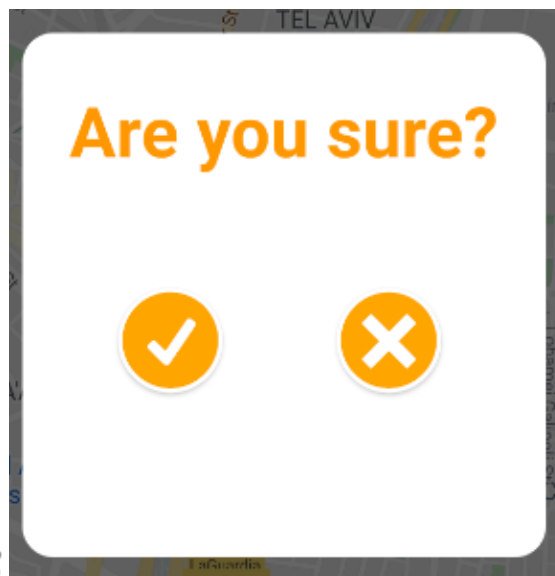
פרגמנט זה כולל בתוכו רק דיאלוג פשוט עם רקע מתחלף שעיצבתי, ובתוכו כתוביות קרדיט, גרסת התוכנה הנוכחית, מייל ליצירת קשר ואייקון קטן. יש בו TextView ו-ImageView.

Exit Fragment

EXIT's Class Diagram



PlantUML diagram generated by Sketchit: (<https://bitbucket.org/pmesteur/sketchit>)
 For more information about this tool, please contact philippe.mesmeur@gmail.com



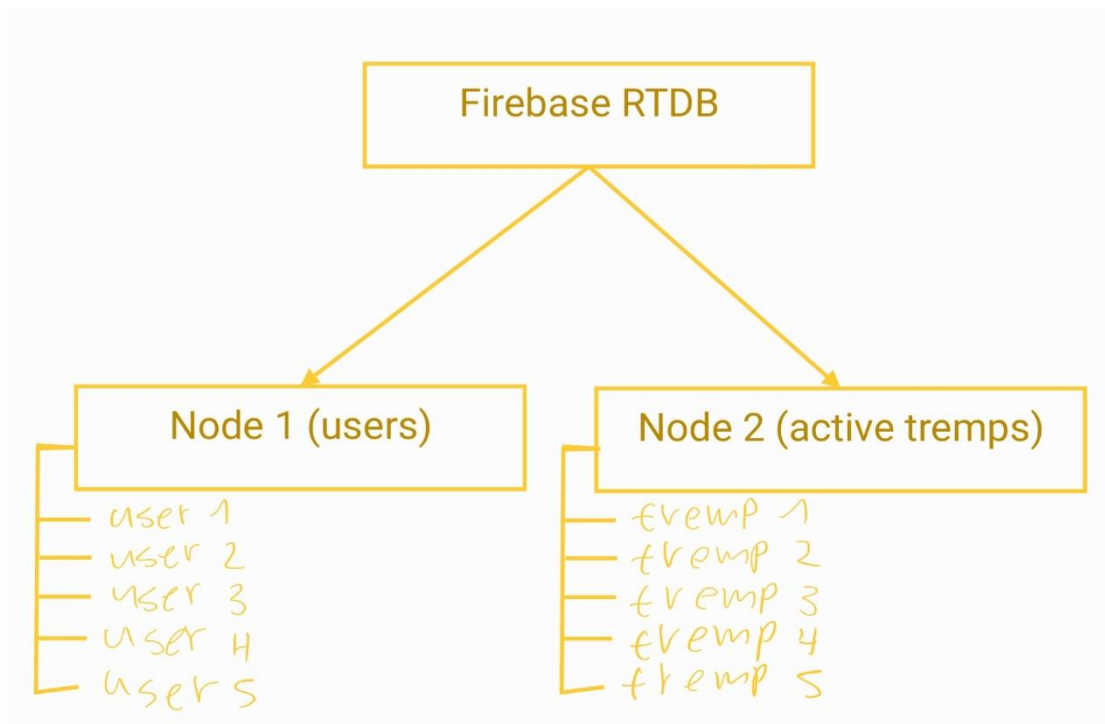
בפרגמנט זה יש דיאלוג פשוט שבתוכו יש כותרת ב-`TextView` וכן שני כפתורים – וי ואיקס. בלחיצה על הוי, האפליקציה מנתקת את המשתמש הנוכחי (מאפסת את המצביע של `Firebase` המכיל את המשתמש הנוכחי), וכן מעבירה את המשתמש באמצעות `Intent` ל-`LoginActivity` ומשנה את ה-`SharedPreferences` המצביע שיש משתמש מחובר. בלחיצה על האיקס, נקראת הפונקציה `onBackPressed()` וחוזרים למסך הקודם.

ישנו פרגמנט נוסף, `HomeFragment`, אך הדבר היחיד שהוא עושה הוא לחזור ל-`MainActivity` בלחיצה על הכפתור המתאים בתפריט וכן למרכז מחדש את המשתמש על המיקום שלו במפה, ולכן לא אפרט עליו כאן.

בסיס נתונים

אני משתמש בבסיסי נתונים של Firebase Database השייכים ל-Google. אני משתמש ב-Firebase Authentication לאחסון מידע בסיסי של משתמשים ולביצוע הזדהות מאובטחת שלהם.

לאחסון הנתונים כמו פרטים מפורטים יותר של משתמשים (כמו מיקום נוכחי וסטטוס אונליין) וכן מיקומים ופרטים נוספים של האפליקציה אני שומר ב-Firebase Realtime Database, שהוא בסיס נתונים שאינו טבלאי, אלא מוגדר באמצעות שיטת key-value שבה לכל עצם או ערך יש מזהה (key). בסיס הנתונים מאורגן בעזרת nodes – חוליות, בצורת "שרשור", כלומר לחוליה הראשית מחוברות תתי חוליות שממשות כקטגוריות ותחתיהן חוליות המתפקדות כערכים במבנה הבא:



בסיס הנתונים מתעדכן בזמן אמת, וישנן פונקציות בקוד שנקראות בכל פעם שיש שינוי בערך מסוים (`value.onDataChanged()`), וכך אפשר לשמור שהמידע יהיה זמין למשתמשים בזמן אמת ובצורה מעודכנת. לקריאה וכתובה לבסיס הנתונים משתמשים בפעולות מובנות המשתמשות בשיטת ה-Nodes גם בקוד. דוגמא לקוד בסיסי לקבלת עצם המכיל את פרטי המשתמש מתוך Firebase Realtime Database:


```

// getting user data from realtime database and updating profile at real
time
DatabaseReference database =
FirebaseDatabase.getInstance("https://thetrempiada-5e41a-default-
rttdb.europe-west1.firebaseio.com/").getReference();
database.child("users").child(firebaseAuth.getCurrentUser().getUid()).addVa
lueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        // getting current user at real time
        User user = snapshot.getValue(User.class);
    }

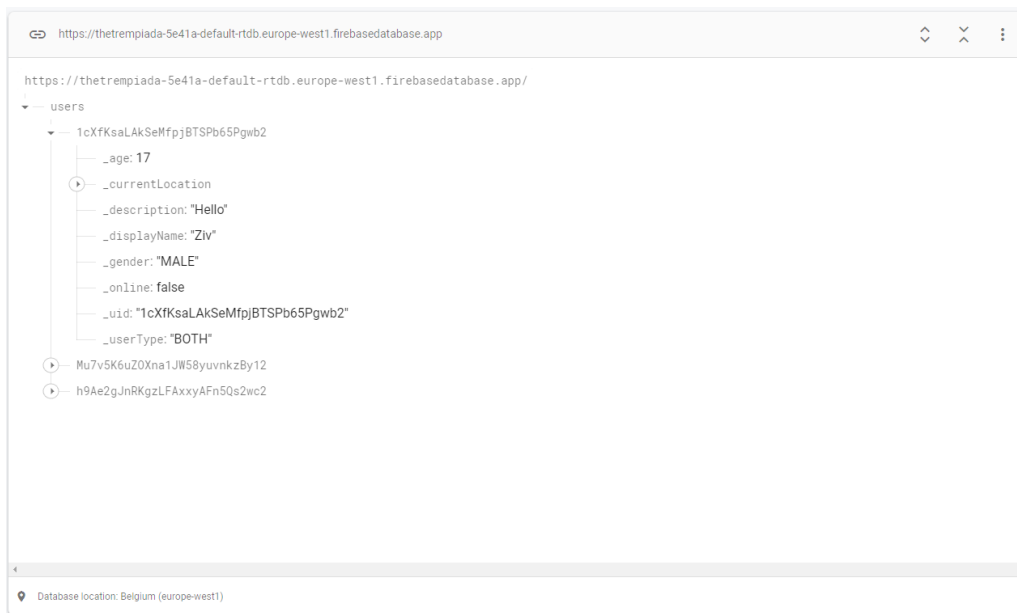
    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});

```

לאחסון תמונות המשתמשים, אני משתמש ב-Firebase Storage. ב-Storage שלי יש תיקייה הנקראת profilePhotos ובה קבצי התמונות של המשתמשים. שם קובץ התמונה של המשתמש זהה ל-UID שלו לצורך זיהוי והתאמה בתוך הקוד.



לשליטה על בסיס הנתונים ישנן קונסולות באינטרנט לכל אחד משלושת בסיסי הנתונים. הקונסולות פשוטות ונוחות לשימוש ונראות כך:



הקונסולה של Realtime Database - שיטת key-value

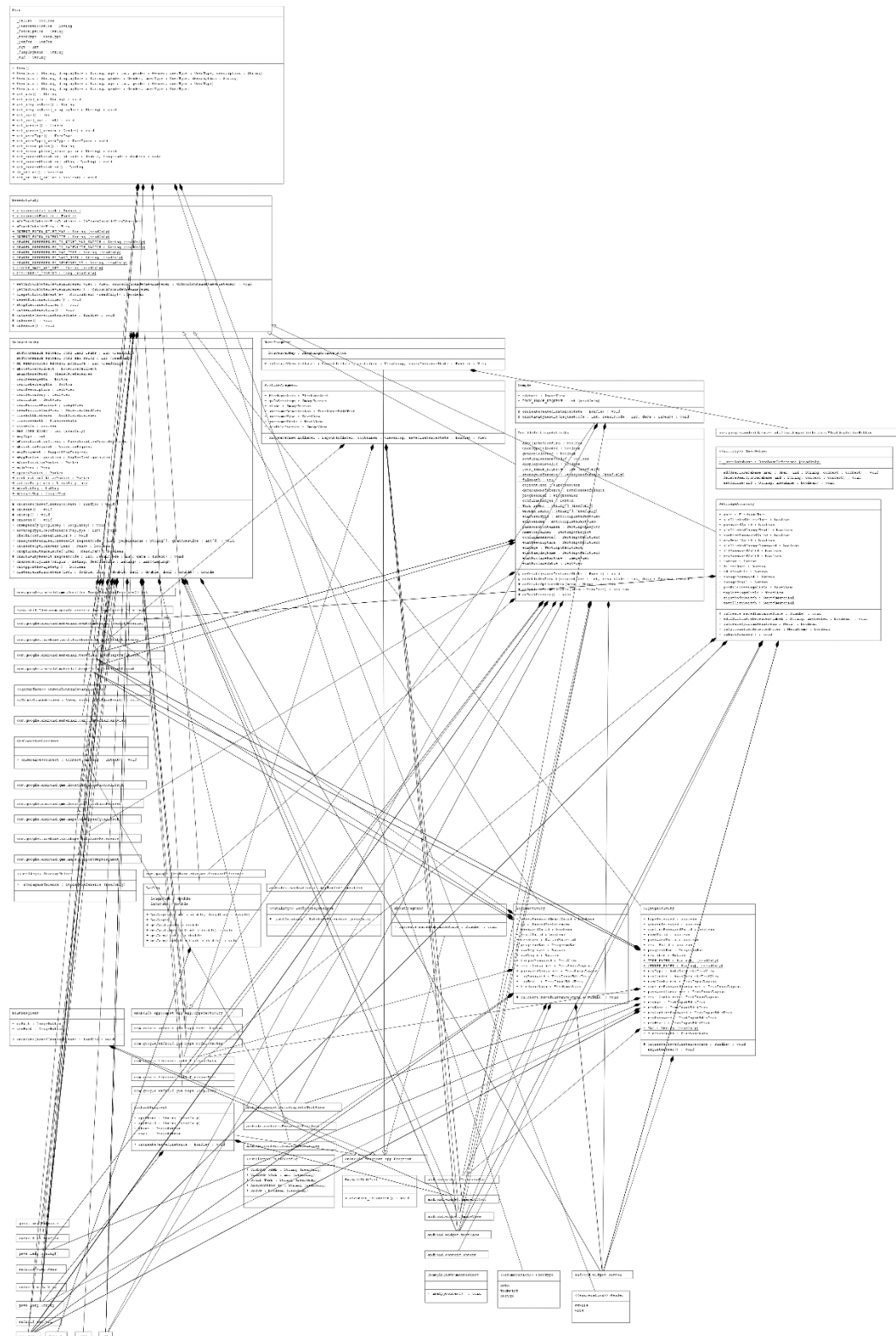
Search by email address, phone number, or user UID					Add user	↻	⋮
Identifier	Providers	Created ↓	Signed In	User UID			
test@gmail.com	✉	Nov 21, 2021	Nov 21, 2021	Mu7v5K6uZ0Xna1JW58yuvnkzBy...			
zivnadel@gmail.com	✉	Nov 19, 2021	May 22, 2022	h9Ae2gJnRKgzLFAxxyAFn5Qs2wc2			
zywn1414@gmail.com	✉	Nov 19, 2021	May 22, 2022	1cXfKsaLakSeMfpjBTSPb65Pgwb2			
					Rows per page:	50 ▼	1 – 3 of 3 < >

הקונסולה של Firebase Authentication

gs://thetrempiada-5e41a.appspot.com > profilePictures					Upload file	+	⋮
<input type="checkbox"/>	Name	Size	Type	Last modified			
<input type="checkbox"/>	 1cXfKsaLakSeMfpjBTSPb65Pgwb2	80.51 KB	image/jpeg	Nov 26, 2021			
<input type="checkbox"/>	 h9Ae2gJnRKgzLFAxxyAFn5Qs2wc2	3.34 MB	image/jpeg	Nov 25, 2021			

הקונסולה של Firebase Storage

פירוט מחלקות עזר ועצמים (מדריך למפתח)

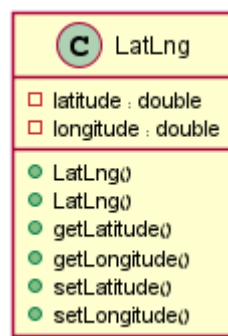


בעמוד הקודם מצורף תרשים UML מלא של כל האפליקציה כולל את כל המחלקות, אקטיביטיז וכן מחלקות מערכת שבהם אני משתמש בפרויקט. מאחר והתרשים עמוס מאוד מצורף קישור לקובץ התמונה הניתן להגדלה והקטנה לצורך צפייה בתרשים:

<https://ibb.co/p39V57D>

בפרויקט שלי אני משתמש במספר מחלקות נוספות שאינן אקטיביטי או פרגמנט, והן פועלות ברקע ומשמשות בתור עזר לתהליכים שמתבצעים באקטיביטיז. תיעוד מלא של קוד המחלקות ניתן לראות בפרק "נספחים" בספר זה.

public class LatLng



מדובר בהרחבה של המחלקה המובנית של גוגל LatLng (Longitude-Latitude – נקודת ציון). כדי להעלות ולהוריד עצמים מ-Firebase RT Database צריך שבמחלקה יהיה קונסטרקטור ריק. במחלקה המובנית של גוגל LatLng אין קונסטרקטור ריק ולכן יצרתי חלופה של המחלקה משלי.

פעולות המחלקה

```
public LatLng(double latitude, double longitude) {
    this.latitude = latitude;
    this.longitude = longitude;
}
```

קונסטרקטור ריק – {}

```
public LatLng() {}
```

```
public double getLatitude() {
    return latitude;
}
```

```
public void setLatitude(double latitude) {
```

```

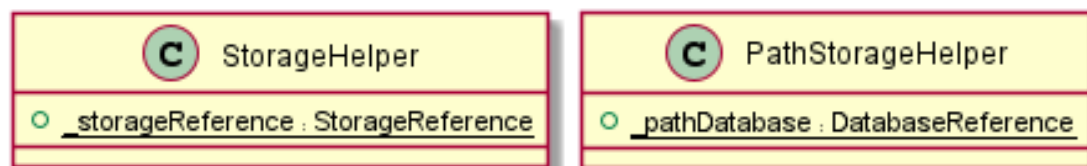
        this.latitude = latitude;
    }

    public double getLongitude() {
        return longitude;
    }

    public void setLongitude(double longitude) {
        this.longitude = longitude;
    }
}

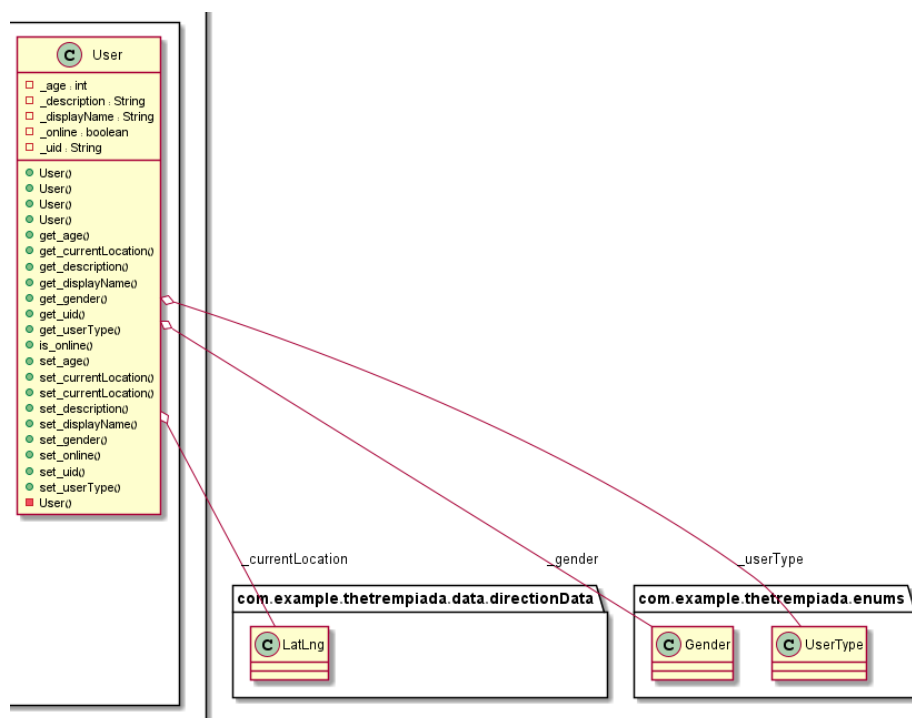
```

public class StorageHelper & PathStorageHelper



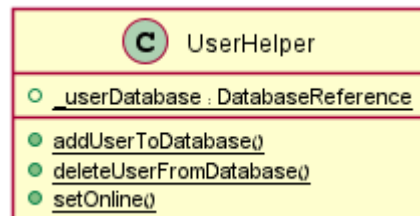
מדובר במחלקות עזר פשוטות ביותר, כי כרגע אין לי שימוש רב ב-Firebase Storage, ולכן המחלקות מכילות רק את הרפרנס לאחסון ולדאטאבייס כרגע, כדי לחסוך את הצורך לכתוב את הכתובות הארוכות בכל שימוש.

public class User



המחלקה מייצגת את העצם User שנמצא ב-Firebase RT Database ובו אני משתמש כדי לאחסן את פרטי המשתמש. במחלקה יש מספר קונסטקטורים שונים וכן פעולות get ו-set לכל התכונות כמתואר בתרשים ה-UML.

public class UserHelper



מחלקה זו היא מחלקת עזר המכילה פעולות עזר הקשורות למשתמש כגון הוספת משתמש לדאטאבייס, מחיקה ועוד. למחלקה יש תוכנה סטטית אחת, שהיא הרפרנס לחולייה (Node) ב-RT Database המכילה את כל המשתמשים.

פעולות המחלקה

```
public static void addUserToDatabase(User user, String uid, Context context)
```

פעולה זו מוסיפה כניסת key-value לדאטאבייס כאשר ה-key הוא ה-UID וה-value הוא העצם מטיפוס User המתקבל. הפעולה שולחת Toast לגבי סטטוס הפעולה (הצליח\נכשל) ב-context המתאים.

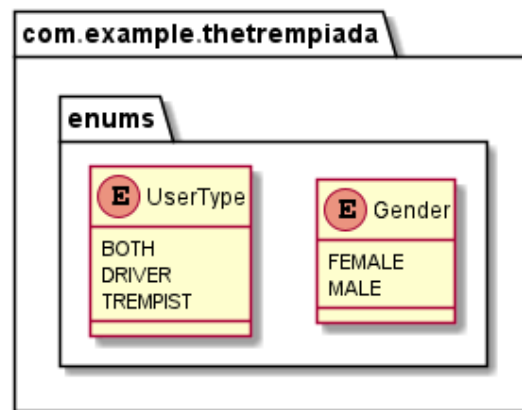
```
public static void deleteUserFromDatabase(String uid, Context context)
```

הפעולה מוחקת ערך בדאטאבייס תחת החולייה users שה-key שלה הוא ה-UID המתקבל (מחיקת משתמש מהדאטאבייס). הפעולה שולחת Toast לגבי סטטוס הפעולה (הצליח\נכשל) ב-context המתאים.

```
public static void setOnline(String uid, boolean isOnline)
```

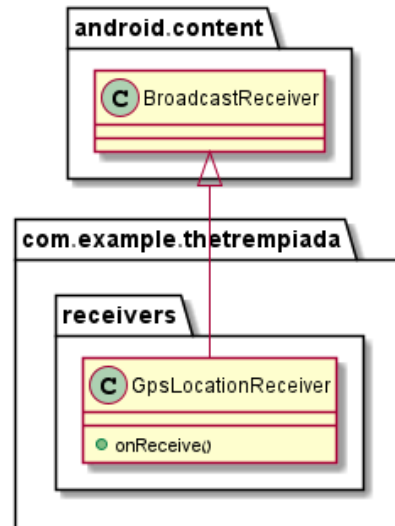
הפעולה משנה את סטטוס ה-online של המשתמש שה-UID שלו הוא ה-UID שהתקבל (כניסה שה-key שלה הוא ה-UID). הפעולה משנה את ערך ה-online של המשתמש בתוך הדאטאבייס לערך שנמצא במשתנה הבוליאני isOnline.

public enum Gender & UserType



המחלקות הללו הן enum, כלומר ניתן להשתמש בהם ליצירת עצם עם מספר קבוע של אופציות להגדרתו. ל-Gender יש 2 אופציות – Male ו-Female, ל-UserType יש 3 אופציות – Driver, Trempist ו-Both.

public class GpsLocationReceiver extends BroadcastReceiver



מחלקה זו פועלת ברקע, ושולחת התראה כאשר שירותי המיקום (GPS) התנתקו, וכאשר שירותי המיקום חזרו לפעול. שירות כזה רלוונטי לאפליקציה שמתבסס כולה על מיקום ומפות. כאשר האפליקציה מזהה שינוי בפעולת שירותי המיקום, היא שולחת Intent עם הפעולה android.location.PROVIDERS_CHANGED, וכאשר ה-BroadcastReceiver מזהה שה-Intent הזה נשלח, הוא קולט אותו בפעולה onReceive() ומוציא פלט (Toast) מתאים.

מדריך למשתמש

דרישות לפני התקנת האפליקציה



- ראשית, יש לוודא כי המכשיר בו משתמשים בעל גרסת אנדרואיד 4.4 ומעלה, אם כי יש לציין שלצורך חוויה מיטבית יש להשתמש במכשיר עם גרסת אנדרואיד 8.0 ומעלה.
- יש לוודא ששירותי המיקום של המכשיר (GPS) פעילים וניתנים לשימוש.
- יש לוודא שיש חיבור אינטרנט (Cellular/WiFi) פעיל במכשיר.

פתיחה ראשונה ומסך ההרשמה

לאחר התקנת האפליקציה ובעת הפתיחה הראשונה, המשתמש ייחשף לדף ההתחברות. יש ללחוץ על כפתור SignUp לצורך הרשמה לאפליקציה ולמלא את הפרטים בעמוד שנפתח.

Sign Up

TheTrempiada


Display Name


Age

Gender ▼

User Type ▼


Email

Password 

Confirm password 

SIGN UP

Email

Password 

Forgot password?

Remember me? ☐

LOGIN

SIGN UP

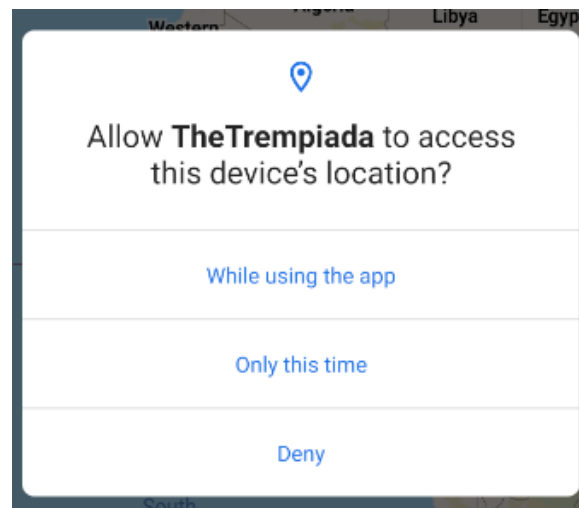
יש להקפיד על הכנסת פרטים תקינים, במידה ולא יוכנסו פרטים תקינים, המשתמש יקבל על כך התראה מתחת לתיבת הטקסט. כאשר כל הפרטים תקינים, צבע הכפתור SignUp במסך ההרשמה ישתנה לצבע כתום וניתן יהיה ללחוץ עליו ולהירשם. לאחר שההרשמה עברה בהצלחה, המשתמש מחובר לראשונה לאפליקציה והמסך הראשי ייפתח.

ניתן תמיד לבצע התחברות חוזרת עם פרטי המשתמש במסך ההתחברות במקרה הצורך. במסך ההרשמה ניתן ללחוץ על הסמן "Remember Me?" כדי שבכניסה הבאה לאפליקציה לא יהיה צורך בהתחברות. ניתן ללחוץ על "Forgot Password?" לצורך שחזור הסיסמא במקרה הצורך. יש להקפיד על תקינות פרטי ההתחברות גם במסך ההתחברות.

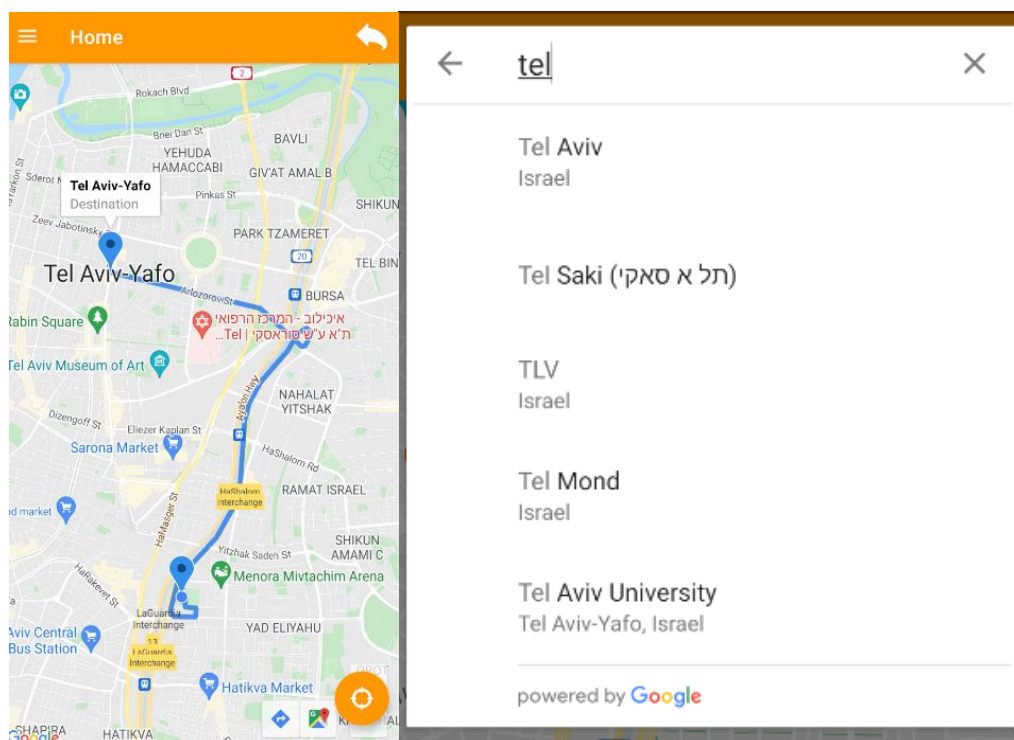
המסך הראשי



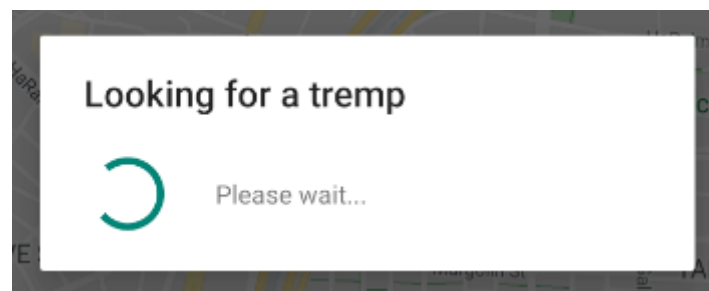
בעת כניסה ראשונה למסך הראשי, המשתמש יראה חלון קופץ שביקש ממנו לאשר שימוש בשירותי המיקום (GPS). יש לאשר את השימוש כדי להשתמש באפליקציה.



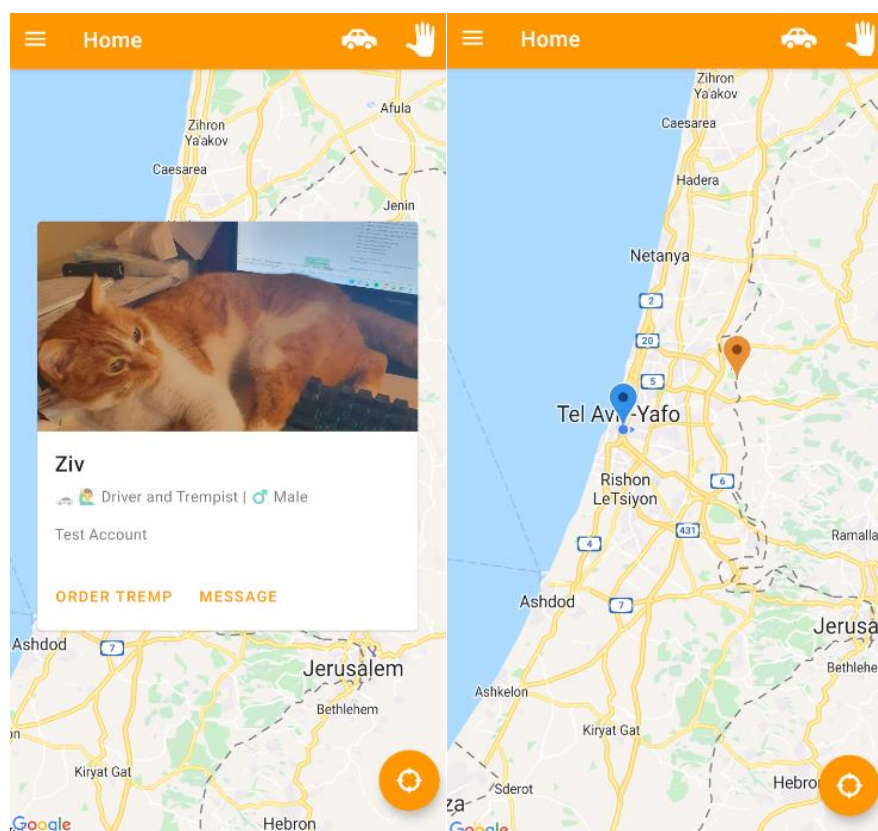
במסך הראשי ניתן לראות נעץ שנמצא במיקום הנוכחי של המשתמש במפה. ניתן ללחוץ על הכפתור בפינה הימנית התחתונה של המפה למרכז ופוקוס של המסך חזרה על המיקום הנוכחי של המשתמש. בפינה הימנית העליונה של המסך ישנם כפתורים המאפשרים שימוש באפליקציה. לחיצה על כפתור המכונית (לנהגים בלבד), מאפשר להכניס יעד לנסיעה והאפליקציה מייצרת לנהג מסלול בזמן אמת, ומשתמשת במסלולו כדי להתריע לטרמפיסטים אחרים אם מסלולו של נהג זה רלוונטי להם.



לחיצה על כפתור כף היד (לטרמפיסטים), מאפשרת להם להכניס יעד ומחפשת עבורם נהג שיכול לקחת אותם קרוב ליעדם.

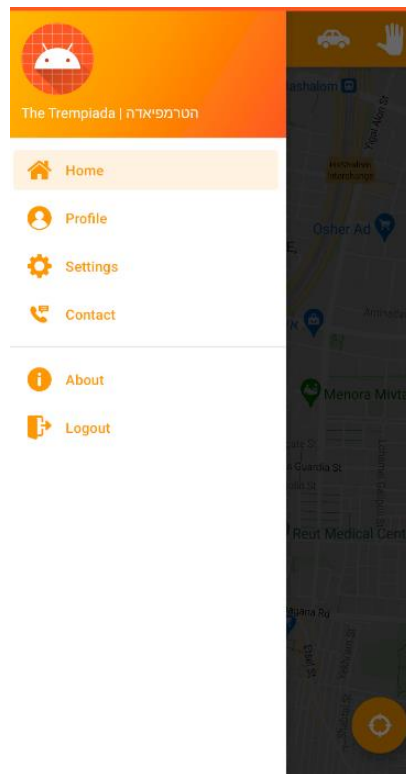


במפה מופיעים נעצים בצבע כתום המייצגים את מיקומם של משתמשים מחוברים אחרים באפליקציה. לחיצה על הנעץ שלהם תפתח את כרטיס הפרופיל שלהם המכיל פרטים ואמצעי תקשורת עם המשתמשים האחרים.

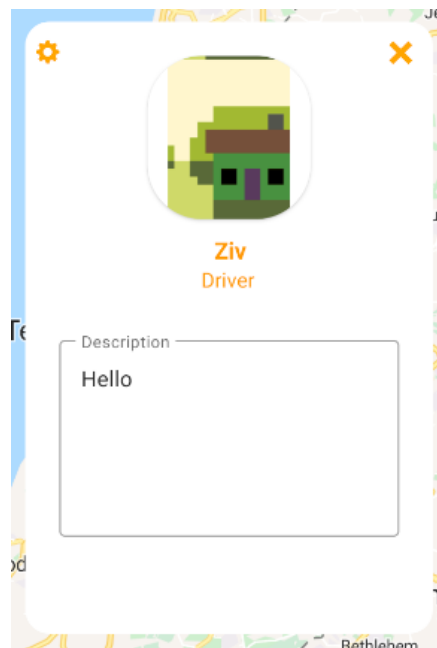


תפריט האפליקציה וחלונות נוספים

לחיצה על כפתור התפריט בפינה השמאלית העליונה של החלון הראשי תפתח את התפריט הראשי של האפליקציה.

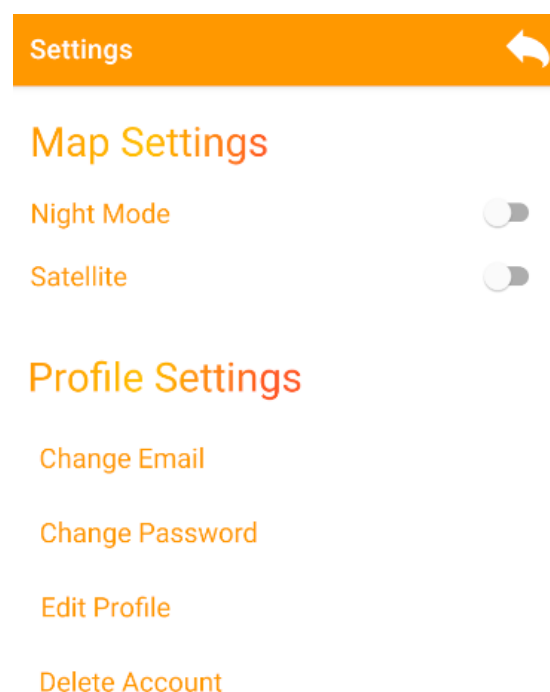


בתפריט האפליקציה יש מספר כפתורים. הכפתור העליון ("Home") יחזיר את המשתמש למסך הבית (המפה). הכפתור השני ("Profile") פותח את פרופיל המשתמש.

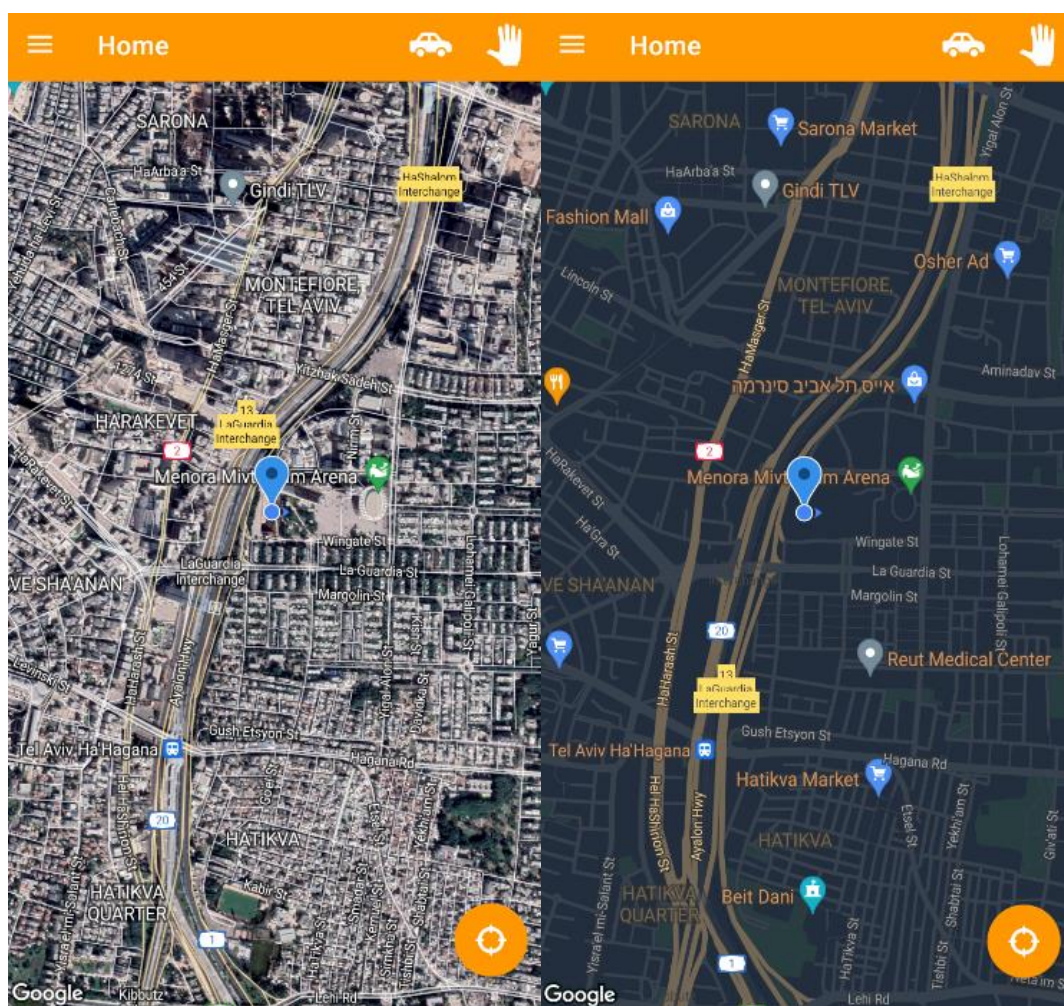


ניתן לסגור את פרופיל המשתמש (המחזיר את המשתמש למסך הבית) ע"י לחיצה על כפתור ה-X בפינה הימנית העליונה של כרטיס המשתמש, ולחיצה על כפתור ההגדרות בפינה השמאלית העליונה של כרטיס המשתמש תעביר את המשתמש להגדרות פרופיל (ראה בהמשך).

הכפתור השלישי בתפריט ("Settings") פותח את הגדרות האפליקציה.



בתפריט ההגדרות ישנן הגדרות של המפה והגדרות של פרופיל המשתמש. בהגדרות המפה ניתן לשנות את סגנון המפה למצב "לילה" או מצב "לווין", בעזרת לחיצה על הסמנים מצד ימין. תמונות להמחשה:



מצב לילה (מימין) ומצב לווין (משמאל)

בלחיצה על הכפתורים "Change Password", "Change Email", ו-"Delete Account" ייפתח חלון קטן המאפשר למשתמש לשנות פרטים אלו (או למחוק את המשתמש שלו) ע"י אימות מחודש באמצעות הסיסמא.

Change Password	Change Email	Delete Account <small>This action cannot be undone!</small>
<input type="password"/> New Password	<input type="text"/> New Email	<input type="password"/> Password
<input type="password"/> Old Password	<input type="password"/> Password	
<input type="button" value="CONFIRM"/>	<input type="button" value="CONFIRM"/>	<input type="button" value="CONFIRM"/>


שים לב שיש ללחוץ פעמיים על כפתור "Confirm" כדי לאשר את השינויים. יש לשים לב לתקינות הפרטים המוקלדים.

בלחיצה על הכפתור "Edit Profile" ייפתח חלון לשינוי פרטי המשתמש:

Profile Settings

↩

Edit Profile



Display Name

Age

Gender

User Type

Description

Password

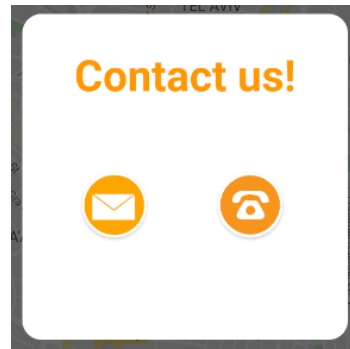
👁

CONFIRM

יש לבצע שינוי לפחות בשדה אחד של פרופיל המשתמש ולהכניס סיסמא תקינה כדי לשנות את פרופיל המשתמש. ניתן ללחוץ על אייקון התמונה כדי לשנות תמונת פרופיל. לחיצה על האייקון תעביר את המשתמש לגלריית התמונות שלו לצורך בחירת תמונה. שים לב שיש ללחוץ פעמיים על כפתור "Confirm" כדי לאשר את השינויים. יש לשים לב לתקינות הפרטים המוקלדים.

ניתן ללחוץ על כפתור החזור (החץ) בפינה הימנית העליונה של דף ההגדרות וכן של דף הגדרות פרופיל המשתמש כדי לחזור לדף הקודם.

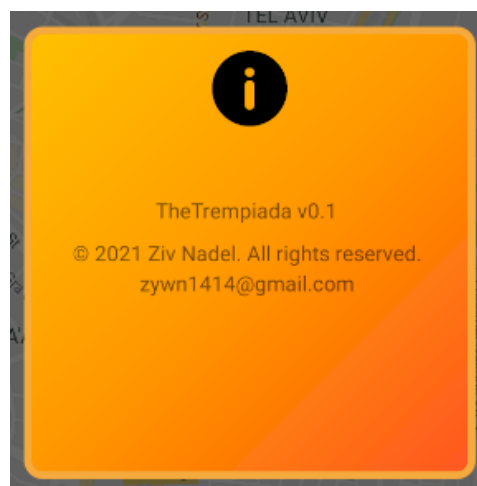
הכפתור הרביעי בתפריט הראשי ("Contact") פותח חלון ליצירת קשר עם מפתח האפליקציה.



לחיצה על כפתור המייל בצד שמאל תפתח את תיבת הדואר במכשיר את המשתמש ובנמען תהיה כתובת המייל של מפתח האפליקציה. באופן דומה, לחיצה על כפתור הטלפון בצד הימין תעביר את המשתמש לאפליקציית החיוג במכשיר שלו כאשר מספר הטלפון לחיוג יהיה מספר הטלפון של מפתח האפליקציה. באופן זה ניתן ליצור קשר עם האפליקציה.

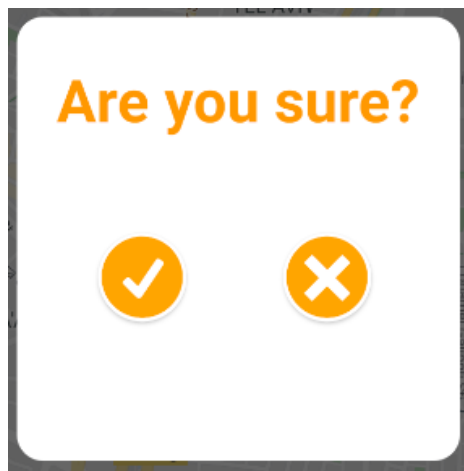
לחיצה מחוץ לדף יצירת הקשר תסגור את דף זה והמשתמש יעבור לדף הראשי.

בלחיצה על הכפתור החמישי בתפריט ("About") ייפתח חלון קטן המכיל את פרטי האפליקציה.



לחיצה מחוץ לחלון זה תסגור את החלון והמשתמש יעבור לדף הראשי.

בלחיצה על הכפתור השישי בתפריט ("Logout") ייפתח חלון המאפשר למשתמש לבחור אם להתנתק או להישאר באפליקציה.



לחיצה על כפתור הוי בצד שמאל תתנתק מהאפליקציה ותעביר את המשתמש חזרה לדף ההתחברות. לחיצה על כפתור האינס בצד ימנית תבטל את הניתוק ותחזיר את המשתמש לדף הראשי של האפליקציה.

רפלקציה

את תהליך העבודה על האפליקציה התחלתי כבר בתחילת כיתה י"א, כאשר התחלנו ללמוד איך לבנות אפליקציות לאנדרואיד בסביבת Android Studio. הנושא והחומר מאוד מעניינים, בגלל שזה יישום של העקרונות התיאורטיים שלמדנו במגמת מדעי המחשב החל מכיתה י'. אני אוהב מאוד ליצור אפליקציות ואתרים מכיוון שאלו מהווים פתרונות ויזואליים לבעיות או לצרכים שמתעוררים בי, ואני ממש רואה איך הדמיון שלי והחשיבה שלי "קמים לתחייה", ורואים את זה ממש בצורה ויזואלית באפליקציה. בחרתי ליצור אפליקציה יחסית מורכבת שענתה על דרישות הפרויקט והרבה מעבר לכך, ולכן באופן טבעי נתקלתי במספר קשיים טכניים ובבעיות מורכבות, ולכן התעורר הצורך ללמוד תכנים נוספים שאינם חלק מתוכנית הלימודים. אני אדם סקרן ולכן נהניתי מהלמידה מאוד, ובאמצעות האינטרנט, גוגל ואתר Stack Overflow הצלחתי לפתור את כל הקשיים הטכניים שהתעוררו תוך כדי פיתוח האפליקציה. בראייה לאחור, אם הייתי יודע שלא אספיק לסיים את האפליקציה כפי שתכננתי אותה, אולי הייתי מתכנן אותה בצורה מינימלית יותר כדי להספיק בזמן, למרות שגם מה שהספקתי עד כה יוצר אפליקציה מרשימה למדי. אם היה לי זמן נוסף, הייתי מסדר ומשיק את האופציה להזמנת טרמפ, שזו המהות של האפליקציה ולצערי זה עדיין לא עובד מושלם. ישנם עוד מספר דברים קטנים שהייתי מוסיף, כמו צ'אט בין משתמשים ואופציה לדירוג ואימות (Verification) של משתמשים. אני בטוח שאם היה לי עוד זמן הייתי חושב על עוד פיצ'רים ואופציות מעניינות ושימושיות שאפשר להוסיף לאפליקציה כמו שלי, ואולי אף הייתי מספיק להשיק אותה השנה לחנות האפליקציות, מה שאני מתכנן לעשות כאשר האפליקציה תושלם. אני מרגיש שלקחתי הרבה כלים לחיים ממגמת הנדסת תוכנה ומפיתוח האפליקציה, שכן אני יודע שאני רוצה לעשות בנושא זה או בנושא דומה בעתיד ולבנות קריירה בנושא, בעיקר בפיתוח אפליקציות ופיתוח אתרים, שנשענים על אותם עקרונות. אני מתכנן לקחת את האפליקציה שיצרתי ולשכלל אותה במהלך החופש הגדול, וכן לבנות עוד אפליקציות וללמוד טכנולוגיות חדשות שיאפשרו לי לבנות פרויקטים מתקדמים, ולהתקדם בנושא של פיתוח אתרים ואפליקציות ברמה שאוכל לעסוק בזה בעתיד.

ביבליוגרפיה

- Firebase Documentation, Retrieved from:
<https://firebase.google.com/docs>
- Maps SDK for Android overview, Retrieved from:
<https://developers.google.com/maps/documentation/android-sdk/overview>
- Places SDK for Android overview, Retrieved from:
<https://developers.google.com/maps/documentation/places/android-sdk/overview>
- The Activity Lifecycle, Retrieved from:
<https://developer.android.com/guide/components/activities/activity-lifecycle>
- Material UI Design, Retrieved from:
<https://material.io/>
- Update UI components with NavigationUI, Retrieved from:
<https://developer.android.com/guide/navigation/navigation-ui>
- Intent, Retrieved from:
<https://developer.android.com/reference/android/content/Intent>
- StackOverflow Website, Retrieved from:
<https://stackoverflow.com/>

נספחים

בפרק זה מצורף הקוד של מחלקות הפרויקט

BaseActivity

```
package com.example.thetrempiada.data;

import android.content.Context;
import android.graphics.Rect;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.ValueEventListener;

/**
 * This class is a base activity which all activities in the app inherit
 * from.
 * This class measures the inactivity time of a user, and if it is above
 * DISCONNECT_TIMEOUT, the user's status will be set to "offline".
 * The user status will also be offline if the user closes the app from
 * the recent menu, this is implemented via service.
 * This class also overrides onCreate and adds inside it a callback for
 * onDisconnect on firebase, which firebase server's will change the status
 * of the
 * user to offline if the app closed/something happened.
 *
 * This class will also save some important variables and constants that
 * will be used
 * all over the app.
 */

// There are practically 2 methods of setting the status of the user:
// one manual, after a timeout of 5 minutes, and one using firebase
// onDisconnect method.

public class BaseActivity extends AppCompatActivity {

    public static final long DISCONNECT_TIMEOUT = 180000; // 3 min = 5 * 60
```

```

* 1000 ms

    // string constants
    public static final String GOOGLE_MAPS_API_KEY =
    "AIzaSyDrNlEtJ6tX_5loQL1KoaguQVwx4bpGtU";
    public static final String SHARED_PREFERENCES_REMEMBER_ME = "remember";
    public static final String SHARED_PREFERENCES_LAST_USER = "lastUser";
    public static final String SHARED_PREFERENCES_MAP_TYPE = "mapType";
    public static final String SHARED_PREFERENCES_IS_SATELLITE_SWITCH =
    "isSatelliteSwitchOn";
    public static final String SHARED_PREFERENCES_IS_NIGHT_MAP_SWITCH =
    "isNightModeSwitchOn";
    public static final String INTENT_EXTRA_SATELLITE = "Satellite";
    public static final String INTENT_EXTRA_NIGHT_MAP = "NightMode";

    private View mTouchOutsideView;
    private OnTouchOutsideViewListener mOnTouchOutsideViewListener;

    /**
     * Sets a listener that is being notified when the user has tapped
     * outside a given view. To remove the listener,
     * @param view
     * @param onTouchOutsideViewListener
     */
    public void setOnTouchOutsideViewListener(View view,
    OnTouchOutsideViewListener onTouchOutsideViewListener) {
        mTouchOutsideView = view;
        mOnTouchOutsideViewListener = onTouchOutsideViewListener;
    }

    public OnTouchOutsideViewListener getOnTouchOutsideViewListener() {
        return mOnTouchOutsideViewListener;
    }

    @Override
    public boolean dispatchTouchEvent(final MotionEvent ev) {
        if (ev.getAction() == MotionEvent.ACTION_DOWN) {
            // Notify touch outside listener if user tapped outside a given
view
            if (mOnTouchOutsideViewListener != null && mTouchOutsideView !=
null
                && mTouchOutsideView.getVisibility() == View.VISIBLE) {
                Rect viewRect = new Rect();
                mTouchOutsideView.getGlobalVisibleRect(viewRect);
                if (!viewRect.contains((int) ev.getRawX(), (int)
ev.getRawY())) {
                    mOnTouchOutsideViewListener.onTouchOutside(mTouchOutsideView, ev);
                }
            }
        }
        return super.dispatchTouchEvent(ev);
    }

    /**
     * Interface definition for a callback to be invoked when a touch event
     * has occurred outside a formerly specified
     * view. See {@link #setOnTouchOutsideViewListener(View,
OnTouchOutsideViewListener).}

```

```

    */
    public interface onTouchOutsideViewListener {

        /**
         * Called when a touch event has occurred outside a given view.
         *
         * @param view The view that has not been touched.
         * @param event The MotionEvent object containing full information
         about the event.
         */
        public void onTouchOutside(View view, MotionEvent event);
    }

    private static Handler disconnectHandler = new Handler(new
    Handler.Callback() {
        @Override
        public boolean handleMessage(Message msg) {
            // ignore
            return true;
        }
    });

    private static Runnable disconnectCallback = new Runnable() {
        @Override
        public void run() {
            // this callback is running when user has passed the inactivity
time
            if(FirebaseAuth.getInstance().getCurrentUser() != null) {
                UserHelper.setOnline(FirebaseAuth.getInstance().getCurrentUser().getUid(),
                false);
            }
        }
    };

    public void resetDisconnectTimer() {
        disconnectHandler.removeCallbacks(disconnectCallback);
        disconnectHandler.postDelayed(disconnectCallback,
DISCONNECT_TIMEOUT);
    }

    public void stopDisconnectTimer() {
        disconnectHandler.removeCallbacks(disconnectCallback);
        if(FirebaseAuth.getInstance().getCurrentUser() != null)
            UserHelper.setOnline(FirebaseAuth.getInstance().getCurrentUser().getUid(),
            true);
    }

    @Override
    public void onUserInteraction() {
        // when user interacts with the app, he is no longer offline
        stopDisconnectTimer();
    }

    /**
     * this onCreate method will serve as a "global" onCreate to
     * prevent code duplication in activities

```

```

    */

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        stopDisconnectTimer();

        // when firebase detects that the user/the app has disconnected, it
        // will change the value of
        if(FirebaseAuth.getInstance().getCurrentUser() != null) {

            // write the last logged on UID to the sharedPreferences

            getPreferences(Context.MODE_PRIVATE).edit().putString(SHARED_PREFERENCES_LAST_USER, FirebaseAuth.getInstance().getCurrentUser().getUid()).apply();

            UserHelper._userDatabase

            .child(FirebaseAuth.getInstance().getCurrentUser().getUid())
                .child("_online")
                .onDisconnect()
                .setValue(false)
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        Log.d("UserOffline: ",
String.valueOf(task.isSuccessful()));
                    }
                });
        }

        // check if "trash" entries to the database exists (as a result of
        // cache), if so - delete
        // the cache will be the deleted as soon as any user opens the app
        // (if user doesn't have the field _uid, it means it is cache, and
        // needs to be deleted)
        UserHelper._userDatabase.addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                for(DataSnapshot ds : snapshot.getChildren())
                    if(!ds.hasChild("_uid")) {
                        ds.getRef().removeValue();
                    }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Log.d("FailedRemovingCache", error.getMessage());
            }
        });
    }

    // measuring the time between onPause and onResume i.e. the inactivity
    // time
    @Override
    protected void onPause() {

```

```

        super.onPause();
        resetDisconnectTimer();
    }

    @Override
    protected void onResume() {
        super.onResume();
        stopDisconnectTimer();
    }
}

```

MainActivity

```

package com.example.thetrempiada;

import android.Manifest;
import android.annotation.SuppressLint;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.location.Location;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.util.Log;
import android.view.MenuItem;
import android.view.Menu;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.bumptech.glide.Glide;

import com.example.thetrempiada.data.BaseActivity;
import com.example.thetrempiada.data.PathStorageHelper;
import com.example.thetrempiada.data.User;
import com.example.thetrempiada.data.UserHelper;
import com.example.thetrempiada.data.directionData.StorageHelper;
import com.example.thetrempiada.enums.Gender;
import com.example.thetrempiada.enums.UserType;
import com.example.thetrempiada.login.LoginActivity;
import com.google.android.gms.common.api.Status;
import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationCallback;
import com.google.android.gms.location.LocationRequest;

```



```

import com.google.android.gms.location.LocationResult;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MapStyleOptions;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.Polyline;
import com.google.android.gms.maps.model.PolylineOptions;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.android.libraries.places.api.Places;
import com.google.android.libraries.places.api.model.Place;
import com.google.android.libraries.places.widget.AutoComplete;
import com.google.android.libraries.places.widget.AutoCompleteActivity;
import com.google.android.libraries.places.widget.model.AutoCompleteActivityMode;
import com.google.android.material.card.MaterialCardView;
import com.google.android.material.navigation.NavigationView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.ValueEventListener;
import com.google.maps.DirectionsApi;
import com.google.maps.DirectionsApiRequest;
import com.google.maps.GeoApiContext;
import com.google.maps.model.DirectionsLeg;
import com.google.maps.model.DirectionsResult;
import com.google.maps.model.DirectionsRoute;
import com.google.maps.model.DirectionsStep;
import com.google.maps.model.EncodedPolyline;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.appcompat.widget.Toolbar;

import org.jetbrains.annotations.NotNull;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;

import static android.content.ContentValues.TAG;

```

```
/**
```

```

* This activity is the gateway and the main page of the application.
* It includes the map functionality.
* It handles the side menu (using drawer navigation menu) and the sidebar
menu (search button, for now).
* More functionality of this activity will be added as more options will
be added to the app.
*/

```

```

public class MainActivity extends BaseActivity implements
OnMapReadyCallback {

    // need to be public static to be accessed from a different location
    public static GoogleMap mGoogleMap;
    public static LatLng mCurLatLng;
    public static List<Polyline> activePolylines = new ArrayList<>();
    public static Marker destinationOfDriveMarker = null;
    public static Marker openedMarker = null;

    private Menu mainMenu;
    private Marker mCurrLocationMarker;
    private AppBarConfiguration mAppBarConfiguration;
    private SupportMapFragment mapFragment;
    private LocationRequest mLocationRequest;
    private FusedLocationProviderClient mFusedLocationClient;
    // mapType = GoogleMap.MAP_TYPE_SOMETHING
    private int mapType = GoogleMap.MAP_TYPE_NORMAL;
    private final int MAP_TYPE_NIGHT = 16;
    private boolean zoomedIn;

    private FirebaseAuth firebaseAuth;
    private FirebaseAuth.AuthStateListener fireAuthListener;

    // user profile card view components
    private MaterialCardView userProfileCardView;
    private ImageView cardProfilePicture;
    private TextView cardTitle, cardSecondary, cardDescription;
    private Button cardTakeTremptBtn, cardMessageBtn;

    // a main shared preferences to save every data that needs to be loaded
on activity/app launch
    // such as settings
    private SharedPreferences mainSharedPref;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);
        mainSharedPref = this.getPreferences(Context.MODE_PRIVATE);

        // getting user data
        firebaseAuth = FirebaseAuth.getInstance();
        fireAuthListener = new FirebaseAuth.AuthStateListener() {
            @Override
            public void onAuthStateChanged(@NonNull FirebaseAuth
firebaseAuth) {
                FirebaseUser tempUser = firebaseAuth.getCurrentUser();
                if(tempUser == null) {
                    // user isn't logged in

```

```

        startActivity(new Intent(MainActivity.this,
LoginActivity.class));
        finish();
    }

    }

};

// --- Setting up the side menu ---
Toolbar toolbar = findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
DrawerLayout drawer = findViewById(R.id.drawer_layout);
NavigationView navigationView = findViewById(R.id.nav_view);
// Passing each menu ID as a set of Ids because each
// menu should be considered as top level destinations.

// inside AppBarConfigurationBuilder() is all items of the side
menu.
mAppBarConfiguration = new AppBarConfiguration.Builder(
    R.id.nav_home)
    .setDrawerLayout(drawer)
    .build();
NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);
NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
NavigationUI.setupWithNavController(navigationView, navController);

// --- Setting up the map ---
mFusedLocationClient =
LocationServices.getFusedLocationProviderClient(this);
mapFragment = (SupportMapFragment) getSupportFragmentManager()
    .findFragmentById(R.id.map);
assert mapFragment != null;
mapFragment.getMapAsync(this);

this.zoomedIn = false;

// handling changes to the main activity from other activities
(such as settings)
Intent intent = getIntent();
assert intent != null;
Bundle extras = intent.getExtras();
if(extras != null) {
    // night mode
    if(intent.getBooleanExtra(INTENT_EXTRA_NIGHT_MAP, false)) {
        this.mapType = this.MAP_TYPE_NIGHT;
    }
    // satellite mode
    if(intent.getBooleanExtra(INTENT_EXTRA_SATELLITE, false)) {
        this.mapType = GoogleMap.MAP_TYPE_HYBRID;
    }
    // not night mode or satellite mode, AKA normal mode;
    if(!(intent.hasExtra(INTENT_EXTRA_NIGHT_MAP)) &&
!(intent.hasExtra(INTENT_EXTRA_SATELLITE))) {
        this.mapType = GoogleMap.MAP_TYPE_NORMAL;
    }
    saveMapTypeToPreference(this.mapType);
}
}

```

```

// Initialize the Places SDK
Places.initialize(getApplicationContext(), GOOGLE_MAPS_API_KEY);

// user profile card view components
userProfileCardView = findViewById(R.id.user_profile_card_view);
cardProfilePicture = findViewById(R.id.profile_card_picture);
cardTitle = findViewById(R.id.profile_card_title);
cardSecondary = findViewById(R.id.profile_card_secondary);
cardDescription = findViewById(R.id.profile_card_description);
cardTakeTremptBtn = findViewById(R.id.profile_card_order_trempt_btn);
cardMessageBtn = findViewById(R.id.profile_card_message_btn);

// close profile card view if clicked outside

this.setOnTouchOutsideViewListener(findViewById(R.id.user_profile_card_view)
), new onTouchOutsideViewListener() {
    @Override
    public void onTouchOutside(View view, MotionEvent event) {
        // used to prevent leaking of UID
        if(openedMarker != null) {
            openedMarker.hideInfoWindow();
        }
        userProfileCardView.setVisibility(View.GONE);
        mGoogleMap.getUiSettings().setScrollGesturesEnabled(true);
    }
});

@Override
protected void onStart() {
    super.onStart();
    firebaseAuth.addAuthStateListener(fireAuthListener);
}

@Override
protected void onStop() {
    super.onStop();

    if(fireAuthListener != null) {
        firebaseAuth.removeAuthStateListener(fireAuthListener);
    }
}

@Override
protected void onPause() {
    super.onPause();

    //stop location updates when Activity is no longer active
    if (mFusedLocationClient != null) {
        mFusedLocationClient.removeLocationUpdates(mLocationCallback);
    }
}

/**
 * Manipulates the map once available.
 * This callback is triggered when the map is ready to be used.

```

```

    * This is where we can add markers or lines, add listeners or move the
    camera. In this case,
    * we just add a marker near Sydney, Australia.
    * If Google Play services is not installed on the device, the user
    will be prompted to install
    * it inside the SupportMapFragment. This method will only be triggered
    once the user has
    * installed Google Play services and returned to the app.
    */
    @Override
    public void onMapReady(@NonNull GoogleMap googleMap) {
        // setting map type
        mGoogleMap = googleMap;
        if(!(this.mainSharedPref.getInt(SHARED_PREFERENCES_MAP_TYPE, 1) ==
16)) {

googleMap.setMapType(this.mainSharedPref.getInt(SHARED_PREFERENCES_MAP_TYPE
, 1));
        }
        else {

googleMap.setMapStyle(MapStyleOptions.loadRawResourceStyle(this,
R.raw.night_map));
        }

        mLocationRequest = new LocationRequest();
        mLocationRequest.setInterval(5000); // 5s interval
        mLocationRequest.setFastestInterval(5000);

mLocationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURA
CY);

        if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {
                // Location Permission already granted

mFusedLocationClient.requestLocationUpdates(mLocationRequest,
mLocationCallback, Looper.myLooper());
                mGoogleMap.setMyLocationEnabled(true);
            } else {
                // Request Location Permission
                checkLocationPermission();
            }
        }
        else {
            mFusedLocationClient.requestLocationUpdates(mLocationRequest,
mLocationCallback, Looper.myLooper());
            mGoogleMap.setMyLocationEnabled(true);
        }

        // raise a profile page on user marker click
        mGoogleMap.setOnMarkerClickListener(new
GoogleMap.OnMarkerClickListener() {
            @Override
            public boolean onMarkerClick(@NonNull Marker marker) {
                if(marker.getSnippet() != null &&
marker.getSnippet().length() == 28) {

```

```

        // marker is a user marker
        // pointer to use later, when dismissing the card view
        openedMarker = marker;
        if (userProfileCardView.getVisibility() != View.VISIBLE)
    {
        String uid = marker.getSnippet();

        UserHelper._userDatabase.child(uid).addValueEventListener(new
        ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
            snapshot) {
                User user = snapshot.getValue(User.class);

                StorageHelper._storageReference.child("profilePictures").child(user.get_uid
               ()).getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                    @Override
                    public void onSuccess(Uri uri) {
                        Glide.with(MainActivity.this)
                            .load(uri)

                            .placeholder(R.drawable.banner_profile)

                            .error(R.drawable.banner_profile)

                                .into(cardProfilePicture);
                    }
                });

                cardTitle.setText(user.get_displayName());

                final String CAR_EMOJI = "\uD83D\uDE97 ";
                final String TREMPIST_EMOJI =
                "\uD83D\uDE4B\u200D🚗 ";

                final String MALE_EMOJI = "♂ ";
                final String FEMALE_EMOJI = "♀ ";

                String userType = (user.get_userType() ==
                UserType.DRIVER ? CAR_EMOJI : user.get_userType() == UserType.TREMPIST ?
                TREMPIST_EMOJI : "") +

                user.get_userType().toString().substring(0, 1) +
                user.get_userType().toString().substring(1).toLowerCase();
                if (user.get_userType() == UserType.BOTH) {
                    userType = CAR_EMOJI + TREMPIST_EMOJI +
                    "Driver and Tremapist";
                }

                String userGender = (user.get_gender() ==
                Gender.MALE ? MALE_EMOJI : FEMALE_EMOJI)
                +
                user.get_gender().toString().substring(0, 1) +
                user.get_gender().toString().substring(1).toLowerCase();

                cardSecondary.setText(userType + " | " +
                userGender);
            }
        });
    }
}

```

```

cardDescription.setText(user.get_description());

        if (user.get_userType() !=
UserType.TREMPIST) {
cardTakeTremptBtn.setVisibility(View.VISIBLE);
        }

        // TODO : implement Message and OrderTrempt
buttons.
cardMessageBtn.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // todo
        }
    });

cardTakeTremptBtn.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // todo
        }
    });

    @Override
    public void onCancelled(@NonNull DatabaseError
error) {
        Log.e("DatabaseError", error.getMessage());
    }
});
userProfileCardView.setVisibility(View.VISIBLE);
// prevent the user from moving the map while
profile card view visible
mGoogleMap.getUiSettings().setScrollGesturesEnabled(false);
    }
    }
    return false;
}
});

}

// function to quickly save map type to the shared preference
public void saveMapTypeToPreference(int mapType) {
    SharedPreferences.Editor editor =
getPreferences(Context.MODE_PRIVATE).edit();
    editor.putInt(SHARED_PREFERENCES_MAP_TYPE, mapType);
    editor.apply();
}

LocationCallback mLocationCallback = new LocationCallback() {

    /**
     * This section (callback) is being called every 5s, and will

```

```

update every data connected
    * to user on the map, such as current location marker, other
user's markers and more.
    */

@Override
public void onLocationResult(LocationResult locationResult) {
    List<Location> locationList = locationResult.getLocations();
    if (locationList.size() > 0) {
        // The last location in the list is the newest
        Location location = locationList.get(locationList.size() -
1);
        Log.i("MapsActivity", "Location: " + location.getLatitude()
+ " " + location.getLongitude());
        if (mCurrLocationMarker != null) {
            mCurrLocationMarker.remove();
        }

        // Place current location marker and add the location to
the user database
        FirebaseUser currentUser =
FirebaseAuth.getInstance().getCurrentUser();
        mCurLatLng = new LatLng(location.getLatitude(),
location.getLongitude());
        assert currentUser != null;

        UserHelper._userDatabase.child(currentUser.getId()).child("_currentLocatio
n").setValue(new
com.example.thetrempiada.data.directionData.LatLng(mCurLatLng.latitude,
mCurLatLng.longitude));
        MarkerOptions markerOptions = new MarkerOptions();
        markerOptions.position(mCurLatLng);
        markerOptions.title(currentUser.getDisplayName());
        markerOptions.snippet("Current location");

        markerOptions.icon(BitmapDescriptorFactory.defaultMarker(210));

        // just before the location updates, the map will clear all
markers to make sure all
        // offline user markers are cleared
        mGoogleMap.clear();
        mCurrLocationMarker = mGoogleMap.addMarker(markerOptions);

        /**
    * this section will handle map changes according to a list
of online users
    * for example, marker to close online users and more
    * all tasks will be done in realtime
    */

        UserHelper._userDatabase.addValueEventListener(new
ValueEventListener() {

            // --- Anonymous class ---

            // List to hold online users
            List<User> onlineUsers;

            @Override

```



```

        public void onDataChange(@NonNull DataSnapshot
snapshot) {

            onlineUsers = new ArrayList<User>();

            for (DataSnapshot ds : snapshot.getChildren()) {
                if
(ds.child("_online").getValue(Boolean.class)) {
                    onlineUsers.add(ds.getValue(User.class));
                }
            }
            // draw marker on the map, of all online users in
100km range - for now
            for (User user : onlineUsers) {
                if (firebaseAuth.getCurrentUser() != null &&
                    // the user isn't the current user

!user.get_uid().equals(firebaseAuth.getCurrentUser().getUid()) &&
                    // user has current location
                    user.get_currentLocation() != null &&
                    // user is within a distance of 100km
from current user's location

distanceInKilometers(mCurLatLng.latitude, mCurLatLng.longitude,
user.get_currentLocation().getLatitude(),
user.get_currentLocation().getLongitude()) <= 100000.0) {
                    MarkerOptions markerOptions = new
MarkerOptions();
                    markerOptions.position(new
LatLng(user.get_currentLocation().getLatitude(),
user.get_currentLocation().getLongitude()));

markerOptions.title(user.get_displayName());

markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFa
ctory.HUE_ORANGE));

                    markerOptions.snippet(user.get_uid());
                    mGoogleMap.addMarker(markerOptions);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Log.d("ErrorGettingUsersData", error.getMessage());
            }
        });

        // move camera to current location and zoom in, only if not
zoomed in already
        if(!zoomedIn) {

mGoogleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(mCurLatLng, 15));
            zoomedIn = true;
        }

        mainMenu.findItem(R.id.action_new_drive).setVisible(true);
        mainMenu.findItem(R.id.action_take_tremp).setVisible(true);
    }
}

```

```

    }
}

};

public static final int MY_PERMISSIONS_REQUEST_LOCATION = 99;
private void checkLocationPermission() {
    if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED) {

        // Should we show an explanation?
        if (ActivityCompat.shouldShowRequestPermissionRationale(this,
            Manifest.permission.ACCESS_FINE_LOCATION)) {

            // Show an explanation to the user *asynchronously* --
            // don't block
            // this thread waiting for the user's response! After the
            // user
            // sees the explanation, try again to request the
            // permission.
            new AlertDialog.Builder(this)
                .setTitle("Location Permission Needed")
                .setMessage("This app needs the Location
permission, please accept to use location functionality")
                .setPositiveButton("OK", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface
dialogInterface, int i) {
                        //Prompt the user once explanation has been
                        shown
                        ActivityCompat.requestPermissions(MainActivity.this,
                            new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                                MY_PERMISSIONS_REQUEST_LOCATION );
                    }
                })
                .create()
                .show();

        } else {
            // No explanation needed, we can request the permission.
            ActivityCompat.requestPermissions(this,
                new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                    MY_PERMISSIONS_REQUEST_LOCATION );
        }
    }

    // result of location permission allowed / rejected
    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           @NotNull String[] permissions,
                                           @NotNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
    }
}

```

```

        if (requestCode == MY_PERMISSIONS_REQUEST_LOCATION) {// If request
is cancelled, the result arrays are empty.
            if (grantResults.length > 0
                && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {

                // permission was granted, yay! Do the
                // location-related task you need to do.
                if (ContextCompat.checkSelfPermission(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)
                    == PackageManager.PERMISSION_GRANTED) {

mFusedLocationClient.requestLocationUpdates(mLocationRequest,
mLocationCallback, Looper.myLooper());
                    mGoogleMap.setMyLocationEnabled(true);
                }

            } else {

                // permission denied, boo! Disable the
                // functionality that depends on this permission.
                Toast.makeText(this, "Permission denied",
Toast.LENGTH_LONG).show();
            }

            // other 'case' lines to check for other
            // permissions this app might request
        }
    }

    // create the default menu
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        this.mainMenu = menu;
        getMenuInflater().inflate(R.menu.main_activity_menu, menu);
        return true;
    }

    /**
     * default menu / action bar buttons functionality.
     * search button is presented but without functionality yet.
     * more may be added in the future.
     */

    private static final int AUTOCOMPLETE_REQUEST_CODE_NEW_DRIVE = 1;
    private static final int AUTOCOMPLETE_REQUEST_CODE_TAKE_TREMP = 2;

    @SuppressWarnings("NonConstantResourceId")
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Launch autocomplete search view (with Places SDK) using an
Intent

        // Set the fields to specify which types of place data to
        // return after the user has made a selection.
        List<Place.Field> fields = Arrays.asList(Place.Field.ID,

```

```

Place.Field.NAME, Place.Field.LAT_LNG);
    // create an autocomplete intent
    Intent intent = new Autocomplete
        .IntentBuilder(AutocompleteActivityMode.OVERLAY, fields)
        .setCountry("IL")
        .build(this);

    // switch case for handling button clicks
    switch (item.getItemId()) {
    case R.id.action_take_tremp:
        startActivityForResult(intent,
AUTOCOMPLETE_REQUEST_CODE_TAKE_TREMP);
        break;
    case R.id.action_new_drive:
        // Start the autocomplete intent.
        startActivityForResult(intent,
AUTOCOMPLETE_REQUEST_CODE_NEW_DRIVE);
        break;
    case R.id.action_dismiss_route:
        item.setVisible(false);
        mainMenu.findItem(R.id.action_new_drive).setVisible(true);
        mainMenu.findItem(R.id.action_take_tremp).setVisible(true);
        // remove any existing polylines and destination marker
        for(Polyline polyline : activePolylines) {
            polyline.remove();
        }
        if(destinationOfDriveMarker != null) {
            destinationOfDriveMarker.remove();
            destinationOfDriveMarker = null;
        }
        // remove the path of the current user from the database

PathStorageHelper._pathDatabase.child(firebaseAuth.getCurrentUser().getUid(
)).removeValue().addOnCompleteListener(new OnCompleteListener<Void>() {
    @Override
    public void onComplete(@NonNull Task<Void> task) {
        Log.d("RemovedUserPath",
String.valueOf(task.isSuccessful()));
    }
});
        break;
    }
    return super.onOptionsItemSelected(item);
}

@Override
protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
    // if request code is one of the two buttons
    if (requestCode == AUTOCOMPLETE_REQUEST_CODE_NEW_DRIVE ||
requestCode == AUTOCOMPLETE_REQUEST_CODE_TAKE_TREMP) {

        if (resultCode == RESULT_OK) {

            // remove any existing polylines and destination marker
            for(Polyline polyline : activePolylines) {
                polyline.remove();
            }
            if(destinationOfDriveMarker != null) {

```

```

        destinationOfDriveMarker.remove();
        destinationOfDriveMarker = null;
    }

    // set go back button visible
    MenuItem goBackBtn =
mainMenu.findItem(R.id.action_dismiss_route);
    goBackBtn.setVisible(true);

    mainMenu.findItem(R.id.action_new_drive).setVisible(false);

mainMenu.findItem(R.id.action_take_trempp).setVisible(false);

    assert data != null;
    Place place = Autocomplete.getPlaceFromIntent(data);

    // if action is new drive:
    if(requestCode == AUTOCOMPLETE_REQUEST_CODE_NEW_DRIVE &&
place.getLatLng() != null) {

        // for now, draw a route on the map to the location and
place a marker
        List<LatLng> path = drawGeoPolyline(mCurLatLng,
place.getLatLng());

        // upload the list to firebase

PathStorageHelper._pathDatabase.child(firebaseAuth.getCurrentUser().getUid(
)).setValue(path).addOnCompleteListener(new OnCompleteListener<Void>() {
    @Override
    public void onComplete(@NonNull Task<Void> task) {
        Log.d("UploadedUserPath",
String.valueOf(task.isSuccessful()));
    }
});

        // Draw the polyline, zoom in and draw destination
marker.
        if (path != null && path.size() > 0) {
            PolylineOptions opts = new
PolylineOptions().addAll(path).color(0xDF3480eb).width(15);
            activePolylines.add(mGoogleMap.addPolyline(opts));

mGoogleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(mCurLatLng, 18));
            destinationOfDriveMarker = mGoogleMap.addMarker(new
MarkerOptions()
                .position(place.getLatLng())
                .title(place.getName())
                .snippet("Destination")

                .icon(BitmapDescriptorFactory.defaultMarker(210)));
        }

        mGoogleMap.getUiSettings().setZoomControlsEnabled(true);
    }
    // if action is take trempp:
    // check all the active user's routes if they are clost to
you
    if(requestCode == AUTOCOMPLETE_REQUEST_CODE_TAKE_TREMP) {

```

```

        ProgressDialog progressDialog =
ProgressDialog.show(MainActivity.this, "Looking for a tremp", "Please
wait...", true);

PathStorageHelper._pathDatabase.addValueEventListener(new
ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot
snapshot) {
        for(DataSnapshot paths :
snapshot.getChildren()) {
            UserHelper._userDatabase.child(paths.getKey()).addValueEventListener(new
ValueEventListener() {
                @Override
                public void onDataChange(@NonNull
DataSnapshot snapshot) {
                    boolean isAhead = false;
                    User user =
                    for(DataSnapshot latLng :
paths.getChildren()) {
                        LatLng value =
                        if(value.latitude ==
                        && value.longitude ==
                        user.get_currentLocation().getLatitude()
                        user.get_currentLocation().getLongitude()) {
                            isAhead = true;
                            continue;
                        }
                        if(isAhead) {
                            // we are now at the
                            // check if any of those
                            // LatLngs are in 250m distance of the user
                            // WIP
                        }
                    }
                }
            });
        }
    }

    @Override
    public void onCancelled(@NonNull
DatabaseError error) {
    }
});
});
});

@Override
public void onCancelled(@NonNull DatabaseError
error) {
}
});
});

```

```

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                Toast.makeText(MainActivity.this, "Couldn't
find a tremp for you!", Toast.LENGTH_SHORT).show();
                progressDialog.dismiss();
            }
        }, 3000);
    }
} else if (resultCode == AutocompleteActivity.RESULT_ERROR) {
    Status status = Autocomplete.getStatusFromIntent(data);
    Toast.makeText(this, "Error: " + status.getStatusMessage(),
Toast.LENGTH_SHORT).show();
    Log.i("AutocompleteError", status.getStatusMessage());
} else if (resultCode == RESULT_CANCELED) {
    mainMenu.findItem(R.id.action_new_drive).setVisible(true);
    mainMenu.findItem(R.id.action_take_tremp).setVisible(true);
}
return;
}
super.onActivityResult(requestCode, resultCode, data);
}

/**
 * This function draws a route on the map (an actual drive route)
 * @param origin LatLng
 * @param destination LatLng
 * @return List of LatLngs as the path
 * @rtype List<LatLng>
 */

public List<LatLng> drawGeoPolyline(LatLng origin, LatLng destination)
{
    // Define list to get all LatLng for the route
    List<LatLng> path = new ArrayList();

    // Execute Directions API request
    GeoApiContext context = new GeoApiContext.Builder()
        .apiKey(GOOGLE_MAPS_API_KEY)
        .build();
    DirectionsApiRequest req = DirectionsApi.getDirections(context,
String.valueOf(origin.latitude) + "," + String.valueOf(origin.longitude),
String.valueOf(destination.latitude) + "," +
String.valueOf(destination.longitude));
    try {
        DirectionsResult res = req.await();

        // Loop through Legs and steps to get encoded polylines of each
step
        if (res.routes != null && res.routes.length > 0) {
            DirectionsRoute route = res.routes[0];

            if (route.legs != null) {
                for(int i=0; i<route.legs.length; i++) {
                    DirectionsLeg leg = route.legs[i];
                    if (leg.steps != null) {
                        for (int j=0; j<leg.steps.length; j++){
                            DirectionsStep step = leg.steps[j];
                            if (step.steps != null && step.steps.length

```

```
>0) {  
    for (int k=0; k<step.steps.length;k++){  
        DirectionsStep step1 =  
            EncodedPolyline points1 =  
                if (points1 != null) {  
                    // Decode polyline and add  
points to list of route coordinates  
List<com.google.maps.model.LatLng> coords1 = points1.decodePath();  
for  
(com.google.maps.model.LatLng coord1 : coords1) {  
path.add(new  
LatLng(coord1.lat, coord1.lng));  
}  
} else {  
EncodedPolyline points = step.polyline;  
if (points != null) {  
// Decode polyline and add points  
to list of route coordinates  
List<com.google.maps.model.LatLng>  
for (com.google.maps.model.LatLng  
path.add(new LatLng(coord.lat,  
coord.lng));  
}  
}  
}  
}  
}  
catch(Exception ex) {  
    if(ex.getMessage() != null) {  
Log.e(TAG, ex.getMessage());  
    } else {  
Toast.makeText(this, "Cannot navigate to that location!",  
Toast.LENGTH_SHORT).show();  
mainMenu.findItem(R.id.action_dismiss_route).setVisible(false);  
mainMenu.findItem(R.id.action_new_drive).setVisible(true);  
mainMenu.findItem(R.id.action_take_tremp).setVisible(true);  
return null;  
}  
return path;  
}  
  
@Override  
// settings up what clicking on hamburger (|||) button will actually do  
public boolean onSupportNavigateUp() {  
NavController navController = Navigation.findNavController(this,  
R.id.nav_host_fragment);  
return NavigationUI.navigateUp(navController, appBarConfiguration)  
|| super.onSupportNavigateUp();  

```



```

    }

    /**
     * This function finds the distance (on a sphere i.e. earth), between
     two points in kilometers.
     * The function calculations is based on Haversine formula.
     * @param lat1 : @type double : Latitude of the first point
     * @param lon1 : @type double : Longitude of the first point
     * @param lat2 : @type double : Latitude of the second point
     * @param lon2 : @type double : Longitude of the second point
     * @return : returns the distance between two points in kilometers
     * @rtype double
     */

    public double distanceInKilometers(double lat1, double lon1, double
lat2, double lon2) {
        // The math module contains a function
        // named toRadians which converts from
        // degrees to radians.
        lon1 = Math.toRadians(lon1);
        lon2 = Math.toRadians(lon2);
        lat1 = Math.toRadians(lat1);
        lat2 = Math.toRadians(lat2);

        // Haversine formula
        double dlon = lon2 - lon1;
        double dlat = lat2 - lat1;
        double a = Math.pow(Math.sin(dlat / 2), 2)
            + Math.cos(lat1) * Math.cos(lat2)
            * Math.pow(Math.sin(dlon / 2), 2);

        double c = 2 * Math.asin(Math.sqrt(a));

        // Radius of earth in kilometers.
        final double r = 6371;

        // calculate the result
        return(c * r);
    }
}

```

LoginActivity

```

package com.example.thetrempiada.login;

import android.app.Dialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextUtils;
import android.text.TextWatcher;
import android.util.Patterns;

```

```

import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.app.AppCompatActivity;

import com.example.thetrempiada.MainActivity;
import com.example.thetrempiada.R;
import com.example.thetrempiada.data.BaseActivity;
import com.example.thetrempiada.data.UserHelper;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.switchmaterial.SwitchMaterial;
import com.google.android.material.textfield.TextInputEditText;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class LoginActivity extends BaseActivity {

    private FirebaseAuth firebaseAuth;
    private TextInputEditText logEmail, logPassword;
    private TextInputLayout passwordContainer, emailContainer;
    private TextView forgotPassword;
    private Button btnLogin, btnRegister;
    private ProgressBar progressBar;
    private SwitchMaterial remember;
    private boolean emailValid = false, passwordValid = false;

    // required to be public static to be accessed from settings activity
    // when signing out
    public static SharedPreferences sp;

    // used for reset password dialog
    private boolean resetPasswordEmailValid = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // disable night theme (launcher-activity)
        AppCompatActivity.setDefaultNightMode(AppCompatActivity.MODE_NIGHT_NO);

        // getting the instance of firebase

        firebaseAuth = FirebaseAuth.getInstance();
        sp = this.getSharedPreferences(Context.MODE_PRIVATE);

        // checking if some sort of "cache" is still saved, if so -
        // disconnecting
        if(!sp.getBoolean(SHARED_PREFERENCES_REMEMBER_ME, false) &&
        FirebaseAuth.getInstance().getCurrentUser() != null) {

```

```

UserHelper.setOnLine(FirebaseAuth.getInstance().getCurrentUser().getUid(),
false);
        firebaseAuth.signOut();
    }

    // auto login process
    // move to main activity if user already signed in

    if(sp.getBoolean(SHARED_PREFERENCES_REMEMBER_ME, false) &&
FirebaseAuth.getInstance().getCurrentUser() != null) {
        UserHelper.setOnLine(FirebaseAuth.getInstance().getUid(),
true);
        Intent intent = new Intent(LoginActivity.this,
MainActivity.class);
        startActivity(new Intent(LoginActivity.this,
MainActivity.class));
        finish();
    }

    setContentView(R.layout.activity_login);

    logEmail = findViewById(R.id.username_login);
    logPassword = findViewById(R.id.password_login);

    passwordContainer = findViewById(R.id.password_login_container);
    emailContainer = findViewById(R.id.username_login_container);

    btnLogin = findViewById(R.id.login);
    btnRegister = findViewById(R.id.go_to_register);

    progressBar = findViewById(R.id.loading_login);
    remember = findViewById(R.id.remember_switch);

    forgotPassword = findViewById(R.id.forgot_password);

    // go to register activity

    btnRegister.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            startActivity(new Intent(LoginActivity.this,
SignUpActivity.class));
        }
    });

    // start login process if btnLogin is clicked

    btnLogin.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String email = logEmail.getText().toString();
            String password = logPassword.getText().toString();

            if (TextUtils.isEmpty(email)) {

Toast.makeText(LoginActivity.this.getApplicationContext(), "Enter email
address!", Toast.LENGTH_SHORT).show();
                return;
            }
        }
    });

```

```

    }

    if (TextUtils.isEmpty(password)) {

Toast.makeText(LoginActivity.this, getApplicationContext(), "Enter
password!", Toast.LENGTH_SHORT).show();
        return;
    }

    progressBar.setVisibility(View.VISIBLE);

    // Login user using signInWithEmailAndPassword function and
    handling errors
    firebaseAuth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull
Task<AuthResult> task) {

            progressBar.setVisibility(View.GONE);

            if(!task.isSuccessful()) {
                Toast.makeText(LoginActivity.this,
"Error! " + task.getException(), Toast.LENGTH_SHORT).show();
            }
            else {
                SharedPreferences.Editor editor =
sp.edit();

                editor.putBoolean(SHARED_PREFERENCES_REMEMBER_ME, remember.isChecked());
                editor.apply();
                // changing user status to online

                UserHelper.setOnline(FirebaseAuth.getInstance().getCurrentUser().getUid(),
true);

                Intent intent = new
Intent(LoginActivity.this, MainActivity.class);
                startActivity(intent);
                finish();
            }
        }
    });

    });

    // blocking and enabling login button according to patterns of
    correct email and password
    logEmail.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int
count, int after) {
            emailContainer.setError(null);
        }

        @Override
        public void onTextChanged(CharSequence s, int start, int
before, int count) {
            if(Patterns.EMAIL_ADDRESS.matcher(s).matches()) {
                emailValid = true;
            }
        }
    });

```

```

        } else if(s.length() > 0) {
            emailValid = false;
            emailContainer.setError("Not a valid email!");
        } else {
            emailContainer.setError("This field cannot be blank!");
        }
    }

    @Override
    public void afterTextChanged(Editable s) {
        btnLogin.setEnabled(emailValid && passwordValid);
    }
});

logPassword.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int
count, int after) {
        passwordContainer.setError(null);
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int
before, int count) {
        if(s.toString().matches("^(?=.*[a-zA-Z])\\w{6,16}$")) {
            passwordValid = true;
        } else if(s.length() > 0) {
            passwordValid = false;
            passwordContainer.setError("Password must be 6-16
numbers, letters or underscore, and contain at least 1 letter.");
        } else {
            passwordContainer.setError("This field cannot be
blank!");
        }
    }

    @Override
    public void afterTextChanged(Editable s) {
        btnLogin.setEnabled(emailValid && passwordValid);
    }
});

// go to reset password if clicked on forgot password
forgotPassword.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        Dialog resetPasswordDialog = new
Dialog(LoginActivity.this);

        resetPasswordDialog setContentView(R.layout.reset_password_dialog);
        resetPasswordDialog.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
        resetPasswordDialog.show();

        TextInputEditText email =
resetPasswordDialog.findViewById(R.id.reset_password_email);
        TextInputLayout emailContainer =
resetPasswordDialog.findViewById(R.id.reset_password_email_container);

```

```

        Button reset =
resetPasswordDialog.findViewById(R.id.btn_reset_password);

        // blocking and enabling reset button according to correct
email pattern
        email.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int
start, int count, int after) {
                emailContainer.setError(null);
            }

            @Override
            public void onTextChanged(CharSequence s, int start,
int before, int count) {
                if(Patterns.EMAIL_ADDRESS.matcher(s).matches()) {
                    resetPasswordEmailValid = true;
                } else if(s.length() > 0) {
                    resetPasswordEmailValid = false;
                    emailContainer.setError("Not a valid email!");
                } else {
                    emailContainer.setError("This field cannot be
blank!");
                }
            }

            @Override
            public void afterTextChanged(Editable s) {
                reset.setEnabled(resetPasswordEmailValid);
            }
        });

        // send reset email when reset button is enabled
reset.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                progressBar.setVisibility(View.VISIBLE);

                firebaseAuth.sendPasswordResetEmail(email.getText().toString())
                    .addOnCompleteListener(new
OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull
Task<Void> task) {
                            if(task.isSuccessful()) {

                                Toast.makeText(LoginActivity.this, "Instructions to reset your password
have been sent to you by mail.", Toast.LENGTH_SHORT).show();
                            } else {

                                Toast.makeText(LoginActivity.this, "Failed to send reset email!",
                                Toast.LENGTH_SHORT).show();
                            }
                        }
                    });

                progressBar.setVisibility(View.GONE);
                resetPasswordDialog.dismiss();
            }
        });
    }
}

```

```

    });
  }
});
}
}

```

SignUpActivity

```

package com.example.thetrempiada.login;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.textEditable;
import android.text.TextWatcher;
import android.util.Log;
import android.util.Patterns;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.Toast;

import com.example.thetrempiada.MainActivity;
import com.example.thetrempiada.R;
import com.example.thetrempiada.data.BaseActivity;
import com.example.thetrempiada.data.User;
import com.example.thetrempiada.data.UserHelper;
import com.example.thetrempiada.enums.Gender;
import com.example.thetrempiada.enums.UserType;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.textfield.TextInputEditText;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.UserProfileChangeRequest;

import java.util.Objects;

public class SignUpActivity extends BaseActivity {

    private FirebaseAuth firebaseAuth;
    private static final String TAG = "SignUpActivity";
    private TextInputEditText regEmail, regPassword, regConfirmPassword,
    regName, regAge;
    private TextInputLayout emailContainer, passwordContainer,
    confirmPasswordContainer, nameContainer;
    private AutoCompleteTextView regGender, regType;
    private static final String[] GENDER_PATHS = {"Male", "Female"};
    private static final String[] TYPE_PATHS = {"Driver", "Tremapist",
    "Both"};

```

```

    private Button register;
    private ProgressBar progressBar;
    private boolean emailValid = false, passwordValid = false, nameValid =
false, confirmPasswordValid = false, genderSelected = false, typeSelected =
false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_up);

        firebaseAuth = FirebaseAuth.getInstance();

        regEmail = findViewById(R.id.register_username);
        regPassword = findViewById(R.id.register_password);
        regConfirmPassword = findViewById(R.id.register_confirm_password);
        regAge = findViewById(R.id.register_age);
        regName = findViewById(R.id.register_display_name);

        emailContainer = findViewById(R.id.email_container);
        passwordContainer = findViewById(R.id.password_container);
        confirmPasswordContainer =
findViewById(R.id.confirm_password_container);
        nameContainer = findViewById(R.id.name_container);

        register = findViewById(R.id.btn_register);
        progressBar = findViewById(R.id.loading_register);

        // setting up the gender dropdown menu
        regGender = findViewById(R.id.reg_gender);
        ArrayAdapter<String> genderAdapter = new
ArrayAdapter<String>(SignUpActivity.this, R.layout.dropdownmenu_items,
GENDER_PATHS);
        genderAdapter.setDropDownViewResource(R.layout.dropdownmenu_items);
        regGender.setAdapter(genderAdapter);
        regGender.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view,
int position, long id) {
                genderSelected = true;
                register.setEnabled(emailValid && passwordValid &&
nameValid && confirmPasswordValid && genderSelected && typeSelected);
            }
        });

        // setting up the user type dropdown menu
        regType = findViewById(R.id.reg_type);
        ArrayAdapter<String> typeAdapter = new
ArrayAdapter<String>(SignUpActivity.this, R.layout.dropdownmenu_items,
TYPE_PATHS);
        typeAdapter.setDropDownViewResource(R.layout.dropdownmenu_items);
        regType.setAdapter(typeAdapter);
        regType.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view,
int position, long id) {
                typeSelected = true;

```



```

        register.setEnabled(emailValid && passwordValid &&
nameValid && confirmPasswordValid && genderSelected && typeSelected);
    }
});

// registering user on click of sign up button
register.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        registerUser();
    }
});

// checking validity of fields in order to activate the button
regEmail.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int
count, int after) {
        emailContainer.setError(null);
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int
before, int count) {
        if(Patterns.EMAIL_ADDRESS.matcher(s).matches()) {
            emailValid = true;
        } else if(s.length() > 0) {
            emailValid = false;
            emailContainer.setError("Not a valid email!");
        } else {
            emailContainer.setError("This field cannot be blank!");
        }
    }

    @Override
    public void afterTextChanged(Editable s) {
        register.setEnabled(emailValid && passwordValid &&
nameValid && confirmPasswordValid && genderSelected && typeSelected);
    }
});

regPassword.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int
count, int after) {
        passwordContainer.setError(null);
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int
before, int count) {
        if(s.toString().matches("^(?=[a-zA-Z])\\w{6,16}$")) {
            passwordValid = true;
        } else if(s.length() > 0) {
            passwordValid = false;
            passwordContainer.setError("Password must be 6-16
numbers, letters or underscores, and contain at least 1 letter.");
        } else {
            passwordContainer.setError("This field cannot be

```

```

blank!"));
    }
}

@Override
public void afterTextChanged(Editable s) {
    register.setEnabled(emailValid && passwordValid &&
nameValid && confirmPasswordValid && genderSelected && typeSelected);
}
});

regConfirmPassword.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int
count, int after) {
        confirmPasswordContainer.setError(null);
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int
before, int count) {
        if(s.toString().equals(regPassword.getText().toString())) {
            confirmPasswordValid = true;
        } else if(s.length() > 0) {
            confirmPasswordValid = false;
            confirmPasswordContainer.setError("Confirmation doesn't
match!");
        } else {
            confirmPasswordContainer.setError("This field cannot be
blank!");
        }
    }
}

@Override
public void afterTextChanged(Editable s) {
    register.setEnabled(emailValid && passwordValid &&
nameValid && confirmPasswordValid && genderSelected && typeSelected);
}
});

regName.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int
count, int after) {
        nameContainer.setError(null);
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int
before, int count) {
        if(s.toString().matches("^\\w\\.]{3,16}$")) {
            nameValid = true;
        } else if(s.length() > 0) {
            nameValid = false;
            nameContainer.setError("Display name must be 3-16
numbers, letters, spaces, underscores or dots.");
        } else {
            nameContainer.setError("This field cannot be blank!");
        }
    }
}

```

```

    }

    @Override
    public void afterTextChanged(Editable s) {
        register.setEnabled(emailValid && passwordValid &&
nameValid && confirmPasswordValid && genderSelected && typeSelected);
    }
});

}

private void registerUser() {
    String userEmail = regEmail.getText().toString();
    String userPassword = regPassword.getText().toString();

    progressBar.setVisibility(View.VISIBLE);

    // register user
    firebaseAuth.createUserWithEmailAndPassword(userEmail,
userPassword)
        .addOnCompleteListener(SignUpActivity.this, new
OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task)
{
                Log.d(TAG, "New user registration: " +
task.isSuccessful());
                if (!task.isSuccessful()) {
                    Toast.makeText(SignUpActivity.this,
"Authentication failed." + task.getException(), Toast.LENGTH_SHORT).show();
                    progressBar.setVisibility(View.GONE);
                } else {
                    String userDisplayName =
regName.getText().toString();

                    int userAge = 0;
                    if (!regAge.getText().toString().matches(""))
                        userAge =
Integer.parseInt(regAge.getText().toString());
                    else userAge = -1;

                    Gender userGender = null;
                    switch (regGender.getText().toString()) {
                        case "Male":
                            userGender = Gender.MALE;
                            break;
                        case "Female":
                            userGender = Gender.FEMALE;
                            break;
                    }

                    UserType userType = null;
                    switch (regType.getText().toString()) {
                        case "Driver":
                            userType = UserType.DRIVER;
                            break;
                        case "Tremapist":
                            userType = UserType.TREMPIST;
                            break;
                    }
                }
            }
        });
}

```

```

        case "Both":
            userType = UserType.BOTH;
            break;
    }

    UserProfileChangeRequest displayName = new
    UserProfileChangeRequest.Builder().setDisplayName(userDisplayName).build();

    Objects.requireNonNull(firebaseAuth.getCurrentUser()).updateProfile(display
    Name);

    //adding user to database
    User user = new
    User(firebaseAuth.getCurrentUser().getUid(), userDisplayName, userAge,
    userGender, userType);

    user.set_online(true);
    UserHelper.addUserToDatabase(user,
    user.get_uid(), SignUpActivity.this);

    progressBar.setVisibility(View.GONE);

    //starting the app
    SignUpActivity.this.startActivity(new
    Intent(SignUpActivity.this, MainActivity.class));
    SignUpActivity.this.finish();
    }
    }
    });
}
}

```

SettingsActivity

```

package com.example.thetrempiada.ui.settings;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.app.Dialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.graphics.LinearGradient;
import android.graphics.Shader;
import android.graphics.Typeface;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.os.Handler;
import android.text.Editable;
import android.text.TextPaint;
import android.text.TextWatcher;

```

```

import android.util.Patterns;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.example.thetrempiada.MainActivity;
import com.example.thetrempiada.R;
import com.example.thetrempiada.data.BaseActivity;
import com.example.thetrempiada.data.UserHelper;
import com.example.thetrempiada.login.LoginActivity;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.switchmaterial.SwitchMaterial;
import com.google.android.material.textfield.TextInputEditText;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.auth.AuthCredential;
import com.google.firebase.auth.EmailAuthProvider;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

import org.w3c.dom.Text;

public class SettingsActivity extends BaseActivity {

    private SwitchMaterial satelliteSwitch, nightModeSwitch;
    private TextView mapSettingsTitle, profileSettingsTitle;
    private Button changeEmail, changePassword, editProfile, deleteUser;
    private Intent intent;

    // used for change password dialog
    private boolean newPasswordValid = false, oldPasswordValid = false,
    alrClickedChangePassword;

    // used for change email dialog
    private boolean newEmailValid = false, confirmPasswordValid = false,
    alrClickedChangeEmail;

    // used for delete user dialog
    private boolean passwordValid = false, alrClickedDeleteUser;

    private FirebaseUser user;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_settings);

        // getting current user
        user = FirebaseAuth.getInstance().getCurrentUser();

        // apply to custom toolbar from XML
        Toolbar toolbar = findViewById(R.id.toolbar_settings);
        toolbar.setTitle("Settings");

```

```

setSupportActionBar(toolbar);

intent = new Intent(this, MainActivity.class);
// this flags are added so when we exit SettingsActivity, old
instance of MainActivity is deleted
// only the new one with updated settings will be presented.

intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TA
SK);

this.nightModeSwitch = findViewById(R.id.nightModeSwitch);
this.satelliteSwitch = findViewById(R.id.satelliteSwitch);

// saving the states of the switched to shared prefernces
SharedPreferences sharedPref =
getPreferences(Context.MODE_PRIVATE);

this.nightModeSwitch.setChecked(sharedPref.getBoolean(SHARED_PREFERENCES_IS
_NIGHT_MAP_SWITCH, false));

this.satelliteSwitch.setChecked(sharedPref.getBoolean(SHARED_PREFERENCES_IS
_SATELLITE_SWITCH, false));

satelliteSwitch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
        intent.putExtra(INTENT_EXTRA_SATELLITE, isChecked);

editSwitchPreferences(SHARED_PREFERENCES_IS_SATELLITE_SWITCH, isChecked);
        if (isChecked) {
            nightModeSwitch.setChecked(false);

editSwitchPreferences(SHARED_PREFERENCES_IS_NIGHT_MAP_SWITCH, false);
        }
    }
});

nightModeSwitch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
        intent.putExtra(INTENT_EXTRA_NIGHT_MAP, isChecked);

editSwitchPreferences(SHARED_PREFERENCES_IS_NIGHT_MAP_SWITCH, isChecked);
        if (isChecked) {
            satelliteSwitch.setChecked(false);

editSwitchPreferences(SHARED_PREFERENCES_IS_SATELLITE_SWITCH, false);
        }
    }
});

// settings the gradient color of the titles
// MUST add the first color of the gradient as textColor in the XML
for it to work!
mapSettingsTitle = findViewById(R.id.mapSettingsTitle);

```

```

    profileSettingsTitle = findViewById(R.id.profileSettingsTitle);

    TextPaint paint = mapSettingsTitle.getPaint();
    float width =
    paint.measureText(mapSettingsTitle.getText().toString());
    Shader textShader = new LinearGradient(0, 0, width,
    mapSettingsTitle.getTextSize(), new int[]{
        Color.parseColor("#FF9800"),
        Color.parseColor("#FFC107"),
        Color.parseColor("#FF5722"),
    }, null, Shader.TileMode.CLAMP);
    mapSettingsTitle.getPaint().setShader(textShader);
    profileSettingsTitle.getPaint().setShader(textShader);

    changeEmail = findViewById(R.id.change_email);
    changePassword = findViewById(R.id.change_password);
    editProfile = findViewById(R.id.btn_edit_profile);
    deleteUser = findViewById(R.id.delete_user);

    // custom font for buttons
    Typeface font = Typeface.createFromAsset(getAssets(), "Roboto-
    Regular.ttf");
    changePassword.setTypeface(font);
    editProfile.setTypeface(font);
    changeEmail.setTypeface(font);
    deleteUser.setTypeface(font);

    // initializing check booleans for already clicked on changed
    button
    alrClickedChangeEmail = false;
    alrClickedChangePassword = false;
    alrClickedDeleteUser = false;

    // dialogs for changing the profile settings
    // change email dialog
    changeEmail.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            Dialog changeEmailDialog = new
            Dialog(SettingsActivity.this);

            changeEmailDialog setContentView(R.layout.change_email_dialog);
            changeEmailDialog.getWindow().setBackgroundDrawable(new
            ColorDrawable(Color.TRANSPARENT));
            changeEmailDialog.show();

            TextInputEditText newEmail =
            changeEmailDialog.findViewById(R.id.new_email_edit);
            TextInputEditText confirmPassword =
            changeEmailDialog.findViewById(R.id.confirm_password_edit);
            TextInputLayout emailContainer =
            changeEmailDialog.findViewById(R.id.change_email_email_container);
            TextInputLayout passwordContainer =
            changeEmailDialog.findViewById(R.id.change_email_password_container);
            Button confirmChanges =
            changeEmailDialog.findViewById(R.id.confirm_email_change);
            ProgressBar progressBar =
            changeEmailDialog.findViewById(R.id.loading_change_email);

```

```

        // blocking and enabling change button according to
        patterns of correct email and password
        newEmail.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int
start, int count, int after) {
                emailContainer.setError(null);
            }

            @Override
            public void onTextChanged(CharSequence s, int start,
int before, int count) {
                if (Patterns.EMAIL_ADDRESS.matcher(s).matches()) {
                    newEmailValid = true;
                } else if (s.length() > 0) {
                    newEmailValid = false;
                    emailContainer.setError("Not a valid email!");
                } else {
                    emailContainer.setError("This field cannot be
blank!");
                }
            }

            @Override
            public void afterTextChanged(Editable s) {
                confirmChanges.setEnabled(newEmailValid &&
confirmPasswordValid);
            }
        });

        confirmPassword.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int
start, int count, int after) {
                passwordContainer.setError(null);
            }

            @Override
            public void onTextChanged(CharSequence s, int start,
int before, int count) {
                if (s.toString().matches("^(?=.*[a-zA-
Z])\\w{6,16}$")) {
                    confirmPasswordValid = true;
                } else if (s.length() > 0) {
                    confirmPasswordValid = false;
                    passwordContainer.setError("Not a valid
password!");
                } else {
                    passwordContainer.setError("This field cannot
be blank!");
                }
            }

            @Override
            public void afterTextChanged(Editable s) {
                confirmChanges.setEnabled(newEmailValid &&
confirmPasswordValid);
            }
        });

```



```

    });

    // button is enabled if newEmailValid &&
confirmPasswordValid
    confirmChanges.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View v) {

            if (!alrClickedChangeEmail) {
                alrClickedChangeEmail = true;

                Toast.makeText(SettingsActivity.this, "Click
again to confirm!", Toast.LENGTH_SHORT).show();

                // slowmode
                confirmChanges.setEnabled(false);
                Handler handler = new Handler();
                handler.postDelayed(new Runnable() {
                    @Override
                    public void run() {
                        confirmChanges.setEnabled(true);
                    }
                }, 500);
                return;
            }

            alrClickedChangeEmail = false;

            progressBar.setVisibility(View.VISIBLE);
            final String email = user.getEmail();
            assert email != null;
            AuthCredential credential =
EmailAuthProvider.getCredential(email,
confirmPassword.getText().toString());

            // using reauthenticate to assure user has entered
the right old password, if succeeded, update to new email

            user.reauthenticate(credential).addOnCompleteListener(new
OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void>
task) {

                    if (task.isSuccessful()) {

                        user.updateEmail(newEmail.getText().toString())
                            .addOnCompleteListener(new
OnCompleteListener<Void>() {
                                @Override
                                public void
onComplete(@NonNull Task<Void> task) {

                                    if
(task.isSuccessful()) {

                                        Toast.makeText(SettingsActivity.this, "Email is updated. Please sign in
with the new Email!", Toast.LENGTH_SHORT).show();

                                        progressBar.setVisibility(View.GONE);

```

```
// signing out
UserHelper.setOnLine(FirebaseAuth.getInstance().getUid(), false);
FirebaseAuth.getInstance().signOut();

SharedPreferences.Editor editor = LoginActivity.sp.edit();
editor.putBoolean(SHARED_PREFERENCES_REMEMBER_ME, false);
editor.apply();
startActivity(new
Intent(SettingsActivity.this, LoginActivity.class));
finish();

} else {

Toast.makeText(SettingsActivity.this, "Failed to update email!",
Toast.LENGTH_SHORT).show();

progressBar.setVisibility(View.GONE);

}

});

} else {
    Toast.makeText(SettingsActivity.this,
"Wrong password!", Toast.LENGTH_SHORT).show();
    progressBar.setVisibility(View.GONE);
}

});

});

});

});

// change password dialog
changePassword.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        Dialog changePasswordDialog = new
Dialog(SettingsActivity.this);

changePasswordDialog setContentView(R.layout.change_password_dialog);
changePasswordDialog.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
changePasswordDialog.show();

        TextInputEditText newPassword =
changePasswordDialog.findViewById(R.id.new_password_edit);
        TextInputEditText oldPassword =
changePasswordDialog.findViewById(R.id.old_password_edit);
        TextInputLayout newPasswordContainer =
changePasswordDialog.findViewById(R.id.change_password_new_password_contain
er);

        TextInputLayout oldPasswordContainer =
changePasswordDialog.findViewById(R.id.change_password_old_password_contain
er);
```

```

        Button confirmChanges =
changePasswordDialog.findViewById(R.id.confirm_password_change);
        ProgressBar progressBar =
changePasswordDialog.findViewById(R.id.loading_change_password);

        // blocking and enabling change button according to
patterns of correct email and password
        newPassword.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int
start, int count, int after) {
                newPasswordContainer.setError(null);
            }

            @Override
            public void onTextChanged(CharSequence s, int start,
int before, int count) {
                if (s.toString().matches("(?=.*[a-zA-
Z])\\w{6,16}$")) {
                    newPasswordValid = true;
                } else if (s.length() > 0) {
                    newPasswordValid = false;
                    newPasswordContainer.setError("Not a valid
password!");
                } else {
                    newPasswordContainer.setError("This field
cannot be blank!");
                }
            }

            @Override
            public void afterTextChanged(Editable s) {
                confirmChanges.setEnabled(oldPasswordValid &&
newPasswordValid);
            }
        });

        oldPassword.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int
start, int count, int after) {
                oldPasswordContainer.setError(null);
            }

            @Override
            public void onTextChanged(CharSequence s, int start,
int before, int count) {
                if (s.toString().matches("(?=.*[a-zA-
Z])\\w{6,16}$")) {
                    oldPasswordValid = true;
                } else if (s.length() > 0) {
                    oldPasswordValid = false;
                    oldPasswordContainer.setError("Not a valid
password!");
                } else {
                    oldPasswordContainer.setError("This field
cannot be blank!");
                }
            }
        });

```

```

        @Override
        public void afterTextChanged(Editable s) {
            confirmChanges.setEnabled(oldPasswordValid &&
newPasswordValid);
        }
    });

    // button is enabled if oldPasswordValid &&
newPasswordValid
    confirmChanges.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View v) {

            if (!alrClickedChangePassword) {
                alrClickedChangePassword = true;

                Toast.makeText(SettingsActivity.this, "Click
again to confirm!", Toast.LENGTH_SHORT).show();

                // slowmode
                confirmChanges.setEnabled(false);
                Handler handler = new Handler();
                handler.postDelayed(new Runnable() {
                    @Override
                    public void run() {
                        confirmChanges.setEnabled(true);
                    }
                }, 500);
                return;
            }

            alrClickedChangePassword = false;

            progressBar.setVisibility(View.VISIBLE);
            final String email = user.getEmail();
            assert email != null;
            AuthCredential credential =
EmailAuthProvider.getCredential(email, oldPassword.getText().toString());

            // using reauthenticate to assure user has entered
the right old password, if succeeded, update to new password

            user.reauthenticate(credential).addOnCompleteListener(new
OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void>
task) {

                    if (task.isSuccessful()) {

                        user.updatePassword(newPassword.getText().toString())
                            .addOnCompleteListener(new
OnCompleteListener<Void>() {
                                @Override
                                public void
onComplete(@NonNull Task<Void> task) {

                                    if
(task.isSuccessful()) {

```

```

Toast.makeText(SettingsActivity.this, "Password is updated. Please sign in
with the new password!", Toast.LENGTH_SHORT).show();

progressBar.setVisibility(View.GONE);

// signing out
UserHelper.setOnline(FirebaseAuth.getInstance().getUid(), false);
FirebaseAuth.getInstance().signOut();

SharedPreferences.Editor editor = LoginActivity.sp.edit();
editor.putBoolean(SHARED_PREFERENCES_REMEMBER_ME, false);
editor.apply();
startActivity(new
Intent(SettingsActivity.this, LoginActivity.class));
finish();
} else {

Toast.makeText(SettingsActivity.this, "Failed to update password!",
Toast.LENGTH_SHORT).show();

progressBar.setVisibility(View.GONE);

}
});
} else {
progressBar.setVisibility(View.GONE);
Toast.makeText(SettingsActivity.this,
"Wrong password!", Toast.LENGTH_SHORT).show();
}
});
}
});
});
});
});

// starting edit profile menu (activity)
editProfile.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
startActivity(new Intent(SettingsActivity.this,
ProfileSettingsActivity.class));
finish();
}
});

// delete user dialog
deleteUser.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {

Dialog deleteUserDialog = new
Dialog(SettingsActivity.this);

deleteUserDialog setContentView(R.layout.delete_user_dialog);

```

```

        deleteUserDialog.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
        deleteUserDialog.show();

        TextInputEditText confirmPassword =
deleteUserDialog.findViewById(R.id.confirm_password_delete_edit);
        TextInputLayout passwordContainer =
deleteUserDialog.findViewById(R.id.delete_user_password_container);
        Button confirmChanges =
deleteUserDialog.findViewById(R.id.confirm_delete_user);
        ProgressBar progressBar =
deleteUserDialog.findViewById(R.id.loading_delete_user);

        // checking if password is valid in order to enable button
confirmPassword.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int
start, int count, int after) {
        passwordContainer.setError(null);
    }

    @Override
    public void onTextChanged(CharSequence s, int start,
int before, int count) {
        if(s.toString().matches("^(?=.*[a-zA-
Z])\\w{6,16}$")) {
            passwordValid = true;
        } else if(s.length() > 0) {
            passwordValid = false;
            passwordContainer.setError("Password must be 6-
16 numbers, letters or underscores, and contain at least 1 letter.");
        } else {
            passwordContainer.setError("This field cannot
be blank!");
        }
    }

    @Override
    public void afterTextChanged(Editable s) {
        confirmChanges.setEnabled(passwordValid);
    }
});

confirmChanges.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View v) {

        if (!alrClickedDeleteUser) {
            alrClickedDeleteUser = true;

            Toast.makeText(SettingsActivity.this, "Click
again to confirm!", Toast.LENGTH_SHORT).show();

            // slowmode
            confirmChanges.setEnabled(false);
            Handler handler = new Handler();
            handler.postDelayed(new Runnable() {
                @Override

```

```

        public void run() {
            confirmChanges.setEnabled(true);
        }
    }, 500);
    return;
}

alrClickedDeleteUser = false;

progressBar.setVisibility(View.VISIBLE);
final String email = user.getEmail();
assert email != null;
AuthCredential credential =
EmailAuthProvider.getCredential(email,
confirmPassword.getText().toString());

user.reauthenticate(credential).addOnCompleteListener(new
OnCompleteListener<Void>() {
    @Override
    public void onComplete(@NonNull Task<Void>
task) {
        if(task.isSuccessful()) {
            String userId = user.getUserId();
            user.delete().addOnCompleteListener(new
OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull
Task<Void> task) {
                    if(task.isSuccessful()) {
                        // user successfully
                        deleted, delete user data from database
                        UserHelper.deleteUserFromDatabase(userId, SettingsActivity.this);
                        progressBar.setVisibility(View.GONE);

                        // signing out
                        FirebaseAuth.getInstance().signOut();
                        SharedPreferences.Editor
                        editor = LoginActivity.sp.edit();
                        editor.putBoolean(SHARED_PREFERENCES_REMEMBER_ME, false);
                        editor.apply();
                        startActivity(new
Intent(SettingsActivity.this, LoginActivity.class));
                        finish();
                    } else {
                        Toast.makeText(SettingsActivity.this, "Error deleting user!",
                        Toast.LENGTH_SHORT).show();
                        progressBar.setVisibility(View.GONE);
                    }
                }
            });
        } else {
            Toast.makeText(SettingsActivity.this,
            "Wrong password!", Toast.LENGTH_SHORT).show();

```

```

        progressBar.setVisibility(View.GONE);
    }
    });
}
});
}
});
}

public void editSwitchPreferences(String label, boolean isChecked) {
    SharedPreferences sharedPref =
this.getPreferences(Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPref.edit();
    editor.putBoolean(label, isChecked);
    editor.apply();
}

// create the menu which will store the back button (action button).
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is
    present.
    getMenuInflater().inflate(R.menu.go_back_menu, menu);
    return true;
}

// when the action button is clicked, go back.
// used switch in case of future addons.
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_go_back:
            onBackPressed();
    }
    return super.onOptionsItemSelected(item);
}

@Override
public void onBackPressed() {
    Bundle extras = intent.getExtras();
    if(extras == null) {
        intent.putExtra(INTENT_EXTRA_NIGHT_MAP,
nightModeSwitch.isChecked());
        intent.putExtra(INTENT_EXTRA_SATELLITE,
satelliteSwitch.isChecked());
    }
    startActivity(intent);
    finish();
}
}

```


ProfileSettingsActivity

```
package com.example.thetrempiada.ui.settings;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.app.ProgressDialog;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.graphics.LinearGradient;
import android.graphics.Shader;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.provider.MediaStore;
import android.text.Editable;
import android.text.TextPaint;
import android.text.TextWatcher;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.example.thetrempiada.R;
import com.example.thetrempiada.data.BaseActivity;
import com.example.thetrempiada.data.UserHelper;
import com.example.thetrempiada.data.directionData.StorageHelper;
import com.example.thetrempiada.enums.Gender;
import com.example.thetrempiada.enums.UserType;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.textfield.TextInputEditText;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.auth.AuthCredential;
import com.google.firebase.auth.EmailAuthProvider;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.UserProfileChangeRequest;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.storage.FileDownloadTask;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
```

```

import com.google.firebase.storage.UploadTask;

import java.io.IOException;
import java.util.Objects;

public class ProfileSettingsActivity extends BaseActivity {

    private TextView editProfileTitle;
    private ImageView editProfilePicture;
    private TextInputEditText editDisplayName, editAge, editDescription,
confirmPassword;
    private TextInputLayout nameContainer, passwordContainer;
    private AutoCompleteTextView editGender, editUserType;
    private static final String[] GENDER_PATHS = {"Male", "Female"};
    private static final String[] TYPE_PATHS = {"Driver", "Tremapist",
"Both"};
    private Button confirmChanges;
    private ProgressBar progressBar;
    private DatabaseReference databaseReference;
    private FirebaseUser currentUser;

    // variables used for uploading a photo
    private static Uri filePath;
    private static final StorageReference storageReference =
StorageHelper._storageReference.child("profilePictures");
    private final int PICK_IMAGE_REQUEST = 22;

    private boolean displayNameValid = true, confirmPasswordValid = false,
genderSelected = false, userTypeSelected = false;
    private boolean alrClickedConfirm = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profile_settings);

        databaseReference =
FirebaseDatabase.getInstance("https://thetrempiada-5e41a-default-
rtdb.europe-west1.firebaseio.com/").getReference();
        currentUser = FirebaseAuth.getInstance().getCurrentUser();

        // apply to custom toolbar from XML
        Toolbar toolbar = findViewById(R.id.toolbar_settings_profile);
        toolbar.setTitle("Profile Settings");
        setSupportActionBar(toolbar);

        // styling the title
        editProfileTitle = findViewById(R.id.edit_profile_title);
        TextPaint paint = editProfileTitle.getPaint();
        float width =
paint.measureText(editProfileTitle.getText().toString());
        Shader textShader = new LinearGradient(0, 0, width,
editProfileTitle.getTextSize(), new int[] {
            Color.parseColor("#FF9800"),
            Color.parseColor("#FFC107"),
            Color.parseColor("#FF5722"),
        }, null, Shader.TileMode.CLAMP);
        editProfileTitle.getPaint().setShader(textShader);

```

```

// uploading profile photo
editProfilePicture = findViewById(R.id.edit_profile_picture);
// on pressing the image, the user will be moved to gallery to
select a file
editProfilePicture.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Defining Implicit Intent to mobile gallery
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_OPEN_DOCUMENT);
        startActivityForResult(
            Intent.createChooser(
                intent,
                "Select Image from here..."),
            PICK_IMAGE_REQUEST);
    }
});

editDisplayName = findViewById(R.id.edit_display_name);
editAge = findViewById(R.id.edit_age);
editDescription = findViewById(R.id.edit_description);
confirmPassword = findViewById(R.id.edit_confirm_password);

nameContainer = findViewById(R.id.edit_name_container);
passwordContainer =
findViewById(R.id.confirm_password_to_edit_container);

// blocking and enabling change button according to pattern of
correct name and password confirmation
editDisplayName.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int
count, int after) {
        nameContainer.setError(null);
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int
before, int count) {
        if(s.toString().matches("^([\\w \\.] {3,16})") {
            displayNameValid = true;
        } else if(s.length() > 0) {
            displayNameValid = false;
            nameContainer.setError("Not a valid display name!");
        } else {
            // display name is valid if empty because display name
doesn't have to be changed
            displayNameValid = true;
        }
    }

    @Override
    public void afterTextChanged(Editable s) {
        confirmChanges.setEnabled(displayNameValid &&
confirmPasswordValid);
    }
});

```

```

        confirmPassword.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int
count, int after) {
                passwordContainer.setError(null);
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int
before, int count) {
                if(s.toString().matches("^(?=.*[a-zA-Z])\\w{6,16}$")) {
                    confirmPasswordValid = true;
                } else if(s.length() > 0) {
                    confirmPasswordValid = false;
                    passwordContainer.setError("Invalid password!");
                } else {
                    passwordContainer.setError("This field cannot be
blank!");
                }
            }

            @Override
            public void afterTextChanged(Editable s) {
                confirmChanges.setEnabled(displayNameValid &&
confirmPasswordValid);
            }
        });

        // creating the dropdown menus
        editGender = findViewById(R.id.edit_gender);
        ArrayAdapter<String> genderAdapter = new
ArrayAdapter<String>(ProfileSettingsActivity.this,
R.layout.dropdownmenu_items, GENDER_PATHS);
        genderAdapter.setDropDownViewResource(R.layout.dropdownmenu_items);
        editGender.setAdapter(genderAdapter);
        editGender.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view,
int position, long id) {
                genderSelected = true;
            }
        });

        editUserType = findViewById(R.id.edit_type);
        ArrayAdapter<String> typeAdapter = new
ArrayAdapter<String>(ProfileSettingsActivity.this,
R.layout.dropdownmenu_items, TYPE_PATHS);
        typeAdapter.setDropDownViewResource(R.layout.dropdownmenu_items);
        editUserType.setAdapter(typeAdapter);
        editUserType.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view,
int position, long id) {
                userTypeSelected = true;
            }
        });
    }
}

```

```

progressBar = findViewById(R.id.loading_change_profile);

confirmChanges = findViewById(R.id.confirm_profile_changes);

// attempting to change profile settings on button click
confirmChanges.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        if(!alrClickedConfirm) {
            alrClickedConfirm = true;

            Toast.makeText(ProfileSettingsActivity.this, "Click
again to confirm!", Toast.LENGTH_SHORT).show();

            // slowmode
            confirmChanges.setEnabled(false);
            Handler handler = new Handler();
            handler.postDelayed(new Runnable() {
                @Override
                public void run() {
                    confirmChanges.setEnabled(true);
                }
            }, 500);
            return;
        }

        alrClickedConfirm = false;

        progressBar.setVisibility(View.VISIBLE);

        // using reauthentication to assure user has entered the old
password correctly
        AuthCredential authCredential =
EmailAuthProvider.getCredential(currentUser.getEmail(),
confirmPassword.getText().toString());

        currentUser.reauthenticate(authCredential).addOnCompleteListener(new
OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if(task.isSuccessful()) {
                    // if task is successful, user has entered the
right password, so we can proceed and edit the profile
                    // profile name

if(!editDisplayName.getText().toString().matches("")) {
                        UserProfileChangeRequest displayName = new
UserProfileChangeRequest.Builder().setDisplayName(editDisplayName.getText()
.toString()).build();

                        Objects.requireNonNull(currentUser.updateProfile(displayName)).addOnComple
eListener(new OnCompleteListener<Void>() {
                            @Override
                            public void onComplete(@NonNull
Task<Void> task) {
                                if(!task.isSuccessful()) {

                                    Toast.makeText(ProfileSettingsActivity.this, "An error has occurred,

```

```

display name is not updated", Toast.LENGTH_SHORT).show();
    } else {

UserHelper._userDatabase.child(currentUser.getId()).child("_displayName").
setValue(editDisplayName.getText().toString()).addOnCompleteListener(new
OnCompleteListener<Void>() {

    @Override
    public void

onComplete(@NonNull Task<Void> task) {

    if(!task.isSuccessful()) {

Toast.makeText(ProfileSettingsActivity.this, "An error has occurred,
display name is not updated", Toast.LENGTH_SHORT).show();
    }
    }
    });
    }
    });
    }
    // age
    if(!editAge.getText().toString().matches("")) {

UserHelper._userDatabase.child(currentUser.getId()).child("_age").setValue
(Integer.parseInt(editAge.getText().toString()).addOnCompleteListener(new
OnCompleteListener<Void>() {

    @Override
    public void onComplete(@NonNull

Task<Void> task) {

        if(!task.isSuccessful()) {

Toast.makeText(ProfileSettingsActivity.this, "An error has occurred, age is
not updated", Toast.LENGTH_SHORT).show();
    }
    }
    });
    }
    // description

    if(!editDescription.getText().toString().matches("")) {

UserHelper._userDatabase.child(currentUser.getId()).child("_description").
setValue(editDescription.getText().toString()).addOnCompleteListener(new
OnCompleteListener<Void>() {

    @Override
    public void onComplete(@NonNull

Task<Void> task) {

        if(!task.isSuccessful()) {

Toast.makeText(ProfileSettingsActivity.this, "An error has occurred,
description is not updated", Toast.LENGTH_SHORT).show();
    }
    }
    });
    }
    // gender
    if(genderSelected) {
        Gender gender = null;

```

```

        switch (editGender.getText().toString()) {
            case "Male":
                gender = Gender.MALE;
                break;
            case "Female":
                gender = Gender.FEMALE;
                break;
        }

        UserHelper._userDatabase.child(currentUser.getUid()).child("_gender").setValue(gender).addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull

Task<Void> task) {

                if(!task.isSuccessful()) {

                    Toast.makeText(ProfileSettingsActivity.this, "An error has occurred, gender
                    is not updated", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
    // userType
    if(userTypeSelected) {
        UserType userType = null;
        switch (editUserType.getText().toString())
    {
        case "Driver":
            userType = UserType.DRIVER;
            break;
        case "Tremapist":
            userType = UserType.TREMPIST;
            break;
        case "Both":
            userType = UserType.BOTH;
            break;
    }

    UserHelper._userDatabase.child(currentUser.getUid()).child("_userType").setValue(userType).addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull

Task<Void> task) {

            if(!task.isSuccessful()) {

                Toast.makeText(ProfileSettingsActivity.this, "An error has occurred, user
                type is not updated", Toast.LENGTH_SHORT).show();
            }
        }
    });
}
// profile picture
if(filePath != null) {
    // Upload the file to firebase storage

    storageReference.child(currentUser.getUid()).getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
        @Override
        public void onSuccess(Uri uri) {

```

```

// File already exists, delete and
make a new one

storageReference.child(currentUser.getUid()).delete();
    }
});

storageReference.child(currentUser.getUid()).putFile(filePath).addOnComple
eListener(new OnCompleteListener<UploadTask.TaskSnapshot>() {
    @Override
    public void onComplete(@NonNull
Task<UploadTask.TaskSnapshot> task) {
        Log.d("UploadedToStorage",
String.valueOf(task.isSuccessful()));
        if(task.isSuccessful()) {
            // download file and put it in
user auth database

storageReference.child(currentUser.getUid()).getDownloadUrl().addOnComple
Listener(new OnCompleteListener<Uri>() {
    @Override
    public void
onComplete(@NonNull Task<Uri> task) {
        if(task.isSuccessful())
        {
            UserProfileChangeRequest pictureChangeRequest = new
UserProfileChangeRequest.Builder().setPhotoUri(task.getResult()).build();

currentUser.updateProfile(pictureChangeRequest).addOnCompleteListener(new
OnCompleteListener<Void>() {
    @Override
    public void
onComplete(@NonNull Task<Void> task) {
        if(!task.isSuccessful()) {
            Toast.makeText(ProfileSettingsActivity.this, "An error has occurred,
profile picture is not updated", Toast.LENGTH_SHORT).show();
        }
    }
});
        } else {
            Toast.makeText(ProfileSettingsActivity.this, "An error has occurred,
profile picture is not updated", Toast.LENGTH_SHORT).show();
        }
    }
});
        } else {
            Toast.makeText(ProfileSettingsActivity.this, "An error has occurred,
profile picture is not updated", Toast.LENGTH_SHORT).show();
        }
    }
});
    }

progressBar.setVisibility(View.GONE);

```



```

        startActivity(new
Intent(ProfileSettingsActivity.this, SettingsActivity.class));
        finish();
    } else {
        Toast.makeText(ProfileSettingsActivity.this,
"Wrong password!", Toast.LENGTH_SHORT).show();
        progressBar.setVisibility(View.GONE);
    }
    }
    });
    });
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent
data) {

    super.onActivityResult(requestCode,
        resultCode,
        data);

    // checking request code and result code
    // if request code is PICK_IMAGE_REQUEST and
    // resultCode is RESULT_OK
    // then set image in the image view
    if (requestCode == PICK_IMAGE_REQUEST
        && resultCode == RESULT_OK
        && data != null
        && data.getData() != null) {

        // Get the Uri of data
        filePath = data.getData();
        try {
            // Setting image on image view using Bitmap
            Bitmap bitmap = MediaStore
                .Images
                .Media
                .getBitmap(
                    getContentResolver(),
                    filePath);
            editProfilePicture.setImageBitmap(bitmap);
        } catch (IOException e) {
            // Log the exception
            e.printStackTrace();
        }
    }

    // create the menu which will store the back button (action button).
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
        present.
        getMenuInflater().inflate(R.menu.go_back_menu, menu);
        return true;
    }
}

```

```

// when the action button is clicked, go back.
// used switch in case of future addons.
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_go_back:
            onBackPressed();
    }
    return super.onOptionsItemSelected(item);
}

@Override
public void onBackPressed() {
    super.onBackPressed();
    startActivity(new Intent(ProfileSettingsActivity.this,
SettingsActivity.class));
    finish();
}
}

```

HomeFragment

```

package com.example.thetrempiada.ui.home;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProvider;

import com.example.thetrempiada.MainActivity;
import com.example.thetrempiada.R;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.material.floatingactionbutton.FloatingActionButton;

/**
 * This is the home, practically main fragment.
 * It will be launched above the map with the app launching, and can be
 * accessed via "Home" button in the side menu.
 * It contains layouts which will float above the map on the main page
 * (such as the current "welcome" layout).
 */

public class HomeFragment extends Fragment {

    private FloatingActionButton btnCenterMap;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle
savedInstanceState) {

```

```

        View view = inflater.inflate(R.layout.fragment_home, container,
false);

        this.btnCenterMap = view.findViewById(R.id.btn_center_map);
        btnCenterMap.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(MainActivity.mCurLatLng != null)

MainActivity.mGoogleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(MainAc
tivity.mCurLatLng, 15));
            }
        });

        return view;
    }
}

```

ProfileFragment

```

package com.example.thetrempiada.ui.profile;

import android.content.Intent;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import com.bumptech.glide.Glide;
import com.example.thetrempiada.R;
import com.example.thetrempiada.data.User;
import com.example.thetrempiada.data.directionData.StorageHelper;
import com.example.thetrempiada.ui.settings.ProfileSettingsActivity;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.material.textfield.TextInputEditText;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.io.IOException;
import java.util.Objects;

/**
 * This fragment will show profile and profile data.

```

```
* Work in progress.
*/
```

```
public class ProfileFragment extends Fragment {

    private ImageView profilePicture;
    private TextView usernameTitle, usernameType;
    private TextInputEditText usernameDescription;
    private ImageButton close, goToSettings;
    private FirebaseAuth firebaseAuth;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle
savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_profile, container,
false);

        profilePicture = view.findViewById(R.id.profilePicture);
        usernameTitle = view.findViewById(R.id.usernameLine);
        usernameType = view.findViewById(R.id.userTypeLine);
        usernameDescription = view.findViewById(R.id.description_view);

        // getting users data
        firebaseAuth = FirebaseAuth.getInstance();
        // setting usernameTitle to user's display name

        usernameTitle.setText(Objects.requireNonNull(firebaseAuth.getCurrentUser())
.getDisplayName());
        // settings profilePicture to user's profile picture
        Glide.with(ProfileFragment.this)
            .load(firebaseAuth.getCurrentUser().getPhotoUrl())
            .placeholder(R.drawable.ic_profile)
            .error(R.drawable.ic_profile)
            .into(profilePicture);

        // getting user data from realtime database and updating profile at
real time
        DatabaseReference database =
FirebaseDatabase.getInstance("https://thetrempiada-5e41a-default-
rtdb.europe-west1.firebaseio.com/").getReference();

        database.child("users").child(firebaseAuth.getCurrentUser().getUid()).addVa
lueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                // getting current user at real time
                User user = snapshot.getValue(User.class);
                assert user != null;

                switch (user.get_userType().toString()) {
                    case "DRIVER":
                        usernameType.setText("Driver");
                        break;
                    case "TREMPIST":
                        usernameType.setText("Tremapist");
                        break;
                    case "BOTH":

```

```

        usernameType.setText("Driver and Trempist");
        break;
    }
    usernameDescription.setText(user.get_description());
}

@Override
public void onCancelled(@NonNull DatabaseError error) {

}

});

close = view.findViewById(R.id.btn_x);
close.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        requireActivity().onBackPressed();
    }
});

goToSettings = view.findViewById(R.id.btn_go_to_profile_settings);
goToSettings.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(requireActivity(),
ProfileSettingsActivity.class));
        requireActivity().finish();
    }
});

return view;
}
}

```

ContactFragment

```

package com.example.thetrempiada.ui.contact;

import android.app.Dialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.net.Uri;
import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageButton;
import android.widget.Toast;

import com.example.thetrempiada.MainActivity;

```

```

import com.example.thetrempiada.R;

public class ContactFragment extends Fragment {

    private ImageButton email, phone;
    private final String appEmail = "zywn1414@gmail.com";
    private final String appPhone = "0528485980";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Dialog contactDialog = new Dialog(getContext());
        contactDialog.setContentView(R.layout.fragment_contact);
        contactDialog.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
        contactDialog.show();

        email = contactDialog.findViewById(R.id.btn_email);
        phone = contactDialog.findViewById(R.id.btn_phone);

        email.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_SENDTO,
Uri.parse("mailto:" + appEmail));
                startActivity(intent);
            }
        });

        phone.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_DIAL,
Uri.parse("tel:" + appPhone));
                startActivity(intent);
            }
        });

        contactDialog.setOnDismissListener(new
DialogInterface.OnDismissListener() {
            @Override
            public void onDismiss(DialogInterface dialog) {
                requireActivity().onBackPressed();
            }
        });
    }
}

```

AboutFragment

```

package com.example.thetrempiada.ui.about;

import android.app.Dialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;

import androidx.fragment.app.DialogFragment;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.example.thetrempiada.MainActivity;
import com.example.thetrempiada.R;
import com.example.thetrempiada.ui.home.HomeFragment;

import java.util.Objects;

/**
 * This fragment will launch a dialog containing "about" information.
 */

public class AboutFragment extends Fragment {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Dialog aboutDialog = new Dialog(getActivity());
        aboutDialog setContentView(R.layout.fragment_about);
        aboutDialog.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
        aboutDialog.show();

        // when dialog dismissed, go back to main page.
        aboutDialog.setOnDismissListener(new
DialogInterface.OnDismissListener() {
            @Override
            public void onDismiss(DialogInterface dialog) {
                requireActivity().onBackPressed();
            }
        });
    }
}

```

ExitFragment

```
package com.example.thetrempiada.ui.exit;

import android.app.Dialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageButton;

import com.example.thetrempiada.MainActivity;
import com.example.thetrempiada.R;
import com.example.thetrempiada.data.BaseActivity;
import com.example.thetrempiada.data.UserHelper;
import com.example.thetrempiada.login.LoginActivity;
import com.google.firebase.auth.FirebaseAuth;

import java.util.Objects;

public class ExitFragment extends Fragment {

    private ImageButton yesExit, noExit;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Dialog exitDialog = new Dialog(getActivity());
        exitDialog setContentView(R.layout.fragment_exit);
        exitDialog.getWindow().setBackgroundDrawable(new
        ColorDrawable(Color.TRANSPARENT));
        exitDialog.show();

        yesExit = exitDialog.findViewById(R.id.yes_exit);
        noExit = exitDialog.findViewById(R.id.no_exit);

        yesExit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                UserHelper.setOnline(FirebaseAuth.getInstance().getCurrentUser().getUid(),
                false);

                FirebaseAuth.getInstance().signOut();
                SharedPreferences.Editor editor = LoginActivity.sp.edit();
```



```

editor.putBoolean(BaseActivity.SHARED_PREFERENCES_REMEMBER_ME, false);
        editor.apply();
        Intent intent = new Intent(requireActivity(),
LoginActivity.class);
        startActivity(intent);
        requireActivity().finish();
    }
});

noExit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        exitDialog.dismiss();
    }
});

exitDialog.setOnDismissListener(new
DialogInterface.OnDismissListener() {
    @Override
    public void onDismiss(DialogInterface dialog) {
        requireActivity().onBackPressed();
    }
});
}
}

```

GpsLocationReceiver

```

package com.example.thetrempiada.receivers;

import android.app.AlertDialog;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.location.LocationManager;
import android.util.Patterns;
import android.view.WindowManager;
import android.widget.Toast;

import com.example.thetrempiada.MainActivity;
import com.example.thetrempiada.data.UserHelper;
import com.google.android.material.dialog.MaterialAlertDialogBuilder;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class GpsLocationReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        if(intent.getAction().matches("android.location.PROVIDERS_CHANGED")) {
            LocationManager locationManager = (LocationManager)
context.getSystemService(Context.LOCATION_SERVICE);
            boolean gpsEnabled =
locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
            FirebaseUser currentUser =

```

```

FirebaseAuth.getInstance().getCurrentUser();
    if(!gpsEnabled) {
        Toast.makeText(context, "GPS Alert: Lost connection to the
gps, the app won't be able to provide location services.",
Toast.LENGTH_LONG).show();
        if(currentUser != null)
            UserHelper.setOnLine(currentUser.getId(), false);
    } else {
        Toast.makeText(context, "GPS Alert: GPS and location
services are back on", Toast.LENGTH_SHORT).show();
        if(currentUser != null)
            UserHelper.setOnLine(currentUser.getId(), true);
    }
}
}
}

```

Gender

```

package com.example.thetrempiada.enums;

public enum Gender {
    MALE, FEMALE
}

```

UserType

```

package com.example.thetrempiada.enums;

public enum UserType {
    DRIVER, TREMPIST, BOTH
}

```

User

```

package com.example.thetrempiada.data;

import com.example.thetrempiada.data.directionData.LatLng;
import com.example.thetrempiada.enums.Gender;
import com.example.thetrempiada.enums.UserType;

/**
 * This class will hold all the data about the user.
 * An object of this class will be uploaded to Firebase Realtime Database.
 * Display Name and Profile Image aren't included here since those are
included in the default Firebase User Database
 */

public class User {

    // UID from Firebase Database

```

```

private String _uid;
private String _displayName;
// value -1 = no age entered
private int _age;
private Gender _gender;
private UserType _userType;
// empty string ("" ) = no description
private String _description;
// current user location
private LatLng _currentLocation;
// user connectivity status
private boolean _online;

// empty constructor
private User() {}

// all members constructor
public User(String uid, String displayName, int age, Gender gender,
UserType userType, String description) {
    this._uid = uid;
    this._displayName = displayName;
    this._age = age;
    this._gender = gender;
    this._userType = userType;
    this._description = description;
    this._currentLocation = null;
    this._online = false;
}

// no age constructor
public User(String uid, String displayName, Gender gender, UserType
userType, String description) {
    this._uid = uid;
    this._displayName = displayName;
    this._age = -1;
    this._gender = gender;
    this._userType = userType;
    this._description = description;
    this._currentLocation = null;
    this._online = false;
}

// no description constructor
public User(String uid, String displayName, int age, Gender gender,
UserType userType) {
    this._uid = uid;
    this._displayName = displayName;
    this._age = age;
    this._gender = gender;
    this._userType = userType;
    this._description = "";
    this._currentLocation = null;
    this._online = false;
}

// no description and age constructor
public User(String uid, String displayName, Gender gender, UserType
userType) {
    this._uid = uid;

```

```
        this._displayName = displayName;
        this._age = -1;
        this._gender = gender;
        this._userType = userType;
        this._description = "";
        this._currentLocation = null;
        this._online = false;
    }

    public String get_uid() {
        return _uid;
    }

    public void set_uid(String _uid) {
        this._uid = _uid;
    }

    public String get_displayName() {
        return _displayName;
    }

    public void set_displayName(String _displayName) {
        this._displayName = _displayName;
    }

    public int get_age() {
        return _age;
    }

    public void set_age(int _age) {
        this._age = _age;
    }

    public Gender get_gender() {
        return _gender;
    }

    public void set_gender(Gender _gender) {
        this._gender = _gender;
    }

    public UserType get_userType() {
        return _userType;
    }

    public void set_userType(UserType _userType) {
        this._userType = _userType;
    }

    public String get_description() {
        return _description;
    }

    public void set_description(String _description) {
        this._description = _description;
    }

    public void set_currentLocation(double latitude, double longitude) {
        this._currentLocation = new LatLng(latitude, longitude);
    }
}
```

```

    }

    public void set_currentLocation(LatLng latLng) {
        this._currentLocation = new LatLng(latLng.getLatitude(),
latLng.getLongitude());
    }

    public LatLng get_currentLocation() {
        return this._currentLocation;
    }

    public boolean is_online() {
        return _online;
    }

    public void set_online(boolean _online) {
        this._online = _online;
    }
}

```

UserHelper

```

package com.example.thetrempiada.data;

import android.content.Context;
import android.util.Log;
import android.widget.Toast;

import androidx.annotation.NonNull;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

// this class is used to handle some of the operations in using users

public class UserHelper {

    public static final DatabaseReference _userDatabase =
FirebaseDatabase.getInstance("https://thetrempiada-5e41a-default-
rtdb.europe-west1.firebaseio.com").getReference().child("users");

    // adds the user to the database

    public static void addUserToDatabase(User user, String uid, Context
context) {
        _userDatabase.child(uid).setValue(user).addOnCompleteListener(new

```

```

OnCompleteListener<Void>() {
    @Override
    public void onComplete(@NonNull Task<Void> task) {
        if (task.isSuccessful()) {
            Toast.makeText(context, "Successfully signed up!",
Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(context, "An error has occurred.",
Toast.LENGTH_SHORT).show();
        }
    }
});
}

// deletes the user from the database

public static void deleteUserFromDatabase(String uid, Context context)
{
    _userDatabase.child(uid).removeValue().addOnCompleteListener(new
OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            if (task.isSuccessful()) {
                Toast.makeText(context, "Successfully deleted user
(UUID: " + uid + ")", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(context, "An error has occurred.",
Toast.LENGTH_SHORT).show();
            }
        }
    });
}

// sets user state to online or offline
// also adds/removes user from online users database

public static void setOnline(String uid, boolean isOnline) {
    _userDatabase.child(uid).child("_online").setValue(isOnline).addOnCompleterL
istener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            Log.d("UserChangedState: ",
String.valueOf(task.isSuccessful()));
            Log.d("UserState: ", String.valueOf(isOnline));
        }
    });
}
}

```

StorageHelper

```
package com.example.thetrempiada.data.directionData;

import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;

/**
 * this class will handle anything connected with Firebase Storage
 * pretty empty for now.
 */

public class StorageHelper {

    public static final StorageReference _storageReference =
        FirebaseStorage.getInstance("gs://thetrempiada-
5e41a.appspot.com/").getReference();

}
```

PathStorageHelper

```
package com.example.thetrempiada.data;

import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

public class PathStorageHelper {

    public static final DatabaseReference _pathDatabase =
        FirebaseDatabase.getInstance("https://thetrempiada-5e41a-default-
rtbd.eu-west-1.firebaseio.com/").getReference().child("paths");

}
```

LatLng

```
package com.example.thetrempiada.data.directionData;

public class LatLng {

    // helper class that defines my own LatLng, since google's doesnt have
    // an empty constructor which firebase needs in order to restore data from the
    // database.

    private double latitude;
    private double longitude;

    public LatLng(double latitude, double longitude) {
        this.latitude = latitude;
        this.longitude = longitude;
    }

}
```

```
public LatLng() {}

public double getLatitude() {
    return latitude;
}

public void setLatitude(double latitude) {
    this.latitude = latitude;
}

public double getLongitude() {
    return longitude;
}

public void setLongitude(double longitude) {
    this.longitude = longitude;
}
}
```