**Doggie Application**

****

Project By:

Noa Binenbaum 305201998

Alex Dashevski 314358425

Guided By:

Roman Kaplan

תוכן עניינים

[תקציר 3](#_Toc6333362)

[מטרת הפרויקט 3](#_Toc6333363)

[מימוש 4](#_Toc6333364)

[רישום ואימות משתמשים 4](#_Toc6333365)

[אתחול שירותי מיקום 8](#_Toc6333366)

[הצגת פרטי המשתמש 9](#_Toc6333367)

[הוספת כלב חדש 11](#_Toc6333368)

[הצגת כל הכלבים הרשומים במערכת 13](#_Toc6333369)

[עריכת פרופיל המשתמש/ כלב 15](#_Toc6333370)

[שימוש ב-Firebase Storage 16](#_Toc6333371)

[חישוב גיל הכלב מתוך תאריך לידה 18](#_Toc6333372)

[תצוגת מפה 19](#_Toc6333373)

[צ'ט עם משתמש נוסף 20](#_Toc6333374)

[הצעות לעתיד 21](#_Toc6333375)

# תקציר

הפרויקט בוצע במסגרת המעבדה לתוכנה ורשתות בפקולטה להנדסת חשמל בטכניון.

הפרויקט עסק בבנייה ועיצוב של רשת חברתית למכשירי אנדרואיד (אפליקציה).

האפליקציה כוללת חלקים רבים אולם העיקריים הם:

* מערכת התחברות (רישום וזיהוי המשתמש) המאפשרת גישה לתכן האפליקציה
* שימוש בשרת לצורך אחסון נתוני כל המשתמשים ושליפתו בזמן אמת
* עדכונים אוטומטיים של קואורדינטות GPS לצורך חישוב מרחקי כל המשתמשים זה מזה תוך כדי תנועה
* הצגת מיקום המשתמשים על מפה

האפליקציה פותחה בשפות JAVA ו-XML, בסביבת הפיתוח Android Studio בעזרת Android-SDK בגרסה 28.

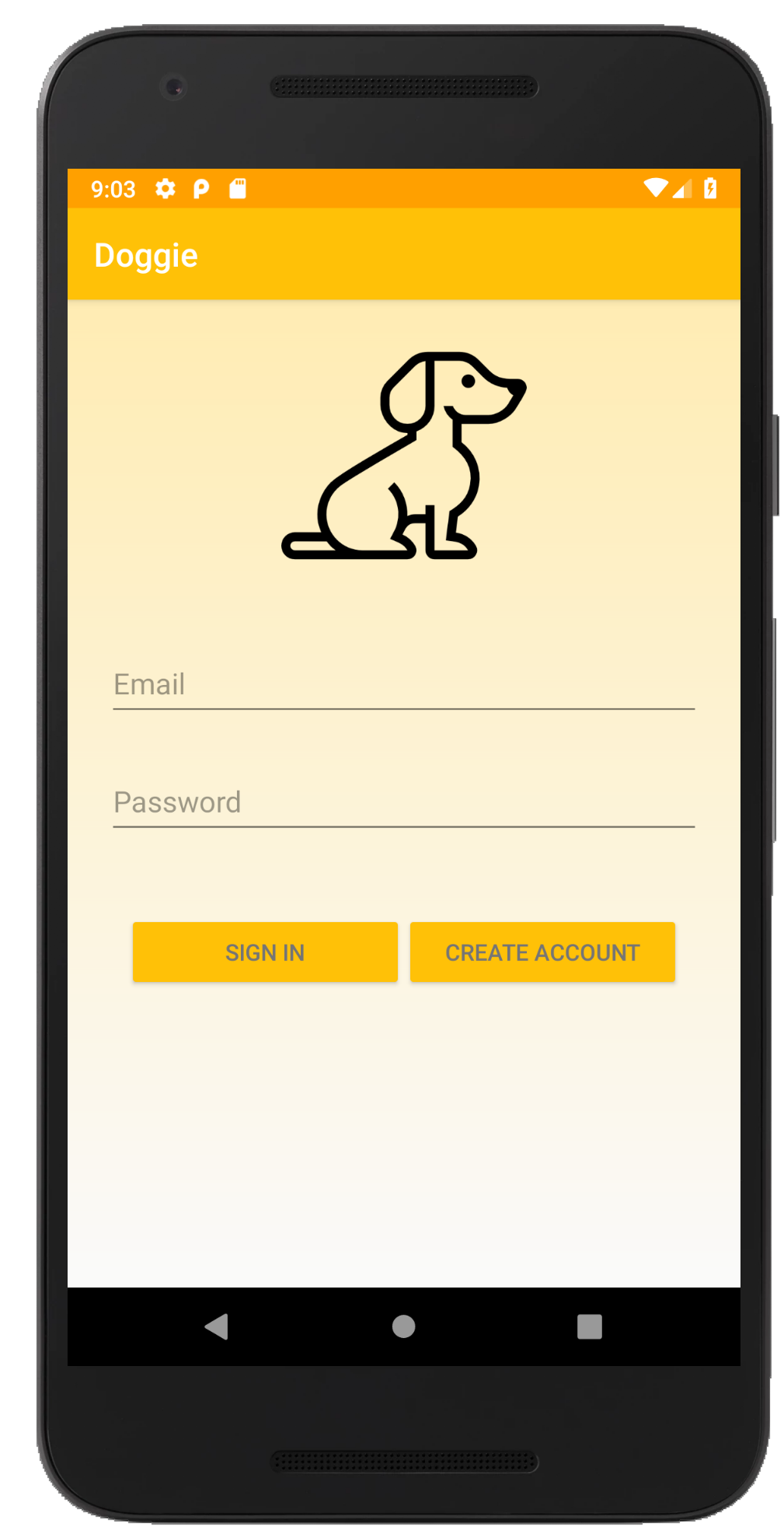
# מטרת הפרויקט

בתור בעלי כלבים אנו ערים לצרכים של כלבינו לפרוק אנרגיה ולשחק עם כלבים אחרים, מה שמכונה סוציאליזציה. אך לצערנו לא בכל מקום קיימת גינת כלבים מסודרת בה ניתן לשחרר את הכלבים לרוץ ולשחק, ובנוסף, אין לנו מושג מתי בעלי כלבים אחרים יוצאים לטייל עם כלביהם כך שנוכל לפגוש אותם, והכל נותר ליד המקרה. מטרת האפליקציה היא ליצור סוג של רשת חברתית בין בעלי הכלבים שמציגה את מיקום בעלי הכלבים המעוניינים בכך על מפה, לתקשר איתם דרך צ'ט ובכך להקל על יצירת מפגש בין הכלבים, דבר שכאמור טוב לבריאותם וטוב לרוגע שלהם בבית.

# מימוש

## רישום ואימות משתמשים

בשלב הראשון של השימוש באפליקציה, על המשתמש להזדהות מול המערכת על מנת שנוכל לאמת את זהותו, לטעון את המידע ששמור ב-database לגביו ולהתריע בפני יתר משתמשי המערכת לגבי מצבו של המשתמש. המסך שמוצג למשתמש הינו:



לאחר שהמשתמש ממלא את פרטי הטופס (כתובת דוא"ל וסיסמא) עליו לבחור בין שתי האפשרויות:

1. SIGN IN – התחברות משתמש קיים. אפשרות זאת מפעילה את הפונקציה signIn שראשית בודקת את תקינות מילוי הטופס (לא קיימים שדות ריקים או תווים אסורים) ובמידה ולא מתעוררת בעיה, מתבצעת התחברות למערכת האימות של firebase (Firebase Authentication System) אשר מנהלת את המשתמשים הרשומים למערכת ומבצעת את עבודתה "מאחורי הקלעים". המערכת בודקת האם קיים המשתמש במאגר המשתמשים הרשומים, האם הסיסמה שלו תואמת את הסיסמה המוצפנת ששמורה עבורו במערכת והאם כתובת הדוא"ל של המשתמש אומתה על ידו (לאחר שנשלח אליו לינק אימות לכתובת שסיפק בעת ההתחברות הראשונית). בהתאם לכך מתאפשר (או לא מתאפשר) מעבר המשתמש למסך הבא.
2. CREATE ACCOUNT – התחברות ראשונית של משתמש. אפשרות זאת מפעילה את הפונקציה createAccount שבאותו אופן בודקת את תקינות מילוי פרטי הטופס ע"י המשתמש. לאחר מכן האחריות לביצוע הוספת המשתמש למאגר המשתמשים עובר למערכת האימות של Firebase. באמצעות הקריאה פשוטה:

FirebaseAuth.getInstance().createUserWithEmailAndPassword(**email**, **password**)

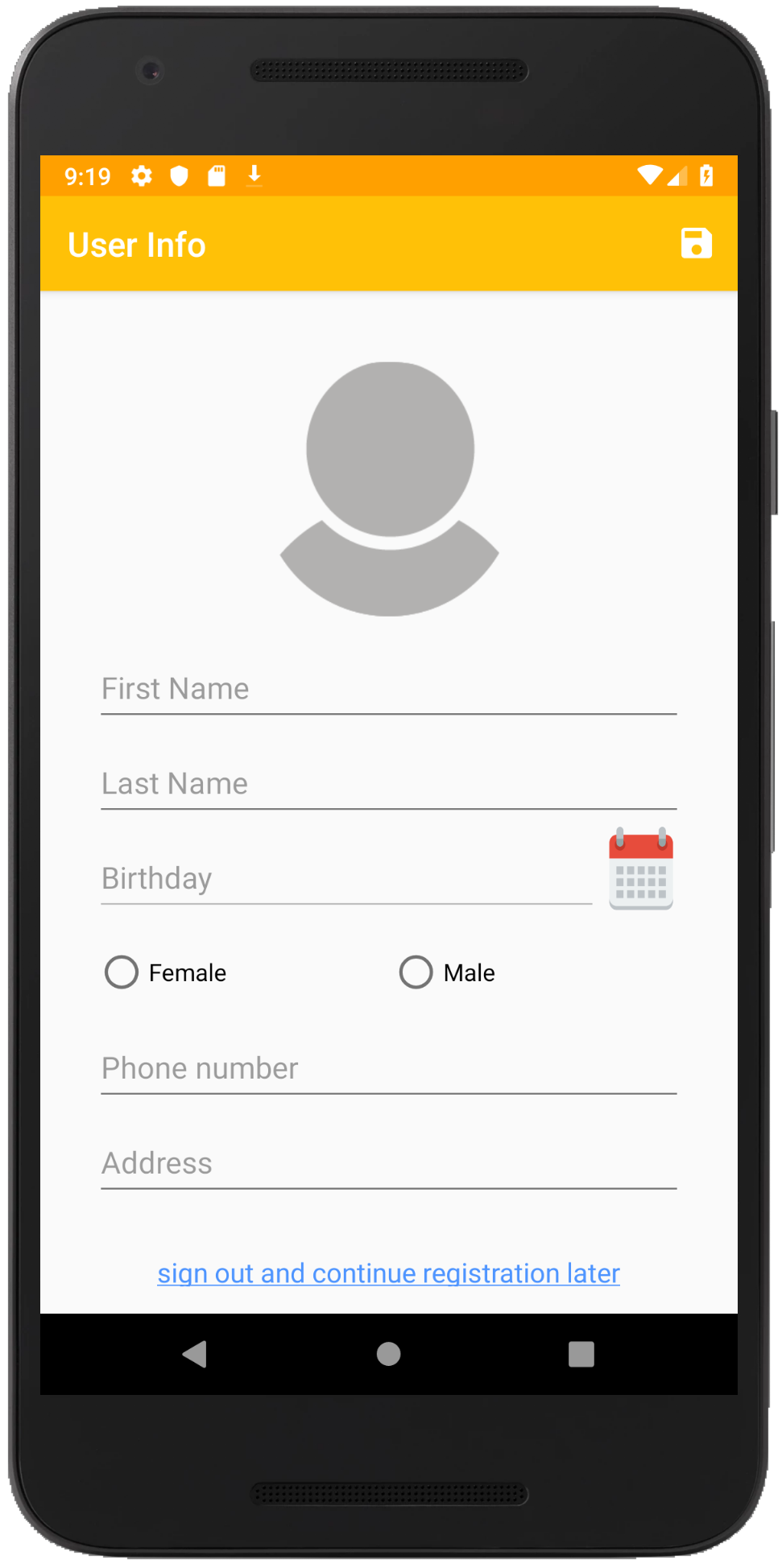
מתבצעות סדרת הפעולות הבאות:

* ווידוא כי כתובת הדוא"ל אינה קיימת במאגר המשתמשים של האפליקציה.
* ווידוא תקינות כתובת הדוא"ל והסיסמה שהוקלדו (התו '@' מופיע בכתובת הדוא"ל, אורך הסיסמא הינו לפחות 6 תווים ועוד).
* הוספת המשתמש למאגר המשתמשים הרשומים במערכת והצפנת סיסמתו שנשמרת בשרת של Firebase (המערכת לא שומרת בשום מקום סיסמאות משתמשים כפי שהן מטעמי אבטחה. ניתן בהגדרת הפרויקט לבחור מפתח ערבול או קוד טביעת אצבע על מנת להגביר את אבטחת המידע).
* המשתמש ניתן לזיהוי במערכת האימות באמצעות מפתח חח"ע (User UID) המאפשר את שליפתו ממאגר המשתמשים והתייחסות למידע השמור לגביו בשרת.



לאחר שהאימות עובר בהצלחה, כאשר המשתמש מבצע התחברות מתבצעת בדיקה ב-Database של Firebase, האם קיים מידע שמור אודות המשתמש הנוכחי (שמזוהה כאמור בעזרת מפתח אישי). הבדיקה הנ"ל, בניגוד לבדיקה שמבוצעת ע"י מערכת האימות של Firebase, אינה הכרחית לצורך זיהוי ואימות המשתמש, אך היא נועדה לצורכי מימוש הפונקציונליות של האפליקציה, כך שנוכל בכל עת לבצע פעולות שונות על המשתמש המחובר כמו שמירת מיקומו הגאוגרפי, שמירת פרטי הכלבים שהוא מוסיף לאחר התחברותו ועוד.

אם המשתמש אינו מופיע ב-database, מבצעים מעבר של מסכי האפליקציה (Activity חדש) לקליטת פרטים נוספים אודות המשתמש:



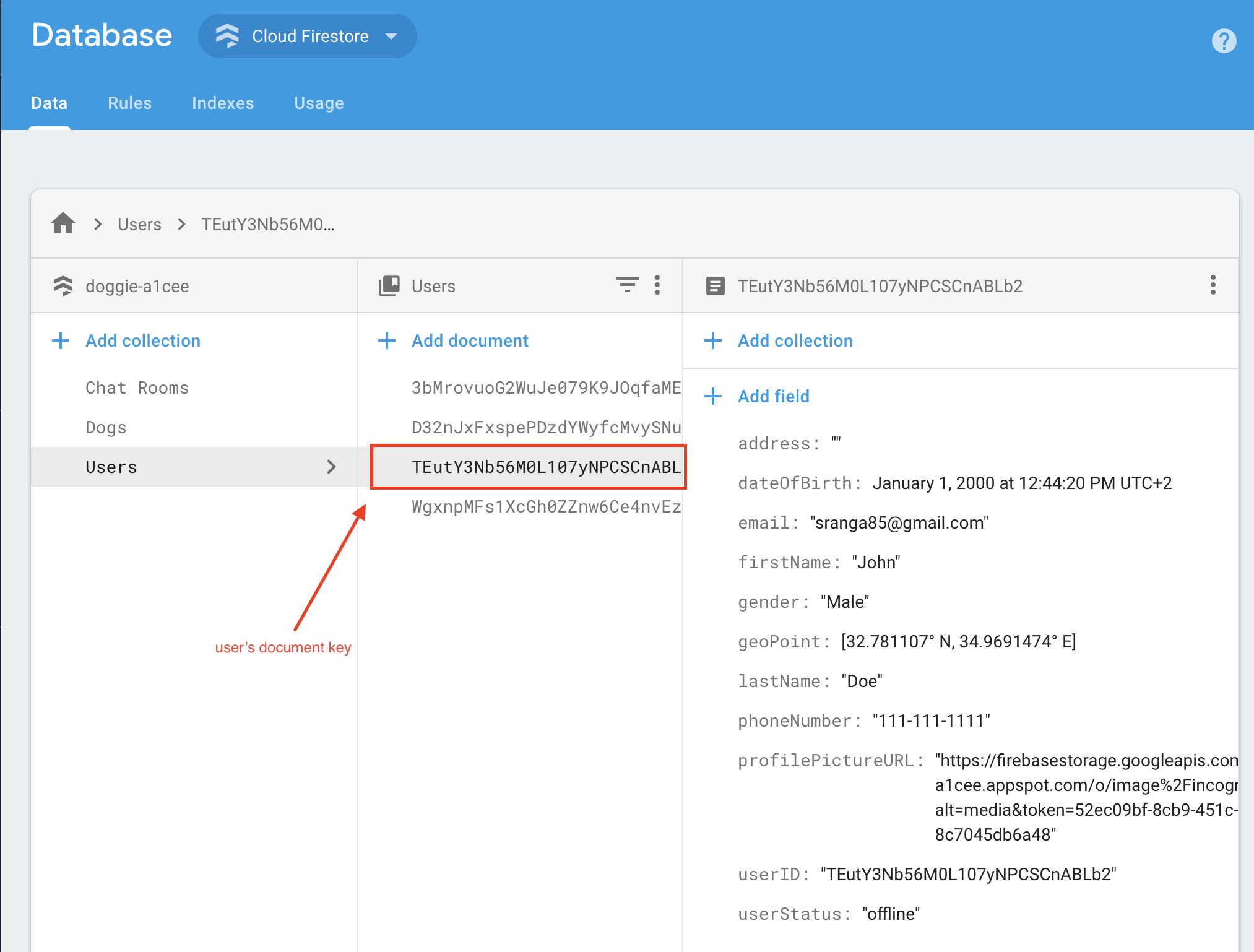
תמונה, שם פרטי ושם משפחה הינם שדות חובה לצורך זיהוי המשתמש בפני שאר המשתמשים והתייחסות אליו במסכים השונים.

שדות נוספים שהינם אופציונליים: תאריך לידה, זכר/ נקבה, מספר טלפון וכתובת. בשלב זה המשתמש יכול לבחור שלעצור ולהמשיך בשלב מאוחר יותר ע"י לחיצה על הקישור שבתחתית המסך. אפשרות זאת תבצע ניתוק שלו מהמערכת ואמנם לא תשמור את פרטי הטופס הנ"ל שבחר שלא לסיים, אך בפעם הבאה שיתחבר לאפליקציה, מערכת האימות של Firebase תזהה אותו ותדע לשלוף את המפתח האישי שלו כך שימשיך במילוי פרטיו האישיים בדיוק באותה נקודה בה נעצר.

תמונת המשתמש מאוחסנת ב-storage של Firebase (פירוט בהמשך).

לאחר סיום מילוי הטופס ווידוא תקינות השדות באמצעות קריאה לפונקציה validateForm, נוצר אובייקט חדש מסוג User שמכיל לפחות את שמו הפרטי, שם משפחתו וכתובת URL שבה מוחזקת תמונת הפרופיל של המשתמש בשרת. לאחר מכן יוצרים DocumentReference, אובייקט שמכיל פרטי מידע בפורמט ששרת ה-Firebase מזהה כך שניתן לבצע עליו שאילתות לצורך שליפת מידע, לערוך ולמחוק אותו. המידע אודות המשתמש מאוחסן ב-database תחת מפתח ייחודי (שזהה למפתח האישי שניתן לו ע"י מערכת האימות של Firebase). למשתמש קיים סטטוס המתאר את מצב התחברותו למערכת (online/ offline). בעת יצירת משתמש חדש נקבע הסטטוס שלו לערך ברירת מחדל של online.

בכל פעם שהמערכת מאמתת בהצלחה משתמש קיים, הסטטוס שלו, כפי ששמור ב-database, מתעדכן ל-online ובכל פעם שמבוצע sign out מהאפליקציה, הסטטוס של המשתמש משתנה ל-offline. מכיוון שמשתמש יכול לסגור את האפליקציה באופן לא צפוי (ביצוע swipe של האפליקציה מרשימת האפליקציות הפתוחות) ואין לנו אפשרות להתמודד עם מצב זה ולהתכונן אליו- במקרה של יציאה לא מסודרת, ללא ביצוע sign out, הסטטוס של המשתמש נשאר online ומיקומו ייחשף ליתר המשתמשים.



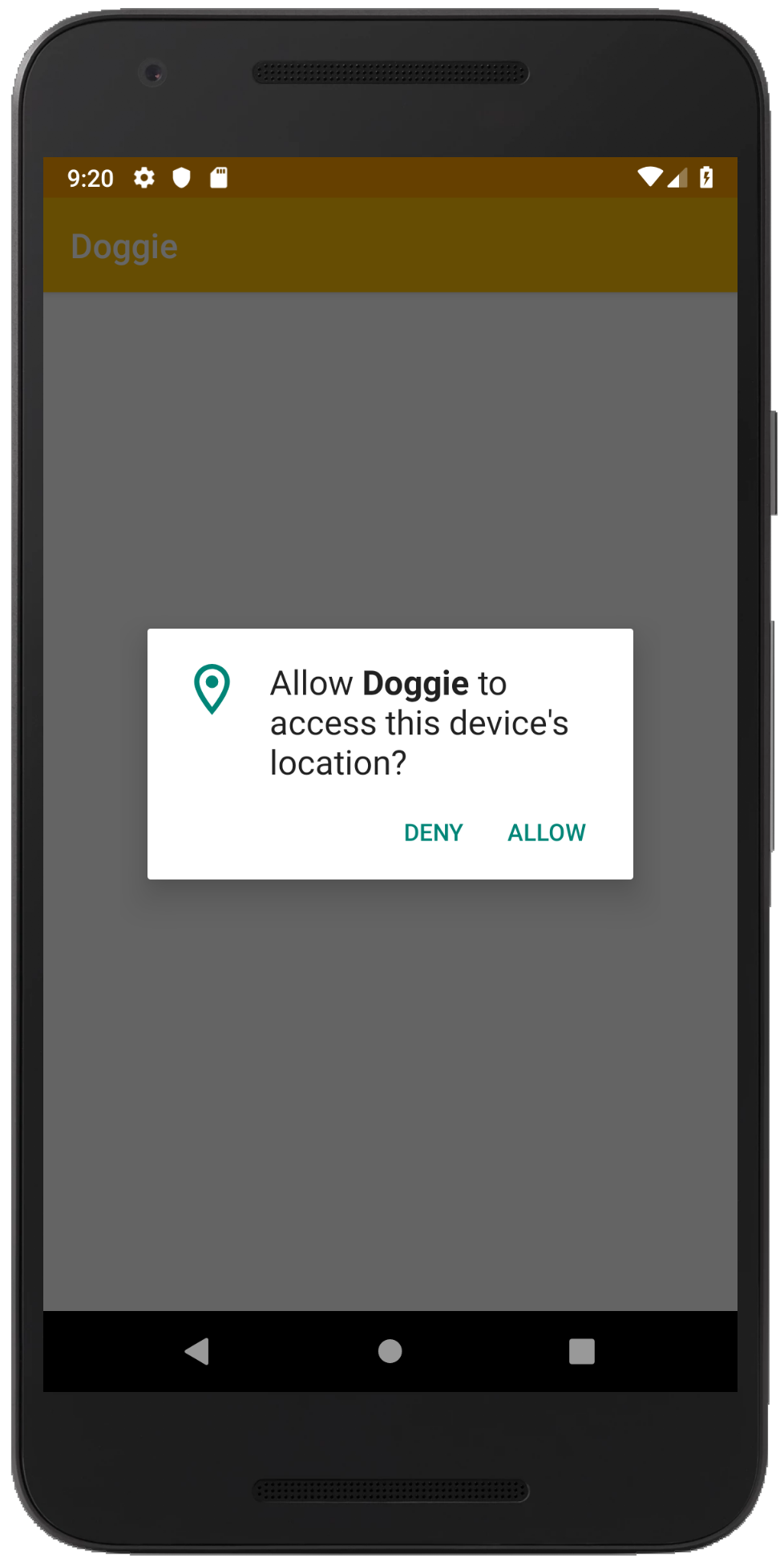
לאחר סיום הוספת המשתמש ל-database בהצלחה (או במקרה של התחברות חוזרת למערכת) יש לבצע אתחול והגדרת שירותי מיקום ולכן נוצר Intent חדש שמבצע זאת באופן חד פעמי לאחר התחברות המשתמש.

## אתחול שירותי מיקום

בעזרת שירותי המיקום של Google (Google location services), מיקומו הגאוגרפי של מכשיר האנדרואיד אליו מחובר המשתמש נשלח לשרת ומבצע עדכונים בכל פרק זמן מוגדר במקרה של שינוי מיקומו של המכשיר. זיהוי מיקומו של כל משתמש מאפשר להציג אותו על מפה ליתר המשתמשים המחוברים ולחשב את מרחקו מהם.

השלבים שמתבצעים לאחר התחברות המשתמש לצורך אתחול שירותי המיקום:

1. ווידוא כי שירותיGoogle play מותקנים על מכשיר האנדרואיד של המשתמש (ללא שירותי Google play לא ניתן להשתמש במפה).
2. ווידוא כי שירותי GPS מאופשרים במכשיר לצורך שימוש באפליקציה שלנו.
3. קבלת הרשאות שירותי מיקום באופן מפורש מהמשתמש.



לאחר קבלת ההרשאות הדרושות וביצוע כל הבדיקות הנ"ל, יוצרים תהליך שרץ ברקע בלולאה (Looper) עד שה-thread בו הוא מוגדר נהרס. הלופר מבצע קריאה של מיקום המשתמש מהמכשיר וכתיבתו ל-database בכל 30 שניות. המשתמש מועבר ל-Activity הבא שמציג את מסך הבא שלו (UserHomeProfile).

## הצגת פרטי המשתמש

במסך הבית של המשתמש מוצגים פרטיו ורשימת הכלבים שלו שהוסיף למערכת. בשלב הראשוני (ב-onCreate) מבצעים גישה ל-database באמצעות פניה עם ה-path המתאים שבו מוחזק ה-DocumentReference של המשתמש הנוכחי שמחובר למערכת (ומזוהה ע"י מערכת האימות של Firebase). שליפת המידע מתבצעת באופן אסינכרוני ולכן אנו מתחזקים EventListener שמתחיל להאזין לרפרנס של המשתמש ב-database ומסיים את האזנתו ברגע שה-activity נעצר (ב-onStop). ה-EventListener הוא ממשק שמגדיר פונקציה אחת בלבד, onEvent, שנקראת בכל פעם שמתבצע שינוי ב-Document שאליו ה-listener הוצמד. לכן, בעת הצמדת ה-listener לרפרנס, המידע שב-database טרם נקרא ולכן הרפרנס הוא null. לאחר שמתבצעת קריאה מה-database לתוך הרפרנס:

**private** DocumentReference **mUserRef**;

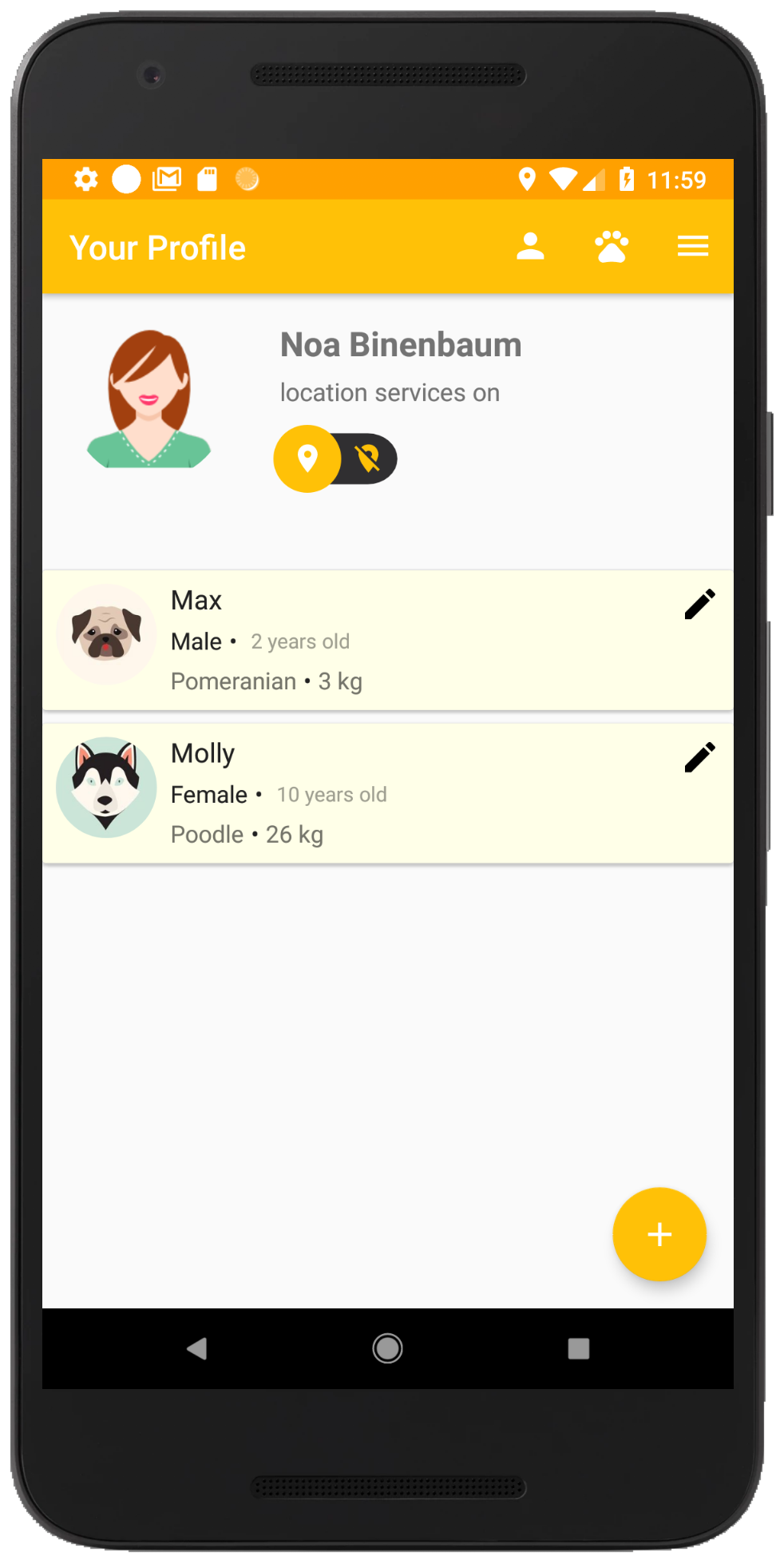
**private** FirebaseUser **mUser** = FirebaseAuth.*getInstance*().getCurrentUser();

**mUserRef** = FirebaseFirestore.*getInstance*().

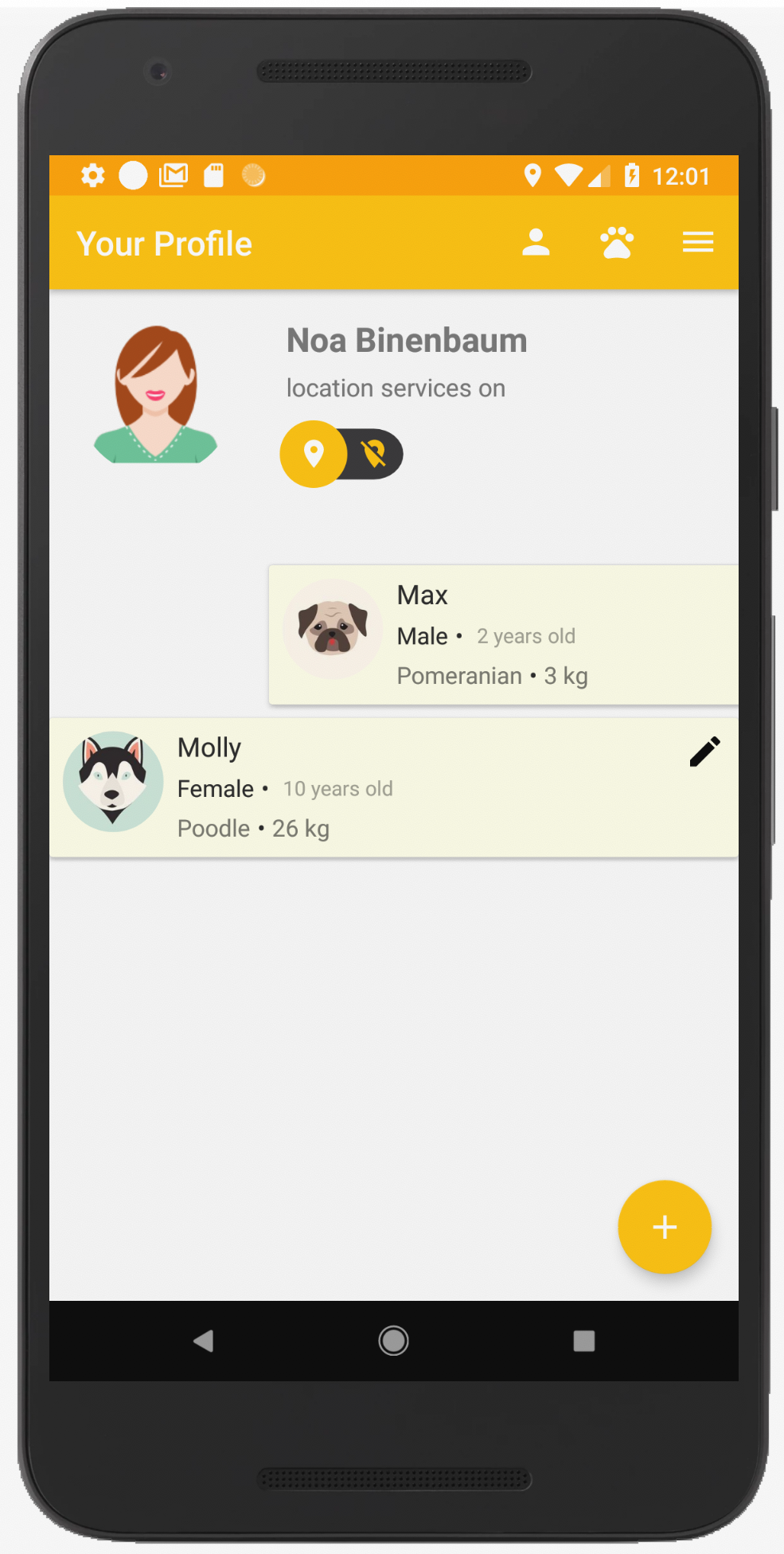
collection(**"Users"**).document(**mUser**.getUid());

הפונקציה onEvent נקראת. בשלב זה יש לטעון למסך את הפרטים המתארים את המשתמש- תמונת הפרופיל שלו, שמו המלא והסטטוס שלו. הסטטוס מוצג ע"י התיאור you are online\ you are offline ובנוסף מופיע toggle שמאפשר לשנות את הסטטוס הנוכחי. משתמש שאינו מעוניין לחשוף את מיקומו הנוכחי (או לצפות במיקומם של משתמשים אחרים) יכול לשנות את הסטטוס שלו ל-offline ע"י הזזת ה-toggle בהתאם. לרכיב ה-toggle שהינו מסוג IconSwitch, הוצמד listener שמאזין לכל תזוזה שלו. בהתאם למיקומו של ה-toggle, שמאל או ימין, מופעלת הפונקציה setStatus שתפקידה לבצע כתיבה ל-database ולשנות את ערכו של הסטטוס של המשתמש (שאת זהותו היא מקבלת כפרמטר).

במקביל לשלב טעינת פרטי המשתמש מה-database ב-onCreate, מבצעים אתחול של RecycleView שתפקידו להציג את רשימת כל הכלבים של המשתמש. ה-RecycleView ממומש בעזרת adapter שמציג כל אובייקט מרשימת הכלבים באופן שהוגדר. לשם כך, יש לבצע שאילתה שניגשת ל-database ומחזירה את כל הכלבים ששדה ה-ownerID שלהם שווה למפתח הזיהוי של המשתמש הנוכחי שאומת במערכת ה-Firebase. עבור כל כלב שמוחזר בשאילתה, מתבצעת קריאה של כל המידע הרלוונטי ששמור עבורו ב-databse והמידע מוצג על המסך. ה-adapter יורש מהמחלקה המופשטת FirestoreRecyclerAdapter ככה שבהינתן שאילתה שמופעלת על collection נתון ב-Firebase database, ההתממשקות בין הפונקציה הדורסת שמציגה אובייקט בודד על המסך לבין הרפרנס אליו בשרת, מתבצעת ומנוהלת ע"י Firebase. כלומר, בעת שינוי של המידע השמור ב-database כמו הסטטוס של בעלי הכלב, מחיקת כלב או הוספתו, Firebase מקבל על כך התראה (באמצעות listeners) ומפעיל את הפונקציה הנדרסת onBindViewHolder שמציגה את פרטי האובייקט עבורו היא נקראה למסך. באופן זה, כל שינוי שמתבצע ב-database לגבי המידע המופיע ב-recycler view, מוצג מיד על המסך למשתמש, ללא צורך בריענון ה-activity, או ביצוע כל פעולה מצדנו- המפתחים ☺.

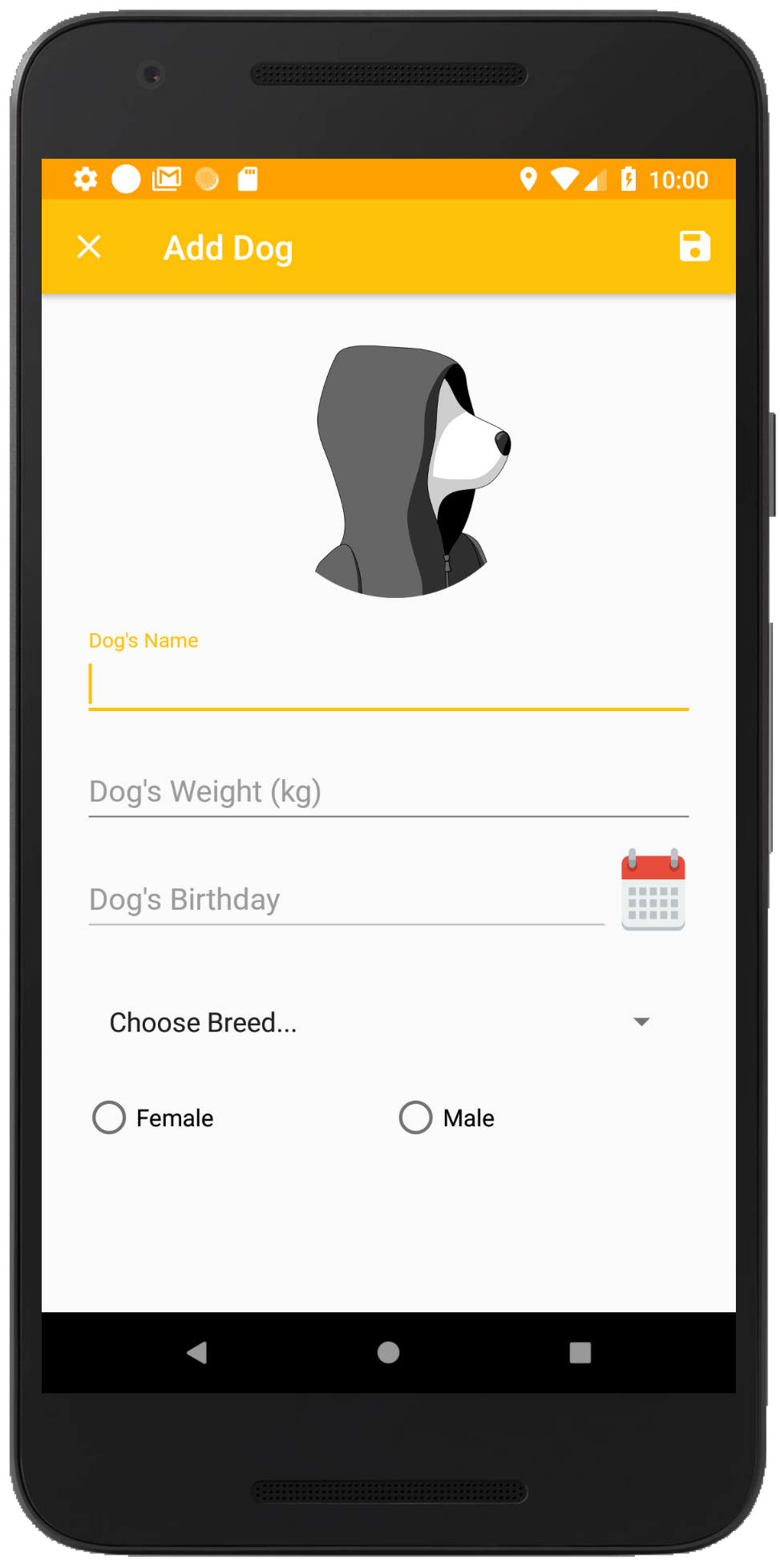


מסך הצגת פרטי המשתמש מציג כמה הפתעות נוספות:

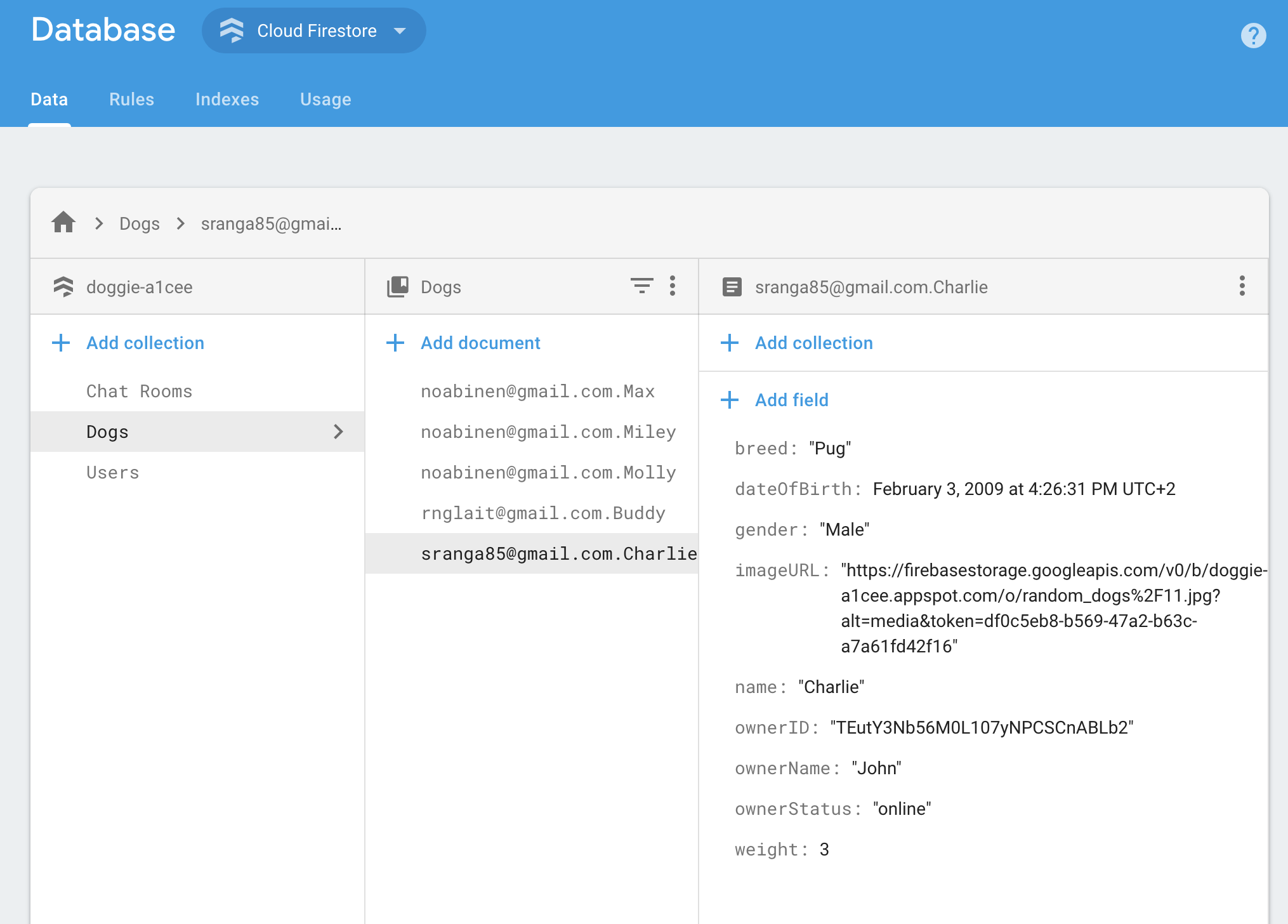
* ****אפשרות לערוך את פרטי הכלב שלו ע"י לחיצה על הכפתור המתאים עבור הכלב הרלוונטי. תוספת זו כוללת click listener שמגיב ללחיצה על ה-view המתאים וטוען את דף פרטי הכלב עליו התבצעה הלחיצה לצורך עריכת המידע המופיע בו.
* אפשרות למחוק כלב מרשימת הכלבים של המשתמש באמצעות swipe (החלקת האצבע ימינה ושמאלה על כרטיס הכלב.
* אפשרות להוסיף כלבים נוספים תחת אותו משתמש מתוך לחיצה על כפתור ההוספה המרחף (FloatingActionButton).

## הוספת כלב חדש

האפשרות ליצור כלב חדש זמינה הן מהתפריט הראשי והן ממסך הבית של המשתמש. על המשתמש להזין את פרטי הכלב שמתארים את חיית המחמד שלו: שמו, תמונתו, המשקל שלו, תאריך הלידה שלו, הגזע שלו ומינו (זכר/ נקבה). כל השדות בטופס הם חובה, פרט לתמונה, על מנת לאפשר למשתמשים אחרים לסנן מבין כלל הכלבים הרשומים במערכת את הכלבים המתאימים להם ביותר לצורך מפגש עם הכלב שלהם (למשל לא נרצה להפגיש בין כלב מבוגר עם כלב צעיר ופעלתן). לאחר שהמשתמש מזין את הפרטים ובוחר בכפתור השמירה, מתבצע אימות של הטופס הנ"ל כך שלא קיים שדה שהושאר ריק או שנכתב באופן שגוי. אם המשתמש לא בחר תמונת פרופיל לכלב שלו, תמונה אקראית מבין שלל תמונות השמורות בשרת תיבחר וכתובת ה-URL שמפנה ל-Firebase storage, שם מופיעה התמונה הנבחרת, תישמר כשדה המאפיין את אותו כלב. המשתמש מצהיר על תאריך הלידה של הכלב (ולא על גילו) על מנת שנוכל לחשב את גילו באופן דינמי בכל פעם שלמשתמש מוצגים פרטי הכלב.

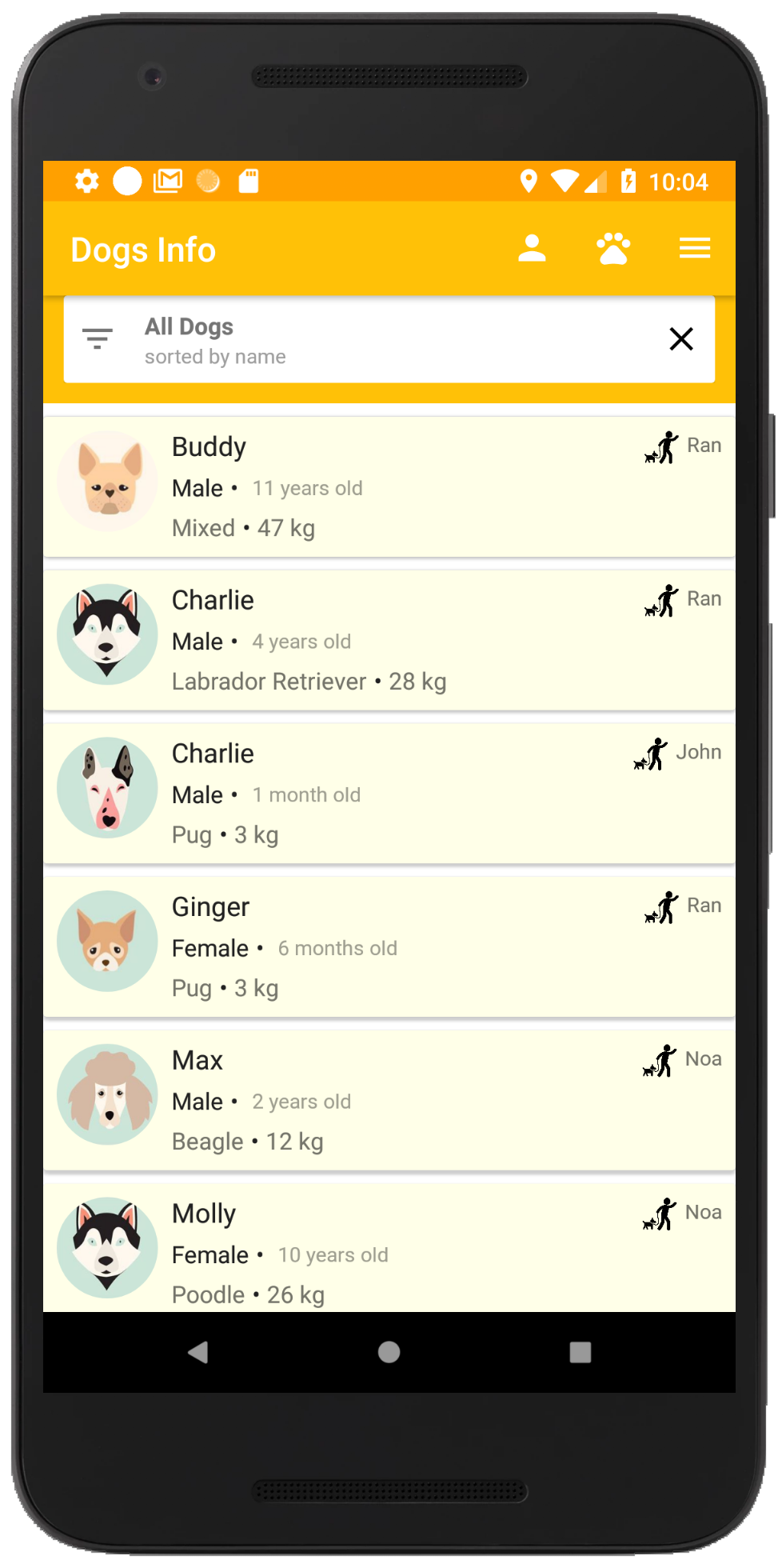


לאחר אימות תקין של שדות הטופס, נוצר אובייקט מסוג Dog המאותחל ע"י כל הפרטים שהקליד המשתמש ומתבצעת כתיבה ל-database לתוך מיכל מסוג Dogs, כאשר המזהה חד ערכית של פרטי הכלב הוא צירוף כתובת הדוא"ל של המשתמש (מערכת האימות של Firebase מונעת משני משתמשים שונים להתקיים עם אותה כתובת דוא"ל) עם שם הכלב. ההנחה היא כמובן שלא קיימים שני כלבים שונים עבור אותו משתמש עם אותו שם. במקרה זה של ניסיון יצירת כלב חדש עם שם זהה לכלב שרשום תחת אותו משתמש, תתבצע דריסה של המידע ב-database והפרטים החדשים המאפיינים את הכלב שנוצר שאותם הזין המשתמש יחליפו את הישנים ששמורים במערכת.



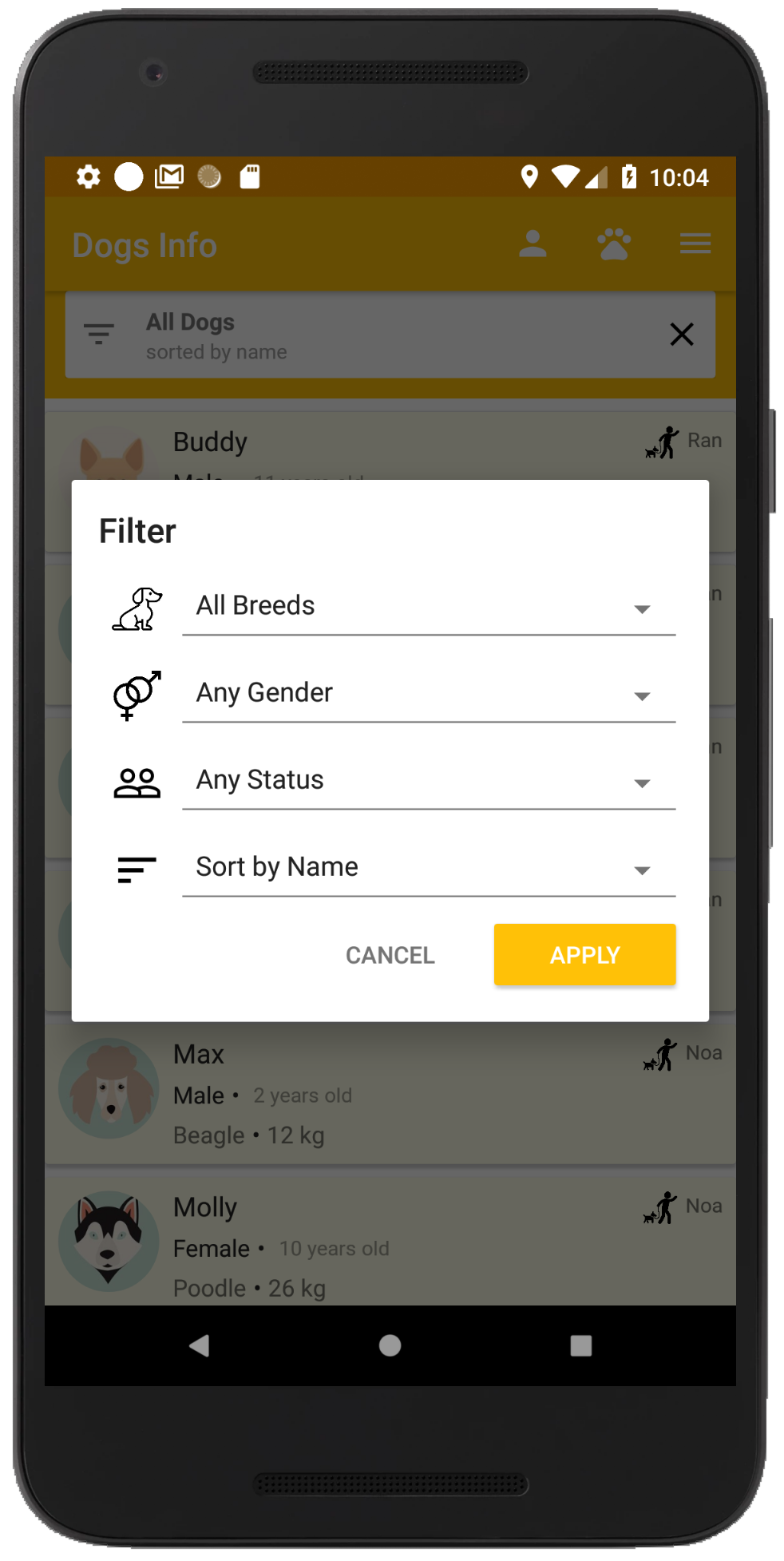
## הצגת כל הכלבים הרשומים במערכת

מתוך התפריט הראשי, המשתמש יכול לצפות בכל הכלבים הרשומים במערכת ולסנן אותם לפי בחירתו. כמו במסך עמוד הבית, גם כאן אופן התממשקות תצוגת המסך עם ה-database נעשית בעזרת FirebaseRecyclerAdapter, אלא שהפעם השאילתה שמופעלת על מיכל הכלבים בשרת, משתנה בהתאם לבחירת הסינון של המשתמש. במצב ברירת המחדל, מוצגים כל הכלבים במערכת, ממוינים לפי שמם. כל כלב מופיע עם תמונתו, שמו, גילו, גזעו, משקלו ובעליו.



משתמש שמעוניין לצפות בכלבים שלפי דעתו מתאימים למפגש עם הכלב שלו בהתאם לקריטריונים לסינון המוצגים יוכל להפעיל את הסינון הבא:

* הצגת כל הגזעים האפשריים או הצגת גזע מסוים בלבד.
* הצגת כלבים זכרים בלבד/ נקבות בלבד או לא משנה (לרוב כלבים זכרים ודומיננטיים לא מסתדרים עם כלבים זכרים אחרים ולכן חשוב למנוע התקוטטויות).
* הצגת הכלבים שהבעלים שלהם מחוברים למערכת (עם סטטוס online), מנותקים מהמערכת (סטטוס offline), או לא משנה.
* מיון הכלבים המוצגים לפי שם, גיל או גודל (משקל).



בכל פעם שמשתמש מפעיל את אפשרות הסינון, מופעלת שאילתה חדשה שמתאימה לאפשרויות הסינון של המשתמש ונשלחת ל-adapter לצורך שינוי התצוגה.

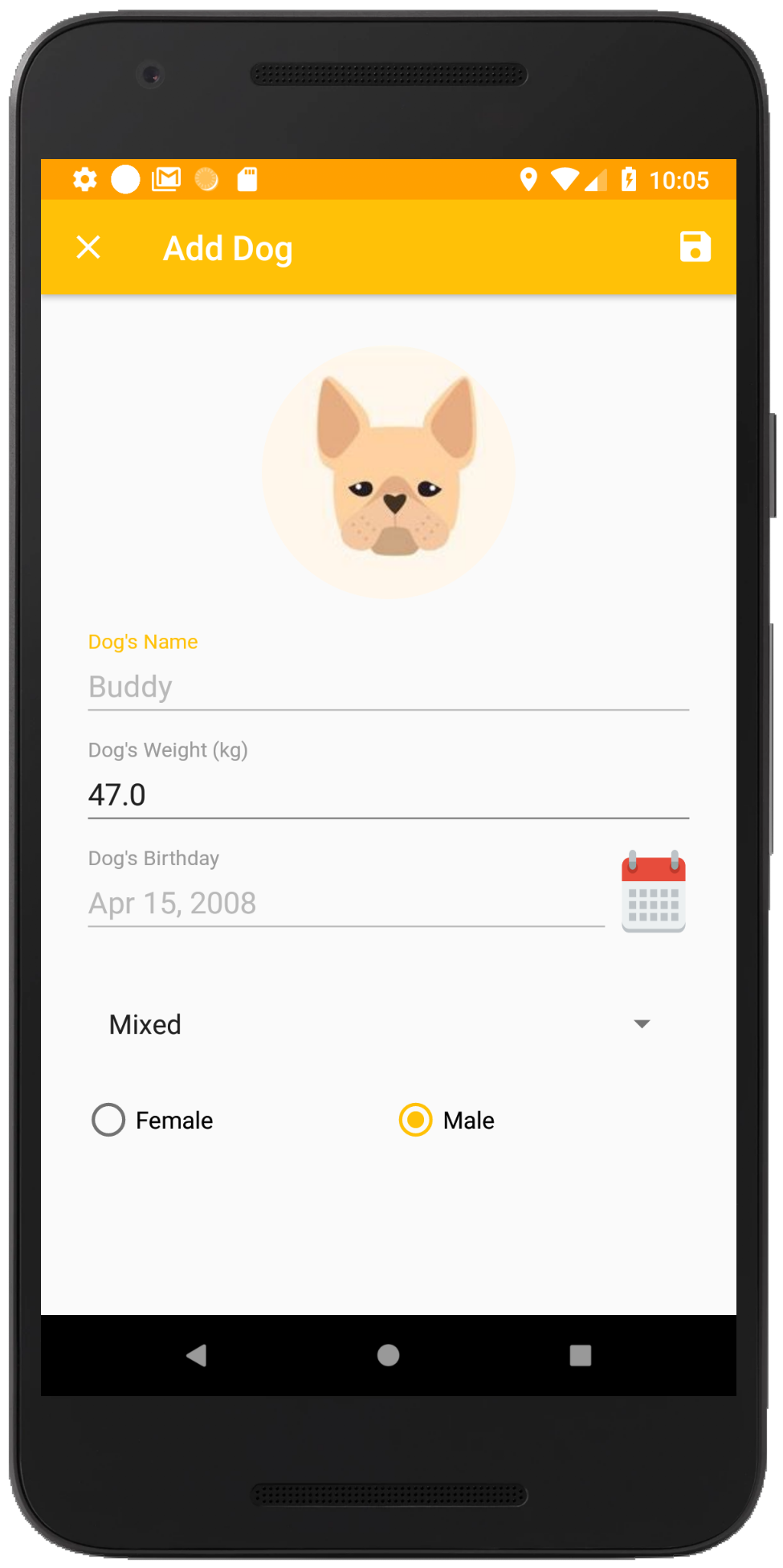
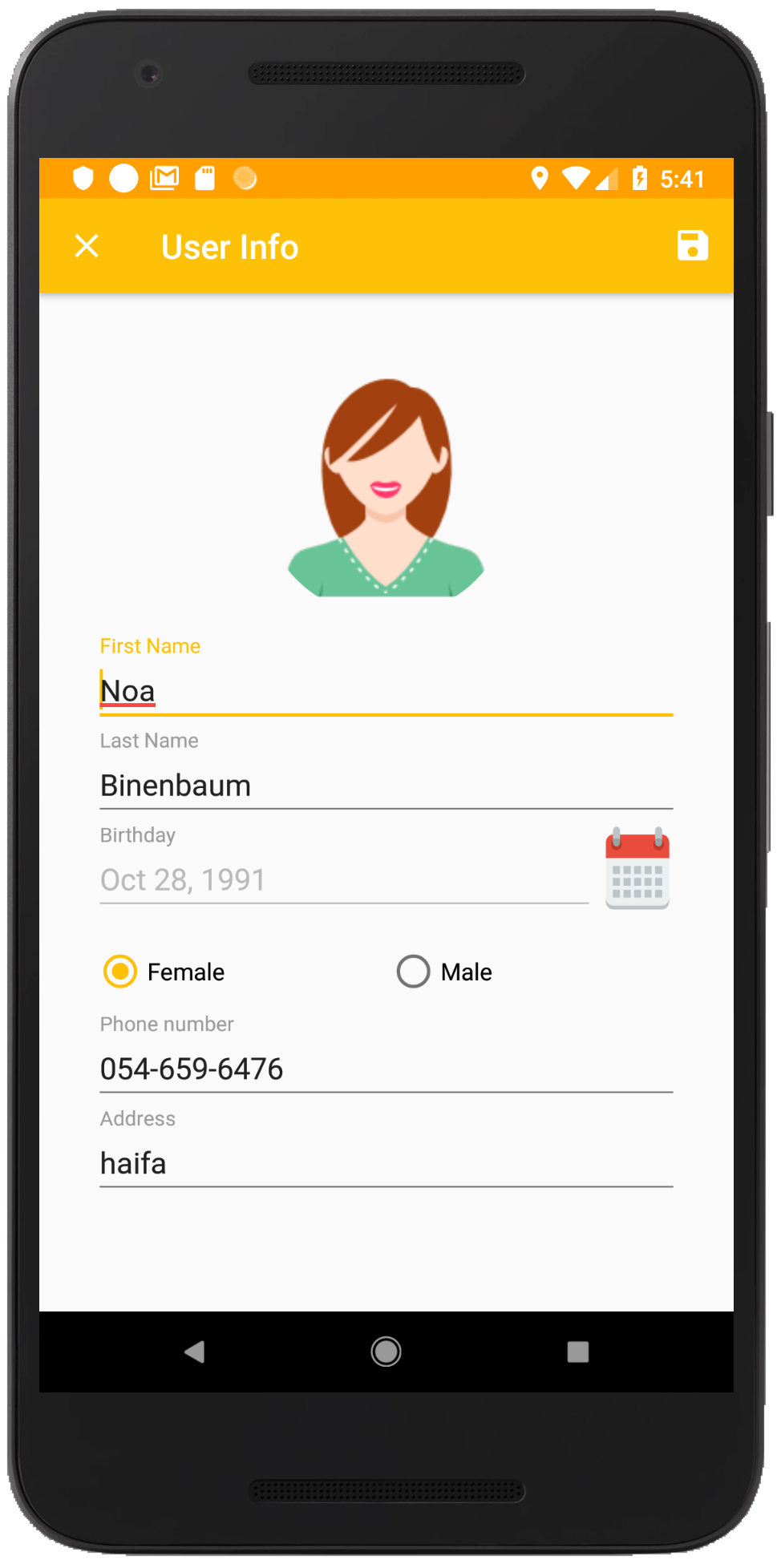
מתוך מסך זה, קיים click listener שמאזין ללחיצה על כל אחד מהכלבים המוצגים ב-adapter ומאפשר בפתיחת מסך ההתכתבות עם הבעלים של הכלב הנבחר למטרות תיאום מפגש בין הכלבים.

## עריכת פרופיל המשתמש/ כלב

מתוך התפריט הראשי ובאמצעות לחיצה על כפתור העריכה של כלב המופיע ברשימת הכלבים של המשתמש הנוכחי, קיימת אפשרות לערוך את פרטי המשתמש/ הכלב ולבצע עדכון ב-database של השדות המתארים את אותו אובייקט. כיוון שלוגיקה עבור עריכת שני סוגי האובייקטים זהה, נתייחס למקרה של עריכת פרטי כלב בלבד:

לחיצה על כפתור העריכה של פרטי הכלב מפנה את המשתמש ל-activity שהוא למעשה אותו אחד שאחראי על יצירת משתמש חדש, אך כעת באמצעות putExtras מעבירים ל-intent החדש את המזהה של הכלב ב-database אותו מעוניינים לערוך. ב-activity של הוספת כלב חדש מתבצעת בדיקה האם הועבר מ-activity אחר מפתח זיהוי של הכלב. אם התשובה היא לא, אז אין כלל התייחסות חריגה וממשיכים באותו אופן כפי שתואר במקרה של יצירת כלב חדש. אם התשובה היא כן, ניגשים ל-database בעזרת מפתח הזיהוי שהתקבל, שולפים את נתוני הכלב המתאים משם ומצמידים לרפרנס של ה-document המתאים listener שמאזין וממתין לרגע שליפת התוכן הרלוונטי. כיוון שמאותו רגע של הצמדת ה-listener, בכל פעם שמתבצע שינוי במידע שמוחזק עבור אותו אובייקט ב-database תתבצע קריאה ל-onEvenet ואנו מעוניינים שטעינת פרטי הכלב יתבצעו פעם אחת בלבד, קיים הדגל shouldLoadDog שמשנה את ערכו לאחר ש-onEvent נקרא בפעם הראשונה עבור טעינת פרטי הכלב. ברגע שיש שינוי ברפרנס כך שמצביע ל-document המכיל את פרטי הכלב, מבצעים לו casting לטיפוס המתאים אליו הוא מתייחס (Dog) וטוענים את כל פרטי הכלב לתוך טופס הרישום של כלב חדש. כיוון שמשך הזמן הדרוש לקריאת המידע מה-database וכתיבתו למסך המוצג למשתמש הוא קצר ביותר, התהליך האסינכרוני של טעינת הדף מה-database לא מורגש לעין המשתמש והוא אינו נאלץ להמתין רבות בחוסר מעש. מאותו רגע ההתנהלות ממשיכה כאילו מדובר בכלב חדש שהמשתמש מעוניין ליצור וזה עתה הוקלדו פרטיו בטופס המוצג- לאחר שהמשתמש יערוך את הפרטים אותם הוא מעוניין לעדכן וילחץ על כפתור השמירה, תתבצע קריאה מחדש של כל פרטי הטופס המופיעים על המסך, בין אם הם השתנו ובין אם לאו, ותתבצע דריסה של נתוני הכלב ב-database עם המידע החדש, כנדרש.

השדות המאפיינים את הכלב זמינים כולם לעריכה פרט לשם הכלב שמשמש כמפתח לאחסון נתוניו ב-database ולכן שינויו יגרום לשמירת כלב חדש תחת אותו משתמש, בנוסף לכלב הקיים שאותו התבקשנו לערוך, ללא מחיקת הכפילות הנ"ל.



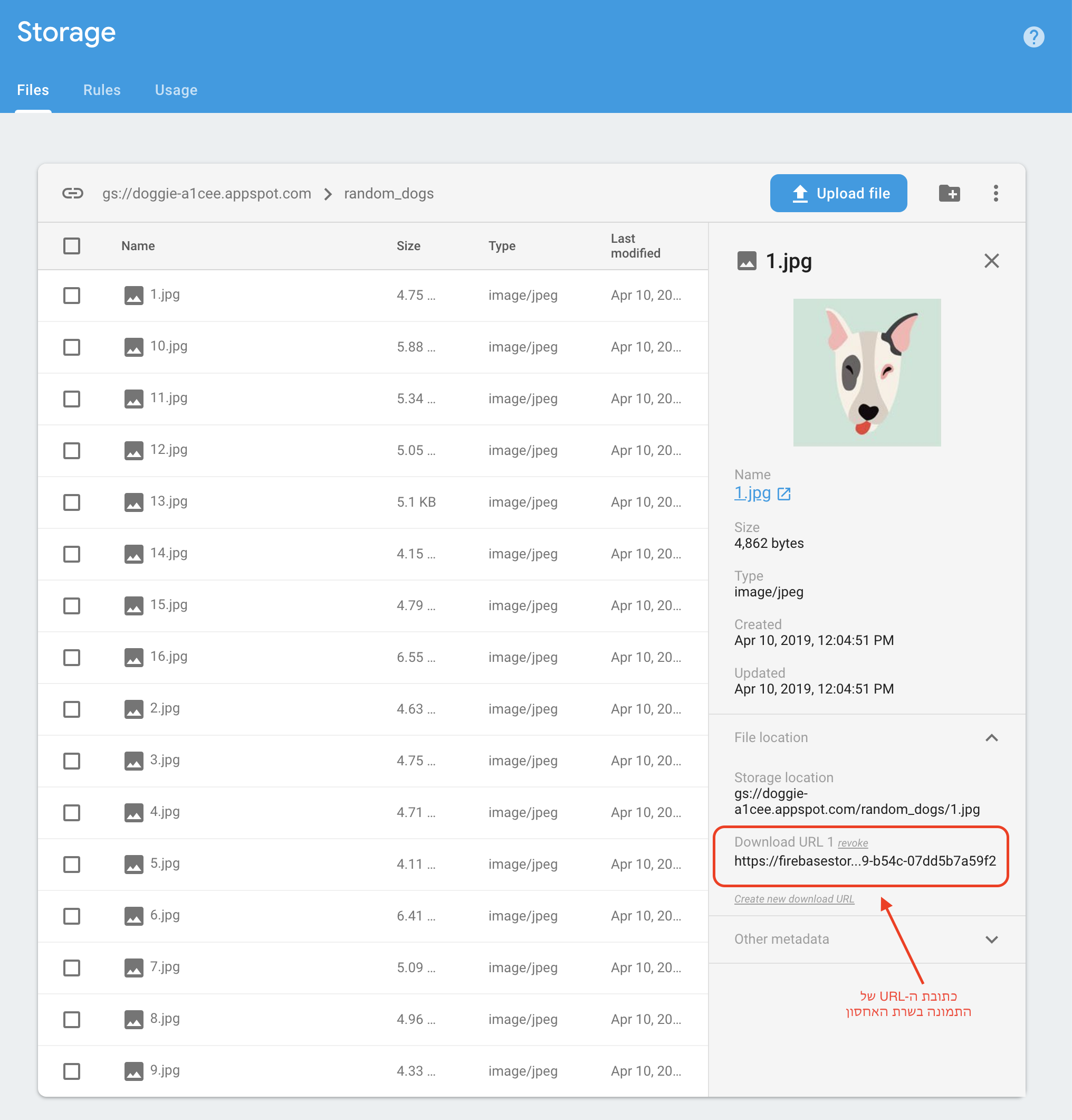
## שימוש ב-Firebase Storage

לצורך אחסון תמונות הפרופיל של משתמשי (וכלבי) האפליקציה, נעשה שימוש ב-Firebase Storage. כלי זה מאפשר לאחסן קבצים בשרת האחסון של Firebase ולגשת אליהם בעזרת כתובת URL המייצגת את מיקומם בשרת. לצורך הצגת תמונת פרופיל, לאחר שמשתמש בחר תמונה מהגלריה של המכשיר שלו או באמצעות מצלמה, הפונקציה onActivityResult מופעלת. בהתאם ל-requestCode שמועבר לפונקציה, אם אכן מדובר בבחירת תמונה ע"י המשתמש, לפונקציה מועבר הקובץ הנבחר בפורמט Uri. תחילה מתבצעת דחיסה של הקובץ על מנת לקצר את זמני ההמתנה הכרוכים בהעלאת הקובץ לשרת וטעינתו. הדחיסה מתבצעת באופן הבא:

Uri selectedImage = data.getData();

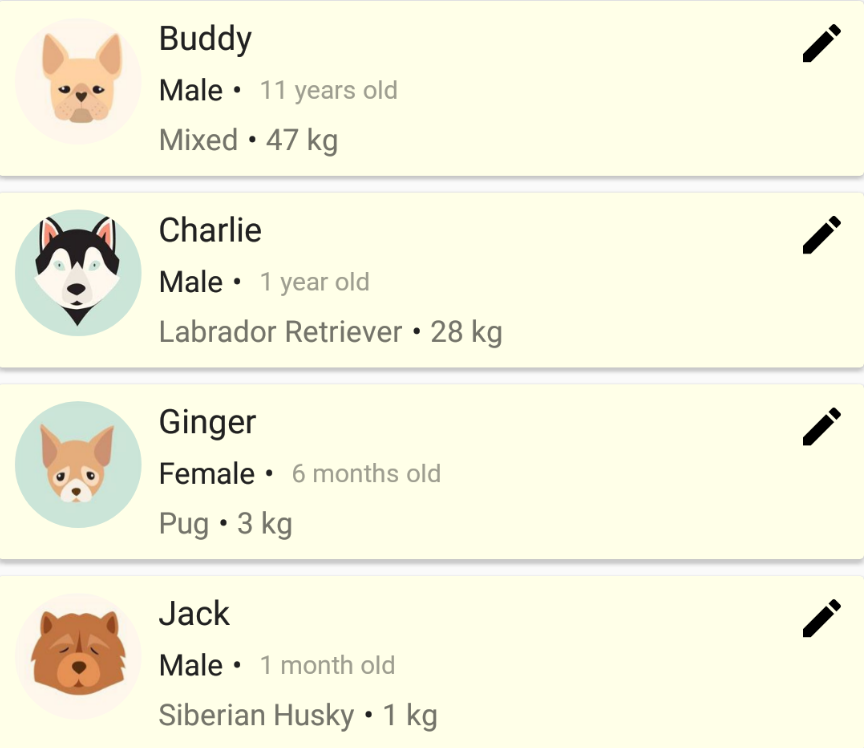
**cameraService**.setmPhotoFile(**cameraService**.getmCompressor()  
 .compressToFile(**new** File(**cameraService**.getRealPathFromUri(selectedImage))));

לאחר מכן, תוצג התמונה הנבחרת על המסך ועדיין לא תתבצע העלאת התמונה לשרת. הסיבה לכך היא שאם באופן אסינכרוני תתבצע ההעלאה לשרת והמשתמש יסיים את שלב מילוי הטופס וילחץ על כפתור השמירה, הוא יסגור את ה-activity הקיים ופעולת ההעלאה ל-storage תיקטע. תמונת הפרופיל שנבחרה ע"י המשתמש לא תישמר עבורו ולכן לא תוצג שוב בהמשך. לכן בחרנו באפשרות של לדחות את הגישה לשרת לצורך העלאת הקובץ עד לרגע שבו מתבצעת לחיצה על כפתור השמירה. ברגע שזה יקרה, למשתמש יוצג progress dialog שמונע ממנו להתקדם ולסגור את ה-activity עד שתושלם פעולת העלאת הקובץ בהצלחה. ברגע שזה יקרה, ה-progress dialog ייעלם, ה-activity הנוכחי ייסגר ויוחלף וה-intent הבא יוצג למשתמש. העלאת התמונה ל-Firebase storage מתבצעת ע"י קריאה לפונקציה saveProfileImageInFirebase שמקבל את ה-Uri של התמונה הנבחרת, יוצר path שבו התמונה תימצא ביחס ל-root ומבצע כתיבה באמצעות הפעלת putFile. לפעולת הכתיבה מוצמד listener, שמפעיל את onSuccess כאשר התמונה נמצאת בשרת. מאותו רגע יש להפיק את כתובת ה-URL של התמונה בשרת ולכן מופעל תהליך אסינכרוני נוסף שפונה לשרת האחסון עם הבקשה getDownloadUrl. במקרה של הצלחה, מבצעים השמה לערך המוחזר (כתובת ה-URL של התמונה) בשדה המתאים בתוך האובייקט שאליו שייכת התמונה וכותבים ל-database את האובייקט הנוצר.



## חישוב גיל הכלב מתוך תאריך לידה

הגיל של הכלב מחושב באופן דינמי בכל פעם שמוצגים למשתמש פרטיו: לאובייקט Dog יש שדה מסוג timestamp ששומר את תאריך הלידה (ללא שמירת גילו של הכלב בנוסף). באופן זה בכל רגע בו יוצג גילו של הכלב למסך, הוא יחושב מתוך ההפרש בין תאריך הלידה לתאריך של היום וכך כלב שפרטיו הוכנסו למערכת כשהיה בן שנה, גילו יהפוך להיות שנתיים לאחר שנה מיום רישום הכלב. החישוב מתבצע באופן הבא- שומרים שני מופעים של תאריך מטיפוס LocalDate המייצגים את תאריך הלידה ותאריך של היום (מבצעים casting מטיפוס timestamp כנדרש) ואז באמצעות המתודה monthsBetween של המחלקה Months, נקבל ספירה מדויקת של מספר החודשים שקיים בהפרש שבין שני התאריכים. כלומר, כלב שנולד לפני חודש ושבועיים, גילו יהיה חודש, בין אם נולד בתחילת החודש ובין אם נולד בסופו. לאחר מכן, על מנת להציג את גילו של הכלב בצורה נכונה בהתייחס ל-plurals, מתבצעת הבדיקה האם הוא בן חודש/ יותר מחודש אך פחות משנה/ שנה/ יותר משנה ובהתאם לכל אחד מהמקרים ייבנה ה-String המתאים.



## תצוגת מפה

מתוך התפריט הראשי ניתן להגיע למסך התצוגה של המשתמשים המחוברים על המפה, זאת במידה והמשתמש הנוכחי מחובר. אם המשתמש שינה את הסטטוס שלו ל-offline כאשר הוא מחובר לאפליקציה, הכפתור Show On Map אינו מאופשר (הופך ל-disabled) ובכך נמנעת מהמשתמש האפשרות לעבור לתצוגת מפה ולצפות במשתמשים אחרים (אם הוא בוחר למנוע את האפשרות של חשיפת מיקומו, מדוע שמשתמשים אחרים יאפשרו לו לצפות במיקומם?). באותו אופן, כאשר המשתמש משנה את הסטטוס שלו מ-online ל-offline, משתמשים אחרים שצופים במשתמשי המערכת בתצוגת מפה כבר לא יוכלו לראות את אותו משתמש והוא ייעלם מהמסך מיד בתום עדכון הסטטוס שלו ב-database (ועדכון התצוגה ע"י ה-adapter באופן אוטומטי לאחר עדכון ה-database).

מסך המפה מפוצל לשני חלקים שווים בגודלם- החלק העליון הוא תצוגת מפה עם סימון המשתמשים המחוברים במערכת והחלק התחתון הוא recycle view המציג את רשימת כל המשתמשים שמוצגים על המפה (ומחוברים באותו רגע במערכת).

לצד שמו של כל משתמש מוצג המרחק הנוכחי בינו לבין המשתמש הנוכחי המחובר במערכת. חישוב המרחק מתבצע על סמך הקואורדינטות השמורות תחת כל משתמש ב-database בפרטי המיקום שלו ומתעדכן בכל פעם שיש שינוי במיקום של אחד המשתמשים. העדכון מתבצע באופן הבא- כאמור השימוש ב-recycle view של Firebase מאפשר קריאה "אוטומטית" לפונקציה שאחראית על הצגת אובייקט בודד למסך בכל פעם שמתרחש שינוי במידע השמור ב-database המאפיין את אותו אובייקט. בתצוגת המפה, ה-adapter מתייחס לערכים מטיפוס User וכיוון שאחד השדות שלו הוא מיקומו הגיאוגרפי של המשתמש וקיים thread שמבצע פעילות רקע לעדכון מיקום המשתמש בכל פרק זמן קבוע ב-database, בכל פעם שהשינוי לגבי מיקום המשתמש ייכתב, תיקרא הפונקציה onBindViewHolder. הפונקציה הנ"ל מקבלת כפרמטר אובייקט מטיפוס User שאותו היא צריכה להציג למסך, וממנו היא לוקחת את התמונה שלו, שמו המלא וכתובת המייל שלו, ואת מיקומו הגאוגרפי. מתבצעת הבדיקה האם ה-user שהתקבל כפרמטר הוא אותו משתמש שמאומת כעת עם מערכת האימות של Firebase, ובמידה ומדובר בשני משתמשים שונים, מחושב המרחק מתוך ידיעת המיקום של שניהם. באותו אופן, כאשר משתמש שמופיע על המפה מתנתק מהמערכת, מתבצע שינוי במידע ששמור לגביו ב-database (השדה status משתנה מ-online ל-offline) ולכן השאילתה שמחפשת את כל המשתמשים המחוברים במערכת משתנה ופרטי המשתמש נעלמים מהמסך.

## צ'ט עם משתמש נוסף

לאחר שמשתמש המעוניין להפגיש בין הכלב שלו לכלב אחר המופיע במערכת בחר בכלב המתאים ביותר עבורו, הוא יכול לפתוח בשיחה עם אותו משתמש שהוא הבעלים של הכלב המתאים. אם הכלב נבחר בהתאם למאפייניו, מתוך לחיצה על הכלב המשתמש מועבר למסך התכתבות עם בעלי הכלב. כיוון שהמשתמשים מדברים ביניהם (ולא הכלבים), ולמספר כלבים יכול להיות אותם בעלים, בחירת הכלב תוביל לשיחה עם בעליו ושמור את היסטוריית ההודעות בין שני המשתמשים, בין אם ההתאמה הייתה בין כל אחד מהכלבים של כל אותם משתמשים. כיוון שמשתמש יכול לבחור בכלב גם על סמך מיקומו הגאוגרפי של הבעלים שלו, ניתן ממסך התצוגה של המפה להגיע לאותו מסך התכתבות ע"י בחירת המשתמש המתאים. התאמה בין בעלי הכלב לכלב עצמו נעשית מתוך מסך כל הכלבים, שם לצד כל כלב מוצג שם בעליו.

ברגע שנוצרת שיחה חדשה בין שני משתמשים מתווספת ל-database כניסה חדשה בתוך המיכל Chat Rooms שמכיל בתוכו את כל ההודעות שהועברו בין אותם משתמשים עד כה. כל הודעה מתואר בעזרת האובייקט ChatMessage ששומר את זהות המשתמש ששלח את ההודעה, timestamp של זמן שליחת ההודעה ותוכן ההודעה. באופן זה היסטוריית ההודעות של כל המשתמשים נשמרת וניתנת לטעינה באופן המדמה צ'ט אמתי. האובייקטים מוצגים למסך כאשר הם ממוינים לפי ה-timestamp כך שההודעה הישנה ביותר תופיע ראשונה ובנוסף אופן הצגת ההודעות נעשה כך שכל ההודעות שנשלחות ע"י אותו משתמש מיושרות לאותו צד של המסך וצבועות באותו צבע אחיד. המפתח המזהה של כל כניסה ב-database המתאימה לשיחה בין שני משתמשים מורכב משילוב של המפתחות המזהים של שניהם ובשרשור בסדר לקסיקוגרפי עולה. באופן זה, שיחה בין שני משתמשים תופיע פעם אחת בלבד ב-database וניתנת לשליפה ע"י שני המשתמשים באותה צורה.

# הצעות לעתיד

1. הוספת שיטות התחברות נוספות לצורך אימות המשתמש (למשל Facebook, Gmail, Twitter ועוד).
2. סימון נקודות ציון על המפה כיעדי מפגש מומלצים ודירוגם ע"י משתמשים שונים.
3. הוספת שדות נוספים המאפיינים כלבים לצורך התאמה טובה יותר בין כלבים כגון מזג הכלב, אופיו.
4. הוספת אפשרות של בחירת חברים מועדפים כך שהמשתמש יכול לחשוף את מיקומו בפניהם בלבד (ולהישאר בסטטוס offline מול יתר המשתמשים המחוברים).
5. הוספת התראות של הודעות בין משתמשים.