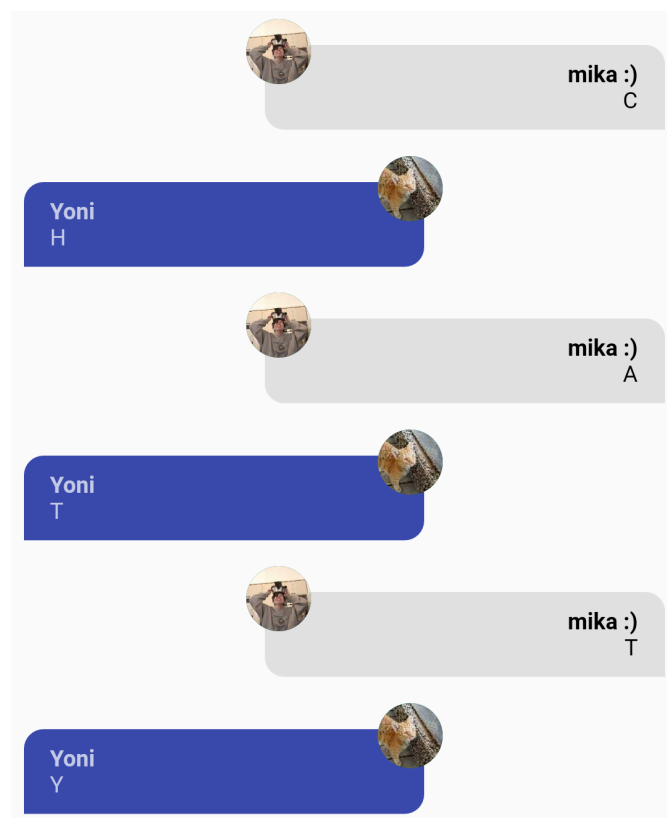


Chatty



תאריך הגשה: 28.06.2021

שם התלמידה: מיקה בוגר שוחט

ת.ז: 214235731

שם המורה: ניר סליקטר

בית הספר: הדמוקרטי ע"ש יעקב חזן, כפר סבא



תוכן עניינים

| | |
|-----------|--|
| 4 | מבוא |
| 4 | קוד הפרויקט ב-GitHub |
| 4 | תכולת הספר |
| 4 | רקע הפרויקט |
| 4 | תהליך המחקר |
| 4 | המצב הקיים בשוק |
| 4 | חידושים בפרויקט |
| 5 | אתגרים מרכזיים |
| 5 | הבעיה איתה התמודדתי |
| 5 | מוטיבציה לעבודה |
| 5 | הצורך עליו עונה הפרויקט וקהל היעד |
| 6 | ארכיטקטורת הפרויקט |
| 6 | הפתרון המוצע והסיבות לבחירתו |
| 7 | שרטוט על של הפרויקט |
| 8 | תפקיד כל יחידה בפרויקט |
| 8 | שרת |
| 8 | לקוח |
| 9 | Use Case Diagram |
| 10 | הצגת מקרה (use case) עבור הפונקציות העיקריות בפרויקט |
| 10 | הרשמה |
| 10 | התחברות |
| 11 | שליחת הודעה בצ'אט |
| 11 | צפייה ברשימת המשתמשים |
| 12 | שינוי שם משתמש |
| 12 | שינוי תמונת משתמש |
| 13 | התנתקות |
| 13 | טכנולוגיות בהן נעשה שימוש בפרויקט |
| 13 | Flutter |
| 13 | Firebase |
| 15 | מדריך למשתמש |
| 15 | הוראות התקנה |

| | |
|-----------|---|
| 15 | שלבי התקנת האפליקציה |
| 17 | היררכיית מסכים |
| 18 | מסכי האפליקציה |
| 18 | מסך הטעינה |
| 19 | מסך ההתחברות |
| 21 | מסך ההרשמה |
| 23 | מסך הצ'אט |
| 24 | Side drawer |
| 25 | מסך המשתמשים |
| 26 | מסך ההגדרות |
| 27 | בסיס הנתונים |
| 27 | הסבר על Firebase |
| 27 | בסיסי נתונים מסוג NoSQL |
| 28 | CREATE: |
| 28 | READ: |
| 28 | UPDATE: |
| 28 | DELETE: |
| 29 | תרשים כללי של בסיס הנתונים בפרויקט |
| 30 | הקשרים בין החלקים השונים ב-Firebase |
| 31 | ניהול בסיס הנתונים |
| 31 | Authentication |
| 31 | Storage |
| 31 | Users collection |
| 32 | Chat collection |
| 33 | מדריך למפתח |
| 34 | קוד הפרויקט ב-GitHub |
| 34 | הסבר כללי על קוד הפרויקט |
| 34 | Features של המכשיר שבשימוש בפרויקט |
| 34 | OOP - תכנות מונחה עצמים |
| 34 | Remote Procedure Call - הפעלת פרוצדורות מרחוק |
| 35 | תכנות אסינכרוני |
| 35 | איך מסודרת תיקיית הפרויקט |
| 36 | התיקה lib |
| 37 | קבצים, פונקציות וחלקי קוד חשובים |
| 38 | הגדרת החבילות שבשימוש בפרויקט |
| 38 | שימוש במצלמת המכשיר וגישה לאחסון התמונות |

| | |
|-----------|---|
| 39 | החלק הלוגי - גישה למצלמה/גלריית התמונות |
| 39 | ממשק המשתמש |
| 41 | תקשורת עם השרת ועם בסיס הנתונים |
| 41 | firebase_auth |
| 43 | cloud_firestore |
| 44 | firebase_storage |
| 44 | רפלקציה |
| 45 | כלים שרכשתי בדרך |
| 45 | קשיים ואתגרים שעמדו בפני |
| 45 | מסקנותי מהפרויקט |
| 45 | מה הייתי עושה אחרת לו הייתי מתחילה היום |
| 46 | מה היה יכול להפוך את עבודתי ליעילה יותר |
| 46 | תכונות שהייתי רוצה להוסיף לפרויקט |
| 46 | ביבליוגרפיה |
| 47 | מקורות כלליים |

מבוא

קוד הפרויקט ב-GitHub

<https://github.com/mb-s/Chatty>

תכולת הספר

ספר זה מציג את פרויקט הגמר שלי במגמת סייבר - אפליקציית צ'אט בשם Chatty. בספר מפורט מבנה הפרויקט, טכנולוגיות בהן נעשה שימוש, על הפונקציות המרכזיות בו ועוד. בנוסף הוא מכיל מדריך למשתמש ומדריך למפתח. בספר מתועד גם התהליך שעברתי בעבודה על הפרויקט, איך הוא השפיע עלי ואילו כלים רכשתי מהעבודה עליו.

רקע הפרויקט

בהתחלה כשחשבתי על איזה פרויקט גמר ארצה לעשות למגמת סייבר, החלטתי לעשות אפליקציה לשיתוף מוזיקה בזמן אמת. בשביל לעשות זאת התחלתי קורס באתר Udemy שמלמד פיתוח אפליקציות בשפת dart עם flutter שהיא פלטפורמה לפיתוח אפליקציות גם לאנדרואיד וגם ל-ios בעזרת codebase אחד שפותחה ומתוחזקת על ידי גוגל.

פרויקט הגמר של הקורס הוא אפליקציית צ'אט בסיסית, ואחרי שהבנתי שעבודה עם אודיו בפיתוח אפליקציות היא מסובכת וקשה מאוד, החלטתי לקחת את התוצר הסופי של הקורס ולשפר אותו לפי מה שנראה לי לנכון.

תהליך המחקר

המצב הקיים בשוק

ישנן המון אפליקציות צ'אט בשוק, הפופולריות ביותר היא whatsapp, וכמובן כמעט לכל פלטפורמת מדיה חברתית כמו instagram או facebook יש גם צ'אט בתוכה. בגלל מגבלת הזמן לא יכולתי לשלב את כל המאפיינים שיש ברוב אפליקציות הצ'אט (כמו הודעות פוש או פתיחת קבוצות רק עם משתמשים מסויימים). למרות זאת השתדלתי להוסיף כמה שיותר פיצ'רים, ואת אלה שנראו לי המשמעותיים ביותר.

חידושים בפרויקט

בגלל היות האפליקציה פשוטה ולא בעלת מרכיבים רבים ומסובכים, היא מתאימה יותר למבוגרים שמתקשים להבין טכנולוגיה ונאבדים בכל הפיצ'רים של אפליקציות כמו whatsapp. היא גם יותר מתאימה לילדים צעירים, שעדיין לא מנוסים בשימוש במבוך של אפליקציות צ'אט.

אתגרים מרכזיים

הבעייה איתה התמודדתי

במסגרת מגמת מדעי המחשב ומגמת סייבר, למדתי שלוש שפות תכנה (python, C#, ASSEMBLY), אבל באף אחת מהן לא למדתי איך לפתח אפליקציות מובייל ולא יצא לי לעבוד על פרויקט בסדר גודל כמו זה. לכן, וגם בגלל העבודה עם שפה חדשה שלא הכרתי לפני כן, בתחילת העבודה על הפרויקט, נתקלתי בכשלונות רבים, רעיון שלא הצלחתי להבין איך לממש, קוד שלא עובד וכדומה, וכל כישלון הפחית את המוטיבציה שלי להמשיך בעבודה עם הפרויקט. למעשה הקושי המרכזי שלי היה להתמודד עם למידה של השפה ועם הכשלונות של עבודה על פרויקט גדול באותו הזמן בלי לאבד את העניין בעבודה.

מוטיבציה לעבודה

כשהתמקדתי קודם בסיום הקורס אליו נרשמתי, שמלמד פיתוח אפליקציות בעזרת dart ו-futter ובו מתנסים בפיתוח של אפליקציות רבות בעלות פיצ'רים שונים, הצלחתי להבין יותר לעומק את העבודה איתן. כך לאחר מכן, כשעברתי לעבודה בלעדית על הפרויקט ידעתי להתמודד יותר טוב עם בעיות בהן נתקלתי ולממש את הרעיונות שהיו לי. בנוסף, העבודה עם בסיס כלשהו (במקרה זה אפליקציית הצ'אט שמפתחים בסוף הקורס) אפשרה לי לחשוב על כל מני שיפורים שניתן להוסיף לה ובכך נתנה לי מוטיבציה חדשה וחזקה לעבוד על הפרויקט.

הצורך עליו עונה הפרויקט וקהל היעד

הפרויקט ענה על הצורך בתקשורת פשוטה ונוחה בין אנשים, בטקסט דרך הטלפונים שלהם. בנוסף האפליקציה מאפשרת לתקשר גם עם זרים שנרשמים אליה. בזכות פשטותה של האפליקציה, קהל היעד שלה הוא כל אחד שיש לו טלפון חכם ורוצה לתקשר עם חבריו - מילדים צעירים לקשישים.

ארכיטקטורת הפרויקט

הפתרון המוצע והסיבות לבחירתו

לפרויקט שלי רציתי לעשות אפליקציית צ'אט שמאפשרת לכל משתמש לכתוב הודעה שכל השאר יוכלו ישר לראות, והיה לי חשוב שהמשתמשים יוכלו לשנות את שם המשתמש שלהם ואת תמונת הפרופיל שלהם. בחרתי לפתח את האפליקציה שרציתי בעזרת השפה dart עם הפלטפורמה flutter.

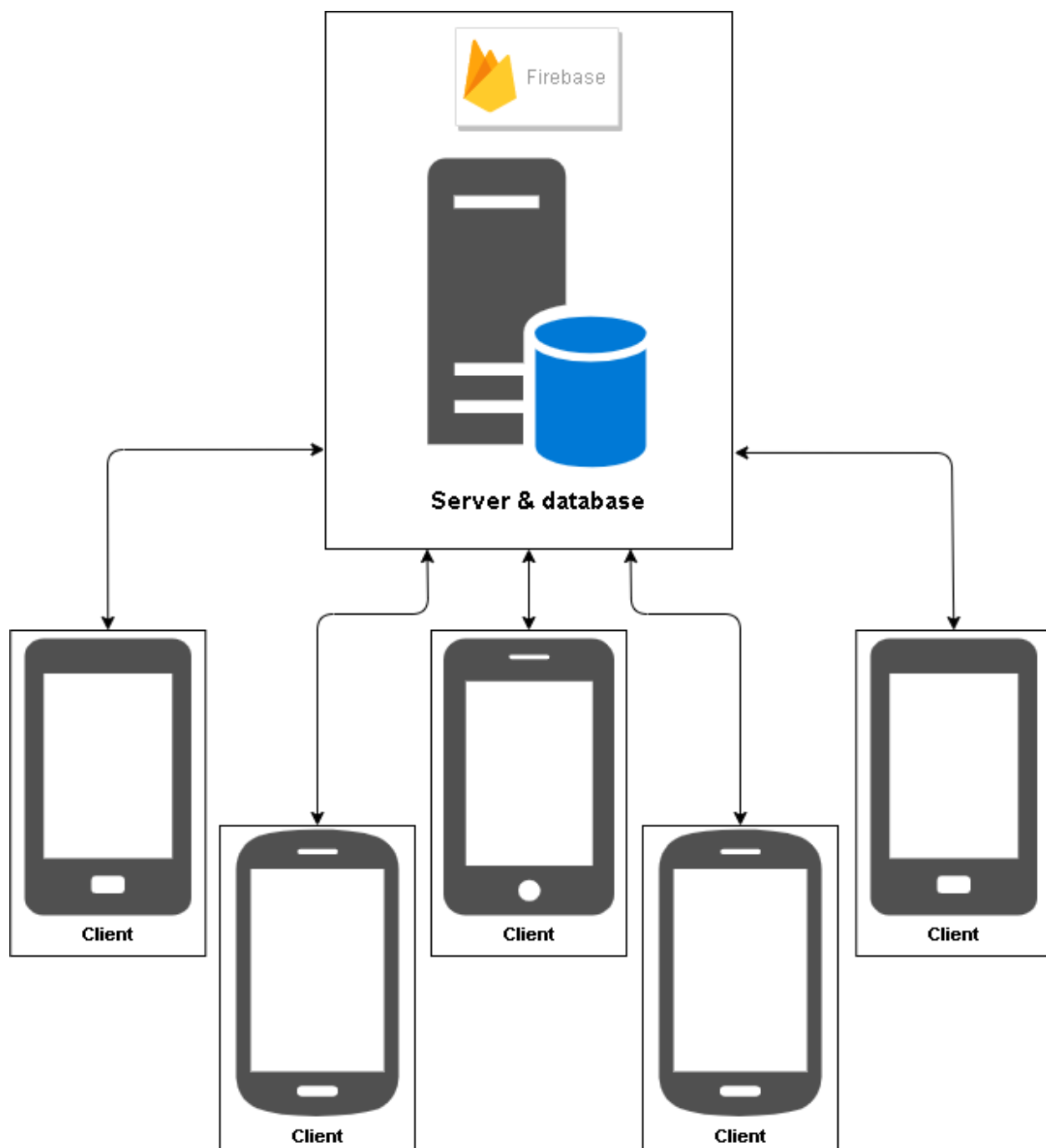
הסיבה העיקרית שבחרתי ב-flutter היא מכיוון שהיא מאפשרת פיתוח אפליקציות גם ל-ios וגם לאנדרואיד. מעבר לכך היא גם נוחה מאוד לשימוש ומבוססת על השפה dart שדומה יחסית לשפות שכבר הכרתי, והיה לה קורס ברור, מקיף ונוח שיכולתי להשתמש בו בשביל ללמוד לעומק איך להשתמש בה.

ישנן כמובן עוד פלטפורמות בעזרתן אפשר לפתח אפליקציות ios ואנדרואיד תחת אותו בסיס קוד (react native) נכאלה שאפילו מתבססות על שפות שכבר הכרתי מראש (kivy שעובדת עם python), אבל flutter נראתה לי מאתגרת, כך שאוכל לצאת מאיזור הנוחות שלי, אך מוכרת, מתוחזקת ומתועדת מספיק שיהיה לי חופש לעשות מה שאני רוצה ושאוכל למצוא פתרון לכל בעייה שבה אתקל.

השתמשתי ב-firebase בתור שרת, כך שלא הייתי צריכה לכתוב את צד השרת לאפליקציה. firebase משמש כבסיס נתונים, ניתן לאחסן בו קבצים כמו תמונות ויש לו אופציית הרשמה של משתמשים לכן היה מאוד נוח לעבוד איתו בתור כל ה-backend של הפרויקט בלי הרבה עבודה מצידי. אני התמקדתי בעיקר בכתיבת צד הלקוח. firebase מותאם ל-flutter ויש חבילות רבות שמקלות מאוד על התקשורת איתו בתור שרת.

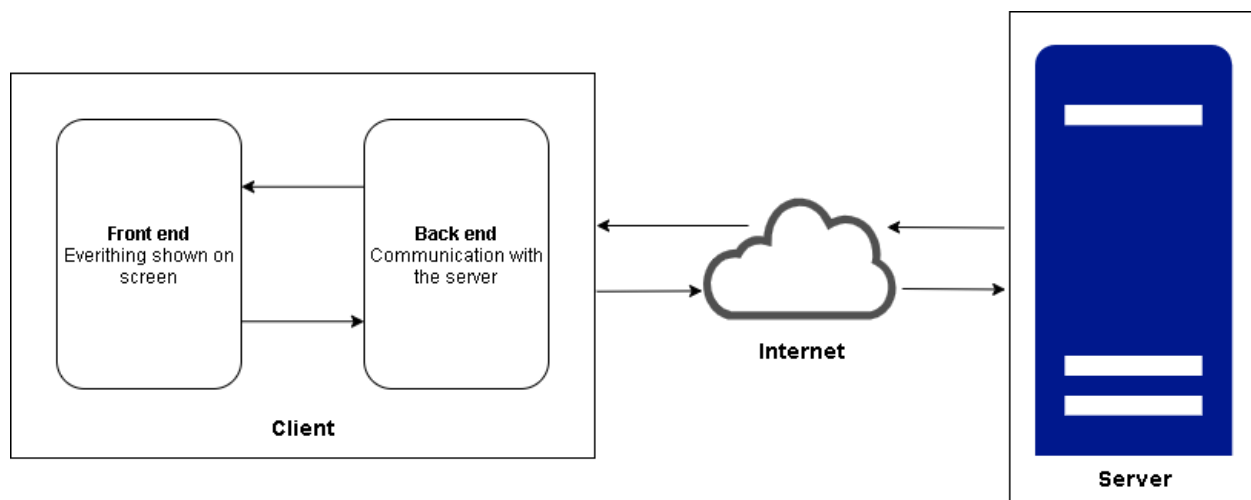
שרטוט על של הפרויקט

איך המשתמשים מתקשרים אחד עם השני דרך השרת.



תפקיד כל יחידה בפרויקט

התקשורת בין הלקוח לשרת:



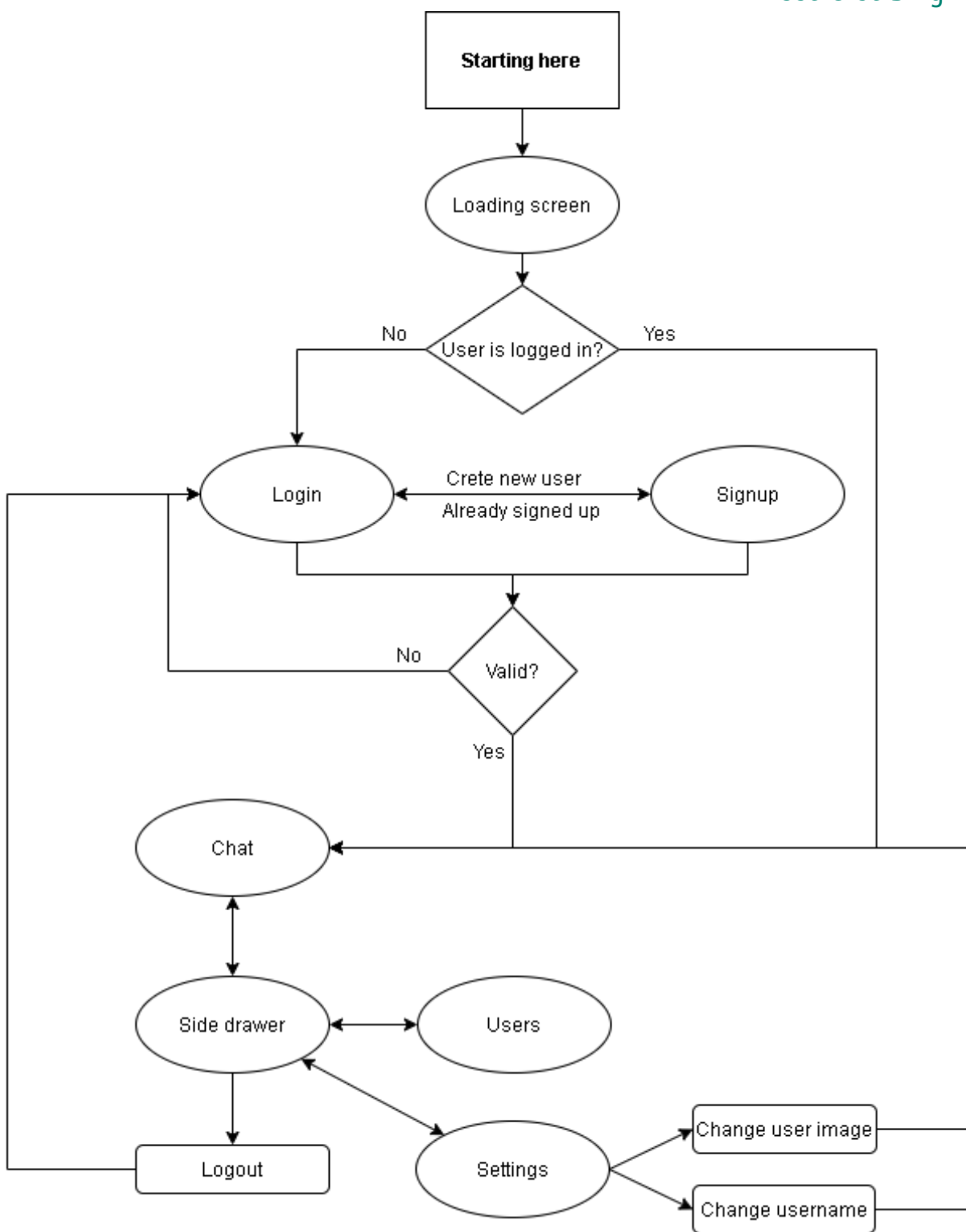
שרת

אל השרת נשלחת כל הודעה ששולח משתמש, והוא שומר אותה בבסיס הנתונים כהודעה בצ'אט יחד עם, תאריך שליחתה וה-ID של השולח, שלמעשה מצביע על מסמך בבסיס הנתונים בו שמור המידע שלו. כאשר משתמש חדש נרשם, כתובת המייל שלו, שם המשתמש שלו, וקישור לתמונת הפרופיל שלו נשמרים במאגר הנתונים תחת ה-ID שמתקבלת בעת הרשמתו. הסיסמה של המשתמש מוצפנת במאגר הנתונים ואין אליה גישה.

לקוח

הלקוח הוא משתמש של האפליקציה. הוא שולח בקשות לשרת ומעדכן את בסיס הנתונים בו (לפי החוקים של בסיס הנתונים). הוא מקבל מהשרת את כל ההודעות שנשלחו בצ'אט ומציג אותן על המסך, וכאשר יש צורך בעדכון בסיס הנתונים (למשל שליחת הודעה, או שינוי שם המשתמש) הוא שולח לשרת את העדכון.

Use Case Diagram



הצגת מקרה (use case) עבור הפונקציות העיקריות בפרויקט

הרשמה

תיאור הפעולה: הרשמת או ניסיון הרשמת המשתמש במערכת.

תנאים מקדימים: המשתמש הפעיל את האפליקציה ונמצא במסך ה-sign up, למשתמש חיבור פעיל לאינטרנט.

תנאי סיום: המשתמש נרשם במערכת.

שלבי פעולה:

1. כניסה למסך ההרשמה
2. מילוי שדות הטקסט במסך ההרשמה (כתובת מייל, שם משתמש, סיסמה ואישור סיסמה) לפי הדרישות בכל אחד ובחירת תמונה
3. אחרי לחיצה על כפתור "Sign up", מתבצעת בצד המשתמש בדיקה של הנתונים שמילא (למשל: כתובת מייל מכילה "@", הסיסמה זהה בשני שדות הטקסט ואורכה לפחות 7 תווים, אף שדה לא ריק). אם נמצא שאחד מהתנאים לא מתקיים, המשתמש מתבקש לתקן את השדה הלא תקין, אחרת ממשיכים לשלב הבא.
4. שליחת פרטי המשתמש לשרת¹ firebase שגם הוא מבצע בדיקה של הנתונים (האם כבר רשום משתמש עם כתובת המייל הזו, אם יש בעייה מוחזרת הודעה שמופיעה על המסך של המשתמש).
5. אם לא עלתה בעייה, פרטי המשתמש נשמרים בבסיס הנתונים והוא מתחבר לאפליקציה ומועבר למסך הראשי - מסך הצ'אט.

התחברות

תיאור הפעולה: התחברות או ניסוי התחברות של משתמש למערכת.

תנאים מקדימים: המשתמש הפעיל את האפליקציה ונמצא במסך ה-login, למשתמש חיבור פעיל לאינטרנט.

תנאי סיום: המשתמש התחבר למערכת.

שלבי פעולה:

1. כניסה למסך ההתחברות
2. מילוי שדות הטקסט לפי הדרישות בכל אחד (כתובת מייל וסיסמה)
3. אחרי לחיצה על כפתור "Login", מתבצעת בצד המשתמש בדיקה של הנתונים שמילא (למשל: כתובת מייל מכילה "@", אורך הסיסמה לפחות 7 תווים, אף שדה לא ריק). אם נמצא שאחד

¹ Firebse הוא שרת ובסיס נתונים בו אני משתמשת בפרויקט שלי. מוסבר עליו עוד בפרק "בסיס הנתונים".

מהתנאים לא מתקיים, המשתמש מתבקש לתקן את השדה הלא תקין, אחרת ממשיכים לשלב הבא.

4. שליחת פרטי המשתמש לשרת firebase שבודק אם כתובת המייל רשומה במערכת ואם הסיסמה שהזין המשתמש מתאימה לה. אם נמצא שלא, משלחת למשתמש הודעה על כך שמופיעה על המסך.
5. אם כתובת המייל והסיסמה תואמות ורשומות במערכת, מתחבר המשתמש למערכת והוא מועבר למסך הראשי - מסך הצ'אט.

שליחת הודעה בצ'אט

תיאור הפעולה: נסיון משתמש לשלוח הודעה בצ'אט

תנאים מקדימים: המשתמש הפעיל את האפליקציה ויש לו חיבור פעיל לאינטרנט, המשתמש כבר מחובר לחשבון כלשהו ונמצא במסך הצ'אט.

תנאי סיום: הודעת המשתמש נשלחה בצ'אט.

שלבי פעולה:

1. כניסה למסך הצ'אט
2. פתיחת שדה הטקסט "Send a message" וכתיבת ההודעה
3. לחיצה על כפתור השליחה (כמו חץ ימינה)
- אם שדה הטקסט ריק המשתמש לא יוכל ללחוץ על הכפתור
4. ההודעה נשלחת לשרת firebase ששומר אותה בבסיס הנתונים תחת ה-collection² שנקרא chat בתור document משלה (יחד עם תוכן ההודעה נשמרים תאריך וזמן שליחתה וה-ID של המשתמש ששלח אותה)
5. כל המשתמשים מקבלים את ההודעה כשהיא מתווספת לבסיס הנתונים (מכיוון שבסיס הנתונים הוא בזמן אמת)

צפייה ברשימת המשתמשים

תיאור הפעולה: נסיון משתמש לצפות במשתמשים הרשומים

תנאים מקדימים: תנאים מקדימים: המשתמש הפעיל את האפליקציה ויש לו חיבור פעיל לאינטרנט, המשתמש כבר מחובר לחשבון כלשהו ונמצא במסך הצ'אט.

תנאי סיום: הוצגה על המסך רשימת המשתמשים הרשומים במערכת

שלבי פעולה:

1. כניסה למסך הצ'אט
2. פתיחת המגירה הצדדית (ע"י לחיצה על הכפתור השמאלי העליון)
3. לחיצה על הכפתור "Users"

² המושגים collection, document וגם ID של משתמש קשורים ל-firebase ומפורטים בפרק "בסיס הנתונים".

- לאחר הלחיצה על הכפתור נשלחת לשרת firebase בקשה לקבלת המידע על כל המשתמשים הרשומים (מה שרשום בבסיס הנתונים, לא הסיסמאות שלהם)
- 4. הצגת רשימת המשתמשים על המסך עם שם המשתמש, התמונה וכתובת המייל של כל אחד

שינוי שם משתמש

תיאור הפעולה: משתמש מנסה לשנות את שם המשתמש שלו

תנאים מקדימים: המשתמש הפעיל את האפליקציה ויש לו חיבור פעיל לאינטרנט, המשתמש כבר מחובר לחשבון כלשהו ונמצא במסך הצ'אט.

תנאי סיום: שם המשתמש שונה והמשתמש הוחזר למסך הצ'אט.

שלבי פעולה:

1. כניסה למסך הצ'אט
2. פתיחת המגירה הצדדית (ע"י לחיצה על הכפתור השמאלי העליון)
3. לחיצה על הכפתור "Settings"
4. שינוי שם המשתמש (כתוב בתיבת הטקסט "Username") ולחיצה על כפתור "Update"
- הלחיצה על הכפתור שולחת לשרת firebase בקשה לשינוי השדה username ב-document ששמו הוא ה-ID של המשתמש תחת ה-collection ששמו users בבסיס הנתונים
5. חזרה אל מסך הצ'אט אחרי בנייתו מחדש (בשביל להציג את שם המשתמש החדש)

שינוי תמונת משתמש³

תיאור הפעולה: משתמש מנסה לשנות את התמונה שלו

תנאים מקדימים: המשתמש הפעיל את האפליקציה ויש לו חיבור פעיל לאינטרנט, המשתמש כבר מחובר לחשבון כלשהו ונמצא במסך הצ'אט.

תנאי סיום: תמונת המשתמש שונתה והמשתמש הוחזר למסך הצ'אט

שלבי פעולה:

1. כניסה למסך הצ'אט
2. פתיחת המגירה הצדדית (ע"י לחיצה על הכפתור השמאלי העליון)
3. לחיצה על הכפתור "Settings"
4. לחיצה על הכפתור "Change image" ובחירת מקור התמונה החדשה (גלריית התמונות או המצלמה)
5. בחירת התמונה החדשה (צילומה או בחירה מגלריית התמונות) ואישור
- כעת תופיע התמונה החדשה בעיגול העליון במקום התמונה הקודמת של המשתמש

³ ניתן לשנות באותה הפעם גם את שם המשתמש וגם את תמונתו, כל שנדרש הוא עדכון שני השדות לפני לחיצה על הכפתור "Update".

6. לחיצה על כפתור "Update"

- הלחיצה על הכפתור שולחת לשרת firebase בקשות למחיקת תמונה באחסון ששמה הוא ה-ID של המשתמש, לשמירת התמונה החדשה תחת שם זה ולעדכון השדה imageUrl ב-document ששמו הוא ה-ID של המשתמש תחת ה-collection ששמו users בבסיס הנתונים
7. חזרה אל מסך הצ'אט אחרי בנייתו מחדש (בשביל להציג את התמונה החדשה)

התנתקות

תיאור הפעולה: נסיון התנתקות משתמש מהמערכת.

תנאים מקדימים: המשתמש הפעיל את האפליקציה ויש לו חיבור פעיל לאינטרנט, המשתמש כבר מחובר לחשבון כלשהו ונמצא במסך הצ'אט.

תנאי סיום: המשתמש נותק מהחשבון.

שלבי פעולה:

1. כניסה למסך הצ'אט
 2. פתיחת המגירה הצדדית (ע"י לחיצה על הכפתור השמאלי העליון)
 3. לחיצה על כפתור "Logout"
- לאחר הלחיצה על הכפתור נשלחת לשרת בקשה לנתק את המשתמש והוא עושה זאת, רק לאחר קבלת אישור על הניתוק ממשיכים
4. מעבר למסך ההתחברות


טכנולוגיות בהן נעשה שימוש בפרויקט

Flutter

כפי שכבר צוין, flutter היא פלטפורמה לבניית אפליקציות שפותחה על ידי גוגל. היא מאפשרת לבנות בעזרת בסיס קוד אחד, בשפת dart, אפליקציות אנדרואיד, ios, linux ועוד. היא נחשבת לפלטפורמה נוחה מאוד, בין השאר הודות לדוקומנטציה הנרחבת שלה.

השפה בה מפתחים אפליקציות עם flutter היא dart, שפה קלה להבנה שדומה מעט ל-C# ול-python, השפות שלמדתי במסגרת מגמת מדעי המחשב ומגמת סייבר בבית הספר. מכיוון שהשפה כל כך קלה ללמידה, ומכיוון שכבר הכרתי שפות דומות, הייתי יכולה לקפוץ יש לפיתוח ב-flutter בלי להשקיע הרבה בלמידת השפה עצמה, והלמידה להשתמש ב-flutter לא הייתה מסובכת מדי, שכן הכל די ברור לפי שמו.

Firebase



בפרויקט שלי, firebase, שגם היא פלטפורמה שפותחה על ידי גוגל בשביל אפליקציות מובייל, משמשת כשרת (server), בסיס נתונים בזמן אמת (real time database שנקרא firestore), ואחסון קבצים (תמונות).

מכיוון שהיא מתוחזקת על ידי גוגל, firebase היא פלטפורמה נוחה מאוד וקלה לשימוש ולניהול, עם אתר מוסבר מאוד ונוח למשתמש, שאפשר לראות בעזרתו בקלות את כל המאוחסן בבסיס הנתונים. יש הרבה מאוד הסברים באינטרנט על העבודה עם כל מאפיין שלה, והיא תומכת ישירות באפליקציות flutter כך שנוח מאוד לתקשר איתה דרך הקוד, בעזרת מגוון רחב של חבילות.

מדריך למשתמש

הוראות התקנה

את האפליקציה שלי אמור להיות אפשר להריץ על הפלטפורמות: אנדרואיד ו-ios. מכיוון שאין לי גישה למחשב בעל מערכת הפעלה macOS, אני לא יכולה לבדוק האם היא באמת רצה על מכשירי ios, שכן רק כך ניתן לבדוק.

אם תועלה האפליקציה לחנות האפליקציות (כרגע היא לא הועלתה), התקנתה במכשיר תהיה קלה הרבה יותר; פשוט למצוא אותה בחנות האפליקציות של המכשיר וללחוץ על התקנה.

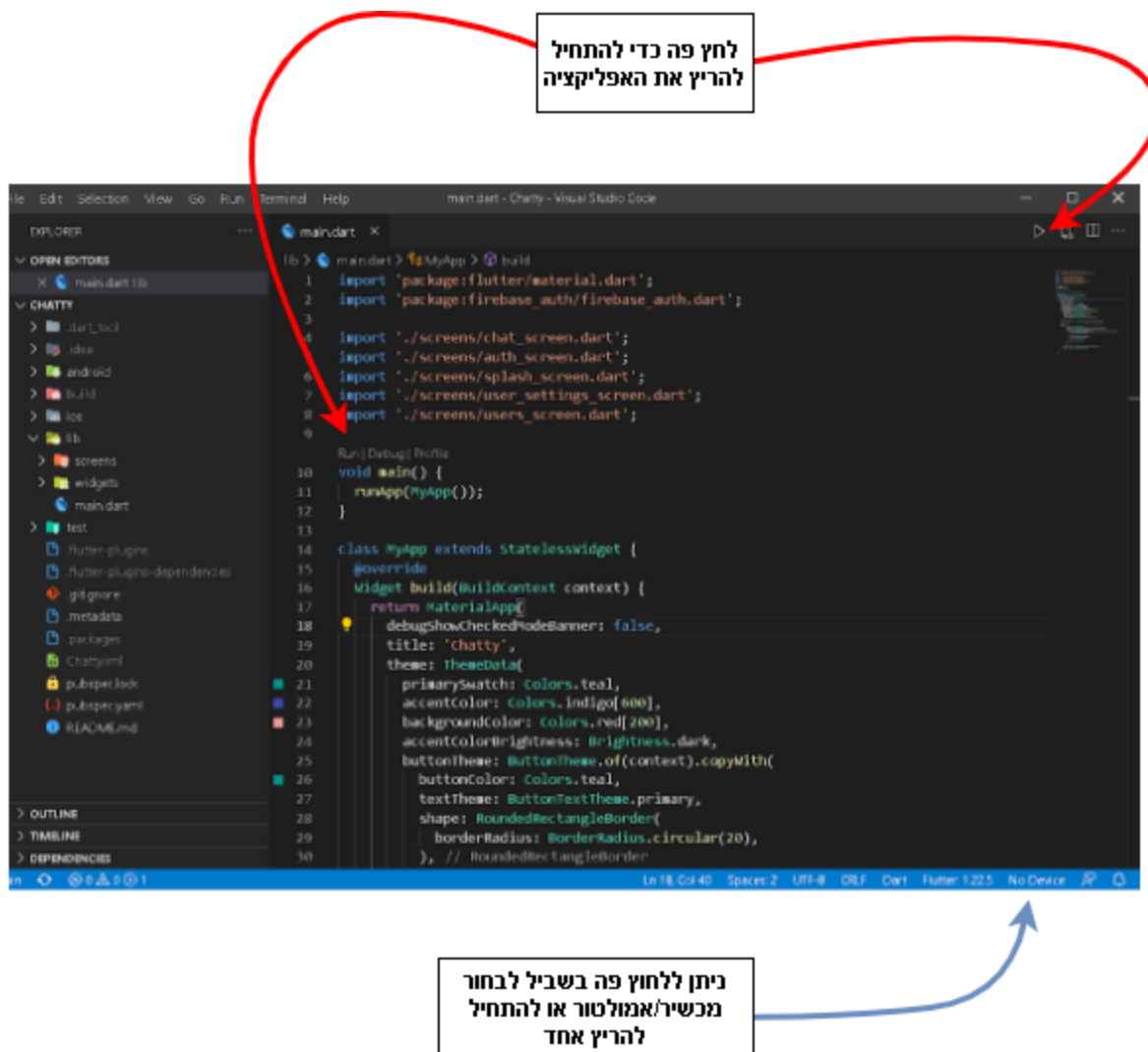
שלבי התקנת האפליקציה

1. התקנת flutter על המחשב בעזרת המדריך: [ל-windows](#), [ל-macOS](#)
2. **אנדרואיד:** להתקין Android Studio וליצור אמולטור של מכשיר אנדרואיד שירוצ על המחשב בלינק - <https://www.youtube.com/watch?v=WkEf1fa1sn0>
- ios** (אפשרי רק על מחשבים בעלי מערכת הפעלה macOS): להתקין Xcode מחנות האפליקציות וליצור אמולטור של מכשיר ios שירוצ על המחשב בלינק - <https://bit.ly/3tUKqYi>
- אפשר גם להתקין את האפליקצי ישירות על הטלפון ע"י חיבורו למחשב בכבל: USB [windows](#), [macOS](#)
3. להתקין Visual Studio Code (לא חובה אבל הוא נוח מאוד)
4. להוריד את קוד הפרויקט מ-GitHub ולפתוח ב-VSCode
- אם לא רוצים לעבוד עם עורך קוד, אפשר להריץ ב-command prompt את הפקודה [flutter doctor](#) כדי לוודא שיש מכשיר כלשהו מחובר ([ליצירת אמולטור דרך Android Studio](#)), אמור להיות כתוב:

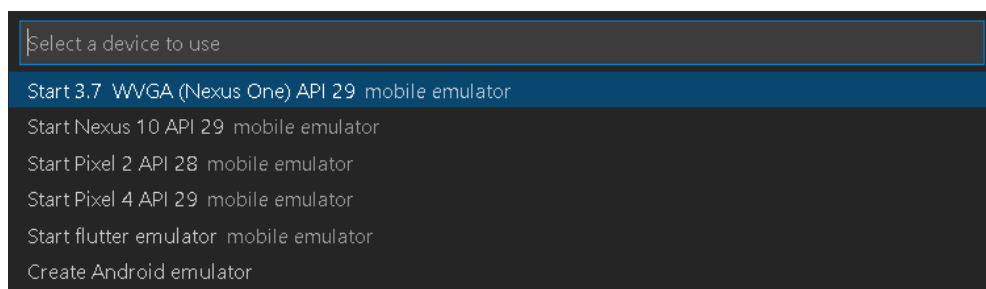
```
[✓] Connected device (1 available)
```

כאשר יש מכשיר מחובר, יש להיכנס לתיקיה של הפרויקט דרך cmd ולהריץ את הפקודה `flutter run`.

5. לפתוח את התיקיה lib ובתוכה את הקובץ main.dart. המסך יראה בערך כך:



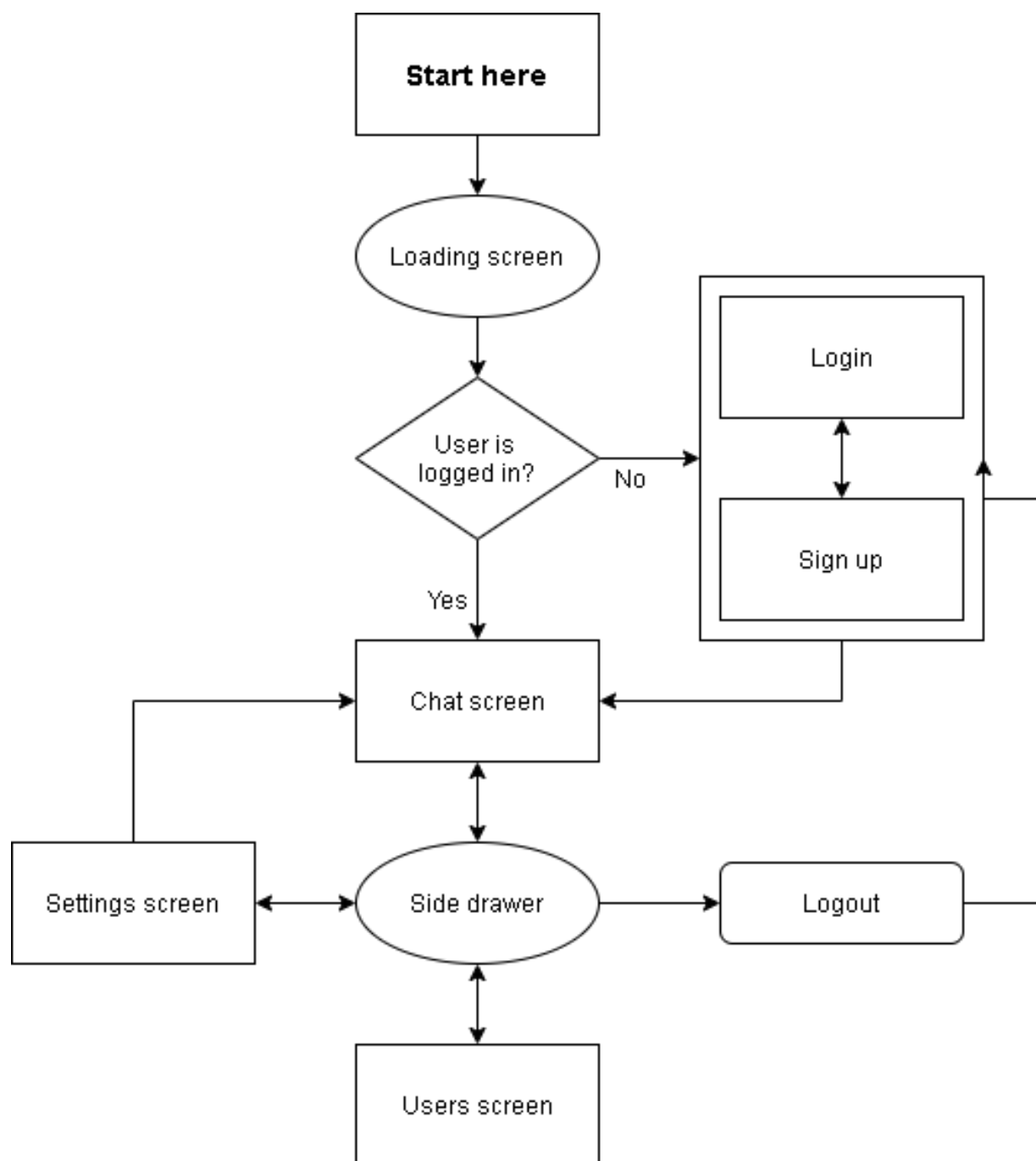
6. אם יש מכשיר מחובר, האפליקציה תתחיל לרוץ עליו ישר אחרי לחיצה על הכפתור המתאים, אם לא, תיפתח בראש המסך אפשרות לבחור מכשיר מהאמולטורים שכבר קיימים על המחשב:



יש לבחור באחד מהאמולטורים והוא יתחיל לרוץ עם האפליקציה עליו

7. זהו! זה לוקח קצת זמן אבל האפליקציה תותקן על המכשיר

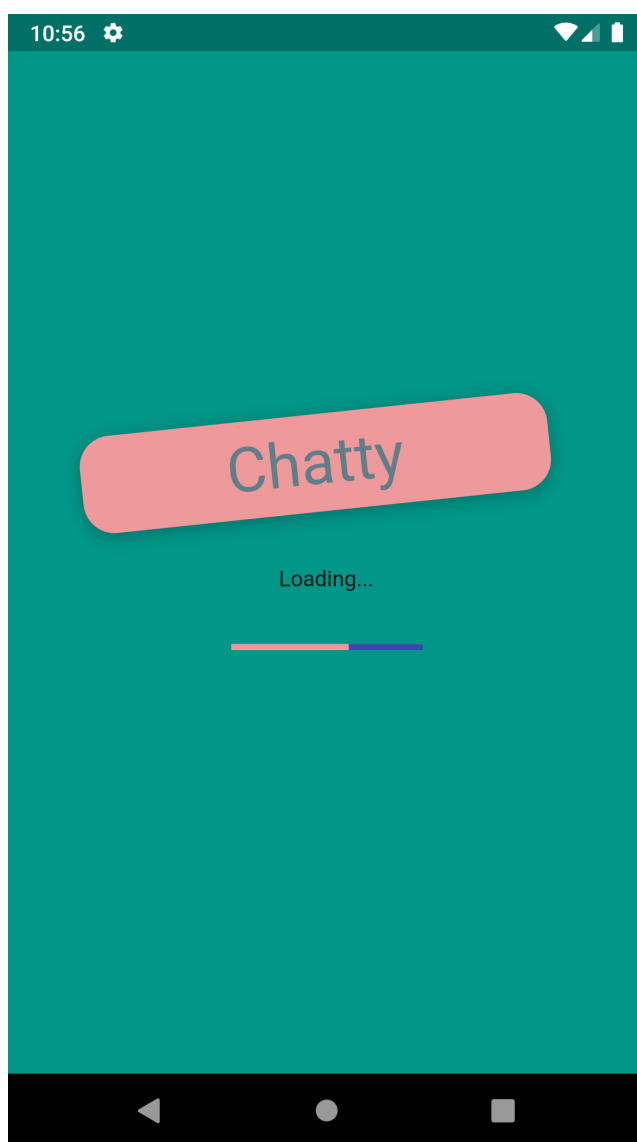
היררכיית מסכים



מסכי האפליקציה

מסך הטעינה

מסך זה מוצג למשתמש בזמן שהאפליקציה נטענת.



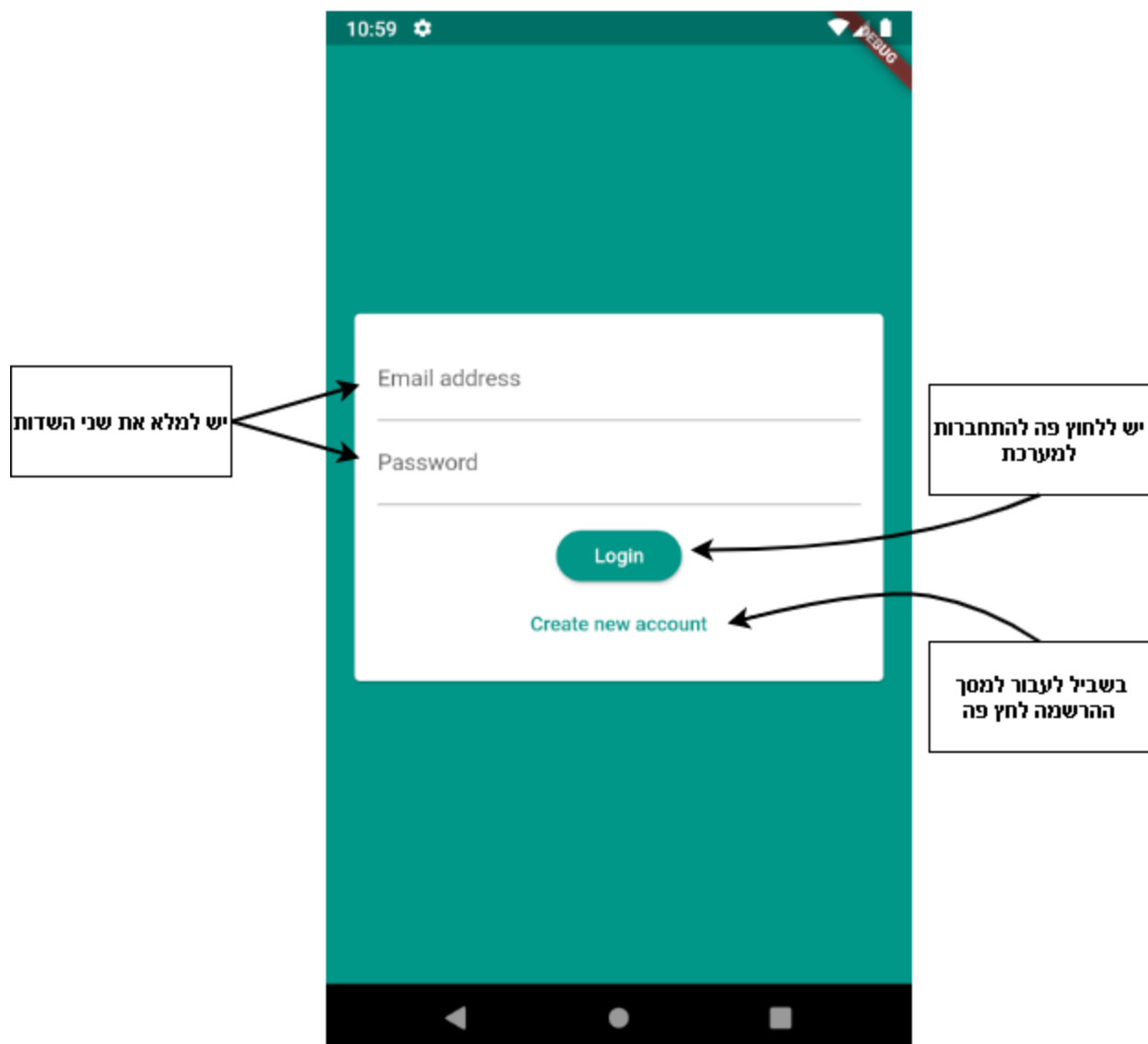
מסך ההתחברות

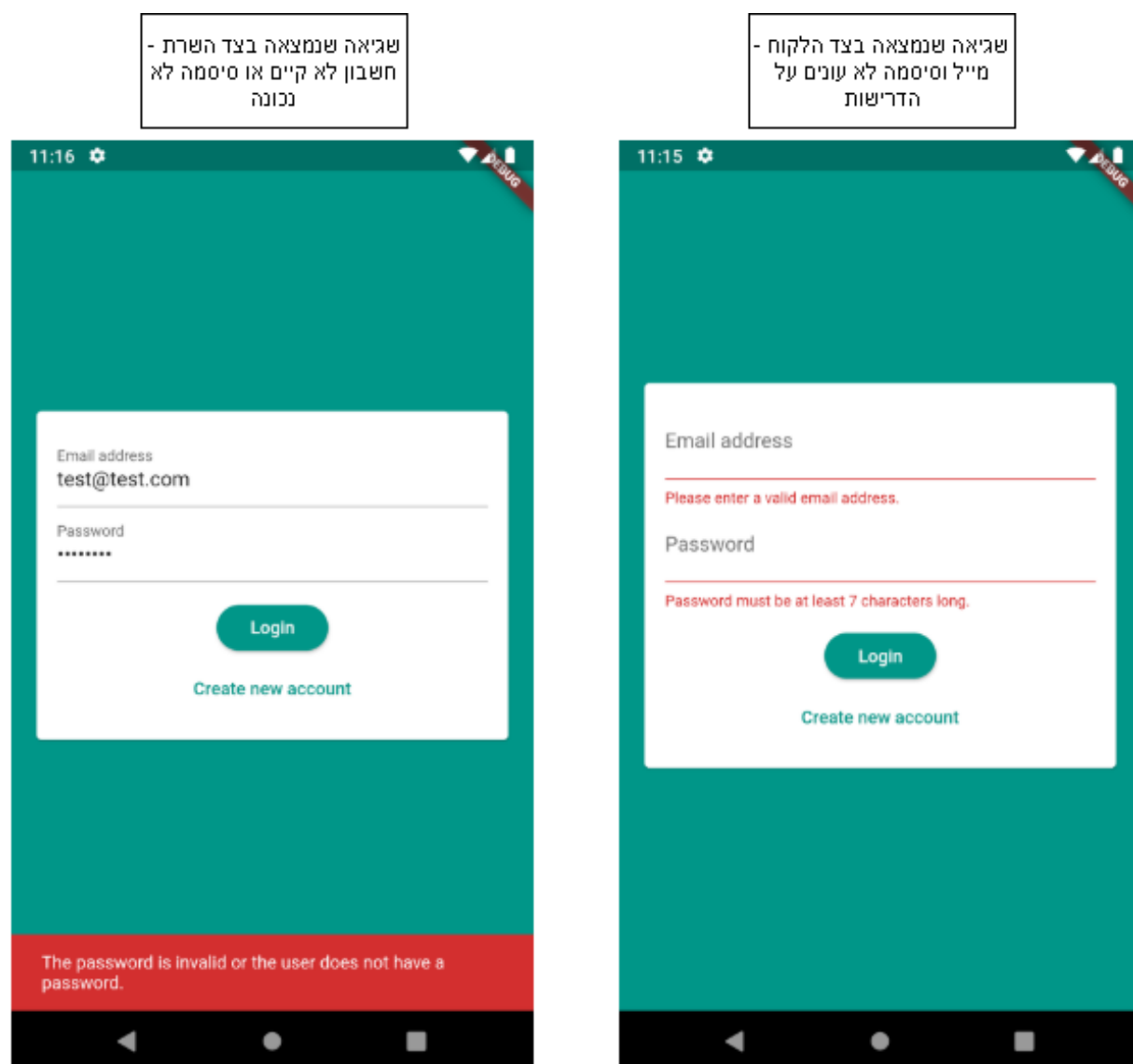
תיאור: המסך הראשון שמוצג למשתמש אם עוד לא השתמש באפליקציה הוא מסך ההתחברות. כל עוד המשתמש עוד מחובר למערכת, הוא לא יראה את המסך הזה בפחיתחת האפליקציה, אלא ישר ילך למסך הצ'אט. לעומת זאת אם התנתק המשתמש מהמערכת, הוא יועבר שוב למסך זה.

קלט: כתובת המייל והסיסמה של המשתמש.

פלט: אם נמצאת שגיאה בתיקוף פרטי המשתמש, מוצגת הודעה על כך, אם אין, המשתמש מועבר למסך הצ'אט.

*בין מסך ההתחברות ומסך ההרשמה ניתן לעבור על ידי לחיצה על הכפתור "Create new account" במסך ההתחברות או "I already have an account" במסך ההרשמה.





מסך ההרשמה

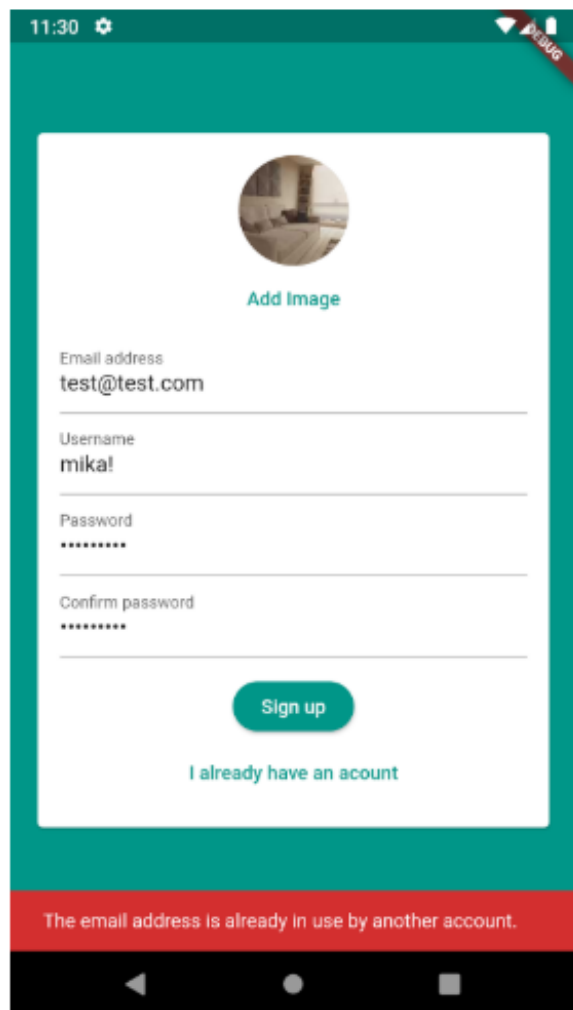
תיאור: למסך זה ניתן להגיע ממסך ההתחברות, כלומר הוא מוצג רק באותם התנאים כמו של מסך ההתחברות. דרך מסך זה המשתמש יכול להירשם למערכת אם אין לו חשבון.

קלט: כתובת מייל (נדרשת בשביל הרשמה דרך firebase וגם מוצגת במסך המשתמשים), שם משתמש, סיסמה, אישור סיסמה, תמונת משתמש.

פלט: אם נמצאה שגיאה בפרטי המשתמש (חסר שדה, כתובת מייל לא קיימת וכדומה) מופיעה הודעה על כך, אם לא פרטי המשתמש נרשמים במערכת והוא מועבר למסך הצ'אט.

*בין מסך ההתחברות ומסך ההרשמה ניתן לעבור על ידי לחיצה על הכפתור "Create new account" במסך ההתחברות או "I already have an account" במסך ההרשמה.

שגיאה שנמצאה בצד השרת -
המשתמש כבר קיים במערכת



11:30

Add Image

Email address
test@test.com

Username
mika!

Password

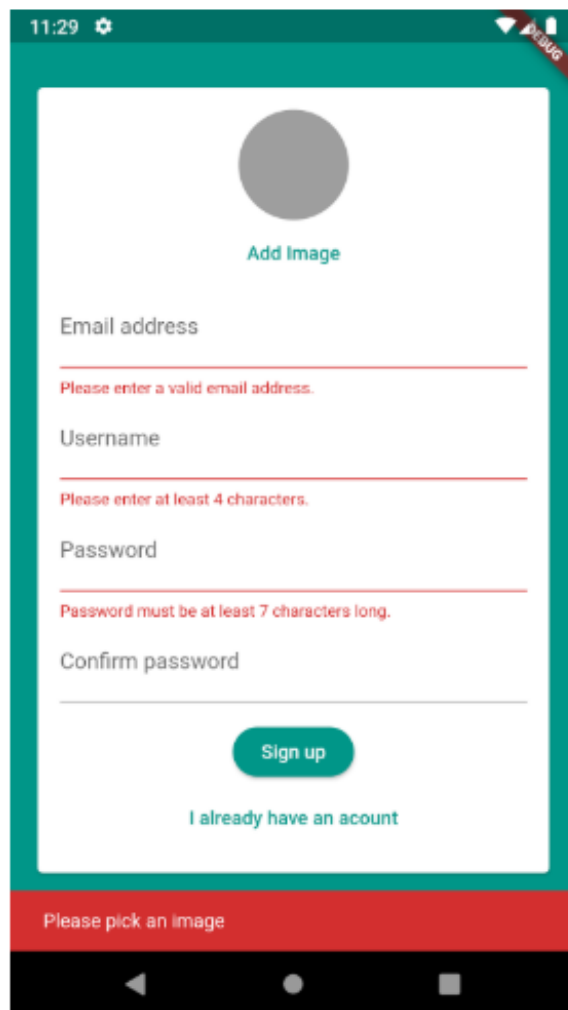
Confirm password

Sign up

I already have an account

The email address is already in use by another account.

שגיאה שנמצאה בצד הלקוח -
חלק מהשדות לא עונים על
הדרישות או ריקים



11:29

Add Image

Email address
Please enter a valid email address.

Username
Please enter at least 4 characters.

Password
Password must be at least 7 characters long.

Confirm password

Sign up

I already have an account

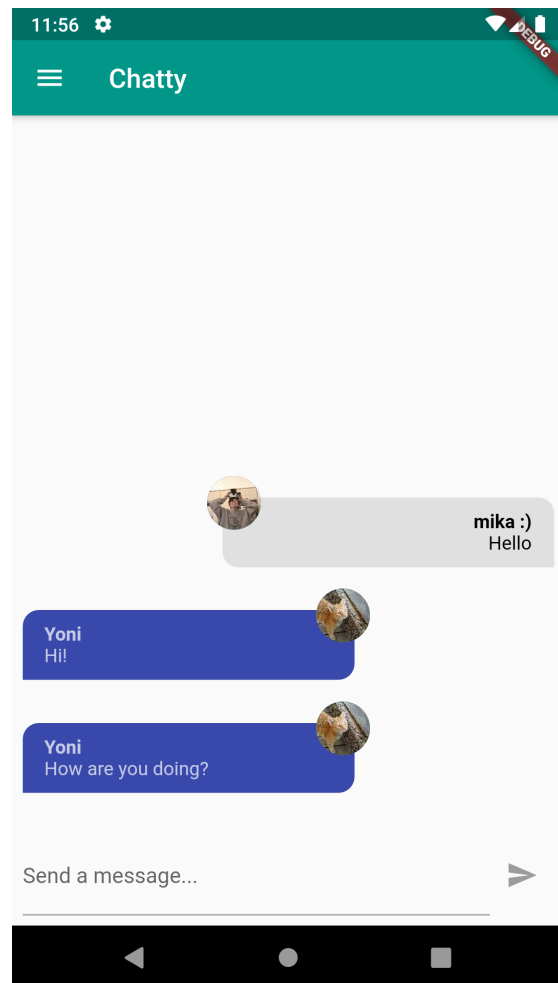
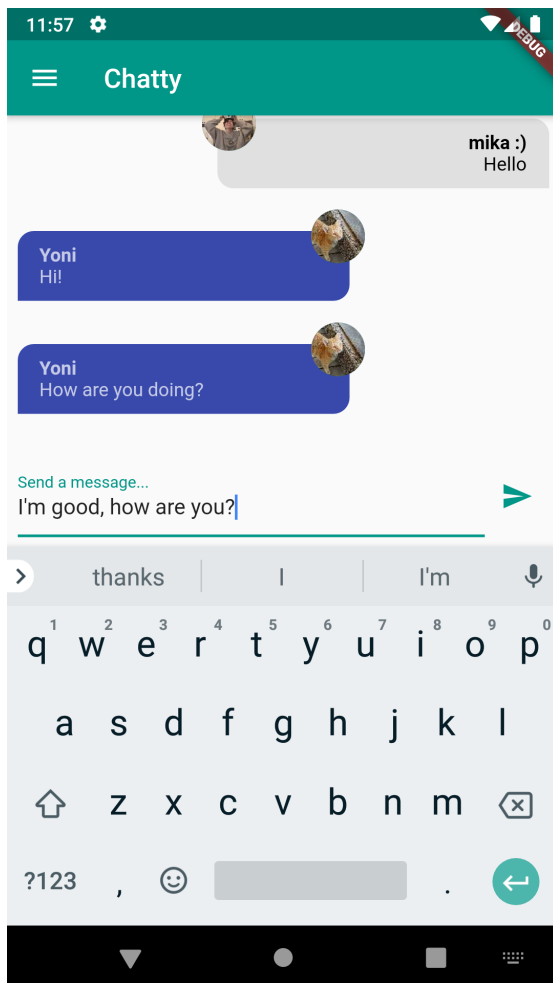
Please pick an image

מסך הצ'אט

תיאור: המסך הראשי, בו ניתן לשלוח הודעות לכל המשתמשים הרשומים ולראות את כל ההודעות הקודמות שנשלחו.

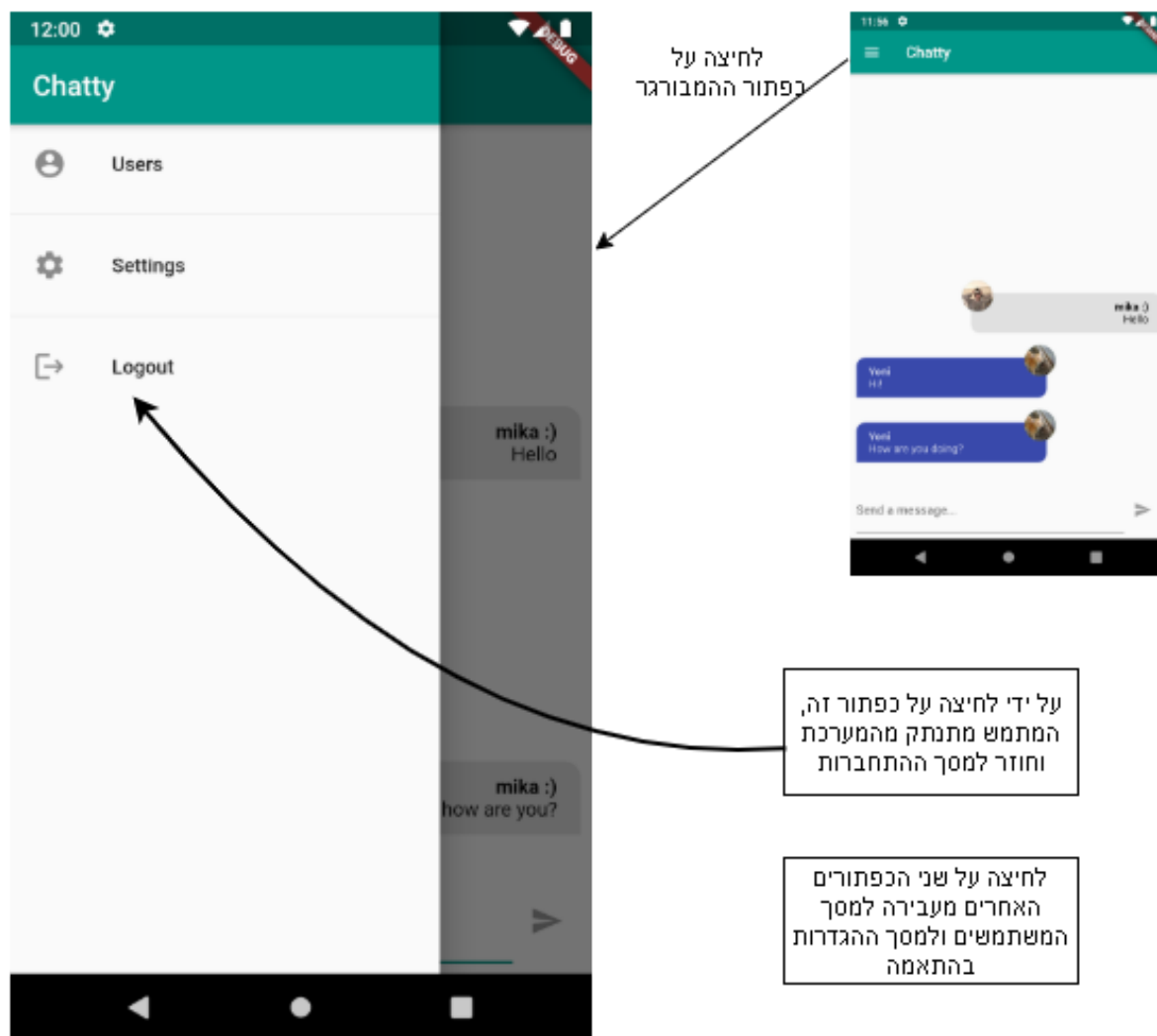
קלט: אם המשתמש מעוניין לשלוח הודעה, טקסט ההודעה.

פלט: כל ההודעות שנשלחו בצ'אט מוצגות כך שהעדכנית ביותר מופיעה בתחתית המסך. לכל הודעה כתוב גוף ההודעה, שם המשתמש של השולח והתמונה שלו. גם הודעה שנשלחה לפני שנוצר המשתמש או בזמן שלא היה מחובר תופיע על המסך.



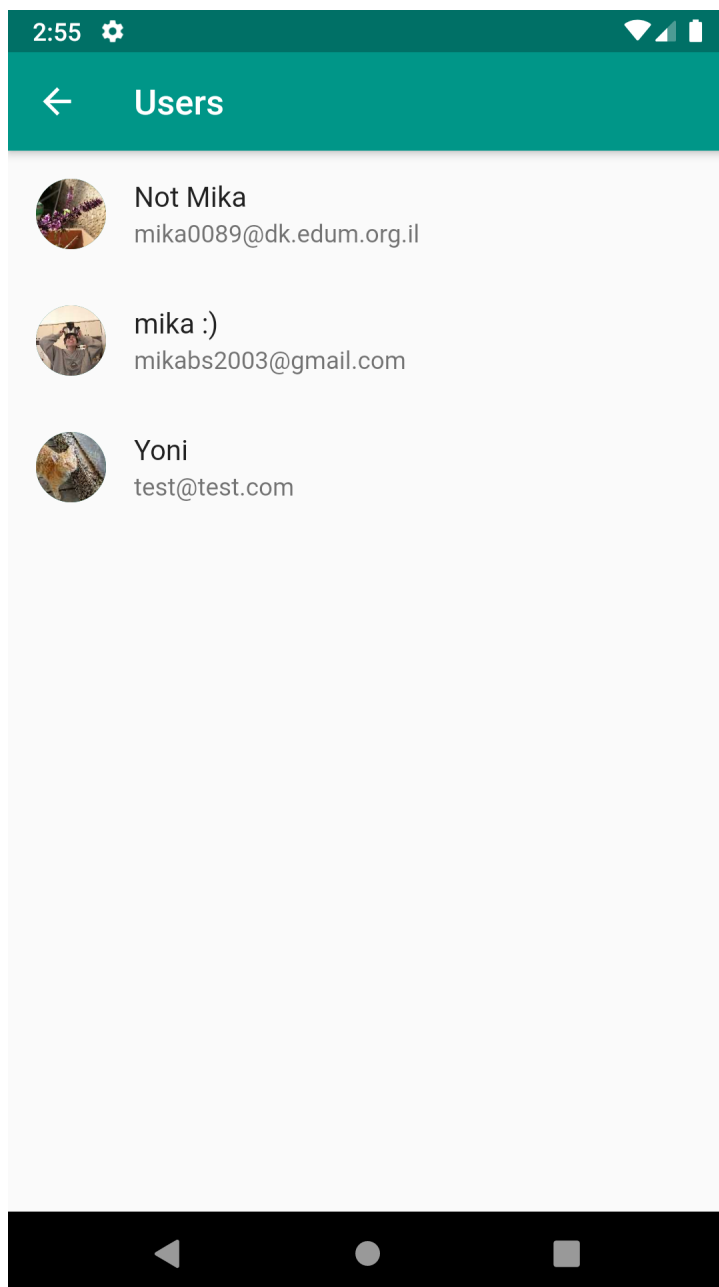
Side drawer

תיאור: מגירה צדדית שמאפשרת גישה לכמה מסכים נוספים באפליקציה. למגירה הצדדית ניתן לגשת דרך הכפתור העליון השמאלי במסך הצ'אט שנראה כמו "המבורגר".



מסך המשתמשים

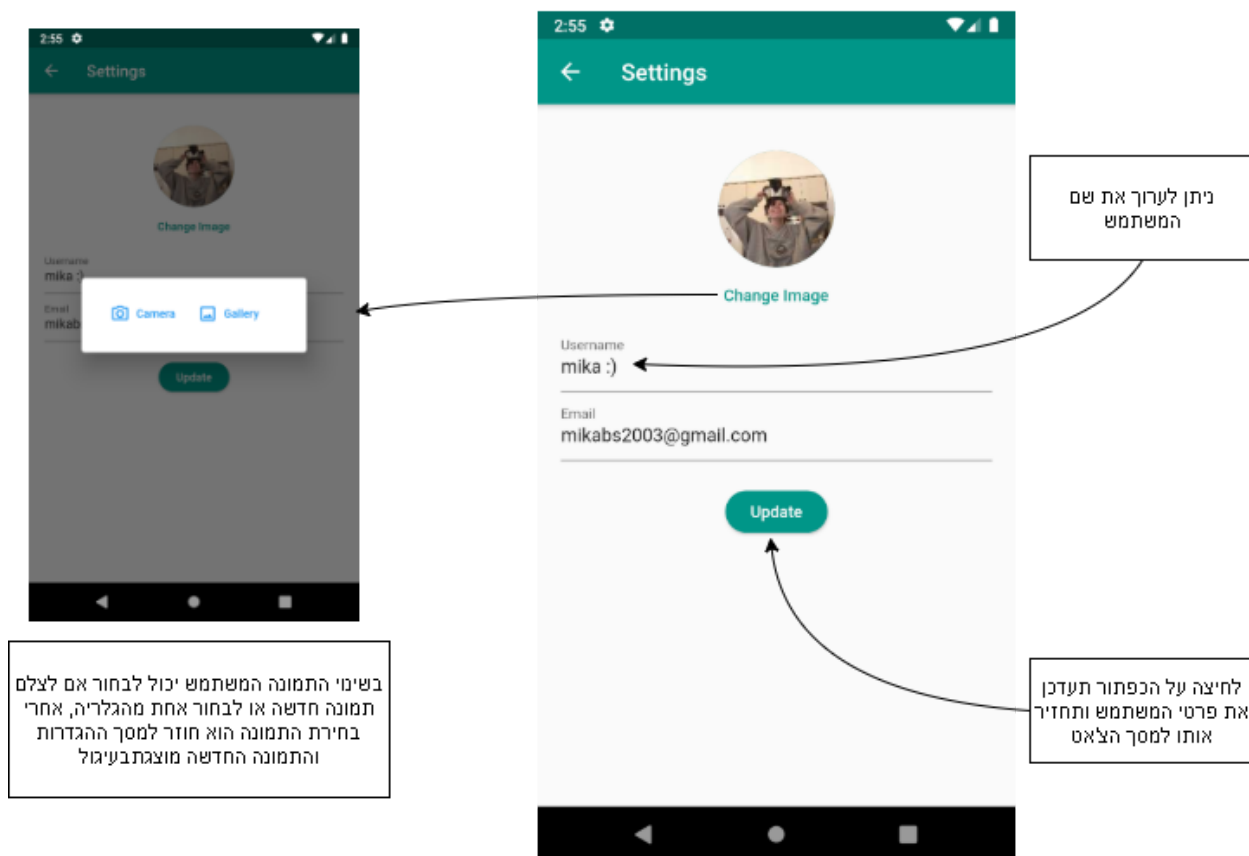
תיאור: במסך זה מוצגת רשימה ל כל המשתמשים הרשומים לאפליקציה, יחד עם התמונה של כל משתמש וכתובת המייל שלו. למסך ניתן להגיע מהמגירה הצדדית.



מסך ההגדרות

תיאור: במסך זה, יכול המשתמש לראות את שם המשתמש, המייל והתמונה שלו ולשנות את שם המשתמש והתמונה שלו אם הוא רוצה בכך. למסך מגיעים דרך המגירה הצדדית, וכאשר לוחצים על כפתור עדכון הפרטים המשתמש מועבר בחזרה למסך הצ'אט.

קלט: שינוי בשדה הטקסט של שם המשתמש ו/או תמונת משתמש חדשה.



בסיס הנתונים

הסבר על Firebase

כפי שכבר הסברתי, Firebase היא פלטפורמה של גוגל לשרתים ולבסיסי נתונים בשביל אפליקציות מובייל ורשת. בשנת 2011 היא הוקמה כחברה עצמאית, וב-2014 היא נקנתה על ידי גוגל.

Firebase מציעה מגוון שירותים, כולל שרת, בסיס נתונים בזמן אמת שנקרא Realtime Database ואחד קצת יותר מתוחכם אך נוח לשימוש שנקרא Firestore Database. היא משמשת כ-Firebase backend להרבה אפליקציות, והיא נוחה מאוד לשימוש, בין השאר הודות לממש המשתמש הנוח של האתר שלה, שדרכו אפשר בקלות לצפות בל המידע המאוחסן בבסיס הנתונים ולראות מידע על התעבורה.

הפלטפורמה מתעדכנת כל הזמן ויש הרבה משאבים באינטרנט ללמידת השימוש בה, ולהתקשרות איתה בפיתוח בכל מני שפות. בין השאר, Firebase מאוד נוחה לשימוש בפיתוח על flutter וישנן הרבה חבילות (packages) וספריות (libraries) להתקשרות איתה דרך אפליקציה הכתובה ב-flutter.

בפרויקט זה השתמשתי בבסיס הנתונים בזמן אמת של Firestore של Firebase, שהוא בסיס נתונים מסוג NoSQL. מעבר לכך השתמשתי גם בפיצ'רים אחרים של Firebase: לאחסון קבצי התמונות של המשתמשים השתמשתי ב-Firebase Storage כך שרק הקישור לכל תמונה נמצא בבסיס הנתונים במסמך של כל משתמש, לניהול המשתמשים באפליקציה השתמשתי ב-Authentication של Firebase שמנהל בעצמו את ההרשמה של משתמשים וגם את חיבורם למערכת. כמובן גם השתמשתי ב-Firebase בתור שרת, כפי שכבר ציינתי. התקשורת עם Firebase היא בזמן אמת כך שכל הזמן מתבצעים עדכונים גם למצב האפליקציה וגם לבסיס הנתונים ולמאפיינים אחרים של הפלטפורמה. את בסיס הנתונים ניתן לנהל גם דרך האתר של Firebase, על ידי כניסה עם החשבון שאליו רשום הפרויקט ב-Firebase.

בסיסי נתונים מסוג NoSQL

רוב בסיס הנתונים הם בעלי מבנה של טבלאות, אך לבסיסי נתונים מסוג NoSQL יש מבנה של "עץ" במקום. כלומר כל הנתונים מסודרים אחד בתוך השני; נתונים בתוך תיקיה בתוך תיקיה וכדומה.

בבסיס הנתונים של Firebase כל תיקיה נקראת אוסף - collection ובתוך כל אוסף ניתן לשמור מסמכים - documents המכילים בהם נתונים בשדות - fields שונים או אוספים נוספים.

בעבודה עם בסיס הנתונים השתמשתי בארבע הפקודות הבסיסיות של בסיסי נתונים - CRUD:

- CREATE - הוספת מידע
- READ - קריאת מידע
- UPDATE - שינוי/עדכון מידע קיים

• DELETE - מחיקת מידע

העבודה עם בסיס הנתונים לא הייתה ישירה, אלא באמצעות חבילות ודרך השרת, לכן למעשה בקוד השתמשתי בפקודות שמפעילות את פונקציות CRUD בבסיס הנתונים.

:CREATE

```
Firestore.instance.collection('collection name').add({Map4 of new data});
```

:READ

```
Firestore.instance.collection('collection name').getDocuments();
```

או

```
Firestore.instance.collection('users').documents('document name').get();
```

:UPDATE

```
Firestore.instance.collection('collection name').document('document name').updateData({Map of new data});
```

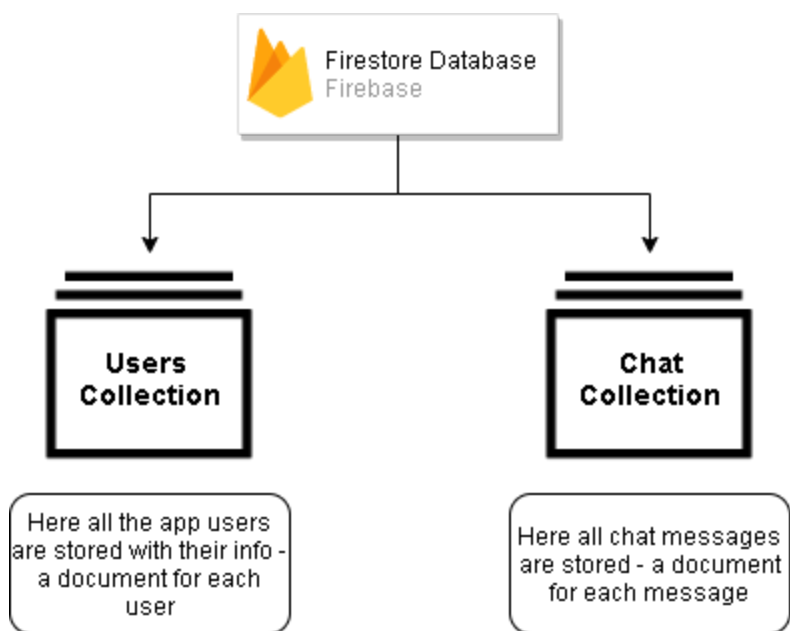
:DELETE

```
Firestore.instance.collection('collection name').document('document name').delete();
```

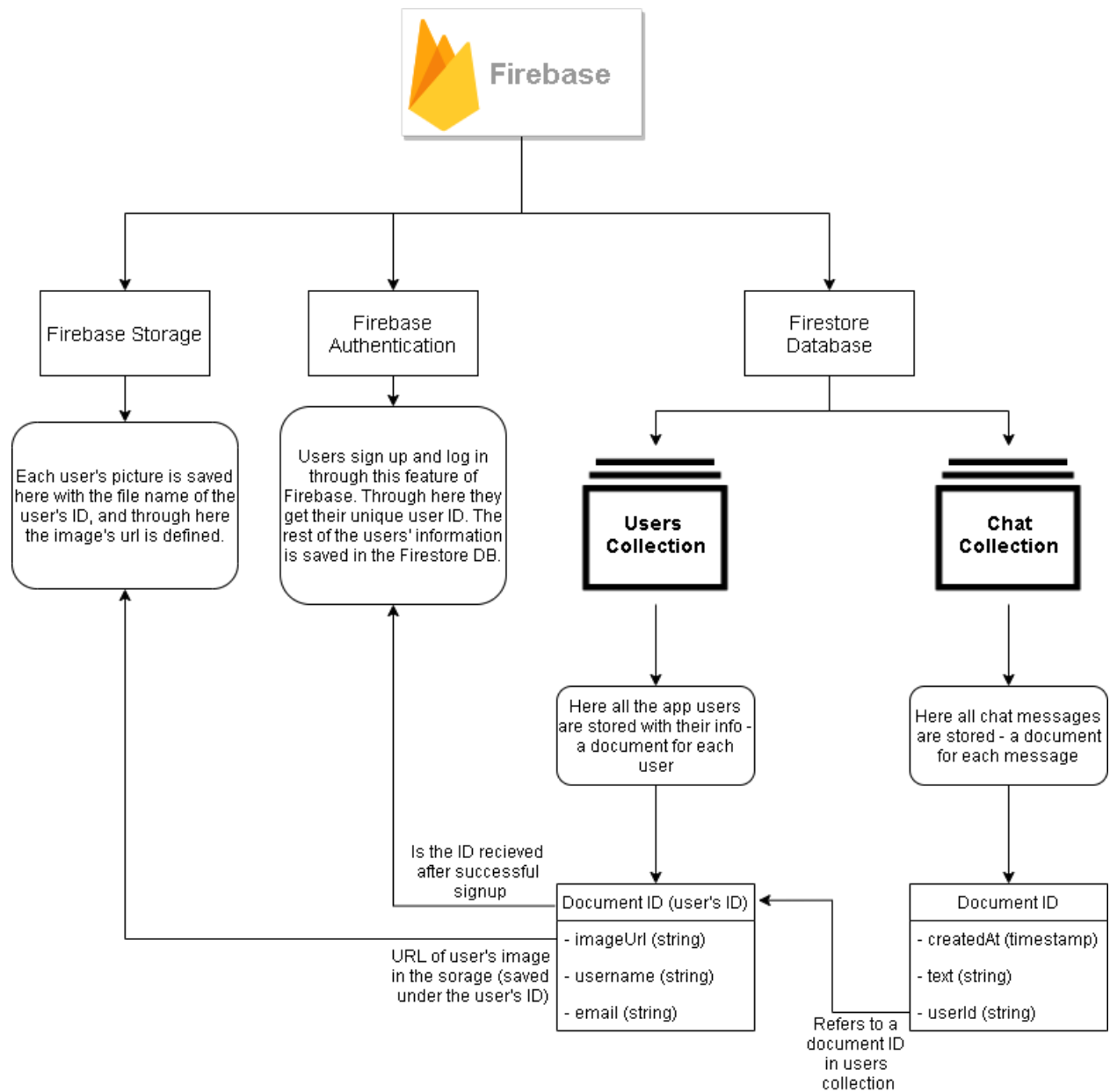
*פקודות אלה מתבצעות בשימוש בחבילה `cloud_firestore`.

⁴ Map (מוכר בשפות תכנות אחרות גם כ-dictionary) הוא מבנה נתונים בעל key ו-value כך שכל key חוזר על עצמו פעם אחת. מבנה נתונים זה דומה מאוד לאיך שפועל בסיס הנתונים של Firebase.

תרשים כללי של בסיס הנתונים בפרויקט



הקשרים בין החלקים השונים ב-Firebase



ניהול בסיס הנתונים

צילומי המסך שמוצגים כאן זמינים אך ורק למנהל של בסיס הנתונים (אני) ודרך האתר של Firebase.

Authentication

בחלק זה מופיעה רשימה של המשתמשים, מידע על הרשמתם והתחברותם ואת הפרטים של כל אחד הקשורים להתחברות למערכת, כלומר: כתובת מייל וה-ID שניתנה לו.

Authentication

Users Sign-in method Templates Usage

Search by email address, phone number or user UID Add user

| Identifier | Providers | Created | Signed in | User UID ↑ |
|-------------------------|-----------|-------------|-------------|-------------------------------|
| mika0089@dk.edum.org.il | ✉ | 31 May 2021 | 27 Jun 2021 | 4GsNSMJSGyOidUn8HpX62sqvHQ... |
| mikabs2003@gmail.com | ✉ | 31 May 2021 | 27 Jun 2021 | 5tdl7Co2R7O9ZvMs3BoX4wmFFAf2 |
| test@test.com | ✉ | 27 Jun 2021 | 27 Jun 2021 | IKW5yZANAKVISpVlieZSZfKXw1B3 |

Rows per page 50 1 - 3 of 3

Storage

כרגע בחלק זה של Firebase נשמרות רק התמונות של המשתמשים (בעתיד אולי אוסיף גם שליחת תמונות בצ'אט או צ'אטים פרטיים עם תמונות משלהם, והן ישמרו כאן). התמונות שמורות בתיקיה `user_images`:

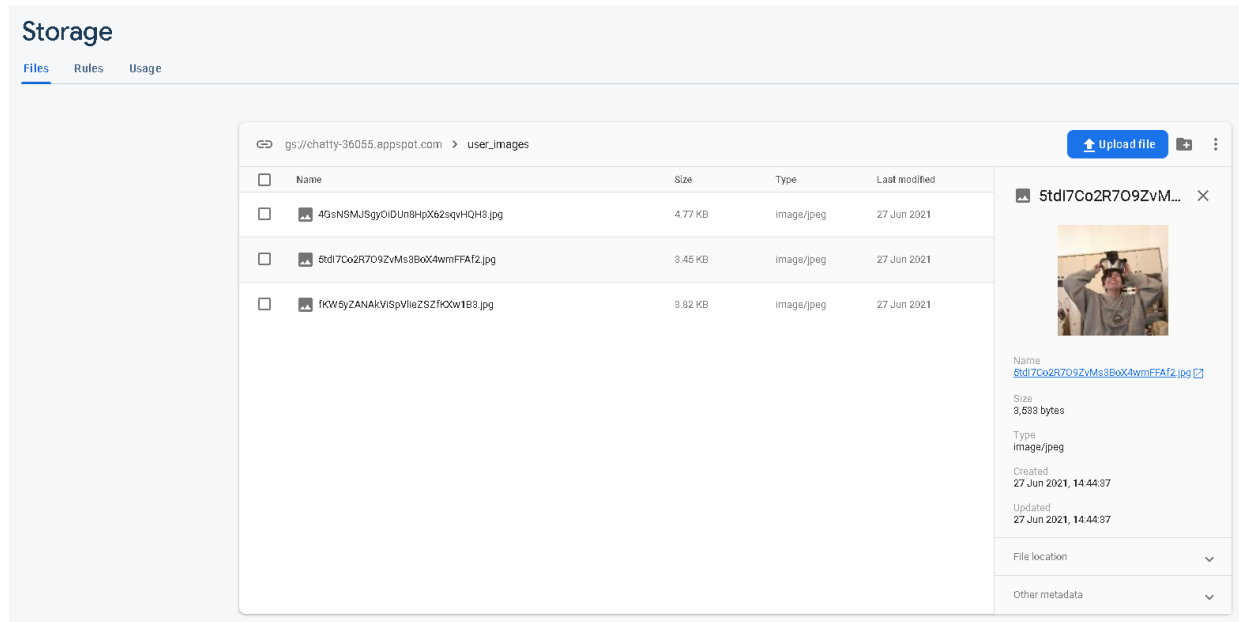
Storage

Files Rules Usage

gs://chatty-36055.appspot.com Upload file

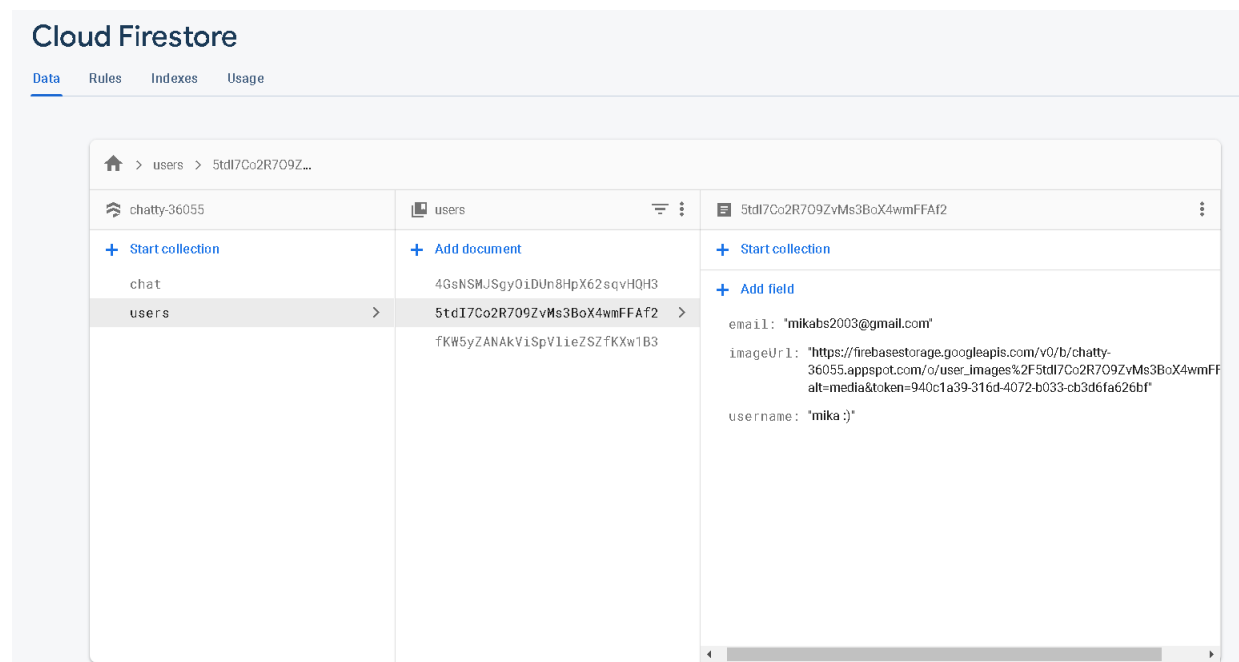
| Name | Size | Type | Last modified |
|--------------|------|--------|---------------|
| user_images/ | — | Folder | — |

השם של כל תמונה הוא לפי ה-ID של המשתמש אליה היא שייכת, כך שניתן למצוא כל תמונה בקלות:



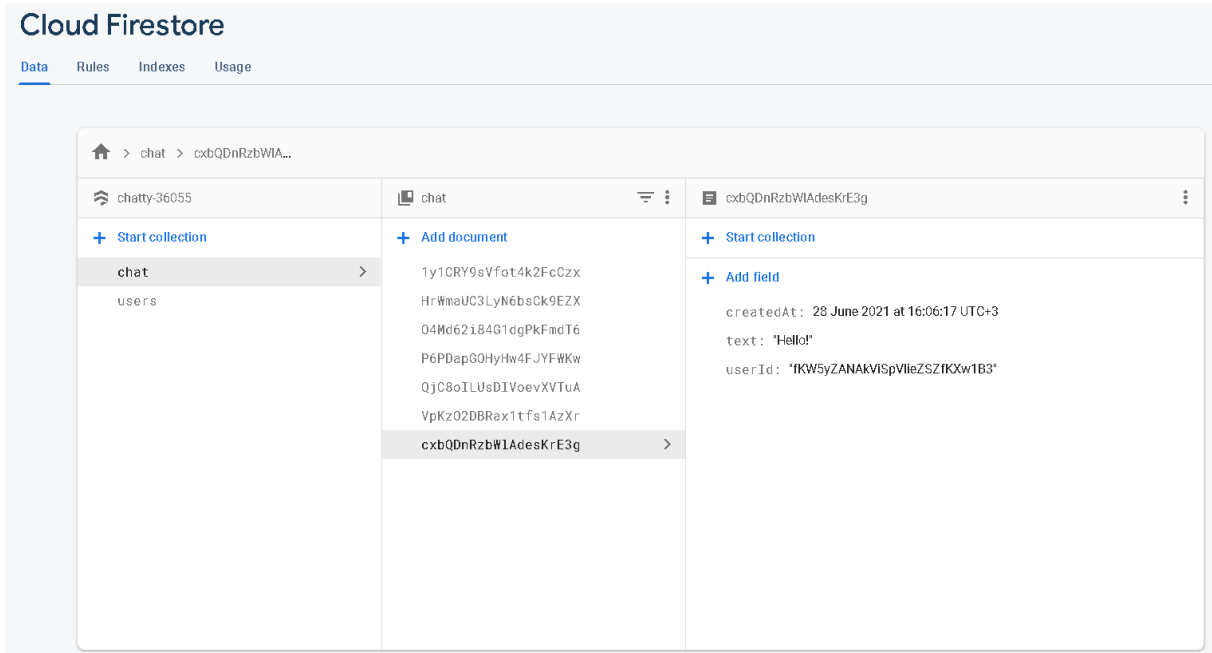
Users collection

לאחר הרשמת המשתמש דרך ה-Firebase Authentication, הפרטים שלו (גם אלה שלא היו דרושים בהרשמה דרך Authentication כמו שם המשתמש) נשמרים בבסיס הנתונים, כדי שהגישה אליהם תהיה קלה.



Chat collection

כל הודעה שנשלחת בצ'אט נשמרת באוסף זה בבסיס הנתונים, יחד עם ה-ID של המשתמש ששלח אותה (בשביל לאפשר הצגה של שם המשתמש והתמונה שלו על המסך) והזמן בו נשלחה.



מדריך למפתח

קוד הפרויקט ב-GitHub

<https://github.com/mb-s/Chatty>

הסבר כללי על קוד הפרויקט

הפרויקט שלי כתוב בשפת dart שהיא השפה איתה עובדת הפלטפורמה לפיתוח אפליקציות flutter. בפרויקט כתבתי רק את צד הלקוח, כלומר את התקשורת עם השרת ועם רכיבים המכשיר של המשתמש ואת ממשק המשתמש, אך לא כתבתי קוד שרת ובמקום השתמשתי בשירותים המגוונים של firebase. בעבודה על הפרויקט השתמשתי בחבילות שעזרו לי בתקשורת עם firebase, בגישה למצלמה ועוד. בבניית אפליקציות בעזרת flutter, הקוד ב-dart משמש בשביל כל המטרות ואין צורך בשפות שונות לתפקידים שונים בפיתוח האפליקציה. כלומר אם בפיתוח אתרים משתמשים בשלוש שפות ראשיות: html לבסיס ממשק המשתמש, css לעיצוב ו-javascript לכל מה שב-backend - בעבודה עם flutter הקוד ב-dart ממלא את כל התפקידים האלה. אותה שפה שבעזרתה אני מחליטה מה יופיע על המסך גם שולחת בקשות לשרת.

Features של המכשיר שבשימוש בפרויקט

בשביל שלכל משתמש תהיה תמונת פרופיל, הוא יכול לבחור תמונה מהגלריה, או לצלם תמונה חדשה. לשם כך הפרויקט שלי עושה שימוש במצלמה של המכשיר וניגש לאחסון התמונות שלו, על כך מפורט בהמשך ("שימוש במצלמת המכשיר וגישה לאחסון התמונות").

OOP - תכנות מונחה עצמים

רוב התכנות ב-flutter תלוי בשימוש באובייקטים שנקראים widgets. בעבודה על הפרויקט גם יצרתי classes חדשות שיורשות מה-widets הבסיסיים בשביל להקל על כתיבת הקוד ולסדר אותו. ב-flutter ישנם שני widgets בסיסיים - stateless ו-stateful, שכל מחלקה אחרת מתבססת עליהם. הרבה widgets מסובכים יותר צריכים לקבל widgets אחרים בתור קלט כדי להיבנות, או שניתן להרכיב בתוכם widgets אחרים.

Remote Procedure Call - הפעלת פרוצדורות מרחוק

בפרויקט שלי השתמשתי ב-firebase בתור שרת, ולא כתבתי שרת משלי. את התקשורת עם השרת עשיתי בעזרת הפרוטוקול (RPC (remote procedure call.

בעזרת RPC, יכול הלקוח להפעיל פעולות אצל השרת כאילו הם נמצאים על אותו מכשיר, ולקבל מהשרת את הפלט של פעולות אלה. כלומר השרת מקבל בקשה להפעלת הפרוצדורה ואת הפרמטרים הדרושים להפעלתה והשרת מפעיל אותה, כמו כאשר קוראים לפונקציה שנמצאת בקובץ אחר על המחשב⁵.

השימוש ב-RPC בפרויקט שלי הוא קריאה לפרוצדורות מוגדרות מראש אצל השרת של firebase. הפונקציות המרכזיות בהן השתמשתי היו מחיקה, כתיבה ועריכה של ערכים ו-documents בבסיס הנתונים.

תכנות אסינכרוני

פעולות כמו שליחת בקשות לשרת דורשות הרבה זמן לביצוע ואי אפשר לדעת מתי ואם תוחזר תגובה, ולכן השתמשתי בתכנות אסינכרוני לביצוען.

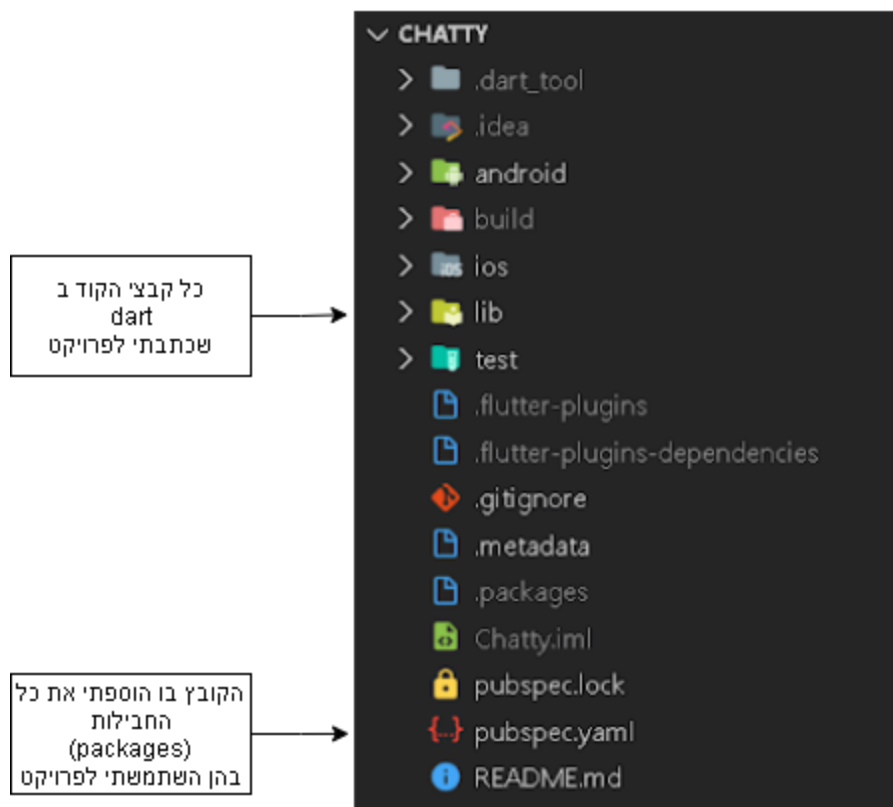
תכנות אסינכרוני הוא כתיבת פקודות וקטעי קוד שמתבצעים במקביל לשאר הקוד שרץ, כך שהם לא יעקבו את ריצת כל הקוד עד שיסיימו לרוץ. ניתן גם לקבל עדכונים על מצב הפעולה - האם היא ממתינה, סיימה לרוץ או נכשלה, מה שמאפשר התאמה של שאר מהלך הריצה לפי מצבה.

ניתן לראות את התכנות האסינכרוני בא לידי ביטוי בפרויקט למשל כאשר ישנו מסך שנטען. כל עוד מחכים להגעתו של מידע מהשרת מופיע סימן טעינה, וכאשר המידע מגיע ומוכן להיות מוצג על המסך סימן הטעינה נעלם ובמקומו מופיע המסך הרצוי⁶.

⁵ ניתן לקרוא עוד על RPC בעמוד הויקיפדיה - https://en.wikipedia.org/wiki/Remote_procedure_call או לראות את הסרטון של Neso Academy בקישור - <https://www.youtube.com/watch?v=QmhTjsOOrlw>

⁶ עוד על תכנות אסינכרוני - [https://en.wikipedia.org/wiki/Asynchrony_\(computer_programming\)](https://en.wikipedia.org/wiki/Asynchrony_(computer_programming))

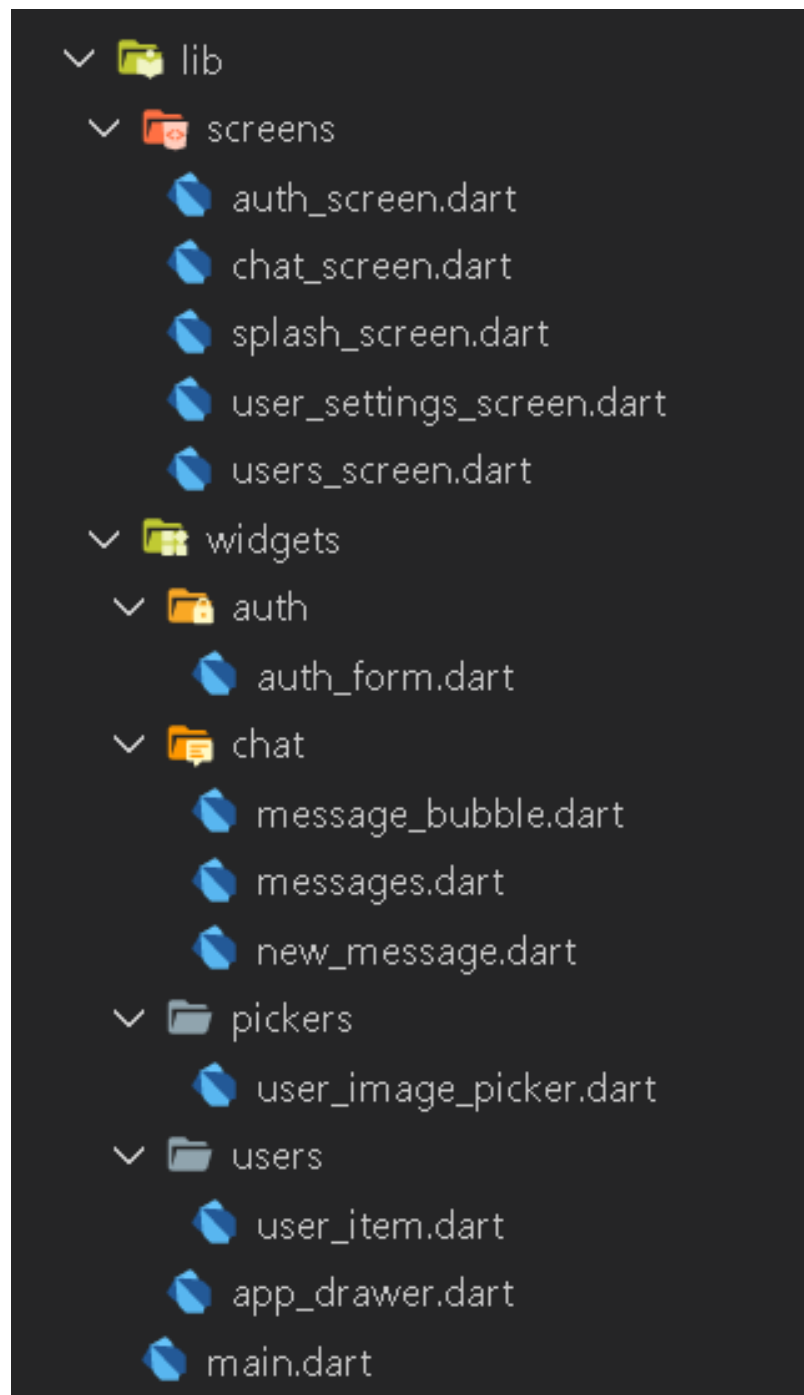
איך מסודרת תיקיית הפרויקט



כל שאר הקבצים קשורים או לניהול הגרסאות של git או לקונפיגורציה של flutter (נוצרו עם יירת הפרויקט על ידי flutter) ולהרצה על מערכות הפעלה שונות.

התיקיה lib

בתיקיה זו נמצא למעשה כל הקוד שכתבתי לפרויקט. הקוד כתוב בשפת dart, והוא קובע את ממש המשתמש, התקשורת עם השרת והתקשורת בין רכיבים של האפליקציה ושל המכשיר של המשתמש. כל הקבצים נמצאים בתיקיות נוספות שממיינות אותם, מלבד קובץ ההרצה של הפרויקט, main.dart.



קבצים, פונקציות וחלקי קוד חשובים

הגדרת החבילות שבשימוש בפרויקט

קובץ: pubspec.yaml

מיקום: lib\pubspec.yaml

בקובץ זה, שנוצר על ידי flutter עם יצירת הפרויקט, ניתן בין השאר להגדיר את החבילות⁷ (packages) בהן "תלוי" הפרויקט, כלומר החבילות בהן יש שימוש בקוד. זה השימוש שאני עשיתי בקובץ.

החבילות רשומות בקובץ תחת dependencies:

```
dependencies:
  flutter:
    sdk: flutter
  cloud_firestore: ^0.13.5
  firebase_auth: ^0.16.1
  image_picker: ^0.6.5+3
  firebase_storage: ^3.1.5
```

שימוש במצלמת המכשיר וגישה לאחסון התמונות

קובץ: user_image_picker.dart

מיקום: lib\pickers\user_image_picker.dart

בקובץ זה מוגדר גם ה-widget של בחירת התמונה (מה שרואים כשמתבקשים לבחור תמונה) וגם ה-logic מאחורי בחירת התמונה. הוא משמש בשביל בחירת התמונה של המשתמש בתהליך ההרשמה וגם כאשר המשתמש מעוניין לעדכן את תמונתו, דרך מסך ההגדרות.

בקובץ יש שתי מחלקות - `UserImagePicker`, `_UserImagePickerState`. כפי שצויין קודם, ישנן שתי מחלקות מהן כל ה-widgets ב-flutter יורשים: `StatelessWidget` ו-`StatefulWidget` כאשר הראשונה היא בשביל widget שלא משתנה על המסך אחרי שבנו אותו והשנייה ביא בשביל אחד שכן. במקרה הזה, מכיוון שאני מעוניינת להציג את התמונה החדשה שנבחרה במקום הקודמת/עיגול ריק, השתמשתי ב-`StatefulWidget`.

⁷ ניתן למצוא כל מני חבילות בקישור - <https://pub.dev/packages>

החלק הלוגי - גישה למצלמה/גלריית התמונות

ה-logic בחלק זה היא בפונקציה `_pickImage()`. הפונקציה אסינכרונית מכיוון שלא ידוע כמה זמן יקח למשתמש לבחור את התמונה, והיא בעיקר מתבססת על החבילה `image_picker` שעושה את רוב העבודה מאחורי הקלעים של גישה למצלמה ולאחסון התמונות.



ממשק המשתמש

חלק זה קובע מה יוצג על המסך, הוא נמצא בפעולה `build()`⁸ של המחלקה `_UserImagePickerState` ובפונקציית ה-`void` באותה מחלקה `_pickSource()`.

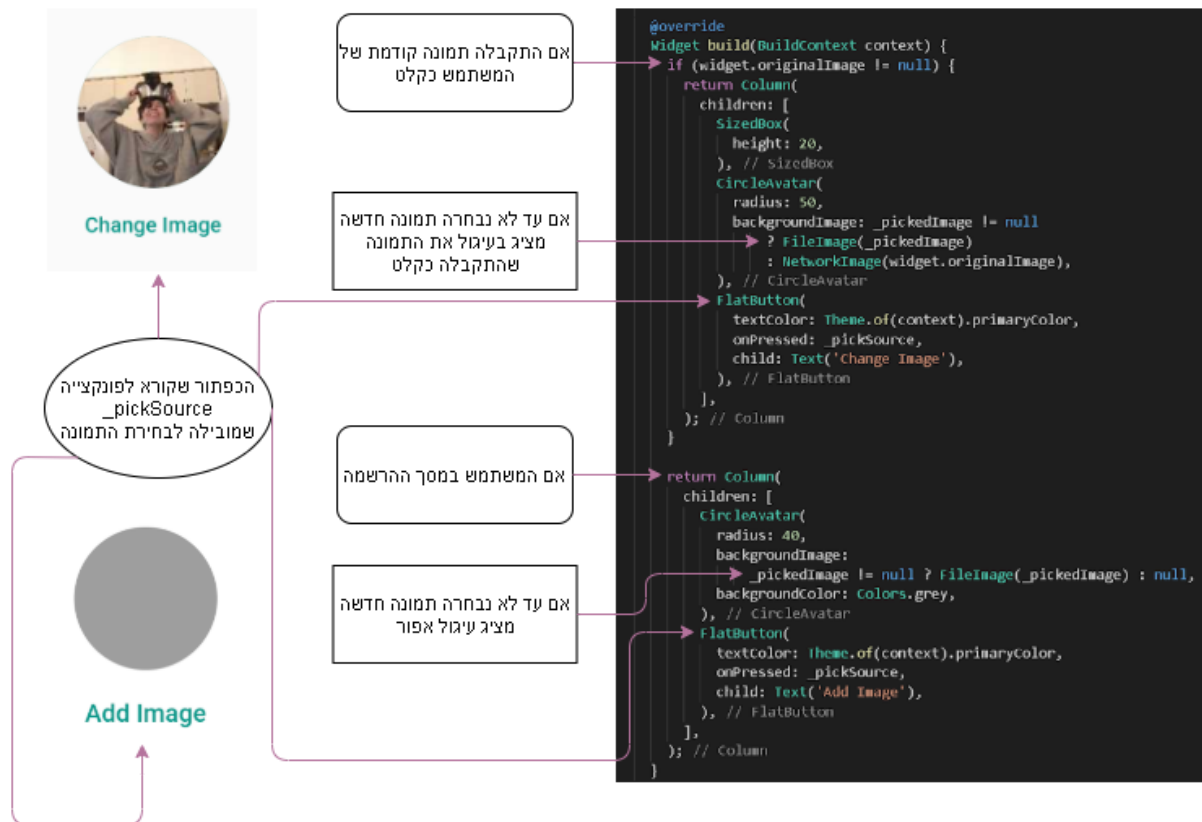
ממשק המשתמש מורכב כך שיבנה widget שונה אם המשתמש מחובר ורוצה לשנות את תמונתו (תופיע תמונתו המקורית, לפני שבחר תמונה חדשה) לזה שיבנה אם המשתמש בתהליך ההרשמה (יופיע עיגול ריק, לפני שבחר תמונה חדשה). מתחת לתמונה, מופיע כפתור שעל ידי לחיצה עליו מתבקש המשתמש לבחור מאיפה ירצה לבחור את התמונה החדשה (ממומש בפונקציה `_pickSource`), ואחרי בחירתו, יועבר המשתמש בהתאם למסך לבחירת/צילום התמונה.

widgets בהם השתמשתי:

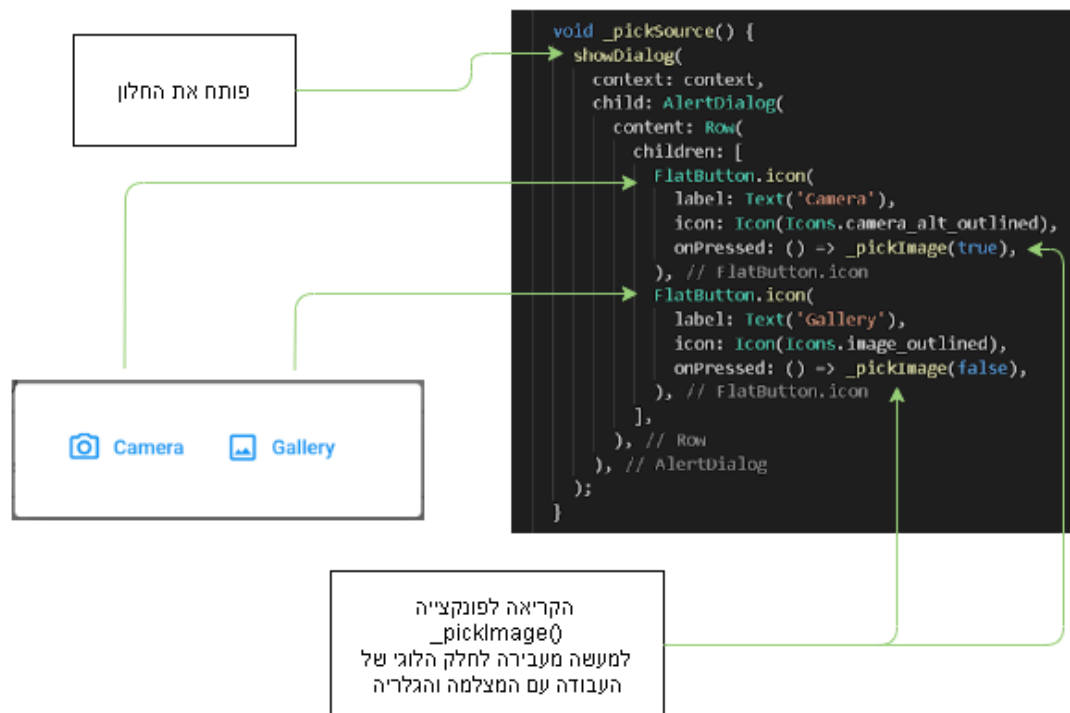
- Column - מאפשר לשים כמה widgets (שמוגדרים ברשימה `children[]`) בטור

⁸ פעולת build היא פעולה של כל מחלקה ב-flutter שמחזירה למעשה את מה שיבנה לבסוף על המסך.

- **SizedBox** - קופסה ריקה שניתן לקבוע את גודלה, משמשת פה בשביל יצירת מרווח בין widgets אחרים
- **CircleAvatar** - עיגול בו ניתן להגדיל צבע או תמונת רקע
- **FileImage** - מציג תמונה מסוג File
- **NetworkImage** - מציג תמונה מלינק
- **FlatButton** - כפתור לא בולט, מגדירים לו פונקצייה `onPressed` שמופעלת כאשר הוא נלחץ
- **Icon** - מציג אייקון על המסך
- **Text** - ממיר טקסט ל-widget שניתן לעצב אותו, כלומר לשנות את הצבע שלו, הגודל שלו ועוד
- **AlertDialog** - חלון שמופיע על המסך (אך לא מכסה אותו לגמרי) עם פעולות שונות



הפונקציה `_pickSource`:



תקשורת עם השרת ועם בסיס הנתונים

התקשורת עם השרת, ודרכו עם בסיס הנתונים, משולבת ברוב חלקי הקוד שכתבתי לפרויקט. כלומר בהרבה פונקציות, מחלקות וקבצים בפרויקט אני שולחת בקשות לשרת של Firebase. לכן, אציג פה את כל הפונקציות בהן השתמשתי בשביל תקשורת זו, ואדגים את השימוש בחלק מהן בקוד.

להתקשרות עם השרת ועם בסיס הנתונים השתמשתי בכמה חבילות.

firebase_auth

חבילה זו שימשה אותי בפרויקט בהרשמה, בחיבור ובניתוק של המשתמש מהמערכת. גם השתמשתי בה בשביל לקבל את ה-ID של המשתמש אחרי שכבר התחבר.

קבצים בהם יש שימוש בחבילה: `auth_screen.dart`, `user_settings_screen.dart`, `messages.dart`, `new_message.dart`, `app_drawer.dart`, `main.dart`

פונקציות:

התחברות למערכת-

```
FirebaseAuth.instance.signInWithEmailAndPassword(email: [user's email], password: [user's password]);
```

הרשמה למערכת-

```
FirebaseAuth.instance.createUserWithEmailAndPassword(email: [user's email],
password: [user's password]);
```

גישה ל-ID של המשתמש הנוכחי-

```
FirebaseAuth.instance.currentUser().uid;
```

בדיקה אם המשתמש מחובר-

```
FirebaseAuth.instance.onAuthStateChanged();
```

התנתקות מהמערכת-

```
FirebaseAuth.instance.signOut();
```

ברוב הקבצים, כמו ב-`new_message.dart`, החבילה משמשת בשביל לקבל מידע על המשתמש המחובר, ספציפית ה-ID שלו (בקובץ `main.dart` נבדק האם המשתמש מחובר למערכת):

מקבל מהחבילה
firebase_auth
מידע על המשתמש המחובר

כותב מידע בבסיס הנתונים
בעזרת החבילה
cloud_firestore

מידע שהתקבל על המשתמש,
ספציפית ה-ID

```
void _sendMessage() async {
  FocusScope.of(context).unfocus();
  final user = await FirebaseAuth.instance.currentUser();

  Firestore.instance.collection('chat').add({
    'text': _enteredMessage,
    'createdAt': Timestamp.now(),
    'userId': user.uid,
  });
  _controller.clear();
  setState(() {
    _enteredMessage = '';
  });
}
```

*זוהי פונקציית void שנמצאת בקובץ `lib/widgets/chat/new_message.dart`. לקובץ שתי מחלקות הקשורות אחת לשנייה: `NewMessage` ו-`_NewMessageState` הוא אחראי על תיבת שליחת ההודעה במסך הצ'אט ועל שליחת ההודעה (חלק זה הוא מה שרואים בפונקציה).

בקבצים `app_drawer.dart` ו-`auth_screen.dart` החבילה משמשת בשביל לנתק את המשתמש ולחבר/להרשים אותו בהתאמה:



*קטע הקוד הראשון הוא חלק מ-widget שנבנה בפונקציה build במחלקה היחידה בקובץ lib/widgets/app_drawer.

*קטע הקוד השני הוא חלק מהפונקציה _submitAuthForm שאחראית על ניסיון ההתחברות/הרשמה של המשתמש מול השרת בקובץ lib/screens/auth_screen.dart, בו יש שתי מחלקות: AuthScreen ו- _AuthScreenState.

cloud firestore

חבילה זו שימשה אותי כדי לקרוא ולשנות חלקים מבסיס הנתונים של Firebase שנקרא Firestore.

קבצים בהם יש שימוש בחבילה: auth_screen.dart, user_settings_screen.dart, users_screen.dart, messages.dart, new_message.dart.

השימוש בחבילה מפורט בפרק "בסיס הנתונים" ולכן לא אפרט אותו שוב, אלא רק אראה את אחד מהשימושים בה - עדכון שם המשתמש (שליחת הודעה בעזרת החבילה מופיעה בדוגמה של קבלת ה-ID של משתמש בעזרת החבילה firebase_auth).

```

await Firestore.instance.collection('users').document(id).updateData(
  {
    'username': _userNameController.text.trim(),
  },
);

```

*קטע קוד זה הוא חלק מהפונקציה `_updateUserDetails` שמטרתה לבדוק את פרטיו החדשים של המשתמש (כאשר ביקש לעדכן אותם במסך ההגדרות) ולעדכן אותם בבסיס הנתונים אם אפשר. פונקציה זו נמצאת בקובץ `lib/screens/user_settings_screen.dart` שלו יש רק מחלקה אחת - `UserSettingsScreen`.

firebase storage

חבילה זו שימשה אותי בפרויקט בשביל לגשת לתמונות של המשתמשים שנשמרו באחסון של Firebase, לעדכן אותן ולקבל את ה-URL של כל אחת מהן.

קבצים בהם יש שמוש בחבילה: `auth_screen.dart`, `user_settings_screen.dart`:
פונקציות:

שמירת תמונה-

```
final ref = await FirebaseStorage.instance.ref().child('folder name').child('new image name' + '.jpg');
ref.putFile([image in File type]);
```

קבלת ה-URL של תמונה-

```
final ref = await FirebaseStorage.instance.ref().child('folder name').child('image name' + '.jpg');
ref.getDownloadURL();
```

```
final ref = FirebaseStorage.instance
  .ref()
  .child('user_images')
  .child(authResult.user.uid + '.jpg');

await ref.putFile(image).onComplete;

final url = await ref.getDownloadURL();
```

*קטע קוד זה חלק מהפונקציה `_submitAuthForm` שאחראית על ניסיון ההתחברות/הרשמה של המשתמש מול השרת בקובץ `lib/screens/auth_screen.dart`, בו יש שתי מחלקות: `AuthScreen` ו-`_AuthScreenState`.

רפלקציה

למרות הקשיים הרבים בעבודה על הפרויקט, בין אם בלמידה של שפה חדשה לגמרי, התמודדות בפעם הראשונה עם פיתוח אפליקציות או כל מכשול אחר שבו נתקלתי בדרך, העבודה עליו הייתה מרתקת ביותר ולימדה אותי הרבה.

כלים שרכשתי בדרך

מכיוון שבחרתי להתנסות בתחום שאני לא מכירה כלל, רכשתי הרבה מאוד כלים תוך מחקר ועבודה עליו. למדתי שפה חדשה - dart, ואיך להשתמש בה לפיתוח אפליקציות בעזרת הפלטפורמה flutter. בכך גם שיפרתי את יכולות הלמידה העצמאית שלי, למדתי קצת על איך בנויות אפליקציות מובייל ועל השוני שלהן במבנה ובדרישות מקוד למחשב ועל העבודה עם תכנות מונחה עצמים. מעבר לכך, בשביל הפרויקט שלי למדתי לעבוד עם בסיס הנתונים Firebase, באחסון קבצים, הרשמת משתמשים ושמירת מחיק ושינוי ערכים באוספים של בסיס הנתונים.

את כל הידע שרכשתי אוכל לקחת איתי בהמשך וליישם בכתיבת קוד, כמו למשל בכתיבת קוד מונחה עצמים, בשימוש בשפה החדשה שלמדתי, או בעבודה עם Firebase.

מעבר לידע היישומי הרב שרכשתי, גם גיליתי על עצמי מהעבודה על הפרויקט. גיליתי שאני מסוגלת ללמוד שפה שלמה בעצמי, בלי ליווי של מורה, ולמדתי לאמוד נכון יותר את יכולותי ולהגביל את עצמי בשאיפותי לפי המגבלות שעומדות בפני (במקרה זה מחסור בידע בהתחלה ולחץ של זמן).

קשיים ואתגרים שעמדו בפני

בתחילת העבודה על הפרויקט עבדתי על אפליקציה אחרת, שרק אחרי כמה זמן הבנתי שאינה מתאימה לי יחסית להיכרות שלי עם עבודה עם אודיו (שהייתה בבסיסה של האפליקציה), לניסיון ולידע שיש לי בפיתוח אפליקציות. מהעבודה על הרעיון הראשון כבר הייתי מיואשת והתקשיתי למצוא מוטיבציה מספקת לעבודה על הפרויקט ולהתקדמות. כל פעם שנתקלתי בבעייה כלשהי שלא הצלחתי לפתור רציתי לוותר.

למזלי התעקשתי מספיק והתקדמתי מספיק כדי להגיע לרמה של היכרות מספיק טובה עם השפה והפלטפורמה בשביל לשמור על המוטיבציה וההתלהבות שלי מהפרויקט ולהתקדם איתו.

מסקנות מהפרויקט

המסקנה המרכזית שלי מהעבודה על הפרויקט היא ששווה להתעקש לעבוד קשה, גם אם בדרך קשה ומייאש מאוד, כי בסוף יוצאים עם הרבה מאוד. בסופו של דבר אני מאוד נהנית מהעבודה על הפרויקט ואני שמחה שהצלחתי להשלים כזה פרויקט גדול ומאתגר.

מה הייתי עושה אחרת לו הייתי מתחילה היום

הייתי מתחילה את הלמידה מוקדם יותר, ומתעקשת להכיר טוב את הבסיס לפני שאני קופצת לעבודה על הפרויקט עצמו. הלמידה עצמה לא הייתה קשה והיתה אפילו מהנה, ואחת הסיבות שהתקשיתי בעבודה על הפרויקט היא כי נתקלתי בבעיות שעוד לא היו לי הכלים לפתור ואפילו לא הכרתי מספיק בשביל להבין איך למצוא פיתרון באינטרנט.

אני חושבת שאם הייתי מתחילה מוקדם יותר, עוסקת קודם בסיום הלמידה הבסיסית וגם משקיעה יותר זמן בה הייתי יכולה אפילו לעבוד על אפליקציה מקורית לגמרי משלי וגם להפוך אותה למיוחדת ומושקעת יותר.

מה היה יכול להפוך את עבודתי ליעילה יותר

אם הייתי מבקשת יותר עזרה ועובדת יותר עם אחרים אני חושבת שהייתי מבינה מהר יותר חלק מהרעיונות שלמדתי ומצליחה לפתור בעיות מהר יותר. על אותו עיקרון אני חושבת שאם הייתי מדווחת לניר, המורה שלי, יותר בתכיפות על מצבי בתחילת העבודה אולי הוא היה עוזר לי להבין מהר יותר שעלי למקד יותר את המטרות שלי וללכת על משהו מוכר לפני שאני מתפזרת לכיוונים חדשים.

תכונות שהייתי רוצה להוסיף לפרויקט

הייתי רוצה להוסיף לאפליקציה שכתבתי יותר מאפיינים סטנדרטים של אפליקציית צ'אט, כמו צ'אטים פרטיים, שליחת תמונות וקבצים אחרים, אישור שההודעה נקראה, הודעות קוליות, וקבוצות בהן נמצאים רק חלק מהמשתמשים. בנוסף הייתי רוצה להוסיף לה מאפיינים ייחודיים: כמו מצב פשוט יותר שמותאם לילדים או למי שמתקשה עם טכנולוגיה ורוצה להשתמש באפליקציה עם פחות אפשרויות וכפתורים (כלומר מסך נקי יותר), ובחירת פלטת הצבעים של האפליקציה ממספר אפשרויות.

ביבליוגרפיה

Schwarz Müller, M. (2021). Flutter & Dart - The Complete Guide [2021 Edition].

Retrieved from

<https://www.udemy.com/course/learn-flutter-dart-to-build-ios-android-apps/>

מקורות כלליים

DartPad: <https://dartpad.dev/>

Firebase: <https://firebase.google.com/>

Flutter.dev documentation: <https://flutter.dev/docs>

GitHub: <https://github.com/>

StackOverflow: <https://stackoverflow.com/>

YouTube - <https://www.youtube.com/>