helpi

FOR A BETTER WORLD

קישור למצגת

חברי הצוות

שי גאלי – 315202242

בר יחזקאל – 314990938

אמרי שי – 213023500

עידו מנגדי – 315310250

GITHUB PROJECT

חלוקת תפקידים

חלוקת התפקידים היית בעיקר במפגשי זום. העלנו ביחד את הדרישות הבאות וחילקנו משימות. לכל חבר צוות היה את הbranch שלו, וכאשר הוא סיים כל שלב הוא מירגג' אותו עם המאסטר.
כל חבר צוות לקח חלק ברוב חלקי הקוד, אבל היו כאלה שהעדיפו להתמקד בנושאים ספציפים.

- שי: עבד ברוב חלקי הפרויקט – גם הלוגיקה ועבודה מול הfirebase וגם בעיצוב והUI.
- בר: frontend, בנוסף מסכי Home,login, register, עיצוב חלק גדול מהאפליקציה, UI אינטואיטיבי ונגיש.
- אמרי: בעיקר backend, תהליכי אדמינים, dashboard, reports, תהליך חיפוש פוסטים, תהליך עדכון פרטי משתמש.
- עידו: בעיקר backend, התשתית לעבודה במודל ה– MVVM, תהליך פרסום מודעה, צ'אט ו– push notifications.

GITHUB INS


ShayGali
#1
123 commits 21,796 ++ 17,106 --
ImriShai
#2
29 commits 4,300 ++ 939 --
IdoMangadi
#3
24 commits 15,099 ++ 2,206 --
BarYechezkel
#4
21 commits 4,065 ++ 1,232 --
20
15
10
5
0
Contributions
Dec '24
Jan '25
Feb '25

שפת התכנות

בחרנו להשתמש בפלטפורמת אנדרואיד סטודיו,
כאשר שפת התכנות היא JAVA

DB

בחרנו להשתמש בשיטה של Serverless, כאשר
השרת שאיתו אנחנו מדברים הוא firebase.
הDB העיקרי שלנו הוא firestore אבל
השתמשנו בעוד כלים כמו: realtime DB עבור
ההודעות, Cloud Storage בשביל אחסון
תמונות.

Background

בתחילת המלחמה, כאשר עלה צורך בציבור לעזרה במתן ציוד ושירות מיידיים, הפלטפורמות הקיימות של מסירת ציוד לא נתנו מענה מספק. עמותות וארגונים שרצו לסייע בזמן אמת נדרשו לעשות זאת דרך פלטפורמות לא ייעודיות ומסורבלות כמו קבוצות ווטסאפ. מצד שני, גם הזקוקים לעזרה (חיילים/ מפונים וכו') לא מצאו דרך לעשות את הדברים בצורה יעילה או לפחות בפלטפורמה אחת מרכזית שמותאמת לצורך הנ"ל.

המצב הזה הוביל אותנו לרצות לפתח מערכת ייעודית שהמטרה שלה היא לתת למציעי/מבקשי מוצרים/שירותים אפשרות למצוא אחד את השני ולתקשר ביניהם בצורה מיידית ואופטימלית

דרישות

דרישות פונקציונליות

- רישום משתמש ואימות.
- רשימת מוצרים/שירותים.
- חיפוש עם מסננים מותאמים אישית.
- מערכת התראות בזמן אמת.
- פונקציית צ'אט משולבת.
- כלי ניהול למנהלים.
- התאמה מבוססת מיקום.

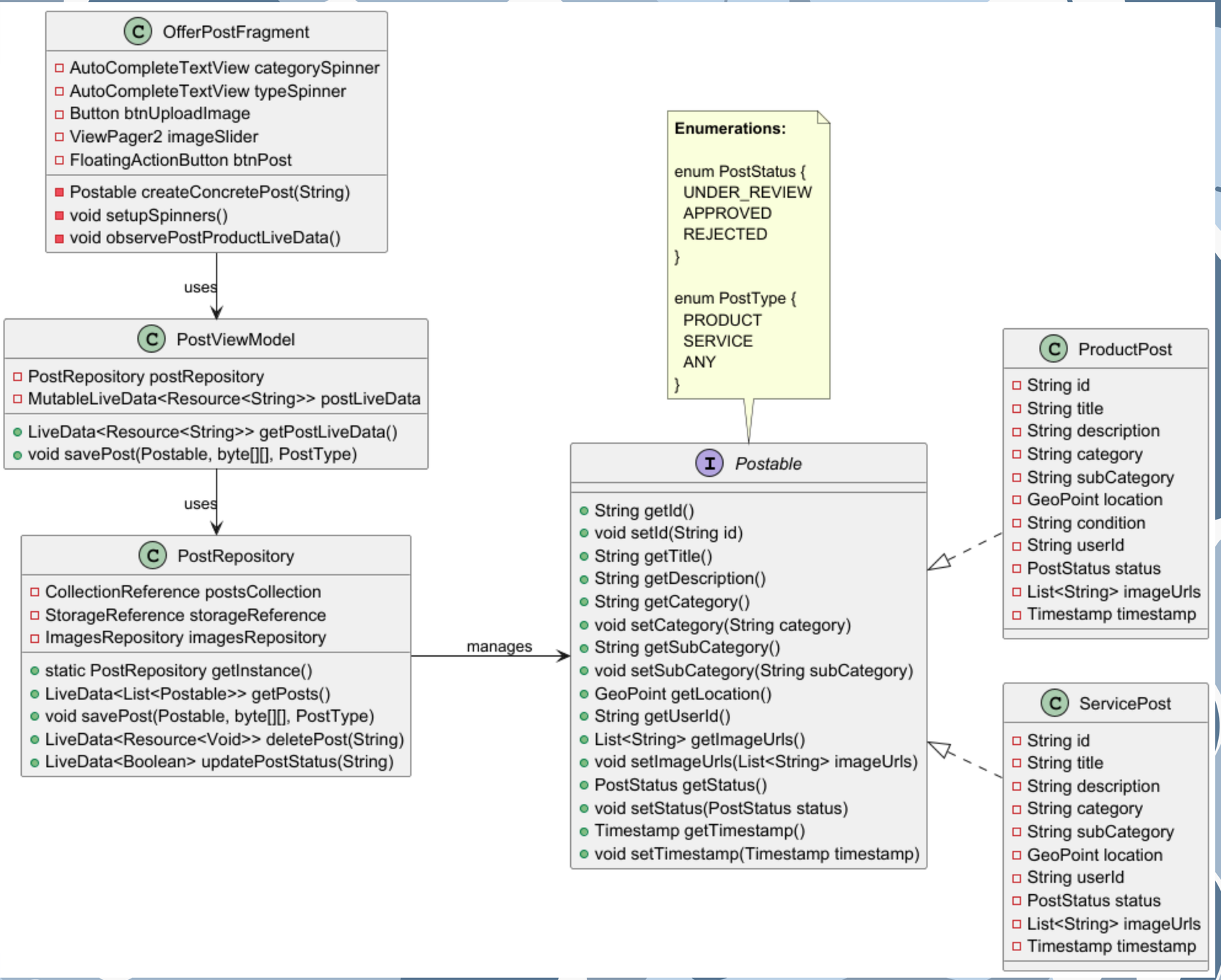
דרישות לא פונקציונליות

- ביצועים:
  - תמיכה במספר משתמשים במקביל.
  - עיבוד נתונים יעיל.
- יכולת גידול:
  - יכולת להתרחב אופקית.
  - תמיכה בבסיס משתמשים גדל.
  - תשתית ענן גמישה.
- אבטחה:
  - אימות משתמש מאובטח.
  - מעקב אחר התנהגות משתמשים.

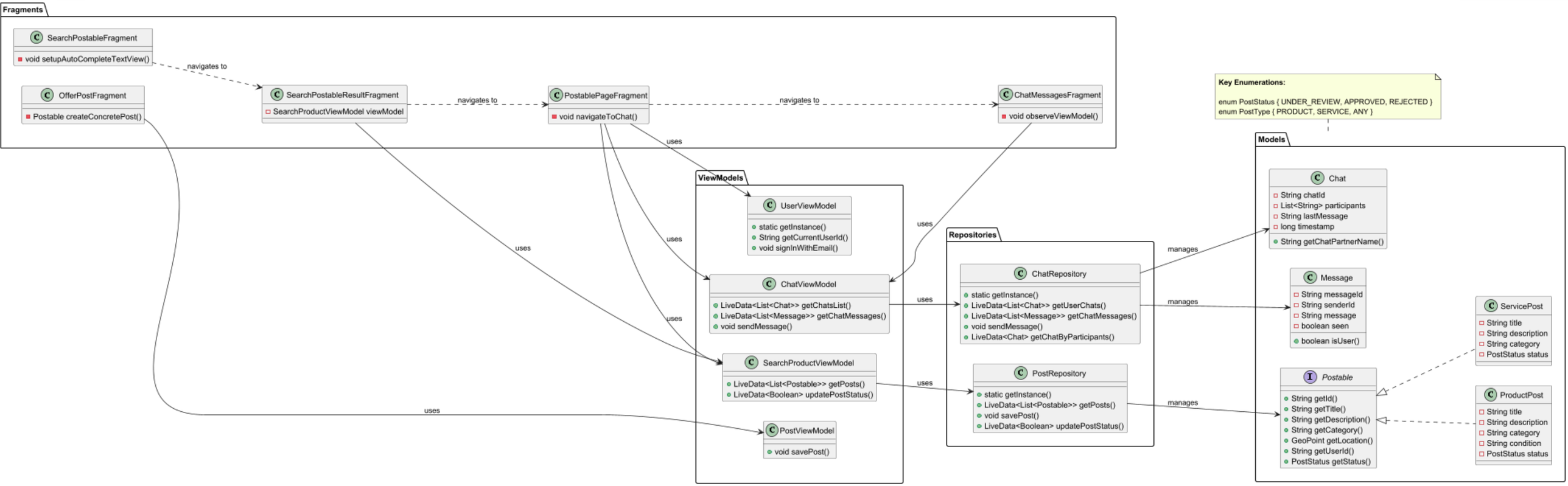
UML Diagrams

we used 'plantUml integration'

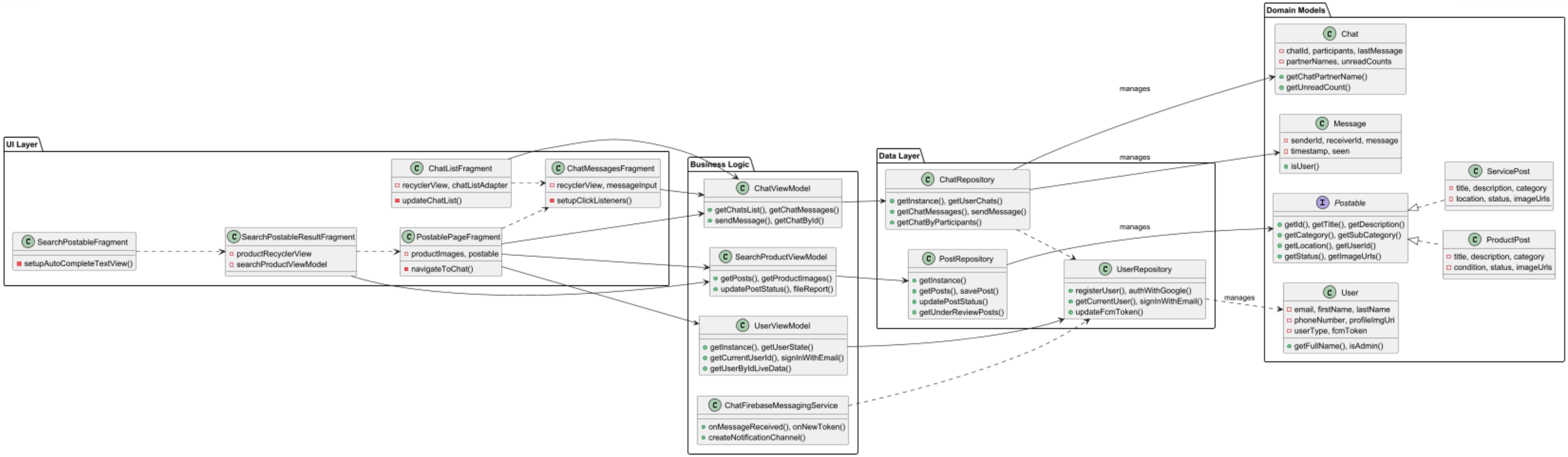
posting a product /service



searching a product /service



chating with publisher



ארכיטקטורת המערכת:

המערכת שלנו בנויה לפי עקרונות ארכיטקטורת **MVVM (Model-View-ViewModel)**, כאשר הקוד מחולק באופן ברור לשכבות שונות המשקפות את האחריות השונה של כל רכיב במערכת. להלן הסבר על מבנה המערכת ואיך חלוקת הקוד תומכת בארכיטקטורה שנבחרה

1. שכבת המודל

מיקום בקוד: **com.sibi.helpi.models**

שכבה זו מכילה את הייצוג של הנתונים והלוגיקה העסקית:

- **Postable** - ממשק המגדיר את המבנה הבסיסי של פוסטים במערכת
- **ProductPost** - מימוש של **Postable** עבור מוצרים
- **ServicePost** - מימוש של **Postable** עבור שירותים
- **Chat** - מייצג שיחה בין משתמשים
- **Message** - מייצג הודעה בודדת בצ'אט
- **User** - מייצג משתמש במערכת

כל המודלים האלה עוסקים אך ורק בתיאור הנתונים עצמם ולא מכילים לוגיקה כיצד לשמור או לשלוף נתונים אלה. זה תואם את עקרון ההפרדה ב-**MVVM**.

2. שכבת הנתונים **(Repositories)** - חלק משכבת המודל

מיקום בקוד: **com.sibi.helpi.repositories**

שכבה זו אחראית על הגישה לנתונים ומבצעת את התקשורת עם מקורות הנתונים החיצוניים (ממומשים בתור **singletons**):

- **PostRepository** - מספק גישה לפוסטים בבסיס הנתונים
- **ChatRepository** - מנהל את הגישה לצ'אטים והודעות
- **UserRepository** - מנהל את הגישה לנתוני משתמשים ואימות

המחלקות הללו מסתירות את מורכבות הגישה לנתונים **(Firebase Firestore, Realtime Database)** מהשכבות האחרות, ובכך יוצרות הפרדה ברורה בין הלוגיקה העסקית לבין גישה לנתונים.

3. שכבת ה-**ViewModel**

מיקום בקוד: **com.sibi.helpi.viewmodels**

שכבה זו מקשרת בין התצוגה **(Fragments)** לבין שכבת הנתונים:

- **SearchProductViewModel** - מנהל חיפוש והצגת פוסטים
- **ChatViewModel** - מנהל את לוגיקת הצ'אט והודעות
- **UserViewModel** - מנהל את הלוגיקה של המשתמש הנוכחי

ה-**ViewModels** מכילים את הלוגיקה של ממשק המשתמש מבלי להיות תלויים בתצוגה עצמה. הם מספקים **LiveData objects** שהתצוגה יכולה להירשם אליהם ולהגיב לשינויים בנתונים.

5. שכבת התצוגה **(UI Layer)**

מיקום בקוד: **com.sibi.helpi.fragments**

שכבה זו מכילה את הממשק למשתמש:

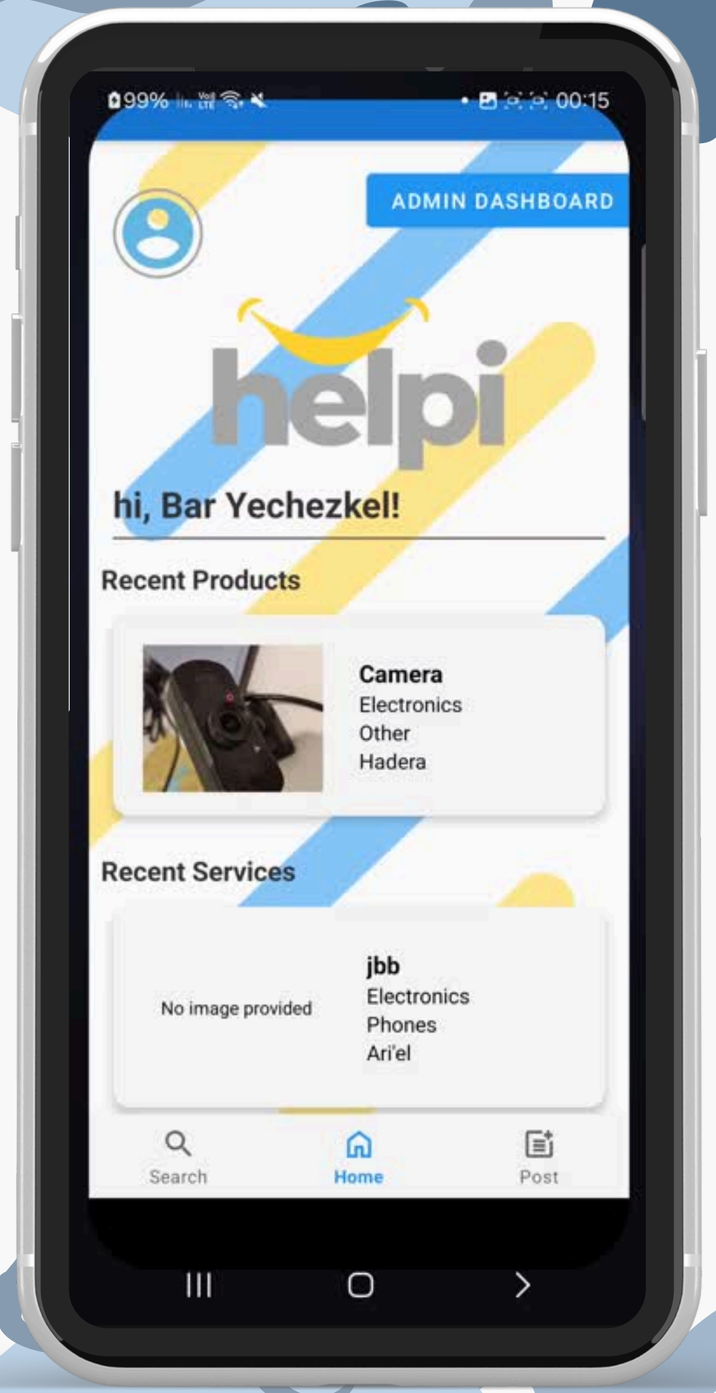
- **SearchPostableFragment** - מסך חיפוש פוסטים
- **SearchPostableResultFragment** - מציג תוצאות חיפוש
- **PostablePageFragment** - מציג פוסט ספציפי
- **ChatMessagesFragment** - מציג הודעות בצ'אט
- **ChatListFragment** - מציג רשימת צ'אטים

איך החלוקה תומכת בארכיטקטורה:

1. הפרדת אחריות: כל שכבה אחראית על היבט ספציפי של האפליקציה.
   - המודלים מייצגים את הנתונים בלבד
   - ה-**Repositories** מטפלים בגישה לנתונים
   - ה-**ViewModels** מכילים את הלוגיקה של ממשק המשתמש
   - ה-**Fragments** מציגים את הנתונים למשתמש
2. קוד בר-בדיקה: ההפרדה מאפשרת בדיקה יעילה של כל שכבה בנפרד.
3. תחזוקה קלה: שינויים בשכבה אחת (למשל, בממשק המשתמש) לא דורשים שינויים בשכבות אחרות.
4. קוד מודולרי: קל להחליף חלקים מסוימים בקוד, למשל להחליף את מנגנון האחסון מבלי לשנות את שאר המערכת.
5. סינכרוניזציה באמצעות **LiveData**: השימוש ב-**LiveData** מאפשר עדכון אוטומטי של ממשק המשתמש כאשר הנתונים משתנים.

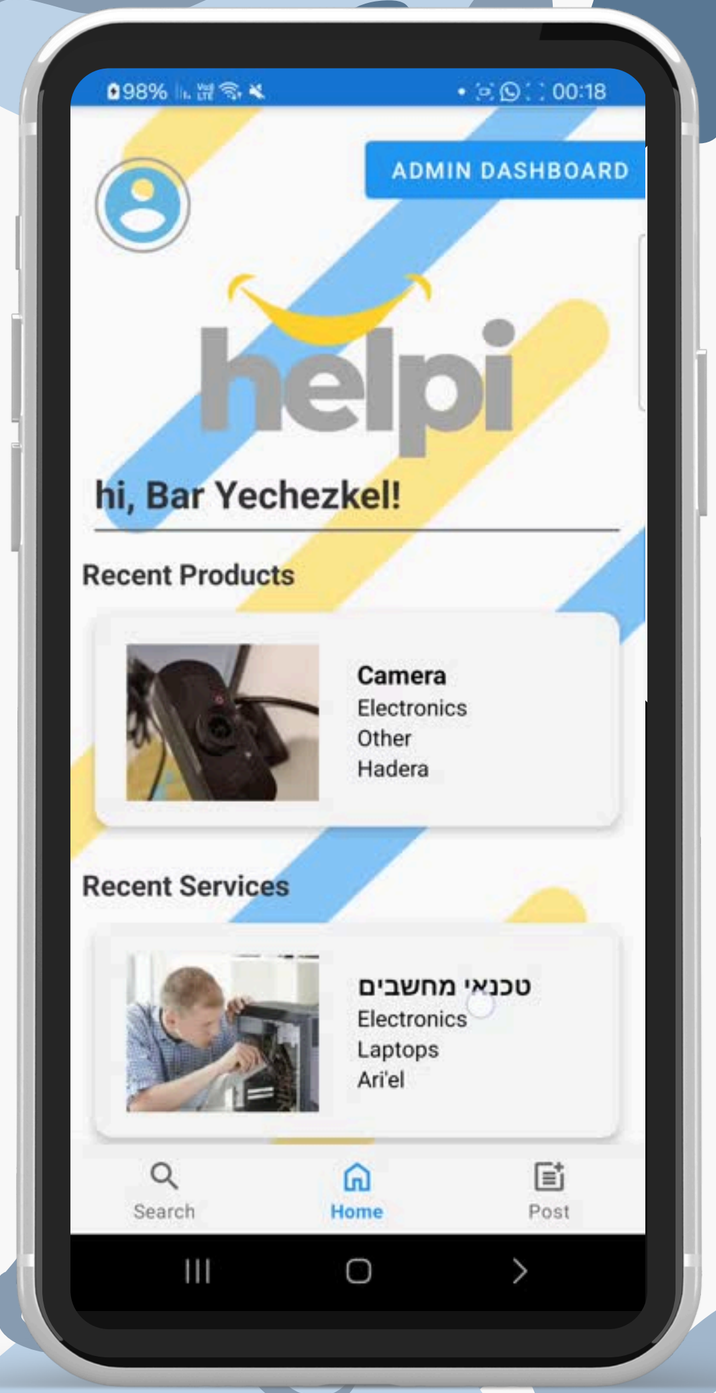
דוגמה מעשית לאופן שבו הארכיטקטורה תומכת בפיתוח היא תהליך הצגת רשימת צ'אטים:

1. **ChatRepository** מתחבר ל-**Firebase** ומאזין לשינויים בצ'אטים
2. **ChatViewModel** מבקש נתונים מה-**repository** ומחזיק **LiveData**
3. **ChatListFragment** 'נרשם' ל-**LiveData** של ה-**ViewModel** ומציג את הנתונים



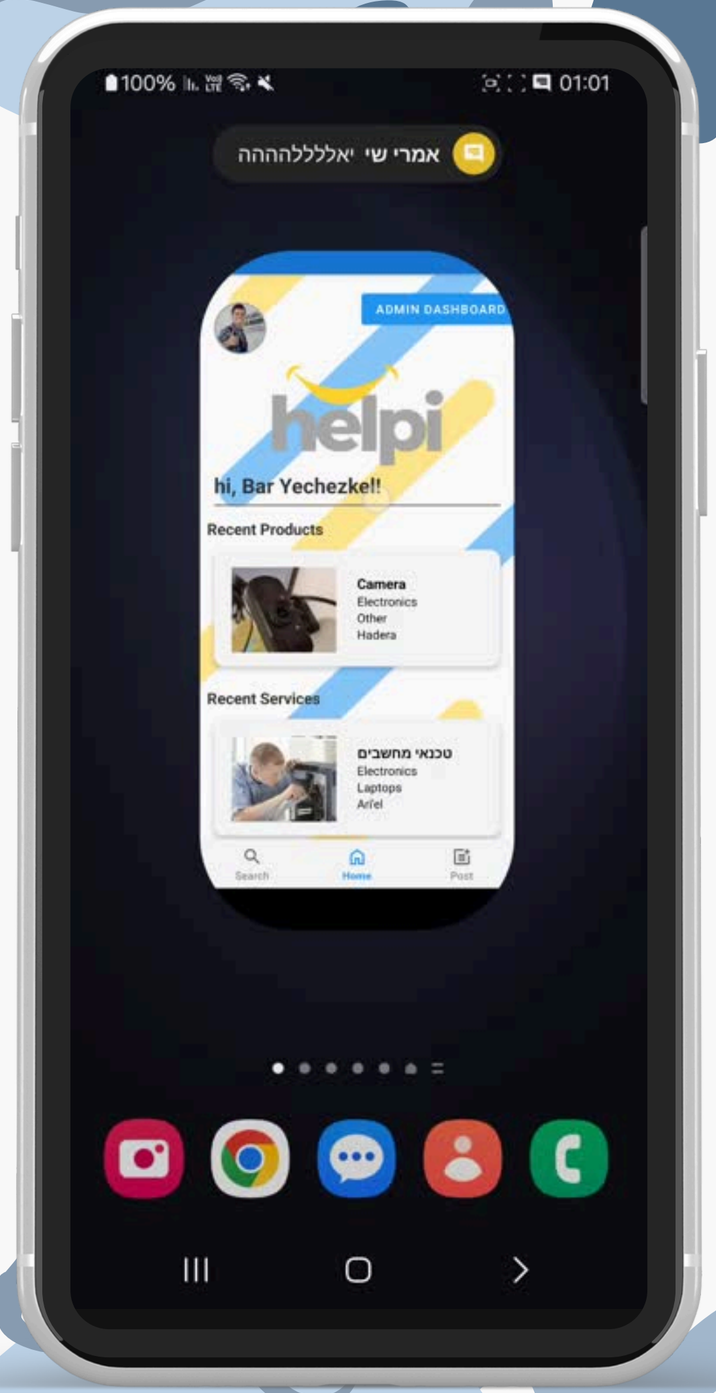
סרטוני הדגמה:
פרסום פוסט-שירות
+אישור פוסט ע"י מנהל

קישור לסרטון בדרייב



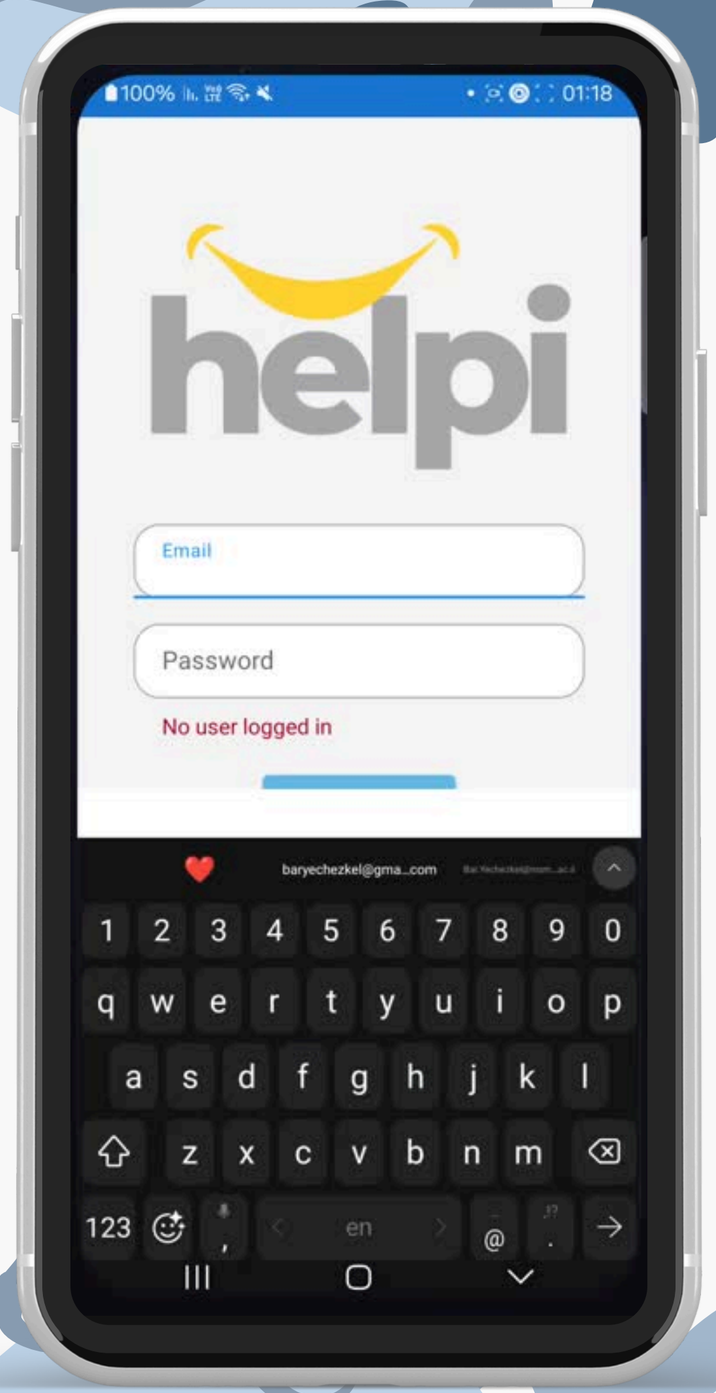
חיפוש פוסט־מוצר

קישור לסרטון בדרייב



קבלת התראה וצ'אט

קישור לסרטון בדרייב



החלפת שפה+כללי

קישור לסרטון בדרייב

פערים:

1. התראות נוספות עבור מנהלים, לדוגמה כאשר משתמש מעלה דיווח, או משתמש מעלה פוסט חדש שזקוק לאישור מנהל, המנהלים יקבלו הודעה מתאימה. ניתן גם לסנן לפי מנהל מקומי, שיקבל הודעה רק אם הפוסט/דיווח ברדיוס מסוים.
2. התראות עבור משתמש ששלח פוסט/דיווח והמנהל הגיב עליהם.
3. הוספה של אפשרות לבקש שירות/מוצר.
4. הצגה של מוצרים על מפה.
5. אפשרות לפרסום אנונימי.

לבטים:

1. עבודה עם **android studio** לעומת **react native**.
2. שימוש ב- **raeltime** או ב **firestore**.
3. חלוקת עבודה בצוות.
4. עיצוב - כניסה לעולם לא מוכר ושאיבת מידע ממקורות שונים שגרמו למחלוקות.
5. צורת עבודה - יצירת **branch**'ים לפי משימות או לפי אנשים.
6. מתח בין הספק לאיכות הקוד.

כדאי להוסיף לאפליקציה:

1. אפשרות לפרסם בקשת שירות/ מוצר.
2. התראות על העלאת פוסט לאדמינים.
3. התראות על נתינת הרשאות אדמין למשתמש.
4. חלוקת אזורים לאדמינים מקומיים כדי שהעומס יתחלק בין כולם.
5. דאשבורד אדמינים משותף לפרסום הודעות ועדכונים לאדמינים.
6. אינטגרציה לניווט לנקודת איסוף מוצר.
7. דירוג של בעלי מקצוע שמציעים שירותים.

helpi

FOR A BETTER WORLD

Thank You !