מסמך אפיון עבור אפליקציית מובייל "EasyWedding"

(טיוטה)

מגיש: עידן דמרי

<u>הערה</u>

בסיס הנתונים שונה ל-Firebase Realtime Database .

דבר לא השתנה במובן הלוגי.

תוכן העניינים

הערה

מסמך זה אינו בהכרח שלם. חלקים מסוימים בנוסח עלולים להתעדכן מספר פעמים לפני הגשה סופית של הפרויקט. המסכים המוצגים בקובץ זה קיימים רק כדי להמחיש את הפונקציונאליות הבסיסית של האפליקציה. המראה והתחושה בפועל יפותחו לאורך זמן.

1. רקע

כיום קיימות מגוון אפליקציות מובייל הנועדו לעזור לזוגות אשר עתידים להתחתן לארגן את חתונתם. מניסיון אישי ובדיקה מקיפה הבחנתי כי חלק גדול מאפליקציות אלו מציעות ממשק מורכב למדי. באפליקציה בה השתמשתי בחתונה שלי היו מספר רב של פרטים אשר העמיסו על זיכרון העבודה שלי. לבסוף פיתחתי אפליקציה מאד פשוטה וספציפית עבורי, בה נעשה שימוש רב והיא הועילה לי מאד. מכאן החלטתי בפרויקט זה לפתח אפליקציית מובייל מתאימה, גנרית יותר ועם ממשק פשוט.

2. תיאור האפליקציה

EasyWedding היא אפליקציית מובייל לארגון קבוצה מסוימת של מידע הקשור לחתונה.

3. קהל היעד

זוגות הנמצאים במעמד לפני חתונה.

4. מטרת האפליקציה

מטרת האפליקציה היא להציע ממשק מובייל פשוט המאפשר ביצוע מניפולציות על מידע אשר נדרש עבורו מעקב. הממשק פונה לסוגי המידע הבאים: רשימת פיצ'רים (למשל DJ) ורשימת מוזמנים. שני אלו מהווים חלק בלתי נפרד מהאירוע המיועד.

5. פלטפורמות וטכנולוגיות

- EasyWedding היא אפליקציית native המיועדת עבור טלפונים חכמים מבוססי Android.
- גרסת ערכת פיתוח תכנה מינימאלית: Android 5.0 API level 21 Lollipop.
- סביבת פיתוח: Android Studio.
- בסיס נתונים: Google Cloud Firestore.

6. פיצ'רים ופונקציונאליות

- <u>Sign In Flow</u>

המידע הקיים באפליקציה הוא אישי עבור כל משתמש. האפליקציה תעשה שימוש ב- FirebaseUI Authentication על מנת לספק אמצעי לאימות המשתמש. קיימות שתי דרכים באמצעותן המשתמש יוכל להתאמת:

1. שילוב של אימייל וסיסמא.
2. חשבון גוגל.

אם המשתמש מתקין לראשונה את האפליקציה הוא יבצע הרשמה חד פעמית.

- <u>התמדת נתונים במצב לא מקוון</u>

Cloud Firestore תומך בהתמדת נתונים גם כאשר למשתמש אין גישה לשרתי Cloud Firestore בנקודת זמן כלשהי. תכונה זו מאפשרת הטמנת עותק של נתוני בסיס הנתונים בהם האפליקציה משתמשת באופן פעיל, כך שהאפליקציה יכולה לגשת לנתונים כשהמכשיר נמצא במצב לא מקוון.
ניתן לקרוא, להאזין ולבצע שאילתות עבור הנתונים במטמון. כאשר המכשיר חוזר להיות במצב מקוון , Cloud Firestore מסנכרן את כל השינויים המקומיים שבוצעו באפליקציה לשרתיו.

תהליך זה מתבצע באופן שקוף למתכנת.

- <u>רשימת פיצ'רים עבור החתונה</u>

המשתמש יוכל לבצע מניפולציות על רשימת פיצ'רים בהם מעוניין שיהיו בחתונה.

- פעולות על רשימת הפיצ'רים:
  1. הוספת פיצ'ר חדש.
  2. עדכון פיצ'ר קיים.
  3. מחיקת פיצ'ר קיים.
- מעקב אחר תשלום לספק הפיצ'ר.
- תפריט אפשרויות עבור רשימת הפיצ'רים:
  1. מחיקת כל הפיצ'רים.
  2. מחיקת פיצ'רים לפי שם הספק.
  3. מיון הרשימה לפי: ספק, יתרת תשלום, פיצ'רים עבורם עדיין לא נבחר ספק, או סדר אלפביתי.
  4. ייצוא טקסטואלי של הפיצ'רים (כל אפליקציה במכשיר המשתמש היכולה להתמודד עם קבלת קבצי טקסט תוצע למשתמש והוא יבחר לפי שיקול דעתו).
- תרחישים:

תרחיש 1: הילה רוצה להוסיף לחתונתה קיר פרחים אתו המוזמנים יוכלו להצטלם. הילה שילמה לספק קיר הפרחים מקדמה של 200 ₪ עבור קיר הפרחים.

הילה נכנסת לאפליקציה. זו הפעם הראשונה בה היא משתמשת באפליקציה ומבחינה כי היא צריכה לבצע רישום. היא בוחרת להירשם דרך האימייל וכעת לאימייל נשלח אימות אותו היא מאשרת. כעת הילה מגיעה למסך בו היא בוחרת סיסמא ולאחר מכן מתחילה להשתמש באפליקציה.

הילה בוחרת בלשונית רשימת הפיצ'רים ולאחר מכן לוחצת על כפתור המאפשר הוספת פיצ'ר חדש. היא מגיעה למסך חדש בו היא מוסיפה מידע הקשור לפיצ'ר. היא מוסיפה את הפרטים הבאים על הפיצ'ר והספק: שם הפיצ'ר, שם הספק, טלפון הספק, אימייל הספק, קישור לאתר הספק. בנוסף היא מוסיפה כי נתנה מקדמה של 200 ₪ ומציינת את יתרת התשלום לספק.

לאחר שבוע הילה רוצה לשלם טלפונית את מלוא הסכום לספק. היא מחייגת אליו בקלות מהאפליקציה (אשר מפנה אותה לאפליקציית החיוג בטלפון) ולבסוף מעדכנת באפליקציה כי יתרת התשלום היא 0 ₪ .

תרחיש 2: הילה מוסיפה ארבעה פיצ'רים חדשים ורק לשניים מהם קיים ספק. היא נכנסת לתפריט ושם בוחרת בפעולת המיון. כעת מוצג להילה תפריט אפשרויות מיון, בו היא בוחרת למיין את רשימת הפיצ'רים כך שפיצ'רים עבורם לא נבחר ספק יופיעו ראשונים.

- <u>רשימת מוזמנים</u>

המשתמש יוכל לבצע מניפולציות על רשימת מוזמנים לחתונה.

- פעולות על רשימת המוזמנים:
  1. הוספת מוזמן חדש
  2. עדכון מוזמן קיים.
  3. מחיקת מוזמן קיים.
  4. שליחת בקשת אישור הגעה לחתונה.
- תפריט אפשרויות עבור רשימת המוזמנים:
  1. מחיקת כל המוזמנים.
  2. שליחת בקשת אישור הגעה עבור מוזמנים שעדיין לא נשלחה להם בקשת אישור הגעה.
  3. מיון הרשימה לפי: עדיפות, אישור הגעה, או סדר אלפביתי.

4. ייצוא טקסטואלי של המוזמנים.

- מעקב אחר מספר המוזמנים אשר אישרו את הגעתם לחתונה
- שליחת אישור בקשת הגעה למוזמנים עבורם עדיין לא נשלח אישור הגעה
- תרחישים:

תרחיש 1: חיים משתמש קבוע באפליקציה, הזוכרת את פרטיו. הוא בדיוק מסר לשכנו הזמנה לחתונה. חיים נכנס לאפליקציה ומקליק על לשונית המוזמנים. לאחר מכן הוא לוחץ על כפתור המאפשר לו להוסיף איש קשר קיים. כעת הוא בוחר בעזרת איזו אפליקציה להוסיף את פרטי השכן (למשל אפליקציית אנשי הקשר של Google).

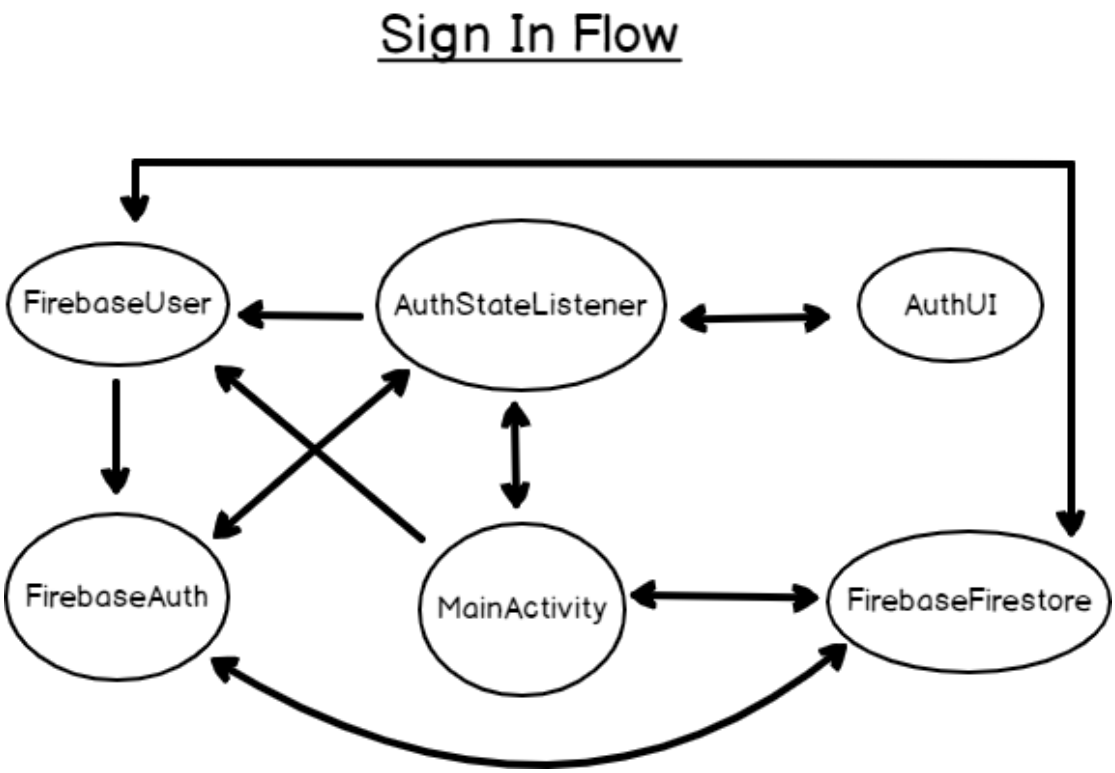
פרטי השכן נשמרים באפליקציה EasyWedding. חיים לוחץ על המוזמן החדש שהוסיף ומגיע למסך חדש המכיל את פרטי המוזמן. במסך זה, חיים מציין כי שכנו עדיין לא הוזמן ובנוסף מזין לו עדיפות עליונה כיוון שחשוב לו כי שכנו יגיע.

לאחר מספר ימים חיים שולח לשכנו בקשה לאישור הגעה לחתונה. הבקשה מגיע לשכן והוא מאשר הגעתו. חיים מבחין בסימון חזותי על האישור באפליקציה.

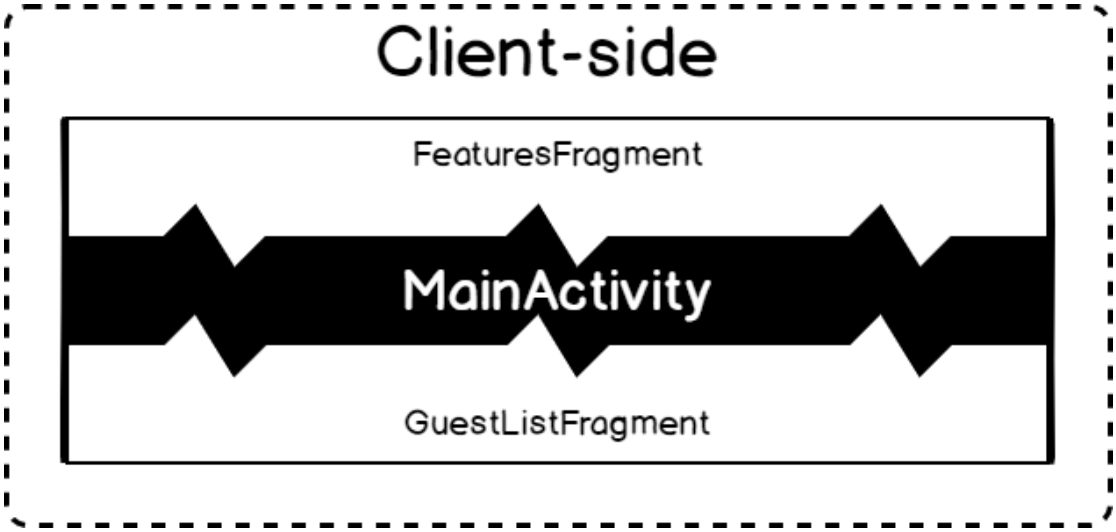
חיים מבחין כי כעת מספר המאשרים גדל באחד.

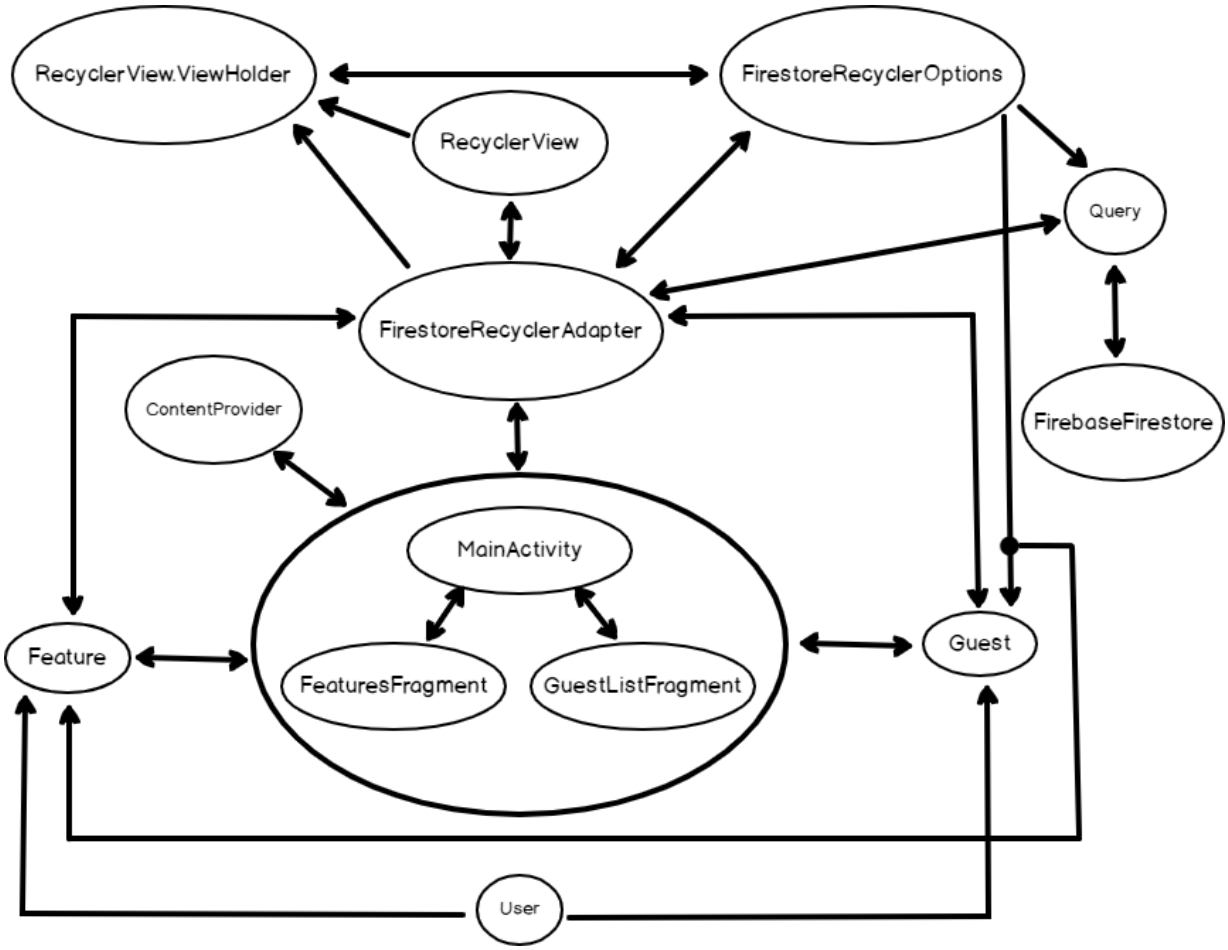
לפני שחיים עוזב את האפליקציה הוא רואה כי רשימת המוזמנים ממוינים בסדר אלפביתי. הוא לוחץ על התפריט במסך הראשי (כאשר הלשונית הנבחרת היא רשימת המוזמנים) ונכנס להגדרות. הוא מציין כי על רשימת המוזמנים להיות ממוינת בסדר יורד לפי העדיפות שניתנה לכל מוזמן.

7. רכיבי מערכת עיקריים



רכיב (קבצי Java)	תיאור
FirebaseAuth	נקודת הכניסה לכל פעולות האימות ב Firebase (צד שרת). באפליקציה זו התפקיד של רכיב זה הוא להיות אמצעי לאימות המשתמש וכן לדווח האם המשתמש התאמת.
AuthStateListener	מאזין המגיב לשינויים במצב ה- FirebaseAuth. רכיב זה פועל כאשר: • משתמש מבצע Sign in (התחברות). • משתמש מבצע Sign out (התנתקות). • כאשר מתבצע רישום מאזין בעת אתחול ה- FirebaseAuth. מאזין זה יכיל את הקוד הבונה את מסך ה- Sign In Flow (מוצג בהמשך), בעזרת הרכיב AuthUI כאשר המשתמש מתחבר וכן את הקוד ההורס את מסך זה כאשר המשתמש מתנתק.
FirebaseUser	מייצג מידע על פרופיל המשתמש בהקשר של בסיס הנתונים Cloud Firestore. בנוסף, מאפשר לנהל את מצב אימות המשתמש. מצב המשתמש יאפשר לדעת כיצד יש לפעול בעת פעולת המאזין AuthStateListener.
AuthUI	נקודת הכניסה לממשק הגרפי של תהליך האימות. בעזרת רכיב זה ניתן לבנות UI מובנה המאפשר שימוש בחשבונות בהתאם לשיקול המתכנת (למשל התחברות עם חשבון גוגל, או על ידי שילוב של אימייל וסיסמא). בניית ה- UI נעשית בתוך הרכיב המאזין AuthStateListener. בסיום פעולת המשתמש לאחר פירוק UI זה תוצאת פעולת המשתמש נשלחת חזרה למאזין.
MainActivity	המסך הראשי של האפליקציה. תיאורו יורחב בהמשך. רכיב זה נמצא באיור לעיל לשם שלמות האיור.
FirebaseFirestore	בסיס הנתונים בו האפליקציה משתמשת. בהתאם להרשאות הגישה אל אזורים מסיומים בבסיס הנתונים המידע המתאים מוצג למשתמש לאחר שביצע אימות.


Client-side
FeaturesFragment
MainActivity
GuestListFragment


RecyclerView.ViewHolder
FirestoreRecyclerOptions
RecyclerView
Query
FirestoreRecyclerAdapter
ContentProvider
FirebaseFirestore
MainActivity
Feature
FeaturesFragment
GuestListFragment
Guest
User

רכיב (קבצי Java)	תיאור
<u>Data Models – User, Feature, Guest</u> לכל מודל נתונים (ישות) קיימות שיטות GETTERS ושיטות SETTERS הנוהגים על פי דפוסי השמות של JavaBean. דבר זה מאפשר ל- Firestore למפות את הנתונים לשמות של שדות. למשל getProviderName() מספק שם (providerName) של ספק. לכל מחלקה של ישות יהיה בנאי ריק. דבר זה הכרחי עבור Firestore כדי לבצע מיפוי אוטומטי של נתונים.	
User	מודל הנתונים המייצג את ישות המשתמש. לכל משתמש יש הפניה לפיצ'רים ומוזמנים השייכים לו.
Feature	מודל הנתונים המייצג את ישות הפיצ'ר. <u>שדות:</u> שם פיצ'ר, שם ספק, טלפון ספק, מקדמה, יתרת תשלום.
Guest	מודל הנתונים המייצג את ישות המוזמן לחתונה. <u>שדות:</u> השם הפרטי של המוזמן, שם משפחה של המוזמן, טלפון מוזמן, עדיפות, הגעה לחתונה.
יתר הרכיבים	
Query	מייצג שאילתה עבור בסיס הנתונים.
FirestoreRecyclerAdapter	כורך Query ל RecyclerView ומגיב לאירועי זמן-אמת (הנראים באופן חזותי ב- UI) הכוללים הוספה, הסרה, שינוי מיקום, או עדכון של פריטים.

	אובייקט זה משתמש ב ViewHolder כדי ליצור "פריט רשימה" עבור ה-RecyclerView ולאחר מכן ממקם אותו ב RecyclerView.
FirestoreRecyclerOptions	מייצג מאפיינים הקובעים את התצורה של FirestoreRecyclerAdapter. בנוסף בעזרת אובייקט זה נודע ל FirestoreRecyclerAdapter מהו ה Query.
RecyclerView.ViewHolder	מתאר תצוגה של פריט ב-RecyclerView ואת ה metadata לגבי מיקום הפריט ב-RecyclerView. להבדיל מהמתודה findviewbyid אשר יוצרת בכל קריאה במקום שונה בזיכרון את כל האובייקטי-תצוגה הקשורים לפריט כלשהו ב RecyclerView, ViewHolder מטמין את מיקומי האובייקטי-תצוגה עבור כל פריט ולכן יותר יעיל (ולכן ניתן לבצע מיחזור של פריטי RecyclerView אחרים שברגע נתון אינם נראים על המסך).
RecyclerView	רשימה ניתנת לגלגול (scrolling) המכילה "פריטי רשימה" .

	באפליקציה זו הרשימה הינה רשימה אנכית.
FeaturesFragment	הפיסה המודולרית, בפעילות המרכזית של האפליקציה, האחראית על רשימת הפיצ'רים.
GuesListFragment	הפיסה המודולרית, בפעילות המרכזית של האפליקציה, האחראית על רשימת המוזמנים.
MainActivity	נקודת הכניסה של האפליקציה. פעילות זו מורכבת משני אובייקטים מטיפוס Fragment. ויזואליזציה של פעילות זו ניתן לראות בחלק של המסכים המופיע בהמשך המסמך.
FirebaseFirestore	בסיס הנתונים
ContentProvider	האפליקציה תעשה שימוש ברכיב ה-ContentProvider של אפליקציית אנשי הקשר. כאשר המשתמש ירצה להוסיף מוזמן חדש הוא ילחץ על כפתור המאפשר לו לעשות זאת. לאחר מכן האפליקציה תבקש מידע עבור השאילתה המתאימה מה-Contacts Content Provider.

8. בסיס הנתונים

האפליקציה עושה שימוש בבסיס הנתונים Cloud Firestore.

CloudFirestore הוא בסיס נתונים NoSQL מודולרי המבוסס על מחשוב ענן. NoSQL אינו ממודל במבנה טבלאי סטנדרטי אשר נפוץ בבסיסי נתונים יחסיים.

מודל הנתונים בבסיס נתונים זה יותר אינטואיטיבי. אין טבלאות ושורות אלא המידע מאחסן ב Documents (מסמכים), כאשר אלו מאורגנים בתוך Collections (אוספים).

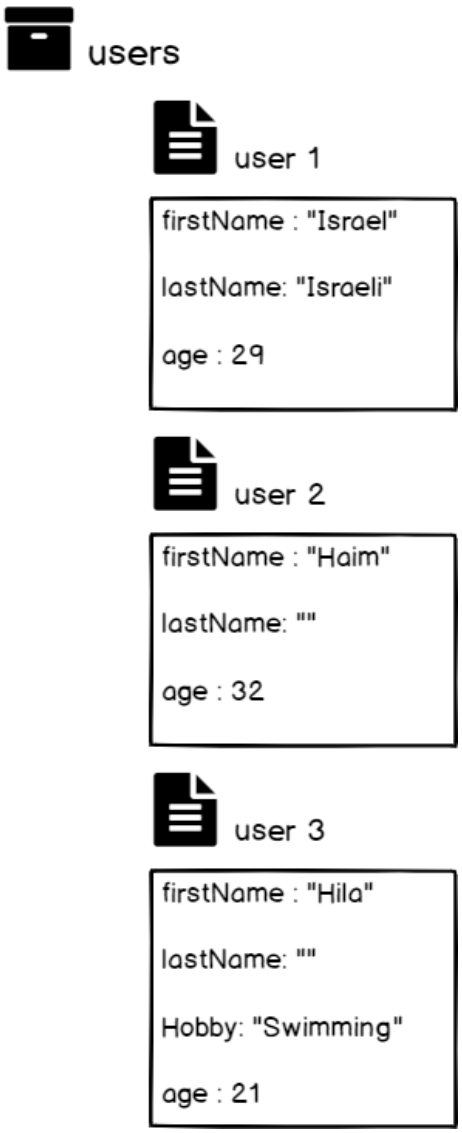
כל מסמך מכיל קבוצה של זוגות סדורים כך שהאיבר הראשון בכל זוג סדור הוא מפתח והאיבר השני הוא הערך הכרוך אל מפתח זה.

כל מסמך יכול להכיל הפנייה אל subcollections (תתי-אוספים).

מסמכים ואוספים יכולים להכיל טיפוסי נתונים פרימיטיביים כמו מספרים, או אובייקטים מורכבים כמו רשימות.

מסמכים ואוספים נוצרים באופן עקיף ב Cloud Firestore. אם מנסים ליצור מסמך או אוסף שאינם קיימים אז Cloud Firestore יוצר אותם. למשל, כך יכול להיראות מסמך המייצג משתמש כלשהו באפליקציה.

אוסף של משתמשים יכול להיראות למשל כך:



אפשר להבחין ממשתמשת מספר 3 כי

אצלה קיים שדה שלמשתמשים האחרים אין.

מסמכים שונים יכולים להכיל שדות שונים.

אם משווים בסיס נתונים מבוסס טבלאות ושורות

לבסיס הנתונים בו האפליקציה משתמשת אז

אפשר לומר כי אוסף מייצג טבלה

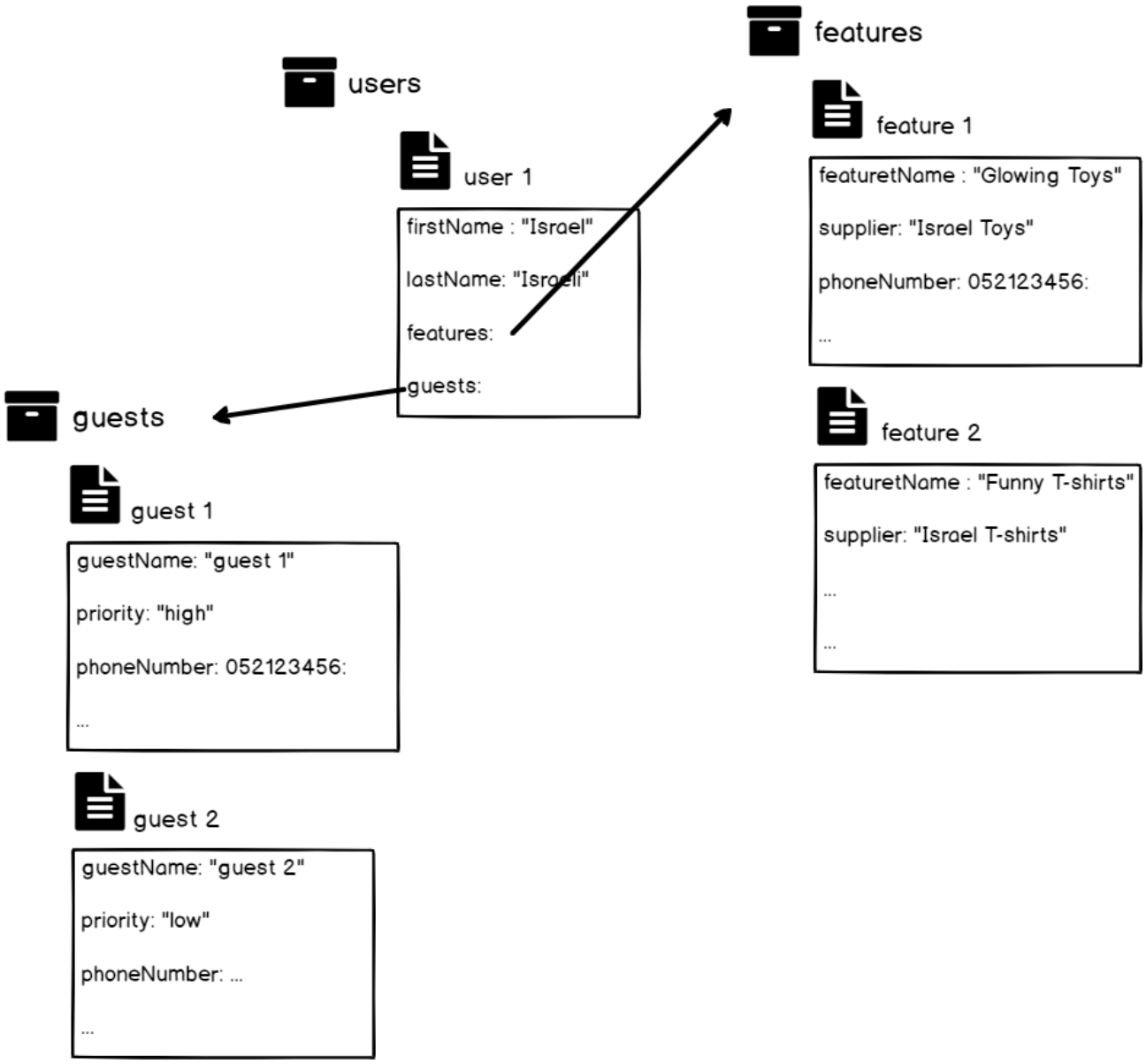
במקרה הזה טבלת המשתמשים.

וכל מסמך (משתמש) הוא שורה טבלה, השוני העיקרי במבנה הוא שאין בהכרח שדות זהים למסמכים שונים וכי שדה כלשהו במסמך יכול

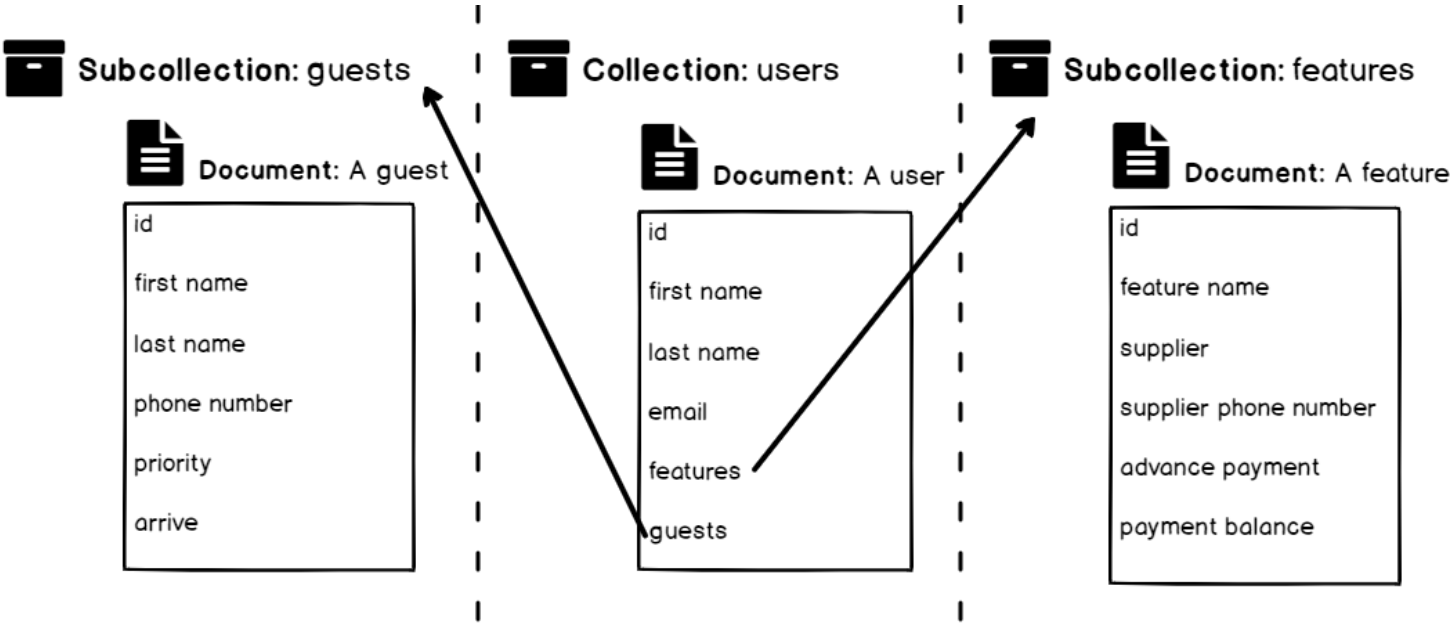
להכיל מצביע לתת-אוסף אחר.

דוגמה לאוסף משתמשים המכיל משתמש אחד (מסמך אחד). למשתמש זה יש שדות לשם פרטי, משפחה, מצביע לתת-אוסף הפיצ'רים ומצביע לתת-אוסף המוזמנים.

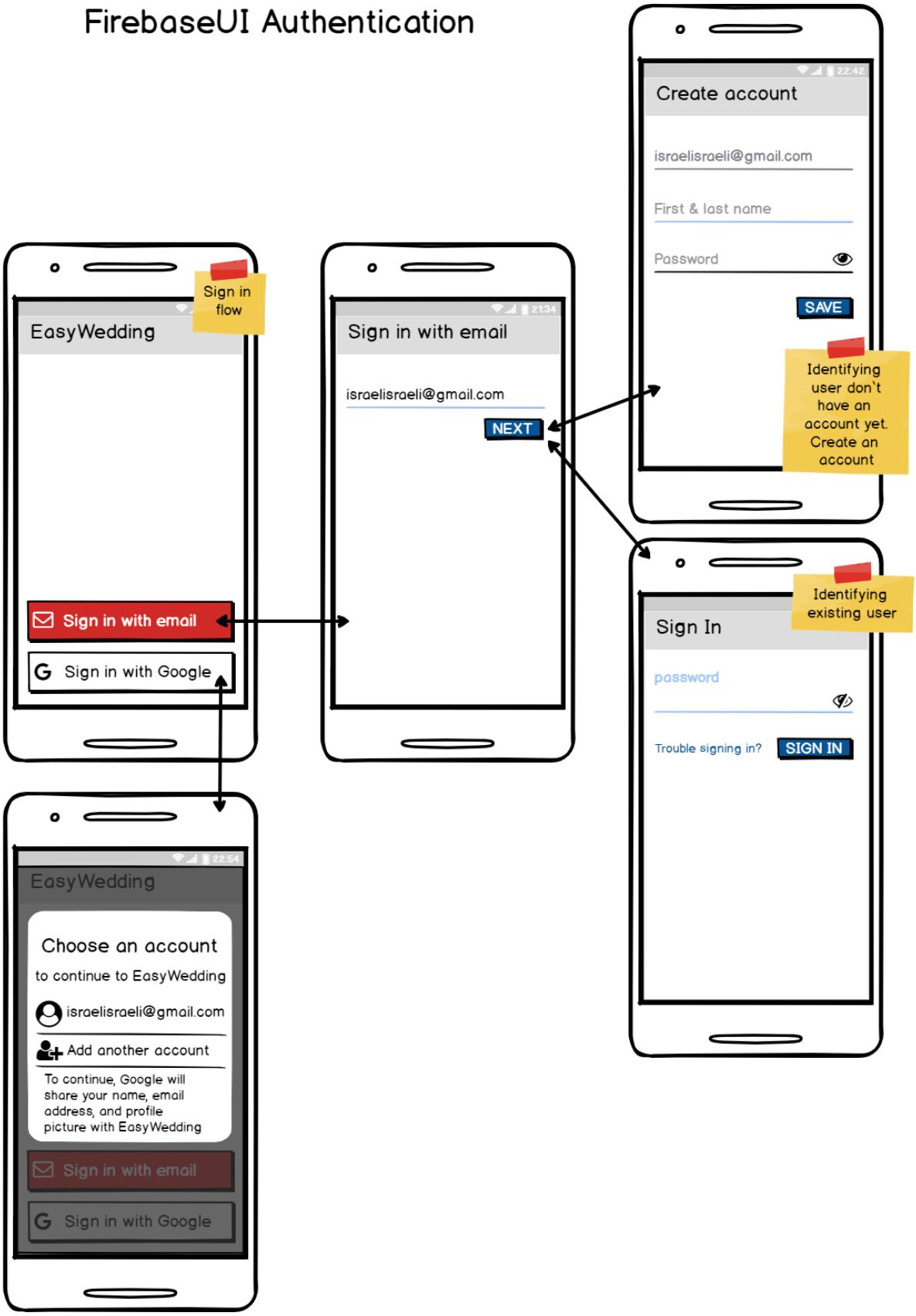
תת-אוסף הפיצ'רים מכיל שני פיצ'רים (מסמכים) ותת-אוסף המוזמנים מכיל שני מוזמנים (מסמכים).


features
users
feature 1
user 1
featuretName : "Glowing Toys"
supplier: "Israel Toys"
phoneNumber: 052123456:
...
firstName : "Israel"
lastName: "Israeli"
features:
guests:
guests
feature 2
guest 1
featuretName : "Funny T-shirts"
supplier: "Israel T-shirts"
...
...
guestName: "guest 1"
priority: "high"
phoneNumber: 052123456:
...
guest 2
guestName: "guest 2"
priority: "low"
phoneNumber: ...
...

לסיכום, בסיס הנתונים נראה כך:

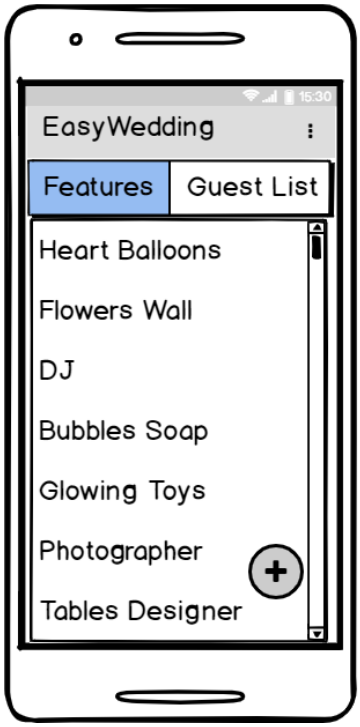


9. מסכים

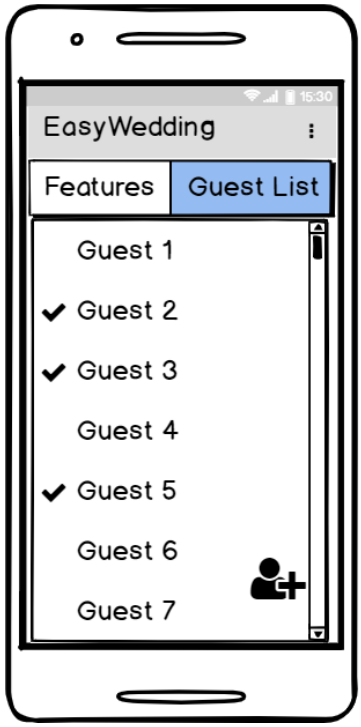

FirebaseUI Authentication
Sign in flow
EasyWedding
Sign in with email
Sign in with Google
Sign in with email
israelisraeli@gmail.com
NEXT
Create account
israelisraeli@gmail.com
First & last name
Password
SAVE
Identifying user don't have an account yet. Create an account
Identifying existing user
Sign In
password
Trouble signing in?
SIGN IN
EasyWedding
Choose an account
to continue to EasyWedding
israelisraeli@gmail.com
Add another account
To continue, Google will share your name, email address, and profile picture with EasyWedding
Sign in with email
Sign in with Google

Fragments

Features Fragment



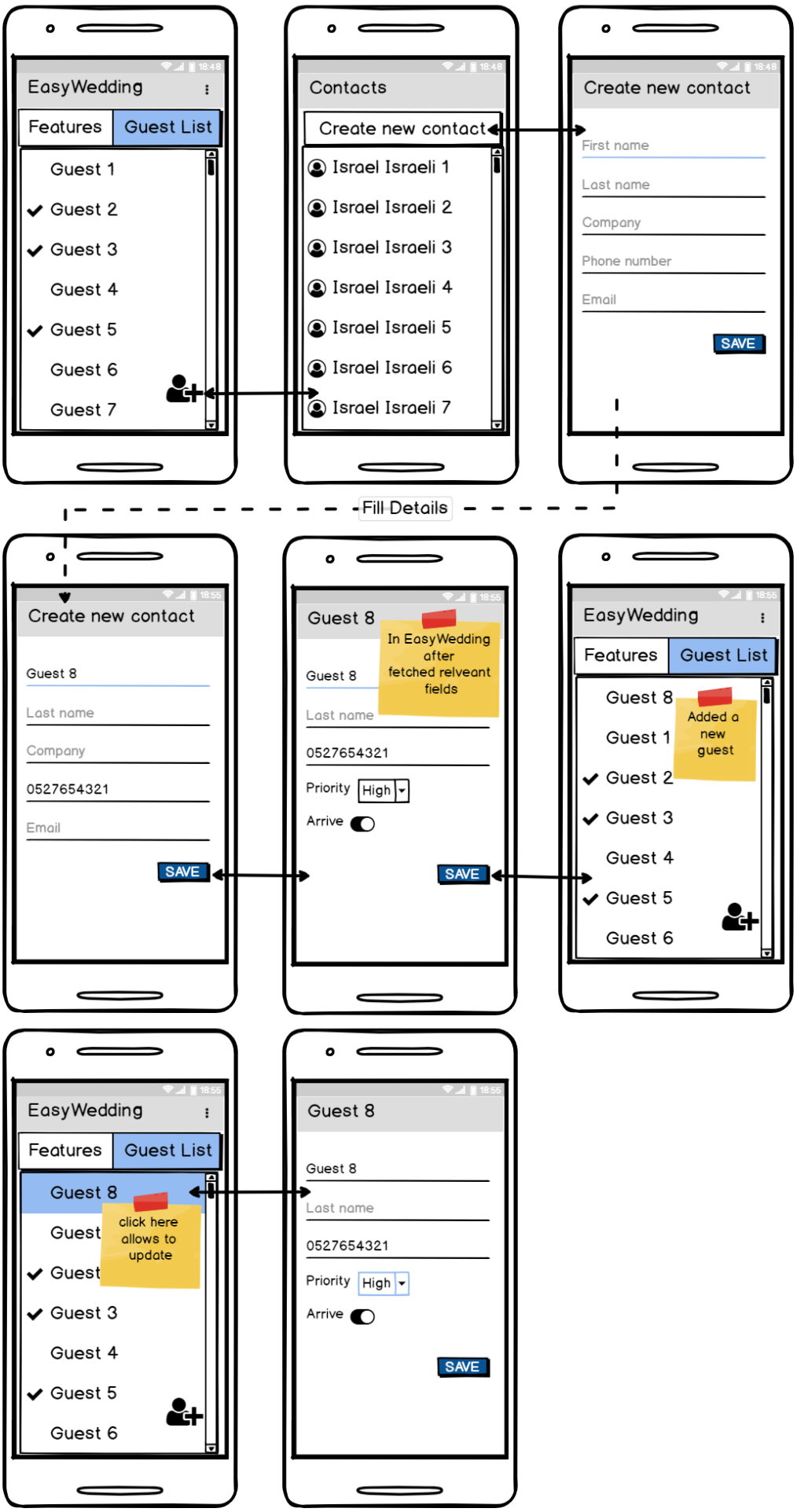
Guest List Fragment



Features Fragment

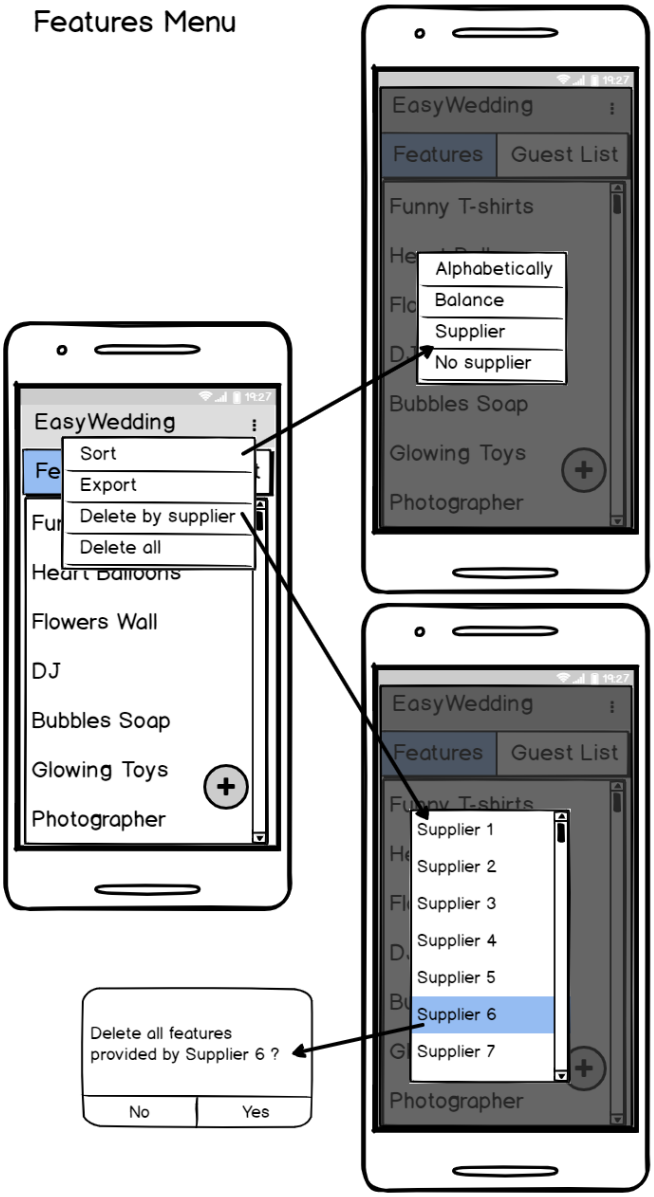
Guest List Fragment

Using Contacts Content Provider



Menus

Features Menu



Guest List Menu