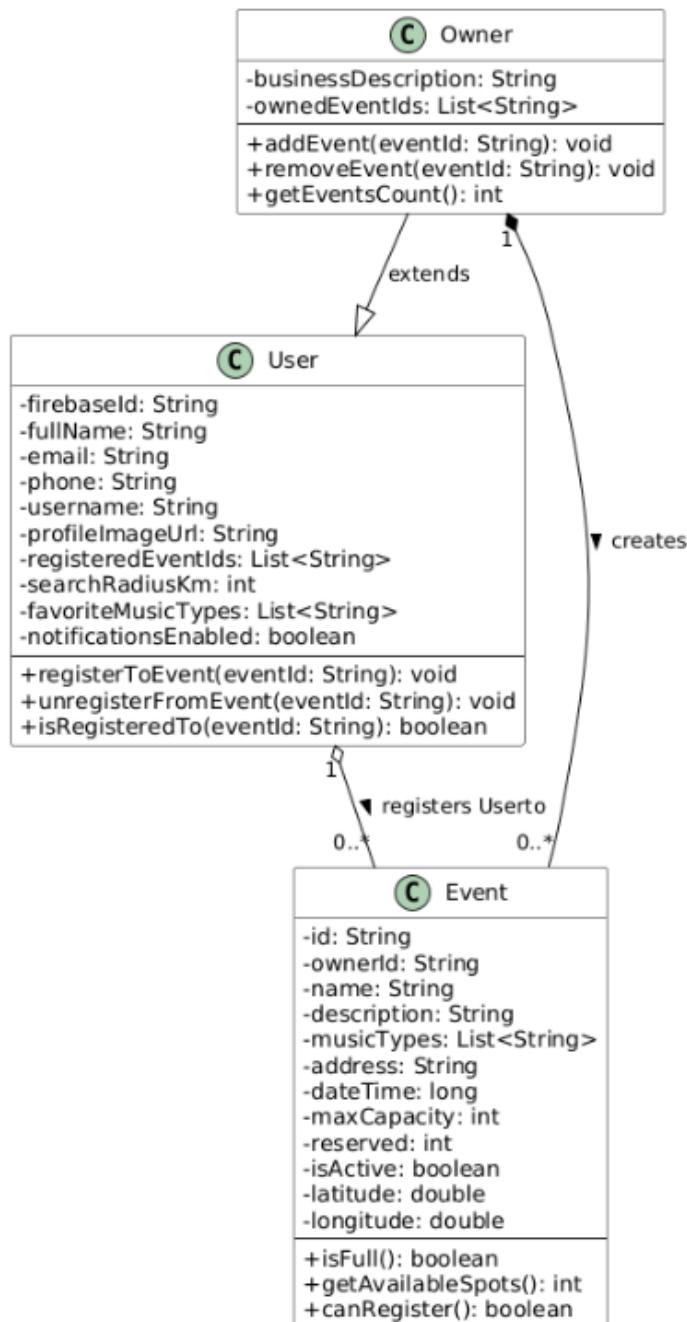


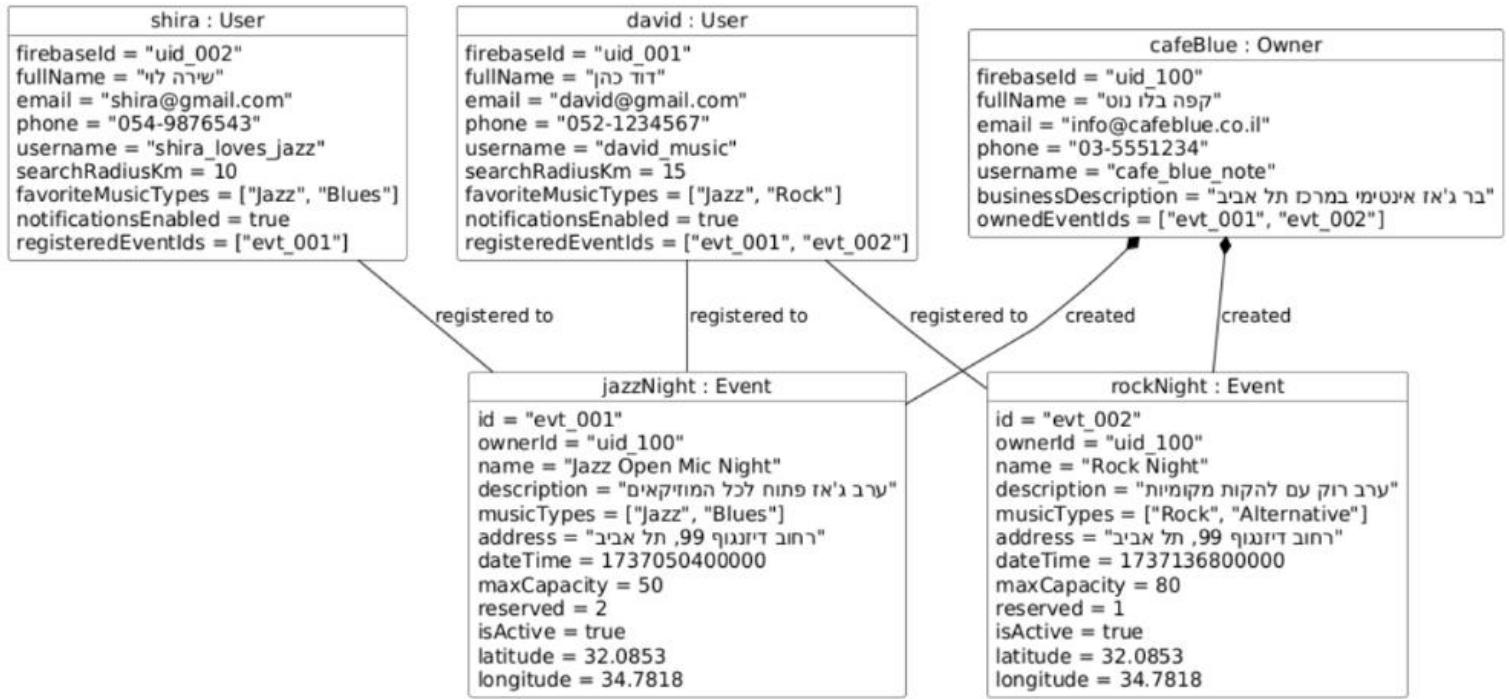
חברי הצוות: רעות גרבֶר, אורן דרַעַי, שירת הוטרָה.

מסמך ניתוח - דיאגרמות:

: Class Diagram.1

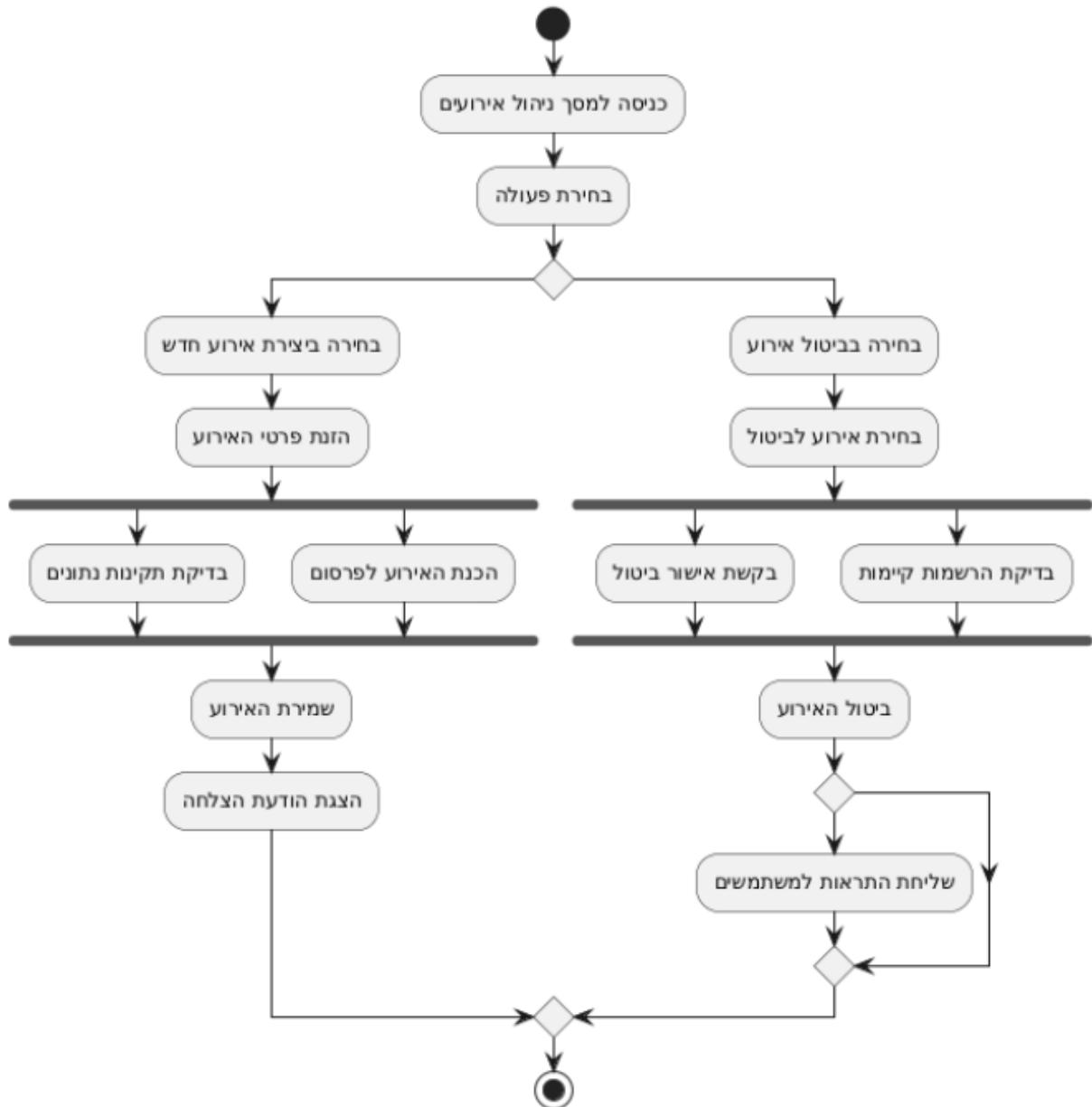


:Object Diagram .2

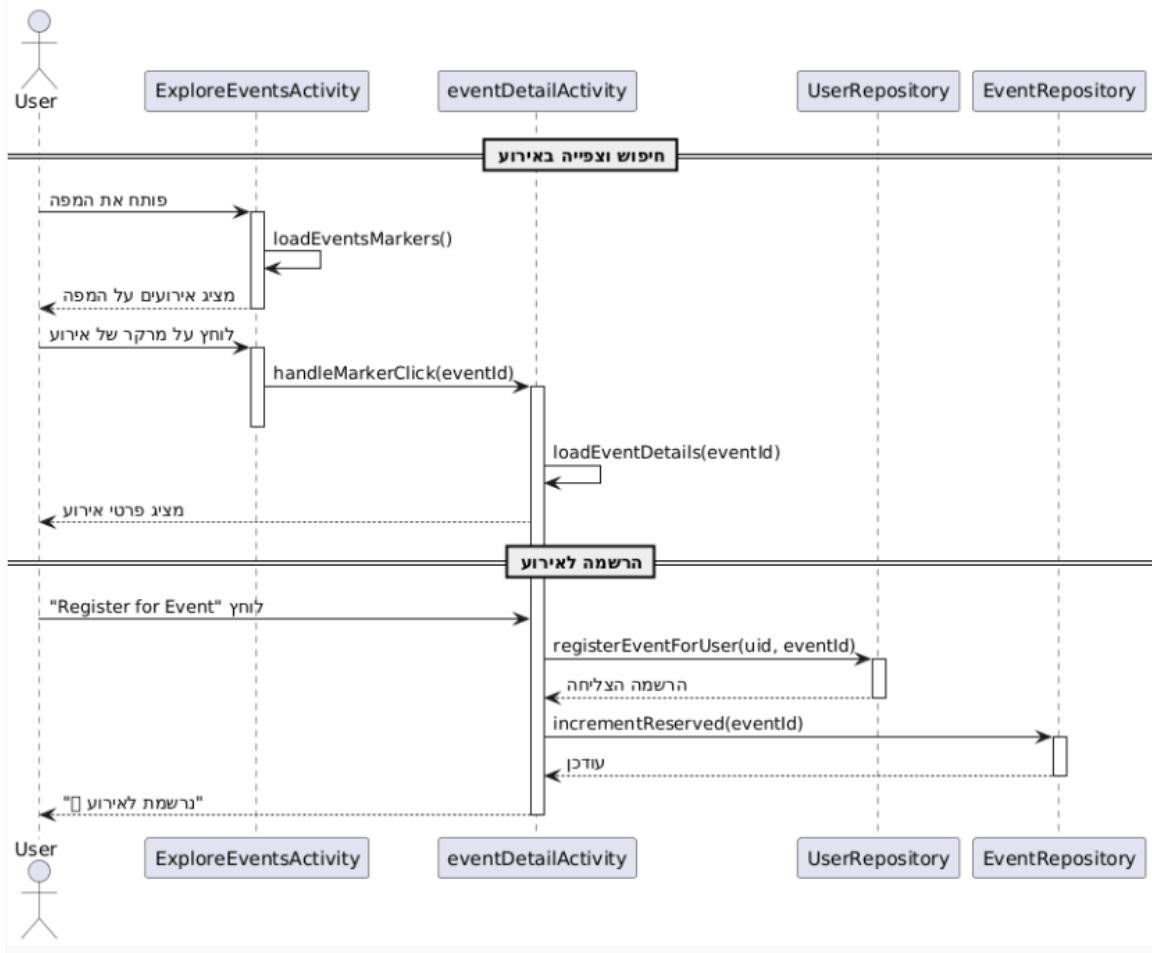


Activity Diagram.3

הדיאגרמה מתארת את תהליך ניהול האירועים על ידי בעל עסק, הכלול יצירת אירוע חדש או ביטול אירוע קיימ.

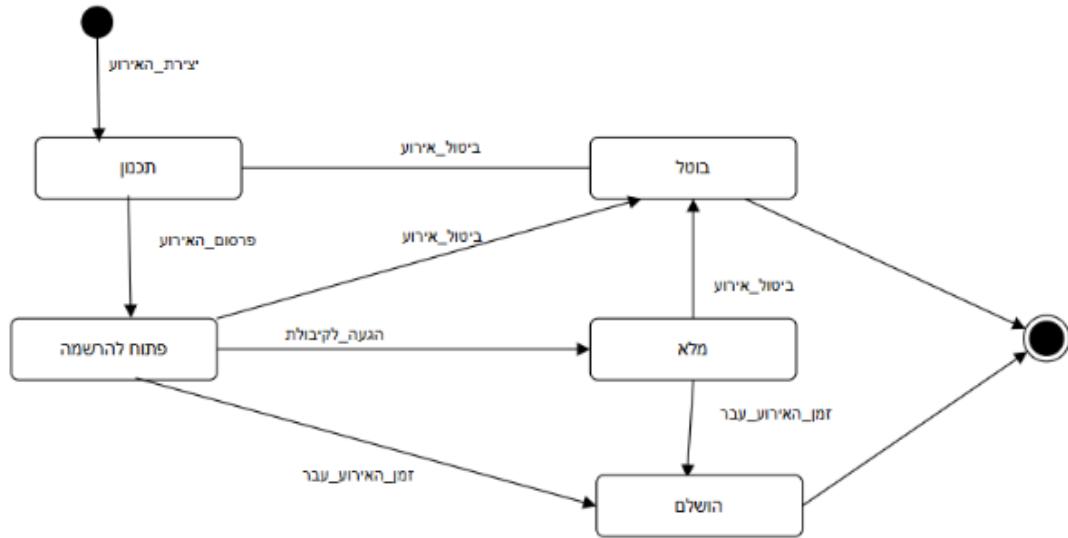


.Sequence Diagram.4

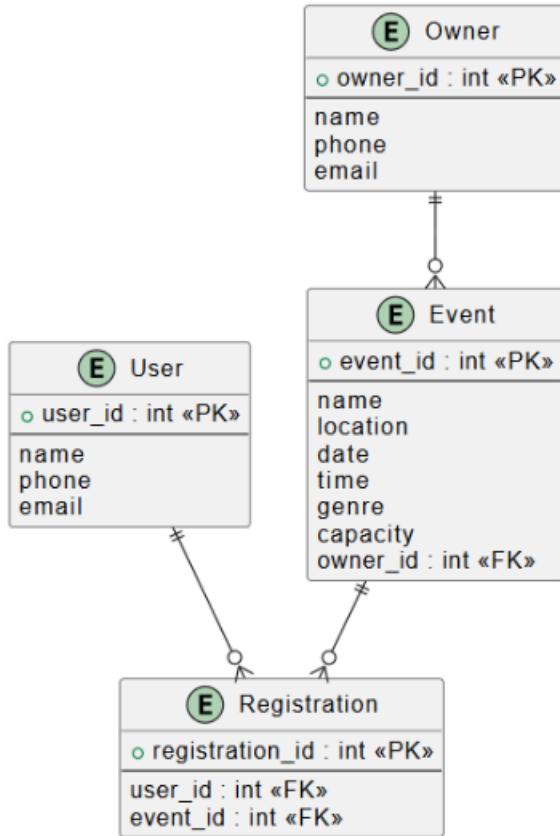


: State Machine Diagram.5

תיאור של אירוע (Event)



:ERD – Entity Relationship Diagram .6



הדיagramה מציגה את מבנה בסיס הנתונים של מערכת לניהול אירועים, המנורמל לרמה 3-NF.

המערכת כוללת ארבע ישותות עיקריות:

User- מייצגת משתמש רגיל במערכת. הטבלה שומרת פרטי זהה בסיסים כגון שם, טלפון ודוא"ל, ולכל משתמש מזהה ייחודי (`user_id`).

Owner- מייצגת מנהל אירועים. למרות הדמיון במאפייני זההו ל-User, מדובר בישות נפרדת עם תפקיד שונה במערכת. לכל Owner מזהה ייחודי (`owner_id`).

Event - מייצגת אירוע במערכת וכוללת פרטים כגון שם, מקום, תאריך, שעה, צ'אנר וקבילות. השדה `id_owner` הוא מפתח זר המצביע על בעל העסק שיצר את האירוע.

Registration - מייצגת הרשמה של משתמש לאירוע. טבלה זו משמשת כטבלת קשר בין `User` ל-`Event`. קשרים בין הtablאות:

Owner one-to-many Event - בעל עסק אחד יכול ליצור מספר אירועים, וכל אירוע יש בעל עסק אחד.

User one-to-many Registration . משתמש יכול להירשם למספר אירועים.

Event one-to-many Registration . לאירוע יכולים להיות מספר נרשמים.

אין קשר ישיר בין `User` ל-`Event`, כל הרשמה מתבצעת דרך טבלת `Registration`.

רמות נרמול:

NF1 כל הtables מכילות שדות אוטומטיים בלבד, אין שדות המכילים רשימות או ערכים מרובים, וכל הטבלה מפתח ראשי המזוהה באופן חד משמעי כל רשומה.

NF2 כל שדה שאינו מפתח תלוי ישירות בפתח הראשי של הטבלה, ולכן אין תלות חלונית בין שדות.

NF3 אין תלות טרנסיטיבית בין שדות שאינם מפותחות, כל שדה שאינו מפתח תלוי ישירות בפתח הראשי של הטבלה שלו בלבד, והמידע על משתמשים, בעלי עסקים או אירועים מופרד לשויות נפרדות.

system screens structure

:רשימת מסכים מלאה:

מסכים למשתמש רגיל (User):

Login Screen, Register Screen, Home Screen, Event Details Screen, Profile Screen, My Events Screen, Notifications Screen.

מסכים למנהל אירועים (Owner):

Owner Home, Create Event Screen, Edit Event Screen, Owner Notifications Screen, Update profile screen.

פירוט על תפקיד כל מסך:

מסך משתמש רגיל:

1. Login Page
עמוד הת欢迎您 שבו המשתמש מזין אימייל וסיסמה, עם אפשרות לעבור להרשמה או לשחזר סיסמה.

Register Page .2

עמוד ייצרת חשבון חדש, שבו המשתמש מזין פרטים בסיסיים כמו שם, אימייל וסיסמה כדי לפתח משתמש במערכת.

Home Page .3

עמוד הבית המציג רשימת אירועים זמינים. כאן המשתמש יכול לעיין באירועים, לפתח פרטי אירוע, לבצע חיפוש ולהפעיל סינונים.

Event Details Page .4

עמוד המציג פרטים מלאים על אירוע מסוים - תיאור, מקום, תאריך, מושבים פנויים ועוד. אפשר להירשם לאירוע או לבטל הרשמה.

User Profile Page .5

עמוד פרופיל אישי שבו המשתמש יכול לעורך את פרטיו, לעדכן רדיוס חיפוש מועדף, ולבחור ז'אנרים מועדפים.

My Event Page .6

עמוד המציג למשתמש את כל האירועים אליו הם הוא נרשם. אפשר לפתח פרטי אירוע או לבטל הרשמה.

Notification Page .7

עמוד המציג התראות שהמערכת שלחה למשתמש: ביטול אירוע, שינוי בפרט אירוע, סולד אוט, מקום שהתפנה וכדומה.

מסך בעל אירוע:

Owner Dashboard Page .8

עמוד ניהול עבור בעל אירוע, המציג את כל האירועים שייצר ומאפשר לעבור לניהול האירועים השונים.

Create Event Page .9

עמוד ייצרת אירוע חדש - כולל טופס להזנת שם אירוע, תיאור, תאריך, שעה, מקום וכתובת מושבים.

Edit Event Page .10

עמוד לעדכנת אירוע קיימ, המאפשר לעדכן פרטים או לבטל את אירוע לחלווטן.

Owner Notifications Page .11

עמוד המציג לבבעל אירוע התראות כגון: סולד אוט, מושבים שהתפנו, שינויים הקשורים לנרשמים ועוד.

Update profile page .12

עמוד המציג את פרטי ניהול אירועים אותם הכנס כשןרשם, ומאפשר לו כעת לעורך את הפרטים (כמו למשל שם, תמונה פרופיל או כתובות).

זרימת ניוט למשתמש רגיל:

1. Login Page → Home Page
 לאחר שהמשתמש מזין אימייל וסיסמה תקפים ולוחץ על כפתור "Log In", המערכת מבצעת אינומת מול Firebase Authentication.
 אם ההתחברות הצליחה, המשתמש מועבר אוטומטית למסך הבית (Home Page).

Login Page → Register Page → Home Page
 אם המשתמש לוחץ על כפתור "Create Account", הוא עובר למסך ההרשמה (Register Page).
 לאחר הקמת חשבון חדש בהצלחה, המערכת מפנה אותו לשירות למסך הבית.

2. Home Page → Event Details Page
 כאשר המשתמש לוחץ על אירוע מסוים מרשימה האירועים במסך הבית או על אחד האיקונים המופיעים במאפה, המסמנים אירועים שקייםים, המערכת פותחת את מסך פרטי האירוע. שם הוא רואה תיאור מלא, מקום, תאריך ומספר מקומות זמינים.

3. שתי אפשרויות פעולה:
 א. Event Details → Register (הרשמה לאירוע)
 אם המשתמש לא רשום לאירוע והוא לוחץ על כפתור "Register", המערכת תבצע בדיקת זמינות מקום ועדכן מצב האירוע.
 לאחר הפעולה, המשתמש נשאר באותו מסך והכפתור משתנה ל "Cancel Registration".

ב. Event Details → Cancel Registration
 אם המשתמש לוחץ על "Cancel Registration", המערכת מוחקת את ההרשמה, מעדכנת את מספר המקומות הפנויים, המשתמש נשאר במסך, אך הכפתור חוזר להיות "Register".

4. Home Page → Profile Page
 בלחיצה על כפתור התפריט (≡) במסך הבית, נפתח תפריט ניוט צדדי.
 דרך התפריט, המשתמש יכול לבחור באפשרות "Profile", ולהגיע למסך הפרופיל שבו ניתן לעורר פרטים אישיים, לעדכן רדיוס חיפוש ולהגדיר ז'אנרים מועדפים..

5. Home Page → My Events Page → Event Details Page
 בלחיצה על הכפתור "My Events", המשתמש מגיע למסך המרכז את כל האירועים אליו הם הוא נשאים.
 שם ניתן ללחוץ על כל אירוע ולפתח שוב את מסך פרטי האירוע (Event Details Page).

6. Home Page → Notifications Page
 דרך תפריט הניוט, המשתמש בוחר באפשרות "Notifications".
 המסך מציג התראות כגון ביטול אירוע, שינוי פרטי אירוע או עדכון בזמנים מקומות.

זרימת ניוט - בעל עסק (Owner)

1. Login Page → Owner Dashboard Page
 כאשר בעל אירוע נכנס עם חשבון שמקורו לתפקיד Owner, המערכת מפנה אותו למסך הבית שלו.

Login → Register → Owner Dashboard
אם הוא לוחץ על "Create Account" ופותח חשבון חדש, ולאחר מכן מוגדר כבעל עסק - הוא מגיע ללוח הבית.

2. Owner Dashboard → Create Event Page.
כאשר בעל העסק לוחץ על כפתור "Create Event", הוא מגיע למסך יצרת אירוע חדש. שם הוא מזין פרטים (שם האירוע, תאריך, מקום, מקומות וכו') ושמור את האירוע.

3. Owner Dashboard → Edit Event Page.
לחיצה על אירוע קיים בלוח הבית, שם מופיעים האירועים אותם יצר בעל, פותחת את מסך עירית האירוע.
כאן ניתן לעדכן פרטים, לשנות מקומות, לבטל/למחוק את האירוע.
לאחר השמירה, המשתמש חוזר ללוח הבית.

4. Owner Dashboard → Owner Notifications Page.
דרך תפריט הניווט, בעל העסק בוחר באפשרות "Notifications".
המסך מציג התראות כגון:
אירוע שהגיע לsol'd אוט, אירוע שהתפנו בו מקומות, שינויים שהמערכת זיהתה ברישומים, התראות פעוליות נוספות.

5. Owner Dashboard → Update profile Page.
דרך תפריט הניווט, בעל העסק בוחר באפשרות "Edit Profile". המערכת פותחת מסך עדכון פרטים, בו ניתן לעורר פרטי בעל העסק, פרטי המקום וטיור כללי.

עיצוב ויזואלי:

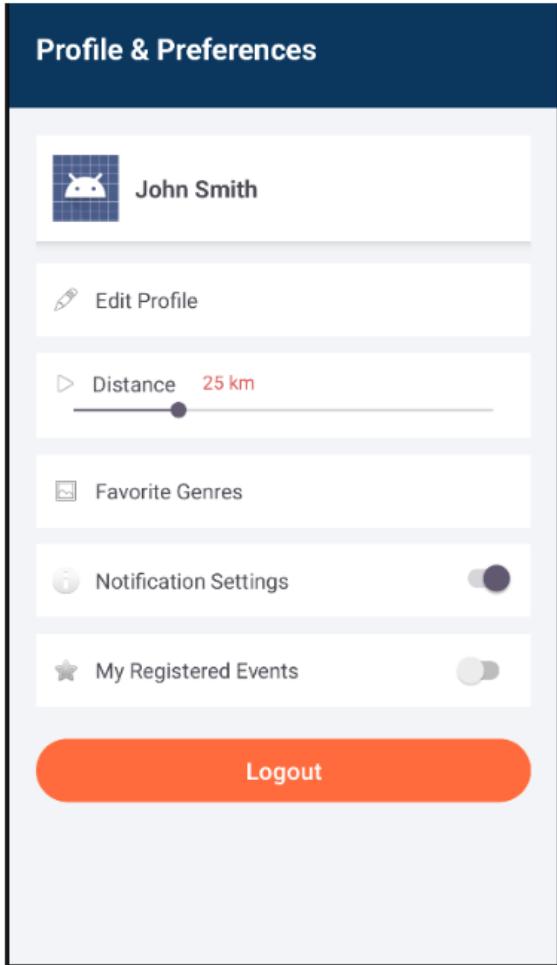
register:

The registration form is titled "register". It contains fields for "full name", "email", "Password", and "confirm password". Below these is a "Select user type:" section with two radio buttons: "User" (selected) and "Event Manager". At the bottom is a purple "register" button and a link "Already have an account? Login".

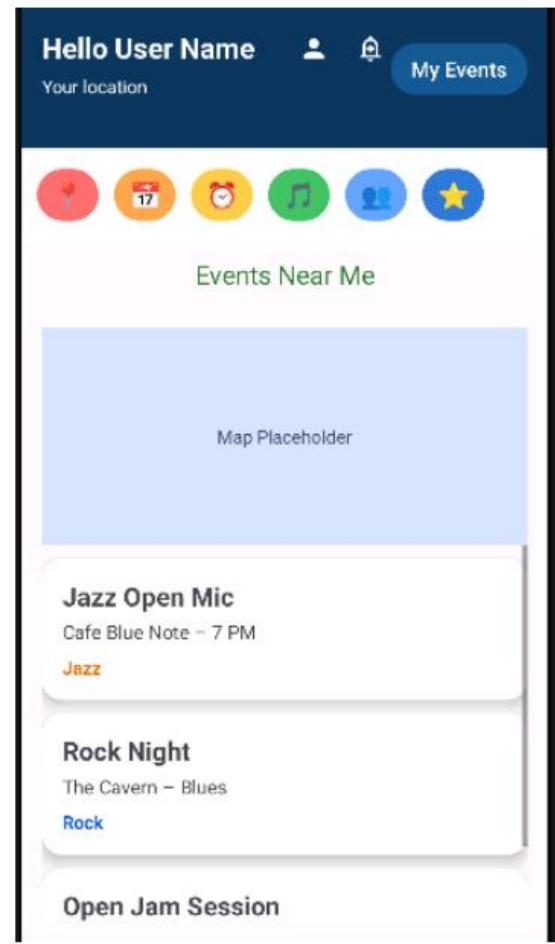
Login:

The login form is titled "Login". It has fields for "Username" and "Password", and a blue "Login" button. Below the button is a link "Not registered? Create an account".

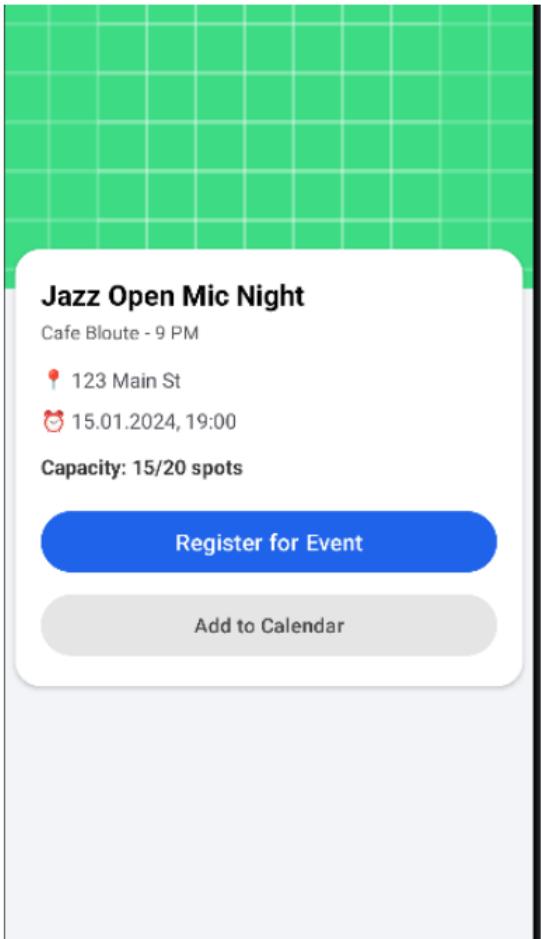
User Profile:



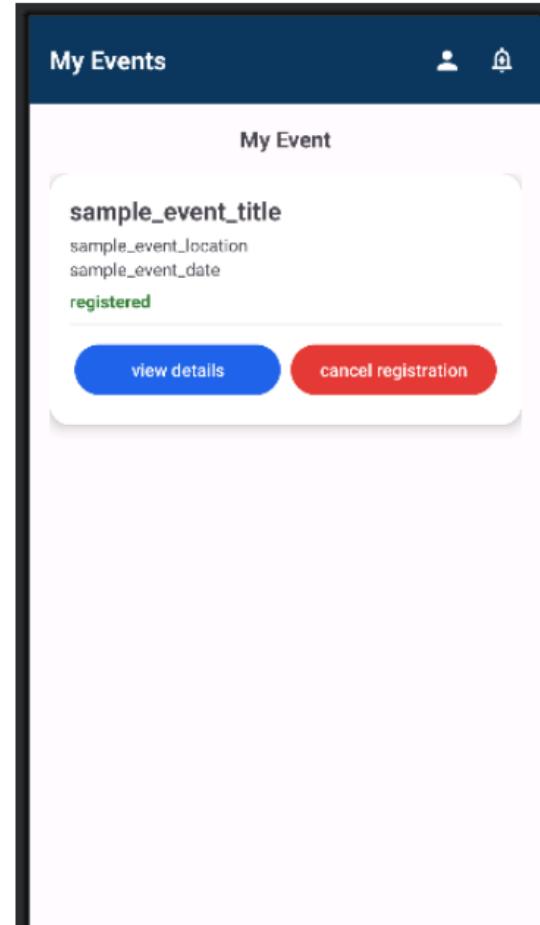
Home:



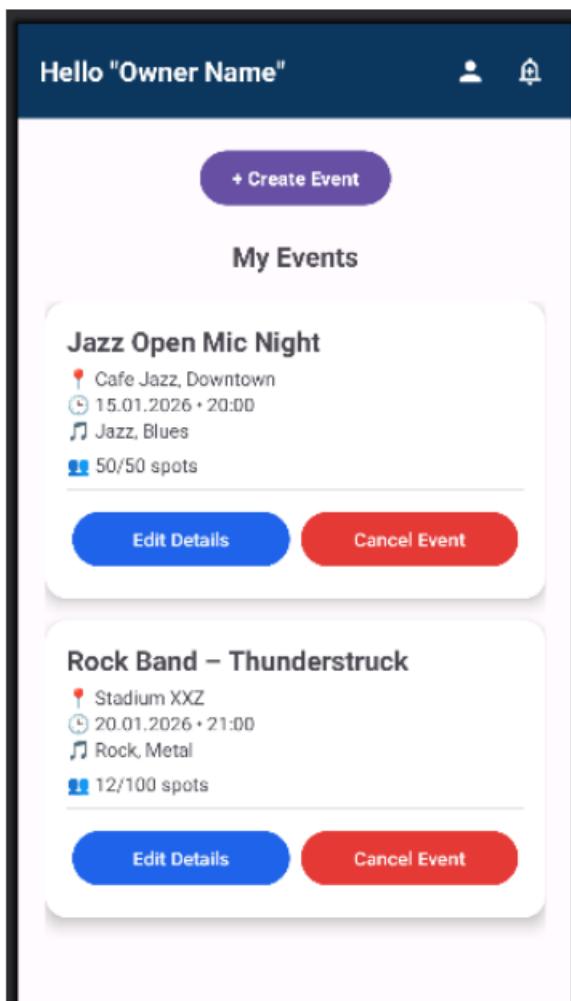
Event details:



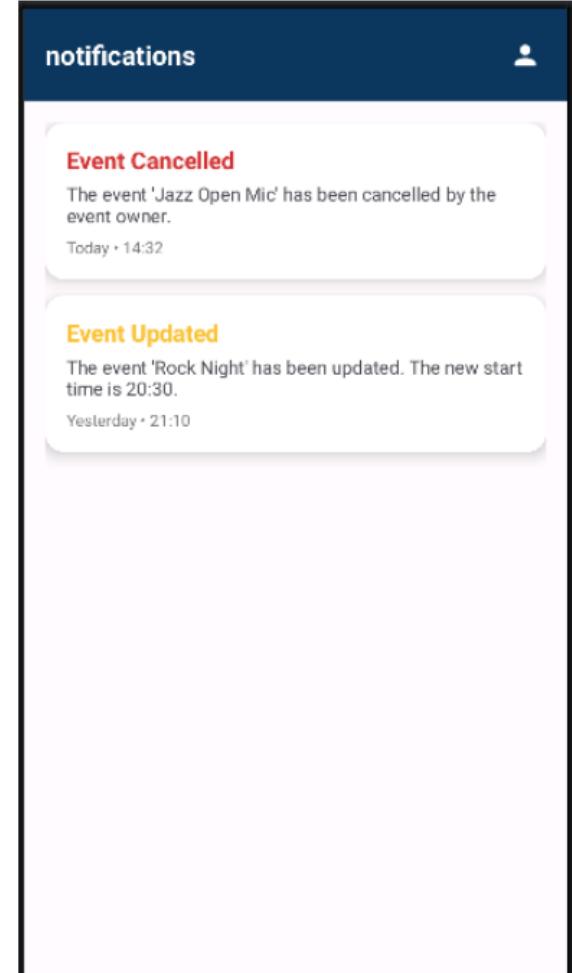
My Event (User)



Owner Dashboard:



Notifications (User)



Create Event:

Create New Event

Event Name

Location

Date

Time

Music genre

Max Capacity

Event Description

Publish Event

Cancel

update owner details:

Edit Profile

 owner name

Business / Owner Name

Email

Phone Number

Address

Save Changes

Notification (owner)

notifications

Event Sold Out
Your event 'Jazz Open Mic' has reached full capacity (sold out).
Today • 13:10

Seats Available
Several seats have opened for your event 'Rock Night' following cancellations.
Yesterday • 18:45

Event Updated
You updated details for 'Acoustic Night'. All registered users were notified.
05/01/2025 • 16:57

Edit Event:

Edit Event

Event Title

Event Description

Location

Date (DD/MM/YYYY)

Time (HH:MM)

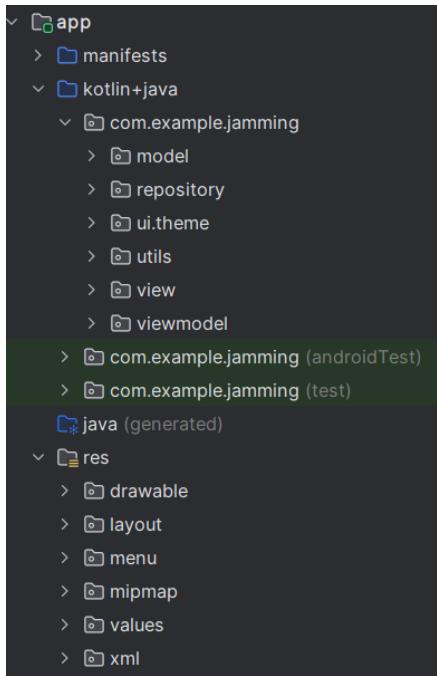
Capacity

Save Changes

Delete Event

Task Details: Design Patterns for Android Studio

בפרויקט בחרתי במבנה תיוקן שմבוסס על עקרונות של **הפרדת אחריות וארქיטקטורה שכביתית**, במטרה לשמר על קוד קריין, מודולרי, וקל לתחזקה ולהרחבה.



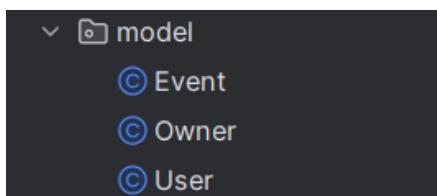
מבנה כללי של הפרויקט:

בתיקנית קם נמצאים כל רכיבי האפליקציה:

manifests מכיל את `AndroidManifest.xml` שבו מוגדרים רכיבי האפליקציה,,`Activities`,`services` וכו'.

Kotlin+java מכיל את קוד המקור של האפליקציה.

Res מכיל את כל משאבי ה `xml` (טמונות, צבעים, מחרוזות וכו').



מבנה קוד המקורי מכיל תמי תיוקנות:

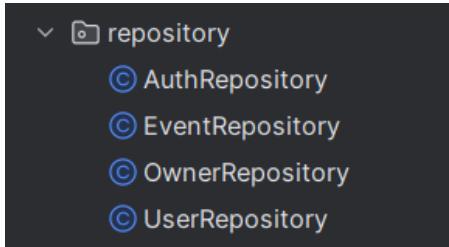
model

תיקיה זו מכילה את מחלקות המודל:

משתמשים (`User`), אירועים (`Event`) וב בעלי אירועים(`Owner`)

אלן מחלקות שמייצגות ישות לוגיות באפליקציה, לרוב עם שדות, בנאים ו-`getters` הן אינן תלויות בהן או בסיסד הנתונים.

בחירה במבנה זה מאפשרת שימוש אחיד בנתונים בכל שכבות האפליקציה.
בנוסף, המבנה יוצר חלוקה נוחה של עבודה ביןינו נפרט על כך בהמשך.



repository

תיקייה זו אחראית על **גישה לנוטונים**:

תקשורות עם Firebase (Authentication, Firestore) שליפת
נתונים, שמירה ועדכון שלהם.

ה- **repository** ניגש **שירות** למסד הנתונים, אלא דרך שכבה זו.

AuthRepository

אחראי על פעולות אימות משתמשים, (Authentication) כגון: התconnexion(Login) הרשמה
(Register) ניהול משתמש מחובר(FirebaseAuth)

UserRepository

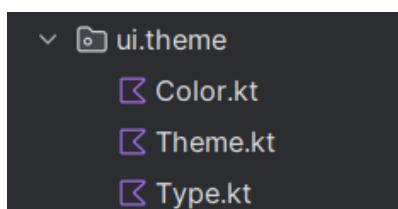
אחראי על ניהול נתונים משתמשים כגון: שליפת פרטי משתמש מ-**Firestore**, חיפוש
משתמשים לפי מלאים או שמות, עדכון נתונים משתמש ועוד.

OwnerRepository

אחראי על נתונים של בעלי אירועים(Owners) כגון: שליפת פרטי Owner לפידוי, ניהול מידע
יעודי לבעלי אירועים.

EventRepository

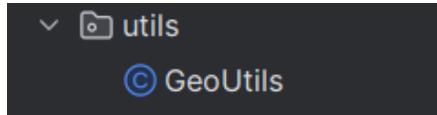
אחראי על ניהול אירועים כגון: ייצור, עדכון ומחיקה של אירועים, שליפת אירועים ממסד הנתונים
שאילות לפי מקום, בעליים, או מאפיינים אחרים.



ui.theme

תיקייה ייעודית לנושא העיצוב(Theme):

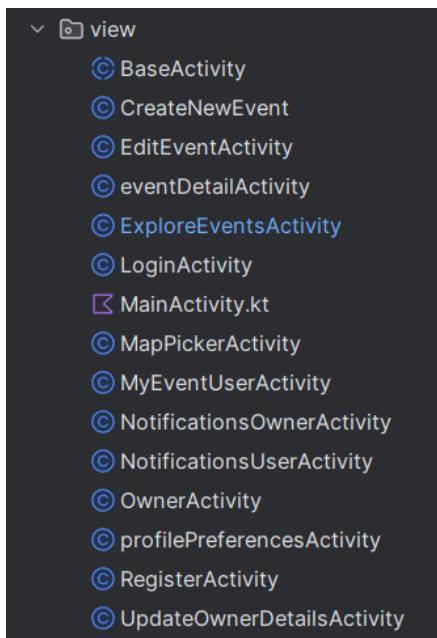
- צבעים
- טיפוגרפיה
- סגנונות כלליים



utils

תיקית עזר שמכילה מחלקות כלליות: פונקציות עזר

- **GeoUtils** מחלקת עזר גיאוגרפית מחשבת מרחק בין שני זוגות של קואורדינטות.



view

שכבה הציגה (UI)

- **Activities / Fragments** אחראית על הצגת המידע למשתמש וגובהה לאינטראקטיות (לחיצות, ניווט)

שכבה זו אינה מכילה לוגיקה עסקית או קוד של גישה לנוטונים.

BaseActivity מחלקת בסיס משותפת לכל ה- Activities, מרכזת לוגיקה והתנהגות כללית שחזרת במסכים שונים.

CreateNewEvent מסך ליצירת אירוע חדש והזנת פרטי האירוע על ידי המשתמש.

EditEventActivity מסך לעדכון פרטי אירוע קיימם.

eventDetailActivity מסך להציג פרטי אירוע בודד, כולל מידע מלא ואפשרויות אינטראקטיביות.

ExploreEventsActivity מסך חיפוש וגיליה באירועים זמינים, כולל הצגה על גבי מפה.

LoginActivity מסך התחברות משתמשים באמצעות מערכת האימרות.

MainActivity.kt נקודת הכניסה הראשית לאפליקציה וניהול הניוטו הראשוני.

MapPickerActivity מסך לבחירת מקום גיאוגרפי על גבי מפה.

MyEventUserActivity מסך להציג האירועים שהמשתמש רשום אליהם או משותף בהם.

NotificationsOwnerActivity מסך התראות ייעודי לבני אירועים.

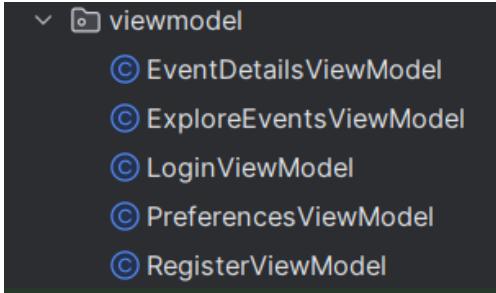
NotificationsUserActivity מסך התראות ייעודי למשתמשים רגילים.

OwnerActivity מסך ראשי לניהול פעולות של בעל אירוע.

profilePreferencesActivity מסך להגדרת והעדפת פרטי פרופיל המשתמש.

RegisterActivity מסך הרשמה משתמש חדש למערכת.

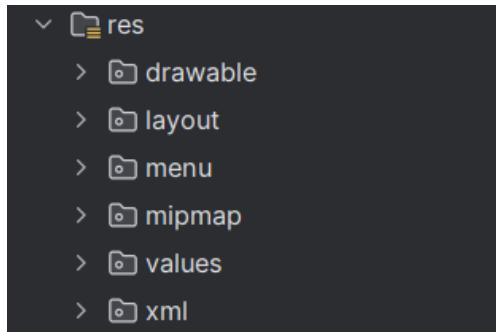
UpdateOwnerDetailsActivity מסך לעדכן פרטיים אישיים של בעל אירוע.



.viewmodel

שכבה ביןים בין ה- View-Repository

- מכילה לוגיקה שקשורה ל-UI
- מנהלת state של מסכים



תיקית – res משאבי UI

בתיקית res נשמרים כל משאבי הממשק:

קובץ XML של מסכים Layout

אייקונים ותמונות drawable

תפריטים menu

סגולות, צבעים, סגנונות values

קובץ קונפיגורציה (כמו backup_rules.xml)

בחרנו במבנה זה משום שהוא מאפשר הפרדת אחריות ברורה בין שכבות האפליקציה (UI, לוגיקה ונתונים, משורט את קריאות הקוד ותחזוקתו, מקל על הרחבה עתידית והוספת פיצרים, ומונע תלות ישירה בין מסכי המשמש לבין מקורות הנתונים, בהתאם לעקרונות פיתוח מקובלים. בנוסף, חלוקה ברורה זו מאפשרת לנו קובוצה לעבוד במקביל ובשיתוף פעולה מוביל לדריש קוד של חבר אחר.

ה יתרונות המרכזיים של הארכיטקטורה שבחרנו (MVVM + Repository) הם:

תחזוקתיות גבוהה (Maintainability): הפרדת אחריות ברורה – ה- view מציג UI בלבד,

ה-.viewmodel מחזיק לוגיקה ומצב, וה- repository מטפל בנתונים (Firebase/DB). כך שניים ב- UI כמעט לא "שובר" שכבות אחרות ולהפוך.

סקילבilities (Scalability): קל להוסיף מסכים/פיצ'רים חדשים כי יש תבנית עבודה קבועה : Activity/Fragment → ViewModel → Repository → Data. מונע "ספגטי" שכמות הקוד גדלה.

בדיקות קלות יותר (Testability): לוגיקה נמצאת ב- ViewModel/Repositories ולא בתוך Activities, וכך ניתן לבדוק אותה בצורה נקייה יותר בלי להסתמך על UI.

ביטחונות וחווית משתמש (Performance/CX): ה- UI נושא "רזה" ולא חוסם את ה- main thread גישה ישירה לנתונים; בנוסף אפשר לשפר בקלות/caching מניעת קריאות כפולות דרך ה- Repository.

גמישות להחלפת תשתיות: אם בעtid מחליפים Firebase ב- API אחר DB/אחר—השינוי מתרחש בעיקר ב- repository ולא בכל המרכיבים.

חלוקת התפקידים בפרויקט בוצעה באופן ברור אך גמיש: אוור היה אחראי בעיקר על צד בעל האירוע, רעות על חווית המשתמש, ושירות על ניהול הנתונים ושכבות ה- Repository. העבודה התנהלה בשיתוף פעולה מלא, וכל אחד מחברי הצוות חרג מתחום אחוריותו העיקרי ותרם גם לחלקים נוספים בפרויקט, בהתאם לצרכים שעלו במהלך הפיתוח.