

Let's meet

סדנה בתכנות מונחה עצמים - 20586

מנחה: דני כלפון

סטודנט: מעיין קסטלר

סמסטר: 2021א'



Let's meet

תוכן עניינים

2	תוכן עניינים
3	מסמך אפיון וניתוח
3	ייעוד המערכת
3	דרישות המערכת
3	דרישות פונקציונליות
4	דרישות כלליות
4	מרכיבי המערכת
4	אפליקציה
4	מסד נתונים
4	שירותי מפות
4	משתמשי המערכת
4	מנהל המערכת
4	משתמש רגיל
4	מנהל מפגש
5	דיאגרמת תרחישים
5	ניהול משתמשים
5	ניהול חברויות
6	ניהול מפגשים
6	ניהול סוגי מפגשים
7	דיאגרמת פעילות
7	יצירת מפגש
8	מסמך תכנון ועיצוב
8	ארכיטקטורה כללית של המערכת
9	שכבות המערכת
9	שכבת גישה לנתונים – DAL
13	שכבת תצוגה – PL
18	שכבת האפליקציה – BL
20	דיאגרמת מחלקות
21	שינויים עתידיים אפשריים
21	שינויים תשתיתיים
21	שינויים אפליקטיביים
22	מדריך למשתמש

22	תיאור המערכת
22	משתמש רגיל
22	התחברות למערכת
24	חיפוש משתמשים
25	יצירת מפגש
27	חיפוש מפגש
28	מנהל המערכת

מסמך אפיון וניתוח

ייעוד המערכת

Let's meet הינה אפליקציה לקביעת מפגשים מסוגים שונים בצורה אינטרנטית. המטרה היא לאפשר לקבוע בקלות מפגשים או פעילויות קבוצתיות בין עם על ידי קביעה עם חברים ובין עם על ידי היכרות עם אנשים חדשים דרך האפליקציה.

לא מעט פעמים קורה שאנשים רוצים להיפגש לשחק כדורגל או לצאת לבר למשל אבל הם לא מצליחים למצוא קבוצה מתאימה של אנשים לפעילות הזו או שהם לא מצליחים לנהל את המפגש בצורה מסודרת. Let's meet באה לפתור את הבעיה הזאת, היא מאפשרת ליצור מפגשים או לחפש מפגשים למטרות שונות לפי קטגוריות (למשל פעילויות ספורט, משחקי קופסא, מפגשים חברתיים וכו') ונותנת יכולת לארגן ולנהל בנוחות את קבוצת האנשים שבאה לאותו מפגש.

אם למשל יש מישהו שאוהב מאוד לשחק כדורסל חובבני אבל אין לו חברים שרוצים לשחק אז Let's meet יכולה לעזור לו למצוא בקלות משחקי כדורסל בסביבתו עם אנשים בקבוצת הגיל הרלוונטית.

Let's meet מעודדת אנשים לצאת למפגשים ופעילויות, להכיר אנשים חדשים ולהפחית שעות "מבזבזות" בבית חסרות מעש.

דרישות המערכת

דרישות פונקציונליות

- האפליקציה תאפשר יכולות ניהול של משתמשי המערכת
 - הרשמה למערכת בתור משתמש חדש
 - התחברות למערכת עם משתמש קיים
 - התנתקות מהמערכת
 - חיפוש משתמשים
 - יכולת למשתמש למחוק את עצמו
 - יכולת למנהל המערכת למחוק משתמשים אחרים
- האפליקציה תנהל יחסי חברות בין משתמשים
 - יכולת להוסיף משתמש אחר בתור חבר
 - יכולת להסיר משתמש מרשימת החברים
 - יכולת להסיר קבוצת חברים מרשימת החברים
 - חיפוש מפגשים של חברים
- האפליקציה תאפשר יכולות ניהול של מפגשי המערכת
 - יצירה של מפגש חדש
 - חיפוש מפגשים ברשימה
 - חיפוש מפגשים במפה
 - הצטרפות למפגש קיים
 - עזיבה של מפגש קיים
 - ביטול מפגש שבבעלותי

- המערכת תאפשר למנהל המערכת לשלוט בסוגי המפגשים
 - יצירה של סוג מפגש חדש
 - יצירה של קטגורית מפגשים חדשה

דרישות כלליות

- Let's meet תהיה זמינה לשימוש באמצעות אפליקציה לטלפון החכם (או מכשיר חכם אחר)
- Let's meet תהיה נוחה לשימוש ואינטראקטיבית
- כל פעולה ב-Let's meet תתעדכן לכל שאר המשתמשים שמחוברים לאינטרנט

מרכיבי המערכת

אפליקציה

תוכנה שתתקין על המכשיר החכם של משתמשי המערכת (לקוחות, מנהלים וכו') ותאפשר בעצם להשתמש במערכת בצורה פשוטה ונוחה.

האפליקציה היא בעצם המוצר שאותו הלקוחות מקבלים, יש לה ממשק גרפי נוח ולא צריך ידע טכנולוגי מקדים בשביל להשתמש בה.

מסד נתונים

מקום אחסון ברשת האינטרנט שבו נשמר כל המידע של המערכת למשל מפגשים, משתמשים, קטגוריות וכו', ממנו מתבצעות השליפות של המידע שמאפשרות את השימוש באפליקציה.

מסד הנתונים הוא הלב של המערכת והוא זה שמאפשר להעביר את המידע בין המשתמשים, כל המידע שמוצג על המסך נשלף ממנו.

רק למנהל הערכת יש גישה ישירה למסד הנתונים והוא אחראי על מבנה המידע שלו.

שירות מפות

שירות חיצוני שמאפשר עבודה עם מפות, מאפשר לחפש מפגשים לפי מיקום.

משתמשי המערכת

מנהל המערכת

האחראי על המערכת, הוא מוגדר מראש ויש לו הרשאות לראות את כל המידע של המערכת. רק מנהל המערכת יכול להסיר משתמשים מהמערכת (אם עשו בה שימוש לרעה למשל) ולהוסיף סוגי מפגשים חדשים וקטגוריות חדשות של מפגשים.

משתמש רגיל

משתמש פשוט במערכת, אין לו השפעה על איך שהיא עובדת. יכול לעשות פעולות פשוטות כמו לחפש מפגשים או ליצור מפגשים חדשים (ואז יש לו הרשאות של מנהל מפגש עבור המפגש שלו).

בשביל להיות משתמש מסוג זה פשוט צריך להוריד את האפליקציה ולהירשם, בכל שלב המשתמש יכול למחוק את עצמו מהמערכת (לא יכול למחוק אחרים).

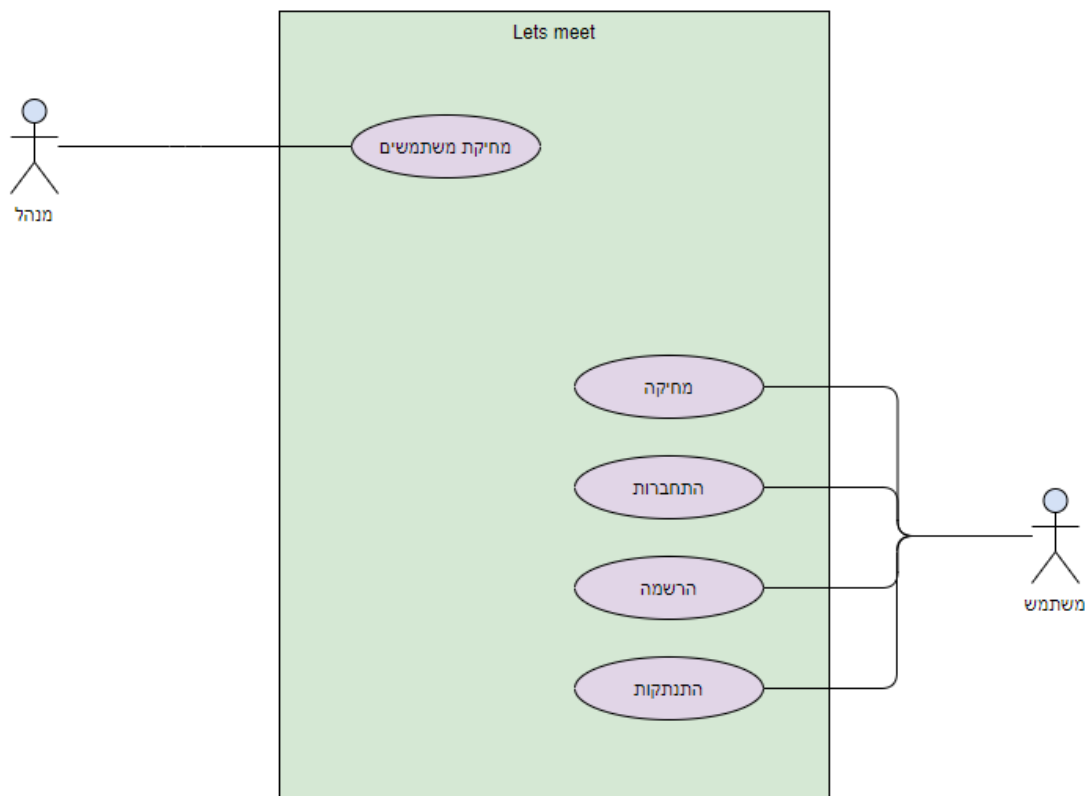
רוב המשתמשים הם מסוג זה והם נקראים גם הלקוחות של המערכת, המטרה של המערכת היא לספק את צרכיהם.

מנהל מפגש

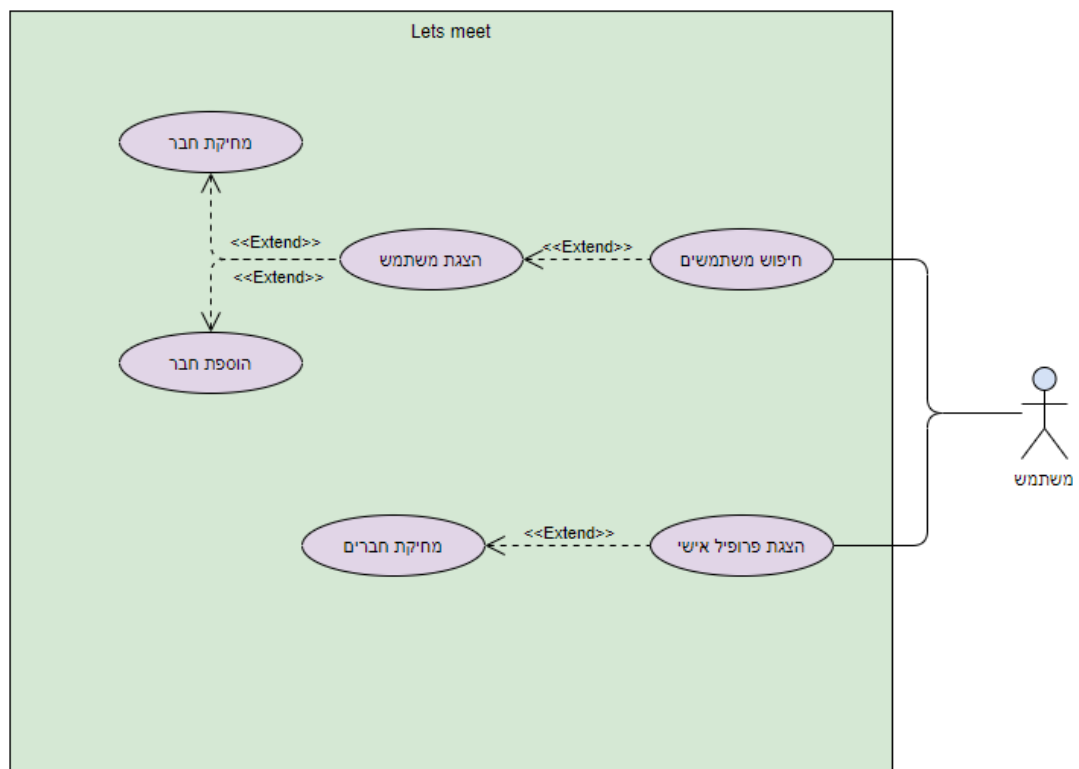
למשתמש שיצר מפגש יש הרשאות נוספות במסגרת אותו מפגש, הוא יכול להסיר אנשים מהמפגש וגם לבטל את המפגש.

אם מנהל המפגש נמחק אז גם המפגש עצמו נמחק.

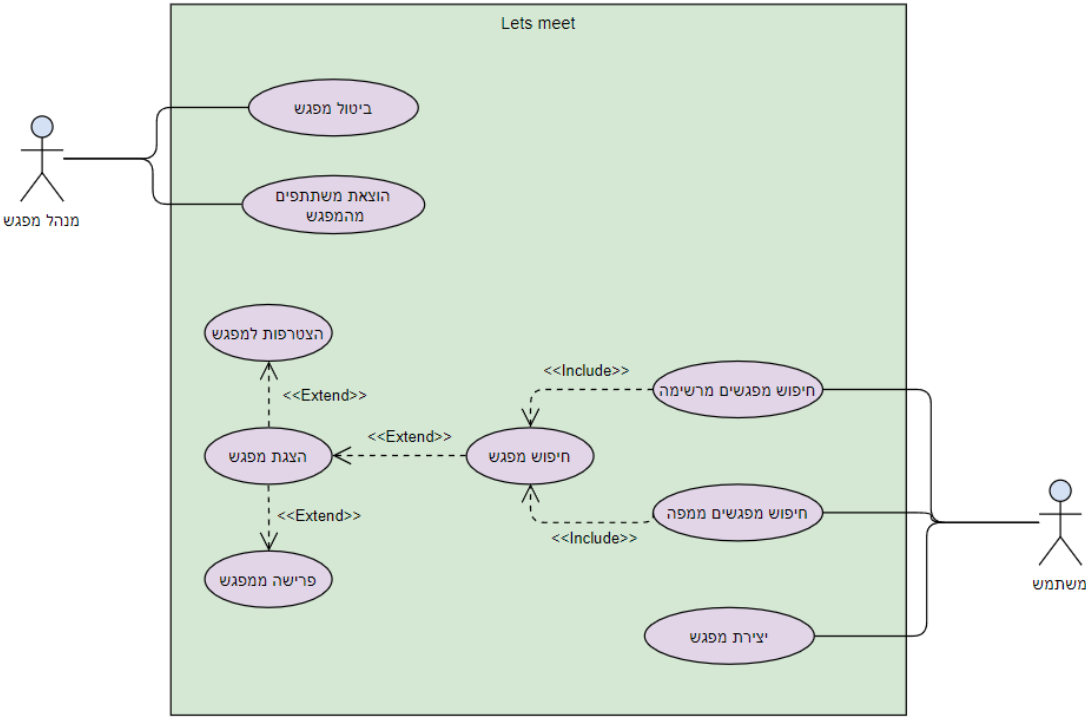
דיאגרמת תרחישים
ניהול משתמשים



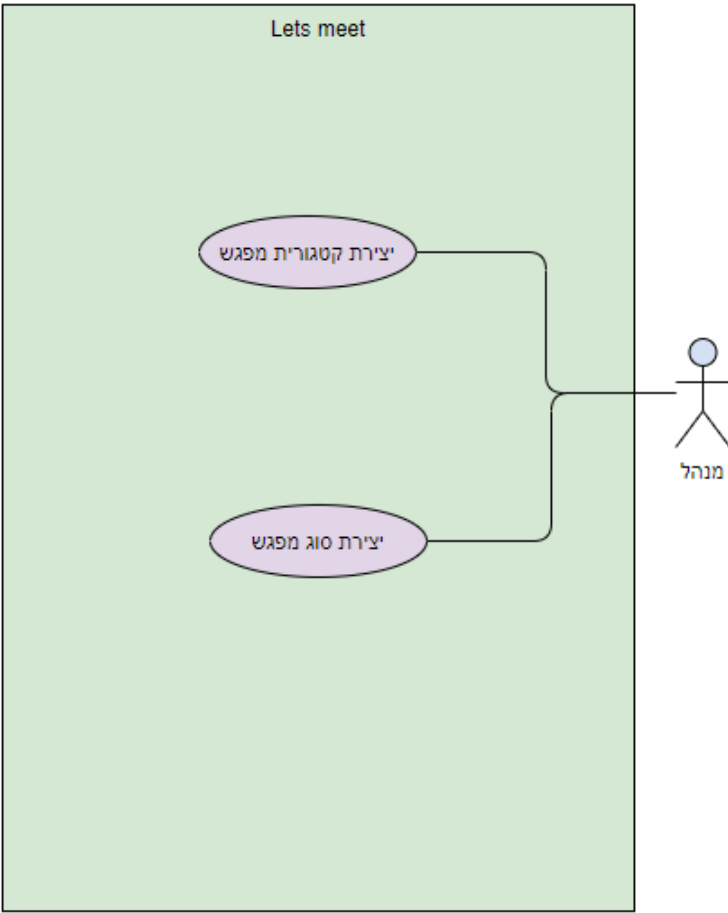
ניהול חברויות



ניהול מפגשים

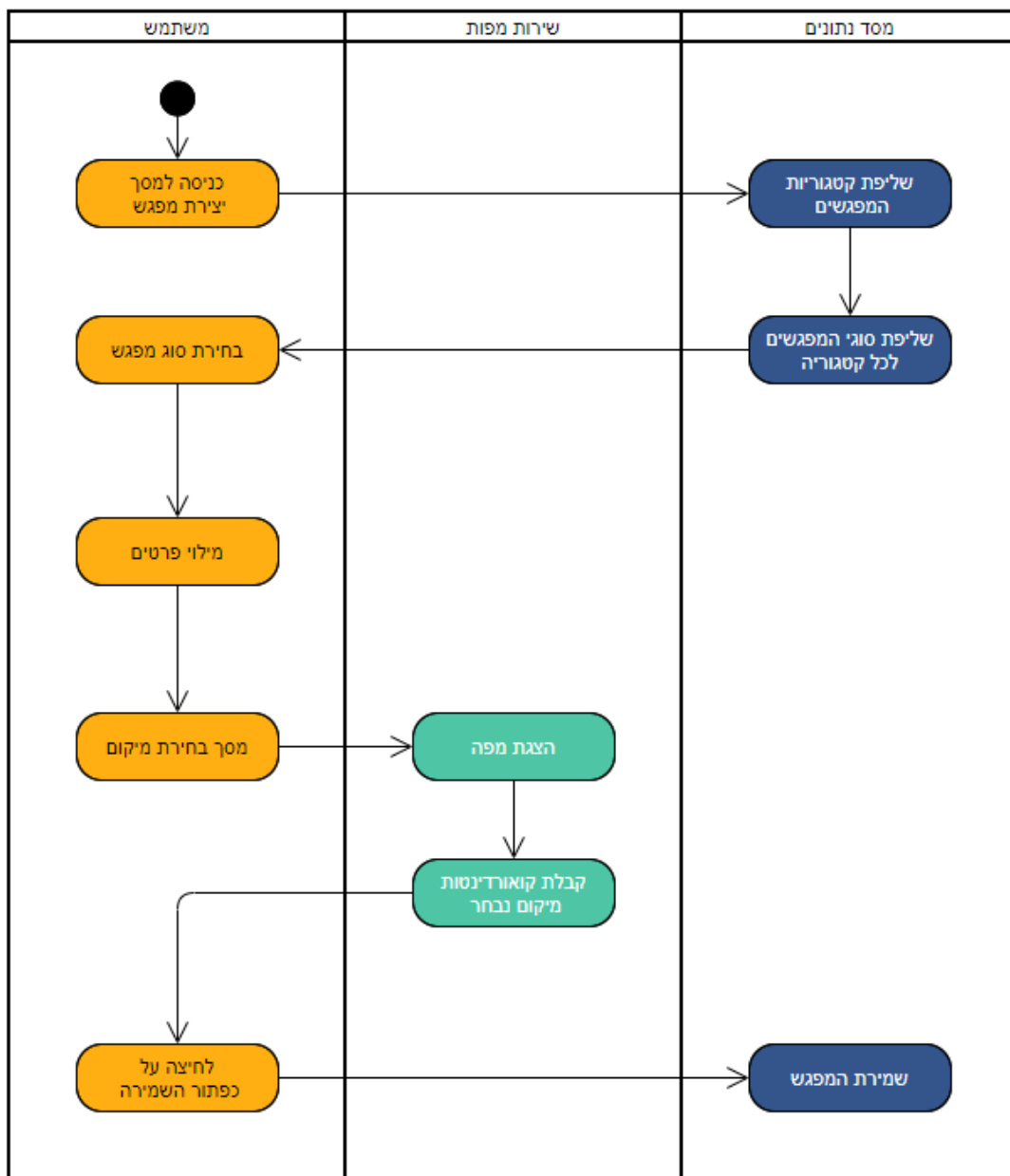


ניהול סוגי מפגשים



דיאגרמת פעילות

יצירת מפגש



מסמך תכנון ועיצוב

ארכיטקטורה כללית של המערכת

המערכת Lets meet כתובה בxamarin.forms, חבילת תוכנה שמאפשרת לבנות ב.NET. אפליקציות שיכולות לרוץ גם על mobile (ios, android) וגם על windows כך שהלוגיקה מתבצעת על ידי קוד c# והתצוגה על ידי קבצי xaml. השתמשתי ביכולת ה-shell שמאפשרת להגדיר בצורה נוחה את הנתונים השונים במערכת והמעבר ביניהם.

האפליקציה נכתבה לפי דפוס העיצוב MVVM(Model, View, ViewModel) שמאוד מתאים עבור אפליקציות Xamarin, בדפוס זה המערכת מחולקת לפי הלוגיקה של המחלקות מה שנקרא Model, הגדרות הנראות של המערכת (דפי התצוגה) View והחיבור בין התצוגה למחלקות על ידי ה ViewModel. חלוקה זו מאפשרת ליצור הפרדה בין המחלקות לדפי התצוגה כך שיהיה ניתן להחליף אותם יחסית בקלות במידת הצורך.

הקוד המשותף מחולק לכמה תיקיות:

בתיקיה הראשית יש את כל שאר התיקיות ובנוסף את הקבצים App.xaml שהיא הבסיס של האפליקציה ו-AppShell.xaml שבה כתובות הגדרות הניווט (גם פונקציונליות וגם מבחינת נראות).

בתיקיה Views יש את קבצי ה-xaml שמגדירים את הנראות של הדפים השונים באפליקציה.

בתיקיה ViewModels נמצאים קטעי הקוד שמממשים את הלוגיקה של התצוגה ומחברים בינה לבין המחלקות השונות.

בתיקיה Models נמצאות המחלקות של המערכת, כאן מוגדרות הפונקציות והערכים שלהן.

בתיקיה Data נמצאות המחלקות שמבצעות את הלוגיקה של כתיבה ושליפה ממסד הנתונים עבור המחלקות הבסיסיות של המערכת. מחלקות אלה משתמשות במחלקה מסוג singleton שמחזיקה את client שמבצע בפועל את הפעולות עם מסד הנתונים.

מסד הנתונים שבו המערכת משתמש הינו MongoDB, מדובר במסד מסוג NoSQL (אחד המובילים בעולם בתחום) שעובד עם documents (מסמכים) שיכולים להכיל ערכים מורכבים כמו אובייקטים ורשימות. המסמכים הם מסוג bson (binary json) והם נאגרים בקבוצות לפי נושאים שנקראות collections. הגמישות של mongo עוזרת בשמירת מידע על שדות מורכבים של המחלקות כמו רשימת חברים של משתמש, אובייקט שמייצג את המיקום של מפגש וכו'.

מסד הנתונים עצמו הוקם מעל [atlas](#), פלטפורמה של MongoDB להרמת מסד נתונים כשירות, בהתחברות פשוטה למערכת ולחיצת כפתור הקמתי cluster שמשמש את Lets meet ואפילו ניתן לראות נתונים על הסביבה ולערוך את המידע בצורה נוחה.

מבחינת סטנדרטים המערכת תשתמש באופן טבעי בקונבנציית שמות מוכרת עבור שפת c# שמוגדרת בעמוד הבא <https://www.c-sharpcorner.com/UploadFile/8a67c0/C-Sharp-coding-standards-and-naming-conventions>

במסד הנתונים השדות ושמות הcollection יישמרו בצורת Pascal Case.

שכבות המערכת

שכבת גישה לנתונים – DAL

מבוא

שכבה זו אחראית על העבודה מול מסד הנתונים, כמעט כל הפעולות של המערכת מתבססות בסופו של דבר על שליפה או כתיבה של נתונים.

המחלקה המרכזית בשכבה זו היא MongoDBConnection שעובדת בדפוס singleton ומחזיקה את client שמבצע גישה למסד הנתונים.

מימוש singleton מתבצע על ידי החזקת התכונה

```
private static MongoDBConnection instance
```

שמחזיקה את המופע הנוכחי של המחלקה, ביחד עם

```
public static MongoDBConnection GetInstance
```

שקוראת לבנאי רק עם instance עוד לא מוגדר.

תכונות נוספות של המחלקה:

בנאי שמבצע את ההתחברות למסד הנתונים:

```
public MongoDBConnection()
```

client לcluster של מסד הנתונים:

```
public MongoClient Client { get; }
```

data base של LetsMeet:

```
public IMongoDatabase DataBase { get; }
```

מבנה המידע

המידע שמור ב-4 collection'ים (אחד לכל מחלקה):

Users

שדה	משמעות
_id	מזהה חד ערכי
FriendsIds	רשימה שמכיל האת המזהים של החברים
Name	שם שיוצג עבור המשתמש
UserName	שם עבור התחברות
Password	סיסמא עבור התחברות
Type	סוג המשתמש, רגיל/מנהל (יכול להיות בעתיד סוגים חדשים של משתמשים)
IconURL	כתובת לתמונת הפרופיל
BornDate	תאריך לידה של המשתמש

מסמך לדוגמא:

```

_id: ObjectId("618082cb3d68541167f46220")
> FriendsIds: Array
  Name: "user"
  UserName: "user"
  Password: "user"
  Type: 1
  IconURL: "https://png.pngtree.com/png-clipart/20190917/original/pngtree-characte..."
  BornDate: 2003-11-02T00:00:00.000+00:00

```

Meetings

שדה	משמעות
_id	מזהה חד ערכי
MembersIds	רשימה שמכילה את המזהים של המשתתפים
Name	שם המפגש
MinMembers	כמות מינימלית של משתתפים לקיום המפגש
MaxMembers	כמות מקסימלית של משתתפים לקיום המפגש
MinAge	גיל מינימלי להשתתפות במפגש
MaxAge	גיל מקסימלי להשתתפות במפגש
IconURL	כתובת לתמונת המפגש
StartTime	זמן תחילת המפגש
EndTime	זמן סיום המפגש
Status	מצב המפגש (זמין, נסגר, בתהליך, נגמר וכו')
TypeId	מזהה לסוג המפגש

מסמך לדוגמא:

```

_id: ObjectId("6186fba43d6854112a2a1bcc")
> MembersIds: Array
  OwnerId: "6186faa83d6854112a2a1bcb"
  Name: "catan heros"
  MinMembers: 3
  MaxMembers: 4
  MinAge: 16
  MaxAge: 60
  IconURL: "https://m.media-amazon.com/images/I/81+okm4IpFL._AC_SL1500_.jpg"
  StartTime: 2021-11-09T00:00:00.000+00:00
  EndTime: 2021-11-10T00:00:00.000+00:00
  Status: 0
> Location: Object
  TypeId: "61802c873d6854140ffc755c"

```

MeetingsCategories

שדה	משמעות
_id	מזהה חד ערכי
Name	שם שיוצג עבור הקטגוריה
IconURL	כתובת לתמונת הפרופיל

מסמך לדוגמא:

`_id: ObjectId("61802c863d6854140ffc7552")`
`Name: "Sport"`
`IconURL: "https://png.pngtree.com/png-clipart/20190613/original/pngtree-cartoon-..."`

MeetingsTypes

שדה	משמעות
<code>_id</code>	מזהה חד ערכי
<code>Name</code>	שם שיוצג עבור הקטגוריה
<code>IconURL</code>	כתובת לתמונת הפרופיל
<code>CategoryId</code>	מזהה של קטגוריית המפגש

מסמך לדוגמא:

`_id: ObjectId("61802c873d6854140ffc7558")`
`Name: "soccer"`
`IconURL: "https://png.pngtree.com/png-clipart/20200225/original/pngtree-soccer-b..."`
`CategoryId: "61802c863d6854140ffc7552"`

UsersData

מבצעת פעולות על משתמשי המערכת במסד הנתונים.

מחזיקה את התכונות הבאות:

מופע של הcollection במסד הנתונים

```
private static IMongoCollection<User> collection
```

רשימת כל המשתמשים:

```
public static List<User> AllUsers
```

מתודה לקבלת משתמש לפי שם משתמש וסיסמא:

```
public static User GetUser(string UserName, string Password)
```

מתודה לקבלת משתמש לפי id:

```
public static User GetUser(string UserId)
```

מתודה להוספת מסמך של משתמש למסד הנתונים:

```
public static void CreateUser(User NewUser)
```

מתודה למחיקת מסמך של משתמש ממסד הנתונים:

```
public static void RemoveUser(User UserToRemove)
```

מתודה לעדכון מסמך של משתמש במסד הנתונים:

```
public static void UpdateUser(User User)
```

MeetingsData

מבצעת פעולות על המידע של פגישות במסד הנתונים.

מחזיקה את התכונות הבאות:

מופע של collection במסד הנתונים

```
private static IMongoCollection<Meeting> collection
```

רשימת כל המפגשים:

```
public static List<Meeting> AllMeetings
```

קבלת כל המפגשים של משתמש:

מציאת מפגש לפי Id:

הוספת מסמך של מפגש למסד הנתונים:

```
public static void CreateMeeting(Meeting m)
```

מחיקת מסמך של מפגש ממסד הנתונים:

```
public static void RemoveMeeting(Meeting MeetingToRemove)
```

עדכון מסמך של מפגש במסד הנתונים:

```
public static void UpdateMeeting(Meeting Meeting)
```

מחיקת מפגשים של משתמש מסוים:

```
public static void RemoveMeetingByOwner(string OwnerId)
```

חיפוש מפגשים לפי מסננים:

```
public static List<Meeting> SearchMeetings(bool OwnedByMe, bool OwnedByFriend,  
bool Member, MeetingStatus status, User CurrenUser)
```

MeetingCatogriesData

מבצעת פעולות על המידע של קטגוריות במסד הנתונים.

מחזיקה את התכונות הבאות:

מופע של collection במסד הנתונים:

```
private static IMongoCollection<MeetingCategory> collection
```

רשימת כל הקטגוריות:

```
public static List<MeetingCategory> AllMeetingCategories
```

שליפת קטגוריה לפי id:

```
public static MeetingCategory GetCategory(string id)
```

קבלת קטגוריה ראשונה ברשימה (לברירת מחדל):

```
public static MeetingCategory GetFirst()
```

יצירת קטגוריה במסד הנתונים:

```
public static void CreateMeetingCategory(MeetingCategory m)
```

MeetingTypesData

מבצעת פעולות על המידע של סוגי מפגשים במסד הנתונים.

מכילה את התכונות הבאות:

מופע של collection במסד הנתונים:

```
private static IMongoCollection<MeetingType> collection
```

שליפת סוג לפי id:

```
public static MeetingCategory GetCategory(string id)
```

קבלת סוג ראשון ברשימה (לברירת מחדל):

```
public static MeetingCategory GetFirst()
```

יצירת מסמך של סוג מפגש במסד הנתונים:

```
public static void CreateMeetingCategory(MeetingCategory m)
```

שכבת תצוגה – PL

מבוא

שכבת התצוגה אחראית על הדרך שבה הלקוחות משתמשים באפליקציה, היא מגדירה ממשק משתמש גרפי (GUI) ששואף לתת חווית שימוש מיטבית במערכת.

מאחורי הקלעים כל מה שקורה בשכבת התצוגה מתממשק לשכבת הלוגיקה ושכבת הנתונים (שהיא זו שגורמת לפעולות שהמשתמש עושה לעדכן בפועל את המידע במסד הנתונים).

שכבה זו מכילה את הViewModels ואת קבצי הxaml שמגדירים את האובייקטים (controls) שיופיעו על המסך, המיקום שלהם, האינטראקציה שלהם עם שכבת הלוגיקה (על ידי data binding) וכו'. בנוסף לאובייקטים הבסיסיים המובנים בxamarin השתמשתי גם בהרחבה בשם Xamarin Community Toolkit שנותנת יכולת נוספות.

הקישור לשכבת הלוגיקה מתבצע בקבצי xamal.cs שנקראים גם code behind ובדרך כלל מכילים רק את הקישור לViewModel הרלוונטי (במצעות BindingContext).

הViewModels נמצאים איפשהו באמצע בין שכבת התצוגה לשכבת הלוגיקה כי מצד אחד הוא לא מגדיר את הנראות של דפי האפליקציה והוא מבצע פעולות לוגיות ומצד שני הייעוד שלו זה לנהל את המידע שמוצג בדפים ולהתמודד באופן ישיר עם פעולות המשתמש.

חשוב לשים לב בשכבה זו לinterface בשם INotifyPropertyChanged שבאמצעותו נאמר לעדכן אובייקטים בUI לפי שינויים בתכונות הרלוונטיות אליהן.

App.xaml – עמוד האפליקציה

העמוד שמגדיר את האפליקציה.

אין בו עמודים של המערכת אבל תחתיו יושבים resources הרוחביים של האפליקציה, במקרה זה datatemplates שמגדירים איך פריטים עם מבנה מידע מסוים צריכים להיות מוצגים (משתמשים, מפגשים וכו').

code behind של העמוד מאותחלת האפליקציה.

AppShell.xaml – הגדרות ניווט

עמוד זה מגדיר את סרגלי הניווט שיופיע ברוב העמודים באפליקציה, הוא מכיל את הקישורים לעמודים הראשיים לפי סדר ואת הנראות שלהם (icon, כותרת וכו').

code behind של העמוד מוגדרים נתיבים נוספים של עמודים במערכת.

MainViewModel.cs

מחלקה שמכילה פונקציות ומשתנים שרלוונטיים לכל הviewmodels.

היא עובדת בדפוס של singleton באמצעות מופע של עצמו שמאותחל לNull:

```
private static MainViewModel instance = null;
```

ותכונה בשם GetInstance שמייצרת את instance במקרה שהוא לא הותחל.

המשתנה המרכזי שהמחלקה מחזיקה הוא המשתמש המחובר הנוכחי:

```
public User CurrentUser { get; private set; }
```

ופונקציות שמבצעות עליו פעולות כמו:

```
public void Login(string UserName, string Password)
```

שמקבלת שם משתמש וסיסמא ובמקרה שפרטי ההתחברות נכונים משנה את CurrentUser ובמקרה שהתחבר מנהל מוספיה עמודים של מנהלים.

```
async public void Logout()
```

שמבצעות התנתקות וחזרה לעמוד ההתחברות.

```
public bool IsLoggedIn()
```

שבדורת האם כרגע מחובר משתמש כלשהוא.

AboutPage.xaml – עמוד אודות

עמוד זה מכיל מידע אודות האפליקציה עם הסברים על המסגרת בה היא נבנתה ודרכים ליצירת קשר עם המפתח. מדובר בעמוד די סטטי שלא מתממש לרכיבים אחרים באפליקציה ויכול לעמוד בפני עצמו (ולכן הוא פחות מעניין). בהרבה אפליקציות יש עמוד כזה בשביל לעזור למשתמשים להבין דברים על המערכת ולדעת למי לפנות במידת הצורך.

LoginPage.xaml – עמוד התחברות

עמוד זה הוא הראשון שנפתח שנכנסים לאפליקציה, הוא מציג את הלוגו של Lets meet ומאפשר להזין שם משתמש וסיסמא בשביל להתחבר למערכת.

משתמשים לא רשומים יכולים לעבור ממנו לעמוד ההרשמה.

LoginPageViewModel.cs מכיל משתנים עבור הסיסמא ושם המשתמש שמשתנים לפי הנתונים שמהשתמש מכניס (data binding ל-Entryanudsr בxaml).

בנוסף viewmodel מכיל commands עבור הכפתורים, הcommand של כפתור המעבר לעמוד הרשמה פשוט עובר לעמוד ההרשמה באמצעות Shell.Current.GoToAsync וזה של עפתור ההתחברות מבצע התחברות דרך MainViewModel ואז עובר לפרופיל במקרה של הצלחה.

RegistrationPage.xaml – עמוד הרשמה

עמוד עם טופס הרשמה שמאפשר להכניס פרטים רלוונטיים וליצור לפיהם משתמש, התמונה שבה המשתמש יבחר כתמונת פרופיל תופיע בחלק העליון של העמוד (בהנחה והכניס קישור תקין לתמונה) כאשר יש תמונת ברירת מחדל.

עמוד זהה דומה בעיצוב שלו לטופס ההתחברות כי הם באים ביחד כחלק מפעילות הכניסה למערכת.

RegistrationPageViewModel.cs מכיל משתנים עבור על הפרטים שצריך בשביל ליצור משתמש:

```

public string UserName { get; set; }
public string Password { get; set; }
public string Name { get; set; }
public DateTime BornDate { get; set; } = DateTime.Now.AddYears(-18);
public string IconURL { get; set; } =
"https://osxlatitude.com/uploads/monthly_2019_10/pngtree-vector-user-young-boy-
avatar-icon-png-image_1538408.thumb.jpg.2ec631dd5e0e029b2c5a13fe37f2c122.jpg";

```

כאשר יש לחלקם ערכי ברירת מחדל.

בנוסף מכיל פקודות לכפתורים כאשר `public ICommand SignIn { get; }` עובר בחזרה לעמוד ההתחברות ו- `public ICommand SignUp { get; }` יוצר user חדש במערכת ומתחבר אליו.

UserDetailPage.xaml – הצגת משתמש

עמוד זה משמש להצגת משתמש יחיד.

הוא המצא בשימוש גם שלקוח נכנס לפרופיל שלו וגם שמסתכלים על משתמש אחר במערכת.

העמוד מכיל מידע על המשתמש (שם, תמונה, תאריך לידה וחברים) ופעולות שאפשר לבצע, אם מסתכלים על משתמש אחר אפשר לצרף אותו כחבר או להסיר אותו מרשימת החברים (אם הוא נמצא בה כמובן) כאשר מנהל יכול גם למחוק את המשתמש. אם משתמש מסתכל על העמוד שלו הוא יכול לבצע logout ולחזור למסך ההתחברות, למחוק את עצמו מהאפליקציה ולהסיר מרשימת החברים קבוצה של חברים.

UserDetailViewModel.cs מכיל תכונות רלוונטיות לעמוד:

`public string UserId` מכיל את id של המשתמש המוצג כאשר כברירת מחדל נבחר המשתמש המחובר נוכחי.

`User User` `public` האובייקט של המשתמש, מתעדכן ש-`UserId` משתנה

`public bool IsLoggedInUser` האם המשתמש המוצג הוא המשתמש המחובר (רלוונטי לכפתור מחיקת המשתמש)

`public bool IsFriend` האם המשתמש במוצג הוא חבר של המשתמש המחובר (רלוונטי לכפתור ההוספה כחבר/מחיקה מרשימת החברים)

`public bool IsAdmin` האם המשתמש במחובר הוא מנהל (ואז יש לו יותר הרשאות)

בנוסף מכיל משתנים עבור היכולת של בחירת קבוצת חברים והסרה שלהם:

`public ObservableCollection<object> SelectedObjects { get; set; }` של collection משתנים שיכולים להיות מוצגים על ידי `CollectionView` ומייצגים את אלה המסומנים

`public List<User> SelectedMembers { get; set; }` הייצוג של האובייקטים המסומנים כמשתמשים מהמחלקה `user`.

ה `ViewModel` מכיל גם פונקציות עבור הכפתורים בעמוד.

UserPage.xaml – הצגת משתמשים

עמוד ראשי זה מאפשר להסתכל על משתמשים במערכת ולבצע עליהם סינונים וחיפוש.

בחלק העליון של הקובץ מוגדר resource שמשנה את הצבע של רקע תפריט הניווט, לכל עמוד ראשי מוגדר צבע אחר וזה עוזר להבין איפה אתה נמצא וגם משפר את חווית השימוש.

בעמוד מוגדר חלון חיפוש `<Shell.SearchHandler>` שמימנתי בשם `UsersSearchHandler` שיועד לבצע חיפוש על כל המשתמשים ולהציג את אלה שהמחרוזת שהמשתמש הזין היא תת מחרוזת מהשם שלהם.

החלק המרכזי העמוד ממומש בעזרת `CollectionView`, אובייקט שיועד להציג פריטים מחלקה מסוימת (לפי `data template` נתון) כאשר מוגדר לו בעת לחיצה לעבור לעמוד של אותו משתמש.

בתחתית העמוד יש `StackLayout` פשוט שמכיל את סרגל הסינון.

`UsersViewModel.cs` מכיל את הcollection של המשתמשים המוצגים שמשתנה בהתאם לסינון ואת המשתנים הרלוונטיים לסינון:

```
public bool IsFriend הם להראות רק חברים
public int MinAge גיל מינימלי של משתמשים מוצגים
public int MaxAge גיל מקסימלי של משתמשים מוצגים
```

CreateMeetingPage.xaml – יצירת מפגש

עמוד ראשי שמאפשר ליצור מפגשים חדשים.

בראש העמוד ניתן לבחור את סוג המפגש לפי רשימה של קטגוריות שכל קטגוריה נפתחת לרשימה של סוגי מפגשים, המימוש של רכיב זה הוא על ידי `CollectionView` של הקטגוריות שמשתמש ב `Expander` של `Xamarin Community Toolkit` שמאפשר לפתוח ולסגור רכיבי תצוגה כך שמה שמוצג הוא עוד `collectionview` של הסוגים באותה קטגוריה. כברירת מחדל שינוי סוג המפגש משנה את תמונת המפגש לזו של הסוג אבל אפשר לשנות אחר כך למה שרוצים.

שאר העמוד מכיל קלטים של השדות הרלוונטיים ליצירת מפגש כאשר יש שימוש גם בתאריך באמצעות `DatePicker` והכנסת שדות מספריים על ידי הגדרת `Keyboard="Numeric"`.

בתחתית העמוד יש כפתור שמאפשר לבחור מיקום למפגש על ידי מעבר לעמוד אחר במערכת כך שהמידע מוחזר על ידי `QueryAttributes` שמוגדרים `IQueryable interface`. בנוסף מוגדר שעם ההופעה של העמוד תתבצע בדיקה לגבי מיקום המכשיר כך שכברירת מחדל מיקום המפגש יהיה המיקום הנוכחי.

ב `ViewModels` מוגדרים משתנים רגילים עבור כל הקלטים שלחלקם יש ערך ברירת מחדל (נגדי זמן ההתחלה הוא מחר) כאשר בנוסף יש משתנה עבור סוג המפגש ששינוי שלו משפיע על תמונת המגש ויש לו לוגיקה של שליפת המידע רק אם חייבם בשביל לחסוך פעולות מיותרות מול מסד הנתונים.

בנוספים מוגדרים הפעולות עבור הכפתורים השונים, מעבר לעמוד בחירת מקום ויצירת המפגש עצמו שאחריו מתבצע מעבר לעמוד הצגת המפגשים.

MapGetLocationPage.xaml – עמוד בחירת מיקום

עמוד שמכיל מפה שמציגה את המיקום הנבחר על ידי שימוש ב `pin` כאשר לחיצה על מיקום אחר במפה מזיזה את `Pin`.

ב `ViewModels` מוגדר המשתנה של המיקום הנוכחי:

```
public Position CurrentPosition { get; set; } = new Position();
```

פונקציה שמתמודדת עם לחיצה על המפה ומשנה את `CurrentPosition`.

פונקציה שקוראת בהופעת העמוד ועוברת למיקום הנוכחי של המכשיר (מתוך הנחה שרוב המשתמשים יבחרו מקום מפגש שקורב למקומם).

ולבסוף פונקציה שמתרחשת על ידי שנבחר המיקום של המפגש וחוזרת לעמוד יצירת המפגש עם הפרמטרים של המיקום מעוברים על ידי `query`.

[MeetingDetailPage.xaml – עמוד הצגת מפגש](#)

מכיל מידע על המפגש (סוג, זמן, תמונה וכו') ורשימת משתתפים כאשר יש אפשרות להצטרף או לצאת מהמפגש ומנהל המפגש יכול גם לבטל אותו או להוציא משתתפים.

ההצגה של תוכנים (תווך גילאים ותווך כמות משתתפים) בוצעה על ידי `MultiBinding` עם `string` format כך שיהיה אפשר להגדיר כמו משתנים שישפיעו על טקסט של `label` אחד. בהצגה של הכפתורים יש שימוש ב-`MultiTrigger` שמאפשר להגדיר כמה `trigger` שיכלים לגוון לשינוי של הגדרות באובייקט תצוגה (כמו האם יוצג או לא) לפי `binding` למשתנה מסוים.

`MeetingDetailViewModel.cs` מכיל את המזהה של המפגש המוצג ואובייקט של המפגש ביחד עם פונקציות עבור הכפתורים ומשתנים בוליאניים נוספים (האם במפגש והאם מנהל המפגש) שעל ידם מוגדר מה יוצג למשתמש.

בנוסף `ViewModel` יודע להתמודד עם שינוי המשתתפים הנבחרים כך שיהיה ניתן להוציא אותם מהמפגש על ידי לחיצה על הכפתור המתאים.

[MeetingsListPage.xaml – רשימת מפגשים](#)

עמוד ראשי שמציג מפגשים ברשימה ומאפשר לפלטר עליהם.

בעת לחיצה על מפגש ייפתח עמוד המפגש.

המימוש ל העמוד מאוד דומה לעמוד המשתמשים בזה שהוא מבוסס על `CollectionView` של המפגשים שמוצג לפי `datatemplate` עם סרגל סינון למטה.

`MeetingsViewModel.cs` מכיל את הרשימה הרלוונטית של מפגשים שיוצגו ביחד עם משתני הפילטור והפונקציה המרכזית שמבצעת את הפילטור (שמתבצעת קריאה אליה בכל שינוי של משתנה רלוונטי).

[MeetingsMapPage.xaml – הצגת מפגשים על מפה](#)

עמוד זה מציג את המפגשים על המפה.

מגיעים אליו באותה צורה עמו רשימת המפגשים רק בשינוי `tab` בסרגל למעלה.

הוא משתמש גם ב-`MeetingsViewModel.cs` מה שמדגיש את היתרונות בשימוש ב-MVVM (נסכון של כפילות קוד), האובייקט הראשי בעמוד זה הוא מפה שמוגדר לה מקור של פריטים להצגה שהוא רשימת המפגשים המפולטרים כאשר כל מפגש מיוצג על ידי `MeetingPin` שזה `control` שיצרתי שיושר `pin` המקורי של `Xamarin.Forms.Maps` ומוסיף עוד תכונה של המפגש שהוא מייצג.

בנוסף הייתי צריך להמיר בין המחלקה `Location` שבה המחלקה `Meeting` משתמש למחלקה `Position` שבה `Map` משתמש, ההמרה בוצעה באמצעות `converter` בשם `LocationToPosistionConverter` שמוגדר ב-`ViewModel`.

[CreateMeetingCategoryPage.xaml – עמוד יצירת קטגוריית מפגשים](#)

עמוד שרק מנהל יכול להיכנס אליו ומאפשר ליצור קטגוריה חדשה של מפגשים.

נמצא `tab` תחת עמוד `admin` שמשמש את המנהל.

מכיל קלטים של תמונת הקטגוריה ושם שלה שלהם יש משתנים מתאימים ב-`ViewModel` ביחד עם עפתור שמבצע את היצירה.

[CreateMeetingTypePage.xaml – עמוד יצירת סוגי מפגש](#)

עמוד שרק מנהל יכול להיכנס אליו ומאפשר ליצור סוג חדש של מפגש.

נמצא `tab` תחת עמוד `admin` שמשמש את המנהל.

הוא מכיל רשימה של קטגוריות מפגשים שבבחרת מבניהם יוכל הסוג החדש וקלטים של שם סוג המפגש ותמונה שלו.

ViewModel מכיל את המשתנים של יצירת סוג המפגש ופונקציה שיוצרת לפיהם את סוג המפגש ועוברת לעמוד יצירת המפגש.

שכבת האפליקציה – BL

שכבה זו מכילה את הלוגיקה העסקית של המערכת, היא מכילה את מחלקות הבסיס של המערכת.

User.cs – מחלקה של משתמש

מייצגת משתמש של המערכת.

מכילה בנאי ליצירת מפעים של משתמשים.

מכילה את התכונות הבאות:

```
public string Id { get; set; }
public string Name { get; set; }
public string UserName { get; set; }
public string Password { get; set; }
public UserType type { get; private set; }
public HashSet<string> FriendsIds = new HashSet<string>();
public HashSet<User> Friends
public string IconURL { get; set; }
public DateTime BornDate { get; set; }
public int age
public void AddFriend(User user)
public bool IsFreind(User user)
public void RemoveFriend(User friend)
public void RemoveFriends(List<User> Friends)
```

בשביל להפריד בין סוגי משתמשים במערכת נשתמש ב-enum:

```
public enum UserType
{
    Admin,
    User
}
```

שמגדיר את סוגי המשתמשים האפשריים בצורה שבנויה להרחבה לסוגים חדשים בעתיד.

MeetingCategory.cs – מחלקה של קטגוריית מפגשים

מייצגת קטגוריה של סוגי מפגשים במערכת.

מכילה את התכונות הבאות:

```
public string Id { get; private set; }
public string Name { get; set; }
public string IconURL { get; set; }
public List<MeetingType> Types
collection של סוגי המפגשים
```

MeetingType.cs – מחלקה של סוג מפגש

מייצגת סוג של מפגש.

מכילה את התכונות הבאות:

```

        public string Id { get; private set; }
        public string Name { get; set; }
        public string IconURL { get; set; }
        public string CategoryId { get; private set; }
    }
    // שייך (foreign key)
    public MeetingCategory category (CategoryId לפי)

```

Meeting.cs – מחלקה של מפגש

מחלקה שמייצגת מפגש.

זוהי המחלקה הכי מרכזית שעליה כל המערכת בנויה.

יש לה בנאי שמטפל במופעים חדשים של מפגש.

מכילה את התכונות הבאות:

```

        public string Id { get; private set; }
        public string Name { get; set; }
        public int MinMembers { get; set; }
        public int MaxMembers { get; set; }
        public int MinAge { get; set; }
        public int MaxAge { get; set; }
        public string IconURL { get; set; }
        public MeetingType Type (נשמר בתור מזהה)
        public HashSet<string> MembersIds = new HashSet<string>();
    }
    // משתתפי המפגש
    public HashSet<User> Members (נלקח לפי MembersIds)
    {
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public MeetingStatus Status { get; private set; }
        public string OwnerId;
        public User Owner (נלקח לפי OwnerId)
        public Location Location { get; set; }
    }
    // סטטוס המפגש נשמר לפי enum שמייצג סטטוסים אפשריים

```

```

public enum MeetingStatus
{
    Available,
    Cancelled,
    Done,
    InProgress,
    Template
}

```

לא כל הסטטוסים יהיו בשימוש מההתחלה, חלקם נוצרו לשימוש אפשרי עתידי.

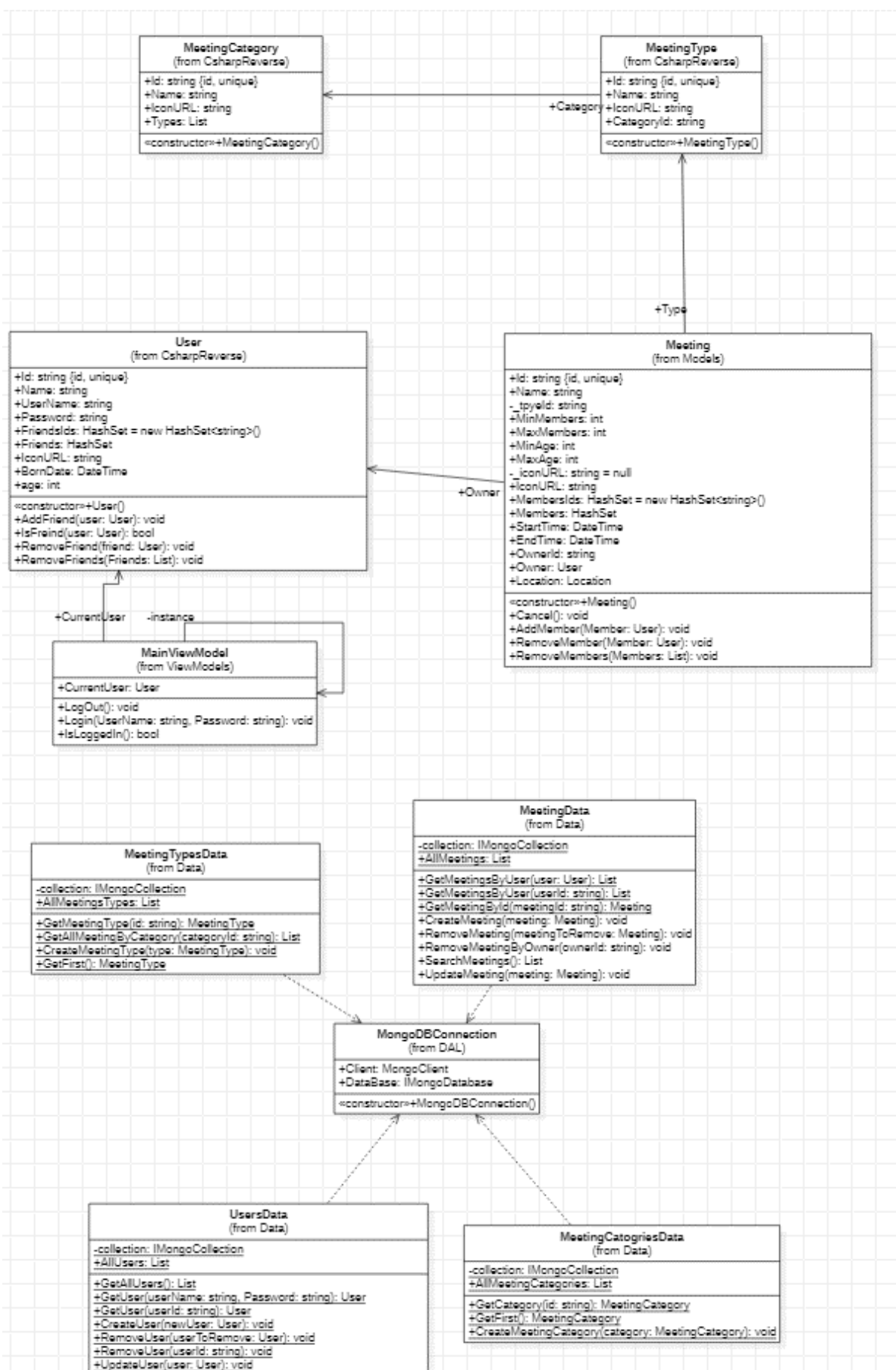
המחלקה מכילה את הפונקציות הבאות:

```

    public void Cancel()
    public void AddMember(User Member)
    public void RemoveMember(User Member)
    public void RemoveMembers(List<User> Members)

```

דיאגרמת מחלקות



שינויים עתידיים אפשריים

שינויים תשתיתיים

ישנם כמה שינויים בתשתית של המערכת שיאפשרו לה להשתפר.

ניתן להוסיף מרכיב חדש במערכת, שרת שיקבל בקשות api מהאפליקציה ויבצע את כל הפעולות שלא קשורות ישירות לתצוגה לדוגמה הכתיבה למסד הנתונים. ישיתרון אבטחתי בהוספת שרת, זה עוזר למנוע ממשתמש זדוני לגלות מהאפליקציה את צורת ההתחברות למסד הנתונים ולהקשות עליו לגלות את צורת העבודה של המערכת. בנוסף יכול להיות יתרון מבחינת ביצועים, במקרה של ביצוע פעולות מורכבות שדורשות הרבה משאבים כנראה שהרצתם על שרת חזק שמיעוד לכך תהיה מהירה יותר מהרצה על המכשיר של הלקוחות וגם תחסוך שימוש בסוללה למשל (מה שמשפר את חווית השימוש). שינוי זה ידרוש להעביר את המחלקות שפונות למסד הנתונים לאותו שרת ובמקומן לממש מחלקות שיודעו לעבוד עם הapi החדש. חוץ מעיבוד יהיה אפשר להשתמש בשרת גם לאחסון קבצים כך שתינתן אפשרות למשתמשים להעלות תמונות למערכת במקום לתת לה קישורים לתמונות ברשת.

שינוי תשתיתי נוסף שאפשר לעשות שיכול לשפר את הביצועים הוא להשתמש בcluster פרטי למסד הנתונים במקום בכזה שיתופי מרוחק, שינוי זה יכול לשפר את זמני השליפה והכתיבה של המידע.

שינויים אפליקטיביים

ישנם כמה שינויים ברמת הקוד שאפשר לבצע בשביל להעלות את הרמה של המערכת.

אפשר להוסיף חיווי יותר טוב על פעולות במערכת, לדוגמה ניסיון התחברות שנכשל יקפיץ הודעה בצבע אדום שכתוב בה login failed או לדוגמה בניסיון הצטרפות למפגש שלא מתאפשר תקפץ הסיבה לכך. החיווי יכול להיות גם על פעולות שהצליחו, הודעות כמו המפגש שלך נוצר בהצלחה יכולות לעזור למשתמש להבין שמה שהוא ניסה לעשות בוצע. מימוש תכולה זו יכול להתבצע על ידי יכולות של Xamarin בשם behaviors שמאפשרות לתת פונקציונליות לcontrols.

ישנם גם כמה יכולות שניתן להוסיף למערכת:

- אפשר לתת למנהל המערכת אפשרות גם למחוק סוגי מפגשים על ידי מימוש עמוד חדש שמציג את הסוגים ובלחיצת כפתור שולח את הid של הסוג שנמחק ולפי זה מוחק אותו ממסד הנתונים, בנוסף צריך להציג במפגשים מסוג שנמחק שהסוג לא ידוע.
- מתן יכולות לשנות פרטים של מפגש (נגיד לשנות שעה), ימומש על ידי כפתור edit שיעביר לעמוד יצירת המפגש עם כל הפרטים הנוכחיים ובמקום אפשרות ליצור חדש יתבצע עדכון לפי הid.
- הגדרת סוגי מפגש מועדפים למשתמש וחיפוש לפיהם, ימומש על ידי הוספת רשימת סוגי מפגשים תחת המחלקה של משתמש + עוד יכולות פילטור בעמוד הצפייה במפגשים
- בצפייה במפגשים לפי מפגש אפשר לשנות את הPin שיראה את תמונת המפגש, ניתן לממש על ידי כתיבת מחלקת רינדור חדשה שיורשת ממחלקת הרינדור של ה Pin המקורי
- צמצום זמני הטעינה בכניסה למערכת ומעבר בין עמודים, יכול להתבצע על ידי הכנסת קריאות אסינכרוניות לחלק מהפונקציות במערכת, בעיקר לשליפות ממסד הנתונים שיכולות לקחת זמן (פעולות io bound).

שינוי נוסף שאפשר לעשות הוא תמיכה בעמדות windows ומכשירים מבוססים ios, xamarin תומכת בפיתוח לכמה מערכות הפעלה שונות על ידי בסיס קוד משותף ומה שזה מצריך זה ביצוע התאמות בפרוייקט של כל מערכת הפעלה (הוספת תמונות, תמיכה במפות וכו').

מדריך למשתמש

תיאור המערכת

Let's meet הינה אפליקציה לקביעת מפגשים חברותיים עם אנשים אחרים. מערכת זו מתאימה לאנשים שמחפשים מפגשים להצטרף אליהם או רוצים לנהל בצורה מסודרת מפגשים בעצמם. Let's meet מאפשרת ליצור בקלות מפגשים מסוגים שונים לפי קטגוריות וגם לחפש מפגשים שנוצרו על ידי אחרים ולהצטרף אליהם.

במדריך זה נסביר איך המערכת עובדת ואיך אפשר להשתמש בה.

משתמש רגיל

התחברות למערכת

השלב הראשון בכניסה לאפליקציה הוא התחברות למערכת. משתמשים רשומים יכולים פשוט להזין את שם המשתמש שלהם ואת הסיסמא וללחוץ על כפתור log in, משתמשים חדשים יצטרכו להירשם על ידי לחיצה על כפתור register שיעבור למסך ההרשמה.

במסך ההרשמה יש למלא פרטים (שם משתמש, סיסמא, שם תצוגה, תאריך לידה וקישור לתמונת פרופיל) ואז ללחוץ על כפתור sign up (sign in מחזיר בחזרה למסך ההתחברות).




Username:

Password:

display name:

born date

icon URL

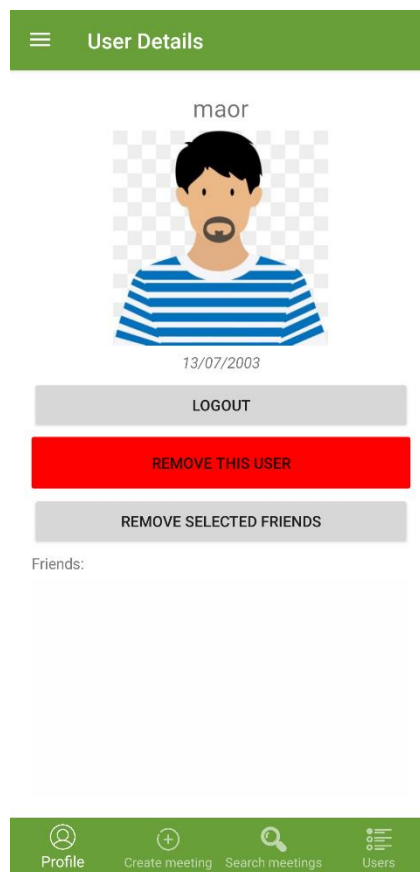
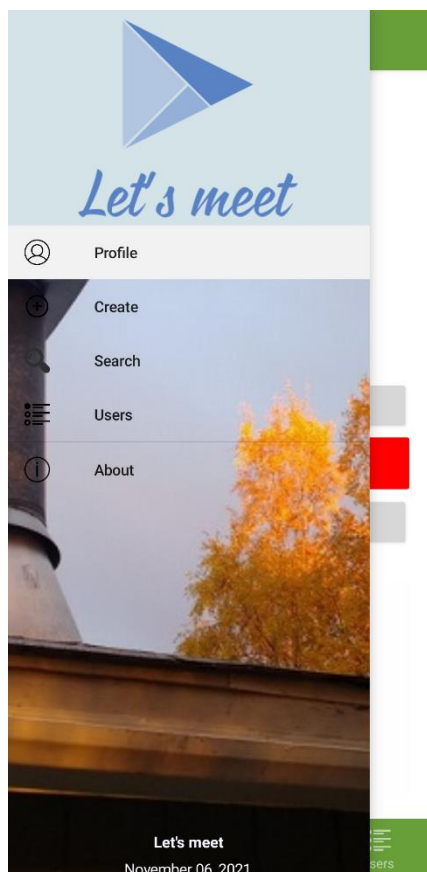


Username:

Password:

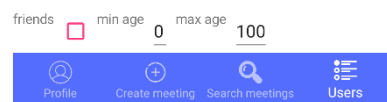
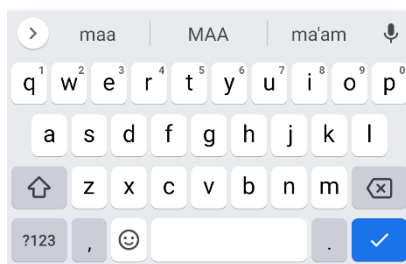
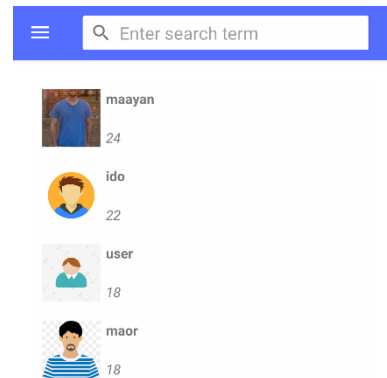
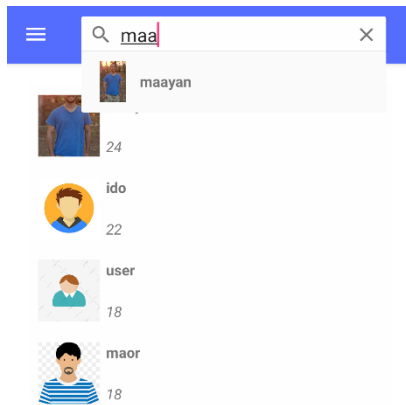
לאחר ההרשמה/ההתחברות נגיע למסך שמציג את פרופיל המשתמש שלנו, בעמוד הפרופיל ניתן להתנתק, למחוק חברים מרשימת החברים ואפילו למחוק את המשתמש שלנו.

נשים לב שלאחר ההתחברות לאפליקציה יופיע לנו תפריט עם כל עמודי המערכת, התפריט נגיש גם בתחתית המסך וגם על ידי לחיצה על כפתור התפריט/גרירתו משמאל לימין.

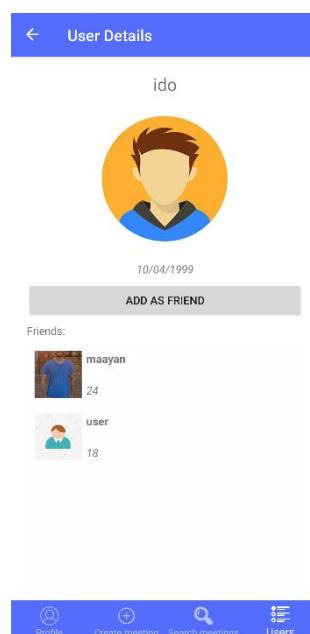
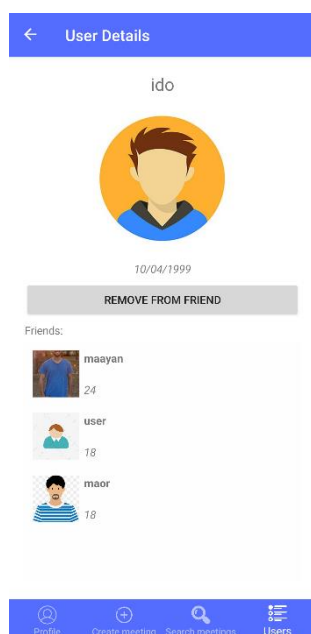


חיפוש משתמשים

בעמוד המשתמשים ניתן לראות משתמשים אחרים במערכת, אפשר לבצע חיפוש על המשתמשים גם על ידי פילטור לפי פרמטרים בתחתית המסך וגם לפי חיפוש טקסטואלי בחלק העליון של המסך.



לחיצה על אחד המשתמשים תעביר לעמוד המשתמש שם ניתן לראות עליו פרטים וגם להוסיף אותו כחבר (או לבטל את החברות איתו).

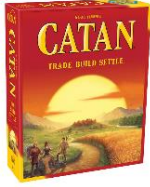



יצירת מפגש

עמוד יצירת המפגש הוא אחד העמודים החשובים באפליקציה, הוא מאפשר ליצור מפגשים חדשים וחושף אותם לכולם כך שמשתמשים אחרים יוכלו להצטרף אליהם.

שיוצרים מפגש תחילה יש לבחור את סוג המפגש לפי הקטגוריות השונות, שלב זה מתבצע על ידי לחיצה על קטגוריית מפגש ואז בחירה מבין סוגי המפגשים ששייכים לה.

לאחר בחירת סוג המפגש יש למלא פרטים חשובים נוספים: שם מפגש, תאריך תחילה וסיום, כמות מינימלית ומקסימלית של משתתפים ותמונת המפגש (כברירת מחדל לפי סוג המפגש).

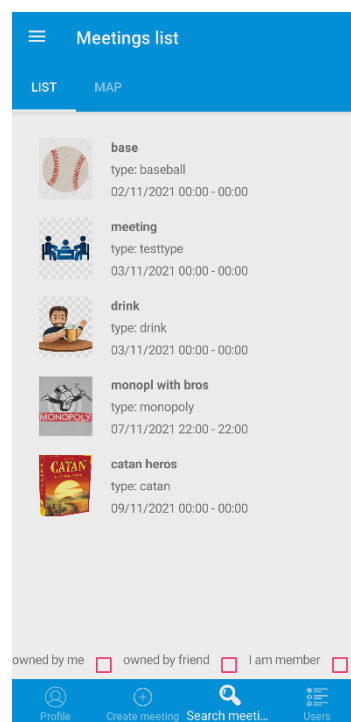
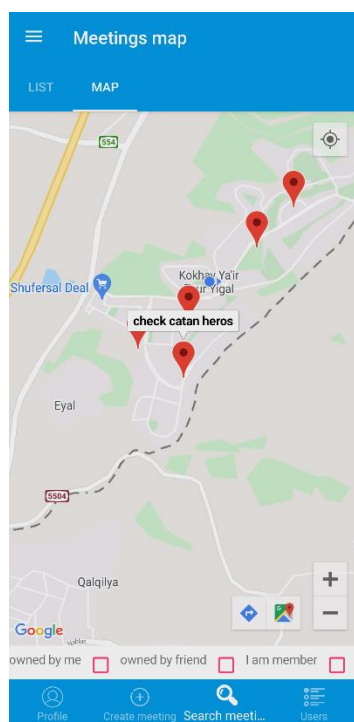
Create Meeting	Create Meeting	Create Meeting
<p>meeting name:</p> <p>catan heros</p> <p>start time</p> <p>09/11/2021 - 12:00:00</p> <p>end time</p> <p>10/11/2021 - 12:00:00</p> <p>min age</p> <p>16</p> <p>max age</p> <p>60</p> <p>min members</p> <p>3</p> <p>max members</p> <p>4</p> <p>icon URL</p> <p>https://m.media-amazon.com/images/I/81+</p>	 <p>choose meeting type: catan</p> <p>Sport</p> <p>Board and cards games</p> <p>poker</p> <p>catan</p> <p>meeting name:</p> <p>name</p> <p>start time</p>	 <p>choose meeting type: soccer</p> <p>Sport</p> <p>Board and cards games</p> <p>Social</p> <p>testmeeting</p> <p>meeting name:</p> <p>name</p> <p>start time</p>
<p>Profile</p> <p>Create meeting</p> <p>Search meetings</p> <p>Users</p>	<p>Profile</p> <p>Create meeting</p> <p>Search meetings</p> <p>Users</p>	<p>Profile</p> <p>Create meeting</p> <p>Search meetings</p> <p>Users</p>

השלב הבא הוא בחירת מיקום המפגש, כברירת מחדל מיקום המפגש הוא המיקום הנוכחי של משתמש האפליקציה והוא מצוין לפי קווי הרוחב והאורך של הקואורדינטות הגאוגרפיות. אם רוצים לשנות את המיקום יש ללחוץ על הכפתור set location שיפתח עמוד עם מפה שמאפשר לבחור על גביה מיקום חדש לפי המיקום הנוכחי של הפין (שזז בעת לחיצה על המפה).

A screenshot of a mobile application's 'Create Meeting' screen. The screen has a red header with the text 'Create Meeting'. Below the header is a form with several input fields: '16' for 'max age', '60' for 'min members', '3' for 'max members', '4' for 'icon URL', 'https://m.media-amazon.com/images/I/81+' for 'Location Latitude', and '32.2209216666667' for 'Location Longitude'. Below the form are two buttons: 'SET LOCATION' and 'CREATE'. The bottom navigation bar is red and contains the same four icons as the previous screen: 'Profile', 'Create meeting', 'Search meetings', and 'Users'.

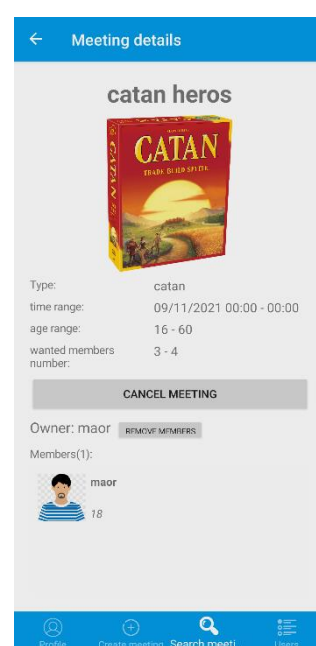
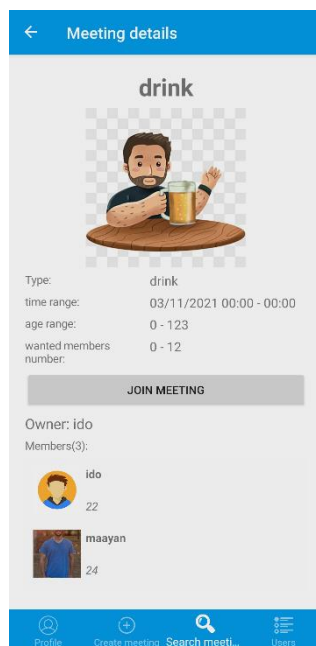
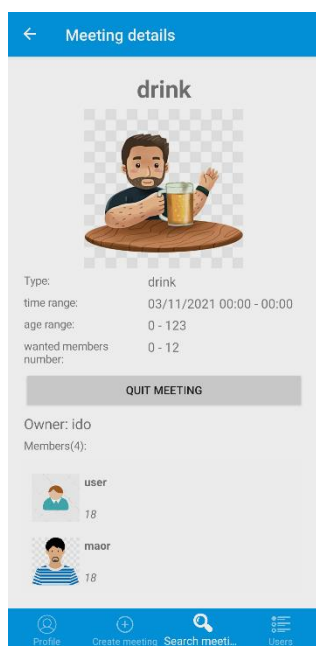
חיפוש מפגש

בעמוד חיפוש המפגשים ניתן לצפות במפגשים קיימים בשני דרכים, על גבי רשימה או על גבי מפה שבה אפשרות לראות מפגשים שקרובים אליי או מפגשים ברחבי העולם. בשני המקרים ניתן לפלטר על מפגשים על ידי האפשרויות בתחתית המסך (לדוגמא לראות רק מפגשים שנוצרו על ידי חברים).



לחיצה על מפגש תוביל לעמוד עם פרטי המפגש וכפתור שמאפשר להצטרף למפגש (בהנחה שמתקיימים תנאי הכניסה למפגש כמו הגבלות גיל וכמות משתתפים) או לצאת ממפגש שאתה שייך אליו, הרעיון הוא כמובן שכל מי שחבר במפגש יגיע אליו בפועל בחיים האמיתיים.

בצפייה בעמוד של מפגש שאני יצרתי ניתן גם להסיר אנשים מרשימת המשתתפים ואפילו לבטל את המפגש במידת הצורך.



מנהל המערכת

למנהל המערכת יש הרשאות נוספות ועוד עמוד ניהול.

כשמנהל המערכת צופה בעמוד של משתמש יש לו הרשאות למחוק את אותו משתמש (למשל אם השתמש לרעה במערכת), המחיקה תתבצע על ידי לחיצה על הכפתור האדום.

בנוסף למנהל המערכת יש יכולות להוסיף סוגים וקטגוריות חדשות של מפגשים בעמוד admin. הוספת קטגוריה מצריכה רק לבחור כותרת לקטגוריה והוספת תמונה כאשר יצירת סוג מפגש דורשת גם לבחור קטגוריה שאותו סוג נכנס תחתיה, כמוכן שלאחר יצירת הסוג יהיה ניתן ליצור מפגשים מאותו הסוג.

