AerobicWithMe

Contents

[Contents 1](#_Toc188264944)

[כללי 2](#_Toc188264945)

[**ייעוד המערכת** 3](#_Toc188264946)

# כללי

אפליקציה זו מתאימה למתאמנים מתחילים ומתקדמים שרוצים להישתפר בזמני הריצה שלהם ולשתף ו\או לבנות מסלולי ריצה עם מתאמנים אחרים .

מדובר באפליקציה מבוססת מיקום שבה המשתמש יכול לבנות מסלול מוגדר מראש בעזרת גישה לGOOGLE MAPS ,לעלות את המסלול לשרת (בפרוייקט זה נעזרתי בREALM שמבוסס על MONGO DB ) .בנוסף לכל משתמש שבנה מסלול ישנה את האפשרות לערוך את המסלול בין אם זה שינוי הכתובות של נקודות הציון על המפה או שינוי המסלול לחלוטין או לחילופין להסיר את המסלול מהשרת לגמרי (כולל מחיקת הרשומות שנשמרו עבור אותו המסלול). לכל משתמש יש גישה לעריכה רק של המסלול שאותו המשתמש בנה וצפייה במסלולים של משתמשים אחרים . יש לציין שכל משתמש יכול לעבור על מסלולים של שאר המשתנים ויכול לעבור על המסלולים האלה ולעלות את זמני הריצה לאפליקציה ועל ידי כך להשוות זמנים עם שאר משתמשי האפליקציה .בנוסף את השוואת הזמנים ומשתמשי האפליקציה שרצו את המסלול ניתן לראות בדף המסלול .

# ייעוד המערכת

האפלקציה נועדה למשתמשי אנדוראיד שרוצים לבנות מסלולי ריצה ולבצע השוואה מקוונת עם שאר המשתמשים. האפליקציה תאפשר ריכוזיות המידע, גישה נוחה ומהירה אל רשומות המסלולים ששאר המשתמשים יצרו ואל רשומות של זמני הריצה, גיבוי וייצוא של הנתונים בעזרת MONGODB, בדגש על פשטות התפעול, יציבות המערכת וזמינותה הגבוהה.

# דרישות המערכת

* המערכת תספק למשתמשי האפליקציה
* הוספה/ הסרה/ עידכון של רשומה שמתארת מסלול ריצה.
* הוספה/ הסרה/ עידכון של רשומה שמתארת את זמני הריצה של משתמשי האפליקציה.
* צפייה ברשומות של משתמשי האפלקציה לפי מסלול הריצה שהם ביצעו .
* שמירת הרשומות באופן מקוון בעזרת MONGO DB לשרת ייעודי .
* גישה מבוקרת אל האפליקציה תתבצע באמצעות שם משתמש וסיסמא, ותאפשר לכל משתמש באפליקציה לערוך\לשנות\ אך ורק את הרשמות שלו וצפייה ברשומות של שאר המשתמשים ללא אפשרות לערוך\לשנות אותם.
* צפייה בזמן אמת במיקום של המשתמש בעזרת GOOGLE MAPS ,כך שהמשתמש ידע היכן והאם הוא נמצא במסלול הנכון .והצגת המרחק של המסלול הנבחר .
* המערכת תספק ממשק נוח ופשוט למשתמשי האפליקציה .
* גיבויים :
* גיבוי רשומות מסלולי הריצה ו\או זמני הריצה של המשתמשים יתרחשו באופן מיידי ויעלו לשרת בזמן של פחות מ0.5msec .
* זמני תגובה-פעולות של מעבר מעמוד של המפה של GOOGLE MAPS לרשומות יארכו לכל היותר שנייה אחת .
* מיון של רשומות של זמני הריצה לפי זמן ריצה,תאריך העלאה של הרשומה לשרת,לפי שם המשתמש (לפי סדר אלפאבתי )
* **אפשריות השרת**
* בעזרת השימוש בREALM שמבוסס על MONGODB ,ישנה אפשרות לשרת של האפליקציה צפייה\עריכה\מחיקה\הוספה של כל הרשמות של כל המשתמשים, בין אם זה רשומות של המסלולים ו\או זמני הריצה שלהם .
* אפשרות למחיקת\הסרת\עריכה\חסימה של משתמשי מהאפליקציה.
* קביעה של אופן ההיתחברות של המשתמש לאפליקציה ,אופני החיבור יכולים להיות :
* כתובת מייל וסיסמא.
* משתמש פייסבוק וסיסמא.
* אימות של GOOGLE .
* אימות של APPLE ID .
* מפתחות API ייעודיים .
* אימות של JWK URI **.**
* האפליקציה תהיה גמישה לשינויים עתידיים והרחבת פונקציונאליות.
* האפליקציה לא תספק את היכולות הבאות בשלב הנוכחי:
* מיון זמני ריצה של המשתמשים לפי מסלול.
* מיון סיווג המסלולים לפי פרמטרים של אורך המסלול,סוג הפעילות האירובית (למשל הליכה).
* עריכת פרטי המשתמש עצמו.
* שמירת מידע היסטורי על מסלולים שהוסרו מהאפליקציה והאפשרות לשחזר אותם .

# מרכיבי המערכת

**בסיס הנתונים**

בסיס הנתונים ימוקם בשרת ייעודי של MONGODB ATLAS .

**סנכרון נתונים**

האפליקציה נעזרת ב REALM שמבוססת על MONGO DB ATLAS ,מה שמאפשר את היתרונות הבאים:

* הנתונים מתעדכנים בזמן אמת גם כאשר אין חיבור לאינטרנט (offline-first), כך שהאפליקציה ממשיכה לעבוד והסנכרון מתבצע כשחוזר החיבור.
* איפשור סנכרון אוטומטי בין מסד הנתונים הלוקאלי לבין השרת.

# משתמשי האפליקציה

משתמשי האפליקציה יכולים להירשם\להיתחבר בעזרת אימייל סיסמא שנשמרת בשרת של MONGODB ATLAS .

למשתמשי האפליקיצה האפשרות לבצע את הפעולות הבאות :

* יצירת מסלולי ריצה והעלתם לשרת.
* יצירת רשומה של זמן הריצה לפי מסלול(כלומר להפעיל טיימר ולרוץ את המסלול) .
* עריכת שמות הנקודות של מסלולי הריצה.
* מיון של רשומות של זמני הריצה של המשתמשים לפי :
  + - * תאריך העלאה של הרשומה לענן.
      * זמני הריצה מהנמוך ביותר עד לגבוהה ביותר .
      * שם המשתמש לפי סדר אלפאבתי .

# דיאגרמת Use Case

ניהול משתמשים באפליקציה

A black and white icon of a person

Description automatically generatedA blue rectangular object with a black stripe

Description automatically generatedA close-up of a word

Description automatically generatedA black and white icon of a person

Description automatically generated

**עקיבה של רשומות של מסלולי הריצה והזמני ריצה של המסלולים.**

**יצירה\עריכה של מסלולי ריצה**

**עריכת פרטי משתמשים קיימים**

**מחיקת משתמשים קיימים**

Server

**הוספת משתמשים חדשים**

**יצירה של רשומת זמן ריצה**

**צפייה ברשומות ומסלולי ריצה של שאר המשתמשים באפליקציה**

App User

**היתחברות\יצירת חשבון חדש באפליקציה**

# ניהול משתמשים במערכת

# מדריך למשתמש-SmartRun

שם המגיש: תומר פולסקי

# תיאור האפליקציה

האפליקציה מיועדת (לפחות כרגע) למשתמשי ANDROID בלבד.מדובר על אפליקציה שמאפשרת בעזרת חיישן המיקום של הפלאפון לGOOGLE MAPS ליצור מסלולי ריצה\הליכה ובנוסף לבצע השוואות זמני ריצה\הליכה של שאר המשתמשים עבור אותו המסלול . האפלקציה מאפשרת למצוא מיקום של המשתמש בזמן שהוא רץ את המסלול וכמו כן להפעיל טיימר ולעלות רשומה של זמני הריצה של המסלול והוספת תגובות לרשומה. לאחר סיום עריכת הרשומה של זמן הריצה ניתן לעלות את הרשומה לשרת של MONGO DB ATLAS . ולאחר מכן עבור כל מסלול ניתן לראות את הרשומות של שאר המשתמשים ולהשוות זמני ריצה.

מסמכי האפיון הניתוח והעיצוב יתארו את המערכת עבור משתמש חדש שרק נרשם לאפליקציה .

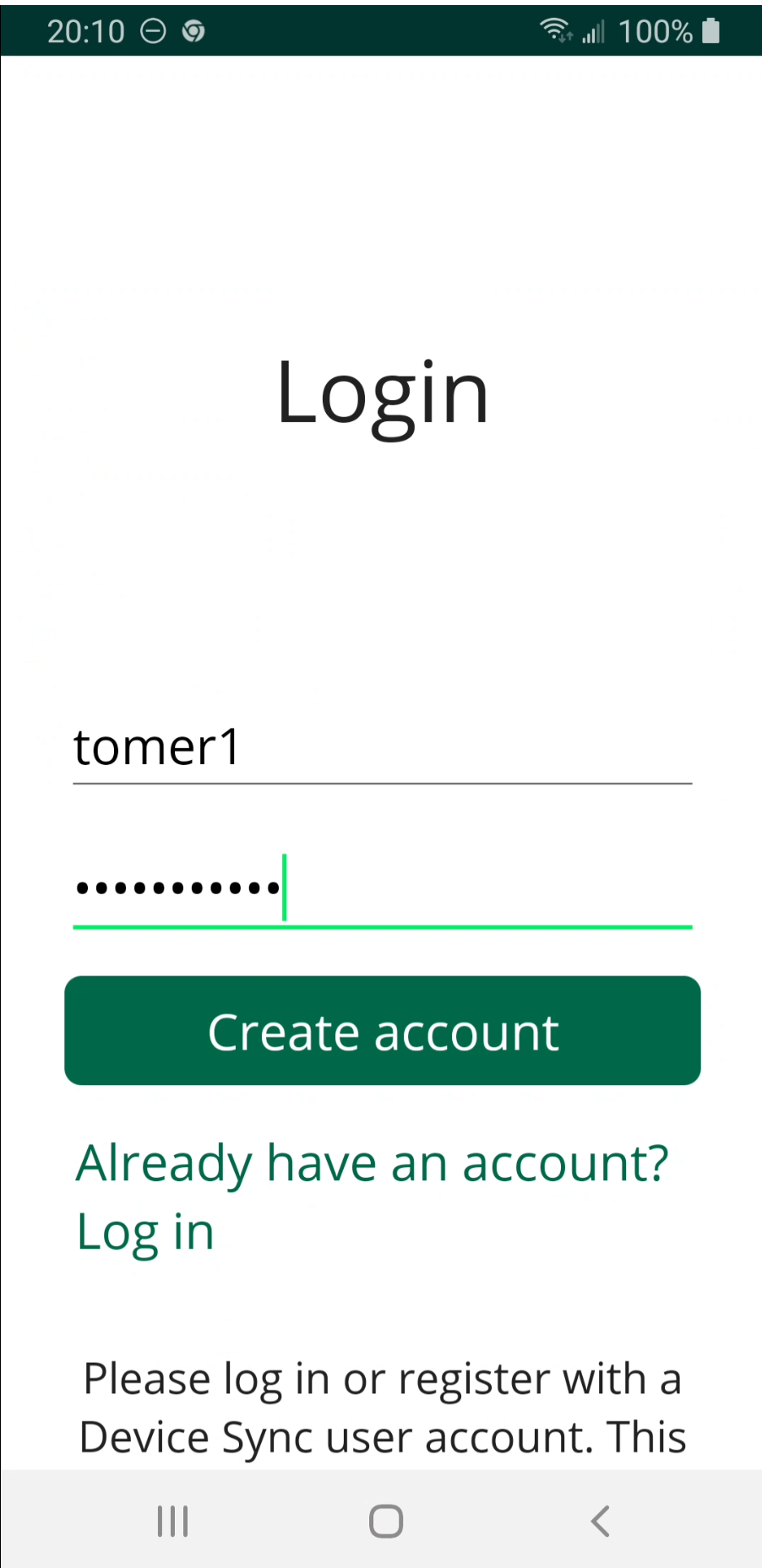
1. האפליקציה נבנתה עבור משתמשי אנרואיד עבור חובבי ריצה\הליכה\פעילות אירובית .
2. משתמשי האפלקיצה-בעלי פלאפון מסוג אנדרואיד שמעוניינים לבצע פעילות אירובית במסלולים מוגדרים מראש והשוואת זמני הפעילות עם שאר משתמשי האפלקציה .
3. יכולות האפלקציה

* רישום משתמשים חדשים באפליקציה .
* היתחברות של משתמשים קיימים לאפליקציה.
* יצירת מסלולים חדשים בעזרת GOOGLE MAPS .
* עריכה של מסלולים קיימים ושינוי שמות הנקודות של המסלול .
* טיימר מובנה באפליקציה שמאפשר לראות בזמן אמת את זמן הריצה של המשתמש שמחובר כעת לאפליקציה.
* צפייה בזמני הריצה לפי מסלולים של שאר משתמשי האפליקציה.
* מחיקת מסלולים קיימים ומחיקת הרשומות של אותו המסלול .
* מיון של רשומת זמני הרציה של המשתמשים לפי :
* תאריך העלאה של הרשומה לענן.
* זמני הריצה מהנמוך עד לגבוהה.
* שם המשתמש לפי סדר אלפאביתי.

**דגשים והנחות לגבי האפליקציה**

* המשתמש חייב להירשם לאפליקציה .
* שם המשתמש אינו ניתן לעריכה.
* הסיסמא של המשתמש (נכון לגירסא זו ) אינה ניתנת לעריכה.
* האפליקציה נבנתה עבור משתמשי אנדרואיד בלבד.
* לשם העלת הרשומות של המסלולים לשרת יש לוודא שישנו חיבור WIFI תקין .
* בשביל לבנות מסלול חדש חייב שהמיקום של הפלאפון יהיה מאופשר על ידי המשתמש .
* במידה והמשתמש מנסה לעלות רשומה לשרת ולא קיים חיבור WIFI ,האפליקציה תעלה באופן אוטמטי את הרשומה ברגע שחיבור הWIFI יחודש.
* במידה והמשתמש מנסה לעלות מסלול לשרת ולא קיים חיבור WIFI ,האפליקציה תעלה באופן אוטמטי את המסלול ברגע שחיבור הWIFI יחודש.
* במידה והמשתמש ירצה ליצור מסלול חדש ,חייב שיהיה לפחות שני נקודות במסלול .
* השם של המסלול והשם משתמש של רשומת זמן הפעילות חייבים להיות לא ריקים .
* רק משתמש שיוצר מסלול יכול לערוך אותו אחרת תקפוץ הודעת אזהרה .
* במידה והמשתמש ירצה למחוק מסלול שהוא בנה הוא ימחוק גם את כל הרשומות של שאר המשתמשים שרצו את המסלול הזה (מופיע הודעת אזהרה לפני מחיקת מסלול ).
* זמני הריצה מופיעים בפורפט HH:MM:SS,ואילו תאריך העלאה יהיה בפורמט dd/MM/yyyy .

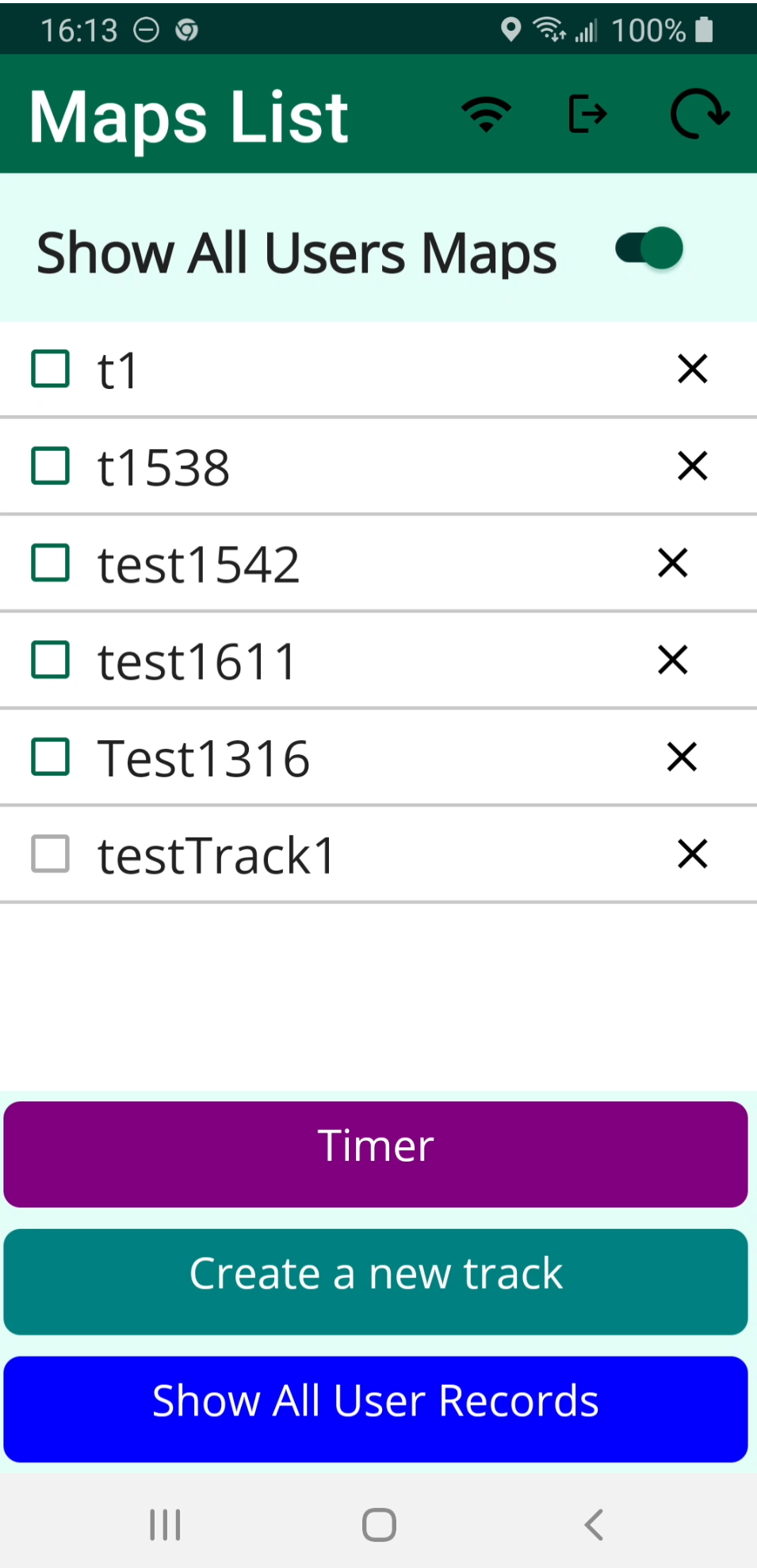
**היתחברות\הרשמה לאפליקציה**

****

* ניתן להירשם לאפליקציה בעזרת הזנת האימייל או שם משתמש והסיסמא ואז לחיצה על כפתור ה- Create account .
* מסך זה מאפשר להיתחבר לאפליקציה באמצעות שם משתמש וסיסמא על ידי לחיצה על כפתור ה Already have an Account .

**מסך ראשי של האפליקציה**

לאחר שהמשתמש היתחבר לאפליקציה יופיע לו המסך שבו מופיעים כל המסלולים ששאר המשתמשים העלו לענן.

****

**מסך זה כולל 3 כפתורים רגילים**

* Timer-כפתור המעביר את המשתמש למסך של טיימר .
* Create a new track- מסך שמעביר את המשתמש לעמוד של GoogleMaps ומאפשר למשתמש ליצור מסלול חדש .
* Show All User Records -מסך שמראה את כל רשומות זמני הריצה של כל המשתמשים .

A black arrow pointing to a green background

Description automatically generated**3 כפתורי תמונות**

-כפתור זה בעצם מרענן את הרשימה של המסלולים (במידה ומשתמש אחר הוסיף מסלול חדש ) אז ניתן לראות את המסלול.

A black arrow on a green background

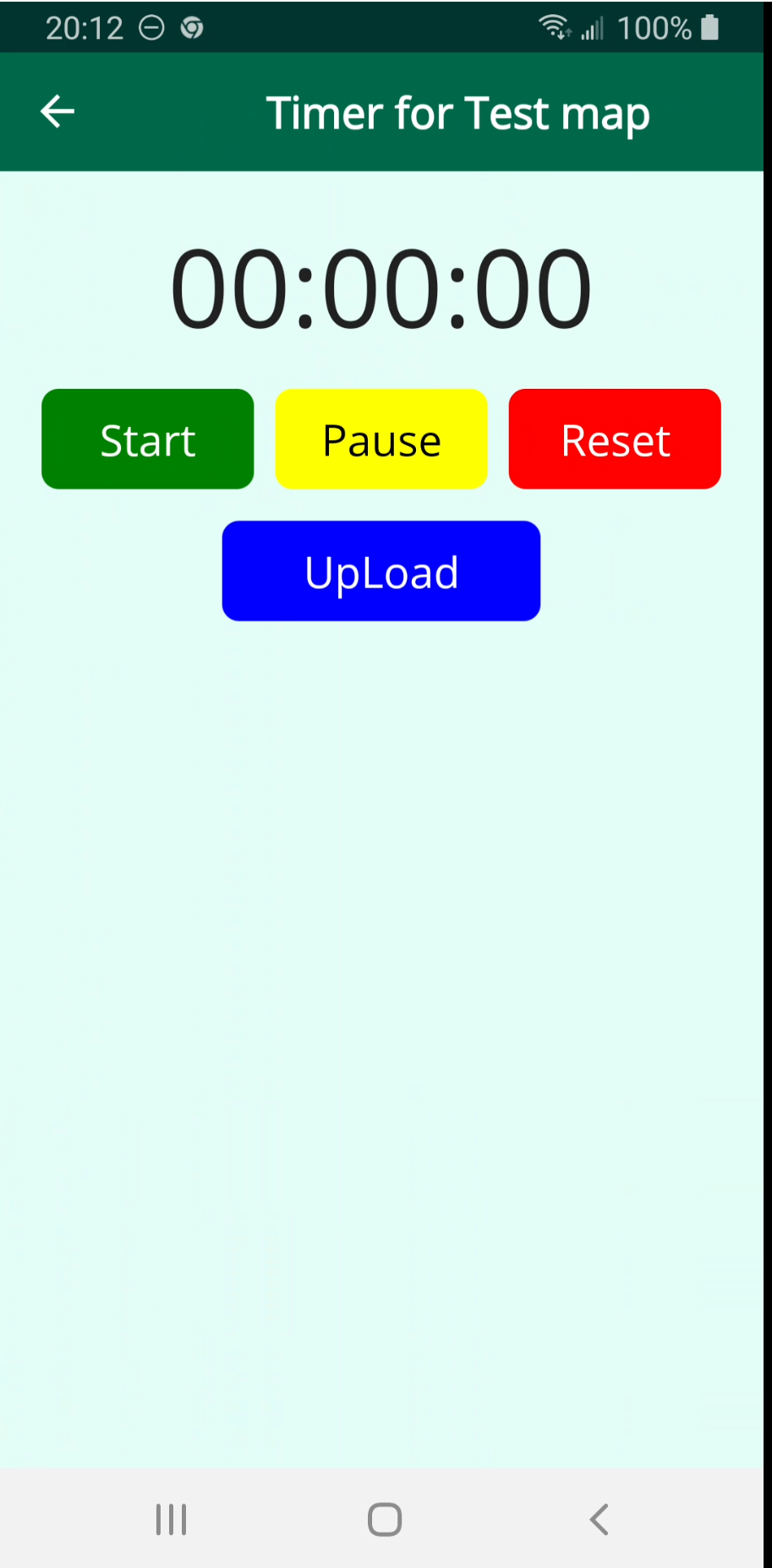
Description automatically generated-כפתור זה מאפשר להיתנתק מהאפליקציה ולחזור למסך של ההיתחברות .

A black wifi symbol on a green background

Description automatically generated-כפתור שמאפשר לכבות את החיבור של הWIFI באפליקציה .

-מפסק שמאפשר לראות את כל המסלולים של המשתמש בלבד או של כל המשתמשים שהעלו מסלולים לאפליקציה .

**מסך הטיימר**

****

* מסך זה מאפשר להפעיל טיימר ללא תלות בבחירת המסלול .
* מסך זה כולל כפתורים של התחלת הספירה של הטיימר,השהייה של הטיימר ,ואיפוס הטיימר .
* UPLOAD-כפתור שמאפשר לעלות לענן את הזמן של המשתמש לענן .

**מסך העלאת הרשומה לענן**

**A screenshot of a message

Description automatically generated**

* מסך זה מאפשר לעלות את את הרשומה של המשתמש לענן .
* מסך זה כולל את השם משתמש שהמשתמש יכול לקבוע,הערות שהמשתמש יכול להוסיף לרשומה,שם המפה,זמן הפעילות האירובית, השעה והתאריך של סיום הפעילות .
* לחיצה על כפתור הOK יוסיף את הרשומה לענן.

**מסך התצוגה לאחר יצירת מסלול חדש**

**A map of the middle east

Description automatically generated**

לחיצה על גבי המפה תוסיף נקודה על המפה שנראית כך: . על גבי המסך של יצירת מסלול חדש בעזרת GOOGLE MAPS .

* A pencil and a rectangle with a map in the background

  Description automatically generated-כפתור שמעביר את המשתמש למסך שבו ניתן לערוך את מספר הנקודה ולשנות את הכתובת,של הנקודה . ברירת המחדל שהכתובות של הנקודות הם מספרים שמתחילים מ1 .
* A red recycle bin on a blue background

  Description automatically generated- כפתור שמאפשר למחוק את המפה שהמשתמש יצר לפני שהוא העלה אותה לענן.
* A red line drawing of a map

  Description automatically generated- כפתור שמעביר את המשתמש למסך שמציג את המרחק מנקודה לנקודה, והמרחק הכללי של המסלול שהמשתמש בנה עד עכשיו .
* A black arrow pointing up

  Description automatically generated-כפתור שמעביר את המשתמש למסך של העלאת המסלול .
* A black magnifying glass with a white cross

  Description automatically generated-כפתור שמבצע זום למיקום של המשתמש ברגע זה.
* A black and white logo

  Description automatically generated-כפתור שמאפשר למחוק את הנקודה האחרונה שהוספה למסלול .

**מסך הוספת שם למסלול והוספה לענן**

**A screenshot of a cell phone

Description automatically generated**

במסך זה ניתן לבחור את שם המסלול ולאחר לחיצה על כפתור הOK לעלות את המסלול לענן של

MONGOD DB ATLAS .

**מסך חישוב המרחק של המסלול**

**A screenshot of a phone

Description automatically generated**

מסך זה מאפשר לראות את המרחק מנקודה לנקודה של המסלול ואת אורך המסלול הכללי. כמובן שהנקודה הראשונה של המסלול מסומנת כ1P השנייה 2P וכך הלאה.

**עריכת כתובת של נקודה**

**A screenshot of a pin code

Description automatically generated**

מסך זה בעצם מאפשר לערוך את הכתובת של נקודה מסויימת,על ידי לחיצת מספר הנקודה ואז הכנסת שם כתובת חדש.

**בחירת מסלול מהרשימה**

במידה והמשתמש החליט לבחור מסלול שהוא בנה יופיע המסך הבא :

A map of a city

Description automatically generated

כפתורי המסך יהיו זהים לכפתורים של המסך לאחר לחיצה על הכפתור שיוצר מסלול חדש,פרט ל2 הכפתורים הבאים :

* -כפתור הטיימר שמעביר את המשתמש למסך הטיימר (שעליו הסברנו כבר בעמוד השלם את מספר העמוד)

A black and white sign with black text

Description automatically generated-כפתור שמאפשר למשתמש לצפות ברשמות זמני הפעילות האירובית בהתאם למסלול הנבחר .

**מסך הרשומות של זמני הפעילות האירובית לפי בחירת המסלול**

**A screenshot of a test

Description automatically generated**

מסך זה בעצם מאפשר לראות זמני הרשמות של הפעילות האירובית של שאר המשתמשים בהתאם למסלול שאותו בוחרים.

-כפתור זה מאפשר להראות את כל הרשמות של המתמשים שביצעו את הפעילות האירובית של המסלול,או אם הכפתור כבוי להראות רק את הרשומות של המשתמש שמחובר כעת לאפליקציה .ניתן לראות את הרשמות של המשתמש שמחובר כעת לאפליקציה בעזרת הריבוע הירוק בצד שמאל ,וניראת כך: ,רשומה של משתמש אחר צבועה בצבע אפור ונראית כך : .



לחיצה על האפשרות של SORT BY . מאפשרת למיין את הרשומות . לאחר הלחיצה יופיע לי החלון הבא :

A screenshot of a phone

Description automatically generated

ניתון למיין את הרשמות לפי שם הפרופיל,משך הפעילות האירובית ,תאריך העלאת הרשומה לענן , או ברירת מחדל.

**צד השרת**

האפליקציה מבוססת על Mongo DB atlas ,שהוא ענן שמיועד להעלעת אובייקטים לענן של MONGO שמיועד בעיקר לאפליקציות ANDORID . בשביל לראות את המשתמשים והאוביקטים שעלו לענן יש צורך להירשם ללינק הבא :   
<https://services.cloud.mongodb.com/groups/66ae39a02e42444be6f2b535/apps>

**משתמשי האפליקציה**

A screenshot of a computer

Description automatically generatedאת משתמשי האפליקציה ניתן לראות על ידי לחיצה על app users

**הרשמה לאפליקציה**

את אופן הרישום לאפליקציה ניתן לבחור על ידי לחיצה על ידי לחיצה על Authertication Providers ,את האפשריות שבעזרת ניתן להירשם לאפליקציה ניתן לראות בתמונה למטה .

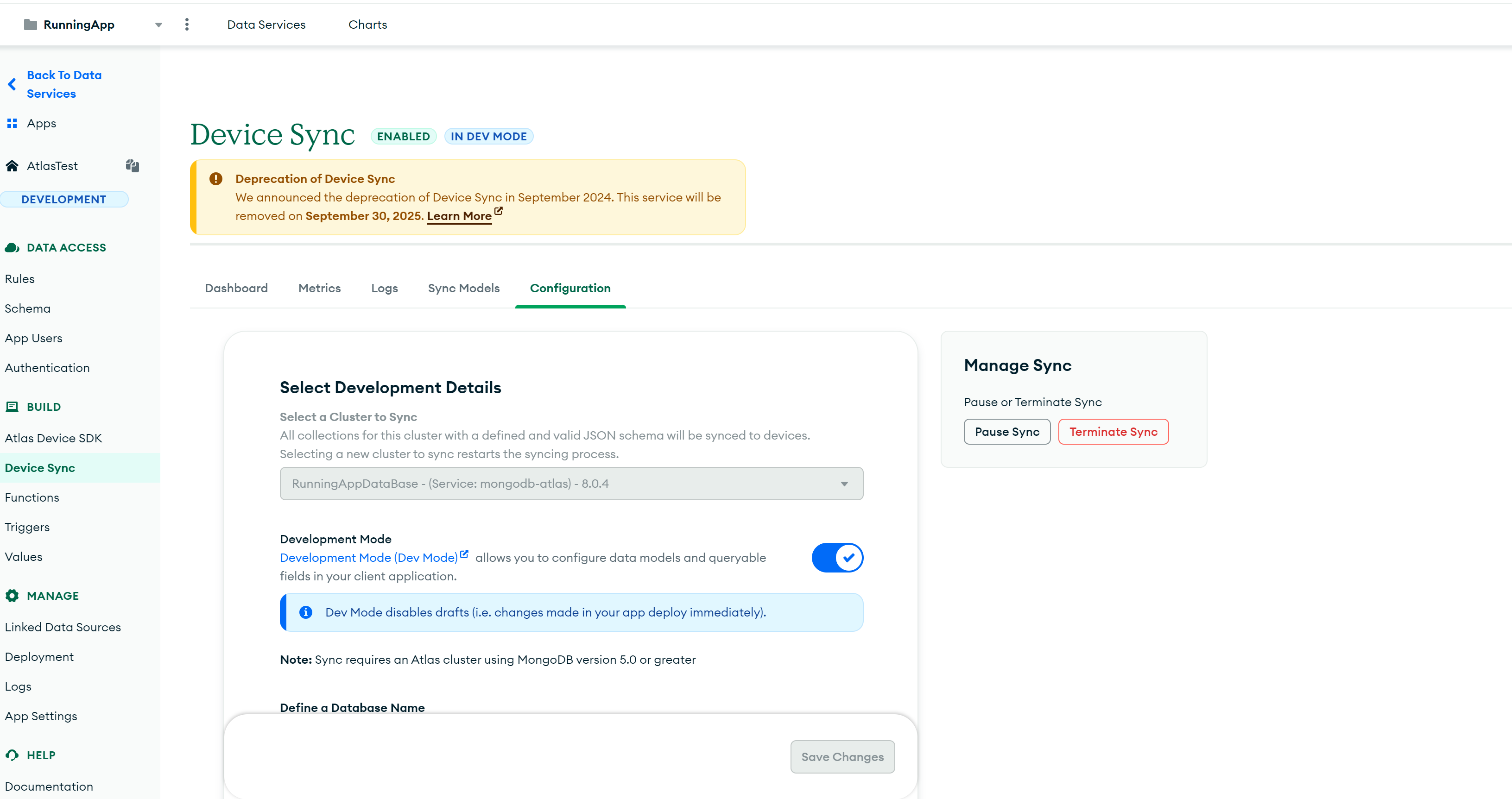
A blue and white line

Description automatically generated with medium confidence

**סינכרון האפליקציה**

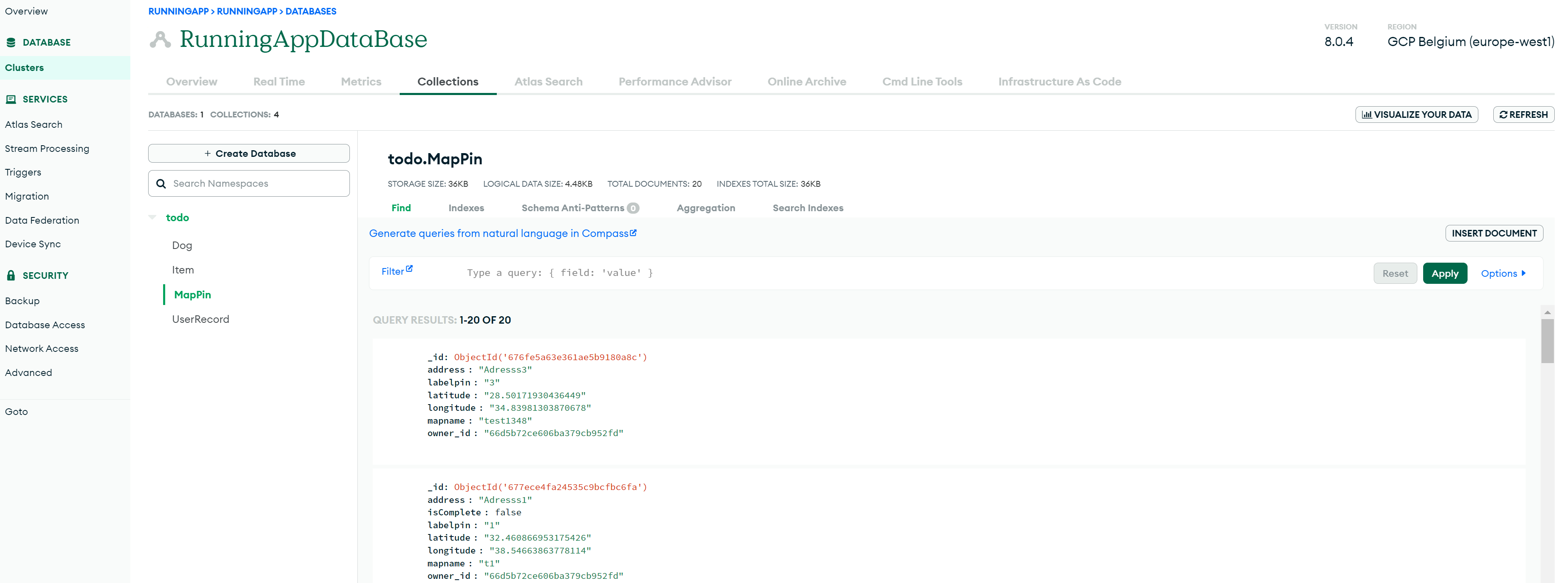
לפני הפעלת האפליקציה יש לוודא שישנו סינכרון של המכשרים לענן של MONGO DB . על ידי לחיצה על לשונית

Device Sync .



**אובייקטים שעלו לענן**

על ידי לחיצה על DataServices ואז על DataBase ,ניתן לראות את האובייקטים שעלו לענן של MONGO DB .



מסמך תכנון ועיצוב

שם המגיש: תומר פולסקי

**תיאור כללי של המערכת**

המערכת בנויה על סמך מודל 3 השכבות (3-TIER )

* שכבת התצוגה (PL-Presenation Layer)
* שכבת האפליקציה (BL-Buisness Logic Layer)
* שכבת גישה לנתונים ( DAL-Data Access Layer)

כל שכבה מתקשרת עם שאר השכבות .ההפרדה לשכבות מאפשרת ליצור פרויקט מסודר שיקל על סידור האפליקציה .

**הנחות עבודה בכתיבת הפרוייקט**

1. האפליקציה פותחה ועובדת אך ורק עבור משתמשי אנדוראיד .
2. משתמשי האפליקציה חייבם להביא הרשאה למיקומם אחרת האפליקציה לא תעבוד בצורה תקינה .
3. בשביל לעלות רשומות או מסלולים שהמתמש בנה יש לוודא שישנו חיבור תקין לWIFI אחרת הרשומה תעלה כשהמשתמש יתחבר לWIFI בהמשך .
4. הזמנים והתאריכים שנשמרים בענן נשמרים בפורמט הבא: DD/MM/Year Hour:Minutes:Seconds
5. הזמנים והתאריכים נשמרים לפי המדינה שבה משתמש האפליקציה נמצא .
6. אסור ליצור מסלול שלא כולל בו לפחות 2 נקודות.
7. כאשר מעלים רשומה של זמן מסויים עבור מסלול מסויים ,שם המשתמש לא יכול להיות ריק.
8. האפליקציה מבוססת על FREE TRAIL של MONGO DB ATLAS ,לכן הגודל המקסמלי של כל האובייקטים שניתן לעלות הם עד 36KB .

**מוסכמות רישום**

1. כל שם מחלקה שכוללת יותר ממילה אחת נכתבת עם אות גדולה בתחילה כל מילה נוספת.
2. שיטות עזר(אלו הן שיטות שמשמשות לחישובים שונים כמו למשל השיטה getDistance2Points שמחשבת את המרחק בין 2 נקודות על גבי המפה ) במחלקה מתחילה מאות קטנה,במידה ושם השיטה כולל יותר ממילה אחת אז רק המילה הראשונה מתחילה מאות קטנה ,שאר המילים מתחילים באות גדולה.
3. שיטות שמשמות בשביל תצוגת הXAML (כמו למשל SaveUserRecord שמשמשת להעלאת רשומה חדשה לMONGO DB ) ושכוללות יותר ממילה אחת יכתבו כל מילה באות גדולה .

**שכבת בסיס הנתונים**

שכבה זו אחראית על התקשורת עם המסד נתונים . והיתחברות של המשתמש לMONGO DB . המערכת עושה שימוש בעזרת מסד נתונים של MONGO DB ATALS REALM .

שכבה זו מכילה מחלק בשם RealmService שנמצאת בתקיית Services .מחלקה זאת אחראית ליצור את ההיתקשרות עם המסד נתונים בעזרת שימוש בשיטה של GetRealm שנעזרת באובייקטים מהספרייה של Realms .נציין שבשביל לסנכרן את האפליקציה עם הענן של REALMS שפתחנו בMONGO DB ATLAS ,יש להעיזר בקובץ atalasConfig.json ותחת השדה של appId להדביק את הId שנוצר באתר של MONGO DB ATLAS .

המחלקה מכילה את השיטות הבאות :

GetRealm()-יוצר מופע של Realm בהתאם **לסוג האובייקט הפעיל** MapPin או UserRecord ומגדיר את הסנכרון המתאים.

RegisterAsync(string email, string password)- רושם משתמש חדש ב-**MongoDB Atlas** עם אימייל וסיסמה.

LoginAsync(string email, string password)- מתחבר לחשבון המשתמש באמצעות אימייל וסיסמה, פותח את Realm, וממתין לסנכרון הנתונים.

LogoutAsync()-מתנתק מהמשתמש הנוכחי שמחובר לאפליקציה, סוגר את התהליכון של mainThreadRealm ומוחק את התהליכון הנוכחי .

SetSubscription(Realm realm, SubscriptionType subType)-

* מגדיר את סוג המנוי Mine or All.
* מוחק את המנוי הקיים ומוסיף חדש בהתאם.
* ממתין לסנכרון הנתונים אם החיבור אינו מנותק.

GetCurrentSubscriptionType(Realm realm)- בודק את **סוג המנוי הנוכחי** (mine או all) ומחזיר את ערכו כ-SubscriptionType.

GetQueryForSubscriptionMapPinType-שיטה שמחזירה יוצרת ומחזירה **שאילתת חיפוש (IQueryable<**MapPin**>) ושם מנוי (string)** עבור אובייקט מטיפוס MapPinבמסד הנתונים **Realm**, בהתאם לסוג המנוי (SubscriptionType).

GetQueryForSubscriptionUserRecordType -שיטה שמחזירה יוצרת ומחזירה **שאילתת חיפוש (IQueryable<UserRecord>) ושם מנוי (string)** עבור אובייקט מטיפוס UserRecord במסד הנתונים **Realm**, בהתאם לסוג המנוי (SubscriptionType)..

* עקרונות מנחים לגבי שכבת בסיס הנתונים :   
  בסיס הנתונים מבוסס על **MongoDB Atlas** הוא שירות **Database-as-a-Service (DBaaS)** שמספק **MongoDB מנוהל בענן**. מדובר בפתרון שמאפשר לך להפעיל, לנהל ולתחזק מסדי נתונים של MongoDB **בענן**, ללא צורך בניהול שרתים פיזיים.

**מאפיינים עיקריים של MongoDB Atlas:**

**מסד נתונים מבוסס NoSQL**

* MongoDB הוא מסד נתונים מסמכי (Document-Oriented) המאחסן נתונים בפורמט JSON דינמי (BSON).
* מאפשר שמירת נתונים בצורה גמישה ומותאמת ליישומים מודרניים.

**תמיכה בפריסה בענן**

* ניתן להריץ את MongoDB Atlas על AWS, Azure או Google Cloud.
* מאפשר לך לבחור אזורי פריסה (Regions) וליצור מסדי נתונים קרובים למשתמשים שלך.

**ניהול אוטומטי של מסד הנתונים**

* גיבויים אוטומטיים, ניטור ביצועים, איזון עומסים.
* עדכונים אוטומטיים של התוכנה כדי לשמור על אבטחה וביצועים מיטביים.

**גמישות וסקלביליות (Scalability)**

* ניתן להגדיל או להקטין את המשאבים בהתאם לצרכים העסקיים שלך.
* תמיכה ב-Sharding (פיצול נתונים בין מספר שרתים) ו-Replication שכפול נתונים לגיבוי ואמינות

**תמיכה בסנכרון נתונים עם Realm**

* מאפשר סנכרון נתונים עם יישומים ניידים או אינטרנט באמצעות MongoDB Realm, פתרון לסנכרון נתונים ביישומים ללא חיבור קבוע לרשת.

**פאנל ניהול אינטואיטיבי**

* ממשק גרפי (Atlas UI) לניהול מסדי נתונים, שאילתות, הרשאות משתמשים וניטור ביצועים.

מאפייני הרשומת של המשתמשים (UseresRecord)

**האפליקציה מעלה שני אובייקטים לענן הראשון הוא אובייקט בשם UserRecord, שהיא רשומה ששומרת את פרטי הפעילות האירובית שהמשתמש ביצעה .**

**Id**-מזהה ייחודי של המשתמש במסד הנתונים(MongoDB) .בעזרת השיטה ObjectId.GenerateNewId נוצר טיפוס חדש בשם ObjectID,עבור כל אובייקט שאנו מעלים .

**OwnerId**-מזהה של המשתמש שהעלה את הרשומה, חובה למילוי .

**ProfileName**-שם הפרופיל של המשתמש,מטיפוס STRING.

**MapName** -שם המפה המשויכת למשתמש ,מטיפוס STRING .

-**TrackTime**משך הזמן של הפעילות האירובית של המשתמש באותו המסלול ,מטיפוס STRING .

-**UploadDateTime**התאריך והשעה שבהם הנתונים הועלו למסד הנתונים (MongoDB),מטיפוס STRING .

-**Comment**הערות שהמשתמש הוסיף לרשומה.מטיפוס STRING.

**IsMine**-שדה מחושב (Read-Only), מחזיר **true** אם המשתמש הנוכחי (ב-RealmService) הוא הבעלים של הרשומה.

**מאפייני הנקודות של המסלול (MapPin)**

**האובייקט השני שאנו מעלים לענן הוא אובייקט מטיפוס MapPin,אובייקט זה כולל את הנקודות של המסלול שאותו המשתמש מעלה לענן. כאשר יותר מ2 נקודות על המפה עם אותו השם בעצם יוצרות מסלול שאותו ניתן לעלות לענן .**

-**Id**מזהה ייחודי של הנקודה שהמשתמש הוסיף על המפה, מיוצר אוטומטית (ObjectId).

-**OwnerId** מזהה של האובייקט של הנקודה שהוספה עלהמפה .

- **Mapname**שם המפה של המסלול . טיפוס מסוג STRING .

- **Labelpin**תווית (Label) המתארת את הנקודה . טיפוס מסוג STRING.

- **Address**הכתובת של המיקום שבו נמצאת הנקודה . טיפוס מסוג STRING.

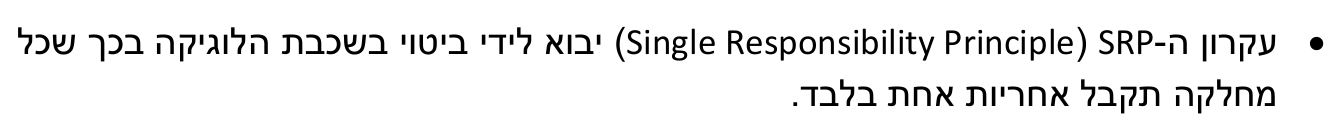
- **Latitude**קו רוחב (Latitude) של הנקודה שנמצאת על המפה.טיפוס מסוג STRING.

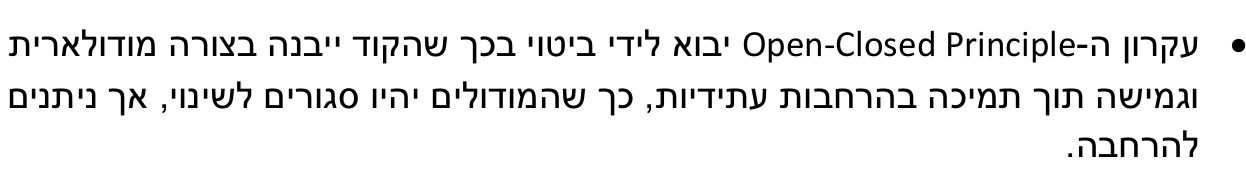
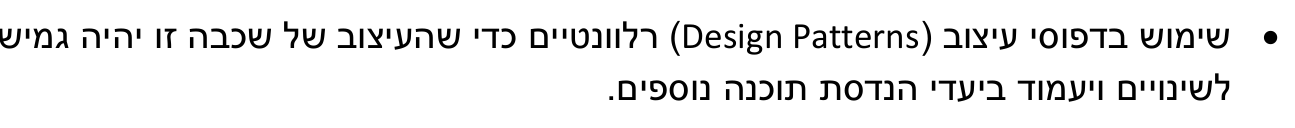
- **Longitude**קו אורך (Longitude) של הנקודה שנמצאת על המפה. טיפוס מסוג STRING.

**שכבת הלוגיקה**

שכבה זו מקשרת בין שכבת הענן\בסיס נתונים לשכבת ההצגה .

עקרונות מנחים בשכבת הלוגיקה





**מחלקות של אובייקטים**

מחלקת MapPin -מחלקה זו מייצגת ,נקודה על גבי המפה .וכוללת בתוכה את השדות הבאות :

* Id: ObjectId
* OwnerId: string
* Mapname: string
* Labelpin: string
* Address: string
* Latitude: string
* Longitude: string
* IsMine: bool

מחלקת UserRecord-מחלקה זו כוללת רשומה של משתמש שמבצע משתמש על גבי מסלול שהוא בוחר . המחלקה כוללת את השדות הבאים :

* Id: ObjectId
* OwnerId: string
* ProfileName: string
* MapName: string
* TrackTime: string
* UploadDateTime: string
* Comment: string
* IsMine: bool

**מחלקות כלליות**

MapHelper.cs-מחלקה זאת היא מחלקת עזר שכוללת את כל החישובים הנכללים בתוך בניית מסלול חדש .

 setMap()

 showTrack()

 getPoint()

 showTrackOnMap()

 setPinsList()

 drawLineBetweenAllPins)

 removePolylineBetweenPins()

 deleteLastPoint)

 getPtrNamesAndPolylines()

 drawLineBetween2Pins()

 calculateTotalDistance()

 getDistance2Points()

 addPointOnMap()

 static getPoint()

TypeFactory-מחלקה זו מייצגת מחלקת סינגלטון שבוחרת את טיפוס המחלקה שיעלה לענן . אחת השיטות שמופיעות במחלקה הם Set שקובעות האם הטיפוס של המחלקה יהיה מטיפוס UserRecord או MapPin. וכללת את השיטות הבאות :

* ObjectSingleton() (private constructor)
* Instance (public static property)
* SetMapPinType()
* SetUserRecordType()
* GetCurrentType()

**מחלקות של ViewModels**

במבנה של MVVM (Model-View-ViewModel), מחלקות שממוקמות בתיקיית ViewModels משמשות כמתווך בין הלוגיקה העסקית (Models, Services, Database) לבין הממשק הגרפי (UI.התקייה של ViewModels כוללות את המחלקות הבאות: :

1. מחלקת- EditMapPinViewModel .
2. מחלקת- EditUserRecordViewModel .
3. מחלקת- MapsViewModel.cs .
4. מחלקת- UserRecordsViewModel.cs.
5. מחלקת - LoginViewModel.cs.

**1.מחלקת EditMapPinViewModel.cs**

ה-ViewModel הזה משמש לעריכת נקודות במפה, טעינת מסלול, והעלאת הנתונים ל.**MongoDB Realmu**וכוללת את השיטות הבאות :

setMapName(string newMapName)  
ApplyQueryAttributes(IDictionary<string, object> query)  
ShowTrack()  
UploadToCloudPins()  
SavePin(Maui.GoogleMaps.Pin newPin)  
Cancel()

**2.מחלקת.cs EditUserRecordViewModel**

ה-ViewModel הזה משמש ליצירה של רשומה של זמני התוצאה של משתמש עבור מסלול שהוא ביצע במפה, טעינת רשומות של משתמשים לפ מסלול, והעלאת הנתונים ל.MongoDB Realmוכוללת את השיטות הבאות :

ApplyQueryAttributes(IDictionary<string, object> query)

SaveUserRecord(UserRecord newUserRecord)

Cancel()

**3.מחלקת MapsViewModel.cs**

מחלקה זו שומרת בתוכה את השיטות שמאפשרות להראות את כל המסלולים ששמורים בענן של האפליקציה וכוללת בתוכה את השיטות הבאות :

public MapsViewModel();

public async void deleteExistingMapPinFromCloude(Maui.GoogleMaps.Pin newPin, string mapNameToDelete);

private static string GetCurrentDateTime();

public void OnAppearing();

public async Task SaveUserRecord();

public async Task GoToUserRecordsList();

public async Task Logout();

private static List<Maui.GoogleMaps.Pin> getPinsListByName(string trackName);

public void TestCommand(MapPin pin);

public async Task ChooseMapFromList(MapPin map);

public async Task ToMapPage();

public async Task DeleteMap(MapPin pinOfChoseMap); public async Task DeleteUsersOfTrack(string trackName);

public async Task DeleteSinglePin(MapPin pin);

public void Refresh();

public async Task ToTimerPage();

public void ChangeConnectionStatus();

public async Task UrlTap(string url);

private async Task<bool> WarningDeletingMyTrack(MapPin chosenPinOfMap);

private async Task<bool> CheckMapOwnership(MapPin chosenPinOfMap);

**4.מחלקת UserRecordsViewModel.cs**

* מחלקה זו נועדה בשביל להכין את כל הרשומות של המשתמשים שרשומים באפליקציה .
* public UserRecordsViewModel()
* public async Task DeleteUserRecord()
* public async Task ShowRecord()
* private async Task<bool> CheckUserRecordOwnership()
* public string UserRecordsTiltle { get; set; }
* public void setTrackName(string newTitle)
* public string getTrackName()
* public async void OnAppearing()
* privateIQueryable<UserRecord>getUserRecrodsOfCurrentUser
* private Iqueryable<UserRecord> getUserRecrodsWithTheSameTrackName()
* public async Task Logout()
* public async Task ToMapPage()
* public async Task DeleteMap(MapPin pin)
* public async Task DeleteSinglePin(MapPin pin)
* public void Refresh()
* public async Task ToTimerPage()
* public void ChangeConnectionStatus()
* private async Task<bool> CheckItemOwnership()
* public async Task GoToMapsList()
* partial void OnSelectedSortOptionChanged()
* public void SortUserRecords()
* async partial void OnIsShowAllTasksChanged()

**5.מחלקת LoginViewModel.cs**

מחלקה זאת אחראית על ההיתחברות הרשמה של המשתמש לאפליקיה וכוללת את השיטות הבאות :

* public async Task Login()
* public async Task SignUp()
* private async Task DoLogin()
* private async Task DoSignup()
* private async Task<bool> VeryifyEmailAndPassword()
* private async Task GoToMainPage()

**שכבת ההצגה -קבצי הxaml של Views**

תיקיית Views בפרויקט .NET MAUI מכילה את מסכי ממשק המשתמש של האפליקציה, לצורך ההיגיון. הוא ממלא תפקיד מכריע בבניית האפליקציה ובשמירה על ארכיטקטורת  
 MVVM (Model-View-ViewModel) נקייה. תקייה זו כוללת את כל הדפים של הXAML.

שכבה זאת מאפשרת בעצם לבצע הפרדה בין השכבה הלוגית שכוללת את כל השיטות שנועדו בשביל להראות חישובים כגון המרחק שהמשתמש עבר או שמירת אובייקטים בענן .לשם כך שכבת ההצגה מיוצגת בתקייה Views .

**עקרונות מנחים בשכבת ההצגה**

* לכל דף תצוגה בקובץ הXAML יהיה ,דף של קובץ הCS שבו ישמרו השיטות שבהם דף התצוגה משתמש .
* כל הכפתורים והעברה לדפים אחרים יבוצעו על ידי שיטות שעובדת עם RelayCommand . [RelayCommand] הוא תכונה (Attribute) בספריית CommunityToolkit.Mvvm, שנועד להגדיר פקודות (Commands) באופן פשוט ונוח בתוך ה-ViewModel בNetMaui . פקודות (ICommand) משמשות להפרדה בין הלוגיקה של האפליקציה לבין קוד ה-UI ומאפשרות חיבור בין כפתורים לאירועים מבלי לכתוב קוד ישירות בקובץ ה- xaml.cs.
* השימוש ב- RelayCommandחוסך כתיבה מיותרת על ידי הפיכת מתודות רגילות לפקודות אוטומטית, במקום ליצור ICommand באופן ידני.
* השימוש במטודה מסוג RelayCommand בקובץ הxaml,תכלול הוספה של המילה השמורה Command לשם המטודה .

הקבצים שמופיעים כחלק משכבת ההצגה יהיו בתקייה של Views כוללת את התיקיות הבאות והקבצים הבאים :

* תקיית MapPin -תקייה זו כוללת את קבצי הXAML שנעזרות באובייקטים מטיפוס MapPin ,כוללת את הדפים הבאים :

1. דף – EditMapPage.xaml
2. דף - MapsPage.xaml

* תקיית MapSettings-הכוללת בתוכה את הדפים שאחראים להגדרות של המסלולים , למשל הדף שאחראי לחישוב המרחק של המסלול .תקייה זו כוללת את הדפים הבאים:

1. דף - AddMapToDbPage.xaml .
2. דף- DistancePage.xaml .
3. דף - EditPinAddr.xaml .
4. דף - MapPage.xaml .

**4.דף הMapPage.xaml**

* דף הXAML הזה הינו הדף שכולל בתוכו את המפה של GOOGLE MAPS שבעזרתה גם ניתן לבנות מסלול חדש וגם לצפות במסלול קיים . הדף הזה בתוכו כפתורים שונים ולכן כולל את השיטות הבאות :
* public static MapPage Instance { get; }
* public string MapTitle { get; set; }
* public void SetTitle(string newTitle);
* public void setPinsList();
* public async Task<bool> IsLocationEnabled();
* public List<Maui.GoogleMaps.Pin> GetPinList();
* private List<Maui.GoogleMaps.Pin> GetPins();
* private void numberOfPoints(object sender, MapClickedEventArgs e);
* public async Task<Location> GetCurrentLocation();
* public void CancelRequest();
* private void CenterMap(double x, double y);
* public async Task GoToTimerPageButton\_Pressed();
* public async Task ZoomToMyLocationButton\_Pressed();
* public async Task GoToUserRecordsListButton\_Pressed();
* public async Task DeletLastPointButton\_Pressed();
* public async Task AddToCloudButton\_Pressed();
* public async Task CalcDistanceButton\_Pressed();
* public async Task ResetMapButton\_Pressed();
* public async Task EditPointButton\_Pressed();
* private void addPointOnMap(object sender, MapClickedEventArgs e);
* private void PrintPinAddresses(object sender, MapClickedEventArgs e);
* public void ShowButtonsOnMap(bool cond);
* public void ShowStartExerciseButton(bool cond);
* public void ShowUsersRecordsButton(bool cond);
* public void ShowTrack\_Clicked();
* private async Task<bool> EnoughPins(int num);
* public void ClearMap();
* private async void toPage(string pageName);
* private double GetDistance(Maui.GoogleMaps.Pin p1, Maui.GoogleMaps.Pin p2);

כאשר השיטות שנגמרות ב Pressed הם הכפתורים שנמצאים על גבי דף הXAML בשם MapPage.

**תקיית UserRecords**

כוללת בתוכה את קבצי הXAML שנעזרות באובייקטים מטיפוס UserRecord ודפי עזר שנועודו בשביל לעלות ערכים לאובייקט של UserRecord,למשל אובייקט מטיפוס UserRecord כולל בתוכה את הזמן הפעילות של המסלול -לכן הדף שכולל בתוכו את הטיימר יהיה בתקייה זו. וכוללת את הדפים הבאים :

* + AddRecordToDb.xaml
  + TimerPage.xaml
  + UserDetails.xaml
  + UserRecordsPage.xaml

דפוסי עצוב (Desigh Patterns)

SingltonPattern

תבנית עיצוב Singleton (סינגלטון) היא דפוס תכנון (Design Pattern) בתחום התכנות שמטרתו להבטיח שלמחלקה מסוימת תהיה אובייקט יחיד בלבד בכל התוכנית, ולספק גישה גלובלית אליו.

בתבנית זו נעשה שימוש במחלקות הבאות :

* 1. MapPage-מחלקה זו מייצגת את המפה של GOOGLE MAPS . מכיוון שבכל רגע נתון המשתמש יכול לקבל גישה למפה אחת של GOOGLE MAPS ,בין אם זה בכדי לבנות מסלול או להעיזר במפה בשביל לבחור מסלול בכדי לבצע פעילות אירובית כלשהי .
  2. TimerPage-מחלקה זו מייצגת את הטיימר של האפליקציה . מכיוון שבשביל למנוע מצב שבו משתמש בוחר מסלול מסויים ואז מחליט לבחור מסלול חדש אז שלא יווצר מצב שבו נעזר בזמן של המסלול הקודם יש צורך שבכל פעם שהמשתמש רוצה להפעיל את הטיימר חייב שיהיה אך ורק מופע אחד של הטיימר .

1. **Factory Pattern**

היא תבנית עיצוב בתכנות מונחה-עצמים (OOP). מדובר בתבנית יצירתית (creational pattern) שמטפלת במנגנוני יצירת אובייקטים, ומטרתה ליצור אובייקטים בדרך שמתאימה למצב. תבנית שיטת המפעל במיוחד מיועדת להגדיר ממשק ליצירת אובייקטים, אך מאפשרת לתת למחלקות תת-המחלקות להחליט איזו מחלקה ליצור. התבנית מאפשרת למחלקה להאציל את האחריות ליצירת אובייקטים למחלקות תת-מחלקות שלה, ובכך מקדמת את עקרון ההפרדה.   
בתבנית זו נעשה שימוש במחלקה הבאה :   
TypeFactory-מחלקה זו בעצם הוא מחלקה שממשת גם סינגלטון כך שבכל הפרוייקט יהיה רק מופע אחד של האובייקט הזה . השימוש במחלקה זו נעשה כאשר אנו רוצים לעלות לענן טיפוס UserRecord או MapPin . דבר זה קריטי מהסיבה שבמחלקה של RealmService בכדי לעלות אובייקט לענן יש צורך לבחור בשיטה GetQueryForSubscriptionMapPinType או GetQueryForSubscriptionUserRecordType בהתאם לטיפוס שאנו רוצים לעלות לענן .

**TypeFactory**

MapPin

UserRecord

1. **Command pattern**

**הדפוס Command** (או "דפוס פקודה") הוא דפוס עיצוב המתמקד בהפרדה בין אובייקט המנחה לבצע פעולה לבין הקוד שמבצע את הפעולה עצמה.  
במקום שהקוד המבצע את הפעולה יהיה ישירות בתוך האובייקט שמנחה אותה (למשל, כשלוחצים על כפתור והפעולה מתבצעת מיד), הדפוס מפריד את שני המרכיבים האלה. במקום זאת, כל פעולה (או פקודה) מנותבת לאובייקט נפרד שנקרא **Command** (פקודה), והאובייקט שמנחה יודע רק לשלוח את הפקודה לאובייקט שיבצע אותה. בגלל שאנו עובדים בDOTNETMAUI חלק מהלוגיקה שמקשרת בין דף הXAML לבין קוד ה#C . ישנה פקודה מיוחד שנקראת RelayCommand .במחלקות הבאות נעשה שימוש ב RealyCommand :  
1) LoginViewModel-זאת המחלקה שבעצם מאפשרת היתחברות לאפליקציה וכוללת בתוכה את השימוש בכפתורים הבאים (שנעזרים בcommand realy) :

* **Login**
* **VeryifyEmailAndPassword**
* **SignUp**
* **DoSignup**
* **Dolgin**
* **GoToMainPage**

**Execute**

**Commands**

GoToMainPage

DoLogin

Login

VerifyEmailAndPassword

DoSighup

SignUp

1. **Composite Pattern**

ה-Composite מאפשר ליצור מבנה של אובייקטים בצורה של היררכיה (עץ). בתוך המבנה הזה, אובייקטים יכולים להיות אובייקטים בודדים (leaf) או קבוצות של אובייקטים (composite). כל האובייקטים, בין אם הם בודדים או קבוצות, מממשים את אותה ממשק (interface) כך שניתן לטפל בהם באופן אחיד, בלי צורך לדעת אם מדובר באובייקט בודד או

בקבוצת אובייקטים.

EditMapPinViewModel-בעצם במחלקה זו ניתן לשמור את הפינים של המפה. כל מסלול שאנו מעלים לענן בנוי מפינים שבונים את המסלול . לכן תחת השדה של MapNameיהיה אותו השם אבל שאר השדות כגון קו רוחב,קו אורך ומספר הפין יהיו שונים .

MapPin1

* ObjectId
* Mapname
* Labelpin
* Address
* Latitude
* Longitude

pinsList

1. **Strategy Pattern**

תבנית עיצוב "אסטרטגיה" (Strategy Design Pattern) היא תבנית עיצוב מבני שמטרתה להפריד בין אלגוריתם (אסטרטגיה) לבין הקוד שמשתמש בו. במקום לקבוע אלגוריתם קבוע בתוך קוד מסוים, תבנית העיצוב מאפשרת לבחור ולהחליף אסטרטגיות שונות במהלך הריצה, מבלי לשנות את הקוד שבונה את המערכת.

במילים פשוטות, תבנית העיצוב הזו מאפשרת לבחור את הדרך שבה תתבצע פעולה כלשהי בזמן הריצה, ובכך מקלה על הוספת אסטרטגיות חדשות או שינוי אסטרטגיות קיימות מבלי לשנות את הקוד שהן פועלות בו.

דוגמה נפוצה לשימוש בתבנית זו היא במערכות שבהן יש אלגוריתמים שונים לפתרון בעיות שונות, כמו חישוב מחירים, חיפוש או מיון, ותצטרך לבחור באלגוריתם המתאים ביותר בהקשר נתון.

UserRecordsViewModel- בגלל שהאפליקציה כוללת בתוכה את האפשרות למיון של רשומות המשתמשים . במחלקה זו ישנה את השיטה SortUserRecords- שמאפשרת למיין (לפי SelectedSortOption ) את הרשימה UserRecordsList (שמציגה את הרשומות של המשתמשים באפליקציה )לפי זמן הפעילות,שם המשתמש שביצע את הפעילות או זמן העלאת הרשומה לענן של מונגו די בי .

1. **Observor Pattern**

בנית עיצוב "צופה" (Observer Design Pattern) היא תבנית עיצוב התמקדת בקשר בין אובייקטים, כך שכאשר מצב של אובייקט משתנה, כל הצופים (Observers) שמנויים על אותו אובייקט מעודכנים אוטומטית על השינוי.

במילים פשוטות, תבנית העיצוב הזו מאפשרת לאובייקטים לעקוב אחרי שינויים באובייקט אחר בלי שהם יצטרכו להיות תלויים אחד בשני באופן ישיר. ברגע שהאובייקט הנתון משתנה, כל הצופים שמנויים עליו מקבלים את העדכון ומבצעים פעולה בהתאם.

דוגמה נפוצה לשימוש בתבנית זו היא במערכות המידע, שבהן יש צורך לעדכן מספר רכיבים (כמו ממשקי משתמש, מערכות ניהול או שירותים אחרים) כאשר מתרחשים שינויים במידע המרכזי.כחלק מהסיפרייה של CommunityToolkit.MVVM ישנו מאפיין שנקרא ObservableProperty . ObservablePropertyהוא רכיב מרכזי ביישום תבנית העיצוב "צופה", שבה שינויים בתכונות מעדכנים אוטומטית את הממשק המשתמש (UI) .

ראש הטופס

תחתית הטופס

UserRecordsViewModel-במחלקה זו קיימת הרשימה userRecordsList שהיא ObservableProperty ,רשימה זו מציגה את הרשומות של המשתמשים לפי אפשרות של מיון או הצגה של הרשומות של אותו המשתמש שמחובר לאפליקציה בלבד .

MapsViewModel-במחלקה זו קיימת הרשימה maps-רשימה זו מציגה את המסלולים של כל המשתמשים שבנו המשתמשים .

1. **Fascade Pattern**

תבנית עיצוב פאסייד (Facade Design Pattern) היא תבנית עיצוב שמטרתה לפשט את הגישה למערכת מורכבת על ידי יצירת ממשק פשוט שמסתיר את הפרטים המורכבים של המערכת. במקום שהמשתמשים או המפתחים יצטרכו להתמודד עם אובייקטים וממשקים שונים ומורכבים, הפאסייד מספק ממשק אחיד ונגיש שקל לעבוד איתו.

במילים פשוטות, תבנית הפאסייד יוצרת "חזית" או "קירות" שמסתירים את המורכבות של המערכת מאחוריהם, ומספקים למשתמש גישה פשוטה ונוחה לפונקציות השונות מבלי להסתבך בפרטים.

דוגמה לשימוש בתבנית זו היא במערכות עם רכיבים רבים וכוללים (כמו מערכות לניהול נתונים, חיבור לרשתות, או מערכות גראפיות), שבהן יש צורך להסתיר את המורכבות ולספק ממשק פשוט לשימוש עבור המפתחים או המשתמשים.

MapsViewModel-במחלקה זו קיימת למשל השיטה DeleteMap,שבעצם מוחקת מסלול מהענן של REALM על ידי שימוש בשיטה DeleteSinglePin שנעזרת בפקודה של Remove שהיא חלק מהסיפרייה של Realm.Sync.

UserRecordViewModel-במחלקה זו קיימת השיטה DeleteUserRecord שמאפשרת למשתמש למחוק את הרשומות שהוא יצר . המחיקה של הרשומה גם כאן מתבצעת בעזרת פקודה של Remove שהינה חלק מהספרייה של Realm.Sync .