המחלקה להנדסת תוכנה

פרויקט גמר – תשע"ח

עיר יער

City Forest

מאת

זהבה פלדמן

מנחה אקדמי: גב' מיכל גולדשטיין	**אישור:**	**תאריך:**
אחראי תעשייתי: יונתן אלון	**אישור:**	**תאריך:**
רכז הפרויקטים: מר אסף שפנייר	**אישור:**	**תאריך:**

#	מערכת	מיקום
1	מאגר קוד	https://github.com/ZehavaFeldman/City-Forest
2	יומן	https://trello.com/b/m0IVneZn/city-forest/calendar
3	סרטון	

מבוא

"ישראל" ארץ אשר בה העבר והעתיד משמשים בערבוביה. לצד בניני היי טק מעוצבים ברוח השעה עומדים להם בנינים הסופנים בתוכם היסטוריה עתיקת יומין. לצד פארקים חדישים עומדים להם זה אלפי שנים יערות בהם נלחמו המכבים או לחילופין הצלבנים.

ההיסטוריה מבצבצת לה מכל פינה, מערות קבורה עם מערות מלחמה. שרידי חומות של ערים בצורות עם אגמים ונחלים לרוב. הילת קדושה אופפת אותה ומזמינה אליה תיירים מכל העמים ומכל הדתות.

טבע ומיסטיקה אומנות וחדשנות חובר לו לפסיפס מרהיב. והכל קיים כאן וליד היכן שאתה נמצא.

פרויקט זה תכליתו להנגיש למטייל מכל נקודה ציבורית בה הוא נמצא את מקומות הטבע וההיסטוריה הסמוכים אליו. כך שבכל נקודת מוצא יוכל להתחבר אל ההיסטוריה שבה או ליהנות מן הטבע הקיים. להכיר וללמוד או פשוט ליהנות.

מילון מונחים

עיר יער - שם הפרויקט.

טייל- משתמש של האפליקציה.

פורצי דרכים – משתמשים בעלי הרשאות עריכה. משתמשים אלו יכולים להוסיף, למחוק ולעדכן מסלולים ונקודות העניין.

מסלול- אובייקט המורכב מרצף של נקודות על גבי המפה אשר מתחיל ומסתיים בתחנת רכבת. אובייקט זה גם מכיל נתונים המאפיינים את המסלול כגון רמת קושי וכו'. מסלול מוצג על המפה בעזרת polyline.

נקודת עניין- אובייקט המכיל נתונים על מקום בקרבת מסלול. נקודות אלו מאפשרות למשתמש להעשיר את הידע שלו בזמן הטיול. ישנם סוגים שונים של נקודות עניין ולכל אחד אייקון אחר**.** נקודות אלו מוצגות על המפה בעזרת Marker**.**

התראה - אובייקט המכיל נתונים על עדכוני שטח בזמן אמת. נתונים אלו ניתנים להוספה ע"י כל משתמשי האפליקציה. ההתראות מוצגות על המפה בעזרת Marker.

Firebse - הממשק של בסיס הנתונים באפליקציה, נותן גם שירות לביצוע אותנטיקציה.

עצי ג'ייסון (Json Trees) **-** אופן שמירת הנתונים בבסיס הנתונים של ממשק Firebase. כל ענף בעץ מזוהה באמצעות מזהה ייחודי (key) המקנה גישה למידע בתוך הענף.

Mapbox – הממשק ליצירת מפה אינטראקטיבית באפליקציה.

Marker – אובייקט השייך לממשק המפות של Mapbox, מייצג נקודה מסומנת על המפה עליה אפשר ללחוץ ולקבל נתונים אודות אותה נקודה.

Polyline – אובייקט השייך לממשק המפות של Mapbox, קו המייצג חיבור של מספר נקודות על המפה.

תיאור הבעיה

דרישות ואפיון הבעיה:

עיר יער היא מערכת קיימת שבכללותה נועדה ליצור קישוריות בין תחנות תחבורה ציבורית לשבילי הליכה בטבע.

המערכת הקיימת כוללת:

1. אפשרות יצירה ומחיקה של נקודות עניין ומסלולים וכן עדכון של מסלול.
2. הצגת מסלולים ונקודות עניין על גבי המפה.

מטרתו של הפרויקט הנוכחי:

1. שיפור ממשק המשתמש:

ממשק המשתמש הקיים מאד לא ידידותי ונוח לשימוש, כאשר הדגש הוא בצד העורך. הנתונים המוצגים על גבי המפה לא ברורים וכל האופן בו יוצרים, עורכים ומוחקים ישויות מהמערכת מסורבל ולא אינסטינקטיבי. גם בצד ה'טייל' נתקלים בקשיים כגון קריאת תוכן המתנגש עם השימוש במפה וכו'

2. הוספת אפשרות סינון וחיפוש:

במצב הקיים ככל שבסיס הנתונים של המערכת תגדל האפליקציה תיעשה כבדה ואיטית יותר; באפליקציה אין אפשרות של סינון או חיפוש, זאת אומרת שהמערכת תטען תמיד את כל המסלולים והנקודות הקיימות במבנה נתונים, כשלרוב אין צורך בכך.

בנוסף, האופן היחיד שהמשתמש יכול למצוא מסלול או כל נק' עניין הוא ע"י חיפוש ידני על גבי המפה.

3. הוספת מערכת התראות זמן אמת:

בתכנון טיול, הפרטים הקטנים המשתנים מעת לעת- כגון גשם, עומס, מקום בשיפוץ, סכנות וכד' משמעותיים מאד.

במערכת הקיימת יכול המשתמש לראות את מיקומו הגאוגרפי של ישות על גבי המפה, ולקרוא תיאור קצר ופרטים טכניים שמוזנים מראש ע"י יוצר המסלול, אך אין כל התייחסות לפרטים המשתנים. לכן, נרצה לאפשר למשתמשי האפליקציה בזמן אמת לשתף את יתר המשתמשים במידע אישי שלהם אודות המקום בו הם נמצאים, או לחילופין לצפות בתוכן שמשתמשים אחרים הזינו.

תיאור הפתרון

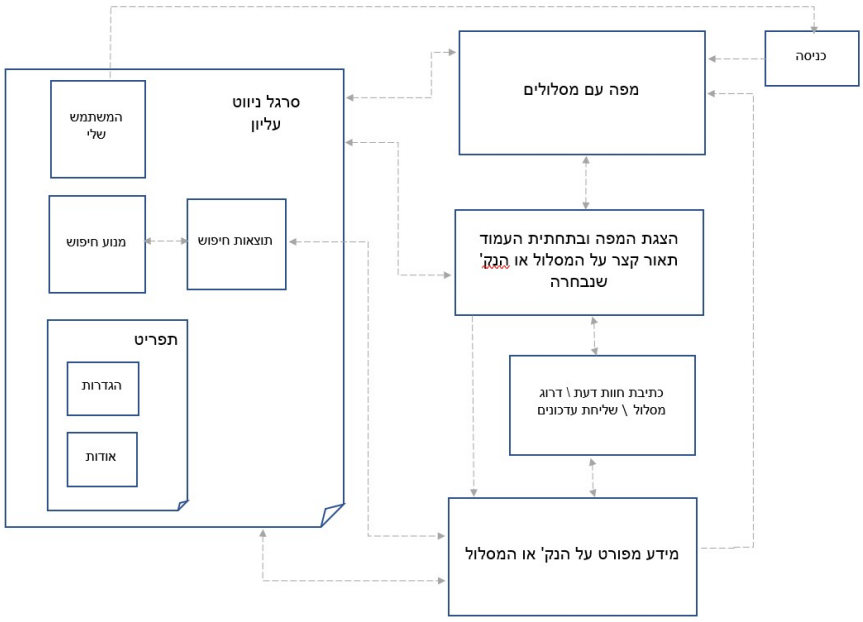
מהי המערכת:

אפליקציה מבוססת אנדרואיד המציעה למשתמש מסלולי טיול מתחנת רכבת קלה אחת אל תחנה אחרת. המערכת תאפשר למשתמש לבצע חיפוש חופשי או לחילופין סינון ע"פ קריטריונים. לאחר החיפוש ו/או הסינון יוצג למשתמש תוצאות החיפוש ע"ג המפה. לחיצה על נקודה במפה תפתח דיאלוג בתחתית המפה שתציג בפניו פרטים עיקריים אודות המסלול או נקודת העניין.

בנוסף, המשתמש יוכל לקרוא מידע נרחב יותר, לשתף את המשתמשים האחרים בעדכונים מהשטח וחוות דעתו האישית- דרוג מסלול וכד', מה שיהפוך את האפליקציה לרשת חברתית.

האפליקציה תומכת בריבוי שפות.

להלן תרשים המציג את המסכים החדשים והמעבר ביניהם.



כמה נקודות לגבי התרשים:

א. המסכים משותפים לשני סוגי המשתמשים -ה'פורצי דרכים' וה'טיילים'. המעבר ממצב אחד לשני יתבצע ע"י בחירת האופציה ב'משתמש שלי'. במקרה של מעבר למצב עריכה המשתמש יתבקש להתחבר ע"י שם משתמש וסיסמא.

ב. הסרגל ניווט משותף לכל המסכים באפליקציה מה שאומר שהחיפוש הוא כללי לכל המערכת, וניתן להגיע אליו וממנו לכל מסך באפליקציה.

תהליכים ונתוני המערכת:

המערכת מטפלת בשני צדדים עקרים:

א. **צד הלקוח-**

- ממשק המשתמש של האפליקציה- נרצה ליצור ממשק חדש שיהפוך את האפליקציה משתי אפליקציות המחוברות ע"י מסך פתיחה, לאפליקציה אחת חכמה שתדע לתמוך במצבים וסוגי משתמשים שונים.
- ממשק החיפוש שיציג למשתמש מידע ע"פ הקלט שהזין. ממשק זה יתממשק מול ספריה חיצונית שתבצע את מנגנון החיפוש.
- התכונה החדשה של שיתוף מידע בין משתמשים- דורשת קריאות רבות לקבלת ושמירת נתונים בשרת. אם נבצע את קריאות אלו בתהליך הראשי של האפליקציה המכונה UI thread שכן תפקידו להריץ את כל מה ששייך לממשק המשתמש זה עלול להאט את הזרימה הכללית של האפליקציה, לכן נריץ תהליך נפרד מאחורי הקלעים שיטפל בקריאות אלו.

ב. **צד השרת-**

- מטפל באחסון וקבלת המידע המוצג למשתמש. מבנה הנתונים בנוי בצורה של עצי ג'ייסון (key/value). כל Reference הוא עץ המכיל ילדים רבים, שכל אחד מהם בעצמו מהווה עץ, כאשר ה key הוא המזהה הייחודי שלו וה value זה רשימות התכונות המאפיינות אותו. (תמונות של מסד הנתונים ניתן למצוא בנספחים- סעיף ה')

מסד הנתונים מכיל כיום שלושה References:

- מסלולים – כל המסלולים המוצגים למשתמש על המפה. לכל מסלול יש תכונות כמו האם המסלול מכיל מים, מתאים למשפחות וכו' ואובייקט מסוג Route שמכיל את כל הנקודות של המסלול .
- קורדינאטות – נקודות על המפה שהמשתמש יכול לבחור להרכיב מהם מסלול.
- נקודות עניין- נקודות מעניינות על המפה כגון בתי קפה, אתרים היסטוריים וכו'

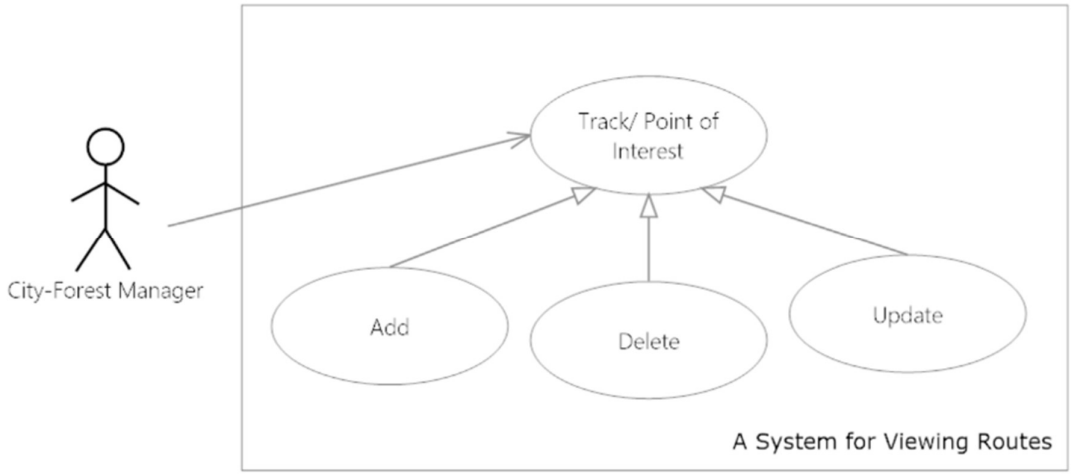
אנו נוסיף reference חדש שיהיה אחראי על שמירת תוכן שמשתמשי האפליקציה מזינים ושיתוף המידע עם יתר משתמשי האפליקציה בזמן אמת. נתונים אלו דינאמיים מאד שמציאותם במסד הנתונים היא תלויית זמן (כגון עומס, גשם וכד') ולכן דרך שמירתם במסד הנתונים שונה מהותית מנתונים סטטיים שמרגע שהוכנסו לא משתנים.

גם עצי הג'ייסון הקיימים כבר בבסיס הנתונים דורשים בנייה ותכנון מחדש- הוספה, ומחיקה של values בתתי העצים. למשל, ב Reference של מסלולים, נוסיף תכונות השייכות לדרוג מסלולים- נרחיב על כך בהמשך בתיאור הפתרון.

- מטפל בסינון המידע המוצג למשתמש. נשתמש בספריה של Algolia המקבלת כקלט מחרוזת ומחזירה לאפליקציה רשימה של נתונים המכילים מחרוזת זו. מעבר להתממשקות מול הספרייה צד השרת יהיה אחראי גם לשמור על סנכרון בין המידע של Firebase ו Algolia.

נציג את תרשימי השימוש המתארים את התהליכים במערכת:

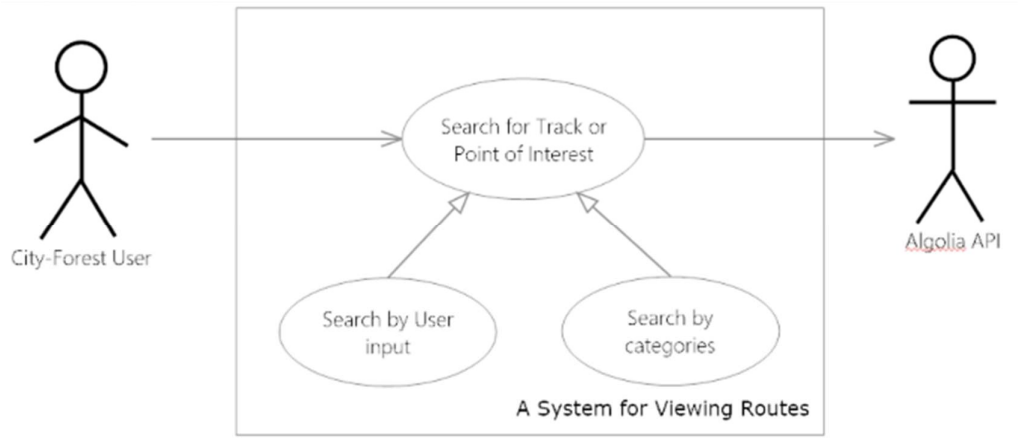
א. עריכת מסלול ונקודות עניין-



צוות הפרויקט אחראי על הוספת ועדכון המסלולים במערכת. יש צורך בשני אפשרויות עריכה:

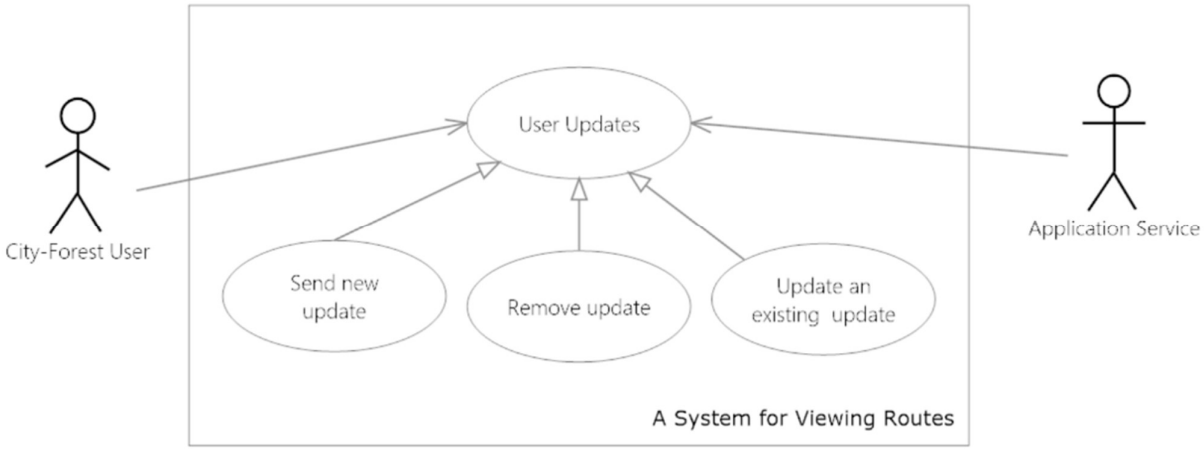
1. שינוי פרטים אישים של המסלול,(כגון רמת קושי, התאמה למשפחות וכו')
2. שינוי המסלול עצמו וזאת ע"י שנאפשר הוזזה של נק' על המפה.

ב. חיפוש מסלול ונקודות עניין



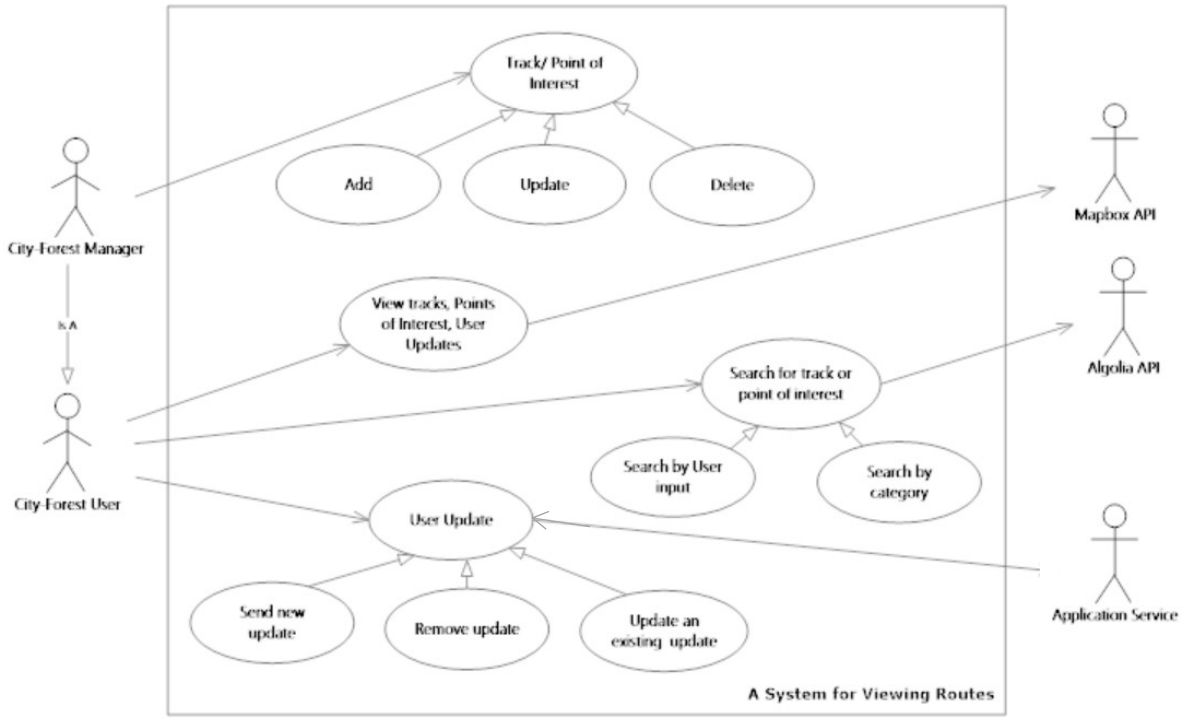
נרצה לאפשר חיפוש חופשי או חיפוש ע"י קטגוריות של מסלולים ונקודות עניין שקיימים במערכת.
לצורך כך נשתמש בממשק החיפוש Algolia, המסוגל לחפש טקסט בתוך שדות של עצי ג'סון.

ג. עדכוני שטח



נרצה להעניק למשתמש אופציה לשלוח עדכונים מזדמנים והצגתם בזמן אמת על המפה. לשם כך נריץ תהליך נפרד באפליקציה שיופעל ברקע וינהל את נתונים אלו. בנוסף, יש את ממשק המפות שמציג את האייקון המתאים על המפה.

ד. כלל המערכת



תיאור הפתרון המוצע:

<u>מנגנון החיפוש של האפליקציה</u>

כיוון ש Firebase לא תומך בשליפת נתונים ע"י שאילתות מרוכבות הייתה חשיבה רבה כיצד לאפשר חיפוש וסינון של נתוני המערכת, שכוללים בין היתר מסלולים ונקודות עניין, בצורה שתשפיע כמה שפחות על חווית המשתמש. שכן זהו תכונה דומיננטית מאד במערכת.

נציג את שתי האפשרויות שעמדו בפנינו:

1. אחסון המידע בצד הלקוח – שימוש ב SQLite ליצירת מסד נתונים מקומי בנוסף למסד הנתונים של Firebase, וכתיבת שאילתות מורכבות לשליפת הנתונים המתאימים לחיפוש או הסינון הרצוי. מסד נתונים זה ישב על המכשיר של המשתמש ויכיל רק את המידע שניתן לסינון וחיפוש- טבלה של מסלולים וטבלה של נקודות עניין.
2. שימוש בממשק חיצוני Algolia, ממשק המוצע ע"י Firebase כדי לבצע חיפוש של טקסט או סינון ע"פ קטגוריות. החיפושים יתאפשרו רק במידה ויש חיבור לאינטרנט

לכל אחת מאפשריות ישנם יתרונות וחסרונות, נציג מספר מהן עבור כל אחת מהאפשרויות:

אחסון המידע בצד הלקוח

יתרונות	חסרונות
אין צורך בחיבור לרשת לבצע חיפוש	ככל שמסד הנתונים יגדל האפליקציה תעשה כבדה יותר ותעמיס יותר על המכשיר
אינו דורש שימוש בספריות חיצוניות, כל הספריות קיימות בסביבת העבודה של אנדרואיד סטודיו.	כדי לבצע את הסנכרון נצטרך לבצע קריאות רבות לשרת לקבלת הנתונים ממסד הנתונים של Firebase, השוואת הנתונים המתקבלים עם הנתונים הקיימים במסד הנתונים המקומי ועדכון במסד הנתונים המקומי בעת הצורך.
	שימוש רב במשאבים של האפליקציה שגורמים לביצועים נמוכים יותר.

שימוש בממשק חיצוני Algolia

יתרונות	חסרונות
תהליכי החיפוש מתבצעים בספריה החיצונית כך שנעשה פחות שימוש במשאבי האפליקציה- ביצועים גבוהים יותר	דורש חיבור לרשת על מנת לבצע את פעולת החיפוש ולהציג את המידע למשתמש
אין צורך בכתיבת שאילתות מכל סוג שהן- מהווה חסכון גדול בכתיבת קוד, וכן מאפשר לבצע שינוי במאפייני הישויות במערכת ללא עדכון נוסף של פונק' החיפוש.	מצריך לימוד אישי של שפות לפיתוח צד שרת כדי לבצע את הסנכרון.
ספריה המוצעת ע"י Firebase לביצוע חיפושים וסינונים במסד הנתונים. ה	

ניתן לראות כי לכל אפשרות יש את היתרונות והחסרונות משלה, והינו רוצים למצוא פתרון שיאפשר חיפוש ללא צורך בחיבור לאינטרנט- יתרון המתייחס לאחסון המידע בצד הלקוח, ומאידך לחסוך כמה שיתור עבודה בתהליך הראשי של האפליקציה - יתרון המתייחס לשימוש בממשק חיצוני.

בהסתכלות כוללת על הפרויקט, אנו חייבים מנגנון שיעבוד בצורה חלקה, מבלי להכביד על האפליקציה, לכן בחרנו להשתמש בממשק של Algolia.
כיוון ש Algolia לא מציעה כל אפשרות של Offline לא נוכל להתגבר על החיסרון המחייב חיבור לאינטרנט, והפיתוח של הפרויקט יעשה עם מודעות לחיסרון זה.

לאומת זאת, על החיסרון של לימוד שפות צד שרת נוכל להתגבר ונעשה זאת ע"י לימוד אישי ממדריכים באינטרנט וכן בעזרתה של חברת צוות מהעבודה שתסייע בנושא זה.

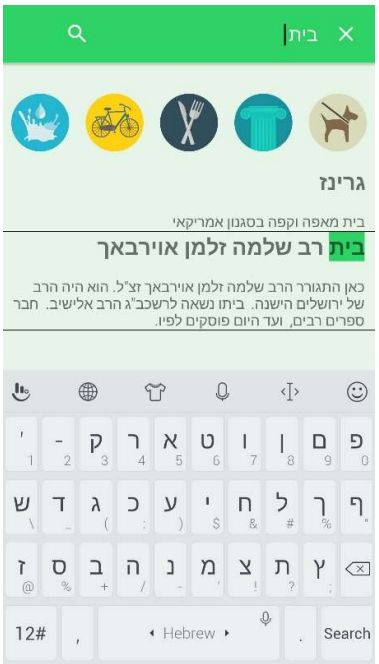
בשימוש בספרייה זו נבצע את החיפוש באופן הבא:

1. צד המשתמש באפליקציה ישגר שאילתה למשק עם הטקסט שהזין המשתמש ולעיתיים גם קטגוריות סינון שבחר.
2. ממשק החיפוש מחשב את השאילתה ומחזיר רשימה של מסלולים ונקודות עניין העונים לקריטריונים.
3. האפליקציה מקבלת את הרשימה ומציגה אותה על המסך.

בפיתוח באנדרואיד, נשתמש ב ListView כדי להציג למשתמש רשימה של נתונים על המסך. הנתונים ב ListView מאוכלסים באמצעות מתאם, המתאם מקבל מערך של אובייקטים וממיר אותם ל Views. במסך שלנו נרצה להציג ברשימה אחת שני סוגי אובייקטים- מסלולים ונקודות עניין. כיוון שלכל ListView יש מתאם יחיד, וכל מתאם יכול להמיר סוג אחד בלבד של אובייקט, יצרתי מחלקה חדשה שמרחיבה את המחלקה ArrayAdapter. מחלקה זו מקבלת מס' בלתי מוגבל של מתאמים ומצרפת אותם יחד למתאם אחד אותו לבסוף נשייך ל ListView.

חיפוש ע"י טקסט חופשי או סינון ע"פ קטגוריות

הצגת הנתונים למשתמש



<u>עריכת מסלול על גבי המפה</u>

כאשר משתמש עורך או יוצר מסלול הוא בוחר את הנקודות שמהם יורכב המסלול, כיום במקרה שהמשתמש טעה והקיש על מיקום לא נכון, הוא יצטרך למחוק את המסלול שיצר וליצור מסלול חדש עם הנקודות הנכונות. לכן ,על מנת לתקן את חווית המשתמש הוספתי אפשרות לתיקון מיקום שגוי ע"י גרירה פשוטה של הנקודה למיקום הנכון.
ממשק המפות של Mapbox לא תומך באפשרות כזו, לכן נדרש מימוש של פונקציונליות הגרירה באופן עצמאי .

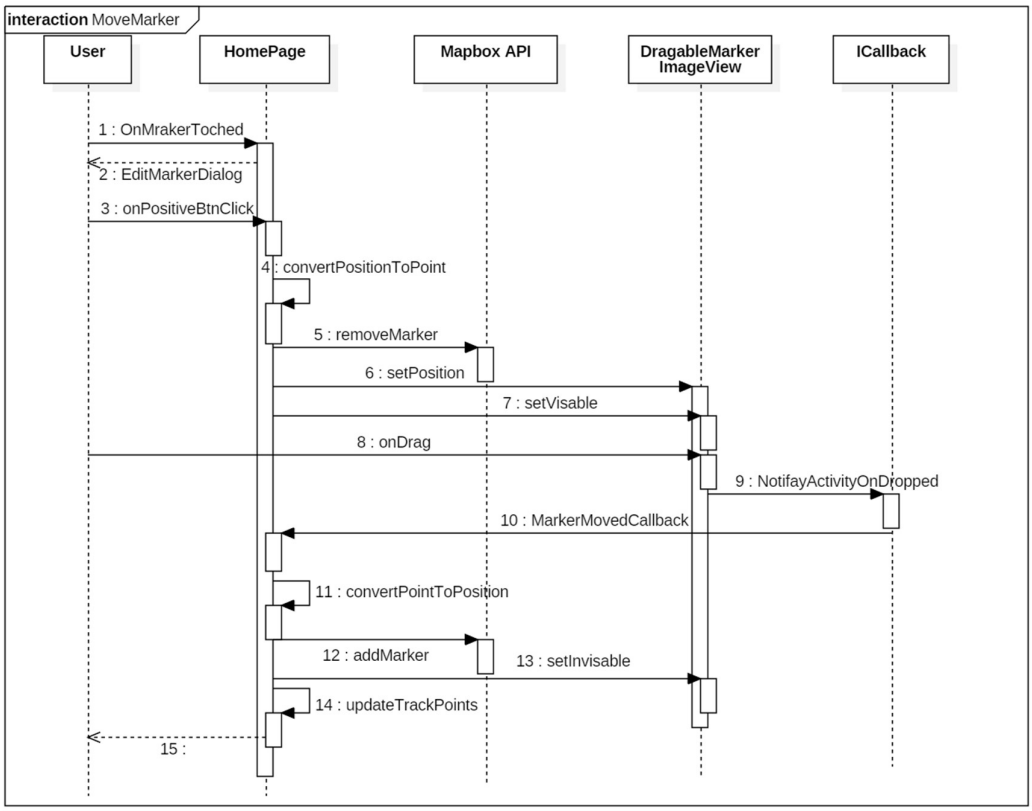
ממוש הפונקציונליות:
בהתחלה חשבתי להרחיב את המחלקה Marker של Mapbox. למחלקה הקיימת ישנו מאזין שמאזין ללחיצה של המשתמש על ה Marker. המחשבה הייתה לדרוס את הפונקציונליות של מאזין זה ושם לבצע את הגרירה. אך בפתרון זה נתקלתי בשתי בעיות -

1. כיוון שה Marker הוא חלק מהמפה נוצרו קונפליקטים בין הזזה של המפה להזזה של הנקודה.
2. המערכת לא תמיד הבחינה בין אירוע של לחיצה רגילה- הצורך בהצגת חלונית מידע, לבין לחיצה ארוכה ששייכת לגרירה.

המשכתי לחפש דרך להתגבר על בעיות אלו, אלא שאז פתחתי את המאגר קוד של Mapbox ב GitHub ושם קראתי כי דרך זו לא מומלצת וגורמת לבעיות רבות בתקשורת של המפה עם ה Marker.
לכן, הבנתי שפתרון זה לא מעשי ועברתי למימוש אחר- פתרון המתבסס על הרחבה של המחלקה ImageView של אנדרואיד סטודיו. ולמחלקה זו הוספנו את הפונקציונליות של הגרירה. כאשר אובייקט זה משמש ככלי עזר לביצוע הגרירה.

תהליך הגרירה כולל ארבע שלבים עיקרים:
1. הסרה של Marker מהמפה והוספה של אובייקט הגרירה במקום זה.
2. גרירה של האובייקט על המפה
3. הסרה של אובייקט הגרירה והוספה של Marker במיקום החדש.
4. עדכון הנקודה במערך המכיל את נקודות המסלול

תיאור תהליך הגרירה באמצעות התרשים הבא:



תיאור התרשים:

בעקבות לחיצה של המשתמש על נקודה במפה יוצג לו חלון שמוודא כי ברצנו לשנות את מיקום הנקודה. לחיצה של המשתמש על אישור מתחיל את תהליך הגרירה.

פונקציית convertPositionToPoint תחשב ותחזיר את מיקומו של הנקודה ביחס למסך של המכשיר.

לאחר מכן נסיר את הנקודה מהמפה ע"י removeMarker של Mapbox וביקום זה נמקם ונציג את אובייקט של המחלקה DragableMarkerImageView המשמש אותנו לגרירה עצמה. setPosition משנה את המיקום ו setVisable מציג אותו למשתמש.

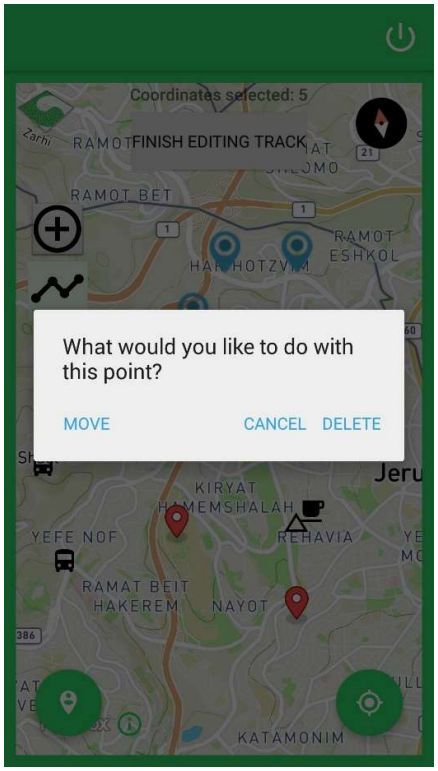
המשתמש מזיז את האובייקט החדש למיקום הרצוי באמצעות onDrag המאזין לאירוע גרירה.

כשמסתיים פעולת הגרירה DraggableMarkerImageView מודיעה על כך ל HomeActivity באמצעות MarkerMovedCallback של ICallback

convertPointToPosition של HomeActivity ממיר את המיקום בסמך לנקודה במפה.

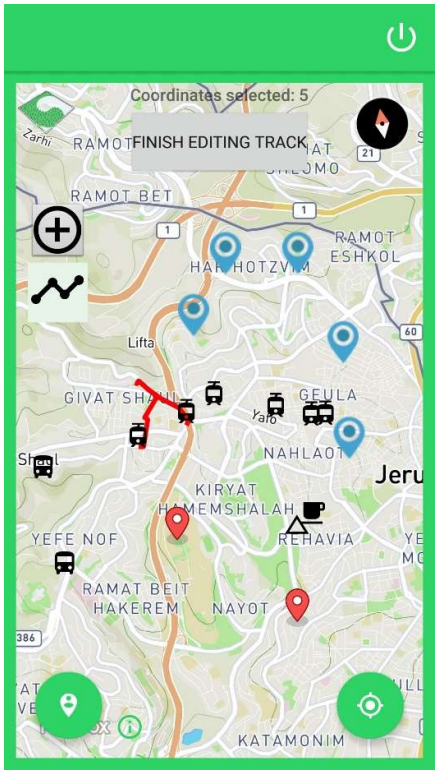
אובייקט הגרירה מוסר מהמסך באמצעות setInvisable של DraggableMarkerImageView ובמיקום זה נוסף Marker חדש על המפה באמצעות addMarker של Mapbox.

HomeActivity מעדכן את רשימת הנקודות של המסלול במיקום החדש באמצעות updateTrackpoints



לחיצה על נקודה במסלול

לחיצה על הזזה – המשתמש יוכל להזיז את הנקודה



<u>זיהוי מסלול על המפה</u>

כדי שמשתמשי האפליקציה יוכלו לחפש מסלולים לפי מיקומם הפיזי נרצה לאפשר להם לבחור מסלול בלחיצה על מסלולים במפה.

ממשק המפות מספק לנו מאזין שמאזין לאירוע של לחיצה על המפה, אך לא מזהה אם המשתמש הקיש על מסלול במפה. על כן אנו נוסיף בדיקת התאמה בין נקודת הלחיצה של המשתמש לנקודה במסלולים הקיימים במערכת.

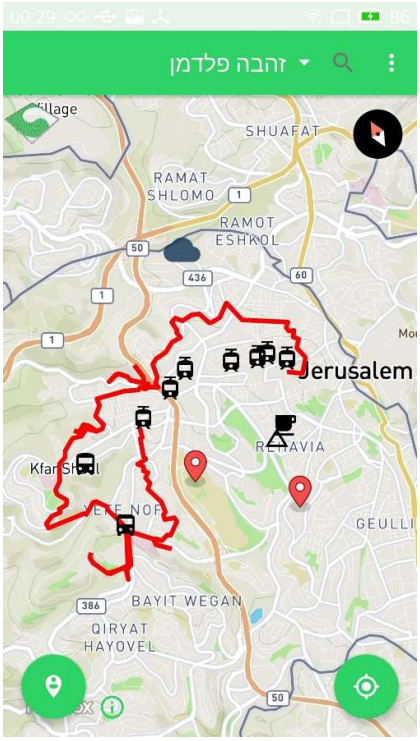
נגדיר התאמה להיות מרחק מקסימלי של 100 מ' בין נקודת הלחיצה לנקודה במסלול. נתאר את תהליך הזיהוי בכמה שלבים:

- המרה של נקודת הלחיצה מפיקסלים לנקודת LatLang המאפיין מיקום על המפה.
- מעבר על רשימת ה Polylines הקיימים על המפה ומשמשים להצגת המסלולים.

- עבור כל Polyline נעבור על הנקודות המרכיבות אותו ונחפש התאמה בינו לנקודה שהמרנו.

נתאר את תהליך קבלת הנתונים והצגתם למשתמש:

- המרה של ה Polyline לאובייקט מסוג Route.
- שיגור שאילתה לשרת לקבלת נתוני המסלול ממסד הנתונים. פרמטר השאילתה הוא ה Route.
- הצגת חלונית בתחתית המסך עם נתונים עיקריים של המסלול.



זיהוי לחיצה על מסלול- הצגת חלונית בתחתית המסך

<u>שיתוף המידע בין משתמשי האפליקציה</u>

האפליקציה מתנהלת כרשת חברתית ומאפשרת לכלל המשתמשים לדרג מסלולים וכן לסמן 'לייק' למסלול מעודף. כדי ליהנות מאפשרויות אלו על המשתמש לבצע Login למערכת.

עבור כל מסלול במסד הנתונים נשמרים הדרוגים שבוצעו המשתמשים, ולכל דרוג מי המשתמש שביצע אותו. באופן זה מנהל האפליקציה יוכל לעקוב על רמת הדרוג בקרב המשתמשים.

שמירה של נתונים אלו במסד הנתונים דורש פונקציונליות שונה מעדכון סטנדרטי של Firebase המתבצע בקריאה פשוטה של databaseReference.updateChild(). שכן, ישנה בעיתיות במקרה שמספר משתמשים ירצו לעדכן דרוג לאותו מסלול בו זמנית - מונה הדרוגים עלול להיות שגוי. לכן

נשתמש ב Transaction לביצוע פעולה זו. בקריאה זו נשלח לשרת שני ארגומנטים - פונקציית עדכון ו callback למקרה של הצלחה. פונקציית העדכון תקרא כל עוד העדכון נכשל בשל עדכון של משתמש נוסף. בצורה זאת מובטח לנו שחוות דעתם של כל המשתמשים ישמר ולמשתמש יוצגו נתונים אמתיים. בשימוש ב Transaction, כמו בכל קריאת שרת אחרת של Firebase במקרה של כישלון בגישה לשרת או לחילופין כישלון של שמירה בעקות חוסר מקום במסד הנתונים וכד' יוחזר Error המודיעה על כך. במקרה כזה נציג למשתמש הודעה מתאימה.

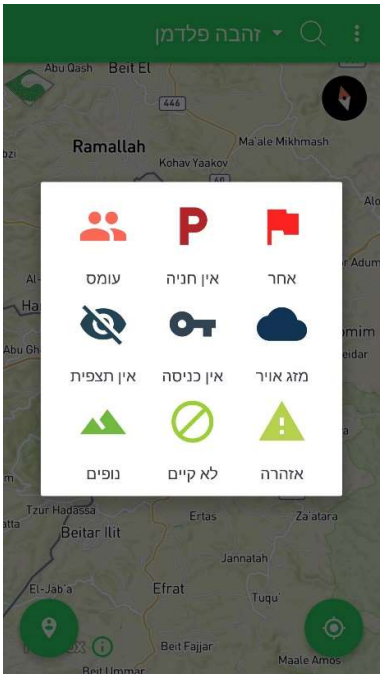
(בתמונות מסך המוצגות למעלה ניתן לראות את החלונית עם אפשרויות הדרוג)

<u>ניהול ההתראות</u>

משתמש יכול להוסיף דיווחים מהשטח, המערכת תקלוט דיווחים חדשים ותשלח התראות בזמן אמת לכלל המשתמשים.
במערכת קיימים סוגים שונים של דיווחים כאשר לכל סוג של דיווח יש את משך הזמן שבו הוא יהיה בתוקף מרגע היצירה. במקרה שהמשתמש הוסיף התראה המערכת תוסיף אותה לשרת. בנוסף, אם משתמש אחר אישר שהדיווח עדין תקף, זמן היצירה של הדיווח יעודכן לזמן הנוכחי.
כדי שהנתונים המוצגים על המפה יתעדכנו בזמן אמת יצרתי Service המופעל ברקע האפליקציה.
ב service יש Interval המופעל כל X שניות ומבצע שתי קריאות:

1. קריאה שתחזיר רשימה של כל ההתראות החדשות מאז הקריאה האחרונה.
2. קריאה עם שאילתה- databaseRefernce.orderBy("timestamp").endAt(curtime) שתחזיר רשימה של כל ההתראות שפג תוקפן ביחס לזמן הנוכחי.

במידה והתקבלו נתונים, ה service יחזיר callback לתהליך הראשי של האפליקציה שבו יתבצע פעולת ההוספה או לחילופין מחיקה של ההתראות מהמפה. ה service גם אחראי למחיקת התראות ישנות מהשרת באמצעות פונקציה מובנית של Firebase למחיקת רשומות ע"י קריאה ל- dataSnopshot.getRef().removeValue().



שליחת עדכונים מהשטח

(בתמונה המצורפת בתמיכה בריבוי שפות ניתן לראות החלונית המוצגת למשתמש בעת לחיצה על התראה במפה)

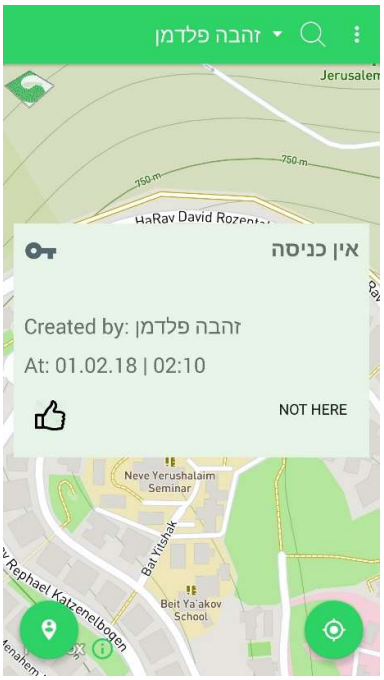
<u>תמיכה בריבוי שפות:</u>

על מנת לתמוך בריבוי שפות הוספתי מספר קבצי string resources בפורמט xml ששומרים את כל המחרוזות הקימות באפליקציה, כאשר כל קובץ שומר מידע עבור שפה אחרת. שמות הקבצים מכילים בסופן את ה symbol של הארץ אליהם שייכים (לדוגמא string-iw.xml).

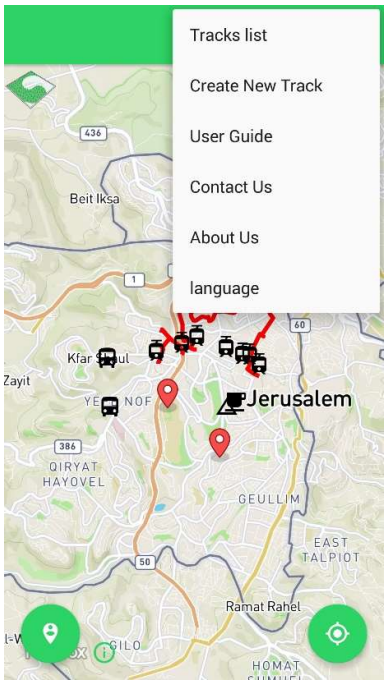
האפליקציה מזהה מה היא השפה המוגדרת של המכשיר ולפי זה בוחרת את הקובץ המתאים. בנוסף המשתמש יוכל לשנות את שפת האפליקציה, בחירה זו תפעיל פונקציה שתשנה את קובץ ה resource של המערכת.

בהגדרות של האפליקציה ניתן להוסיף תמיכה מובנית של שפות הנכתבות מימין לשמאל - "android:supportsRtl="true , אך הוספה זה במקרים רבים לא מספיקה כדי שהמסכים יהיו מותאמים לשני הכיוונים.

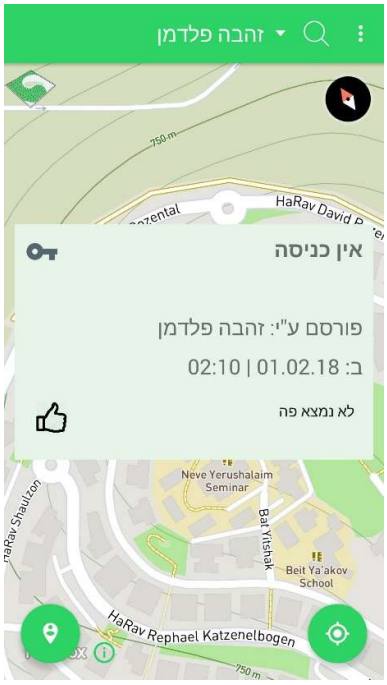
לפיכך, בקבצי ה- layout המגדירים את העיצוב של המסמכים, בכל מקום שנעשה שימוש במאפיין המשתמש בכיוונים כגון android:paddingLeft הוספנו מאפיין המשתמש במונחים של התחלה או סוף - android: paddingStart.



שינוי שפת האפליקציה
מאנגלית לעברית

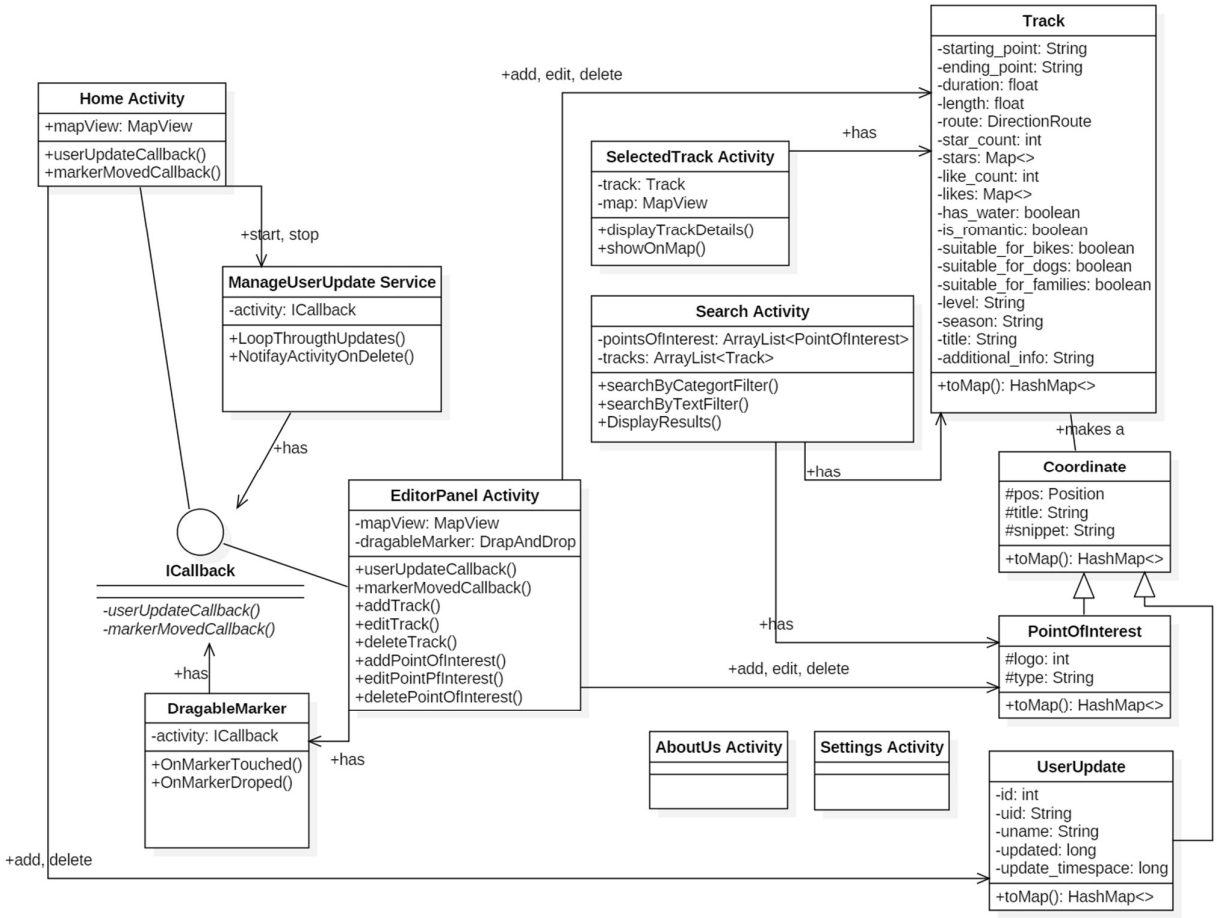


שפת האפליקציה
הוחלפה והנתונים מוצגים
בהתאם



תרשים מחלקות כללי :

נציג תרשים המתאר את מחלקות המערכת והקשרים ביניהם-



תיאור הכלים המשמשים לפתרון :

בפרויקט נשתמש בכלים הבאים למימוש הפתרון:

Android Studio – סביבת העבודה המשמש לבניית אפליקציות בשפת ג'אווה, במהלך הפיתוח נשתמש גם בספריות הרבות של אנדרואיד.

Mapbox - ממשק מפות חיצוני, נשתמש במספר ב **Mapbox Map Api** להצגת ויצירת המסלולים באופן ויזואלי. וב **Mapbox Marker Clustering Api** להגבלת מספר נקודות עניין המוצגות על המפה בהתאם לרמת הקרבה של המפה.

Sublime2 – סביבת העבודה לכתיבת סקריפטים בשפת NodeJS.

Firebase BaaS – פלטפורמה שישמש אותנו למרבית הפעולות שיעשו בצד השרת.

נשתמש בשירותים הרבים ש Firebase מציע, ביניהם אותנטיקציה- כאשר נרצה להבדיל בין משתמש רגיל באפליקציה לבין משתמש בעל הרשאות עריכה.

השירות העיקרי של Firebase שישמש אותנו באפליקציה הוא מסד הנתונים. נבנה מסד נתונים בשיטת ה NoSQL שישב על שרת מרוחק.

שימוש במסד נתונים זה מוליד את הצורך להשתמש בממשק חיצוני נוסף לביצוע חיפושים וסינונים של התוכן המוצג באפליקציה. שכן, נכון להיום Firebase לא מאפשר לנו להשתמש בשאילתות מורכבות בעת קריאת נתונים מהשרת.

Algolia SaaS- ממשק חיצוני לביצוע החיפושים וסינונים באפליקציה. ממשק זה יודע לרוץ על התוכן השמור אצלו בצורת עצי ג'ייסון ולהחזיר נתונים העונים לשאילתות מרוכבות.

כיוון שהמערכת שלנו תומכת בנתונים דינאמיים שעלולים להשתנות בכל רגע נתון הנתונים שניתנים לחיפוש גם הם דינאמיים וצריכים להיות מסונכרנים עם המסד נתונים של המערכת. ממילא, בנוסף לשימוש במשק החיצוני נכתוב סקריפט נוסף שיהיה אחראי לסנכרון בין התוכן הנמצא ב Firebase לתוכן הנמצא ב Algolia

תוכנית בדיקות

בסיום כל שלב בפיתוח המערכת נבצע מספר בדיקות שימולאו בהתאם בטבלאות:

1. בדיקות מערכת

- בדיקת ממשק המשתמש – האם הפקדים והרכיבים במסכי האפליקציה מגיבים ומספקים ממשק חוויתי. וכן האם המשתמש יכול להזין קלט לאפליקציה.
- בדיקות תקינות של מסד הנתונים- האם ניתן להכניס ולשלוף נתונים ממסד הנתונים

מספר בדיקה	מודל נבדק	תיאור הבדיקה	תוצאה צפויה	תוצאה בפועל
1	משגר יישומים - Application Launcher	הפעלת האפליקציה	האפליקציה עולה ומוצגים לוגו ושם האפליקציה	
2	מסך הבית / מסך יצירת מסלול / מסך מסלול	בדיקת המפה	המפה מגיבה לפעולות המשתמש- מזהה לחיצה על נקודה ומסלול. המפה מציגה מסלוליים נקודות עניין והתראות בקרבת המיקום הנוכחי של המשתמש	
3	מסך יצירת מסלול	בדיקת המפה	המשתמש מצליח לגרור נקודה ממקום למקום.	
4	מסך הבית / מסך מסלול	פקדי הדרוג בחלונית פרטי מסלול	הפקדים עוברים כנדרש כשהמשתמש בוחר בהם	
5	תפריט	בדיקת תפריט הנווט	התפריט מנווט בין מסכי האפליקציה	
6	מסך התראות	קלט המשתמש	המשתמש מצליח להזין תיאור אישי על הנעשה בשטח	
7	מסך חיפוש	קלט המשתמש	המשתמש מצליח להכניס קלט לחיפוש	
8	מסך שפות	פקדי שינוי שפה	האפליקציה מגיבה לבחירה של המשתמש ומשנה את שפת האפליקציה בהתאם	

2. בדיקות אינטגרציה

- בדיקות תקינות של מסד הנתונים- האם ניתן להכניס ולשלוף נתונים ממסד הנתונים
- בדיקה האם הנתונים המתקבלים ממסד הנתונים אכן תקינים ומעודכנים בזמן אמת
- בדיקה האם האפליקציה מתקשרת נכון אם השרתים
- בדיקה האם התוכן הנמצא במסד הנתונים מסונכרן עם התוכן הנמצא בממשק של Algolia
- בדיקת גבולות ועומסים על מסד הנתונים (מה קורה כשמסד הנתונים ריק או מלא)

מספר בדיקה	מודל נבדק	תיאור הבדיקה	תוצאה צפויה	תוצאה בפועל
1	התחברות לשרת מסד הנתונים	האפליקציה פונה לשרת מסד הנתונים דרך Firebase Api.	האפליקציה מצליחה להתחבר למסד הנתונים של Firebase	
2	שליפת נתונים	האפליקציה פונה למסד הנתונים על מנת לקבל מידע על הישויות (מסלולים, נקודות עניין והתראות)	האפליקציה מצליחה לקרוא נתונים ממסד הנתונים, והנתונים המתקבלים מעודכנים בזמן אמת	
3	הוספה / עדכון של נתונים	האפליקציה פונה למסד הנתונים על מנת להוסיף או לעדכן נתונים של מסלולים, נקודות עניין והתראות	האפליקציה מצליחה להוסיף ולעדכן נתונים במסד הנתונים גם במצב של עדכונים מרובים לרשומה אחת	
4	מחיקה של נתונים	האפליקציה פונה למסד הנתונים על מנת למחוק נתונים	האפליקציה מצליחה למחוק נתונים ממסד הנתונים	
5	סנכרון מידע	הסקריפט האחראי לסנכרון המידע מסנכרן בין השרתים	הנתונים הנמצאים בממשק של Algolia מסונכרנים כנדרש	
6	שליפת נתונים מסונים	האפליקציה פונה לממשק של Algolia על מנת לקבל נתונים ע"פ קריטריונים	האפליקציה מצליחה לקרוא נתונים מהמשק והנתונים המתקבלים תואמים לשאילתה שנשלחה	

3. בדיקות פונקציונאליות

- בדיקות שמוודות שהמערכת תואמת לדרישות הלקוח

מספר בדיקה	מודל נבדק	תיאור הבדיקה	תוצאה צפויה	תוצאה בפועל
1	מסך הבית /מסך יצירת מסלול	בדיקת מיקום במפה	המפה מציגה כברירת מחדל את המיקום הנוכחי או המיקום האחרון שבו המשתמש היה	
2	מסך הבית /מסך יצירת מסלול	בדיקת המפה	המפה מציגה מספר מוגבל של מסלולים ונקודות עניין בהתאם לרמת הזום של המפה	
3	מסך הבית /מסך יצרת מסלול /מסך מסלול	בדיקת המסלולים המוצגים	מסלולים של עיר יער צבוע בצע אחד ומסלולים של קק"ל צבועים בצבע נוסף	
4	מסך הבית /מסך יצירת מסלול /מסך מסלול	קבלת חוות דעתם של משתמשים אחרים	משתמשים יכולים לשתף משתמשים אחרים בחוות דעתם האישית על מסלולים	
5	מסך חיפוש	בדיקת הנתונים המוצגים	הנתונים המוצגים מתואמים לטקסט שהמשתמש הזין ולקטגוריות שבחר.	

4. בדיקות תאימות

- בדיקה שמוודא שהאפליקציה מוצגת כרגיל במכשירים שונים בעלי גרסאות אנדרואיד שונים, ורזולוציות שונות
- בדיקה שמוודא שהאפליקציה תומכת בריבוי שפות

מספר בדיקה	מודל נבדק	תיאור הבדיקה	תוצאה צפויה	תוצאה בפועל
1	מכשיר אנדרואיד שונים	בדיקה האם האפליקציה מגיבה כנדרש עבור גרסאות ורזולוציות שונות של מכשירים	פונקציונליות המערכת לא משתנה והמסכים מוצגים בהתאם למכשיר.	
2	שפות שונות	בדיקה האם האפליקציה תומכת בריבוי שפות	האפליקציה מזהה את השפה המוגדרת במכשיר או לפי בחירת המשתמש. המסכים מותאמים לשני כיווני טקסט כראוי.	

עבודות דומות בספרות והשוואה

- Google trips – אפליקציה של גוגל לתכנון טיולים. באפליקציה זו המשתמש בוחר אזור, ובהתבסס על המידע שיש במפות של גוגל מוצעים למשתמש שלל אטרקציות ובילויים באזור.

- Trailze –אפליקציית ניווט שטח המציע מגוון גדול של מסלולים ואטרקציות לפי אזור. האפליקציה מציגה על גבי המפה את המסלול הנבחר ונק' עניין בקרבת מקום. המשתמש יכול לדווח על נק' עניין, התראות או סכנות חדשות.

באופן כללי ניתן לראות כי במגוון האפליקציות הקיימות-

א. אין כל התייחסות לתחבורה הציבורית. המסלולים יכולים להתחיל ולהסתיים בכל נק' על גבי המפה, והמשתמש צריך לדאוג בעצמו לדרכי הגעה.

ב. המנגנון עובד ע"פ בחירת עיר ובהתאם לבחירה מוצגים למשתמש אטרקציות ומקומות בילוי בקרבת מקום. אין אופציות מוצעות ע"ג המפה וכן הסינון הוא ע"פ קריטריונים מובנים בלי אפשרות של חיפוש ע"פ טקסט חופשי. מנגנון שמתאים יותר לאנשים בעלי ניסיון בטיולים.

וכאן בא הפרויקט שלנו- אפליקציה שתספק למשתמש איחוד של הקיים יחד עם היתרונות הללו.

נספחים

א. בבילוגרפיה:

1. **Android Developer**- The official website for Android documentation, I used this site to learn anything on known in developing my app.
   https://developer.android.com/index.html
2. **Mapbox Android Sdk**- The interface used for displaying the map with the routes and points on it. Used the site to learn how to integrate the maps and how to use the properties it includes with in it.
   https://www.mapbox.com/android-sdk/
3. **Google Maps Android Api** – Google maps includes many fetchers that Mapbox does not, I used Google maps developers guide lines to help me implement those fetchers for my app.
   https://developers.google.com/maps/documentation/android-api/
4. **Firebase Documentation** – This documentation guided me step by step to create the Database for the app. Save, update and delete date from it. It also leaded me to find different interfaces for filtering data in my app.
   https://firebase.google.com/docs/android/setup
5. **Stackoverflow** – The number one place for asking questions in coding.
   https://stackoverflow.com/
6. **Algolia Documentation** – This is the documentation I used for integrating full text search in my app. I used this as a work around for filtering data stored in Firebase's Realtime Database
   https://www.algolia.com/doc/paths/getting-started-with-algolia/

ב. תכנון הפרויקט:

17.11	פתיחת מאגר גיטהאב ודף וויקי לפרויקט. המאגר שלנו הוא מאגר קוד פתוח, יהיה אפשרי לצפות בהתקדמות פיתוח האפליקציה בל רגע נתון. למאגר הפרויקט נצרף גם את כל המסמכים של הפרויקט. את יומן הפרויקט, שמדבר על ההתקדמות ועוד ננהל ע"י TRELLO.
20.11	פגישה ראשונה עם הלקוח יונתן, מעבר על הפרויקט הקיים וכתיבת דרישות להמשך הפיתוח.
20-22.11	כתיבת הצעת הפרויקט והצגתו למנחה.
	פגישה עם המנחה מיכל, ישבנו יחד ודנו כיצד ניתן להעשיר את היקף הפרויקט. העלנו רעיונות חדשים מעניינים ומאתגרים שניתן להטמיע במערכת ומה הדרך הנכונה ביותר לנהל את תהליך הפיתוח של האפליקציה.
22-28.11	תיקונים בטופס ההצעה והצגתו בפני הלקוח יונתן.
30.11	תאריך אחרון להגשת ההצעה
	המערכת הקיימת משתמשת בממשקים חיצונים שאני לא מכירה. כדי שאוכל להתקדם פתחתי מדריכים רבים באינטרנט והתחלתי ללמוד את הממשקים ואת אופן המימוש שלהם באפליקציות מבוססות אנדרואיד.
	פתיחת הפרויקט בסביבת אנדרואיד סטודיו- הורדתי את הקוד של דוריאן, סטודנט שעסק בפיתוח שנה שעברה, למחשב שלי והתחלתי לבצע את התוספות והשינויים הרלוונטיים. במקביל הייתי צריכה להתחיל לעבוד גם על המסד נתונים. בהתחלה חשבתי לבקש מדוריאן שיצרף אותי למאגר של המסד נתונים הקיים, אך לאחר התייעצות עם יונתן החלטנו שכדאי לפתוח מסד נתונים חדש ולשמור את הישן כ back up, ולכן קשרתי את הפרויקט החדש ל FIREBASE ויצרתי מסד נתונים חדש.
	עבודה על גרסת אלפא של הפרויקט
	במהלך העבודה ראיתי כי כמה מהדרישות של יונתן- כגון הוספת יכולת גרירה של נקודה על גבי המפה, או זיהוי לחיצה על מסלול במפה, אינם ממומשים ע"י ממשק המפות Mapbox כעת והם עובדים על כך. לאחר שראיתי זאת חשבתי לעדכן את יונתן כי אין באפשרותי למלא דרישות אלו. אלא שאז ראיתי כי הממשק של גוגל כן מאפשר זאת. לקחתי הפוגה מהפיתוח והתחלתי לקרוא על הממשק מפות של גוגל והאופן שבו הם ממשים את הנק' הנ"ל ועוד. לאחר שלמדתי לעומק את הממשק של גוגל חזרתי לפרויקט שלי והתחלתי לממש את הדברים גם במפה של Mapbox. עדכנתי את יונתן על ההתקדמות המשמעותית.
	פגישה נוספת עם מיכל. עברנו יחד על האפליקציה שבו ניתן לראות את הפרות לכל העבודה שנעשתה עד כה. דברנו על

	התכונה החדשה – עדכונים בזמן אמת, מה הדרך היעילה ביותר לנהל את מערכת העדכונים- מתי להתחיל ולהפסיק את ה service שאחראי על הקריאות שרת הרלוונטיות. דנו גם מתי משתמש יוכל לאשר או לחילופין להכחיש את קיומו של התראה מסוימת, מה נעשה במצב שבנק' מיקום מסוימת יש כמה התראות וכו'
	כיון ש Firebase לא מציעה שליפת נתונים באמצעות שאילתות מורכבות התחלתי לחפש דרכים חילופיות. אחרי בדיקה של האופציות שעומדות בפני הבנתי שיהיה מורך בהוספת קוד שרת בשפת NodeJs ועלכן נפגשתי עם אורה (חברת צוות בעבודה) כדי לראות אם זה אפשרי ולתכנן את המשך העבודה בהתאם

ג. טבלת סיכונים:

#	הסיכון	חומרה	מענה אפשרי
1	עבודה על קוד קיים לא מוכר הדורש שינויים	בינונית	מעבר על הקוד טרם תחילת הפיתוח. יצירת קשר עם כותב הקוד במקרה הצורך
2	חוסר התאמה בין דרישות הלקוח לתוצאות בפועל	גבוהה	עבודה מול הלקוח באופן שותף
3	אי עמידה בלוחות זמנים	נמוכה	תכנון לוח זמנים הגיוני ומעשי יחד עם חלוקת המשימות בצורה ברורה כדי למנוע בלאגן מיותר- שימוש ב github issues
4	העדר ניסיון בעבודה מול ממשקים המוטמעים במערכת	בינונית	קריאת מאמרים ולמידה ממדריכים באינטרנט

ד. טבלת דרישות:

מס' דרישה	תיאור
1	שינוי כללי בכל ממשק המשתמש הקיים- יצירת אפליקציה נוחה יותר למשתמש. עם עליית האפליקציה יוצגו למשתמש מספר מוגבל של מסלולים ונקודות ענין. לחיצה על מסלול או נקודת עניין מסוים יציג למשתמש בתחתית המסך פרטים מינימליים ולחיצה על הפרטים יעביר למסך מפורט יותר. כאשר המשתמש הוא בעל הרשאות עריכה במצב עריכה יוספו לו אפשרויות הוספה, עריכה ומחיקה למסלול או נק' ענין אך המעבר בין המסכים הוא כנ"ל.
2	מימוש חיפוש מסלולים ונק' עניין. החיפוש יתאפשר ע"י הקלדה חופשית או לחיצה על אייקונים מוצעים כגון מסלול הליכה/ מסלול אופנים/ מסעדות וכו'. המערכת כרגע מציעה למשתמש אפשרויות חיפוש ,אך למעשה לא ניתן לחפש. יש לשנות את אופציות החיפוש הקיימות ולממש את האופציות החדשות.
3	הפיכת האפליקציה לאפליקציה חברתית- המשתמש יוכל להוסיף חוות דעת על מסלול, לדרג מסלול ולשלוח התראות מסוגים שונים בעת שהותו במסלול מסוים.
4	הצגת כל המסלולים על המפה כולל מסלולים רשומים, מסלולים מסומנים

	ומסלולים המוצעים של האפליקציה.
5	הוספת מאפיינים חדשים לישויות הקיימות: 1. נקודות עניין- האם הנק' הוא מסוג אתר או עסק (כאשר בהמשך הוספה של נק' מסוג עסק תדרוש תשלום) 2. מסלול- דרוג (1 עד 5 כוכבים)
6	הוספת אפשרות של ביטול בשעת יצירה של ישות בלי הצורך בשמירה ואז מחיקה.
7	הוספת יכולת שינוי מיקום של נקודה על המפה על ידי גרירה.
8	המערכת תתמוך בשפות מרובות
9	המערכת תזכור את המקום האחרון בו המטייל טייל ובמקרה שהגישה למיקום באפליקציה לא דלוקה המפה תפתח במקום זה.

ה. צילומי מסד הנתונים:

Track – מסלול, מכיל שני סוגי נתונים:

1. מאפיינים של המסלול כגון עונות שמתאימות לטיול- season, רמת קושי- level וכו'
2. אובייקט Route הכולל את כל מרכיבי המסלול- נקודת, זמן, מרחק וכו'


```
tracks
  -L467fpVjnlerYPEXNry
  -L4DJzR3qTO6fejAur0q
    additional_info: ""
    db_key: "-L4DJzR3qTO6fejAur0
    duration: 1.61
    ending_point: "החלוץ"
    ending_point_json_latlng: "{\"altitude\":0.0,\"latitude\":31.788876342024125,\
    has_water: true
    is_romantic: false
    length: 8
    level: "Easy"
    like_count: 0
    route: "{\"distance\":8002.799999999999,\"duration\":5797.
    season: "All Year
    star_count: 0
    starting_point: "יפה נוף"
    starting_point_json_latlng: "{\"altitude\":0.0,\"latitude\":31.783419782344694,\
```

Coordinate – נקודות על המפה שהמשתמש הוסיף, כל נקודה כזו מאופיינת ע"י -position מיקום, שם ותיאור קצר.


```
coordinates
- 351757712
    - position: "{\"altitude\":NaN,\"latitude\":35.17577123417175,\"
    - snippet: ""
    - title: ""
+ 351760602
- 351773549
    - position: "{\"altitude\":NaN,\"latitude\":35.17735492078941,\"
    - snippet: ""
    - title: ""
+ 351779073
+ 351804067
```


User_updates -ה Reference החדש שהוספנו. הוא הרחבה של Coordinate.

מלבד התכונות שיורש מ Coordinate יש לו תכנות נוספות המאפיינות אותו-

סוג ההתראה

לוגו- המשמש להצגה הפיזית על המפה.

שעת יצירה ושם ומזהה של היוצר, (במקרה שהמשתמש היה מחובר למערכת בזמן ששלח את ההתראה)

update_time_space תוקף ההתראה מזמן היצירה, משמש בשביל למחוק נקודות שזמנם פג, כפי שהוסבר בתיאור הפתרון.


```
user_updates
- 351754716
    - logo: 213083768
    - position: "{\"altitude\":NaN,\"latitude\":35.1754716,\"longitu
    - snippet: ""
    - title: "מזג אויר"
    - type: "מזג אויר"
    - uid: "TNHu1lOD4vfz9SY8EEf4bsiQQu
    - uname: "זהבה פלדמן"
    - update_time_space: 90000
    - updated: 151747622
+ 351754717
+ 351754718
```

Point_of_interset גם הוא הרחבה של Coordinate

התכנות הנוספות המאפיינות אותו הם-

סוג הנקודה ולוגו- משמש להצגה הפיזית על המפה

points_of_interest
- 351827283
  - logo: 213090304
  - position: "{\"altitude\":NaN,\"latitude\":35.182728307294155,\
  - snippet: "74 ,33 ,52 ,55 ,ווים
  - title: "כפר שאול"
  - type: "תחנת אוטובוס"
- 351833857
  - logo: 213090304
  - position: "{\"altitude\":NaN,\"latitude\":35.18338577237125,\"
  - snippet: "39 ,21 ,35 ,33 ,ווים
  - title: "יפה נוף"
  - type: "תחנת אוטובוס"
- 351939481
  - logo: 213090305
  - position: "{\"altitude\":NaN,\"latitude\":35.193948158543435,\
  - snippet: ""
  - title: "גשר המיתרים "