Meni Samet, Moshe Chen & Tahel Citron
Department of Computer Science
Ariel University

Supervisor: Dr. Noam Hazon.

TOTAKE

the best trip list builder

תוכן עניינים

תקציר

פרויקט זה עוסק באפליקציית ToTake.
ToTake זוהי אפליקציה חכמה שעוזרת למשתמשים בה להרכיב רשימת ציוד לפני טיול בארץ או בחו"ל.
הרעיון הוא שכל אחד יכול ליצור רשימות במהירות ובקלות על ידי הכנסת יעד ותאריכי הנסיעה. והאפליקציה תציע לו בהתאם לכך את הפריטים הרלוונטיים על ידי מכונת למידה.
על מנת ללמוד השתמשנו במכונת הלמידה מסוג מערכת המלצות.
עבור כל רשימה רצינו לבדוק את רמת הדמיון לרשימות אחרות ולהמליץ למשתמש פריטים חסרים מן הרשימות הדומות ביותר.
המערכת תופעל על רשימות ראשוניות אשר אספנו על ידי אתר שבנינו (על מנת שהרשימות יהיו אמינות) ועל רשימות של המשתמשים באפליקציה.

הערה: יש לנו המון קוד במערכת שלנו, השתדלנו לא להכניס אותו כאן לספר כדי לא להעמיס, ניתן לצפות בכל הקוד, המודלים, הדיאגרמות וכו' שהשתמשנו בהם במהלך הפרויקט ב GitHub שלנו.
https://github.com/meni432/ToTake

תיאור הפרויקט

נושא הפרויקט

הפרויקט עוסק באפליקציה של המלצת פריטים לנסיעה בארץ או בחו"ל.
הרעיון המרכזי הוא שכל אדם יוכל להיכנס אלינו לאפליקציה, לרשום לאן הוא נוסע ואנחנו נעזור לו לבנות רשימת פריטים לאריזה במהירות וביעילות על בסיס אלגוריתמים מיוחדים.
לכן, החלטנו להתעסק בנושאי מכונות למידה מסוג מערכות המלצה, ובמערכות למידה אחרות כדי להמליץ על פריטים ולשער את מספר הפריטים.
לשם כך בנינו אפליקציית אנדרואיד ושרת שישמש את האפליקציה.

מצב קיים

- כאשר אדם רוצה לבנות רשימה הוא מתבסס על רשימות קודמות מיעדים שונים או מהמלצות של חברים שהיו באותו מקום.
- ישנן רשימות מוכנות באינטרנט אך הן לא מותאמות אישית, הן אינן בנויות על המלצות לפי הפרמטרים שלך אלה על המלצות כלליות יותר.
- המערכות שיש כיום הן אינן לומדות תוך כדי הכנסת פרטי המשתמש. אצלנו בשונה משאר מערכות ההמלצות התבצעו מתחילת בניית הרשימה ועד סופה.

מטרות הפרויקט

- לייצר את ההמלצות הטובות והמתאימות ביותר למשתמש על ידי מכונת למידה מסוג מערכת המלצות (Recommendation system) שעליה תתבסס האפליקציה ומאגר הרשימות שלה.
- שכל אדם יוכל להרכיב רשימת ציוד מותאמת אישית אליו עם כמה שפחות מאמץ מצידו להקליד את הפריטים שאותם הוא רוצה לקחת.
- תוך כדי בניית הרשימה, האפליקציה בעזרת מערכת ההמלצות תדע להציע לו פריטים נוספים על בסיס פריטים שהוא כבר בחר.
- האפליקציה תדע להמליץ על דברים נוספים המתאימים ליעד, כמו חיסונים מוקדמים או ביטוח טיסות מתאים.

תיאור המערכת

כל משתמש Android יכול להוריד את האפליקציה ולהתחבר דרך חשבון ה- Facebook שלו. אימות המשתמש מול ה-FireBase. ולאחר מכן מול שרת ה Restfull API שלנו.
המשתמש מכניס פרמטרים על הטיול והאפליקציה ניגשת עם פרמטרים אלו לשרת Restfull API.
השרת מתקשר עם המסד נתונים SQL, ומוסיף את הרשימה החדשה בנוסף מתקשר עם מכונת הלמידה בשביל ההמלצות למשתמש עבור הרשימה החדשה. מכונת הלמידה מסווגת את המשתמש ושולחת חזרה לשרת את ההמלצות עבורו.
מערכת הלמידה רצה על המערכת של Prediction IO, כל הטרנזקציות מולה מבוצעות על ידי שרת האפליקציה שלנו, עבור בקשה להצעות נשלח Query ועבור בחירה של פריט על ידי משתמש מסוים מוזן Event.
ההמלצות התבצעו תוך כדי בניית הרשימה הראשונית, לפי הפרטים הראשוניים אותם הזין המשתמש ואף על כל פריט שהמשתמש יוסיף רשימת ההמלצות תתעדכן בהתאם.
בכל טרנזקציה השרת מבקש שוב את ההמלצות המתאימות לטיול, ושולח המלצות חדשות בהתאם.
בנוסף, באפליקציה יש שימוש נרחב ב Google Place API בכדי לאפשר למשתמש למצוא בקלות את היעד אליו הוא נוסע, ובכדי להראות בצורה יפה יותר את רשימת הפריטים על ידי הצגת תמונה מתאימה ליעד.

רכיבי הפרויקט

אתר איסוף מידע

הקמנו את האתר כדי לאסוף נתונים מחברים, בכדי שבשלב בניית האפליקציה ומערכות הלמידה שלנו כבר יהיה לנו Data Set שעליו נוכל ללמוד ולבחון את האלגוריתמים הראשונים.
בבניית האתר ניסינו לבנות אותו כמה שיותר נוח לשימוש, אך כמה שיותר גם להימנע מהטעיית תוצאות השאלון.
לבסוף הצלחנו להשיג כ 300 רשימות טיול שונות (ואמינות!) על איזה פריטים אנשים לקחו לטיול שלהם, בעזרת האתר.

בנינו את האתר והשרת כשני רכיבים נפרדים, למרות שבסופו של דבר היו נמצאים על אותו שרת מסיבות Cross Origion שנתקלנו בהם במחשבים בארגונים, וגם מעלויות כלכליות. בסופו של יום, הפצנו את אתר ה Client גם על גבי Express והשתמשנו במנוע הרצות PM2 - עבור הרצה ב Production של Node JS.

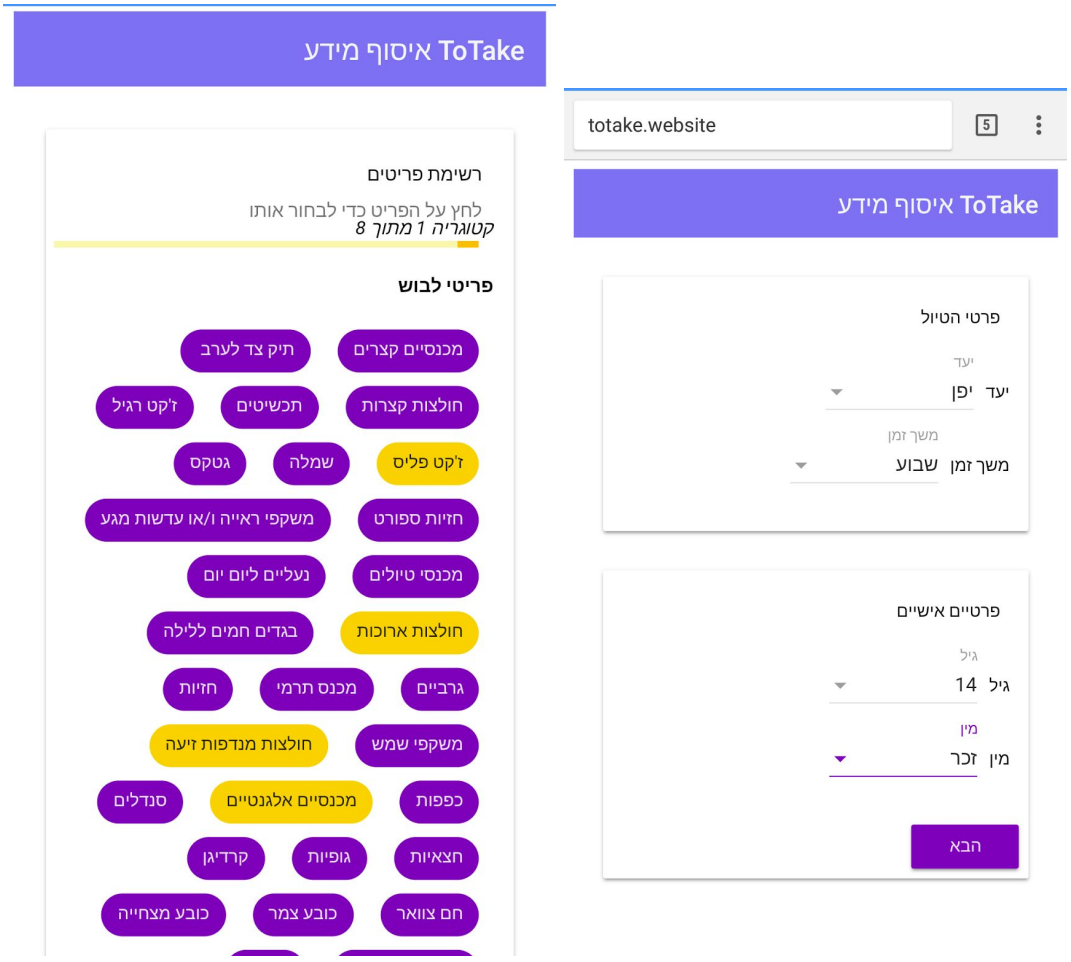
התקשורת בין שני הרכיבים התבצעה על ידי Json, וגם כאן כל התקשורת מול מסד הנתונים התבצעה בעזרת השרת שבנינו לטיפול בשאילתות מצד הלקוח.

מסד הנתונים ברכיב זה הוא מסוג Mysql גם כן, והוא הכיל בסופו של דבר את רשימת היעדים, רשימת הפריטים, את רשימות הטיולים של העונים באתר ופרטי הטיול שלהם.

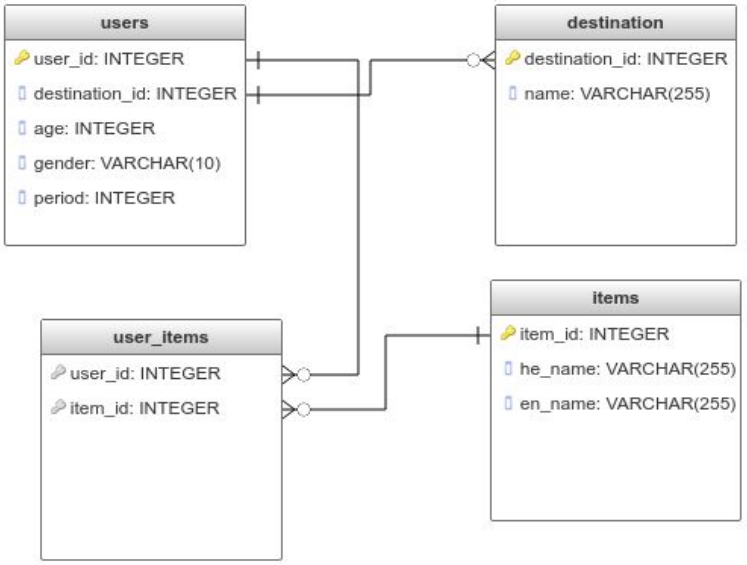
החלטנו לשמור את רשימת היעדים והפריטים גם ב DB בכדי לאפשר שינוי תוך כדי תנועה של רשימת הפריטים (במידת הצורך, בסופו של דבר לא השתמשנו, גם בכדי להימנע מהטעיות של המדגם) וגם כדי לאפשר מעבר נוח יותר לשלבים הבאים של הפרויקט.

מבנה המערכת (אתר איסוף)

- שימוש ב Angular 2 כדי להקים אפליקציה מבוסס Web לאיסוף הנתונים הראשוניים (http://www.totake.website)
- שימוש ב Express & NodeJS כדי להקים שרת אשר יתקשר עם האפליקציה
- הקמת מסד נתונים בMYSQL לשימוש השרת והאפליקציה.



מבנה מסד הנתונים (אתר איסוף)



- מבנה מסד נתונים זה "רזה" יותר ממסד הנתונים הסופי שיש במערכת, לאחר בנייתו בנינו Script המעביר את המידע בינו לבין מסד הנתונים החדש, המתאים למבנה המערכת הסופי.
- בעצם, השתמשנו במסד זה כדי לבנות את "ותכנית האימון" של המערכת שלנו בשלב הסופי על ידי כך שהשתמשנו במידע מכאן כדי ליצור משתמשים ווירטואליים במערכת הסופית (הוספה ל DB, הוספה למערכת ההמלצה, מתן המלצות וכו'). בכדי שהמשתמשים הראשוניים שישתמשו באפליקציה יוכלו לקבל המלצות על בסיס המידע שנאסף באתר הדגימות.

מערכת המלצה

שימוש ב Prediction IO

Prediction IO זוהי סביבה המופצת בקוד פתוח ומאפשרת להפעיל בתוכה אלגוריתמי למידה שונים, לשלב בינם וכמו כן, מספקת interface לתקשורת עם שרת חיצוני לקליטת המידע עבור האלגוריתמים ממקורות חיצוניים ולשאלת שאילתות שונות.

שים לב - התייחסות מפורטת על פעולת האלגוריתמים במערכת זו נמצאת בחלק של האלגוריתמים בספר פרויקט זה.

Prediction IO זוהי מערכת המיועדת לשימוש Machine Lerning ומספקת שירותי building, evaluating, developing engine עבור מערכות למידה.
Event Server - זהו חלק ממערכת זו המאפשר לקבל מידע מפלטפורמות שונות עבור ה Engines השונים, כמו הוספת משתמש חדש למערכת, כמו כן, ניתן להשתמש בו בזמן ריצה, כך שהתאים לנו מאוד לצרכים שלנו.
Query Service - מאפשר לקבל תוצאות למידה און ליין מהמערכת, בקשות המתקבלות באמצעות REST מתקבלות בחזרה עם המלצות בהתאם לבקשה שנשלחה.

מבנה המערכת (PredictionIO - Service)

- הקמת שרת ייעודי ב Digital Ocean המריץ Ubuntu ועליו התקנת Prediction IO.
- שימוש במערכת של Prediction IO תוך כדי התאמה שלה לצרכים שלנו.
- עבודה מול המערכת כ Service מצד שרת המערכת שלנו.
- מסד נתונים נפרד, המאחסן את המידע שהמערכת זקוקה לו למתן המלצות (מנוהל על ידי המערכת מבוסס על BBB DB)

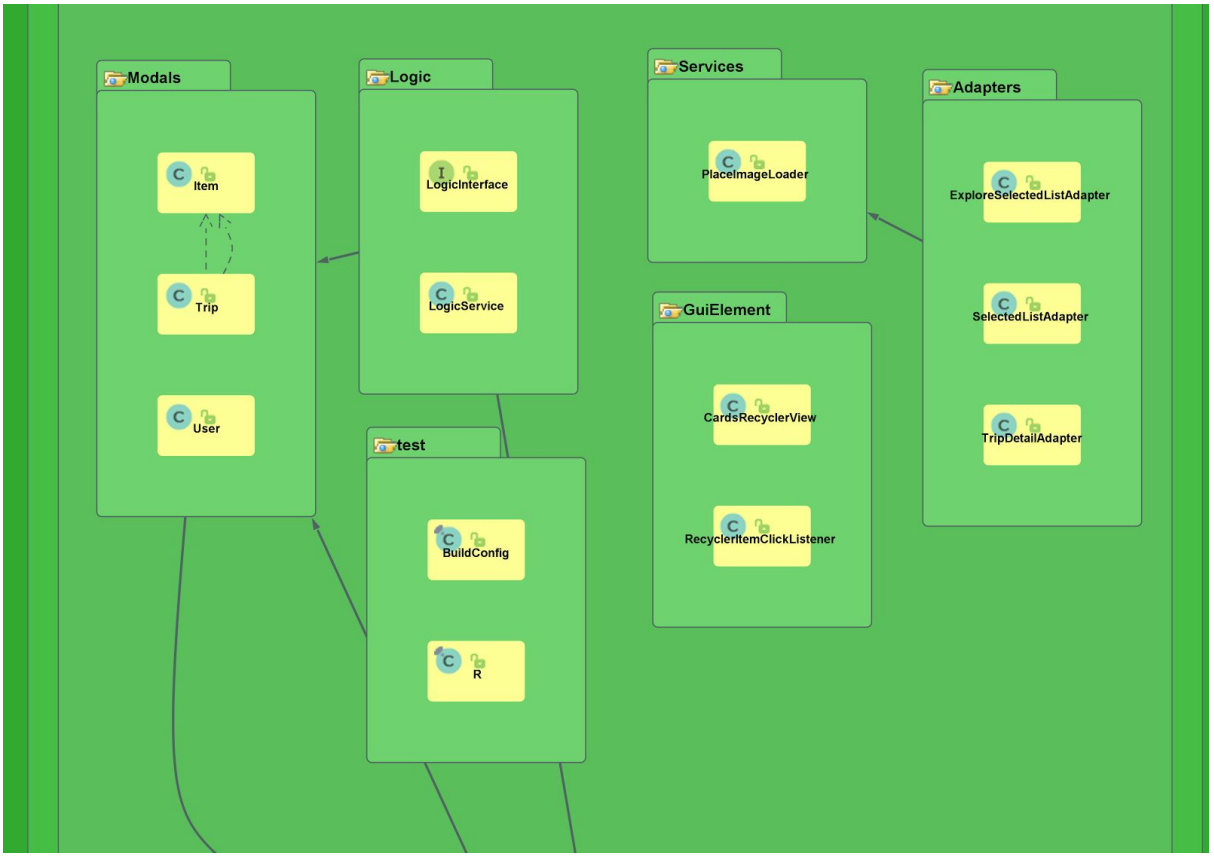
אפליקציה עבור Android

אפליקציית האנדרואיד מהווה את השער של המשתמשים אל המערכת שלנו, היא מאפשרת להם להתחבר באמצעות Facebook אל המערכת, ליצור טיול חדש להוסיף פריטים על בסיס המלצות שמתקבלות ועוד. בשלב בניית האפליקציה היה חשוב לנו לשמור על הפרדה מוחלטת של הרכיבים השונים (GUI, Server Communication, Logic Service) בכדי לאפשר בנייה מודולרית של האפליקציה, ולאפשר לה לתמוך בשינויים שנעשו במהלך הפיתוח.

- רכיבי GUI
  - שימוש ברכיבים מקובלים כגון RecycleView, Fregmants וכו'
  - כל רכיבי ה GUI מקבלים מידע מ Logic Service שהוא מהווה את השער של ה GUI למידע שיש באפליקציה ובשרת (בהתאם לצורך).
  - ביצענו שימוש ב Google Place API Serivce בכדי להכניס רכיבי GUI כמו תמונת יעד הטיסה וחיפוש יעדים נוח עם Place Autocomplete שמתבסס על שירותי המיקום של גוגל.
- רכיבי Logic
  - האפליקציה מבצעת המון תקשורת עם השרת בכדי לספק המלצות עדכניות תוך כדי בניית הרשימה על ידי המשתמש.
  - כדי לצמצם כמה שיותר את הפנייה לשרת, השתמשנו בשכבה זו כדי לשמור בצורה מקומית מידע שניתן לשמור. לדוגמה, רשימת כל הפריטים, או מידע על המשתמש.
  - רכיב זה מעדכן את ה GUI באמצעות Register Callback אודות עדכונים שמתקבלים מהשרת, הגעה של המלצות חדשות וכו'.
- רכיבי Server Communication
  - מהווה את השער של האפליקציה לשרת שכולל בתוכו את ההמלצות השונות, רשימות הפריטים, התחברות וכו'
  - שימוש ב volley כRest Client לצורך תקשורת עם שרת האפליקציה.
  - כל המידע מועבר על גבי HTTP בעזרת Json, שימוש ב Gson כדי להמיר את ה Json המתקבל מהשרת לאובייקטים מקובלים של Java.

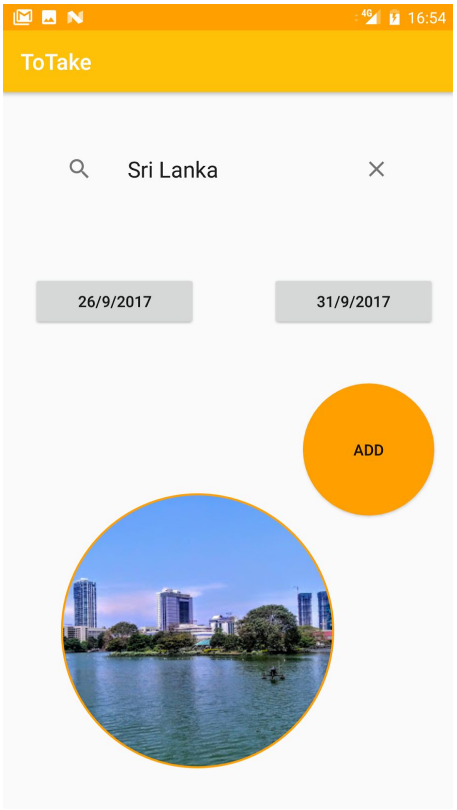
חבילות מרכזיות

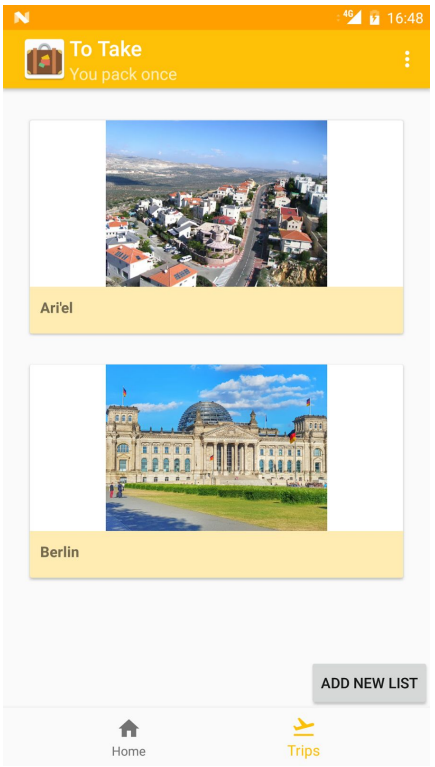
- Server
  - Listeners - חבילה זו מכילה Interfaces המגדירים את סוגי ה Listeners אותם המבקשים בקשות מהשרת צריכים לספק.
  - Interfaces - חבילה זו מכילה את ה Interfaces המשותפים עם השרת, המהווים בעצם Sheread files עם השרת לצורכי אינטגרציה קלה בין השרת לאפליקציה שלנו.
  - Services - מכיל בתוכו GsonRequest המהווה Factory Class עבור Volly.
- Logic
  - Logic Interface זהו Interface עבור רכיבי ה GUI למידע הקיים באפליקציה ובשרת.
- Modals - חבילה זו מכילה מחלקות המהווים את מבני הנתונים של המערכת.
  - מחלקת Item - מייצגת Item ומכילה את המידע עליו כגון שם, מספר Id וכו'
  - מחלקת Trip - מכילה בתוכה מידע על טיול נתון כולל רשימה של כל ה Items שבו.
  - מחלקת User - מכילה בתוכה את המידע על המשתמש, לצורכי כניסה למערכת וכו' וגם את ה Trips השונים של המשתמש.
- Adapters - חבילה זו מכילה בתוכה את ה Adapters המשמשים את ה GUI לתצוגה נוחה של המידע, בנינו מספר Adapters שונים עבור הצרכים שלנו, בניהם גם את ה CardRecycleView .



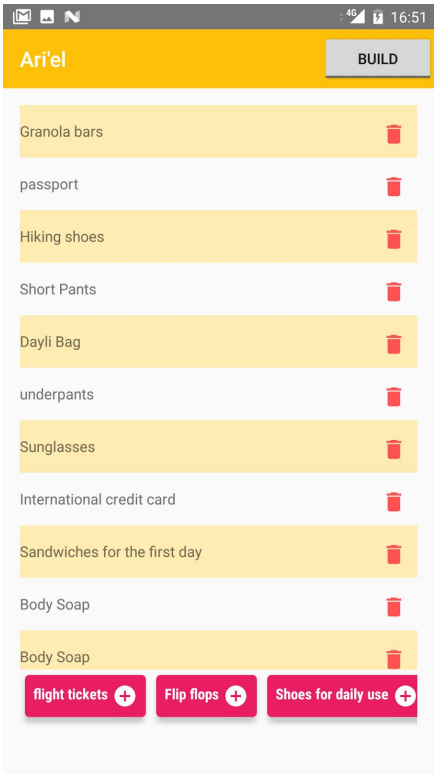
מבנה ה Activities

צילומי מסך

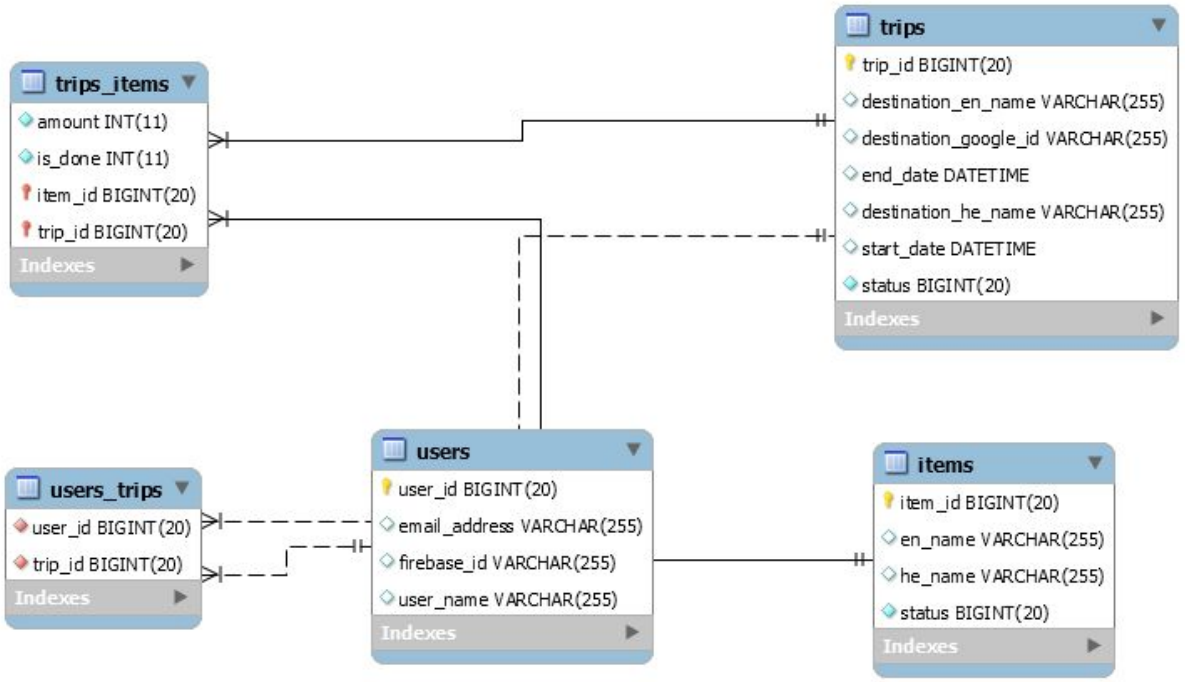

16:54
ToTake
Sri Lanka
26/9/2017
31/9/2017
ADD


16:48
To Take
You pack once
Ari'el
Berlin
ADD NEW LIST
Home
Trips


16:55
ToTake
Facebook Login
Signed Out
Continue with Facebook


16:51
Ari'el
BUILD
Granola bars
passport
Hiking shoes
Short Pants
Dayli Bag
underpants
Sunglasses
International credit card
Sandwiches for the first day
Body Soap
Body Soap
flight tickets
Flip flops
Shoes for daily use

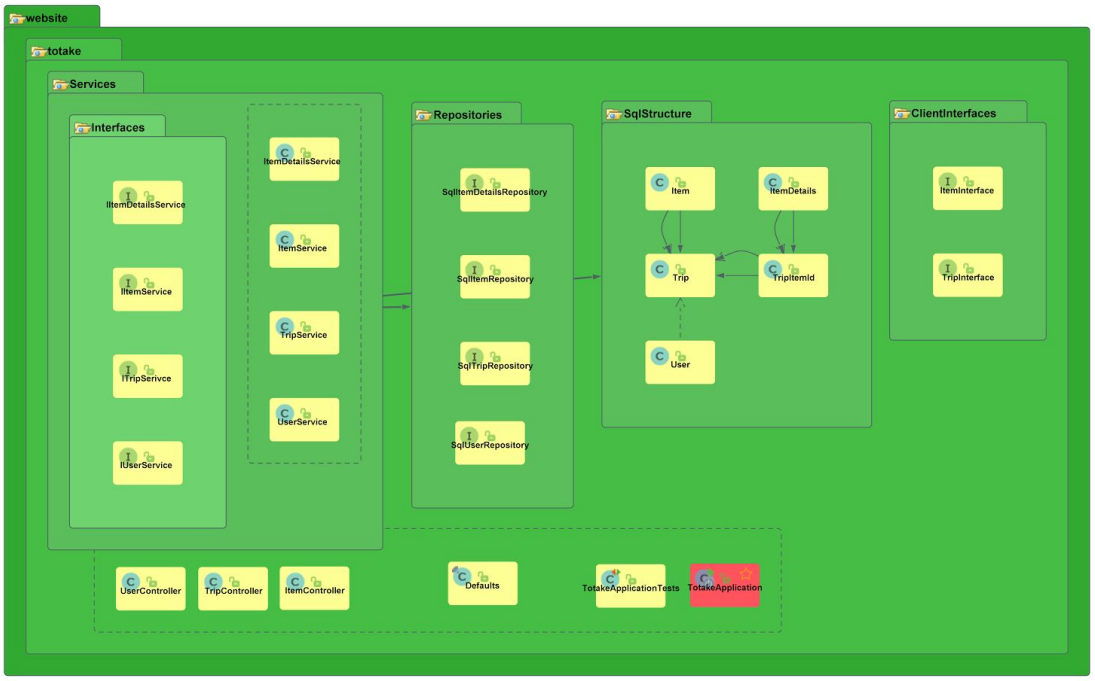
מסד נתונים



מסד נתונים זה נבנה באופן אוטומטי על ידי חבילת ה DB Manager שאנחנו משתמשים בשרת שלנו.
למעשה, בשרת שלנו לא כתובה שורת SQL אחת, וכל הגישה לנתונים מתבצעת דרך החבילה Hibernate ב Spring.

אך כמובן שתכננו את המבנה מראש, והתאמנו את מבנה החבילה לתכנון שלנו.

שרת אפליקציה



- שרת מבוסס Java ביחס עם Spring עבור תקשורת HTTP עם האפליקציה.
- מהווה את הקשר של האפליקציה עם מערכת ההמלצות וה DB.
- שימוש ב Jacson להמרת אובייקטי Java ל Json.
- שימוש ב JPA Hibernate לשימוש בMysql מבוסס אובייקטים בJava.
- בנייה ושימוש ב LRU Cache לצמצום הגישות ל Mysql (כולל טיפול בגישות Multithreads מכיוון שהשרת תומך בהם).
- רץ על גבי Tomcat Server ב Public Cloud המספק שירות זה.
- מבוסס על מבנה Spring סטנדרטי, הכולל שימוש ב Service ו Repositorys לגישה למידע הנמצא על השרת.
  דוגמה: ב Item Service אנחנו בודקים אם הItem נמצא ב LRU Cache ואם כבר נמצא, אנחנו מחזירים אותו במקום לגשת אל ה DB בעזרת Item Repository.

Controller, Service and Repository

- משפחת ה Controller מטפלת בבקשות הנכנסות לשרת, מפעילה תהליך כלשהו לביצוע הבקשה ומחזירה תשובה.

- משפחת ה Service היא הקשר בין ה Controller אל ה Repository , אנחנו הכנסנו Caching בשכבה זו, כך שאם משהו מבוקש כבר קיים ב LRU שלנו, אנחנו לא פונים אל ה DB באמצעות ה DB, כמובן שדבר זה שקוף מבחינת ה Controller והוא מקבל תשובה רגילה.
- משפחת ה Repository הצריכה מאיתנו לממש רק Interface מכיון ש Spring ממשת אותם בעצמה על בסיס שמות הפונקציות. באמצעות ה Repository של כל טבלה ניתן לשמור, לשנות ולהציג מידע על גבי בסיס הנתונים.

אצלנו בפרויקט יש Controller, Service and Repository עבור Trips, Items ו Users.

קטעי קוד נבחרים

- חלק מהמחלקה Trip אשר נמצאת בשרת ומהווה את ההגדרה עבור ה DB לטבלה Trip כולל היחסים בין טבלה זו לטבלאות אחרות (`@OneToMany`)

```
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
@Column(name = "trip_id")
private long tripId;
@Column(name = "destination_en_name")
private String destinationEnName;
@Column(name = "destination_he_name")
private String getDestinationHeName;
@Column(name = "destination_google_id")
private String destinationGoogleId;
@Column(name = "start_date")
private Date startDate;
@Column(name = "end_date")
private Date endDate;
@Column(name = "status", nullable = false)
private Long status;


@OneToMany(mappedBy = "primaryKey.trip",
       cascade = CascadeType.ALL, fetch = FetchType.EAGER)
```

- חלק מהמחלקה TripController המהווה את השער של בקשות הנוגעות ל Trips לשרת, במקרה הזה, מדובר בפונקציה המתאימה להוספת טיול חדש, עם המידע המתקבל על ידי ה HTTP Request מומר על ידי Spring לargs הפונקציה. בנוסף, בהחזרת האובייקט Trp הוא מומר לפורמט Json ומוחזר כ HTTP Response על הבקשה.

```
@RequestMapping("/addNewTrip")
public Trip addNewTrip(@RequestParam(name = "userId", defaultValue = "-1") long
userId,
                       @RequestParam(name = "destinationName", defaultValue =
"none") String destinationName,
                       @RequestParam(name = "startDate", defaultValue = "-1") long
startDate,
                       @RequestParam(name = "endDate", defaultValue = "-1") long
endDate,
                       @RequestParam(name = "googlePlaceId", defaultValue =
"none") String googlePlaceId) {
```

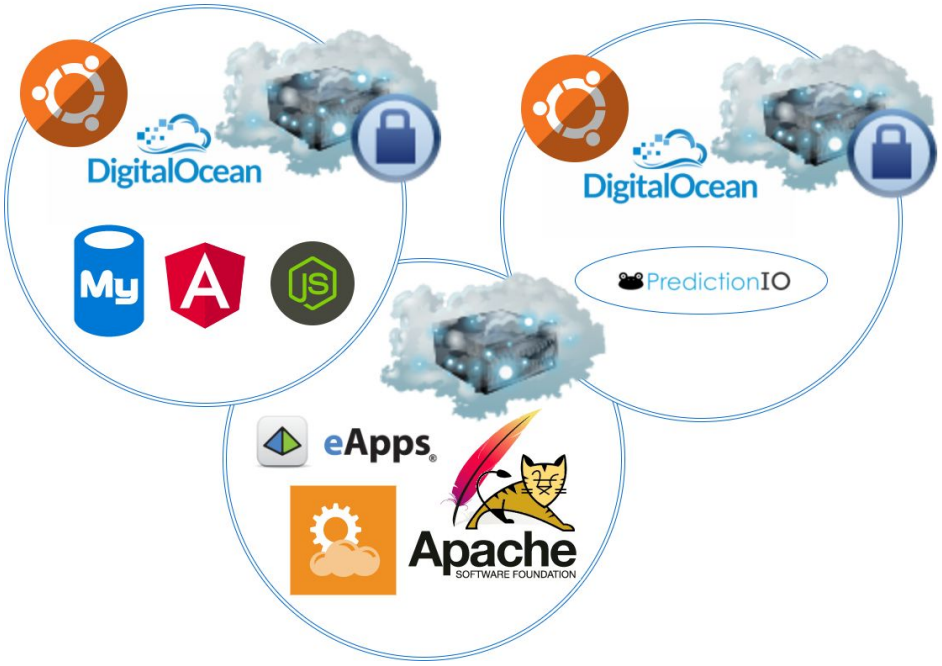
```
   User user = userService.getUser(userId);
   Trip trip = tripService.addNewTrip(destinationName, destinationName,
googlePlaceId, new Date(startDate), new Date(endDate));
   user.addTrip(trip);
   userService.save(user);

   return trip;
}
```

- דוגמה ל Repository , החלק המעניין הוא שאין צורך לממש את המחלקה, אלה היא ממומשת באופן אוטומטי על ידי Spring.
  Spring יודעת לממש את הפונקציה על בסיס השם שלה, כאן אנחנו רוצים לקבל ב Repository רשימת טיול על פי ID, לפי השם, Spring יודע לייצר את שאילתת ה Sql המתאימה.

```
public interface SqlTripRepository  extends CrudRepository<Trip, Long> {
   public Trip findTripByTripId(long tripId);
}
```

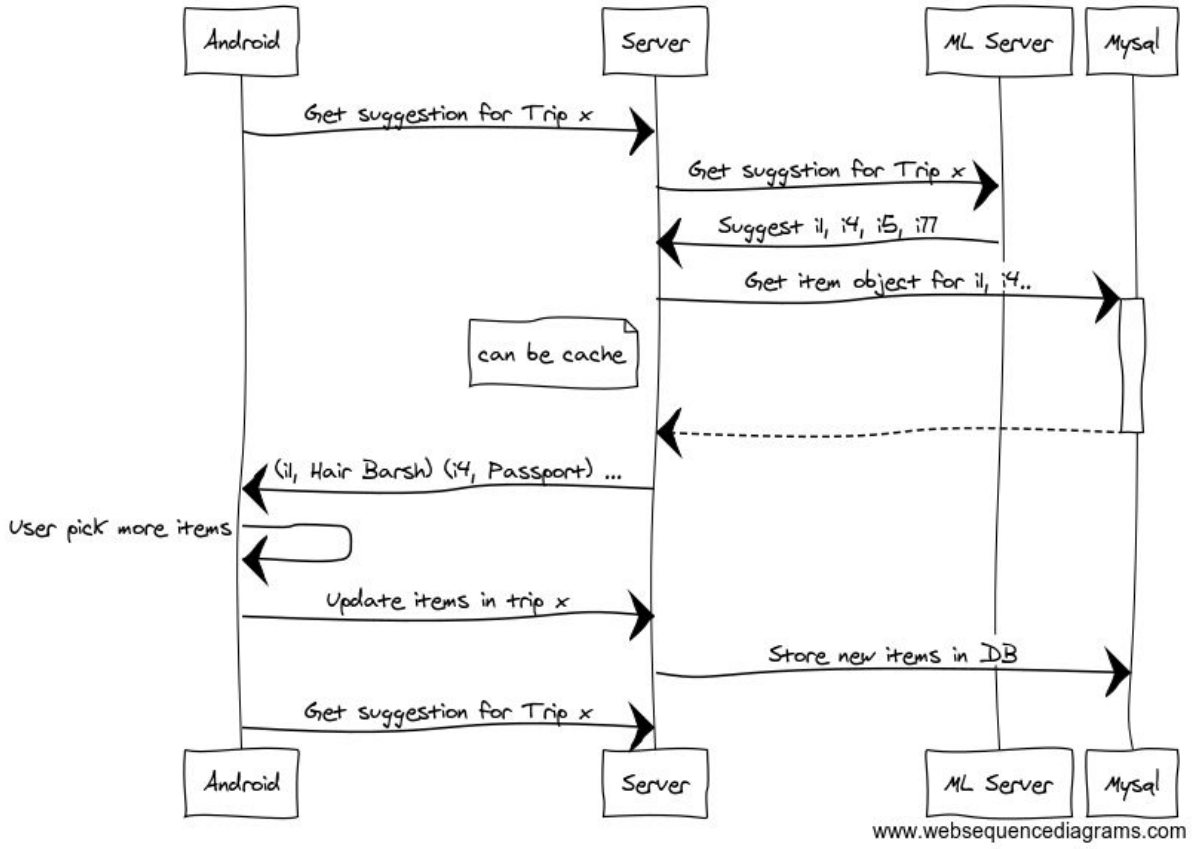
מבנה האחסון בשרתים



- במהלך התכנון של המערכת, ניסינו כמה שיותר להגיע למצב שכל רכיב במערכת פועל באופן אוטונומי, ללא קשר לאיפה הוא מורץ.
- רוב השיקולים בבחירת המבנה של השרתים שלנו היה כלכלי בעיקר, ניתן להפריד כל רכיב לשרת אחר או לשירות שרת פתוח אחר ללא כל בעיה.
- המבנה הנוכחי מורכב מ 3 שרתים שונים:
  - Digitalocean 1 - Ubuntu
    - Prediction IO - כולל מסד הנתונים PostgreSQL ורכיבי המערכת השונים.
  - Digitalocean 2 - Ubuntu
    - שרת ה MYSQL שלנו, כולל בתוכו גם את הטבלאות עבור אתר האיסוף וגם את הטבלה עבור שרת האפליקציה.
    - PM2 - מנוע הרצה של NodeJS המריץ את אתר איסוף המידע (צד לקוח ושרת)
  - eApps - Public Cloud
    - שרת האפליקציה שלנו המבוסס על ה Framework של Spring ורץ על גבי Tomcat Server באמצעות הממשק של Jelastic.

Use Cases

קבלת המלצות לאפליקציה עבור טיול נתון



אלגוריתמים

בעיה מרכזית

החלק המרכזי בפרויקט שלנו הוא Machine Learning: Recommender Systems.
הבעיה החישובית המרכזית שהפרויקט מתעסק עמה היא המלצה יעילה לפריטים לאריזה על בסיס מידע שנאסף מכל המשתמשים במערכת.
בוודאי שאם נבצע חיפוש שלם נגיע לסיבוכיות גבוהה במיוחד ולכן אנו משתמשים באלגוריתם של מערכת המלצה עבור בעיה זו.
לאחר בדיקה מעמיקה של מערכות המלצה שונות, אותן ניסינו על מסד הנתונים שלנו בכדאי למצוא את האלגוריתם הטוב לנו ביותר שימליץ רשימות אמינות ומתאימות, החלטנו להשתמש ב PredictionIO.

Slope One Algorithem

User	Item 1	Item 2	Item 3
Tahel	Took It	Did not take it	Took It
Meni	Did not take it	Took It	Took It
Moshe	Did not take it	Took It	Did not take it

In this case, the cosine between items 1 and 2 is:

$$\frac{(1,0,0)\cdot(0,1,1)}{\|(1,0,0)\|\|(0,1,1)\|}=0,$$

The cosine between items 1 and 3 is:

$$\frac{(1,0,0)\cdot(1,1,0)}{\|(1,0,0)\|\|(1,1,0)\|}=\frac{1}{\sqrt{2}},$$

Whereas the cosine between items 2 and 3 is:

$$\frac{(0,1,1)\cdot(1,1,0)}{\|(0,1,1)\|\|(1,1,0)\|}=\frac{1}{2}.$$

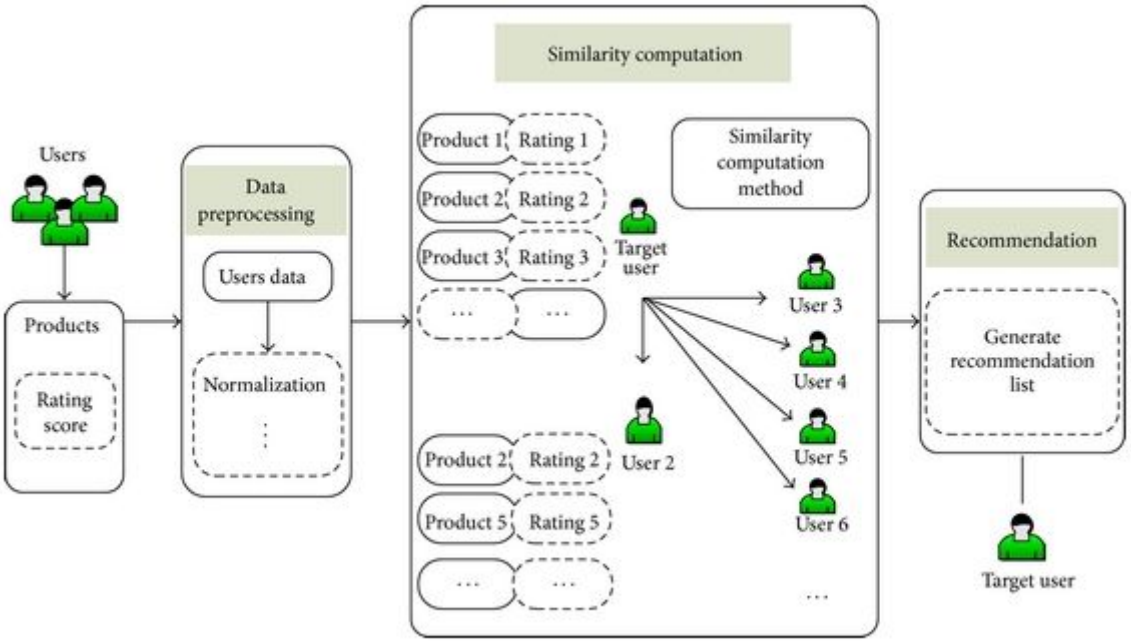
תיאור האלגוריתם

אלגוריתם זה הוא ממשפחת אלגוריתמים הנקראים Collaborative filtering, סינון שיתופי.
סינון שיתופי הוא תהליך של סינון מידע על בסיס ההנחה כי אם שני אנשים בעלי העדפות דומות עבור נושאים מסוימים (במקרה שלנו, עבור אותם פריטים) אז יש סיכוי שהם יהיו בעלי העדפות דומות גם עבור נושאים אחרים.

לאלגוריתם זה יש שיטות שימוש שונות אך המרכזית שבהם היא:

1. מצא משתמשים שבחרו פריטים דומים לפריטים שהמשתמש שאנו רוצים לייצר עבורו חיזוי בחר.
2. נשתמש בפריטים שאותם בחרו המשתמשים שמצאנו בסעיף 1, כדי לייצר חיזוי עבור המשתמש הנוכחי.

עבור משתמש 'u' נרצה לחשב את דירוג ההתאמה של הפריט 'i' עבורו:



הייחוד של Slope One הוא שהאלגוריתם מבוסס על כך שאם משתמש מסוים בחר פריט X הוא כנראה ירצה לבחור גם את פריט Y, ללא קשר להתאמה המלאה בין המשתמשים (ישנה אפשרו לעשות את האלגוריתם בצורה של user-user ולמצוא את ההתאמה לפי המשתמשים עצמם ולא לפי הפריטים, אנחנו רצינו שההמלצות יתבצעו לפי הפריטים הדומים של המשתמשים).

Pseudo-Code For Slope One

[1]*The reprocessing phase*

for every item i
 for every other item j
 for every user u expressing preference for both i and j
 add the difference in u's preference for i and j to an average

The prediction part

for every item i the user u expresses no preference for
 for every item j that user u expresses a preference for
 find the average preference difference between j and i
 add this diff to u's preference value for j
 add this to a running average
return the top items, ranked by these averages

[1] https://dzone.com/articles/slope-one-recommender

בעיות באלגוריתם

לאלגוריתמים מסוג Collaborative filtering ישנם מספר אתגרים למימוש כמו בעיית דלילות נתונים עבור משתמש חדש, שהמשתמש יצטרך לדרג (במקרה של האפליקציה שלנו לבחור) מספר גדול יחסית של פריטים לפני שיוכל לקבל המלצות טובות.
בנוסף בהתחלה לא ניתן לכל פריט דירוג זאת אומרת שלדוגמה שההסתברות שבחור יקבל המלצות מרשימה של בחורה שווה להסתברות שיקבל מרשימה של בחור, או ההסתברות שאדם שטס לאירופה יקבל המלצות מרשימת טיול למזרח שווה להסתברות לקבל המצות מרשימה עם מיקום מתאים יותר.

בדיקת האלגוריתם

לאחר מימוש האלגוריתם בשפת java באופן המתאים למערכת שלנו הרצנו אותו על הנתונים שאספנו בשלב הראשון של הפרויקט.
בהתחלה בדקנו על פריט בודד ונפוץ שהורדנו וציפינו לקבלו בהמלצות (לדוגמה: מברשת שיניים), הפריט התקבל ברשימת ההמלצות.
בבדקה נוספת שערכנו רצינו לקבל רשימה סופית המתאימה לטיול ספציפי. לקחנו רשימה מלאה רשימת ה"רצוי" אך ה"מצוי" היה לא טוב מספיק.

Slope One Algorithem with rating

כדי לפתור את בעיית הדירוג שהוזכרה למעלה הוספנו דירוג לאלגוריתם Slope One.

תיאור האלגוריתם

האלגוריתם לא שונה בהרבה מSlope One הרגיל.
השינוי העיקרי היה שההמלצות נעשו על ידי התחשבות בפרמטרים של המשתמש.
דירוג של רשימה אחרת של משתמש שנוסע לאותו אזור יהיה גבוה יותר מדירוג של פריטים של משתמש שנוסע לאזור אחר. כך שהאלגוריתם יופעל על הרשימה אז דירוג הפריט יהיה גבוה יותר.

- For item ***i***, find other similar items
- Estimate rating for item ***i*** based on ratings for similar items
- Can use same similarity metrics and prediction functions as in user-user model

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

s_{ij}... similarity of items ***i*** and ***j***
r_{xj}...rating of user ***x*** on item ***j***
N(i;x)... set items rated by ***x*** similar to ***i***

הפריטים שלא נבחרו יהיו מאופסים (כי אז הוא פשוט יתנהג כמו האלגוריתם ללא הדירוג ויתחשב רק בכמות הפריטים).

החלוקה בסכום נעשית כדי ליצור שוויון שבין הפריטים שנלקחו לאילו שלא. ממוצע של כל הפריטים שנלקחו (לאחר החלוקה) יהיה 0 וממוצע כל הפריטים שלא נלקחו גם יהיה 0 (טוב למציאת הcosine בין הפריטים השונים).

בעיות באלגוריתם

האלגוריתם עדיין מסוג Collaborative filtering אז הבעיות של אלגוריתם מסוג זה עדיין נשמרות (מפורט למעלה).

בדיקת האלגוריתם

לאחר מימוש האלגוריתם בשפת Java באופן המתאים למערכת שלנו, הרצנו אותו על הנתונים שאספנו בשלב הראשון של הפרויקט.
התוצאות יצאו דומות מאוד לתוצאות ללא הדירוג של הפריטים. תוצאות אלו לא היו טובות מספיק והיו הרבה Outliers שפגעו בביצועי המערכת.

PredictionIO - open source Machine Learning Server

תיאור האלגוריתם

זוהי סביבה המופצת בקוד פתוח ומאפשרת להפעיל בתוכה אלגוריתמי למידה שונים, לשלב בינם וכמו כן, מספקת interface לתקשורת עם שרת חיצוני לקליטת המידע עבור האלגוריתמים ממקורות חיצוניים ולשאלת שאילתות שונות.

אנחנו התבססנו במקרה שלנו על Tamplate בשם Similar Product לאחר שבחנו גם את האפשרות של ה The Universal Recommender אך בשל המחסור שלנו בקטגוריות (לפחות במצב הנוכחי של הפרויקט) ראינו ש Simmilar Product עם התאמות נכונות מספק תוצאה טובה יותר.

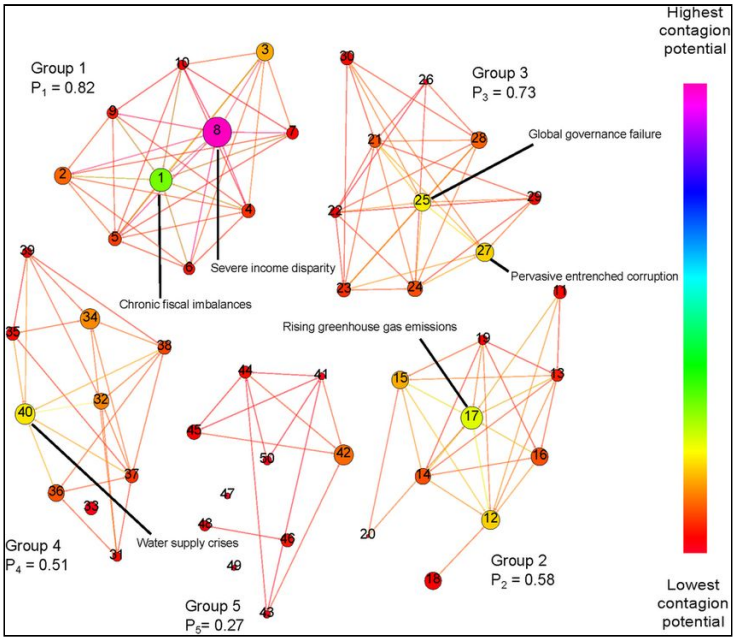
מכונת הלמידה שלנו מתבססת על שני אלגוריתמים:

1. Co-occurrence Algorithm
2. ALS Algorithm

שני האלגוריתמים ממומשים בשפת Scala. ובעזרת Prediction IO שעוטף אותם, אנחנו משתמשים בהם לצורך מימוש מערכת ההמלצה אצלנו בפרויקט.

Co-occurrence Algorithm

אלגוריתם התרחשות משותפת.
הוא מוצא מכנה משותף בין מילים/ מונחים/ מושגים ויודע לקשר בניהם.
התהליך נעשה בעזרת אלגוריתמים NLP (עיבוד שפה טבעית-אלגוריתם מבוסס למידה ממוחשבת לייצוג טקסט על ידי מבנה נתונים מספרי).
כך שאפשר לייצר ייצוג גרפי לכלל הפריטים לקשר בניהם ולהסיק מסקנות ביחס ליחסים של הפריטים, שזה בעצם המטרה שלנו - לקשר בין פריטים.



Alternating Least Square Algorithm

כאשר מדובר על המלצת פריטים למשתמשים, ישנו הקשר בין פריט לפריט אך גם צריך להתחשב בקשר בין משתמש למשתמש אחר.
מעבר על טבלאות אלו בכל המלצה וחישוב ההמלצה בכל פעם זו בעיה NP קשה (ככל שהמסד נתונים יותר גדול כך הבעיה גדולה יותר).
לשם כך צריך לייעל את המעבר על הנתונים.
אלגוריתם ALS בא לפתור את בעיית האופטימיזציה הזו.
באלגוריתם המקורי, הפונקציה ()train מחשבת את מספר הפעמים שהמשתמש צפה באותו פריט.
ב PredictionIO הם הוסיפו גם פרמטר בולאני, אם לקח או לא לקח את הפריט.

יתרונות בPredictionIO

- בשונה מהאלגוריתמים הקודמים המערכת הזו עובדת בזמן אמת.
- מבוססת על שני אלגוריתמים כך שמכסה יותר מקרים ולומדת יותר טוב.
- המעבר על הטבלה בSlope One לוקח זמן רב וב PredictionIO אלגוריתם ALS פותר את הבעיה הזו (בעיית אופטימיזציה).
- ALS משלב גם את הקשרים בין המשתמשים וגם את הקשרים בין הפריטים. בשונה מ Slope One שמסתמך בהמלצותיו רק על אחד מהם.
- ב Slope One הקשר בין הפריטים מבוסס רק על הcosine בניהם (כך שכמות הפריטים הדומים ודירוגם הם הפאקטור היחיד) וב PredictionIO האלגוריתם Co-occurrence מתייחס הרבה יותר לקשר בין הפריטים עצמם ולא רק לדירוג שלהם.
- Co-occurrence בשונה מ Slope One יודע לחבר בין מילים נרדפות ופריטים שווים (משתמש ב NLP)

בדיקת האלגוריתם

הבדיקה הייתה מול ה PredictionIO כמערכת, התוצאות היו טובות.

סקר ספרות

Articles:

- Cafferty, L. (2010, November 22). Gifts.com & Hunch Partner to Build the Ultimate Gift Recommendation Engine; New Feature Provides Personalized Recommendations for Facebook Friends . In PR Newswire. Retrieved March 23, 2013, from http://www.prnewswire.com/newsreleases/giftscom--hunch-partner-to-build-the-ultimate-gift-recommendation-engine-new-featureprovides-personalized-recommendations-for-facebook-friends-109929159.html

- CARLOS A. GOMEZ-URIBE and NEIL HUNT, Netflix, Inc. 2015
  The Netflix Recommender System: Algorithms, Business Value, and Innovation.
  http://dl.acm.org/citation.cfm?id=2843948
  This article discusses the various algorithms that make up the Netflix recommender system, and describes its business purpose. They also describe the role of search and related algorithms, which for us turns into a recommendations problem as well. They explain int the article the motivations behind and review the approach that they use to improve the recommendation algorithms, combining A/B testing focused on improving member retention and medium term engagement, as well as offline experimentation using historical member engagement data. They discuss some of the issues in designing and interpreting A/B tests. Finally, They describe some current areas of focused innovation, which include making there's recommender system global and language aware.

- Linden, G.; Smith, B.; York, J.; , "Amazon.com recommendations: item-to-item collaborative filtering," Internet Computing, IEEE , vol.7, no.1, pp. 76- 80, Jan/Feb 2003 doi: 10.1109/MIC.2003.1167344 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1167344&isnumber=26323

- Joseph A. Konstan · John Riedl
  Recommender systems: from algorithms to user experience
  http://files.grouplens.org/papers/algorithmstouserexperience.pdf

- Matrix Completion via Alternating Least Square(ALS)
  CME 323: Distributed Algorithms and Optimization, Spring 2015
  https://stanford.edu/~rezab/classes/cme323/S15/notes/lec14.pdf

- Soanpet .Sree Lakshmi , Dr.T.Adi Lakshmi,
  Recommendation Systems:Issues and challenges

https://pdfs.semanticscholar.org/18d9/378329888f7d78388ca37493f262eb1a9c3b.pdf

Books:

- Francesco Ricci · Lior Rokach · Bracha Shapira · Paul B. Kantor
  Recommender Systems Handbook
  http://www.cs.ubbcluj.ro/~gabis/DocDiplome/SistemeDeRecomandare/Recommender_systems_handbook.pdf

Existing systems:

- Item list (in Hebrew) suggestion for varius type of trips and users.
  Only list, without any features like custom suggestion for user or destination.
  http://www.xn--9dbhp2bfj.com/trips-camping-equipment-lists/overseas-equipment-list/

- Item list (in English) suggestion for varius type of trips and users.
  Only list, without any features like custom suggestion for user or destination.
  http://www.eaglecreek.com/blog/what-pack-ultimate-travel-packing-checklist

- PredictionIO is built on technologies Apache Spark, Apache HBase and Spray. It is a machine learning server that can be used to create a recommender system. The source can be located on github and it looks very active.
  http://predictionio.incubator.apache.org/index.html

- Racoon Recommendation Engine is an open source Node.js based collaborative filter that uses Redis as a store. It is effectively abandoned.
  https://www.npmjs.com/package/raccoon

- TensorFlow is an open-source software library for Machine Intelligence
  https://www.tensorflow.org/

- Seldon is a Java based prediction engine built on technologies like Apache Spark. It provides a demo movie recommendations application here.
  http://www.seldon.io/

- LensKit is a Java based research recommender system designed for small-to-medium scale.
  http://lenskit.org/

- Oryx v2 a large scale architecture for machine learning and prediction (suggested by Lorand)
  https://github.com/cloudera/oryx