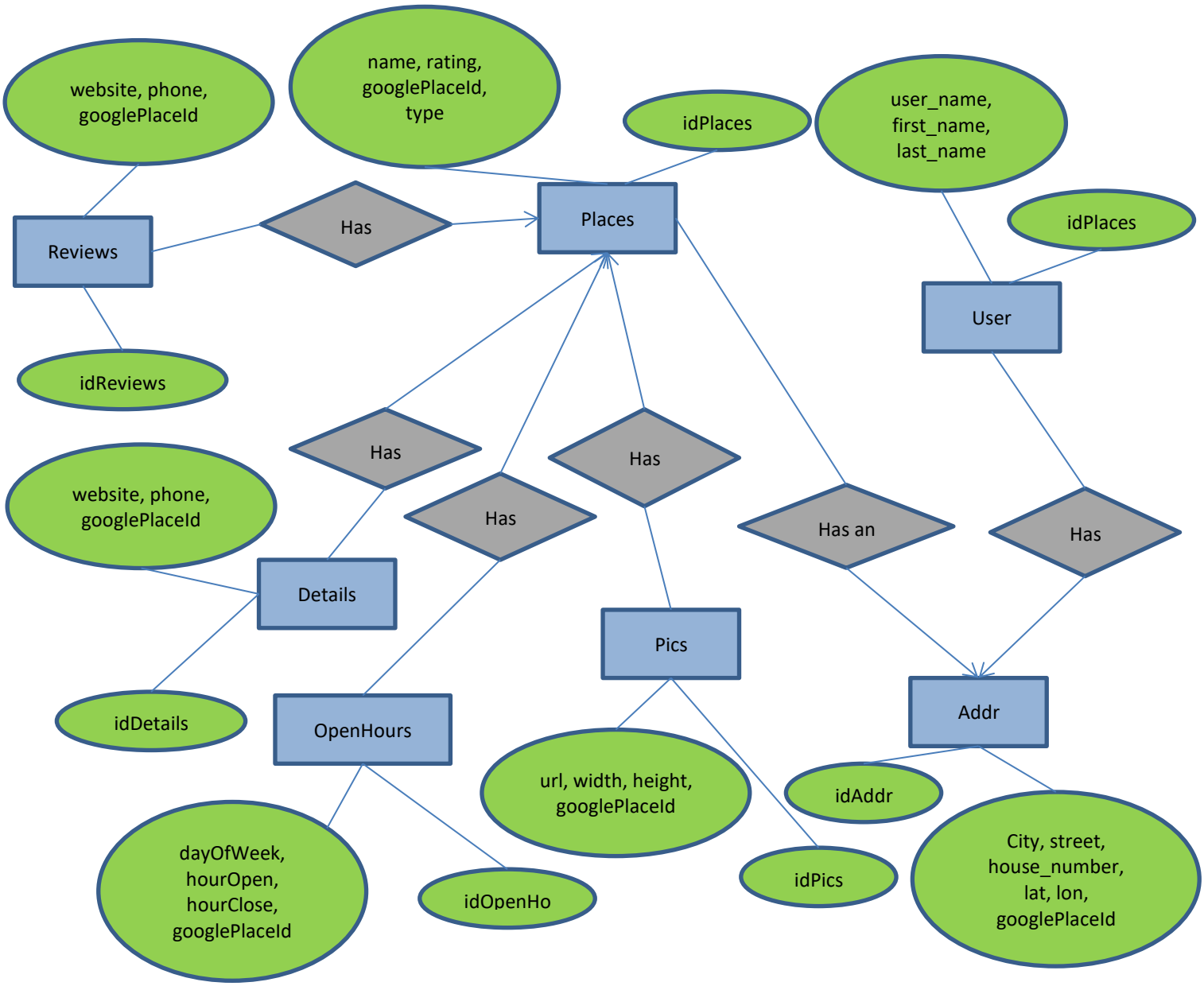
**SOFTWARE-DOCS**

**DB Scheme structure:**

בציור של ה DB להלן שמנו את כל השדות שאינם מפתח הטבלה באותו עיגול על מנת לחסוך במקום




הסברים:

לכל אחת מהטבלאות יש id ייחודי משלה. לטבלאות User וPlaces יש כתובת יחידה, על כן שמנו בטבלאות אלו שדה נוסף בשם addr_id. ניתן היה לחסוך בטבלה על ידי הכנסת הכתובת כחלק מהפרטים בכל שורה תחת הטבלאות Iser וPlaces, אך לכתובת יש פרטים רבים, והיא הייתה מנפחת את גודל הטבלאות האחרות. בנוסף, רצינו לתמוך בהרחבה של הDB מאוחר יותר לטבלאות נוספות שיעשו שימוש בכתובת, ולכן העדפנו לשמור על טבלה זו כטבלה נפרדת.

להרבה מהטבלאות יש שדה בשם googlePlaceId, שזה המזהה בו גוגל משתמש כמזהה חד חד ערכי לכתובת. בחרנו לשמור אותו בטבלאות רבות, ולא רק בטבלת places על מנת לתמוך בהכנסת פרטים על מקומות שאין לנו עדיין בDB, אך ייתכן שיוכנסו אליו בהמשך. למרות קשר הone-to-many שיש בין Places לטבלאות הפרטים, אנחנו מאפשרים לשמור פרטים שאינם מקושרים למקום קיים.

בנוסף, בהרבה מהטבלאות השתמשנו במזהה googlePlaceId בתור אינדקס, על מנת לאפשר שליפות מהירות יותר.

**DB optimizations:**

טבלת Reviews – הטבלה מאפשרת שליפות fulltext, כאשר אינדקסנו את שדה text המכיל את הטקסט של הביקורת. על מנת לאפשר שליפות fulltext, השתמשנו במנוע MyISAM.

טבלת Places – רוב השליפות בטבלה זו מתבצעות על פי עמודות type ו googlePlaceId, על כן דאגנו לאנדקס את שתי העמודות האלו.

טבלת User – אינדקסנו את עמודת שם המשתמש משום שהבדיקה האם המשתמש קיים מתבצעת לפי עמודה זו.

בטבלאות Reviews, Places, Pics, OpenHours, Addr, Details אינדקסנו על פי googlePlaceId

**Query desctiptions:**

updateUserQuery, insertUserQuery, inserAddrQuery – שלוש שאילתות עדכון ה DB שלנו. כאשר משתמש חדש מתחבר לתוכנה/רוצה לשנות פרטים, הוא צריך להכניס פרטים אישיים שיגיעו לטבלת User על ידי אחת משתי השאילתות הראשונות, וכתובת (בעזרת מנגנון השלמה אוטומאטי) שתעדכן את טבלת Addr בעזרת השאילתה האחרונה. הכתובת נשלפת מגוגל בעזרת ה API שהם מספקים, וכל הפרטים הרלוונטיים נשמרים לטבלת הכתובות.

bestAvgTypeQuery – שאילתה זו מחפשת את סוג מקום הבילוי (מסעדה, באר, חנות, מלון, מועדון) שיש לו את הרייטינג הממוצע הכי גבוה. השאילתה ראשית מחשבת בשאילתה בפנימית את הרייטינג הממוצע הכי גבוה שיש לסוג מקום בילוי כלשהו, ואחר כך בודקת בשאילתה החיצונית לאיזה מקום בילוי יש רייטינג ממוצע זה.

כשלעצמה השאילתה לא מאוד מעניינת, אך אנחנו משתמשים בתוצאה שלה כדי לבצע שליפה נוספת שתביא למשתמש מקומות באזור שלו מהסוג עם הרייטינג הכי גבוה.

placesInDistQuery – שאילתה זו מחפשת את כל מקומות הבילוי מסוג כלשהו שנמצאים ברדיוס מקסימאלי מסויים מנקודה נתונה. הקלטים לשאילתה הם הרדיוס המקסימאלי, סוג מקום הבילוי המבוקש והנקודה בה המשתמש נמצא.

השאילתה נעזרת בשאילתה הפנימית placeAndPics ששולפת עבור כל מקום תמונה יחידה (ואת כל הפרטים על אותו מקום)

השאילתה מחזירה את כל הפרטים מטבלת Places על המקום שהיא מצאה, מזהה הurl (שאינו באמת url בפועל אלא מזהה שעוזר לשלוף תמונה מגוגל) של תמונה ממקום זה, ואת המרחק של המקום.

בנוסף, השאילתה משתמשת בתת שאילתה בשם sub שמחשבת עבור כל מקום את המרחק שלו מהמשתמש. במרחק זה משתמשים כדי לפלטר את התוצאות, והוא חוזר כפלט של השאילתה.

כשאנחנו משתמשים בשאילתה זו בקוד, אנחנו ממיינים את התוצאות שהיא מחזירה לפי מרחק או לפי רייטינג על ידי הוספת שורת order by מתאימה לקוד שלה לפני ההרצה שלה וכך אנחנו מקבלים את התוצאות הכי מתאימות לשליפה שרצינו לבצע.

שליפה זו מחפשת בטבלאות על פי googlePlaceId, ולפי Places.type. כדי להפוך אותה ליעילה יותר אינדקסנו שני שדות אלו בטבלאות המתאימות.

serchInReviewsQuery – שאילתת fulltext. מאפשרת לחפש טקסט רצוי בתוך עמודת הtext של טבלת Reviews. זאת כדי שמשתמש יוכל לחפש משהו מעניין שהוא רוצה בביקורות. הנתון לשליפה הוא בטקסט שהמשתמש מחפש בעמודה.

כדי לייעל שאילתה זו, שינינו את מנוע החיפוש של טבלת Reviews לMyISASM, הפכנו את עמודת text למאונדקסת fulltext, והפכנו את עמודת googlePlaceId למאונדקסת (שכן משתמשים גם בעמודה זו כדי לסנן את השליפה ולהתאים את התוצאות לטבלת Addr)

getPictures – שליפה שמאפשרת לקבל את כל התמונות של המקומות שיש להם יותר מכמות נתונה של תמונות. השליפה מחזירה לכל תמונה את פרטי התמונה, את שם המקום ממנו היא נלקחה ואת מזהה הגוגל שלו. המקומות עם הכי הרבה תמונות חוזרים ראשונים.

שליפה זו נועדה למשתמשים שאוהבים לראות תמונות של המקום שהם מעוניינים בו, ורוצים לראות מקומות בהם יש תמונות רבות.

השליפה משתמשת הרבה בgooglePlaceId של הטבלאות השונות, ולכן אינדקסנו אותו כדי לייעל שליפה זו.

isOpenQuery – שליפה המאפשרת לבדוק אם מקום מסויים פתוח בשעה מסויימת או לא. הקלטים לשאילתה הם היום בשבוע, הgooglePlaceId של המקום אותו מחפשים, והשעה הנוכחית. מניחה שאם שעת הסגירה של מקום מסויים היא עד 6 בבוקר, סימן שהיא מסמלת זמן ביום למחרת ולא באותו יום.

השליפה נועדה כדי לבדוק עבור המקום שהמשתמש מסתכל עליו אם הוא פתוח כרגע.

כדי לייעל את השליפה אינדקסנו את googlePlaceId של טבלת שעות הפתיחה.

getTopDetails – שאילתה שמשיגה פרטים רבים על מקום מסויים. הקלטים לשאילתה הם המקום, והיום הנוכחי בשבוע. היא מחזירה את שם המקום, הכתובת שלו (כולל נ"צ), אתר ומספר טלפון, שעות פתיחה ביום הנוכחי, דירוג (rating) ואת ממוצע הrating של כל הביקורות שיש לנו על המקום.

השליפה עוברת על טבלאות רבות, ומוציאה את המידע הרלוונטי למקום מכל טבלה בDB.

היא שולפת מרוב הטבלאות על פי googlePlaceId, ולכן אינדקסנו את הטבלאות השונות לפי פרמטר זה.

# Code Structure

## Client Side

Our app is a single-page app, written with AngularJS using ui-router. Files by type (and folder) -

**Partials:**

- **login_signup** – Landing page, user login (by username only) and signup form in case of new user.
- **main** – Holds the navbar and a change location autocomplete box.
- **tab_content** – Primary content HTML, fills up according to selected tab

**Services:**

- **LoginService** – Provides API access to the server for all user-related actions (login, signup, update).
- **PlacesService** – API access for place related queries, such as getting a list of restaurants/hotels/etc, searching for a place by review text, getting full place details and so on.

**Controllers:**

- **LoginController** – Controller for login/signup flow.
- **MainController** – Injects general functions and utils to the scope, not that major.
- **PlacesController** – Primary app controller, used for controlling the various tabs and full place view, reacting to user actions (such as changing search radius, central address, etc).

**Other:**

- **app.js** – App initialization and configuration.
- **utils.js** – Utilities, such as a function collecting the HTML files from the server.
- **index.html** – The parent HTML file.
- **CSS** folder – Holding several css files we use.

## Server Side

Our server is Django based, with all the DB-related actions written separately using python's MySQLdb library.

**tauwebsite**: Main module.

- **settings** – Django settings file. Added our DB connection and static file serving for the HTML files.
- **urls** – Routing file, connects API calls to the relevant views.
- **utils** – Holds the DBUtils class, our implementation for the SQL queries.

- **serializers** – Basic serialization implementation for serializing query results before sending them back to the client.

**users**: User-related module. Only views.py file is relevant.

- **LoginView** – GET call returns a user by his username. POST call is used for both registering a new user and updating an existing one, differentiated by is_update parameter.

**places**: Place-related module. Only views.py file is relevant.

- **PhotograficPlacesView** – POST call retrieves places around central coordinates with at least 'num' pictures in our DB.
- **GeneralPlacesView** – GET call retrieves places with the highest average ranking type, POST call retrieves the best result of each type in radius 3km around the user.
- **PlacesByReviewView** – POST call gets all places with a text string in their reviews.
- **FoodView** - POST call retrieves restaurants around central coordinates in a certain radius.
- **BarView** - POST call retrieves bars around central coordinates in a certain radius.
- **ClubView** - POST call retrieves clubs around central coordinates in a certain radius.
- **HotelView** - POST call retrieves hotels around central coordinates in a certain radius.
- **ShopView** - POST call retrieves shops around central coordinates in a certain radius.
- **PlaceDetailsView** – GET call retrieves the full details of a place, including all pictures, reviews and opening hours.

**<u>API</u>**

במסגרת הפרויקט עשינו שימוש ב-google places API על מנת להביא מידע על המקומות השונים באתר שלנו.

בשלב ראשון השתמשנו ב-end point של search וביצענו חיפוש על פי קטגוריות אשר מצאנו כרלוונטיות לאתר שלנו ולתכנון התכולות בו. על מנת לקבל מקומות בעיר תל אביב השתמשנו בציון השם תל אביב כ-free text. לאחר קבלת תוצאות החיפוש הבחנו כי ה-API מציג לכל היותר 60 תוצאות לכל חיפוש, כאשר מחצית מהתוצאות חזרו על עצמן בשליפות שונות. בהתאם לכך, המשכנו לעבוד בשני כיוונים על מנת לצמצם את החיפוש:

- הרחבת מספר הקטגוריות – פרסרנו והכנסנו ל-DB את התוצאות הראשונות והתחלנו לנסות להבין יותר טוב מה יש לנו ביד. לאחר שעשינו group by לפי הקטגוריות, מצאנו קטגוריות נוספות שמתאימות לתכולות באתר שלנו. יצוין כי לכל מקום משויכות מספר קטגוריות (types) ולכן לדוגמא פרט לקטגוריה store אותה חיפשנו בהתחלה, מצאנו גם את הקטגוריות shoe store ו-clothes store.
- חיפוש באמצעות מיקום (location) – בחרנו נקודות ציון של כעשרים נקודות מרכזיות בעיר תל אביב (דוגמת כיכר דיזינגוף, מתחם שרונה, נמל תל אביב, יפו, שוק הכרמל וכיוב')

וביצענו חיפוש של כל הקטגוריות (כולל הקטגוריות החדשות שנוספו) מותנה במיקום וברדיוס של כשני קילומטרים.

בשלב שני, לאחר שהגענו למספר גדול דיו של מקומות יחודיים, השתמשנו ב-end point של details על מנת לקבל פרטים נוספים על כל אחד מהמקומות אותם מצאנו בחיפוש. לצורך כך, יצרנו רשימה של כל ה-google place id הייחודיים שחזרו ועבור כל אחד מהם ביצענו קריאה.
בסוף התהליך איחדנו את המידע שקיבלנו מכל אחד מה-end points שהוזכרו לעיל באמצעות המזהה של google place id.

כמו כן, השתמשנו גם ב-end point נוסף של photos, על מנת לקבל ולהציג את התמונות באתר.
בכדי לקבל לינק לתמונה נדרש לשלוח בקשת API נוספת עבור כל תמונה עם ה-reference המתאים לתמונה. כיוון שלא יכולנו לחרוג ממכסת הבקשות שהוקצה לנו (לפי ה-token) ובגלל שהלינק שניתן לכל בקשה של תמונה הוא זמני בלבד, החלטנו כי הבאת התמונות תעשה באמצעות בקשה שתשלח ותעובד במסגרת פתיחה של הדף הרלוונטי.

**General flow of the Application**

1. נכנסים לעמוד הראשי – מכניסים שם משתמש וסיסמא. אם המשתמש אינו קיים, מגיעים לעמוד הוספת משתמש, בו מכניסים את שאר הפרטים של המשתמש.
2. נפתח עמוד הבית ובו דברים שיש לעשות באזור שלך.
3. ניתן לבחור בעזרת תפריט עליון סינון רצוי לפרטים אלו – מקומות לאכול בהם בלבד, מקומות לשתות בהם (פאבים), מקומות לרקוד בהם (מועדונים), מקומות לעשות בהם שופינג ומקומות לינה.
4. ניתן ללחוץ על I'm feeling lucky, ולקבל תוצאות עבור המקום הפופולארי ביותר בלבד.
5. ניתן לבצע חיפוש עבור מילים ספציפיות בביקורת שיש למקום כלשהו בעזרת כתיבת הטקסט בתיבת טקסט ולחיצה על find.
6. ניתן להחליף את הכתובת השמורה עבור המשתמש שלך בעזרת חלונית עדכון כתובת בראש העמוד.
7. ניתן ללחוץ על כפתור pictures ולקבל רק מקומות עם תמונות רבות
8. ניתן ללחוץ על תמונה/שם של מקום בילוי, ולקבל פרטים מלאים על אותו המקום.