FoodFinder

פרוייקט SQL - מסמך תיעוד

מגישים:

חן אילון - 201617032

גל קשי - 204572861

יוגב ניומן - 305541021

בן בלייר - 302678446

תוכן

מבוא

"זמן הוא מה שאנחנו רוצים יותר מכל, אך גם הדבר שבוא אנו משתמשים באופן הכי גרוע" - וויליאם פן.

במציאות של היום לפעמים אין לנו מספיק זמן ללכת לקניות, או למצוא את המתכון הנכון לדיאטה שלנו.

לפעמים אנחנו צריכים מתכון מהיר עם מרכיבים שיש לנו במקרר, ולפעמים אנחנו צריכים מתכון שלא מכיל מרכיבים מסויימים, או ששייך לדיאטה כלשהי.

בשביל להקל על החיפוש יצרנו את FoodFinder, אפלקציה שמאפשרת שליפה חכמה על מתכונים.

FoodFinder מאפשרת שימוש מיטבי בזמן המשתמש, במרכיבים שנמצאים במקרר שלו ובדיאטות שלו.

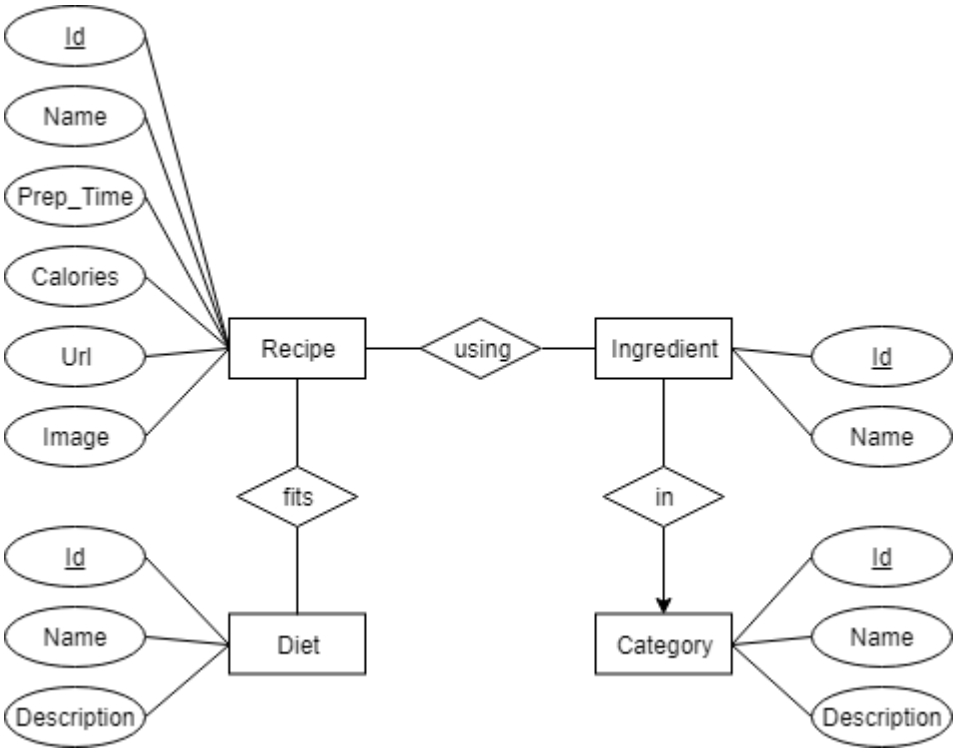
האפלקציה מאפשרת שליפה על מתכון לפי זמן ההכנה שלו, מרכיבים שנמצאים או לא נמצאים בו ולפי דיאטות.

האפלקציה מאפשרת גם שליפות חכמות יותר המותאמות למצבים יומיומיים כגון:

- שליפה על 3 מתכונים שניתן להכין יחד בשעתיים - למקרה שיש זמן מוגבל להכין ארוחה בת כמה מנות,
- שליפה שמאפשרת לבחור מתכון לפי זמן הכנה מקסימלי, שליפת מתכון מוכר, שליפת מתכון בהתאם לדיאטה ועוד.

מאגר הנתונים

תרשים



טבלאות ואינדקסים

להלן רשימת הטבלאות בDB שלנו.

סימונים:

- מפתח ראשי של טבלה יסומן ע"י קו תחתון
- אינדקס מסומן ע"י צבע ירוק
- אינדקס מסוג FULLTEXT מסומן בצבע כחול

Tables:

Category(Id, Name, Description)

Diet(Id, Name, Description)

Ingredient(Id, Name, Category_Id)

Recipe(Id, Name, Prep_Time, Calories, Url, Image)

Recipe_Diet(Recipe_Id, Diet_Id)

Recipe_Ingredient(Recipe_Id, Ingredient_Id, Weight, Description)

רוב השדות בטבלאות מובנים.

נסביר על השדות בטבלה Recipe:

- Id - מזהה יחודיי של המתכון
- Name - שם המתכון
- Prep_Time - זמן ההכנה של המתכון
- Calories - מספר הקלוריות עבור המתכון וכמויות הרכיבים המוכלות בו.
- Url - קישור לאתר שמכיל את המתכון המלא
- Image - קישור לתמונה של המנה

Foreign Keys:

- Ingredient.Category_Id -> Category.Id
- Recipe_Diet.Recipe_Id -> Recipe.Id
- Recipe_Diet.Diet_Id -> Diet.Id
- Recipe_Ingredient.Recipe_Id ->Recipe.Id
- Recipe_Ingredient.Ingredient_Id -> Ingredient.Id

נשים לב כי לכל מתכון, רכיב, דיאטה וקטגורית אוכל יש Id, על מנת שההצמדות יהיו קלות יותר.

כמו כן, מפורטים לעיל המפתחות הזרים שמטרותיהם להצמיד בין טבלאות.

אינדקסים נוספים שאינם מפתחות ראשיים או זרים:

- Ingredient.Name - מאפשר חיפוש קל על רכיבים במתכונים.
- Recipe.Name - מאפשר שליפת טקסט על שם מתכון. לדוגמה - מתכונים של עוגות.
- Recipe.Prep_Time - מאפשר שליפה קלה על זמן הכנה של מתכונים.
- Diet.Name - מאפשר שליפה נוחה על טבלת הדיאטות, אם רוצים לחפש דיאטה מסוימת.
- Category.Name - מאפשר שליפה נוחה על סוגי קטגוריות של רכיבי מזון.

מבנה הDB

נשים לב כי התלויות בDB שלנו הן:

Recipe.Id -> Recipe.Name, Recipe.Perp_Time, Recipe.Calories, Recipe.Url, Recipe.Image

Recipe.Id -> Diet.Id

Recipe.Id -> Ingredient.Id

Recipe.Id, Ingredient.Id -> Recipe_Ingredient .Weight, Recipe_Ingredient.Description

Diet.Id -> Diet.Name, Diet.Description

Category.Id - > Category.Name, Category.Description

Ingredient.Id - > Ingredient.Name, Category.Id

נסתכל על אוסף השדות:

R = { Recipe.Id ,Recipe.Name, Recipe.Perp_Time, Recipe.Calories, Recipe.Url, Recipe.Image, Diet.Id, Ingredient.Id, Recipe_Ingredient.Weight, Recipe_Ingredient.Description, Diet.Name, Diet.Description, Category.Id , Category.Name, Category.Description, Ingredient.Name}

קל לראות שיש הפרה של BCNF.

נפתור את הBCNF ונראה כי חלוקת הטבלאות שלנו תואמת לBCNF.

סתירה: Category.Id - > Category.Name, Category.Description

פירוק:

R1 = {<u>Category.Id</u>, Category.Name, Category.Description}

R2 = { Recipe.Id ,Recipe.Name, Recipe.Perp_Time, Recipe.Calories, Recipe.Url, Recipe.Image, Diet.Id, Ingredient.Id, Recipe_Ingredient.Weight, Recipe_Ingredient.Description, Diet.Name, Diet.Description, Category.Id , Ingredient.Name}

R1 היא BCNF.

R2 לא, סתירה: Diet.Id - > Diet.Name, Diet.Description

פירוק:

R1 = {Category.Id, Category.Name, Category.Description}

R2 = {Diet.Id, Diet.Name, Diet.Description}

R3 = { Recipe.Id ,Recipe.Name, Recipe.Perp_Time, Recipe.Calories, Recipe.Url, Recipe.Image, Diet.Id, Ingredient.Id, Recipe_Ingredient.Weight, Recipe_Ingredient.Description, Category.Id , Ingredient.Name}

R1, R2 הן BCNF.

R3 לא, סתירה: Ingredient.Id - > Ingredient.Name, Category.Id

פירוק:

R1 = {Category.Id, Category.Name, Category.Description}

R2 = {Diet.Id, Diet.Name, Diet.Description}

R3 = { Ingredient.Id , Ingredient.Name, Category.Id }

R4 = { Recipe.Id ,Recipe.Name, Recipe.Perp_Time, Recipe.Calories, Recipe.Url, Recipe.Image, Diet.Id, Ingredient.Id, Recipe_Ingredient.Weight, Recipe_Ingredient.Description}

R1, R2,R3 הן BCNF.

R4 לא, סתירה:

Recipe.Id, Ingredient.Id -> Recipe_Ingredient .Weight, Recipe_Ingredient.Description

פירוק:

R1 = {Category.Id, Category.Name, Category.Description}

R2 = {Diet.Id, Diet.Name, Diet.Description}

R3 = {Ingredient.Id , Ingredient.Name, Category.Id }

R4 = {Recipe.Id, Ingredient.Id, Recipe_Ingredient .Weight, Recipe_Ingredient.Description }

R5 = {Recipe.Id ,Recipe.Name, Recipe.Perp_Time, Recipe.Calories, Recipe.Url, Recipe.Image, Diet.Id, Ingredient.Id}

כעת כל הטבלאות הן BCNF.

נרצה לבצע מספר טיובים על מבנה הטבלאות, כדי שלא יכילו הרבה שורות מיותרות, וגם יתאימו יותר למבנה השליפות שלנו:

1. נוריד מR5 את השדה Ingredient.Id, כיוון שההצמדה שקיימת בטבלה R4 מספקת לנו מספיק מידע.
2. נחלק את R5 ל2 הטבלאות הבאות:

R5 = { Recipe.Id ,Recipe.Name, Recipe.Perp_Time, Recipe.Calories, Recipe.Url, Recipe.Image}

R6 = { Recipe.Id , Diet.Id}

מכיוון שלכל מתכון יכולות להיות מספר דיאטות (Many to Many), אך שאר השדות הם בעלי משמעות אחת (Many to One) - עדיף לפרק את הtuples ל2 סוגים.

העיצוב הסופי מקיים BCNF.

בנוסף, העיצוב מותאם לשליפות שהאפלקציה מאפשרת - שליפות מתכונים לפי דיאטות, רכיבים וקטגוריות מזון.

על השליפות נפרט בהמשך המסמך.

הData שבמאגר הנתונים

את המידע שהכנסנו למאגר הנתונים השגנו מ3 מקורות:

- Edamam API -
  Edamam מציעים שליפות חינמיות על מאגר המתכונים שברשותם באמצעות "Get Requests" שמחזיר מידע בJSONים. מהמידע שהAPI מחזיר סיננו את המידע שמעניין אותנו וחילקנו אותו לטבלאות.
  בנוסף למתכונים, שלפנו מהמאגר לעיל סוגי דיאטות.
  הקשיים העיקריים שנאלצנו לעבוד איתם בAPI הזה:
  - הגבלה על כמות השליפות בפרק זמן מסויים.
  - פרסור של רכיבים במתכונים - הAPI מחזיר רכיבים בצורה של משפט שכולל מספרים, מידות, רכיבים ופעולות, בפורמטים משתנים ולא מוגדרים.

- TheMealDB API -
  מאגר זה כולל פחות מתכונים באופן משמעותי, לכן לא השתמשנו בו לצורך זה.
  מהמאגר הזה שלפנו כ-500 רכיבים של מתכונים. במאגר רכיבים זה השתמשנו על מנת לבנות אוסף שאילתות לשליפה מ-Edamam. כמו כן המרכיבים נשמרו בטבלה Ingredient.
  גם השליפות מהמאגר הזה נעשות באמצעות "Get Requests" שמחזיר מידע בJSONים.

- טיובים -
  - 500 המרכיבים ששלפנו לא היו מספיק מכלילים בשביל לפרסר את כל רכיבי המתכונים שהורדנו. לכן, נאלצנו "לכרות" מהרכיבים הלא מפורסרים שקיבלנו, רכיבים מפורסרים. באופן זה הגדלנו את רשימת הרכיבים לרשימה שמונה 1308 רכיבים.
  - בנוסף, על מנת לאפשר שליפות מתכונים שמכילים קטגוריות של רכיבים (לדוגמה - מתכונים עם ירקות), מיינו את הרכיבים ל14 קטגוריות שונות.

סה"כ הורדנו כ-30,000 מתכונים.
(הקוד לשליפת המידע מצורף במקום הנדרש)

ספריות וחבילות חיצוניות בשימוש

בשימוש ביצירת הDB וכריית המידע -

השתמנו בספריות פייתון הבאות:

- requests
- json
- pandas
- datetime
- time
- os
- re
- math
- pymysql

אפלקציה ושליפות SQL

השליפות שנעשות בDB

1) השלמה אוטומטית לשם חלקי של מתכון או רכיב:

- בשימוש בעמוד 3,4 באתר.
- נעזרות בטבלאות:
  - Recipe ובאינדקס על Recipe.Name
  - Ingredient ובאינדקס על Ingredient.Name

2) סוגי דיאטות:

- בשימוש בעמוד 3 באתר.
- נעזרת בטבלה Diet.

3) מתכון לפי שם חלקי: (שליפה על מחרוזת)

- בשימוש בעמוד 4 באתר.
- נעזרת בטבלה Recipe ובאינדקס על Recipe.Name

4) שליפת מתכון לפי זמן הכנה מינימלי ו/או זמן הכנה מקסימלי:

- בשימוש בעמוד 3,4 באתר.
- נעזרת בטבלה Recipe ובאינדקס על Recipe.Prep_Time

5) מתכון לפי סוג דיאטה:

- בשימוש בעמוד 4 באתר.
- נעזרת בטבלאות: Recipe, Recipe_Diet, Diet. ובאינדקסים Recipe.Id, Diet.Id, Recipe_Diet.Recipe_Id, Recipe_Diet.Diet.Id

6) <u>מתכון לפי רשימת רכיבים כלולים:</u>

- בשימוש בעמוד 3,4 באתר.
- נעזרת בטבלאות Recipe, Recipe_Ingredient, Ingredient ובאינדקסים: Recipe.Id, Ingredient.Id, Recipe_Ingredient.Recipe_Id, Recipe_Ingredient.Ingredient_Id, Ingredient.Name

7) <u>מתכון לפי רשימת רכיבים אסורים:</u>

- בשימוש בעמוד 3,4 באתר.
- נעזרת בטבלאות Recipe, Recipe_Ingredient, Ingredient ובאינדקסים: Recipe.Id, Ingredient.Id, Recipe_Ingredient.Recipe_Id, Recipe_Ingredient.Ingredient_Id, Ingredient.Name

8) <u>רשימת מצרכים מחולקים לקטגוריות לפי רשימת מתכונים:</u>
- בשימוש בעמוד 5 באתר.
- נעזרת בטבלאות Recipe, Recipe_Ingredient, Ingredient, Category ובמפתחות הראשיים והזרים בטבלאות אלו.

9) <u>ארוחה לפי מספר מתכונים\מנות:</u>

- הסבר - שליפה של קבוצת מתכונים (בין 2 ל5 מתכונים)
- בשימוש בעמוד 3 באתר.
- נעזרת בכמה מופעים של הטבלה Recipe.

10) <u>ארוחה לפי רשימת רכיבים כלולים או אסורים:</u>

- הסבר - שליפה של קבוצת מתכונים (בין 2 ל5 מתכונים) כך שכולם מכילים רכיבים מהרשימה של ה Included המוגדרת ו/או שכולם לא מכילים רכיבים מהרשימה של הExcluded.
- בשימוש בעמוד 3 באתר.
- נעזרת בכמה מופעים של בטבלאות Recipe, Recipe_Ingredient, Ingredient.

11) <u>ארוחה לפי זמן הכנה מינימלי ו/או מקסימלי:</u>

- הסבר - שליפה של קבוצת מתכונים (בין 2 ל5 מתכונים) כך שהזמן הכולל של ההכנה של כולם חסום בזמן הכנה מינימלי ו/או מקסימלי.
- בשימוש בעמוד 3 באתר.
- נעזרת בכמה מופעים של הטבלה Recipe.

<u>הערות נוספות:</u>
- קיימות שליפות נוספות על מתכונים לפי כל השילובים האפשריים של שליפות 4-7.
- קיימות שליפות נוספות על ארוחות לפי כל השילובים האפשריים של שליפות 9-11.
- מיונים לכל השליפות ניתן לבצע לפי: שם המתכון עולה\יורד, זמן הכנה עולה\יורד וסך קלוריות עולה\יורד.
- כל שליפות המתכונים והארוחות נשלפות בחלקים בהתאם לעמוד התוצאות הנוכחי.

- בכל שליפה מורכבת נעזרנו באינדקס כמתואר למעלה משום שלאחר מספר נסיונות ראינו כי השליפה קורסת ולפיכך גם בחרנו את האינדקסים.

אפלקציה - מבנה הקוד

האפלקציה בנויה באמצעות Python ו-django בעיקרה.

אין ממשק משתמש או admin.

מודולים חשובים:

- api.py - מרכיב שליפות SQL לפי בקשה ממשק האתר. מתחבר לDB, שולף, ומחזיר את תוצאות השליפה בJSON.
- api_url.py - אחראי על קריאה לשאילתות הנכונות שנקראות ממשק האתר בapi שמתחבר לDB שלנו ושולף תשובות.
- views.py - מחזיר את הדף אותו אנו רוצים לראות.

האתר נכתב באמצעות קבצי html, js ו-css המצורפים לקבצי הSRC (תחת תיקיות static, templates).

ספריות וחבילות חיצוניות בשימוש

ספריות Python בשימוש -

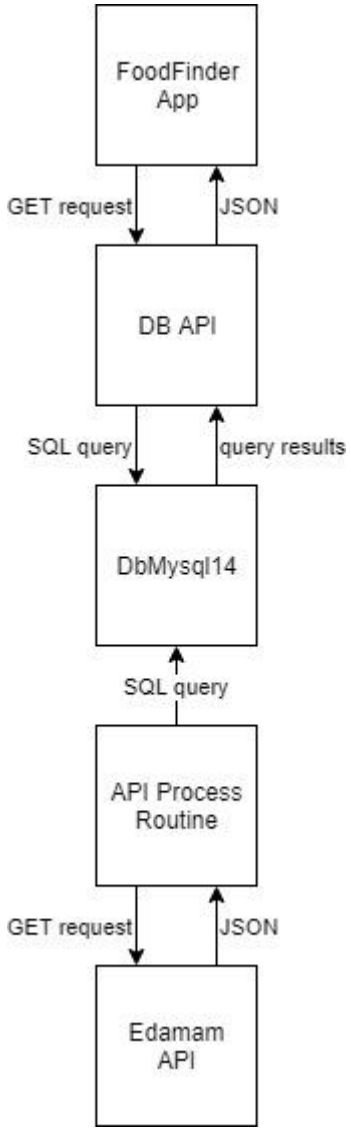
- __future__ (unicode_literals)
- textwrap
- django.http (HttpResponse)
- django.shortcuts (render)
- django.views.generic (TemplateView)
- django.views.generic.base (View)
- django.http (JsonResponse)
- django.apps (AppConfig)
- django.conf.urls (include, url)
- json
- project.views (jsonApi)
- pymysql (cursors)

הסבר כללי על היישום

היישום מאפשר שליפות נוחות מהDB, בצורה קריאה למשתמש הסביר.

מדריך למשתמש מצורף בקובץ נוסף.

להלן תרשים כללי של הפרוייקט:



- שליפות ממאגרי המידע בהם השתמשנו נעשו באמצעות api של "get requests" והחזירו תשובה בצורה של JSON.
- את התשובות פרסרנו והכנסנו לטבלאות בDB שיצרנו (ומוסבר לעיל).
- על מנת להתחבר לDB, יש להשתמש בAPI שכתבנו, שמכין את השליפות, שולף ומחזיר לשואל תשובה בפורמט JSON.
- האתר מועלה לשרת של האוניברסיטה וניתן להתחבר אליו דרך האינטרנט.
- האתר מתשאל את הapi שלנו באמצעות "get request", מקבל תשובה בJSON ומציג אותה למשתמש.