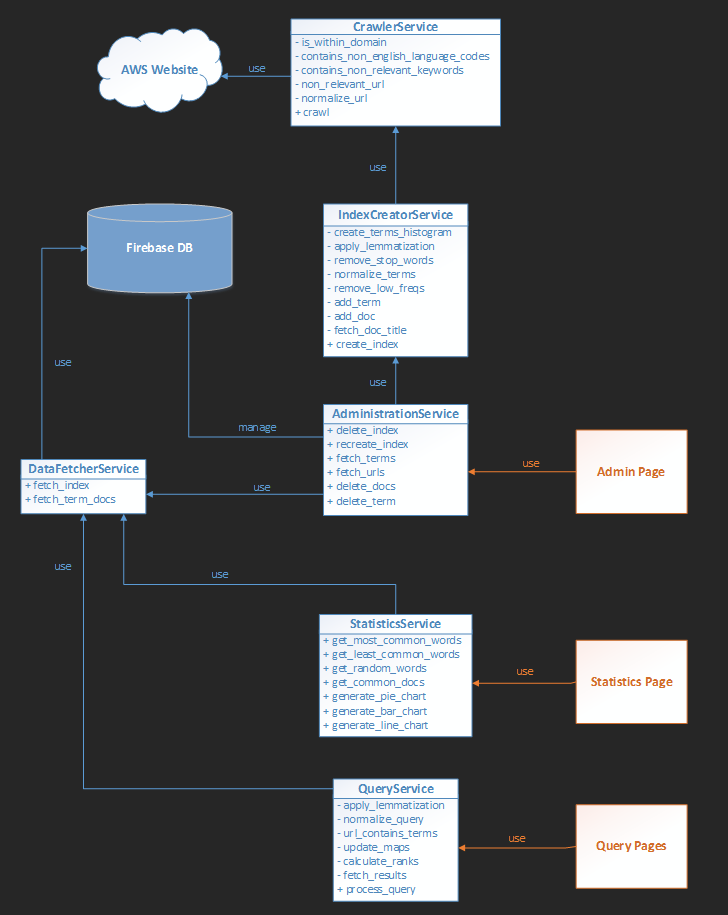
בפרויקט השתמשנו בארכיטקטורת Microservices. להלן דיאגרמה:



הסבר על הארכיטקטורה:

CrawlerService

מיקרו-שירות של זחלן שמחלץ דפי HTML סטטיים מ- Domain מסויים שמועבר אליו. בפרויקט מועבר אליו ה- Domain של AWS והוא מוגדר לחלץ כ- 200 דפים.

**is\_within\_domain** – בדיקה שהאתר עדיין חלק מה- Domain.

**contains\_non\_english\_codes** – בדיקה שכתובת האתר מכילה קוד של שפה שהיא אינה אנגלית (כך שנוכל לדלג על כתובת זו).

**contains\_non\_relevant\_keywords** – בדיקה שכתובת האתר מכילה מילות מפתח שלא רלוונטיות (כך שנוכל לדלג על כתובת זו).

**non\_relevant\_url** – שימוש בשלושת המתודות לעיל, על מנת לפסול כתובת במהלך הזחילה.

**normalize\_url** – מחיקה של Query Parameters ו- Fragments מכתובות במהלך הזחילה (כדי להימנע משכפול כתובות).

**crawl** – פעולת הזחילה שמתבצעת באלגוריתם BFS.

IndexCreatorService

מיקרו-שירות שאחראי על יצירת האינדקס. מאגד מידע טקסטואלי מתוך דפי HTML, ומבצע עליו עיבוד להפקת האינדקס. בפרויקט מקבל stop words מותאמים ל- AWS.

**create\_terms\_histogram** – יצירת היסטוגרמה שמכילה את תדירות ההופעות של מילים בתוך דף HTML מסויים.

**remove\_stop\_words** – הסרת מילים נפוצות שאינן מוסיפות ערך משמעותי להיסטוגרמה.

**normalize\_terms** – המרה של מילים בהיסטוגרמה לצורתן הבסיסית, ע"י שימוש ב- apply\_lemmatization שמבצע למטיזציה למושג מסויים, כדי למנוע כפילויות.

**remove\_low\_freqs** – הסרה של מושגים מההיסטוגרמה שהופיעו בתדירות נמוכה מהרף המוגדר (מוגדר בברירת מחדל  
ל- 7), כדי לצמצם את האינדקס למילים יותר משמעותיות.

**add\_term** – יצירת רשומה חדשה עבור מילה באינדקס אם היא אינה קיימת.

**add\_doc** – הוספת מידע על מסמך מסויים לרשומת מילה באינדקס.

**fetch\_doc\_title** – שליפת כותרת הדף (נשמר כחלק מהמידע על מסמך מסויים).

**create\_index** – יצירת האינדקס על ידי הפעלת שירות הזחלן לחילוץ הדפים, ושימוש במתודות המתוארות לעיל.

AdministrationService

מיקרו-שירות לניהול האינדקס.

**delete\_index** – מחיקת האינדקס הקיים מה- Database.

**recreate\_index** – שימוש בשירות יצירת האינדקס לצורך יצירת אינדקס מחדש (או לראשונה).

**fetch\_terms** – חילוץ כלל המושגים הקיימים באינדקס.

**fetch\_urls** – חילוץ הלינקים תחת מושג מסויים באינדקס.

**delete\_docs** – מחיקה של לינקים ספציפיים תחת מושג מסויים באינדקס.

**delete\_term** – מחיקה של מושג מסויים מהאינדקס.

DataFetcherService

מיקרו-שירות לשליפת מידע מה- Database.

**fetch\_index** – שליפה של כל האינדקס.

**fetch\_term\_docs** – שליפה של כל נתוני המסמכים עבור מושג מסויים.

StatisticsService

מיקרו-שירות לביצוע מניפולציות להפקת סטטיסטיקות וכן ליצירת גרפים בצורה גנרית.

**get\_most\_common\_words** – מבצע מניפולציה על האינדקס כדי להפיק את המושגים הנפוצים ביותר.

**get\_least\_common\_words** – מבצע מניפולציה על האינדקס כדי להפיק את המושגים הכי פחות נפוצים.

**get\_random\_words** – מבצע מניפולציה על האינדקס כדי להפיק את מספר הפעמים שמושגים רנדומליים בשימוש.

**get\_common\_docs** – מבצע מניפולציה על האינדקס כדי להפיק את המסמכים שבהם יש הכי הרבה מושגים.

QueryService

מיקרו-שירות לעיבוד שאילתה ושליפת התוצאות הרלוונטיות מהאינדקס.

**normalize\_query** – נרמול של השאילתה על ידי הפעלת apply\_lemmatization והמרה ל- lower case של כל מושג בשאילתה.

**url\_conatins\_terms** – בדיקה האם מושגי השאילתה נמצאים גם בכתובת (משפר את הדירוג).

**update\_maps** – עדכון של אינפורמציית העיבוד על מסמכים.

**calculate\_ranks** – חישוב הדרגות של המסמכים שנשלפו.

**fetch\_results** – מיון המסמכים לפי הדרגות ושליחתם.

**process\_query** – עיבוד השאילתה. מבצע שימוש במתודות המתוארות לעיל.

**3 – פיצ'ר**

בעמוד הרצת שאילתה, הוספנו פיצ'ר בשם "AWS Fun Fact" המציג בלחיצת כפתור באופן אקראי עובדה מעניינת על שירותי הענן של אמאזון. בכל לחיצה המשתמשים מקבלים עובדה אחרת. הפיצ'ר ממומש בקוד ה- GUI של עמוד הרצת השאילתה, ואין כרגע קישוריות לעמוד המנהל. אם יתאפשר במגבלות הזמן, קיימת האופציה להוסיף את העובדות לבסיס הנתונים ולאפשר למנהל שליטה על הוספה ומחיקה של עובדות אך בשלב זה הדבר אינו מתוכנן.

**הסבר על מנגנון הדירוג שרשמנו**

מנגנון הדירוג הנוכחי מחושב לפי מספר גורמים:

1. **חיפוש המונחים בכתובת ה-URL (URL-Based Boost):**
   * הפונקציה \_\_url\_contains\_terms בודקת האם כתובת ה-URL מכילה מונחים מתוך השאילתה.
   * במידה ויש התאמה, ניתן **Boost** לדירוג של המסמך (כפול 2). כך מסמכים עם התאמה ב-URL זוכים לדירוג גבוה יותר.
2. **התאמה למונחים בשאילתה:**
   * עבור כל מסמך, מחושב יחס ההתאמה בין המונחים שנמצאו במסמך לבין אורך השאילתה.
   * אם יש **Boost** (תוצאה מ-URL), היחס משוקלל עם ה-Boost, אחרת הוא מחושב רגיל.
3. **חישוב הדירוג הסופי:**
   * הדירוג של המסמך (docs\_ranks) מתעדכן על בסיס הכפלה עם "מכפיל התאמה" המחושב עבור אותו מסמך.

**יתרונות המנגנון:**

* מעלה מסמכים בעלי URL רלוונטי.
* לוקח בחשבון את כמות ההתאמות של המונחים מתוך השאילתה.

**שיפור אפשרי:**

* להוסיף שקלול נוסף למילים המופיעות בכותרת הדף (Title Boost) באמצעות fetch\_doc\_title.
* להפחית את דירוג המסמך עבור מילים מאוד נפוצות באינדקס (שקלול הפוך לתדירות).

**פענוח האופרטורים בשאילתה:**

* יש לזהות את האופרטורים (AND, OR) בתוך השאילתה.
* לדוגמה, עבור שאילתה כמו term1 AND term2 OR term3, יש לפרק את השאילתה לחלקים נפרדים:
  + (term1 AND term2) -> חיתוך תוצאות.
  + (result1 OR term3) -> איחוד תוצאות.

הפרויקט יכול להיות ממומש באופן Serverless תוך שימוש ב-AWS. הנה התוכנית:

1. **שירותים מרכזיים:**
   * **CrawlerService**:
     + להפעיל את הזחלן כ-Function ב-AWS Lambda. ניתן להפעיל את הפונקציה כאשר קובץ URL חדש מתווסף ל-S3.
   * **IndexCreatorService**:
     + יישום ב-Lambda שיופעל לאחר ש-Crawler מסיים ומשתמש בתוצאות לזיהוי מילות מפתח והוספה לאינדקס.
     + שמירת האינדקס בדינמיות ב-AWS DynamoDB.
   * **QueryService**:
     + שירות Lambda שמופעל על שאילתות החיפוש, ניגש ל-DynamoDB, מחשב דירוגים ומחזיר תוצאות.
   * **StatisticsService**:
     + ניתן להפעיל פונקציות Lambda ליצירת גרפים סטטיסטיים, כאשר הגרפים עצמם נשמרים ב-S3.
2. **תשתית ללא שרתים:**
   * שימוש ב-AWS API Gateway ליצירת REST API שמתווך בין ה-Client לבין הפונקציות ב-Lambda.
   * אחסון ה-Frontend ב-S3 כ-Website Static Hosting.
   * אחסון הלוגים והשגיאות ב-CloudWatch.
3. **יתרונות Serverless:**
   * אין צורך לנהל שרתים.
   * המערכת גמישה למספר משתמשים משתנה (Scalability).
   * עלות נמוכה יותר (Pay-as-you-go).
4. **תרחיש אפשרי להרחבה:**
   * בעתיד ניתן להוסיף פיצ'ר של "AWS Fun Fact" כאירוע שמופעל מ-DynamoDB Stream ומחזיר עובדות רלוונטיות באופן אוטומטי ללקוח.