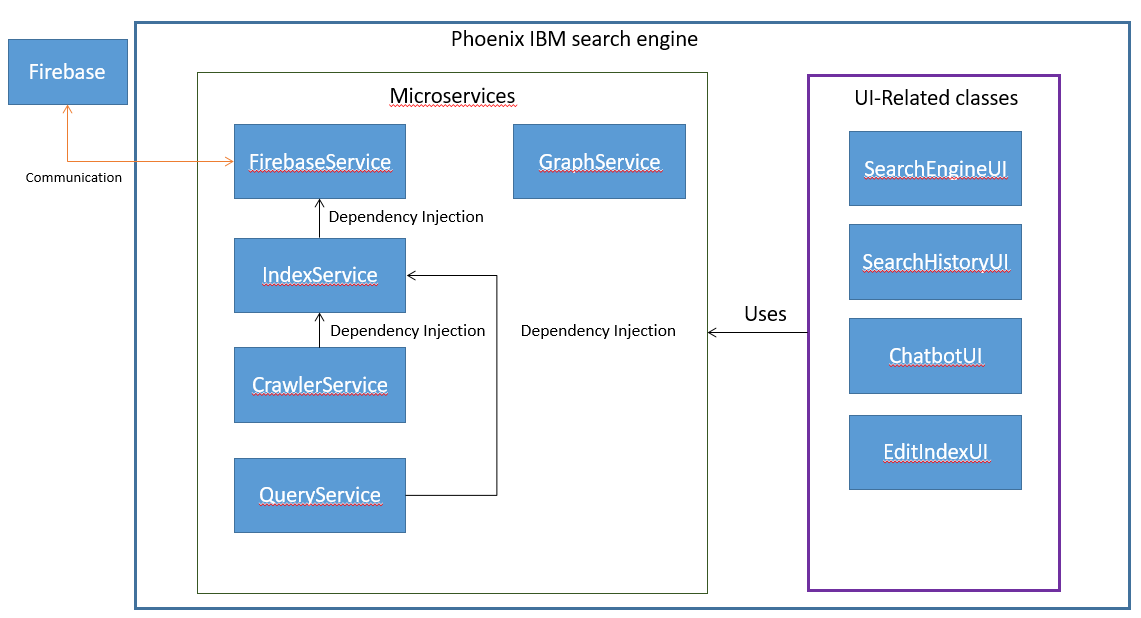
מדדי KPI לפרויקט:

* User Satisfaction
  + שהמשתמש יהיה מרוצה מתוצאת החיפוש, כלומר שהתוצאות חיפוש יכילו תוצאותנכונות.
  + מדידה: שתהיה לפחות תוצאה אחד נכונה \ רלוונטית ב-5 התוצאות הראשונות.
* Performance
  + שהמשתמש לא ימתין יותר מדי זמן לתוצאת החיפוש.
  + מדידה: אחרי לחיצה על כפתור החיפוש לא יעברו יותר מ-3 שניות עד שתוצג תוצאה כלשהי.
* Scalability & Elasticity
  + שהמערכת תהיה מודולרית, וקלה להרחבה.
  + מדידה: פיתוח מודולרי, שכל פונקציה ומחלקה תהיה עצמאית, כלומר בלי תלות במשתנים גלובלים.

בונוס: כיצד לדעתכם ניתן לממש את הפרויקט שלכם כ-serverless?

* ניתן לממש אותו כ-serverless על ידי שמירת כל השירותים לוקלית. ובכך מורידים את התלות שלהם בשרת.

הארכיטקטורה:



* הסבר בעמוד הבא.

הסבר - חלקנו את הפונקציונליות העיקרית למחלקות שירותים:

* FirebaseService: מטרתו לייעל את התקשור עם הבסיס נתונים, לבצע שליפת ועדכון נתונים בבסיס נתונים.
* הממשק:
* FirebaseService(db\_url='defaultConnURL')  
  Initializes the service with a Firebase database connection.
* get\_rev\_index\_from\_DB()  
  Retrieves the revision index from the Firebase database.
* update\_rev\_index\_in\_db(index\_p)  
  Updates the revision index in the Firebase database by clearing it first and then saving the provided data.
* IndexService: מטרת השירות לאחסן את האינדקסים לוקלית, הוא מאחסן אינדקס הפוך לצורך חיפוש, ואת האינדקסים של הדפים (לצורך סטטיסטיקה), והוא יאפשר לעדכן את האינדקס לפי בקשות (כגון הוספת מילה לאינדקס, הסרת דף מהאינדקס וכו'). הוא מאפשר שמירה ושליפה של האינדקס מבסיס הנתונים על ידי dependency-injection של שירות FirebaseService.
* הממשק:

 IndexService(index=None, firebaseService=None)  
Initializes the service with an optional initial index and FirebaseService instance.

 process\_soup(url, soup)  
Processes the given soup (parsed HTML) for a URL and updates the reverse index and URL index.

 get\_reverse\_index()  
Returns the reverse index.

 get\_url\_index(url)  
Retrieves the index of words for a specific URL.

 set\_rev\_index(newRevIndex)  
Sets the reverse index to the provided index and updates internal mappings.

 get\_index\_of\_word(w)  
Retrieves the reverse index entry for a given word.

 add\_new\_word(word)  
Adds a new word to the reverse index. Returns True if successful, otherwise False.

 remove\_word(word)  
Removes a word from the reverse index. Returns True if successful, otherwise False.

 remove\_url(url)  
Removes all entries related to a URL from the reverse index and URL index. Returns True if successful, otherwise False.

 save\_in\_db()  
Saves the reverse index to the Firebase database if a FirebaseService instance is available.

 load\_from\_db()  
Loads the reverse index from the Firebase database if a FirebaseService instance is available.

 index\_toString(minView=True)  
Converts the reverse index to a string representation. A minimal view is used if minView=True.

 resetService()  
Resets the reverse index and URL index to their initial states.

* CrawlerService: מטרתו לבצע שירותי crawling על דפים, ולעדכן את השירות אינדקס תוך כדי (בעזרת dependency-injection).
* הממשק:

 CrawlerService(indexService, baseURLs=set(['https://www.ibm.com/us-en', 'https://www.ibm.com/topics']), maxDepth=100)  
Initializes the crawler with an IndexService instance, optional base URLs, and a maximum crawl depth.

 initCrawlingProcess()  
Starts the crawling process for all base URLs up to the specified maximum depth.

 crawl\_website(base\_url, max\_pages=100)  
Crawls the specified base URL, extracting links and processing pages up to the maximum number of pages.

 crawl\_single\_url(url)  
Crawls a single URL, processes its content, and updates the index. Returns a status message indicating success or failure.

 get\_crawled\_urls()  
Returns a set of all URLs that have been crawled.

 resetService()  
Resets the crawler service, clearing the crawled URLs and count.

* QueryService: משתמש לביצוע שאילתות ומחזיר תוצאות חיפוש, הוא גם מאפשר קבלה של ההיסטוריית חיפוש. השירות צריך dependency-injection של indexService על מנת לבצע את השאילתות.
* ממשק:

 QueryService(indexService)  
Initializes the service with an IndexService instance.

 query(query)  
Executes a search query and returns ranked results.

Supports boolean operations (AND, OR).

 get\_history()  
Returns the query history as a list of past queries and their results.

* GraphService: מטרתו ליצור את הגרפים בצורה אוטומטית עבור אינדקס הפוך שהוא מקבל בבנאי.
* ממשק:

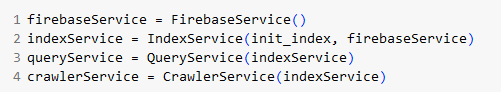
 GraphService(rev\_index)  
Initializes the GraphService with the reverse index data.

 **get\_heatmap(self)**  
Generates and returns a heatmap displaying shared URLs between index terms.

 **get\_barChart(self)**  
Generates and returns a bar chart of word occurrences across webpages, sorted in descending order.

* המשך בעמוד הבא

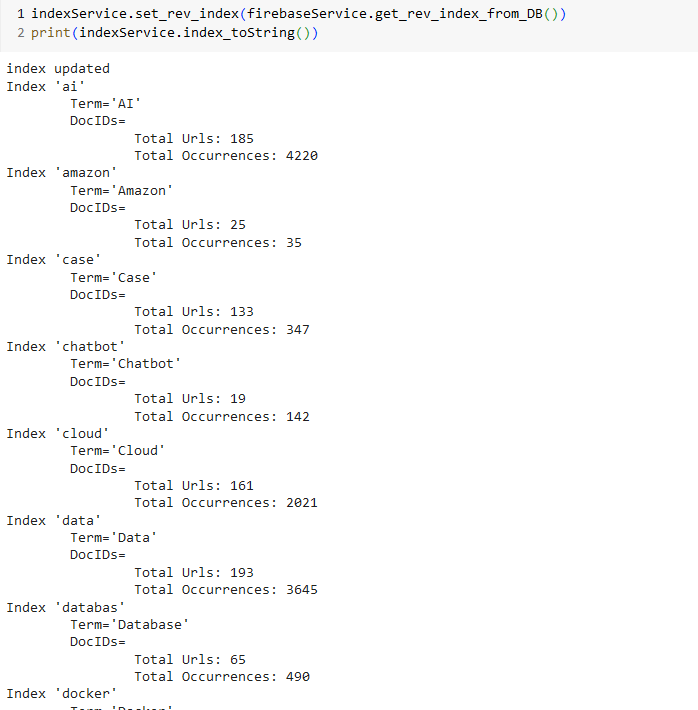
דוגמאות לשימוש:



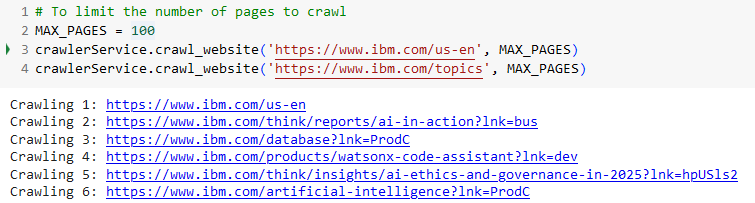
* שירות FirebaseService וב-IndexService:
  + עדכון אינדקס בבסיס נתונים



* + שליפת האינדקס מבסיס הנתונים:



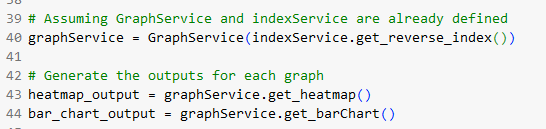
* דוגמא לשימוש ב-CrawlerService:



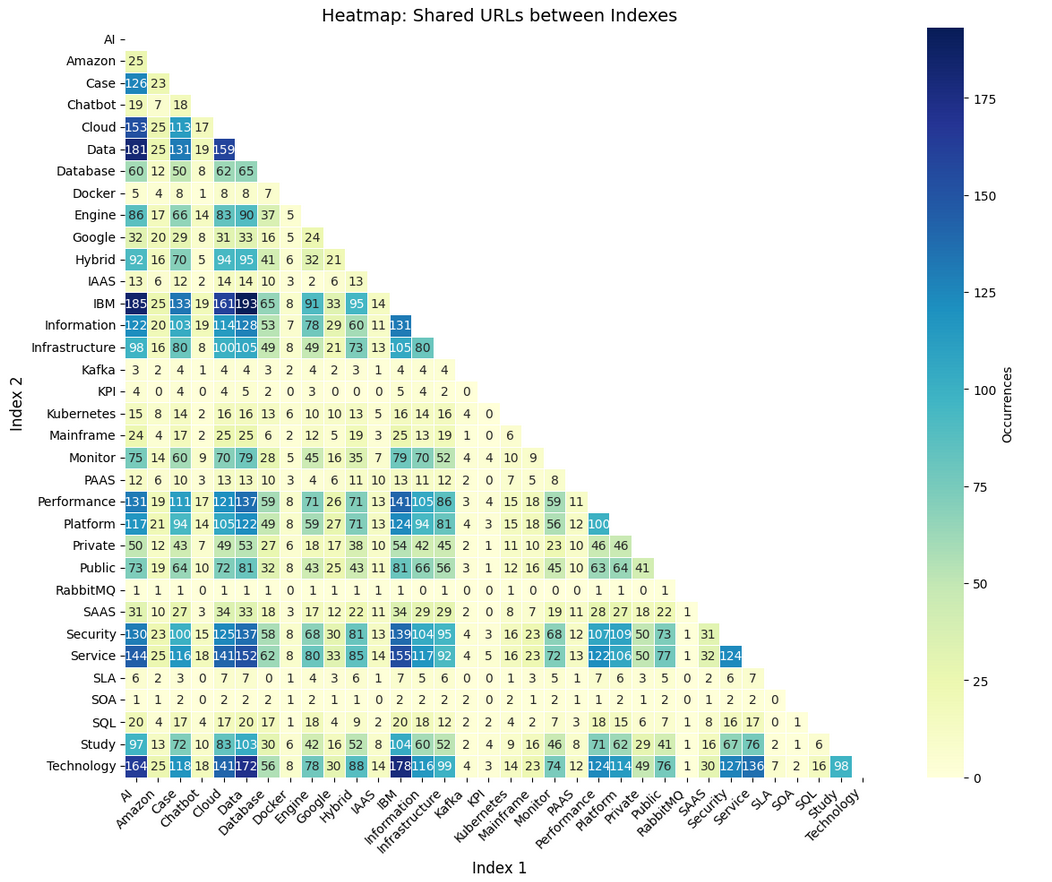
* דוגמא לשימוש ב-QueryService:

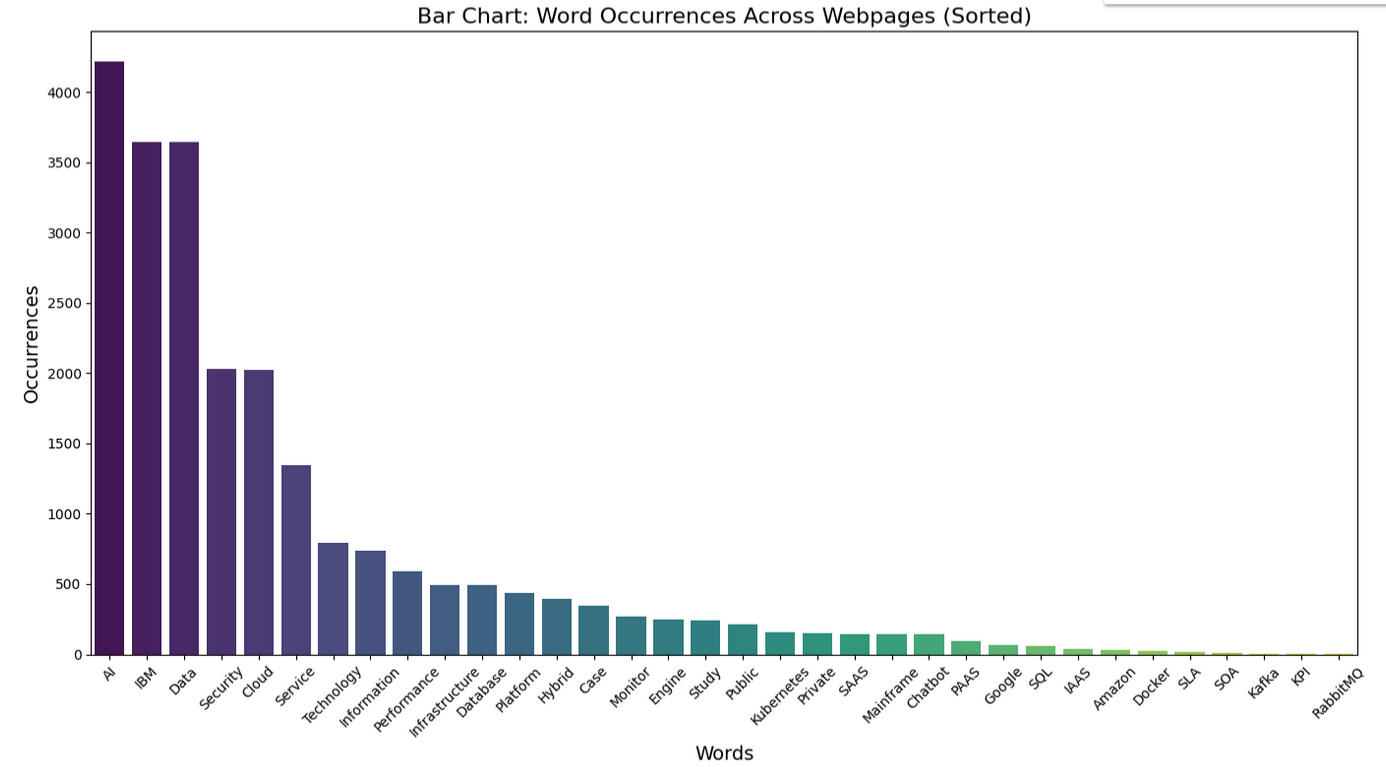


* המשך בעמוד הבא.
* דוגמא לשימוש ב-GraphService



* + התצוגה של הגרפים במסכים:





* המשך בעמוד הבא.

תוצאות בדיקות:

* בדיקת User Satisfaction:



* + תוצאה הכי נכונה הייתה עם הדירוג הכי גבוה.
* בדיקת Performance:
  + החיפוש לקח פחות משנייה.
* בדיקת Scalability & Elasticity:
  + בשירותים אין תלות במשתנים גלובלים, ואם יש תלות בשירות אחר אז הוא מקבל אותו כ-Dependency-Injection בבנאי מה שמאפשר לפיתוח יותר מודולרי.