

רצינו לומר תודה...

לאחר חודשים ארוכים של השקעה, יגיעה ומאמץ חודשים של עליות וירידות, הצלחות וגם נפילות, אנו רוצות לומר תודה לכל המסייעים ואלו שהפכו את הפרויקט הזה לאפשרי.

תודה רבה לגב' מירי ויכלדר, רכזת המגמה על כל הדאגה, המסירות וההדרכה.

תודה מיוחדת לגב' תמר קארפ, על הליווי הצמוד במהלך הפרויקט, עוד משלב החשיבה וגיבוש הנושא המרכזי, היא הייתה שם לכוון אותנו, להקשיב גם לרעיונות פחות טובים ולמצוא גם בהם את הנקודות הטובות עד שהגענו לרעיון שאהבנו. לאורך כל הדרך היא הקשיבה הסבירה וכיוונה אותנו עד הפרטים הקטנים. תודה רבה!

כמו כן אנו רוצות להודות לגאולה המנחה שלנו, קודם כל תודה על הדאגה, הרגשנו שזה חשוב לה לא פחות משזה חשוב לנו.

הרגשנו שתמיד יש לנו למי לפנות בכל שאלה או התייעצות שעלתה. תמיד היה לה מענה מקצועי, היא לא ויתרה לנו לרגע על שום פרט אך עם זאת הייתה שם לעודד שלא נתייאש. ידעה להסביר והכל בסבלנות בלי סוף! מעריכות מאוד.

ונסיים בתודה ענקית לבורא עולם שבזכותו אנחנו כאן.

 תודה

מבוא

אנו חיים בעידן בו כולם מחפשים את הדבר הבא, את החידוש שיהפוך את החיים שלנו לקלים יותר, הכל חייב להיות נגיש אם לא- הוא מהר מאוד הופך להיות לא רלוונטי.

כשחשבנו על רעיון לפרויקט רצינו ליצור משהו שיעמוד בקריטריונים של מהיר ונגיש, היה חשוב לנו שהוא יעסוק בנושא שיגע בעולמות של שתינו, משהו שנראה את עצמנו משתמשות בו גם כן. תמיד עמד מול עינינו החשיבות שהפרויקט יהיה שימושי וכמובן בנוי בצורה הנוחה ביותר למשתמש ועם זאת להתמקד בתהליך הלמידה וההתפתחות שלנו בעזרת הטכנולוגיות הרווחות בשוק כיום על מנת לצבור ידע וניסיון.

וכך נולד האתר **Quili**-Quick list רשימה מהירה.

כמה פעמים נתקלתם במתכון, רציתם להכין אותו ו... חסר לכם מצרך מסוים אז אתם מוותרים ועוברים לדבר הבא או מנסים לאלתר ממה שיש. מה אם נשנה את הגישה, נבנה את רשימת הקניות שלנו לפי המתכונים אותם אנו רוצים להכין.

האתר **Quili** בא לנהל לנו את התכנון היומיומי של ניהול המטבח שלנו, החל מתכנון ארוחות קבועות או חד פעמיות ועד לניהול רשימת הקניות שלנו בצורה נוחה והכי חשוב- מסודרת ויעילה יותר.

שם האתר **Quili**-Quick list רשימה מהירה משקף את הרעיון שעומד מאחורי הפרויקט-בניית רשימת קניות מהירה וקלה.

התרגשנו לראות את הפרויקט נבנה מאפס, למדנו דברים חדשים התנסינו בטכנולוגיות וספריות שלא הכרנו והשתדלנו לעשות את זה בדרך האוטנטית- להרגיל את עצמנו לעבודה "באמת", לא תמיד היה קל, אבל לא ויתרנו לעצמנו ואחרי כל קושי הגיע הסיפוק של ההצלחה והנה התוצאה לפניכם.

הגדרות דרישות ותיאור כללי

מטרת המערכת:

מטרת המערכת היא להפוך, בלחיצה אחת, כל מתכון או תפריט לרשימת קניות חכמה. המערכת מאפשרת למשתמש לשמור ולנהל מתכונים או להוסיף אותם לתפריט המזון השבועי שלו ויוצרת באופן מיידי רשימת קניות על פי מתכונים אלו- באפשרות המשתמש לנהל ולערוך רשימה זו.

היקף העבודה:

מספר השעות שהוקדשו עבור פרויקט זה עומד על כ- 850 שעות.

תיאור חומרת המערכת:

מחשב PC / מכשיר נייד בעל דפדפן וחיבור לרשת.

תיאור תוכנת המערכת:

כתיבת צד לקוח נעשה בשפת Angular8, שפה פופולארית הנוחה מאוד הן למתכנת והן למשתמש.

כתיבת צד שרת נעשה בשפת C#, שפה מתקדמת, שימושית ומלאת פונקציונאליות.

בניית מסד הנתונים נעשה באמצעות Server SQL בטכנולוגית Entity Framework.

מדריך למתכנת:

תיאור המערכת:

אתר המאפשר לכל משתמש לנהל את המטבח שלו, החל מתכנון ארוחות וניהולן ועד לניהול רשימת הקניות בהתבסס על המתכונים/התפריט שבחר. הלקוח בוחר את המתכונים הרצויים ושומר אותם לפי הימים בחודש, המערכת בונה לו לפי המתכונים הרצויים רשימת קניות מתאימה.

ממשק המערכת:

באתר נבנה כמה וכמה ממשקים:

1. ממשק ראשי הכולל:
 - אודות המערכת ?
 - מעלות המערכת ?
 - מתכונים מומלצים על בסיס הפופולאריות שלהם.
2. ממשק התחברות:
 - כניסה – למשתמש רשום.
 - כניסה למשתמש חדש ע"י רישום למערכת.
3. ממשק אזור אישי הכולל:
 - מתכונים מועדפים.
 - לוח תכנון ארוחות.
 - רשימת קניות
4. ממשק חיפוש מתכון/הוספת מתכון:
 - נדרש לחפש מתכון במאגר(שנשלף מהAPI ע"י מילת החיפוש שהמשתמש מקליד)
 - ולהגדיר עבורו תדירות ותאריך הכנה ?

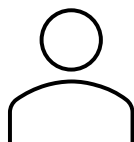
5. ממשק כללי:?

• דף מתכון

אופן זרימת המידע למערכת:

משתמש- רשום:

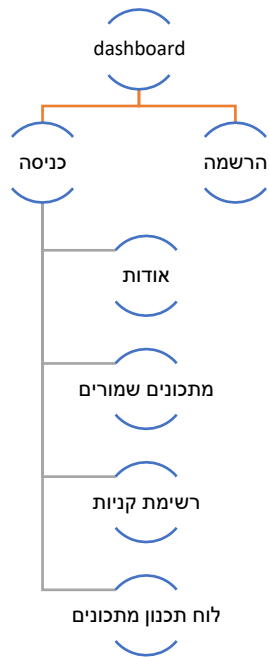
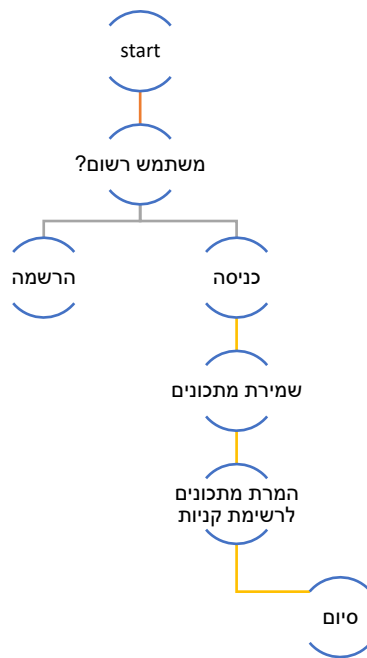
משתמש- אורח:



- הרשמה למערכת

- כניסה למערכת

-



תכנות

תיאור

המערכת נכתבה בשפת C#.NET בסביבת Visual Studio 2019 ועושה שימוש בטכנולוגיות מתקדמות, בתוכם: Web API, Angular8, Entity Framework וכן, בספריית Bootstrap. כאשר צד ה-Server פותח בטכנולוגיית Web API בשיטות המתקדמות ביותר ש-NET מציע וצד ה-Client פותח באמצעות Angular8.

טכנולוגיות מתקדמות

- א. API Web – ממשק תכנות יישומים לשרתי אינטרנט ו/או לדפדפני אינטרנט. מטרתו "לתווך" בין צד הלקוח (client) לצד שרת (server) על מנת לבצע את הפונקציונליות הנדרשת לצד לקוח. עיקר תפקודו הוא בקבלת נתונים מצד הלקוח והעברתם לשרת, ולהפך.
קטע קוד לדוגמא: קבלת בקשת GET לקבלת רשימת המתכונים של לקוח מסוים, קבלת בקשת post להוספת מתכון.
- ב. LINQ – שפת צד שרת, שיטת כתיבה חדשנית מובנית ב-NET, המיועדת לריצה על lists מערכים של אובייקטים וחיפוש בתוכם בצורה פשוטת, חכמה וקלילה.
קטע קוד לדוגמא: שליפת מתכון מסוים ע"י שפת LINQ בצורה ישירה וקלילה, ללא משפטי תנאי מיותרים.
- ג. Entity Framework – טכנולוגית עבודה מתקדמת מול DB. יצירת אובייקטים ומיפוי מול טבלאות ה-DB, כאשר אופן העבודה ועיבוד הנתונים בשפת LINQ כמוזכר לעיל.
- ד. Angular8 – טכנולוגיה מתקדמת המאפשרת פיתוח מלא מקצה לקצה בצד לקוח, כלומר Angular אספה התנהגויות שונות שעד היום נכתבו בצד השרת ואפשרה אותם בצד הלקוח (services, injectable ועוד).
Angular תומכת באפשרות ליצור אפליקציות של Single Page Application – S.P.A, כלומר, רק חלק בדף מסוים מתרענן ומתחלף וכל שאר החלקים נשארים אותו הדבר. דבר זה מאפשר גלישה מהירה וחלקה אשר משפרת את חווית המשתמש. בכל קומפוננט (רכיב המיועד לתצוגה, המכיל את כללי התצוגה ואת הניהול שלה) בה רוצים להציג תוכן כלשהו משתמשים במאפיין – template. מאפיין זה אומר ל-Angular איזה תוכן HTML להציג עבור קומפוננט זו, ה template מכיר את הקומפוננט ואת כל השדות והפונקציות הכתובות בה. Angular משתמשת בשיטת ה- two way data binding, כלומר, כל שינוי הנעשה ב- template משפיע ישירות על הקומפוננט ולהפך.
- ה. Typescript – שפה חדשה הדומה לתקן החדש של JavaScript.
- ו. SweetAlert2, Syncfusion ej2, Angular Material – חבילות components מעוצבות.

עקרונות התכנות

- א. מבחינה מקצועית – בחירת שפות וטכנולוגיות התואמות את צרכי האפליקציה, התמקדנו בחיפוש שפות וטכנולוגיות מהמתקדמות ביותר, המאפשרות מודולריות ופונקציונליות גבוהה, וכן ברצוננו לרכוש ידע מקצועי מהמובילים בתחום.

- ב. מבחינת מבנה הקוד – בנינו את הפרויקט בצורה מודולרית ביותר ע"י חלוקה לשכבות (DAL, BL, GUI) בכדי להפוך את הקוד לנהיר ודינאמי כך שמפתחים נוספים יוכלו להבין את הקוד בקלות, להוסיף על הפרויקט, ובמידת הצורך לשנות.
- ג. מסכי המערכת נכתבו בצורה פשטנית וברורה ביותר לעין, כך שניתנים לשימוש בקלות, ונותנים חווית משתמש.
- ד. התפיסה הרווחת כיום, לגבי אפליקציות Web היא כמה שיותר קוד בצד ה-client מכיוון שכל פנייה לשרת מכבידה על המערכת וגורמת לחוויית משתמש ירודה. השתדלנו שרוב הנתונים שלא צריכים עיבוד בשרת ימומשו בצד ה-client, ועל כן, השתמשנו בספריית angular כדי לחסוך פניות מיותרות לשרת.

חלופות שפות תכנות:

- שפת HTML** - שפה בסיסית שלא מאפשרת הצגת נתונים דינאמיים ומתאימה במיוחד לאתרים פשוטים. השפה מאפשרת למתכנתים לקשר בין מגוון שפות תכנות ולייצר קוד מורכב. בוני אתרים רבים משתמשים בשפה כדי לחתוך ולעצב תבניות ולהציג מידע באינטרנט. כמו כן, ניתן להשתמש בשפה כדי ליצור קישורים לנתונים, כדי להציג טקסטים, תמונות, סרטוני וידאו ומוסיקה וכדי לבצע מגוון פעולות נוספות.
- שפת JavaScript** - הפקודות של JS משפיעות על הדפדפן והמחשב ומקלות על המתכנתים לבנות אלבומי תמונות, קישורים ותפריטים שנפתחים. חשוב לציין כי השפה לא מובנת למנועי החיפוש משום שהיא מתקשרת בין הדפדפן והמחשב ולא עם השרתים, דבר שעשוי לפגוע בתהליך הקידום של האתר.
- MVC** - מקובל לחלק יישום תוכנה למספר שכבות נפרדות: שכבת התצוגה (ממשק משתמש), שכבת התחום העסקי (לעיתים נקראת גם "שכבת הלוגיקה העסקית") ושכבת הגישה לנתונים.
- בתבנית MVC שכבת התצוגה מחולקת בנוסף לתצוגה ובקר.
- יש המחשיבים את התבנית כתבנית עיצוב, אך בהשוואה לתבניות עיצוב אחרות, MVC עוסקת במבנים בקנה מידה בינוני-גדול ולכן נחשבת גם כתבנית ארכיטקטורה.
- מודל** - המודל הוא ייצוג מסוים, מוכוון תחום עסקי, של המידע עליו פועל היישום. המודל, למרות הדעה הרווחת, אינו שם אחר לשכבת התחום העסקי והוא נפרד ממנה.
- תבנית MVC אינה מזכירה במפורש את שכבת הגישה לנתונים, מכיוון ששכבה זו היא מתחת למודל, או נעטפת על ידו.
- תצוגה** – תפקידה להמיר את נתוני המודל לייצוג המאפשר למשתמש לבצע פעולת גומלין כלשהי. לרוב מדובר על המרה לממשק למשתמש כלשהו. תבנית MVC משמשת רבות ביישומי Web, בהם התצוגה היא דף HTML והקוד אוסף מידע דינאמי לדף.
- בקר** - תפקידו לעבד ולהגיב לאירועים המתרחשים בתצוגה, לרוב, כתגובה לפעולה של המשתמש. בעיבוד האירועים, הבקר עשוי לשנות את המידע במודל, באמצעות תפעול שירותים המוגדרים בו. בקרים מורכבים מתבססים לרוב על יישום של תבנית command.

לסיכום:

בחרנו לכתוב את המערכת בטכנולוגיית Angular8 ב- Client side,

Entity Framework, Web API - C# - Server side, מול בסיס נתונים SQL Server, בטכנולוגיית Entity Framework.

בעיות ופתרון:

במהלך הפרויקט נתקלנו במספר אתגרים שהיינו שצריכות להתמודד איתם.

- המידע והנתונים שאיתם אנו מנהלות את האתר ואף חלק מהפונקציות מתבססים על API חיצוני. נוצרת מכך בעיה של תלות האתר בגורם חיצוני שלא בשליטתנו, במידה ואנו מתבססים רק על שליפות מהAPI במקרה של קריסה של הAPI האתר שלנו יקרוס גם כן.
לאחר מחשבה, החלטנו על שמירת המידע באופן יעיל בDB, בכך צמצמנו באופן משמעותי את מספר הקריאות לAPI ואת הפונקציונליות שמחייבת חיבור אליו. בכך במקרה של קריסת הAPI האתר שלנו ימשיך לפעול ולא יקרוס לחלוטין.
- אתגר נוסף שהיינו צריכות להתמודד איתו הוא עבודת הצוות מרחוק. אומנם אנו יודעות להתנהל אחת עם השנייה ואין לנו קושי בהתמודדות עם עבודת הצוות אך המרחק היווה עבורנו אתגר גדול בחלוקת העבודה והסנכרון בין הפרויקטים.
את אתגר זה לקחנו כהזדמנות, החלטנו ללמוד להשתמש בכלי לא מוכר- GIT ולעבוד איתו, מה שיפתור לנו את בעיית המרחק ועם זאת ייתן לנו ידע וניסיון בכלי שימושי בשוק העבודה כיום. בהתחלה זה לא היה קל אך צלחנו את זה.
- אחת הדרישות באתר הוא שהאתר יהיה רספונסיבי. מתחילת כתיבת הפרויקט נתקלנו בבעיות רספונסיביות מכיוון שגדלי המסך שלנו שונים, מה שגרם לנו להבין את חשיבות הרספונסיביות ורצון להשקיע בכך מההתחלה.
פתרון בעיה זו ניהלנו באמצעות ספריות עיצוב של Bootstrap.

אבטחה:

השקענו עמל והשקעה רבה בהגנות מידע של המשתמשים באתר.

על מנת לאבטח את המידע על הנתונים שלנו בSQL ע"י כתיבת הconnectionStrings בקובץ Web.config, מה שהופך אותו למוגן יותר.

כמו כן, דאגנו שהודעות השגיאה של המערכת יהיו ידידותיות למשתמש ולא חושפות מידע פנימי על המערכת שלנו.

שרידות המערכת ותיעוד:

במהלך הפרויקט שמנו דגש על אבטחת הנתונים, לשם כך בחרנו להשתמש בטכנולוגיית Web API מכיוון שהשימוש בטכנולוגיה זו מחייב שימוש בפרוטוקול HTTPS.

פרוטוקול זה הינו בטוח לשימוש ומוכר בעולם התכנות.

בשליחת נתונים מצד הלקוח לצד השרת (post) הנתונים מועברים כאובייקט DTO ולא כשרשור גלוי בשורת Url.

המערכת ארכיטקטורת

את כתיבת הקוד כתבנו במודל השכבות, כאשר לכל שכבה יש תפקיד משלה:

1. **Dal (Data Access Layer)** שכבה המקשרת את הפרויקט לDB.
2. **BI (Business Layer)** שכבה זו כוללת את הלוגיקה בפרויקט.
3. **Entities** שכבה זו ממירה משתנים מסוג DB לEntities ולהפך.
4. **Web API (Application Programming Interface)** תוכנת שרת המספקת אינפורמציה לתוכנת לקוח.
5. **GUI (Graphical User Interface)** שכבה המספקת ממשק משתמש נוח ונעים לעין, שכבה זו נכתבה ב Angular8.

שימוש ב-API חיצוני

מראה הSolution של הפרויקט:

צילום מסך

תיאור קוד פונקציות ומחלקות

-Client Side

GUI (Graphical User Interface)

את צד הלקוח כתבנו בAngular8.

בניית צד הלקוח הינה חלק מרכזי וחשוב בפרויקט.

את עקרונות השימוש בשפה הכרנו במהלך הלימודים והעמקנו את הידע והכתיבה על ידי שימוש בספריות לעיצוב ולמידה מקוונת של פיצ'רים נוספים בשפה.

הבנייה כוללת שימוש ב-Components, Classes ו-Services.

כל Component מורכב מ:

- קובץ html- מראה הממשק.
- קובץ CSS- עיצוב קובץ html.
- קובץ TS (Type Script) - כתיבת הקוד.

Classes – קבצים המגדירים לנו תכונות לאובייקטים מסוימים.

Services – קבצים לוגיים המיועדים לשמירת נתונים ולניהול תקשורת עם הWeb API.

דוגמה לComponent

```
✓ ingredients
# ingredients.component.css
<> ingredients.component.html
TS ingredients.component.spec.ts
TS ingredients.component.ts
```

בקומפוננט זה מוצגת למשתמש רשימת הקניות שלו, באופן ברירת מחדל על פי כל המתכונים ששמר לשבוע הקרוב ובאפשרותו לשנות את ההגדרות על ידי שינוי טווח התאריכים או על ידי סינון לפי מתכונים מתוזמנים. בנוסף על כל מוצר ברשימה המשתמש יכול לסמן האם יש לו צורך בו והאם יופיע ברשימה הסופית. בסיום העריכה המשתמש יכול להדפיס את הרשימה ולייצא אותה.

צילומי מסך

-Server Side

Web API (Application Programming Interface)

בשכבה זו השתמשנו בשירות המאפשר גישה לסביבת הלקוח דרך הדפדפן, שירות זה מאפשר קריאות מסוג POST ומסוג GET.

עבור כל מחלקה בנינו controller המכיל פונקציות שונות מסוגים: GET, PUT, POST ו DELETE.

כל פונקציה בcontroller מתממשקת לסביבת הclient.

צילומי מסך

שכבת ה-BI (Business Layer)

Entities

בשכבה זו עבור כל טבלה בDB בנינו מחלקת Entities.

בכל מחלקה קיימות ארבע פונקציות המרה:

1. פונקציה המקבלת משתנה מסוג DB וממירה אותו לסוג Entities.
2. פונקציה המקבלת משתנה מסוג Entities וממירה אותו לסוג DB.
3. פונקציה המקבלת רשימה מסוג DB וממירה אותה לרשימה מסוג Entities.
4. פונקציה המקבלת רשימה מסוג Entities וממירה אותה לרשימה מסוג DB.

המרות אלו מאפשרות לנו "לתקשר" בין צד השרת לצד הלקוח.

צילום מסך

BI (Business Layer)

פירוט על הBL

(Data Access Layer) Dal

בשכבה זו אנו מתחברים לDatabase (מסד הנתונים).
במחלקת Dal נמצאות כל הטבלאות הקיימות בDB כולל קשרי הגומלין שלהן.
טבלאות אלו הינן הקרובות ביותר לDatabase ומייצגות אותו, הן המגשרות לבין DB לBI.

צילום מסך

:Entity Framework

כלי המאפשר גישה קלה לנתונים הנמצאים בDatabase.
כלי זה יוצר לנו מודל לכל ישות וכל מודל מקושר לטבלה בDB.
בתרשים הבא ניתן לראות את הטבלאות שנוצרו ב Entity Framework:

צילום מסך

מסד הנתונים

בבניית מסד הנתונים השקענו מחשבה ותכנון על מנת להגיע לתוצאה של מבנה נתונים יעיל ונוח לעבודה.

מסד הנתונים מכיל טבלאות המקושרות ביניהן בקשרי גומלין בכדי לשמור על חוקיות הנתונים ותאימות בין הטבלאות השונות. מסד הנתונים מנורמל ומכיל מפתחות ראשיים וזרים שתפקידם למנוע כפילויות ולזרז את שליפת הנתונים.

בתרשים הבא ניתן לראות את מבנה הטבלאות הקיימות בDB ואת קשרי הגומלין שלהן:

צילומי מסך

כמו שכתבנו לעיל השתמשנו בטכנולוגיית Entity Framework על מנת לעבוד עם מסד הנתונים.

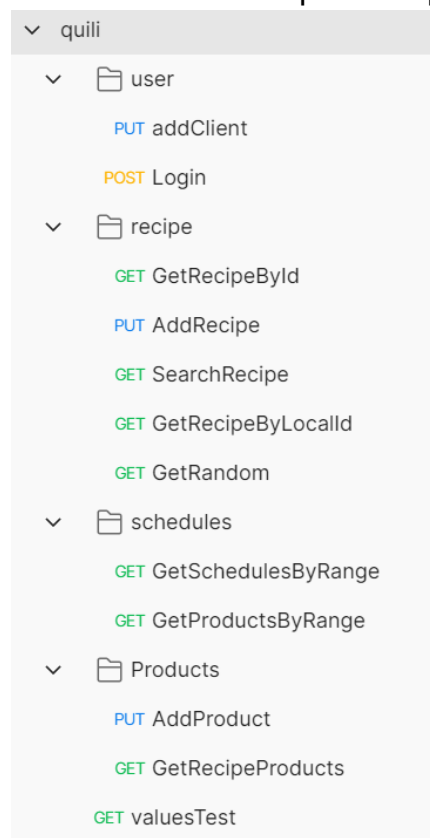
המנגנון של Entity Framework יוצר את Classes על פי החוקיות:

- טבלאות בקשר של יחיד ליחיד- יוצר שני Classes בקשר של הורשה.
- טבלאות בקשר של יחיד לרבים- יוצר Class לכל טבלה, בClass של הרבים יוצר מופע מסוג Class של היחיד ובClass של היחיד יוצר אוסף מסוג Class של הרבים.
- טבלאות בקשר של רבים לרבים- יוצר Class ובו אוספים של Classen המקושרים.

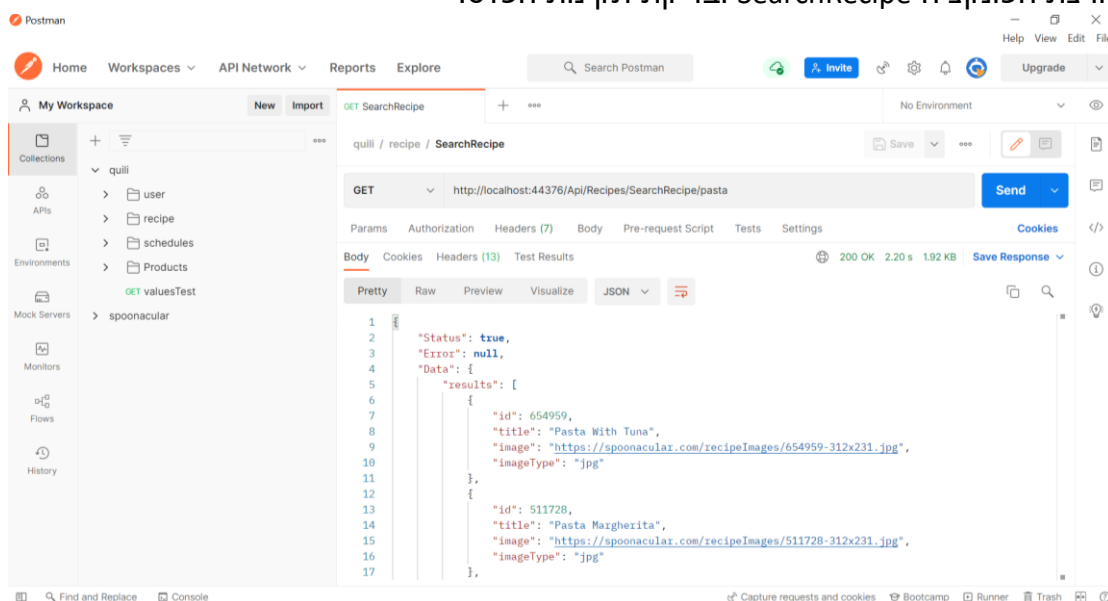
בדיקות תוכנה:

השתמשנו בPostman על מנת לבדוק את שכבת הWeb API, הרצנו כל פונקציה על מנת לעקוב אחרי הפלט שלה ולבדוק את תקינות המידע, הן מהAPI החיצוני והן בבדיקת המידע הנשלף והנשמר בDB.

קבצי הבדיקה בPostman:



הרצת הפונקציה SearchRecipe ובדיקת תקינות הפלט.



טבלת בדיקות מסכמת

סוג הבדיקה	מס' בדיקה	תיאור קצר	עבר/נכשל	באגים
בדיקת פונקציונאליות	1	בדיקה שהמערכת פועלת כמצופה מימנה	עבר	0
בדיקת שימושיות	2	בדיקת נוחות השימוש וחווית המשתמש	עבר	0
בדיקת GUI	3	בדיקת הזדהות נכונה וניווט דפים תקין	עבר	0


:Test case

בדיקות פונקציונאליות:

בדיקה מס'	שלב	תוצאה צפויה	תוצאה בפועל	עבר/נכשל	באגים
1	חיפוש מתכון לפי מילת מפתח	הפונקציה תחזיר רשימת מתכונים מתאימים	קיבלתי רשימת מתכונים המתאימים למילת החיפוש	עבר	0
2	טעינת דף הלוח שנה	המתכונים של המשתמש יופיעו בתאריכים המתאימים	המתכונים הופיעו בהתאם	עבר	0
3	יצירת רשימת קניות	רשימת הקניות תופיע לפי טווח התאריכים התקין	רשימת הרכיבים הופיע כראוי	עבר	0

בדיקות שימושיות:

בדיקה מס'	שלב	תוצאה צפויה	תוצאה בפועל	עבר/נכשל	באגים
-----------	-----	-------------	-------------	----------	-------

0	עבר	המתכונים שנשמרו ע"י המשתמש הופיעו	כל המתכונים שנשמרו ע"י הנוכחי יופיעו	לחיצה על לחצן 	1
0	עבר	המצרכים הופיעו לפי המתכונים הקשורים	רשימת הקניות תופיע לפי המתכונים הרלוונטיים	לחיצה על לחצן Create Shopping List	2
0	עבר	דף המתכון נפתח בהתאם	דף המתכון המתאים יפתח בחלונית חדשה	לחיצה על מתכון	3

בדיקות GUI:

בדיקה מס'	שלב	תוצאה צפויה	תוצאה בפועל	עבר/נכשל	באגים
1	בדיקת תקינות של משתמש קיים	כניסה בשל הזדהות תקינה. הודעת שגיאה בעקבות הזדהות לא תקינה	אכן תוצאה מתאימה האתר אפשר כניסה רק למשתמש רשום	עבר	0
2	בדיקת תקינות מידע בהרשמת משתמש חדש	הרשמה עם מייל תקין וסיסמה מספיק מאובטחת	אכן המערכת אפשרה להירשם רק עם נתונים מתאימים	עבר	0

סיכום

אנו מאוד מרוצים מהתוצאה הסופית שעלתה על הציפיות שלנו, הן מבחינה עיצובית והן מבחינת פיצ'רים תכנותיים הקיימים בה.

במהלך הביצוע העלנו הרבה שאלות בנוגע לביצוע בפועל של התוכניות והניתוח הראשוני של הפרויקט, ביצענו שינויים לפי הצורך ושיפרנו את האלגוריתמים בכדי ליצור ממשק מהיר ונוח למשתמש.

לפני שניגשנו לביצוע של כל חלק בפרויקט העמקנו ולמדנו על ספריות חדשות באותו הנושא על מנת לשפר את נראות הפרויקט ולמצוא דרכים מגוונות ויעילות יותר בביצוע וכן בכדי שנלמד להשתמש בטכנולוגיות וספריות שונות בקוד שלא תמיד היה מוכר לנו.

הפרויקט היווה עבורנו התנסות משמעותית בכתיבת פרויקט Full Stack, בלמידה עצמית ושילוב ספריות לעיצוב ומידע בפרויקט וכן בשימוש בכלים ואלגוריתמים מתקדמים ומוכרים בשוק העבודה כיום.

אנו מרגישות שרכשנו כלים מעשיים מכתובת הפרויקט ובטוחות שהידע שרכשנו יסיע לנו רבות בשוק העבודה.

רכשנו ידע נרחב בשפת C# וב-Angular8, ידע נרחב ומעמיק במבני נתונים, בשימוש בWeb API ובקריאת מידע מDataBases, למדנו על שימוש בAPI חיצוני, שילוב הקוד שלו בקוד שלנו הן בצד שרת והן בצד לקוח, שליפת המידע, שימוש בו ושמירתו בצורה יעילה.

כלי נוסף שהשתמשנו בו היה GitHub, אומנם לא היה קל ללמוד על השימוש בו וקצת חששנו, אך לא ויתרנו ולבסוף למדנו וחוונו עד כמה נוח ויעיל השימוש בו והרווחנו ידע וניסיון בשימוש בכלי שימושי ומתקדם.

בנוסף למדנו על התמודדות עם פרויקט בהיקף גדול, למדנו על סדר העבודה בצורה נכונה, על חשיבות הניתוח והמיקוד בשלב הראשוני, תכנון מוקדם, נכון וטוב יחסוך לנו בעיות בעת הפיתוח. וכן על חשיבות הצבת יעדים וחלוקת משימות בצורה מסודרת בכל שלב בתהליך על מנת למזער קונפליקטים.

היו גם אתגרים וחששות, בעיקר מהדברים שהיינו צריכות ללמוד לבד כמו השימוש בAPI למתכונים, בדקנו וחקרנו על מגוון רחב של API הקיימים בשוק, התייעצנו רבות והגענו להחלטה שהייתה הנכונה ביותר.

נתקלנו אף בקשיים במהלך הפיתוח כמו הקושי בשימוש בGitHub ולא תמיד הכל הלך חלק, למדנו לנסות שוב ולפעמים לשנות כיוון לגמרי עד שלבסוף הגענו לתוצאה הרצויה.

מקורות מידע

[/https://angular.io](https://angular.io)

[/https://www.w3schools.com](https://www.w3schools.com)

[/https://stackoverflow.com](https://stackoverflow.com)

<https://spoonacular.com/food-api>

<https://ej2.syncfusion.com/home/angular.html>