רצינו להגיד 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 עיננו 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.

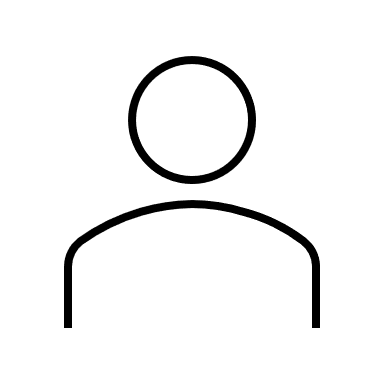
האתר **Quili** בא לנהל לנו את התכנון היומיומי של ניהול המטבח שלנו, החל מתכנון ארוחות קבועות או חד פעמיות ועד לניהול רשימת הקניות שלנו בצורה נוחה והכי חשוב- מסודרת ויעילה יותר.

שם האתר **Quili** -Quick list רשימה מהירה משקף את הרעיון שעומד מאחורי הפרויקט- בנית 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.

אופן זרימת המידע למערכת:

משתמש- לקוח:

* הרשמה למערכת
* כניסה למערכת

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. **Bl (Business Layer)** שכבה זו כוללת את הלוגיקה בפרויקט.
3. **Entities** שכבה זו ממירה משתנים מסוג הDB ל Entitiesולהפך.
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 בAPI חיצוני

מראה הSolution של הפרויקט:

צילום מסך

תיאור קוד פונקציות ומחלקות

Client Side:

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 ימי שימוש בספריות לעיצוב ולמידה מקוונת של פיצ'רים נוספים בשפה.

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

כל Component מורכב מ:

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

Classes – קבצים המגדירים לנו תכונות לאובייקטים מסוימים.

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

**דוגמא ל Component**

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

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

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 ב controllerמתממשקת לסביבת הclient.

צילומי מסך

**שכבת ה- Bl (Business Layer)**

**שכבת הEntities:**

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

**שכבת הDal (Data Access Layer):**

בשכבה זו אנו מתחברים לDataBase (מסד הנתונים).

במחלקת הDal נמצאות כל הטבלאות הקיימות בDB כולל קשרי הגומלין שלהן.

טבלאות אלו הינן הקרובות ביותר לDataBase ומייצגות אותו, הן המגשרות לבין הDB לBl.

**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 הEntity Framework:**

**צילום מסך**

**מבנה 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, ב Classשל הרבים יוצר מופע מסוג הClass של היחיד וב Class של היחיד יוצר אוסף מסוג הClass של הרבים.
* טבלאות בקשר של רבים לרבים- יוצר Class ובו אוספים של ה Classes המקושרים.

סיכום

אנו מאוד מרוצות 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 מכתיבת הפרויקט ובטוחות שהידע שרכשנו יסיע לנו רבות בשוק העבודה.

רכשנו ידע נרחב בשפת בC# וב- Angular8, ידע נרחב ומעמיק במבני נתונים, בשימוש בWeb API ובקריאת מידע מDataBase, 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 הGitHub, אומנם לא היה קל ללמוד על השימוש בו וקצת חששנו, אך לא ויתרנו ולבסוף למדנו וחווינו 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 כיון לגמרי עד שלבסוף הגענו לתוצאה הרצויה.