# 

**המכון הממשלתי להכשרה בטכנולוגיה ובמדע**

# דף כריכה

**פרויקט גמר לתואר הנדסאי מחשבים**



מגישות:

שם המנחה:

תאריך ההגשה: 2022

# דף שער

# הצעה לפרויקט גמר

**הצעה לפרויקט גמר**

1. **פרטי המכללה**

שם המכללה: סמינר הרב וולף .

מגמה: הנדסאות תוכנה .

מסלול: הנדסאות .

תאריך:18/60/10

1. **פרטי הסטודנט והמנחה**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **שנת סיום**  **הלימודים** | **טלפון** | **כתובת** | **ת.ז.** | **שם התלמידה** |
|  |  |  |  |  |
|  |  |  |  |  |

**שם המנחה:**

1. **שם הפרויקט: YourDrive**
2. **רקע**
   1. **תיאור ורקע כללי**  
      YourDrive היא פלטפורמת שיתוף מסמכים ותיקיות. האתר מאפשר להעלות מספר קבצים ותיקיות ולשתף עם משתמשים ללא הגבלה. אתה יכול להעתיק ולהעביר תיקיות או מסמכים בתיקייה אחרת בכונן. אתה יכול להוריד מסמכים ותיקיות בכונן וגם אתה יכול להציג מסמכים בתצוגה מקדימה. המשתמש יכול לכתוב הערות על המסמך וגם לשלוח מסמך באמצעות דואר אלקטרוני.
   2. **מטרות המערכת**  
      המערכת תנהל את מסמכים שלך בצורה יעילה וחכמה.
3. **סקירת מצב קיים בשוק**קימות הרבה תוכנות או אתרים לארגון קבצים אך בכל מערכת או אתר ישנו חסרון אין קיבוץ של כל הצרכים באתר אחד
4. **מה הפרויקט אמור לחדש או לשפר**הפרויקט ינהל מסמכים בצורה חכמה ומוצפנת.
5. **דרישות מערכת ופונקציונאליות**
   1. **דרישות מערכת**צד השרת   
      יועמד על שרת IIS המאפשר גלישה אנונימית.

SQL SERVER כולל מנגנוני התאוששות מקריסה.  
נפח דיסק פנוי של 30GB וזיכרון מומלץ 16GB.

צד הלקוח   
יוכל לפעול על מכשיר או מחשב אישי המריץ דפדפן google chrome (תיתכן תמיכה בדפדפנים נוספים), רוחב פס 100MBPS לפחות.

* 1. **דרישות פונקציונאליות**המערכת תאפשר לגולשים  
     הזדהות

שמירת מסמכים לפי תיקיות ורמת עדיפות

חיפוש מהיר בקבצים ותיקיות  
צפייה במסמך וגרסאות הקובץ

אפשרות לשחזר קובץ לגרסה קודמת  
לעלות מסמך.

לתת הרשאות למסמך

שליחת מייל עם מסמך או תיקייה מצורפים

אפשרות להוסיף הערות לקובץ

המערכת תאפשר למנהל

הגדרת הרשאות למשתמשי האתר

אפשרות חיבור לכל משתמש דרך לוח ניהול

**בעיות צפויות במהלך הפיתוח ופתרונות**

* 1. **תיאור הבעיות**
     1. איך נגן על המסמכים.
     2. משתמש מנסה לעלות קובץ שהפורמט שלו לא נתמך
  2. **פתרונות אפשריים**
     1. לבעיה 1:
* נתמוך בהצפנת מסמך בצורת base64 וsha256.
* לא נבצע הצפנה רק נתינת הרשאות צפייה

הפתרון הנבחר: גם הצפנה וגם נתינת הרשאות כדי שהאתר יהיה ברמת אבטחה גבוהה.

* + 1. לבעיה 2
* נציג הודעת שגיאה על כך שהאתר לא תומך בפורמט.
* תמיכה בפורמטים מרובים – לא יהיה מצב שקובץ לא נתמך

הפתרון הנבחר: תמיכה בפורמטים מרובים + הצגת הודעה במקרה שיש קובץ שלא נתמך.

1. **פתרון טכנולוגי נבחר**:
   1. **טופולוגית הפתרון**  
      המערכת תיושם ברשת האינטרנט, ותהיה מורכבת משרת המנהל מסד נתונים יחסי של SQL SERVER, וממשק משתמש בצד הלקוח. התקשורת ביניהם תתבצע באמצעות WEB API.
   2. **טכנולוגיות בשימוש**  
      Web-API   
      השרת יממש Web API כך שקליינטים יוכלו לפנות אליו ולקבל מידע. התשתית שנשתמש בה לצורך כך היא .NET Core Web API  
      ANGULAR 12  
      Angular היא תשתית תוכנה ליישומי רשת בקוד פתוח המתוחזקת ע"י גוגל.  לAngular יש קהילה הולכת וגדלה של מפתחים שמפתחים את התשתית באופן שוטף ומשמשת בעיקר לפיתוח של אתרים דינמיים.
   3. **שפות הפיתוח**  
      השרת ייכתב בשפת C# שהיא שפה ממשפחת ה NET. , שפת object Oriented, מיועדת בעיקר למטרות פיתוח תוכנות לסביבת Windows של Microsoft וגם לפיתוח צד שרת לאתרי אינטרנט.  
      צד הלקוח ייכתב בשפת TYPESCRIPT השימושית בAngular.  
      בחרנו בשפות אלו שהן פופולריות, שימושיות, ונוחות לפיתוח.
   4. **תיאור הארכיטקטורה הנבחרת**שרת SQL Server יאחסן את הנתונים.

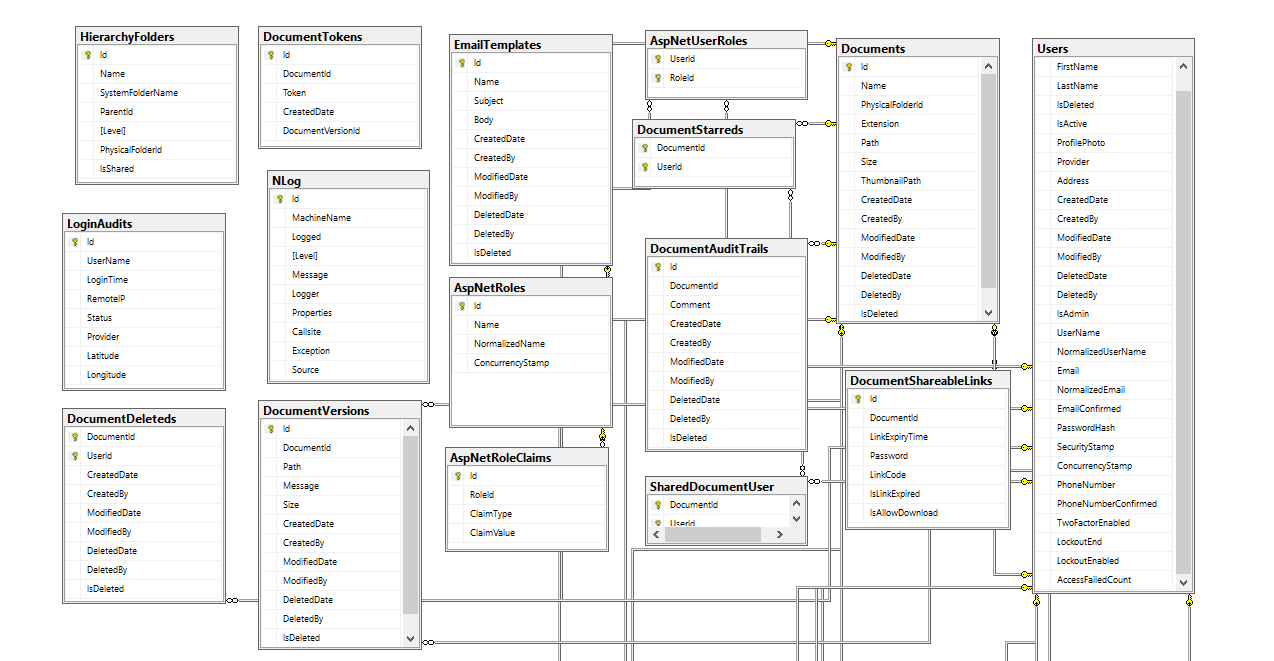
שכבת ניהול הנתונים – Data Layer - תהיה אחראית לשליפת הנתונים הנדרשים ולעדכונם.

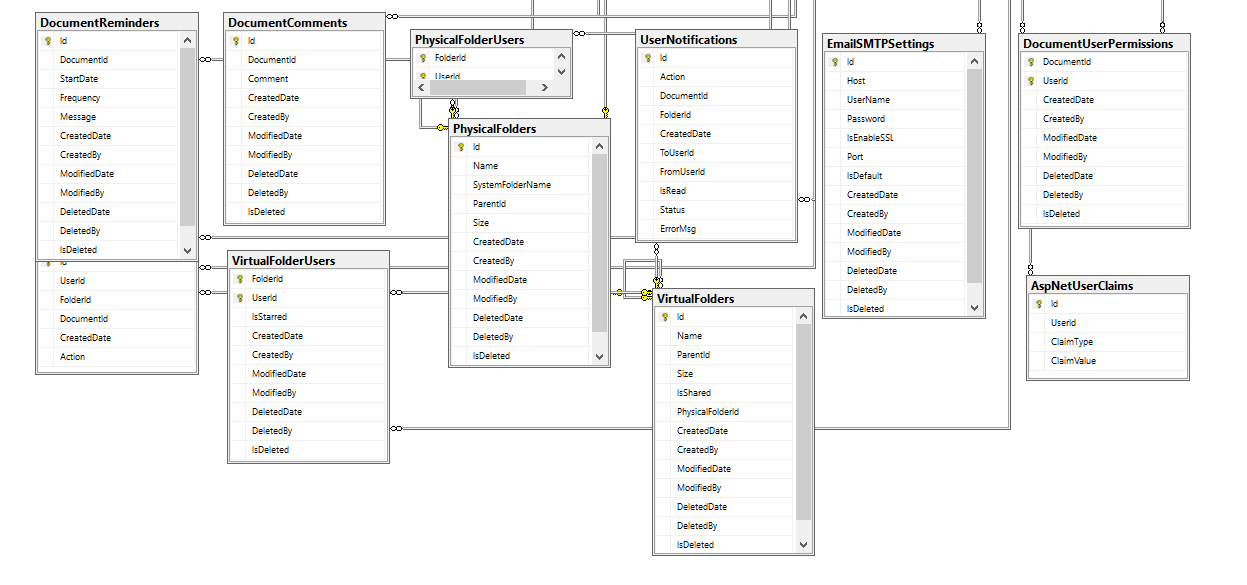
השכבה הלוגית – Business Layer – תנהל את הלוגיקה הנדרשת.

תשתית Web-API תאפשר תקשורת בין השרת ללקוח.

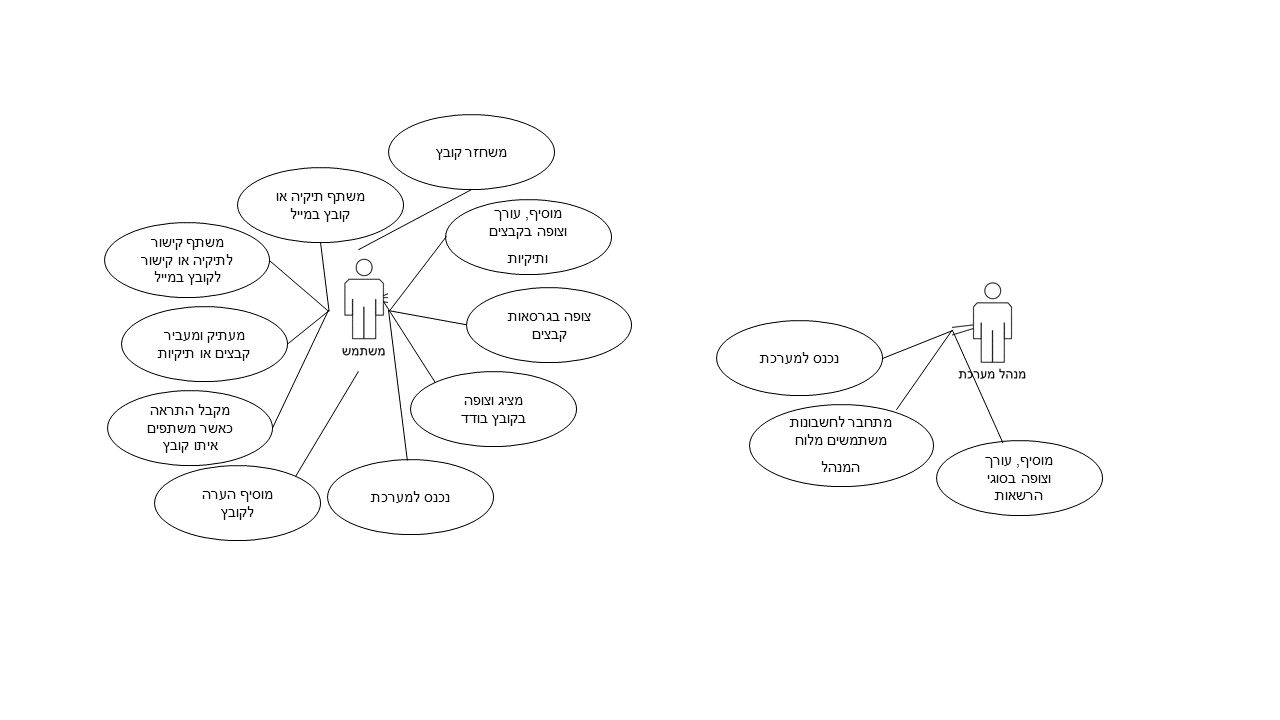
ממשק צד הלקוח בAngular יפנה לWeb-API לצורך קבלת המידע הנדרש או לצורך עדכונו ויציג את דפי האפליקציה למשתמש.

* 1. **חלוקה לתוכניות ומודולים**חלק א': מכלול תוכנה שרתי, שינהל את מרכיבי הניהול העסקי של המערכת.  
     חלק ב': מכלול הלקוח, שיחשוף ממשק משתמש מעל הweb
  2. **סביבת השרת**שרת IIS יארח את הAPI של צד השרת.
  3. **ממשק המשתמש –GUI**בטכנולוגיית Angular ניצור דפי אינטרנט שיוצגו על גבי דפדפן chrome (תיתכן תמיכה בדפדפנים נוספים).
  4. **ממשקים למערכות אחרות**  
     התחברות לשרת gmail כדי לשלוח מיילים.
  5. **שימוש בחבילות תוכנה**במהלך הפיתוח נשתמש בNPM להתקנת חבילות תוכנה נוספות באם תידרשנה.

1. **שימוש במבני נתונים וארגון קבצים:**
   1. הישויות העיקריות במערכת: משתמשים, מסמכים, הרשאות, הרשאות למסמך, התראות, תיקיות, גרסאות למסמך.  
      



* 1. הנתונים יאוחסנו במסד נתונים יחסי של SQL Server
  2. לצורך התאוששות מקריסה יופעלו מנגנוני הגיבוי והשחזור של SQL Server

1. **תרשימי מערכת מרכזיים**
   1. **Use cases**
2. **תיאור המרכיב האלגוריתמי – חישובי**.
   1. נתוני הקבצים וחילוקם לתיקיות ישמרו בdb וניצור מנתונים האלו סטיסטיקות.
3. **אבטחת מידע**  
   לצורך אבטחת הנתונים הרגישים במערכת נבנה מערכת הרשאות, שתדאג לכך שהמידע הרגיש לא יוצג בפני משתמשים בלתי מורשים. כמו כן נסתמך על תשתית האבטחה של שרת IIS מאובטח וכן נצפין את הקבצים השמורים במערכת.
4. **משאבים הנדרשים לפרויקט**
   1. היקף שעות העבודה כ700 שעות, שיחולקו בין המגישות בצורה מאוזנת.
   2. ציוד נדרש: מחשב PC עם זיכרון RAM של 8GB, תשתית רשת המאפשרת שליחת הודעות דואר אלקטרוני.
   3. תוכנות נדרשות:  
      visual studio 2019

Visual Studio Code  
SQL Server

* 1. ידע חדש שנדרש ללמוד לצורך ביצוע הפרויקט:  
     העמקת הידע וההכרות עם הטכנולוגיות שבשימוש הפרויקט.
  2. ספרות ומקורות מידע  
     https://angular.io  
     https://msdn.microsoft.com

https://stackoverflow.com

1. **תכנית עבודה ושלבים לביצוע הפרויקט**התכנון והפיתוח החל מחודש אלול תשפ"ב:
   1. תכנון פונקציונאליות המערכת (כולל ציורי מסכים) – כשלושה שבועות.
   2. יצירת תרשימי מערכת- כשבועיים.
   3. תכנון בסיס הנתונים- כשבועיים.
   4. כתיבת הצעת הפרויקט- כשבוע.
   5. פיתוח צד השרת- כעשרה שבועות.
   6. פיתוח צד הלקוח- כעשרה שבועות.
   7. כתיבת ספר הפרויקט- לאורך כל תהליך הפיתוח.
2. **תכנון הבדיקות שיבוצעו**
   1. בדיקות תהליכיות ברמת משתמש – Full Flow

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מספר בדיקה | מהות הבדיקה | אופן ביצוע הבדיקה | תוצאה רצויה | תוצאה בפועל | עבר או נכשל |
| 1 | הזדהות | הזנת שם וסיסמה, והקשה על אישור | כניסה לדף הראשי של האפליקציה |  |  |
| 2 | הוספת תיקייה | הוספת תיקייה בחלונית הקבצים | נתוני התיקייה נשמרים |  |  |
| 3 | העלאת מסמך | כניסת לעמוד העלאת מסמך | המסמך נשמר |  |  |
| 4 | הגדרת הרשאות למשתמשים | רק משתמש מנהל יכול להיכנס לדף הגדרת הרשאות | נתוני ההרשאות נשמרים |  |  |

* 1. בדיקות יחידה –Unit Test

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מספר בדיקה | מהות הבדיקה | אופן ביצוע הבדיקה | תוצאה רצויה | תוצאה בפועל | עבר או נכשל |
| 1 | שליפת מסמכים | קריאה לWEB-API לפונקציה GetDocument עם פרמטר סינון לפי תיקייה | רשימת מסמכים לפי תיקייה |  |  |
| 2 | בדיקה הרשאות. | קריאה לWEB-API לפונקציה AuthUser עם טוקן משתמש | הרשאות המשתמש. |  |  |
| 3 | שליפת מסמך ופרטיו. | קריאה לWEB-API לפונקציה DocDetails  עם טוקן משתמש | פרטי מסמך |  |  |
| 4 | שליחת התראה | קריאה לWEB-API לפונקציה  SendNotification | אישור שליחת התראה. |  |  |

1. **בקרת גרסאות**על בקרת הגרסאות אחראי שרת Microsoft windows server 2016 המפעיל shadow-copy.

מלבד זאת ניצור בס"ד אחת לשבוע או שבועיים, וכן לאחר התקדמות משמעותית בפיתוח, העתק נוסף של הפרויקט בגרסתו הנוכחית.

###### \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

###### חתימת הסטודנט חתימת המנחה האישי

# אישור הצעת הפרויקט ממה"ט

# תודות

ראשית לכל הננו רוצות להודות לריבונו של עולם שהביאנו עד הלום בידו הרחומה והאוהבת.

אנו אסירות תודה לכמות כה רבה של אנשים שליונו בסבלנות, מאור פנים ואוזן קשבת:

לרכזת המגמה שמנהלת את המגמה ביד רמה, במסירות ובאחריות מרובה שנים כה רבות .לכל צוות המורות במגמה על הכשרתנו המקצועית ועל סבלנותן הרבה , וכמובן למנחה היקרה שליוותה אותנו בסבלנות מרובה במשך התקופה ונתנה לנו הרגשה נעימה, עודדה ותמכה והייתה תמיד לציידנו.

וכן תודה ענקית שלוחה להורינו ולמשפחותינו שליוו אותנו במסירות ונתנו לנו עידוד רב. וכן לכל החברות במגמה ששאלו, התעניינו ועודדו...

שה' ישלם שכרכם!

# הצהרה

הצהרת התלמידים/ות

**אנו** \_\_\_\_\_\_\_\_\_\_\_\_ **ת.ז**.\_\_\_\_\_\_\_\_\_\_\_\_\_

**ו** \_\_\_\_\_\_\_\_\_\_\_\_\_**ת.ז**. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

החתומים/ות מטה, מצהירים/ות בזאת כי כל הפרויקט המוגש בספר זה, הינו פרי עבודתנו בלבד, על בסיס הנחייתה של המנחה \_\_\_\_\_\_ ותוך הסתמכות על לימודינו במכללה והמידע אשר רכשנו תוך כדי העבודה.

חתימת התלמידים/ות: \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

חתימת המנחה: \_\_\_\_\_\_\_\_\_\_\_

# תוכן העניינים

[דף כריכה I](#_Toc526859710)

[דף שער II](#_Toc526859711)

[הצעה לפרויקט גמר III](#_Toc526859712)

[אישור הצעת הפרויקט ממה"ט IV](#_Toc526859713)

[תודות 1](#_Toc526859714)

[הצהרה 2](#_Toc526859715)

[תוכן העניינים 3](#_Toc526859716)

[1. מבוא 4](#_Toc526859717)

[2. מדריך למתכנת: 5](#_Toc526859718)

[2.1. אסטרטגיות טכנולוגיות: 5](#_Toc526859719)

[2.2. תיאור מבנה הפרויקט: 5](#_Toc526859720)

[2.3. עקרונות הפיתוח: 6](#_Toc526859721)

[2.4. עקרונות התכנון/ הבניה/ הניתוח 8](#_Toc526859722)

[2.5. תרשימים 9](#_Toc526859723)

[2.6. מבני נתונים מאוחסן 12](#_Toc526859724)

[2.7. תוכן הפרויקט 12](#_Toc526859725)

[3. מדריך למשתמש: 24](#_Toc526859726)

[3.1. הוראות כלליות לשימוש באתר: 24](#_Toc526859727)

[3.2. מסכים: 25](#_Toc526859728)

[4. סיכום ומסקנות: 26](#_Toc526859729)

[5. נספחים: 37](#_Toc526859730)

[7. ביבליוגרפיה: 39](#_Toc526859731)

## מבוא

ניהול מסמכים אישי היא מערכת ניהול מסמכים מבוססת אינטרנט המתארחת בעצמה, שנועדה לעזור למשתמש לאחסן, לעקוב, לשנות ולנהל את מסמכיו בפלטפורמה מרכזית. התכונות כוללות אפשרות לעלות מספר קבצים ותיקיות ולשתף עם משתמשים ללא הגבלה, העתקת ולהעברת תיקיות או מסמכים מתיקייה לתיקייה אחרת בכונן, להוריד מסמכים ותיקיות בכונן, אפשרות להציג מסמכים בתצוגה מקדימה. המשתמש יכול לכתוב הערות על המסמך וגם לשלוח מסמך באמצעות דואר אלקטרוני.

התוכנה תכיל ממשקים שונים למשתמש:

* ממשקים למשתמש – המשתמש יוכל לעלות מסמכים, לשלוח התראות למשתמשים אחרים, לראות ולצפות ולשתף קבצים, המשתמש יכול להוסיף קטגוריות ולסווג את המסמכים תחת הקטגוריות השונות ולהוסיף הערות ועוד.
* ממשק למנהל האתר- המנהל יכול להוסיף ולשנות הרשאות וכן לצפות בכל תיקיות המשתמש.

מודל הפרויקט מחולק ל3 חלקים:

1. בנית classes מתאימים ויצירת השדות והפונקציות הנדרשות בתוכם.

יצירת controllers המנהלים את קריאות הclient ומתחברים לsql כדי לשמור את הנתונים

2. קוד Typescript הנכתב בפלטפורמת angular ובו שימוש בספריות רבות כsingular

ועוד ספריות רבות(כל הרשימה כתובה בנספחים)

ובאמצעים טכנולוגיים רבים

ובו גם חישובים אלגוריתמים שונים ותקשורת עם הserver

3. תצוגת html המוגשת כאתר בממשק חדשני ונח לשימוש.

**האתר מורכב מכמה מסכים מרכזיים:**

1. מסך login- המשתמש נכנס לאתר.
2. מסך folders- כאן יש אפשרות להוסיף תיקיות או קבצים לתיקיות.
3. מסך user setting - אזור אישי למשתמש לצורך שינוי ההגדרות האישיות.
4. מסך add document- הוספת מסמך חדש.
5. מסך add remind– הוספת הערות למסמך.
6. מסך histories המציג גרסאות שונות על הקבצים שבמערכת.

ועוד מסכי משנה

שלב אפיון וניתוח הפרויקט היווה משימה לא פשוטה ומרכזית בפיתוח המערכת, ודרש ידע רב ואיסוף רחב של מידע. משימת הפיתוח הייתה מורכבת ומאתגרת, ודרשה חשיבה יצירתית בראש גדול והרבה למידה עצמאית.

## מדריך למתכנת:

### אסטרטגיות טכנולוגיות:

בכתיבת פרויקט זה שמנו דגש רב על התאמה טכנולוגית מדויקת ושאיפה לטכנולוגיות חדשניות,

ויחד עם זאת כאלה שלא תפגענה באיכות הפרויקט ותתאמנה לדרישותיו.

צד שרת –server side כתבנו בשפת c# בפלטפורמת Asp.net Web API Core .

צד לקוח – client side כתבנו בשפות Html5, Typescript, SCSS בפלטפורמת .Angular

מסד הנתונים SqlServer

### תיאור מבנה הפרויקט:

#### מבנה הServer side:

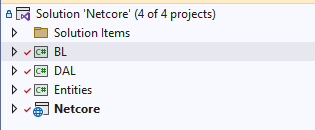
מבנה Server Side השרת נבנה בהתאם למודל השכבות וכולל את הפרויקטים הבאים:

א. DAL- פרויקט זה משמש כשמו Data Access Layer כאחראי על המידע, כלומר על מסד הנתונים. פרויקט זה מתחבר למסד הנתונים הנדרש לפרויקט (SqlServer)

ב. BLL- פרויקט זה מכיל את עיקר הלוגיקה של צד השרת, ומטפל בעיקר בשליפות ושמירת נתונים.

ג. WebApi- חשיפת API עימו מתממשק האתר, כגון: התחברות משתמש, שליפת נתונים, שמירת התראות לקבצים ועוד.

כל שכבה מכירה את השכבה שמעליה בלבד.



מודל שלושת השכבות

#### מבנה הClient side :

מבנה ה-client מורכב משלושה אלמנטים: דפי SCSS דפיHTML ודפי TS.

דפי ה- SCSS אחראיים על העיצוב והמראה הכללי,

דפי ה-HTML מהווים את התצוגה,

ודפי ה-TS כוללים בתוכם פונקציות לוגיות שונות, התחברות לשרתים ולספריות וכן שירותים שונים

ההתחברות לצד השרת ע"י קריאות Http

### עקרונות הפיתוח:

מאחורי קוד המערכת עומדים עקרונות תכנותיים רבים שחלקם לא היו מוכרים לנו. בסעיף זה ננסה

להבהיר את העקרונות העיקריים שעל פיהם פעלנו לאורך כתיבת הקוד.

**Sql**

היא מערכת לניהול בסיס נתונים במודל היחסי (RDBMS) של חברת מיקרוסופט., שפת הפיתוח שבאמצעותה מועברות הפקודות למערכת (על ידי כתיבת פקודות או שימוש בממשק גרפי ששולח פקודות באופן סמוי) היא Transact-SQL ‏(TSQL) שהיא מימוש תקן ANSI של שפת SQL משמשת לתשאול וטיפול בנתונים (DML), יצירת טבלאות והיחסים ביניהן (DDL) ותחזוקת המערכת תוך שימוש בתוכניות שירות שונות.

SQL הוא מסד נתונים בו נשמרים הנתונים בטבלאות בעלות קשרי גומלין, וניתן לשלוף ממנו מידע ע"י שאלתות SQL.

יש לו יכולות רבות נוספות כמו view, פרוצדורות, פונקציות, טרנזקציות ועוד.

זהו מסד הנתונים הנפוץ ביותר.

**ASP.net** **Web API**

Iמה זה API?

API הוא קיצור של .Interface Program Application במונחים פשוטים, זהו הממשק של פיסת

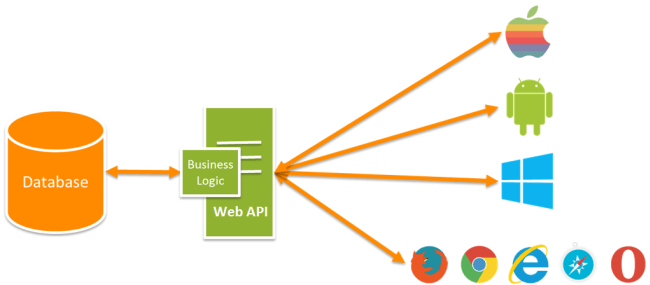
תוכנה עם העולם החיצוני, הקובע איך ישתמשו בה. אלה החוקים שנקבעו עבור האינטראקציה

שלה עם העולם הרחב, אלה יקבעו אילו חלקים של התוכנה יכולות לדבר עם תוכנות אחרות, ואיך

היא תגיב.

מה זה Web API ?

ASP.NET Web API הוא מסגרת המקלה על בניית שירותי HTTP המגיעים למגוון רחב של לקוחות, כולל דפדפנים והתקנים ניידים. ASP.NET Web API הוא פלטפורמה אידיאלית לבניית יישומים RESTful ב- NET Framework.



מה זה REST?

Rest( ראשי תיבות Transfer State Representational )היא סגנון ארכיטקטוני לכתיבת צד שרת.

התפיסה הארכיטקטונית ב- Rest היא תפיסת שרת-לקוח. תפיסה זו מחייבת קיום לקוח ושרת.

לקוחות יוזמים פניות המכילות בקשות לשרתים. השרת מעבד את הפנייה, ומחזיר תגובות

מתאימות.

בכל מצב נתון הלקוח יכול להיות בתהליך של שינוי מצב או במצב מנוחה (rest .(במצב של מנוחה

הלקוח יכול להיות באינטראקציה עם המשתמש, אבל אינו תופס משאבים בשרת. הלקוח שולח פניות

כאשר הוא מוכן לעבור למצב חדש. כאשר קיימת פנייה אחת או יותר שטרם הסתיים הטיפול בהן,

הלקוח נמצא במצב של מעבר ממצב למצב.

REST הוא לא פרוטוקול )כמו SOAP )אלא יותר קונבנציה שבה אנחנו משתמשים. Rest מבוסס ברב

המקרים על פרוטוקול HTTP ,למרות זאת REST היא ארכיטקטורה כללית הניתנת למימוש גם

בסביבות אחרות ולא רק תחת HTTP.

עבודה ב REST-מחייבת התחשבות באילוצים ובמגבלות של ארכיטקטורה זו, נפרט את חלקם:

• שרת-לקוח: עובד רק בתפיסת שרת לקוח, כאשר האחד אינו מושפע ממה שמתרחש באחר,

למעט המסרים העוברים ביניהם.

• Stateless :ההקשר (context (של הלקוח בפניה לשרת, אינו נשמר בשרת. מגבלה זו נועדה

לשפר את מדרגיות (Scalability (השרת.

כשאנו יוצרים שירות שממלא אחר הקונבנציות של REST אנחנו יוצרים שירות RESTful.

:Rest API - לסיכום

ממשק API של REST מגדיר קבוצה של פונקציות אשר מפתחים יכולים לבצע בקשות ולקבל תגובות

באמצעות פרוטוקול HTTP כגון GET ו- POST.

**Typescript**

היא שפת תכנות חינמית ומבוססת קוד פתוח המפותחת ומתוחזקת על ידי מיקרוסופט .היא מכילה

את קבוצת כל פקודות ותחביר JavaScript הפופולרית, כלומר כל קוד

JavaScriptהוא גם קוד Typescript תקין, ומוסיפה עליה טיפוסים סטטיים ותכנות מונחה עצמים

מבוסס מחלקות. בין התכונות שהיא מוסיפה: typing static ,תמיכה במחלקות ותמיכה במודולים

ובדקורטורים.

Typescript היא סופר-סט (superset) של JavaScript כלומר, הקוד מבוסס על JavaScript וחייב

לעבור קומפילציה (תרגום) ל JavaScript-כדי שהדפדפנים יבינו אותו מפני שדפדפנים

מבינים JavaScript בלבד. בסופו של דבר קוד Typescript מעובד לכדי קוד JavaScript ולכן ניתן

לומר "באחריות" שהשפה נתמכת בכל דפדפן שתומך ב- JavaScript.

קבצי Typescript יישמרו בסיומת ts.

### עקרונות התכנון/ הבניה/ הניתוח

הפרדת שכבות:

כל תוכנה/ אתר שנפגוש בעולם מבוססת על ארכיטקטורת שכבות הנקראת בשם Three Tier- -שלש שכבות במבנה שלBL-UI Application DAL-. זוהי תבנית עיצוב בסיסית שמגדירה הפרדת האפליקציה ל: שכבת נתונים, שכבת לוגיקה, ושכבת ממשק משתמש.

לתבנית עיצוב זו יתרונות רבים:

1.תחזוקה:

ניתן להחליף או לתקן מימוש פנימי של שכבה אחת בארכיטקטורה בלי לשנות שכבה אחרת.

2.נוחות פיתוח:

אדם אחד עובד על רכיב בתכנה, אדם אחר עובד על רכיב אחר, כל עוד שהחתימות זהות ניתן לשלב כוחות ולייעל זמני פיתוח.

3. בדיקות:

תקלה כלשהיא מבודדת בכל שכבה בנפרד, לדוגמא אם לא קיבלנו רשימת נתונים לתצוגה נבדוק קודם את שלב הבינים (שכבת ה BL) אם הנתונים שם תקינים נדע בוודאות שגם ברמת שכבת ה DAL הנתונים תקינים וכל שנותר הוא לפתור את התקלה ברמת ה UI .

4. שימוש חוזר:

נניח שהרכיבים BL+ DAL אהובים עלינו ועובדים היטב, ניתן להחליף את פלטפורמת UI לטכנולוגיה אחרת במינימום מאמץ.

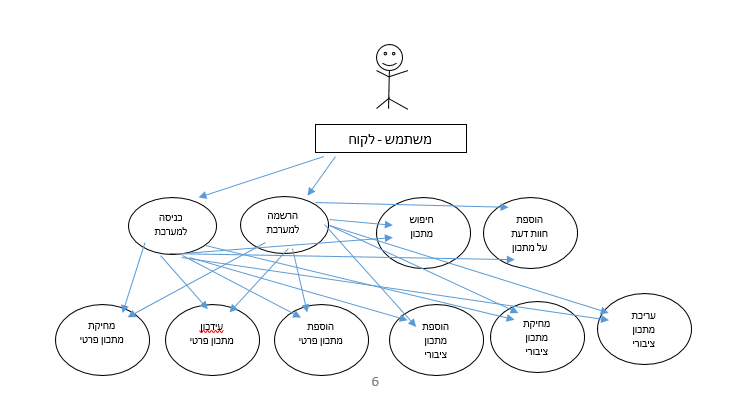
5. אבטחה:

נוח יותר להגדיר Interface API לשכבה מסוימת בלי לחשוף מבני נתונים Logic או Data שלא רלוונטי למשתמש, כמו גם חסימה בפני האקרים (רלוונטי יותר בטכנולוגיות Web ) ברמות שונות.

הפרויקט של צד שרת מכיל את שלושת השכבות הנ"ל.

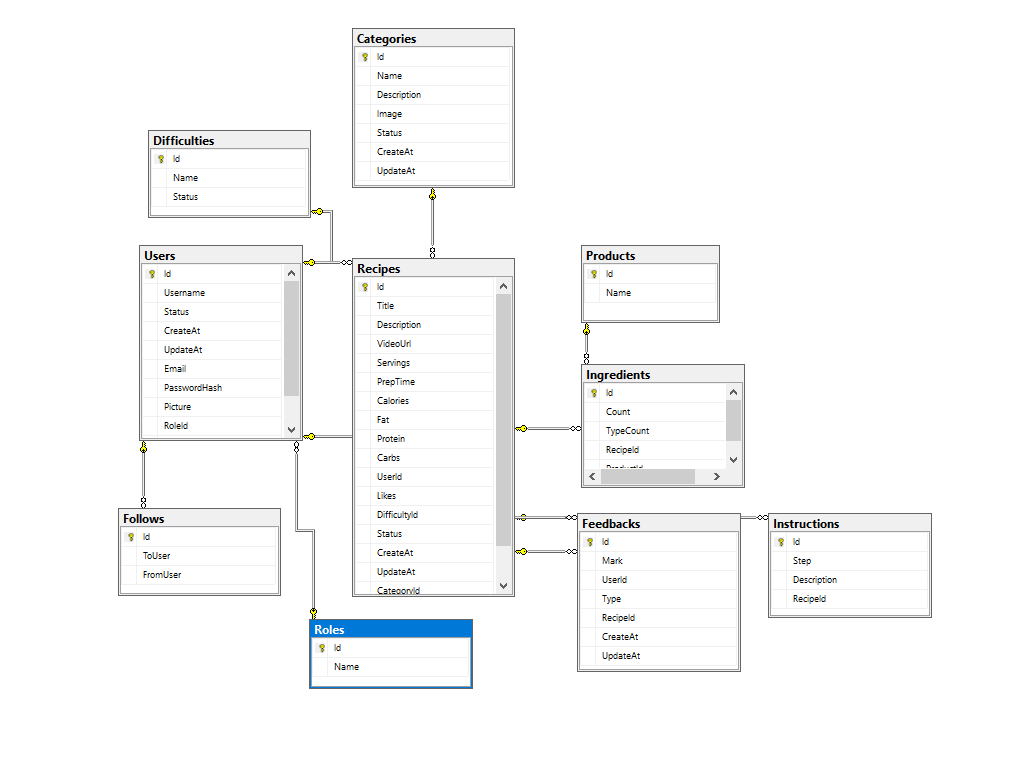
### תרשימים

#### תרשים Uml



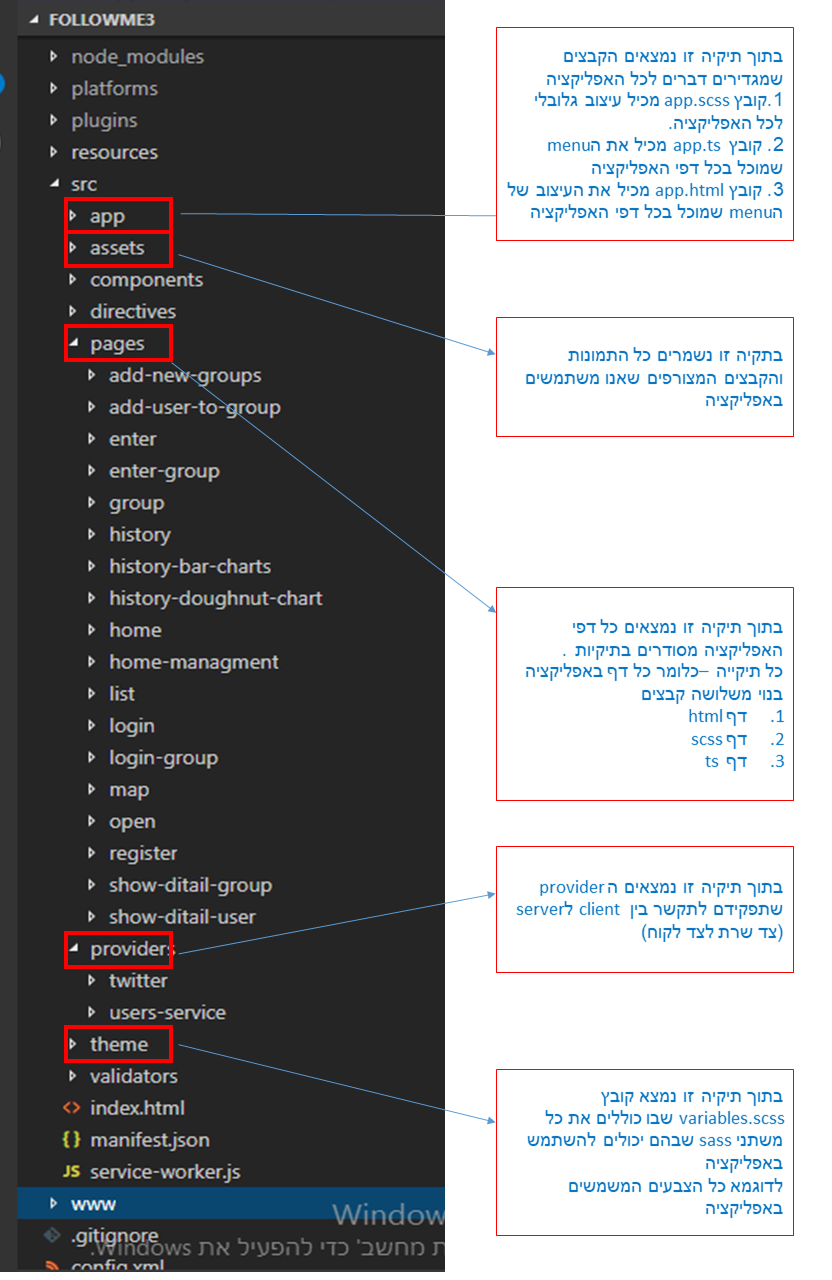
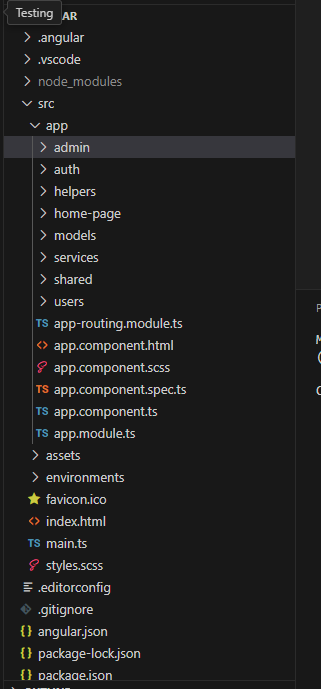
#### תרשים מראה מחלקות

* פרויקט הDal



#### מבנה קבצים ותיקיות

צד client :



בתוך תיקיה זו נמצאים כל דפי האתר מסודרים בתיקיות כל תיקיה- כלומר כל דף באתר בנוי משלושה קבצים

1. דף html
2. דף scss
3. דף ts

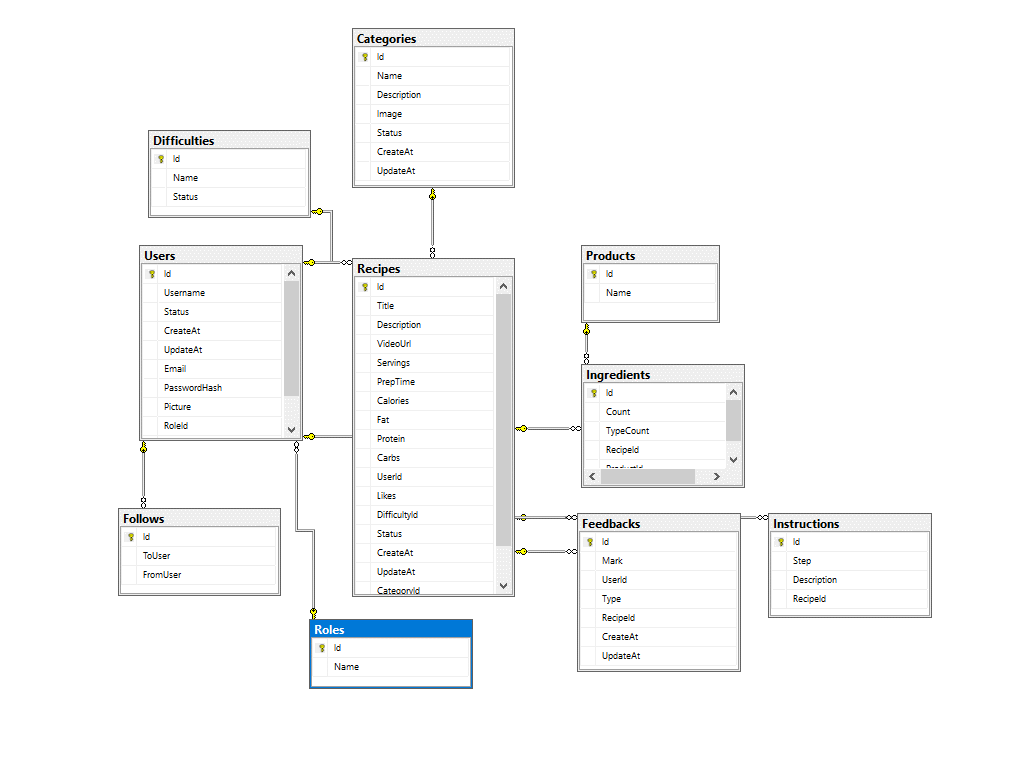
בתקיה זו נשמרים כל התמונות והקבצים המצורפים שאנו משתמשים באתר

בתוך תיקיה זו נמצאים הקבצים שמגדירים דברים לכל האתר

1. קובץ app.scss מכיל עיצוב גלובלי
2. קובץ app.html מכיל את הדף הראשון שעולה באתר שכל הדפים מוכלים בתוכו על ידי router-outlet

### מבני נתונים מאוחסן

#### מבנה Sql



### תוכן הפרויקט

#### תיאור מחלקות

מחלקות ב Dal -חלקי

**Category**-(קטגוריה)

public partial class Category

{

public int Id { get; set; }

public string? Name { get; set; }

public string? Description { get; set; }

public string? Image { get; set; }

public int? Status { get; set; }

public DateTime? CreateAt { get; set; }

public DateTime? UpdateAt { get; set; }

public virtual ICollection<Recipe> Recipes { get; set; } = new List<Recipe>();

}

**User**- (משתמש)

public partial class User

{

public int Id { get; set; }

public string Username { get; set; } = null!;

public int? Status { get; set; }

public DateTime CreateAt { get; set; }

public DateTime? UpdateAt { get; set; }

public string? Email { get; set; }

public byte[]? PasswordHash { get; set; }

public string? Picture { get; set; }

public int? RoleId { get; set; }

public string? FirstName { get; set; }

public byte[]? PasswordSalt { get; set; }

public virtual ICollection<Feedback> Feedbacks { get; set; } = new List<Feedback>();

public virtual ICollection<Follow> FollowFromUserNavigations { get; set; } = new List<Follow>();

public virtual ICollection<Follow> FollowToUserNavigations { get; set; } = new List<Follow>();

public virtual ICollection<Recipe> Recipes { get; set; } = new List<Recipe>();

public virtual Role? Role { get; set; }

}

**Difficuty**- (רמת מתכון)

public partial class Difficulty

{

public int Id { get; set; }

public string Name { get; set; } = null!;

public int? Status { get; set; }

public virtual ICollection<Recipe> Recipes { get; set; } = new List<Recipe>();

}

**Feedback** (פידבק)

public partial class Feedback

{

public int Id { get; set; }

public string? Mark { get; set; }

public int? UserId { get; set; }

public int? Type { get; set; }

public int? RecipeId { get; set; }

public DateTime? CreateAt { get; set; }

public DateTime? UpdateAt { get; set; }

public virtual Recipe? Recipe { get; set; }

public virtual User? User { get; set; }

}

**Follow** (עוקב אחרי משתמש)

public partial class Follow

{

public int Id { get; set; }

public int? ToUser { get; set; }

public int? FromUser { get; set; }

public virtual User? FromUserNavigation { get; set; }

public virtual User? ToUserNavigation { get; set; }

}

**Ingredient** (רכיבי מתכון)

public partial class Ingredient

{

public int Id { get; set; }

public decimal? Count { get; set; }

public int? TypeCount { get; set; }

public int? RecipeId { get; set; }

public int? ProductId { get; set; }

public virtual Product? Product { get; set; }

public virtual Recipe? Recipe { get; set; }

}

**Controllers -דוגמאות:**

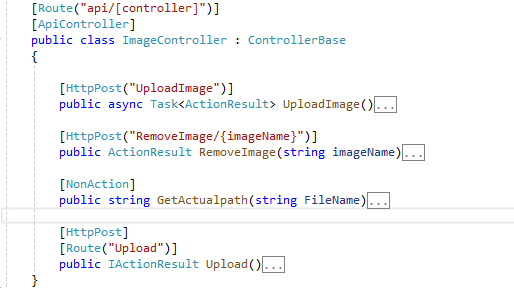
**AuthController** (הרשאות)



**CategoryController** (קטגריות)



**ImageController** (תמונות)

****

#### תאור הפנקציות

**פונקציות בצד שרת:**

1. פונקציות **UploadImage**

פונקציה זו מעלה קבצים ושומרת אותם במחשב ובדטהבייס.

[HttpPost("UploadImage")]

public async Task<ActionResult> UploadImage()

{

bool Result = false;

var Files = Request.Form.Files;

foreach (IFormFile source in Files)

{

string FileName = source.FileName;

string imagepath = GetActualpath(FileName);

try

{

if (System.IO.File.Exists(imagepath))

System.IO.File.Delete(imagepath);

using (FileStream stream = System.IO.File.Create(imagepath))

{

await source.CopyToAsync(stream);

Result = true;

}

}

catch (Exception ex)

{

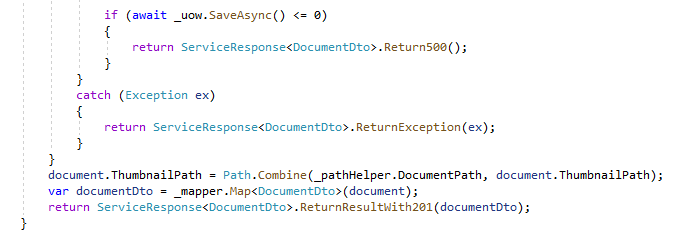
throw ex;

}

}

return Ok(Result);

}



1. פונקצית **UpdateRecipe**

פונקציה שמבצעת עדכון מתכון.

public async Task<ServiceResponse<RecipeDTO>> Update(RecipeDTO recUpdate)

{

var serviceResponse = new ServiceResponse<RecipeDTO>();

try

{

var cat =

await \_context.Recipes

.FirstOrDefaultAsync(c => c.Id == recUpdate.Id);

if (cat is null)

throw new Exception($"Recipe with Id '{recUpdate.Id}' not found.");

cat.UpdateAt = DateTime.Now;

cat.ImageUrl = recUpdate.ImageUrl;

if (recUpdate.ImageUrl.Contains(";base64"))

{

if (!string.IsNullOrEmpty(cat.ImageUrl))

GlobalService.RemoveImage(cat.ImageUrl);

cat.ImageUrl = GlobalService.SaveImage(recUpdate.ImageUrl, "category");

}

cat.Status = recUpdate.Status;

cat.Carbs = recUpdate.Carbs;

cat.VideoUrl = recUpdate.VideoUrl;

cat.Fat = recUpdate.Fat;

cat.Calories = recUpdate.Calories;

cat.Category = \_context.Categories.FirstOrDefault(p => p.Id == recUpdate.CategoryId);

cat.Servings = recUpdate.Servings;

cat.Description = recUpdate.Description;

cat.Difficulty = \_context.Difficulties.FirstOrDefault(p => p.Id == recUpdate.DifficultyId);

cat.PrepTime = recUpdate.PrepTime;

cat.Protein = recUpdate.Protein;

cat.Title = recUpdate.Title;

if (recUpdate.Instructions != null && recUpdate.Instructions.Count() > 0)

{

for (int i = 0; i < recUpdate.Instructions.Count(); i++)

{

recUpdate.Instructions.ToArray()[i].Step = i + 1;

}

}

var addIn = recUpdate.Instructions?.Where(p => p.Id == 0).ToList();

var intDB = cat.Instructions.ToList();

cat.Instructions = cat.Instructions.Where(p => recUpdate.Instructions?.Any(pp => pp.Id == p.Id) == true).ToList();

for (int i = 0; i < cat.Instructions.Count(); i++)

{

var up = recUpdate.Instructions?.First(p => p.Id == intDB[i].Id);

intDB[i].Step = up.Step;

intDB[i].Description = up.Description;

}

if (addIn?.Count() > 0)

{

addIn.ForEach(p =>

{

cat.Instructions.Add(new Instruction() { Description = p.Description, Step = p.Step });

});

}

var addInIngredients = recUpdate.Ingredients?.Where(p => p.Id == 0).ToList();

var intDBIngredients = cat.Ingredients.ToList();

cat.Ingredients = cat.Ingredients.Where(p => recUpdate.Ingredients?.Any(pp => pp.Id == p.Id) == true).ToList();

for (int i = 0; i < cat.Ingredients.Count(); i++)

{

var up = recUpdate.Ingredients?.First(p => p.Id == intDB[i].Id);

//intDBIngredients[i].Product.Name = up.Product.Name;

intDBIngredients[i].Count = up.Count;

intDBIngredients[i].TypeCount = up.TypeCount;

}

if (addInIngredients?.Count() > 0)

{

addInIngredients.ForEach(p =>

{

cat.Ingredients.Add(new Ingredient() { Count = p.Count, Product = \_context.Products.First(pp => pp.Id == p.ProductId), TypeCount = p.TypeCount });

});

}

await \_context.SaveChangesAsync();

serviceResponse.Data = \_mapper.Map<RecipeDTO>(cat);

}

catch (Exception ex)

{

serviceResponse.Success = false;

serviceResponse.Message = ex.Message;

}

return serviceResponse;

}

1. פונקציה **SetPictureByGender**

פונקציה לעדכון תמונת משתמש לפי שם משתמש מזהה את המין ושם תמונה דיפולטיבית

public static UserDTO SetPictureByGender(UserDTO user)

{

if (string.IsNullOrEmpty(user.Picture))

{

using (HttpClient client = new HttpClient())

{

try

{

HttpResponseMessage response = client.GetAsync("https://api.genderize.io/?name=" + user.FirstName).Result;

string responseBody = response.Content.ReadAsStringAsync().Result;

var nameGender = JsonConvert.DeserializeObject<NameGender>(responseBody, new JsonSerializerSettings

{

ContractResolver = new CamelCasePropertyNamesContractResolver()

});

if (nameGender != null)

{

if (nameGender.Gender == Gender.Female)

user.Picture = "female\_defualt.png";

else user.Picture = "male\_default.jpg";

}

}

catch (Exception ex)

{

}

}

}

return user;

}

**פונקציות בצד לקוח:**

בדף AddRecipe

1. פונקציה שמוסיפה מתכון

  saveRecipe() {

    this.recipe.ingredients.forEach((p) => {

      if (typeof p.product == 'string') {

        var name = p.product as string;

        p.product = new Product();

        p.product.name = name;

      }

    });

    this.recipe.userId = this.storageService.getUser().id;

    this.recipeService.add(this.recipe).subscribe(

      (res) => {

        if (res) this.router.navigate(['admin/recipes']);

        else this.toastr.error('Fail to save new recipe', 'Fail save!');

      },

      (err) => {

        this.toastr.error('Fail to save new recipe', 'Fail save!');

      }

    );

  }

1. למחיקת רמת קושי של מתכון

  deleteDifficulty() {

    Swal.fire({

      title: 'Are you sure?',

      text: 'You will not be able to recover this item!',

      icon: 'error',

      showCancelButton: true,

      confirmButtonText: 'YES, DELETE IT!',

      cancelButtonText: 'CANCEL',

    }).then((result) => {

      if (result.value) {

        this.diffService.delete(this.difficulty.id).subscribe(res => {

          if (!res) {

            this.toastr.error('Fail to delete difficulty', 'Fail delete!');

          }

          else {

            this.router.navigate(["admin/difficulties"]);

          }

        })

      } else if (result.dismiss === Swal.DismissReason.cancel) {

      }

    });

  }

  }

1. פונקציה לשליפת כל הקטגוריות

  getCategories() {

    this.categoryService.getAll().subscribe(res => {

      this.categories = res;

      this.cdRef.detectChanges();

    })

  }

## מדריך למשתמש:

### הוראות כלליות לשימוש באתר:

#### מדריך למשתמשים

בעת כניסת משתמש לאתר אחרי דף כניסה בו יכניס את פרטיו יעבור לדף הבית בהתאמה

#### מדריך למשתמש:

בעת כניסה לאתר אם המשתמש רשום הוא מגיע לדף הבית, ואם לא משתמש באתר לחיפוש ומציאת מתכון המותאם עבורו

המשתמש הרשום יכול לצפות בכל המתכונים המשויכים אליו וכן לעדכן ולמחוק אותם, המשתמש יכול לבצע מעקב אחרי משתמש אחר ולקבל התראה כאשר משתמש זה מעלה מתכון חדש, המשתמש יכול להוסיף פדבק על מתכון שאהב או מתכון לא מוצלח .

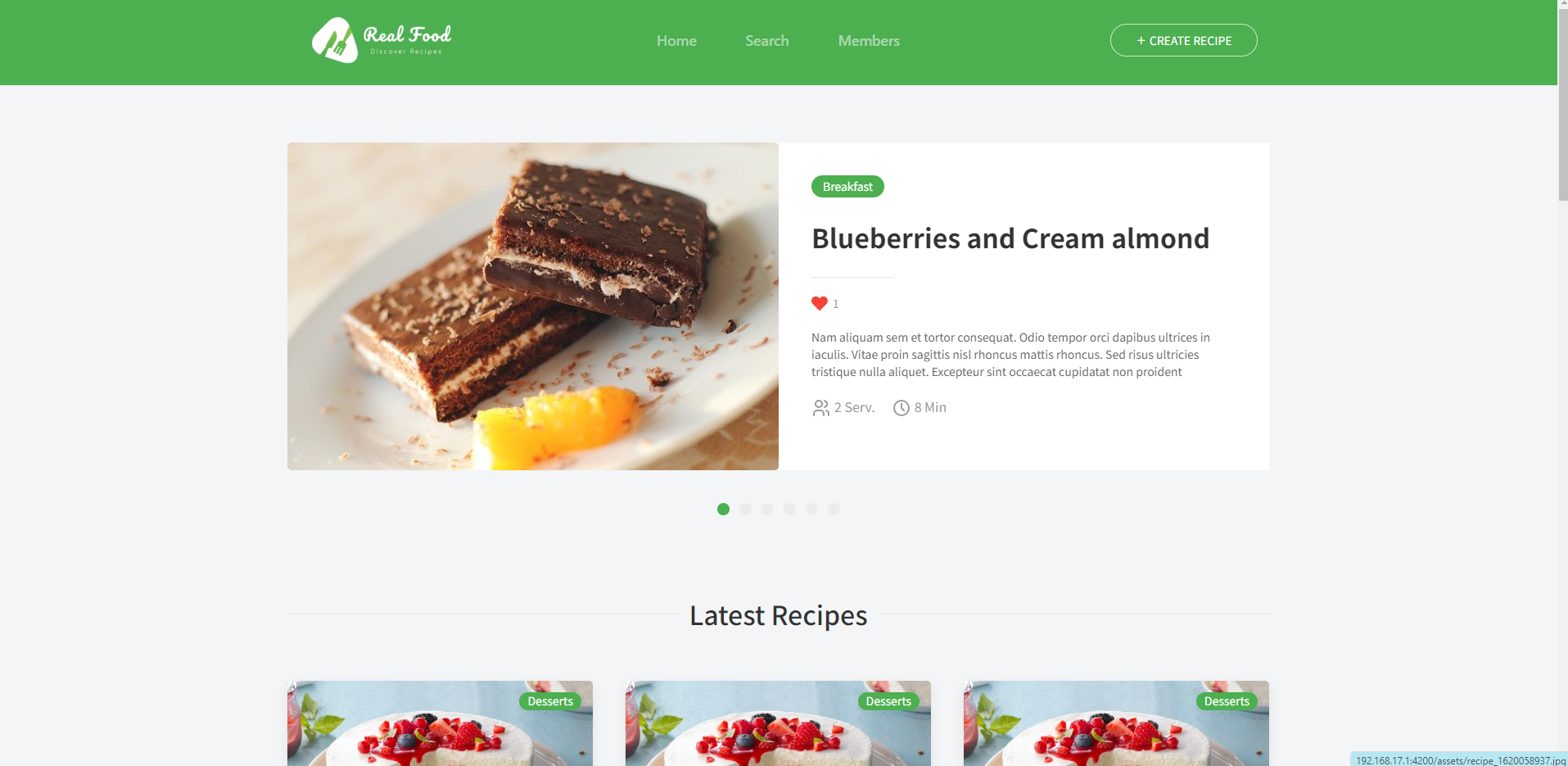
#### מדריך למנהל:

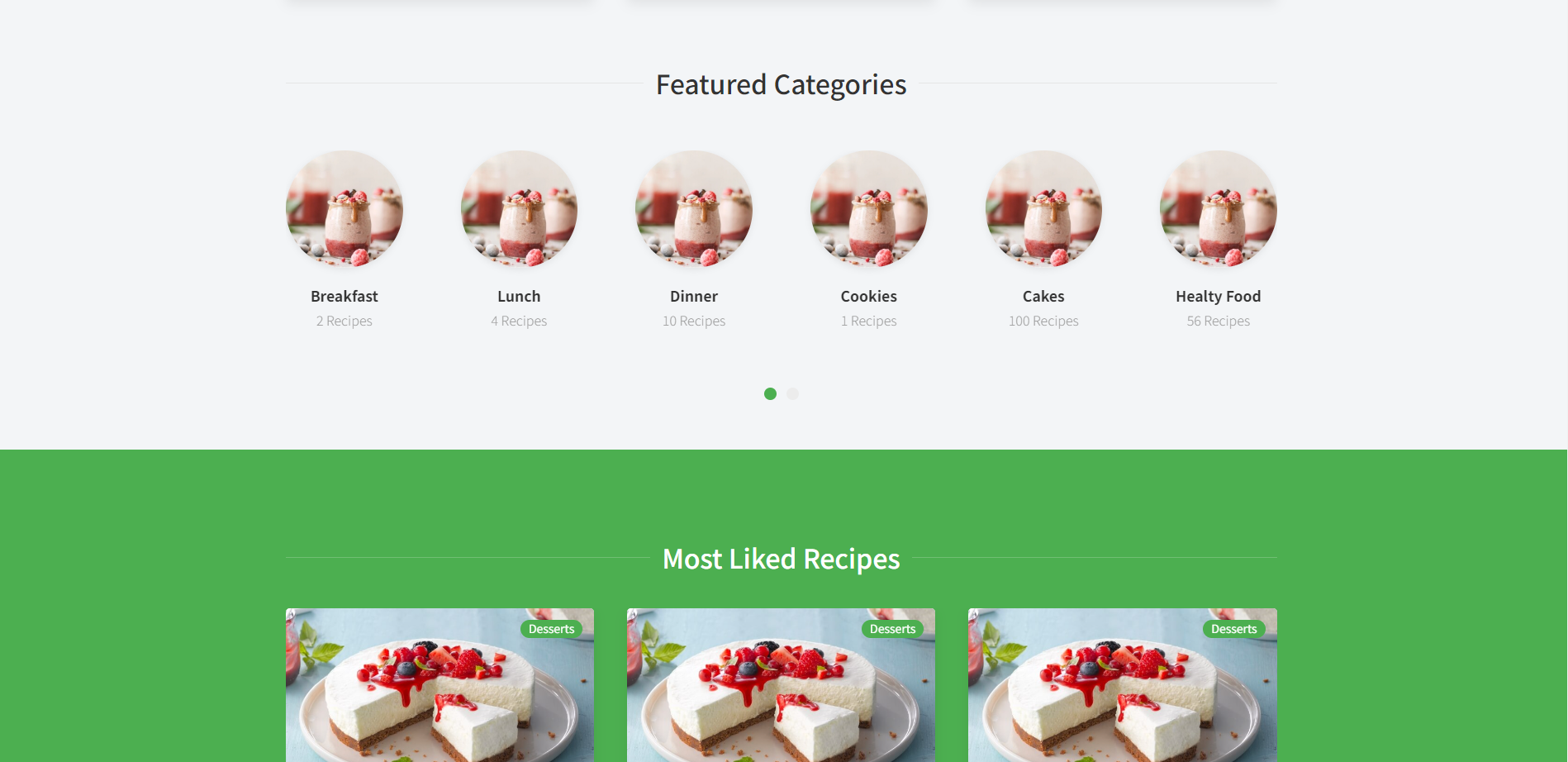
מנהל בנוסף לאפשריות שיש למשתמש- יש לו אפשריות נוספות:

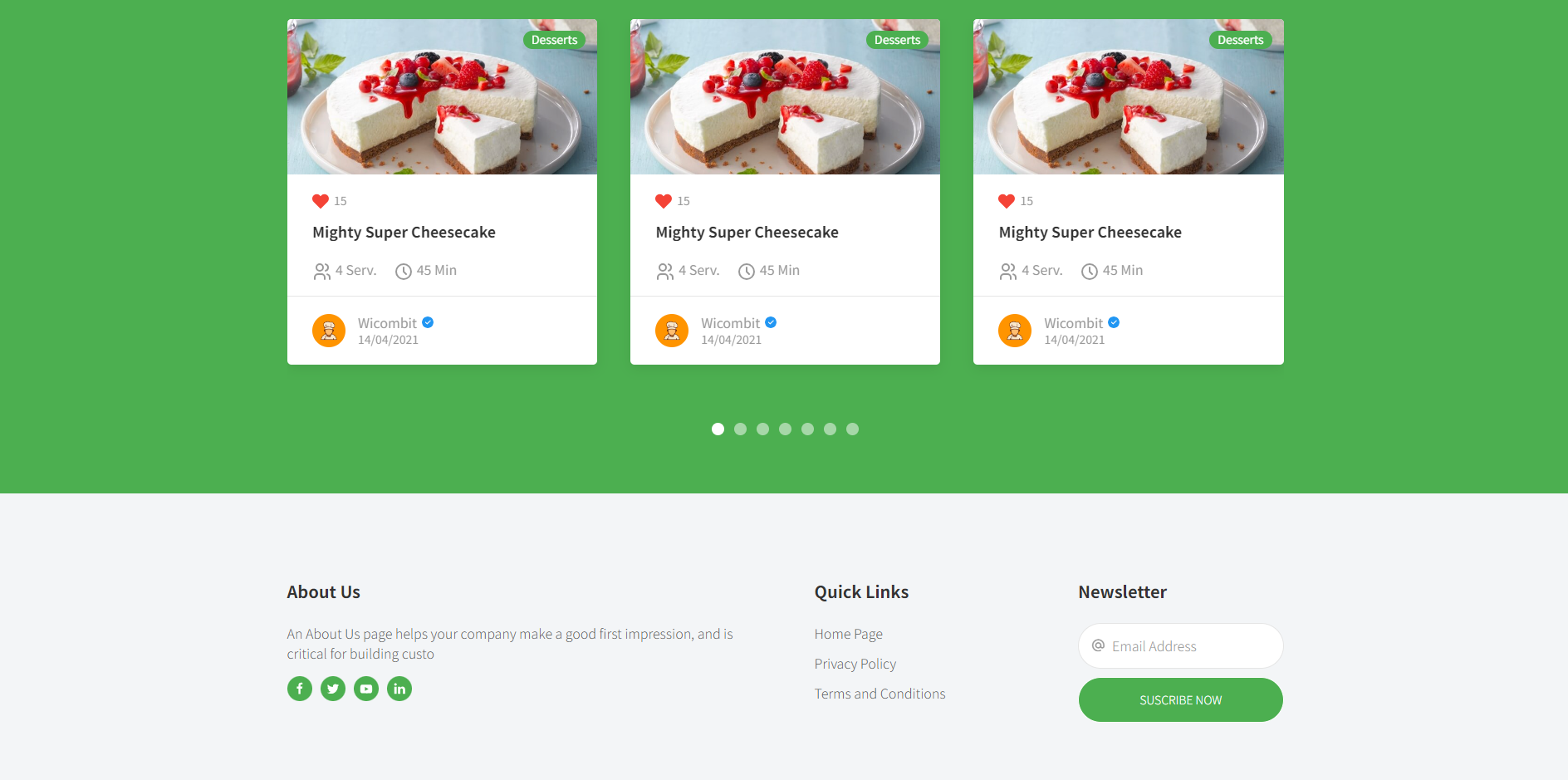
המנהל יכול להוסיף הרשאות וכן לצפות בכל שאר משתמשי המערכת למחוק ולעדכן מתכונים של משתמשים אחרים וכן לשלוח הודעות.

### מסכים:

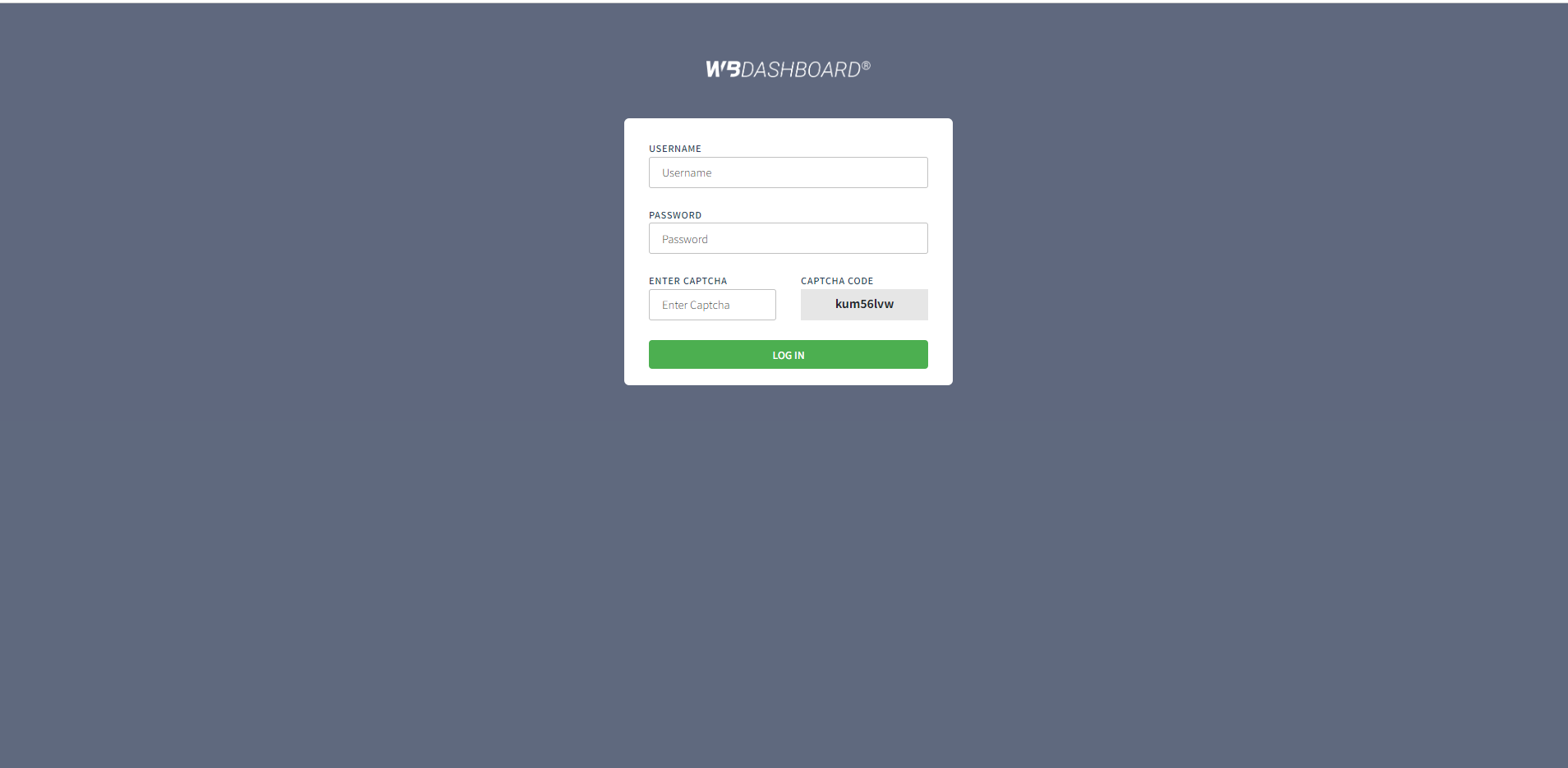
#### מסך פתיחה







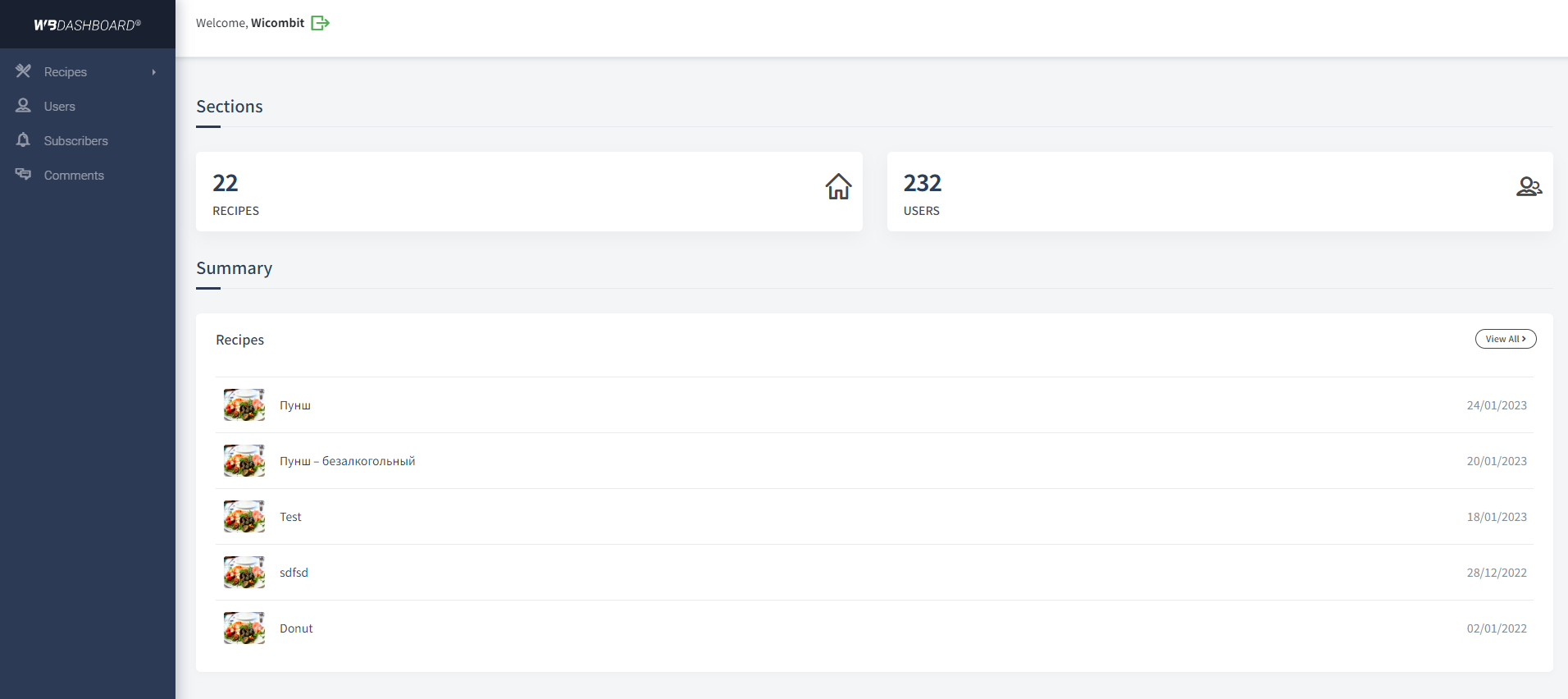
#### מסך כניסת מנהל



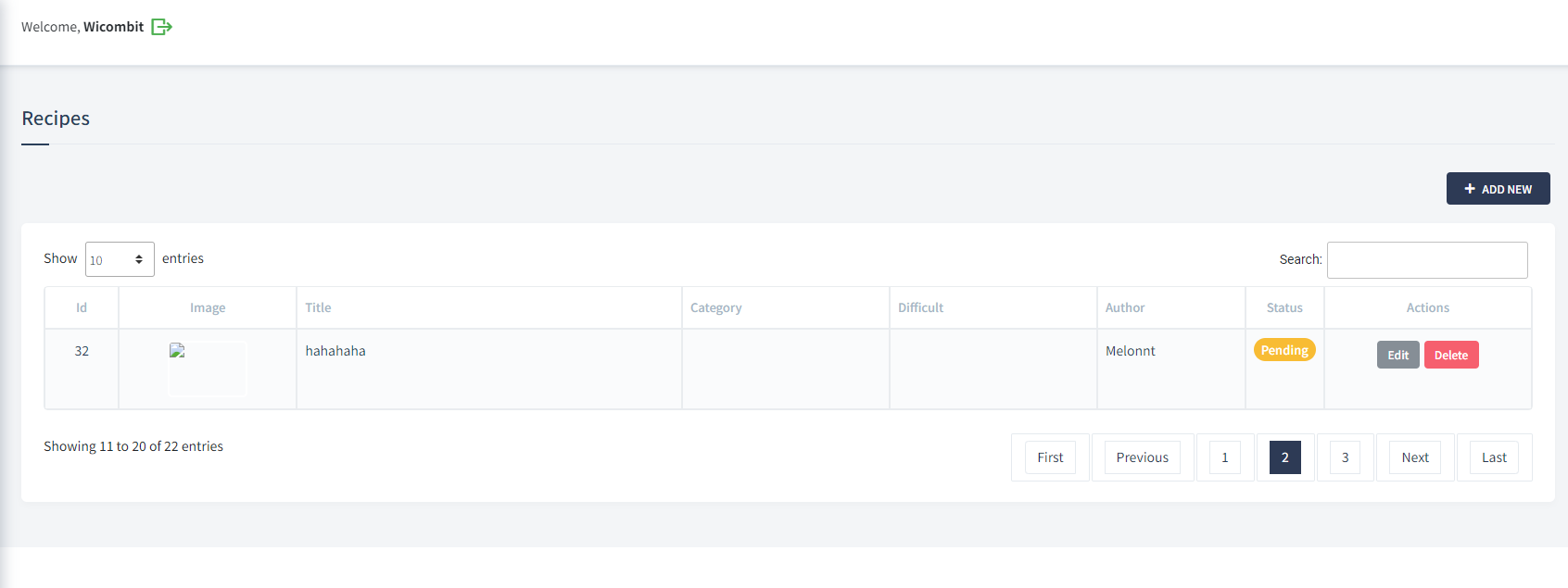
#### Menu תפריט צדדי -מנהל



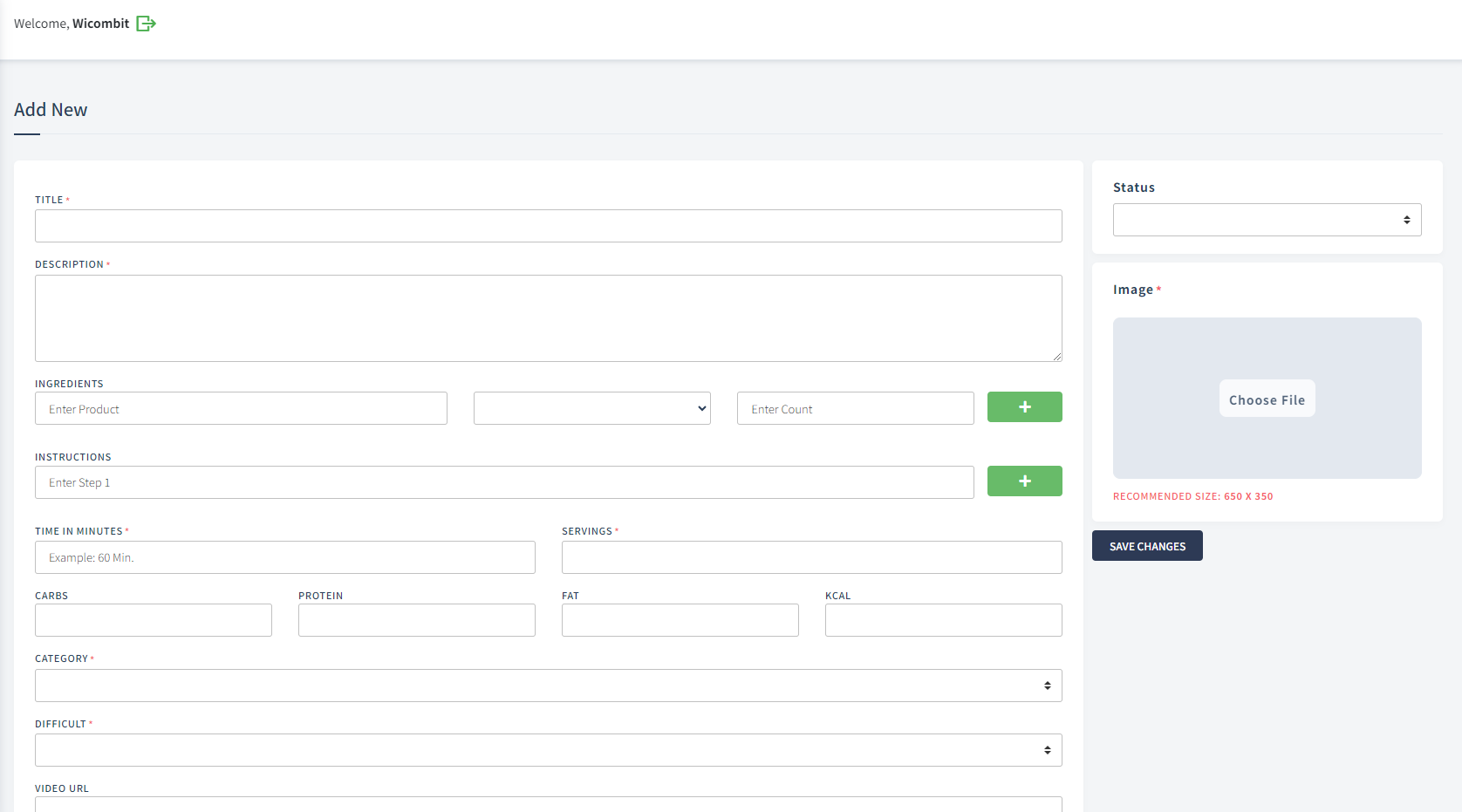
#### מסך ראשי-מנהל



#### מסך מתכונים



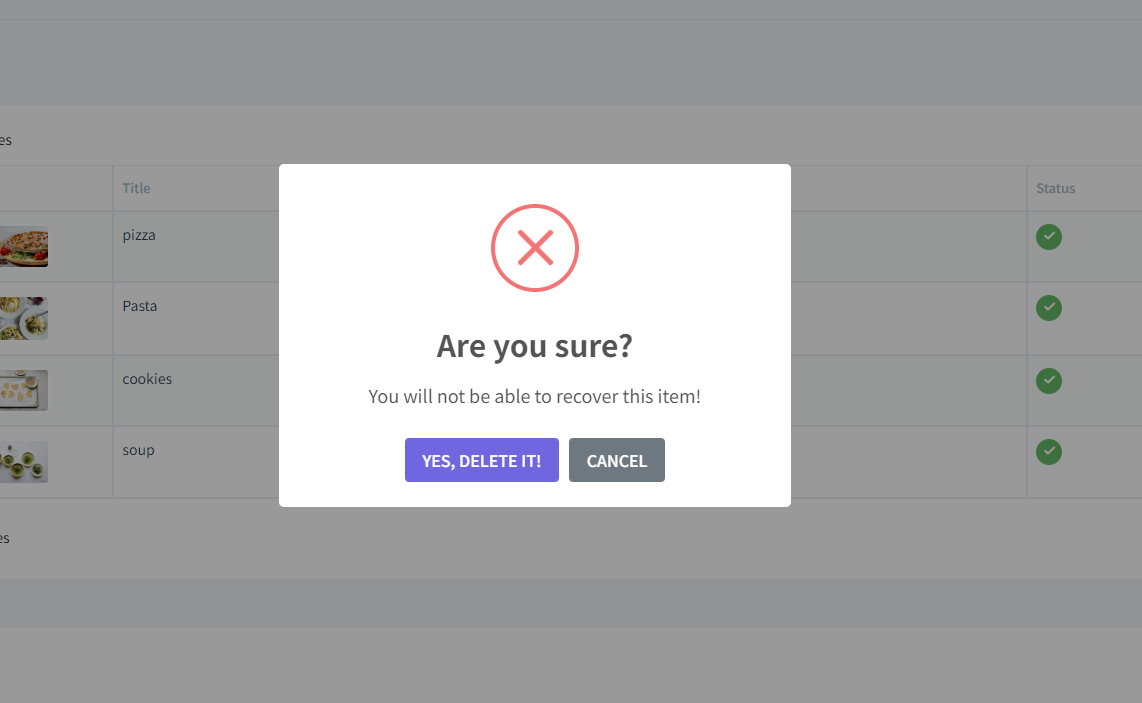
#### מסך הוספת מתכון חדש



#### מסך קטגוריות



#### מחיקת קטגוריה



#### הצגת רמות קושי



## סיכום ומסקנות:

השקענו רבות בבניה נכונה של המערכת ועל תקשורת נכונה עם השרת, ראשית הפרויקט הקנה לנו מרחב הסתכלות ופרספקטיבה רחבה על פרויקט מושלם, משלב ההצעה ועד לתוצר המוגמר תוך דרך ארוכה הכוללת באגים ופתרונם ויצירתיות רבה על מנת להפוך את האפליקציה ליעילה ,חווייתית ומקצועית ככל האפשר. במהלך הפרויקט למדנו הרבה ,ואנו מרגישות כי הוא תרם לנו רבות . תחומים שהכרנו באופן שטחי ,נעשו ידידותיים בהרבה ממה שחשבנו .השקענו זמן ומאמץ מרובים בלמידת נושאים וטכנולוגיות חדשים .גילינו שמהנה גם אם לא קל ללמוד ולהכיר תחומים חדשים ולנסות דרכי חשיבה מגוונות ומפתיעות שלא חשבנו עליהם ממבט ראשון. למדנו לחשוב בהגיון וביעילות על מנת להשיג תוצאות במהירות.

ומעל הכול הפנמנו שסבלנות והתמדה-אין כמותם בכל משימה הנראית קשה עד בלתי אפשרית...

## נספחים:

**Dependences-**

"dependencies": {

    "@angular/animations": "^16.2.10",

    "@angular/cdk": "^17.0.4",

    "@angular/common": "^16.2.8",

    "@angular/compiler": "^16.2.8",

    "@angular/core": "^16.2.8",

    "@angular/forms": "^16.2.8",

    "@angular/localize": "^16.2.12",

    "@angular/platform-browser": "^16.2.8",

    "@angular/platform-browser-dynamic": "^16.2.8",

    "@angular/router": "^16.2.8",

    "@ng-bootstrap/ng-bootstrap": "^15.1.2",

    "@popperjs/core": "^2.11.6",

    "@tabler/icons": "^2.38.0",

    "angular-datatables": "^16.0.0",

    "angular-tabler-icons": "^2.38.0",

    "datatables.net": "^1.13.6",

    "datatables.net-buttons": "^2.4.2",

    "datatables.net-buttons-dt": "^2.4.2",

    "datatables.net-dt": "^1.13.6",

    "jquery": "^3.7.1",

    "ng-select2": "^1.4.1",

    "ngx-take-until-destroy": "^5.4.0",

    "ngx-toastr": "^17.0.2",

    "rxjs": "~7.5.0",

    "select2": "^4.1.0-rc.0",

    "sweetalert2": "^11.7.32",

    "zone.js": "~0.13.3"

  },

  "devDependencies": {

    "@angular-devkit/build-angular": "^16.2.5",

    "@angular/cli": "~16.2.5",

    "@angular/compiler-cli": "^16.2.8",

    "@types/datatables.net": "^1.10.25",

    "@types/datatables.net-buttons": "^1.4.8",

    "@types/jasmine": "~4.3.0",

    "@types/jquery": "^3.5.22",

    "jasmine-core": "~4.5.0",

    "karma": "~6.4.0",

    "karma-chrome-launcher": "~3.1.0",

    "karma-coverage": "~2.2.0",

    "karma-jasmine": "~5.1.0",

    "karma-jasmine-html-reporter": "~2.0.0",

    "tslib": "^2.6.2",

    "typescript": "~5.1.6"

  }

## ביבליוגרפיה:

אתרים בנושא תכנות:

* stackoverflow.com
* webmaster.org.il
* [.c-sharpcorner.com](https://www.c-sharpcorner.com/article/web-api-in-asp-net)
* [.djamware.com](https://www.djamware.com/post/5a48517280aca7059c142972/ionic-3-angular-5-firebase-and-google-maps-location-tracking)
* .joshmorony.com
* .dotnetcurry.com
* .msdn.microsoft.com