פרויקט גמר

הנדסת תוכנה מה"ט

מגישות: אוריה חומרי ומיכל אסולין

מנחה: מירי ויכלדר



תוכן עניינים

מבוא

הגדרת דרישות ותיאור כללי

מדריך למתכנת

תיאור המערכת-תרשימים

תרשימי Use case

תרשים Data flow

דיאגרמת פעילויות

תרשימי זרימת מסכים

תכנות

תיאור

טכנולוגיות מתקדמות

עקרונות התכנות

חלופות שפות תכנות

בעיות ופתרונן

אבטחה

שרידות המערכת ותיעוד

ארכיטקטורת המערכת

תיאור השכבות

תיאור קוד פונקציות ומחלקות

מסד נתונים

בדיקות תוכנה

טבלת בדיקות מסכמת

Test case:

בדיקות פונקציונאליות:

בדיקות שימושיות:

בדיקות GUI:

מדריך למשתמש

סיכום

מקורות מידע

**מבוא**

הנחנו חיים בעידן טכנולוגי בו כולנו מחפשים את הדבר הבא, את החידוש שיהפוך את החיים שלנו לקלים ונוחים יותר, הכל חייב להיות נגיש ומהיר- אם לא הוא עלול להפוך ללא רלוונטי.

כשחשבנו על רעיון לפרויקט רצינו ליצור משו שיעמוד בקריטריונים של נח ונגיש למשתמש.

תמיד עמד מול עיינינו החשיבות שהפרויקט יהיה שימושי וכמובן בנוי בצורה הנוחה ביותר למשתמש ועם זאת להתמקד בתהליך הלמידה וההתפתחות שלנו בעזרת הטכנולוגיות הרווחות היום בשוק על מנת לצבור ידע וניסיון.

וכך נולד האתר Homeowner Association- ועד בית.

בעידן המודרני של היום, עם השימוש הרחב באינטרנט, התשלום דרך פלטפורמות המקוונות הפך לתהליך פשוט ונוח יותר מאי פעם. השימוש באינטרנט מאפשר חיסכון בשעות רבות של עבודה, שכן ניהול הבניינים והתקשורת עם הדיירים נעשים בקלות ובנוחות באמצעות ממשקים מקוונים.

מערכת הניהול המתקדמת שלנו כוללת מגוון רחב של פונקציות ותכונות, כולל אחזקה ותחזוקה שוטפת של כלל מערכות הבניין. באמצעות הטכנולוגיה, ניהול משימות שוטפות נעשה בצורה יעילה יותר ובמהירות גדולה יותר.

אף על פי כן, הטכנולוגיה המתקדמת אינה מקנה רק יתרונות לניהול הבניינים, אלא גם מאפשרת שקיפות מלאה. בזכות המערכות הממוחשבות, הדיירים יכולים להשפיע על החלטות חשובות הנוגעות לבניין בו הם מתגוררים, כך מתאפשרת השתתפות פעילה של הדיירים בניהול ובהחלטות הקשורות למקום מגוריהם.

התרגשנו לראות את הפרויקט נבנה מאפס, למדנו דברים חדשים, התנסנו בטכנולוגיות וספריות שלא הכרנו והשתדלנו לעשות את זה בדרך האוטנטית על מנת להביא את עצמנו לכתיבת אתר ברמה גבוהה בדרך הנהוגה בשוק העבודה כיום.

בכל שלב בפרויקט העמדנו את עצמנו במקום של המשתמש, ניסינו לחוות את האתר מהצד ולהבין איך להשתפר בהתאם. הקשיים והאתגרים הרבים במהלך הפרויקט גרמו לנו להתאים את עצמנו ואת התהליך העבודה שלנו אך לא לוותר בשום שלב על פיצ'רים שימושיים באתר או על חלקים העלולים לפגום בחוויית המשתמשים באתר.

הנה התוצאה לפניכם.

**הגדרת דרישות ותיאור כללי**

**מטרת המערכת:**

מטרת המערכת היא ליצור סדר וניהול הבניין, המערכת הולכת לייעל את התנהלותו של וועד הבניין על כל ההיבטים הכלולים בכך.

במערכת ישנן אפשרויות רבות לניהול הוועד וניהול הדיירים.

**היקף העבודה:**

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

**תיאור חומרת המערכת:**

מחשב PC\ מחשב 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 נתונים נעשה באמצעות Server SQL בטכנולוגית Entity Framework.

**מדריך למתכנת**

**תיאור המערכת:**

**ממשק המערכת:**

הפרויקט פותח בסביבת Web ומספק למשתמש יכולת

**תיאור המערכת- תרשימים**

**תרשימי Use case**



דיירים



מנהלים

**תרשים Data flow**

https(ssl)

**API External**

Data Base

**API**

Bl

Dal

https(ssl)

**Client**

**דיאגרמת פעילויות**

אלפון דיירים

לוח מודעות

**גרף**

**הכנסות**

**הוצאות**

דיירים

תקציב

עדכונים

לא

כן

כניסה

הרשמה

משתמש רשום?

start

**תרשימי זרימת מסכים**

אלפון דיירים

לוח מודעות

**גרף**

**הכנסות**

**הוצאות**

דיירים

תקציב

עדכונים

כניסה

הרשמה

דף הבית

**תכנות**

**תיאור**

המערכת נכתבה בשפת C#.NET בסביבת 2019 Visual Studio ועושה שימוש בטכנולוגיות מתקדמות, בתוכם: Web API, Angular8, Entity Framework וכן בספריית Bootstrap.

כאשר צד ה- Server פותח בטכנולוגיתWeb API בשיטות המתקדמות ביותר ש- NET מציע וצד ה- Client פותח באמצעות Angular8.

**טכנולוגיות מתקדמות**

* 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:

* LINQ- שפת צד שרת, שיטת כתיבה חדשנית מובנית ב-NET, המיועדת לריצה על lists מערכים של אובייקטים וחיפוש בתוכם בצורה פשטנית, חכמה וקלילה.

קטע קוד לדוגמא:

* Entity Framework-טכנולוגית עבודה מתקדמת מול DB. יצירת אובייקטים ומיפוי מול טבלאות הDB, כאשר אופן העבודה ועיבוד הנתונים בשפת LINQ כמוזכר לעיל.
* 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 S.P.A Single Page Application -,

כלומר, רק חלק בדף מסוים מתרענן ומתחלק 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 Angularאיזה תוכן HTML להציג עבור קומפוננט זו, ה templateמכיר את הקומפוננט ואת כל השדות והפונקציות הכתובות בה.

Angular משתמשת בשיטת ה- tow way data binding, כלומר, כל שינוי הנעשה ב- template משפיע ישירות על הקומפוננט ולהפך.

* Typescript- שפה חדשה הדומה לתקן החדש של JavaScript.
* Angular Material, Syncfusion ej2, SweetAlert2- חבילות Components מעוצבות.

**עקרונות התכנות**

* 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.
* 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.
* מסכי המערכת נכתבו בצורה פשטנית וברורה ביותר לעין, כך שניתנים לשימוש בקלות, ונותנים חווית משתמש.
* 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** שפהבסיסית 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 טסטים, תמונות, סרטוני וידיאו ומוסיקה וכדי לבצע מגוון פעולות נוספות.

**שפת Java Script**- 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 גם נקראת "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 MVCאינה מזכירה במפורש את שכבת הגישה לנתונים, מכיוון ששכבה זו 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 HTMLוהקוד 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,

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

**בעיות ופתרונן**

במהלך הפרויקט נתקלנו במספר אתגרים שהיינו צריכות להתמודד איתם.

* כככככ
* אתגר נוסף שהיינו צריכות להתמודד איתן הוא עבודת הצוות מרחוק, אומנם אנו יודעות להתנהל אחת עם השנייה ואין לנו קושי בהתמודדות עם עבודת צוות 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 מהדרישות באתר היא יצירת אתר רספונסיבי. מתחילת כתיבת הפרויקט נתקלנו בבעיות רספונסיביות מכיוון שגדלי המסך שלנו שונים, מה שגרם לנו להבין את חשיבות הרספונסיביות, יצר אצלנו צורך ורצון להשקיע בכך מההתחלה.

פתרון בעיה זו ניהלנו באמצעות ספריות עיצוב של Bootstrap ובאמצעות תכונות רספונסיביות עבור כל רכיב שהשתמשנו בו.

**אבטחה**

השקענו עמל והשקעה רבה בהגנות מידע של המשתמשים באתר.

האתר שלנו מאובטח ע"י פרוטוקול SSL המהווה תקן להצפנת ואבטחת האתר לגלישה בטוחה של המשתמשים.

על מנת לאבטח את המידע על הנתונים שלנו בSQL ע"י כתיבת

**שרידות המערכת ותיעוד**

**ארכיטקטורת המערכת**

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

**תיאור השכבות:**

**Dal (Data Access Layer)** שכבה המקשרת את הפרויקט ל DB.

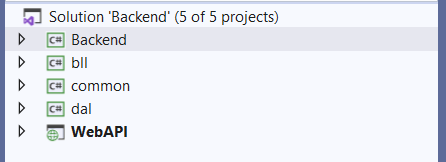
**Bl (Business Layer)** שכבה זו כוללת את הלוגיקה בפרויקט.

**Entities** שכבה זו ממירה משתנים מסוג ה Entities DB ולהפך.

**Web API (Application Programming Interface)** תוכנת שרת המספקת אינפורמציה לתוכנת לקוח.

**GUI (Graphical User Interface)** שכבה המספקת ממשק משתמש נוח ונעים לעין, שכבה זו נכתבה ב Angular.

מראה הSolution של הפרויקט:



**תיאור קוד פונקציות ומחלקות**

**Client Side-**

**GUI (Graphical User Interface)**

את צד הלקוח כתבנו ב Angular8.

בניית צד לקוח 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 Classes , Componentsו-Services.

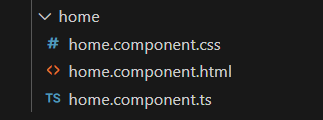
כל Component מורכב מ:

* קובץ html – מראה הממשק.
* קובץ CSS- עיצוב קובץ html.
* קובץ TS(TypeScript) -כתיבת הקוד.

Classes- קבצים המגדירים לנו תכונות לאובייקטים מסוימים.

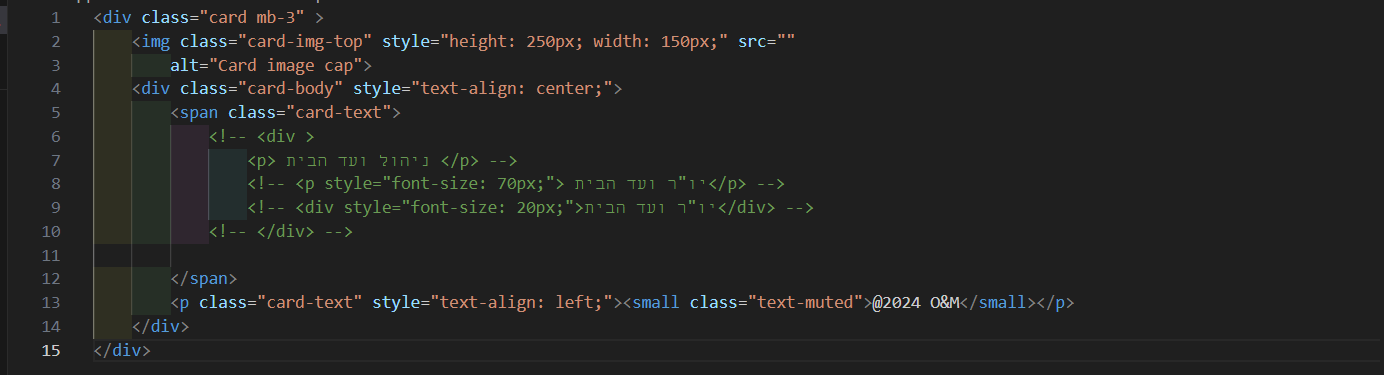
Services- קבצים לוגיים המיועדים לשמירת נתונים ולניהול תקשורת עם ה.Web API

דוגמה ל Component

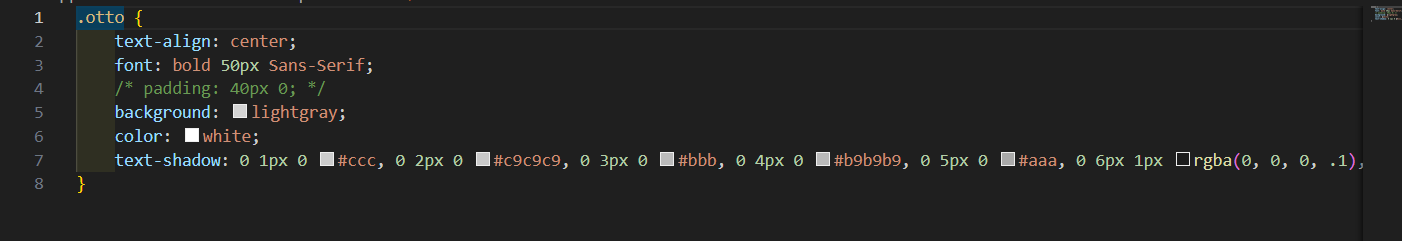


בקומפוננט זה מוצג למשתמש מסך הבית של אתר

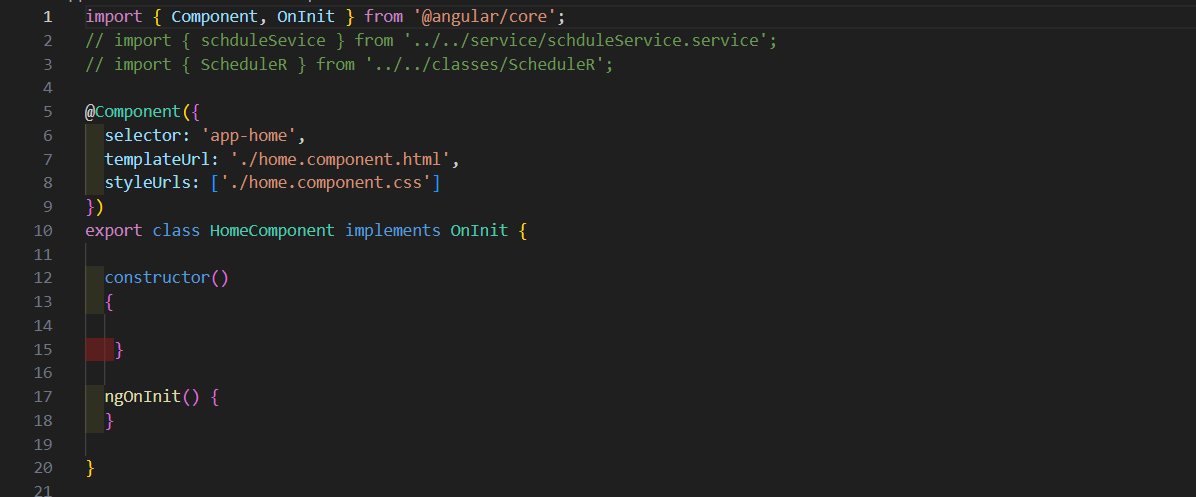
[**home.component.css**](http://home.component.css)



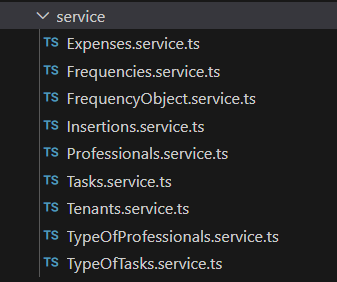
[**home.component.css**](http://home.component.css)



[**home.component.ts**](http://home.component.ts)

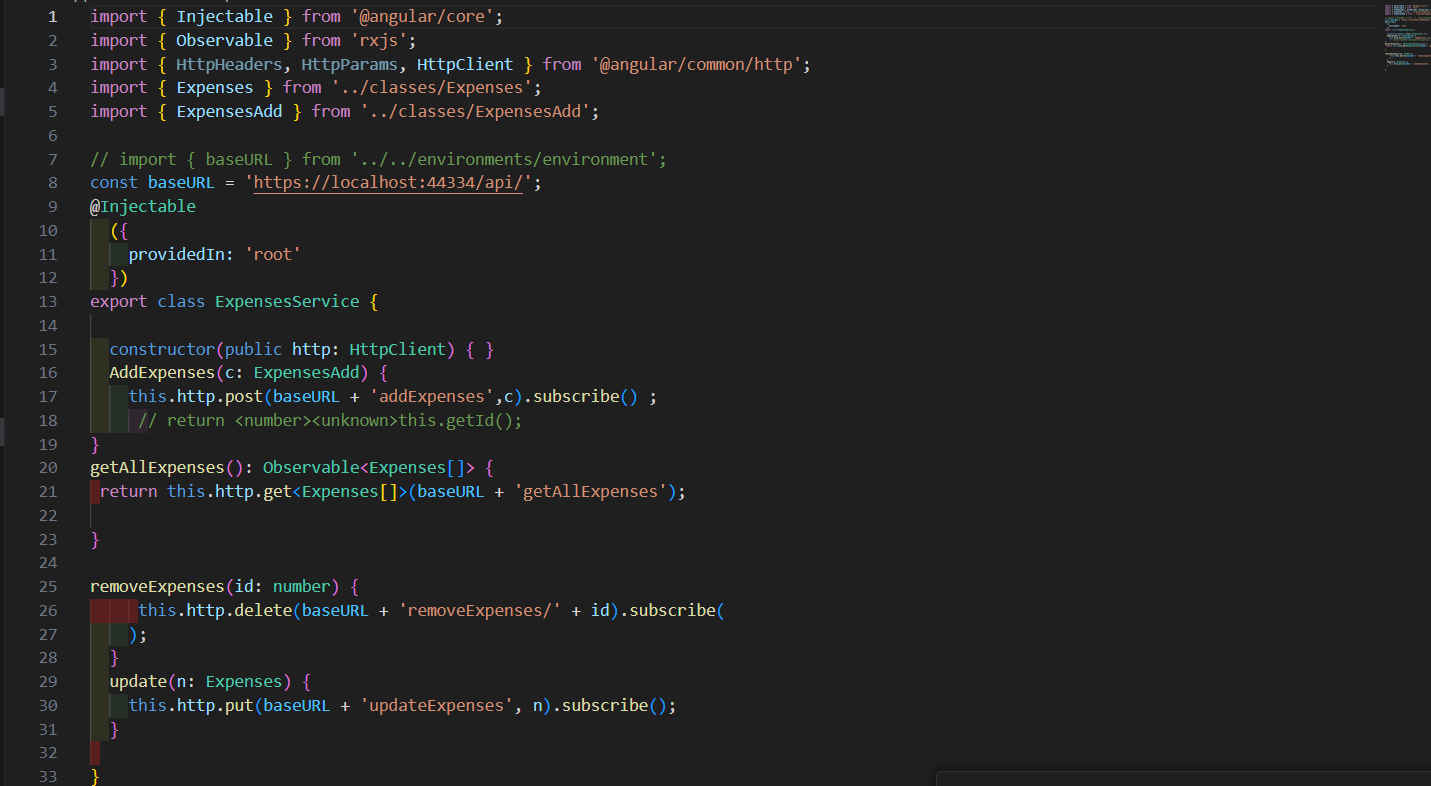


**Services:**

****

**Service דוגמא:**

**Expenses.service.ts**

****

**Server Side-**

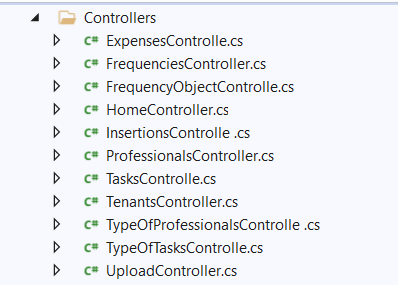
**Web API (Application Programming Interface)**

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 מסוג GET,PUT, POST ו-DELETE.

כל פונקציה ב controllerמתממשקת לסביבת הclient.

**מחלקות ה- Controllers:**

****

**מחלקת controller לדוגמא:**

**Bl (Business Layer)**

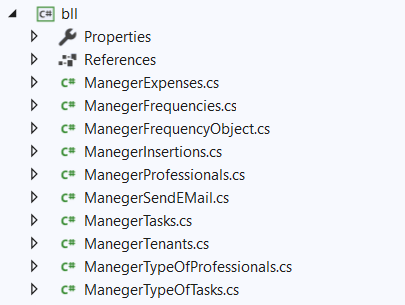
שכבה זו כוללת את הלוגיקה בפרויקט.

עבור כל מחלקה.Blבשכבה זו בנינו מחלקת

מחלקות ה- Bl מגשרות בין מחלקות ה- **controller** למחלקות ה- Dal.

בנוסף בשכבה זו מתבצע החיבור ל API החיצוני, עבור כל פעולת התממשקות עם הAPI יצרנו פונקציה מתאימה המאפשרת לנו גישה קלה בעזרת פונקציות פשוטות לשליפות הרצויות מהAPI.

**מחלקות ה-Bl:**

****

**מחלקת Bl לדוגמא:**

**ManegerExpenses**

****

entities

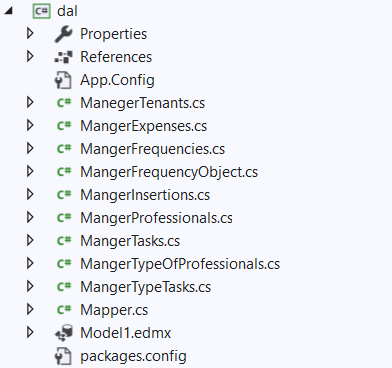
**Dal (Data Access Layer)**

בשכבה זו אנו מתחברים ל Database(מסד נתונים).

במחלקת הDal נמצאות כל הטבלאות הקיימות ב DB כולל קשרי הגומלין שלהן.

טבלאות אלו הינם הקרובות ביותר ל Databaseומייצגות אותו, הן המגשרות בין הDB לBL.

**שכבת ה- Dal:**

****

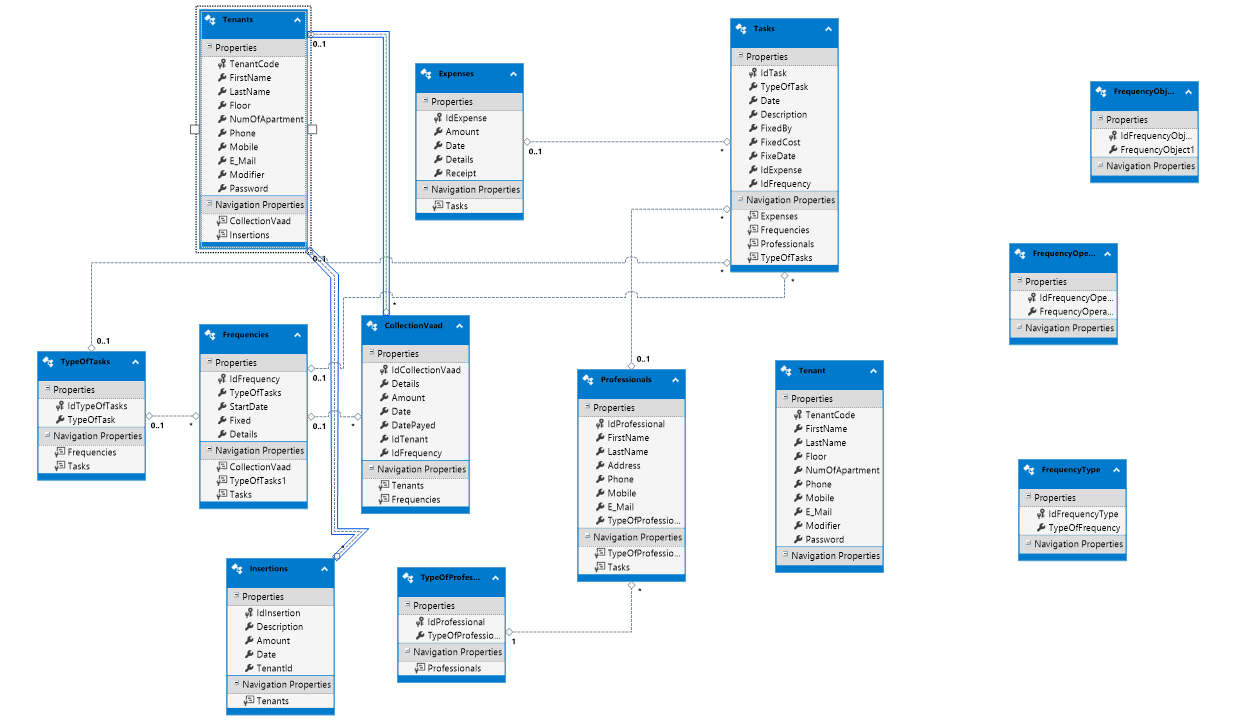
**Entity framework:**

כלי המאפשר גישה קלה לנתונים הנמצאים ב Database.

כלי זה יוצר לנו מודל לכל ישות וכל מודל מקושר לטבלה בDB.

בתרשים הבא ניתן לראות את הטבלאות שנוצרו ב- Entity framework:

**Model1.edmx**

**מסד נתונים**

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

מסד הנתונים מכיל טבלאות המקשרות 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:

SQL

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

המנגנון של Entity framework יוצר את ה Classes על פי החוקיות:

-טבלאות בקשר של יחיד ליחיד- יוצר שני Class בקשר של הורשה.

-טבלאות בקשר של יחיד לרבים- יוצר Class לכל טבלה, ב Classשל הרבים יוצר מופע מסוג ה Class של היחיד וב Class של היחיד יוצר אוסף מסוג ה Class של הרבים.

-טבלאות בקשר של רבים לרבים- יוצר Class ובו אוספים של Classes המקושרים.

רצינו לומר תודה.....

לאחר חודשים ארוכים שך 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 לגברת תמר קארפ, על הליווי הצימוד 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 שהגנו לרעיון שאהבנו. לאורך כל הדרך היא הקשיבה הסבירה וכיוונה אותנו עד הפרטים הקטנים. תודה רבה!

כמו כן אנו רוצות להודות לרות 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