דף כריכה

המכון הממשלתי להכשרה בטכנולוגיה ובמדע

המגמה לתוכנה

מוגש ע"י:

גולדמן שרה

גרינוולד מלכה

בהנחיית: שימונוביץ מרים

תש"פ 2019

דף שער

המכון הממשלתי להכשרה בטכנולוגיה ובמדע

reMEmber

שמות המגישות:

גולדמן שרה וגרינוולד מלכה

בהנחיית: שימונוביץ מרים

רכזת המגמה: ברגמן חנה

תש"פ 2019

reMEmber

הצעה לפרויקט גמר

reMEmber

אישור הצעת הפרויקט ממה"ט

תודות

בראש ובראשונה-**בורא עולם**

אשר מוביל אותנו לכל אורך הדרך

ובסיעתא דשמיא הביאנו עד לשלב הגמר!

הרב וולף שליט"א , מנהל הסמינר,

אשר הטמיע בנו ערכי נצח בד בבד עם מסירותו לנתינת כל הכלים

על מנת לאפשר לנו להצליח במשימתנו-הבאת ה"קמח" לבית של תורה.

הגב ח.ברגמן תח"י, מרכזת המסלול,

אשר מסירותה והשקעתה עבורנו בלי גבול ומידה!

לכל אורך הדרך תמכה ופעלה לכולם יחד ולכל אחת לחוד ולא חוסכת כל מאמץ

לתת את הטוב ביותר שניתן בלימודים, בשיבוצינו במקום עבודה והכל מעל ומעבר!

למנחת הפרויקט הגב מרים שימונוביץ

שנתנה לנו כלים מעשיים הכוונה ויעוץ - והכל בסבלנות ובמאור פנים.

שעשתה כל שביכלתה בעבורנו!!, הקדישה מזמנה שעות של השקעה בבנית הפרויקט,

תכנונו והוצאתו לפועל והכל בחיוך ובשמחה, בהרגשה טובה ונעימה בנכונות אמיתית וכנה

לעזור לולי התמיכה המרובה והעידוד שלה- לא היה פרויקט זה תם ונשלם(וכ"כ מושלם!..)!!!!

ואחרונים חביבים-בני משפחתינו היקרים,

אשר לוו אותנו בעידודם, תמיכתם ותפילתם להצלחתינו

נתנו את הזמן שצרכנו, והתאזרו בסבלנות רבה לכל אורך הדרך! **יישר כח!**

הצהרה

הצהרת סטודנט

שם הסטודנט: שרה גולדמן ת.ז. 314886656

שם הסטודנט: מלכה גרינוולד ת.ז. 317813145

שם המכללה בה לומד הסטודנט: סמינר וולף שלוחת המכללה למנהל ראשון לציון

אני הח"מ, מצהיר בזאת כי פרויקט הגמר וספר הפרויקט המצ"ב נעשו על ידי בלבד.

פרויקט הגמר נעשה על סמך הנושאים שלמדתי במכללה ובאופן עצמאי.

פרויקט הגמר וספר הפרויקט נעשו על בסיס הנחייתו של המנחה האישי.

מקורות המידע בהם השתמשתי לביצוע פרויקט הגמר מצוינים ברשימת המקורות המצוינים בספר הפרויקט.

אני מודע לאחריות שהנני מקבל על עצמי על ידי חתימתי על הצהרה זו שכל הנאמר בה אמת ורק אמת.

חתימת הסטודנט: _____ תאריך: 11/11/2019

חתימת הסטודנט: _____ תאריך: 11/11/2019

<u>אישור המנחה האישי</u>

הריני מאשר שהפרויקט בוצע בהנחייתי, קראתי את ספר הפרויקט ומצאתי כי הוא מוכן לצורך הגשת הסטודנט להגנה על פרויקט גמר.

שם המנחה: _____ חתימה: _____ תאריך: _____

<u>אישור ראש המגמה</u>

הריני מאשר שספר הפרויקט מוכן לצורך הגשת הסטודנט להגנה על פרויקט הגמר.

שם ראש המגמה: _____ חתימה _____ תאריך: _____

תקציר

פרויקט "Remember" נועד להקל על אנשים, ולהזכיר להם לקנות מוצרים שונים להם הם זקוקים, כשהם קרובים לחנויות המוכרות מוצרים אלה.

המשתמש מכניס למערכת אלו מצרכים ברצונו לקנות, וכך קופצת הודעה מתאימה בזמן שהוא קרוב לחנות.

איך זה עובד?

בעלי החנויות מכניסים את פרטיהם, היכן החנות נמצאת- מיקום, ואילו קטגוריות הם מוכרים. אם אין קטגוריה מתאימה לחנות ברשימת הקטגוריות של האתר- יש אפשרות לשלוח בקשה להוספת קטגוריה במייל. בעל החנות יוכל בכל עת לשנות את פרטיו. המשתמש, מקליד איזה מוצר הוא צריך, וכן באיזו קטגוריה. כך הוא מפעיל חיפוש. ברגע שהמשתמש עובר ליד חנות שמוכרת קטגוריה שהוא צריך- קופצת הודעה המודיעה איזה מוצר הוא יכול לקנות עכשיו, ואיפה. המשתמש יוכל להחליט אם ברצונו לקנות עכשיו- להפסיק חיפוש, או לא- להמשיך בחיפוש. המשתמש יכול לראות את היסטורית החיפושים שלו על פי סינונים שונים.

הפרויקט בנוי מאתר לחנויות, ואפליקציה למשתמשים.

הפרויקט נכתב בשפות: #C- צד שרת. Angular- צד לקוח, בעלי החנויות. Ionic- צד לקוח, משתמשים.

תוכן עניינים

1. מבוא

כולנו מכירים את זה: חייבים לקנות משהו קטן בדחיפות, ויש אפילו חנות בדרך לעבודה. אבל, כל יום, כשמגיעים הביתה, נזכרים: שכחתי לקנות.

בשביל לפתור את הבעיה הזו, פיתחנו אפליקציה למשתמש בה הוא כותב את כל המוצרים שאותם צריך לקנות, וברגע שהוא עוברים ליד החנות, קופצת הודעה מתאימה.

התוכנה תכיל שני ממשקים:

<u>בעלי חנויות-</u>

זהו האתר, בו מכניסים בעלי החנויות את פרטיהם, כולל מיקום מדויק וקטגוריות הנמכרות בחנותם. באתר יהיו אפשרויות של עריכת הפרטים, שליחת בקשה להוספת קטגוריה שאינה ברשימת הקטגוריות לבחירה,
סטטיסטיקות של מוצרים נמכרים.

<u>משתמשים-</u>

זו האפליקציה, בה מפעיל המשתמש חיפושים שונים על מוצרים. כשהוא קרוב לחנות המוכרת אחד מהחיפושים, קופצת הודעה. באפליקציה יהיו אפשרויות של הצגת חיפושי המשתמש על פי סינונים שונים, ביטול חיפוש ועוד.

2. תאור הפרויקט

2.1 אסטרטגיות וטכנולוגיות

בכתיבת הפרויקט השתמשנו בשפות ובטכנולוגיות הבאות

- #C - מבית חברת מיקרוסופט העולמית, היא שפת תכנות לשימוש בסביבת הפיתוח NET.המותאמת בין השאר לרשת האינטרנט ולטלפונים החכמים. מדובר באחת מן השפות הנפוצות ביותר כיום לתכנות מונחה עצמים. שפה זו נמצאת בשימוש נרחב בפיתוח פלטפורמות ליישומי אינטרנט, כמו גם ליישומים בסביבת Windows
- Angular - היא תשתית תוכנה (framework) בקוד פתוח ליישומי רשת, המתוחזקת על ידי Google ועל ידי קהילה רחבה של מפתחים. התשתית מיועדת לפתרון אתגרים בפיתוח יישומי דף-יחיד, ופישוט הפיתוח והבדיקות של יישומים אלו, באמצעות תשתית תוכנה לארכיטקטורות צד לקוח כמו MVC או MVVM, יחד עם רכיבים בהם משתמשים בדרך כלל ביישומי אינטרנט עשירים.
- Ionic – לפיתוח אפליקציות בפלטפורמות IOS ו ANDROID

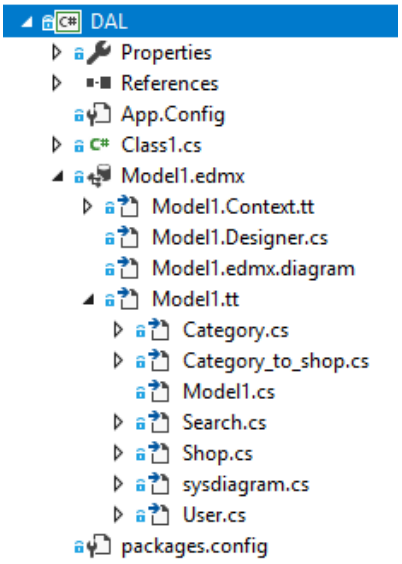
2.2 תיאור מבנה הפרויקט

<u>SERVER SIDE</u>

השרת נבנה בהתאם למודל השכבות:

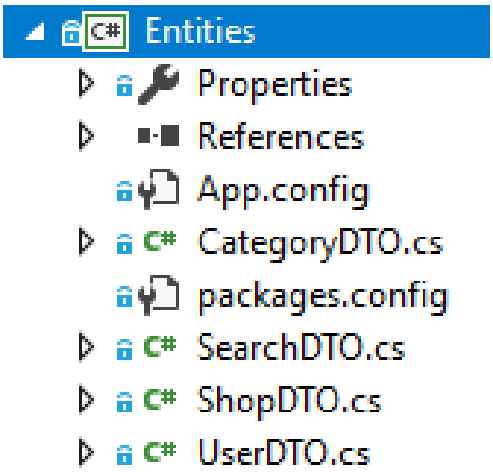
- DAL:

פרויקט זה אחראי על המידע, כלומר בפרויקט זה נמצאות המחלקות המייצגות את הישויות במסד הנתונים.



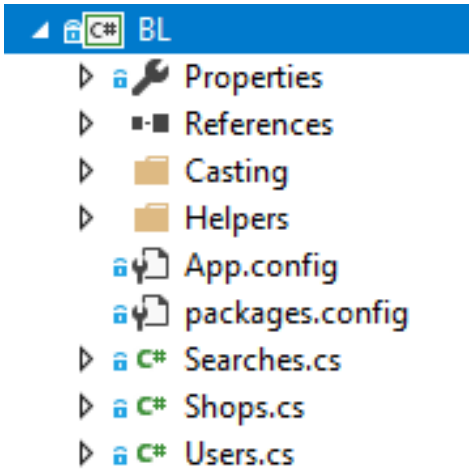
- Entities:

מכיל את המודלים המקבילים לישויות מסד הנתונים ומשמש מעטפת.



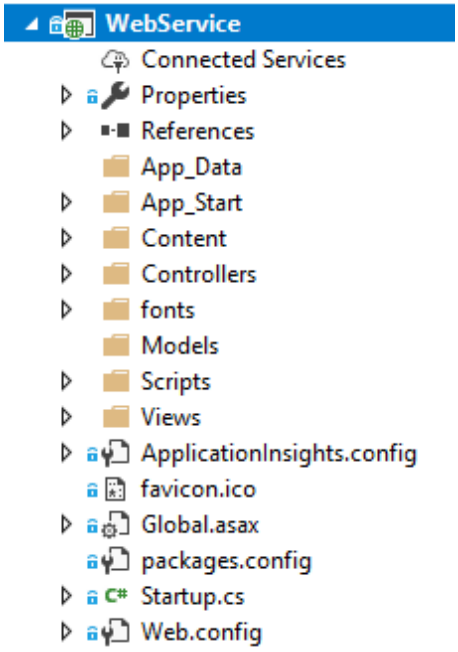
- BL:

פרויקט זה מכיל את עיקר הלוגיקה של הפרויקט, כגון, שליפת נתונים מהDB,הוספה עדכון ומחיקה מה DB.



- ::WebService

חשיפת API עימו מתממשקת ה Client.

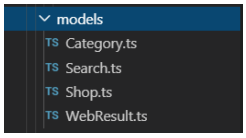


CLIENT SIDE-ANGULAR

מבנה האתר ע"פ מבנה של פרויקט אנגולר, הכולל:

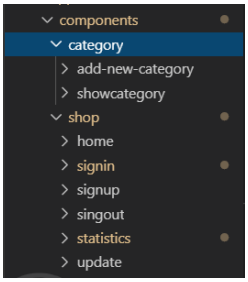
- :MODELS

כאן ממוקמות המחלקות שבאות לידי שימוש באתר.



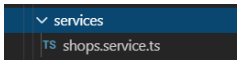
- COMPONNENTS:

המסכים שבאתר .כל מסך כולל דף html המכיל את האלמנטים במסך, דף CSS המכיל את העיצוב ודף ts המכיל את הלוגיקה לכל מסך.



- SERVICES:

שרות המספק מידע לקומפוננטות ע"י בקשות מהServer.

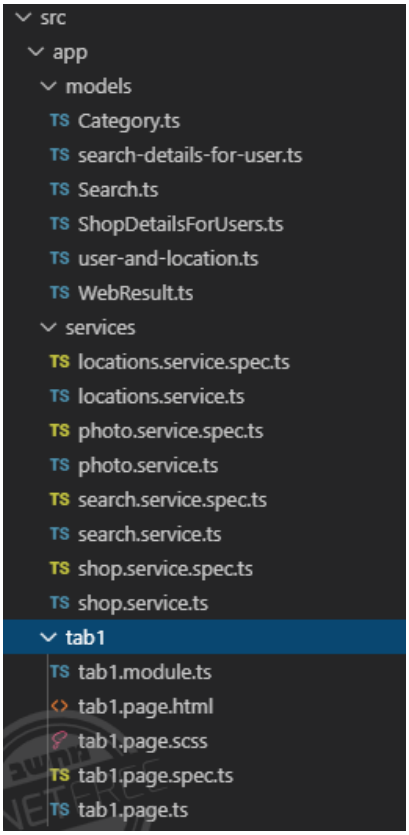


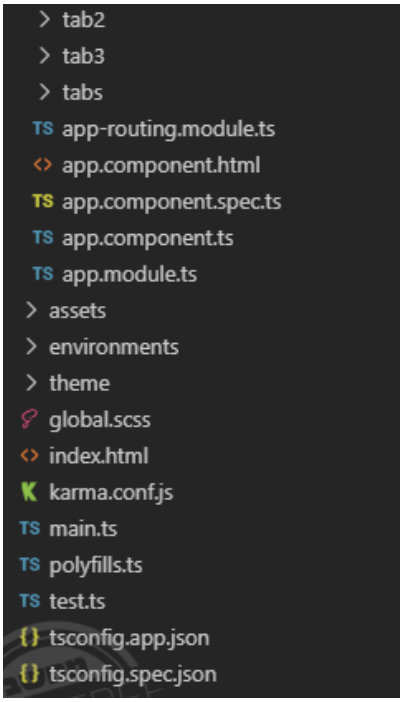
CLIENT SIDE-IONIC

המבנה ע"פ מבנה של פרויקט ionic 3, הכולל:

- App: הרכיב הראשי המכיל את האפליקציה בעצמה.
- Models: כאן ממוקמות המחלקות שבאות לידי שימוש באפליקציה.
- Tabs: הכרטיסיות שבאפליקציה. באפליקציה הזו ישנם שלוש כרטיסיות. כל כרטיסיה כוללת דף html המכיל את האלמנטים בכרטיסיה, דף scss המכיל את העיצוב, ודף ts הכולל את הלוגיקה לכל כרטיסיה.
- Services: כאן ממוקמת הלוגיקה הכללית שמאחורי הפרויקט.

איור מבנה ה-Client:





2.3 עקרונות התכנון/ הבניה/ הניתוח

2.3.1 עקרונות תיאורטיים:

- <u>הפרדת שכבות</u>

כל תוכנה/ אתר שנפגוש בעולם מבוססים על ארכיטקטורת שכבות הנקראת Tree-Tier-Application.

זאת אומרת, שלוש שכבות במבנה של DAL-BL-UI.

זוהי תבנית עיצוב בסיסית שמגדירה הפרדת האפליקציה ל: שכבת נתונים, שכבת לוגיקה, ושכבת ממשק משתמש.

- <u>REST API</u>

מה זה API ?

API הוא קיצור של Application Programming Interface. 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 REST ?

Rest ראשי תיבות (Representational State Transfer) 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 .

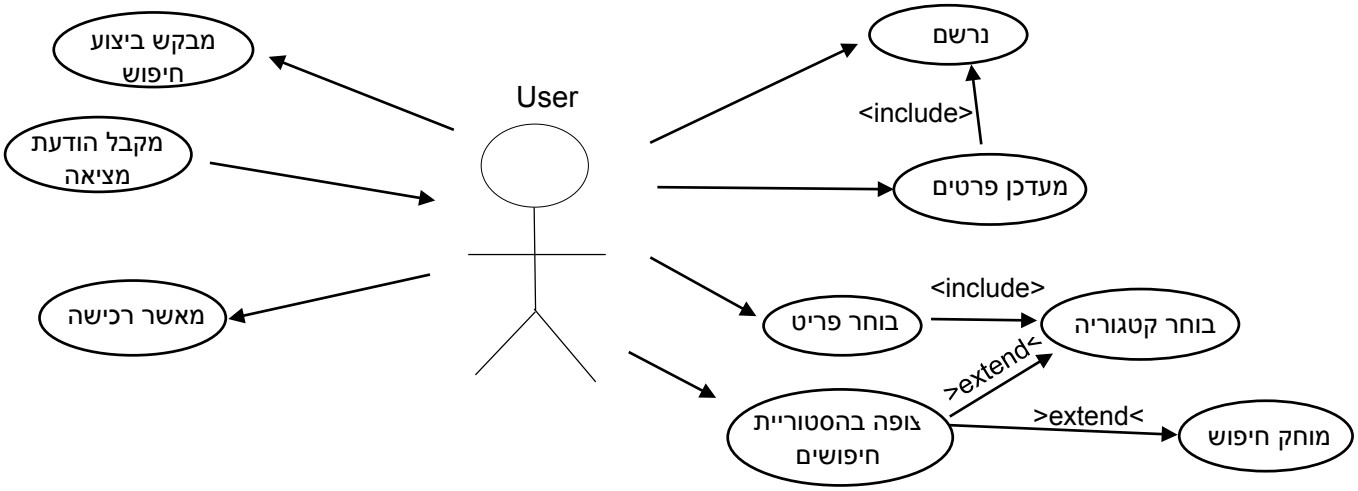
לסיכום Rest API:

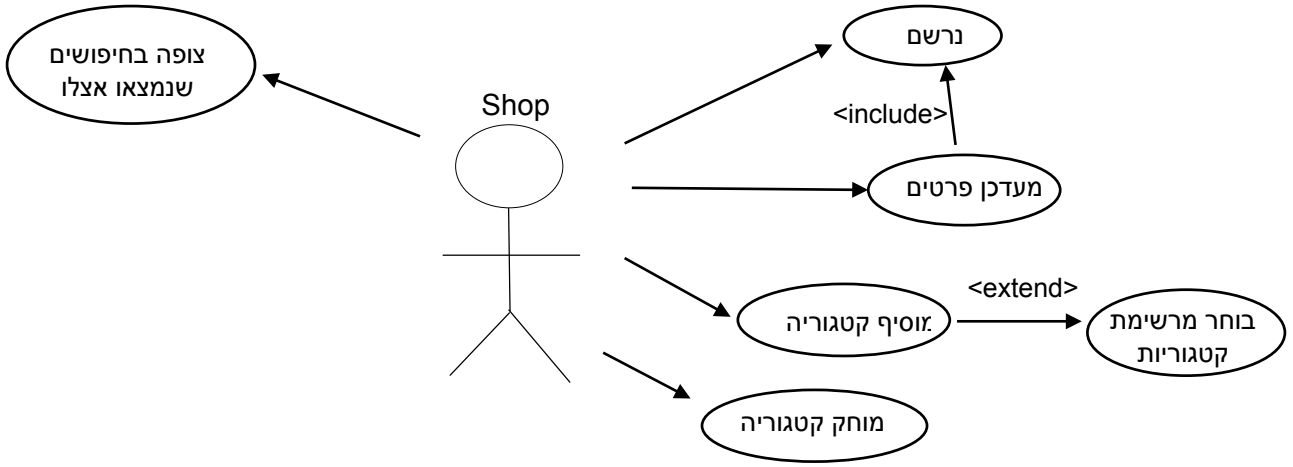
ממשק API של REST מגדיר קבוצה של פונקציות אשר מפתחים יכולים לבצע בקשות ולקבל תגובות באמצעות פרוטוקול HTTP כגון GET ו POST.

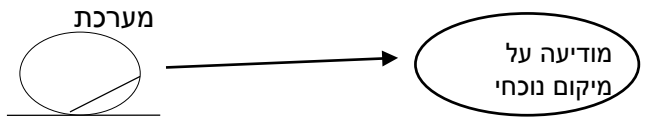
חשפנו את הפרויקט שלנו ל API Rest כך שיכלנו להשתמש מול אותן הפונקציות גם מהאתר – Angular וגם מהאפליקציה Ionic .

2.4 תרשימים

תרשים Uml

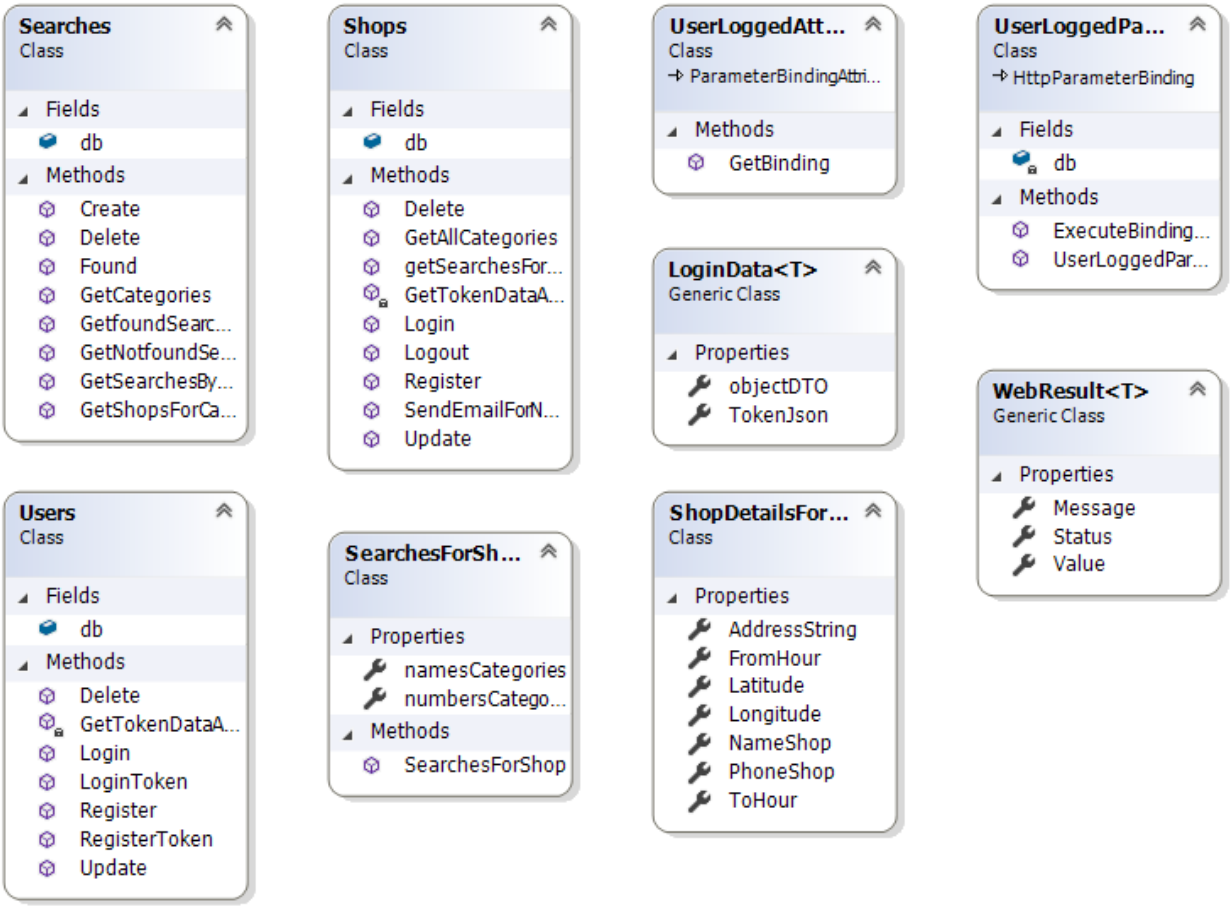



צופה בחיפושים
שנמצאו אצלו
Shop
נרשם
<include>
מעדכן פרטים
<extend>
בוחר מרשימת
קטגוריות
מוסיף קטגוריה
מוחק קטגוריה


מערכת
מודיעה על
מיקום נוכחי

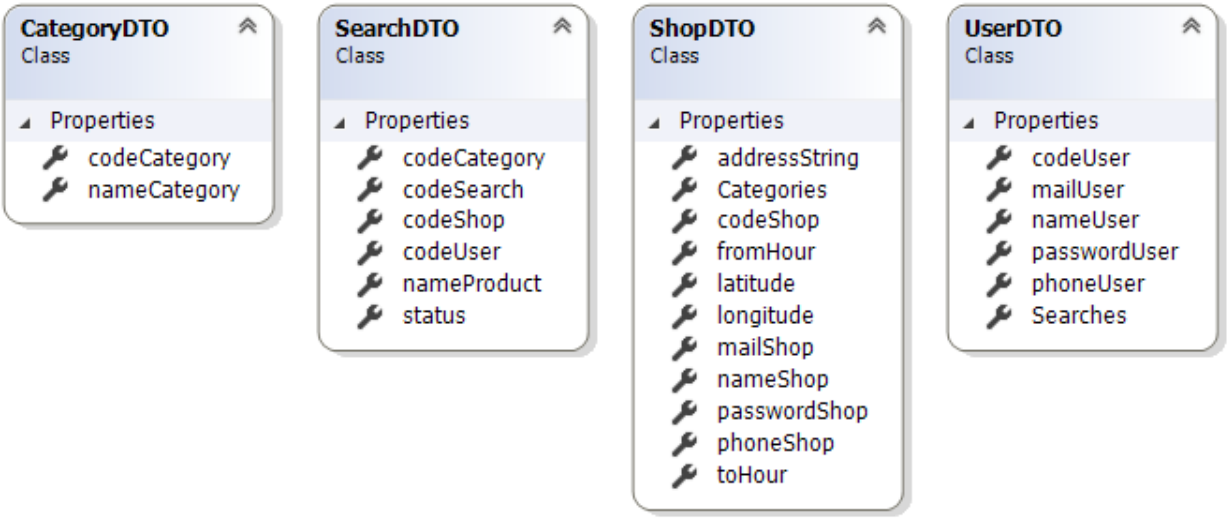
תרשים מראה המחלקות:

בשכבת ה BL:

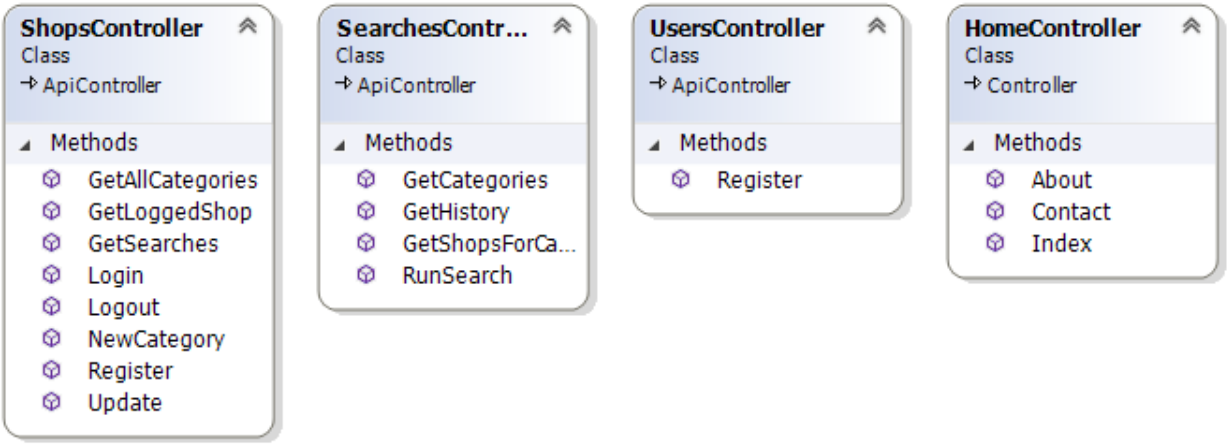


reMEmber

בשכבת ה ENTITIES:

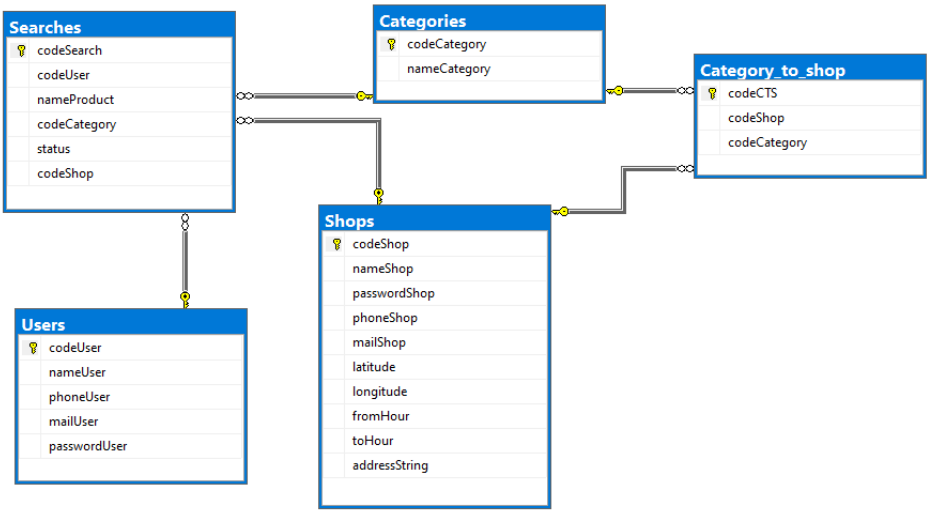


בשכבת ה WEBSERVICE:



2.5 מבנה נתונים מאוחסנים

תרשים SQL:



תיאור הטבלאות:

Users- משתמשים. אלה הם המשתמשים באפליקציה, שמפעילים חיפושים.

Searches- חיפושים. כל החיפושים שהופעלו.

Shops- חנויות. כאן מאוחסנים כל הפרטים על החנויות ומה שהן מוכרות.

Categories- קטגוריות. כל חנות מוכרת קטגוריה/ קטגוריות. כל חיפוש מכיל בתוכו גם את הקטגוריה לחיפוש.

Category_to_shop- טבלה המקשרת בקשר של רבים לרבים את Shops ו-Categories. זאת מכיוון שכל קטגוריה יכולה להימכר בכמה חנויות, וכל חנות יכולה למכור כמה קטגוריות.

2.6 תוכן הפרויקט

כאמור, הפרויקט שלנו מחולק לשכבות. נציג כאן לפי השכבות:

מחלקות ב-Entities

אלו המחלקות שמחוברות לSQL, וכך נשמרים הנתונים.

Entities למעשה הם אותן המחלקות כמו ב-DAL. לכן נביא רק את מחלקות ה-Entities ולא גם את של ה-DAL.

מחלקת חנות:

```
public class ShopDTO
    {
        public int codeShop { get; set; }
        public string nameShop { get; set; }
        public string passwordShop { get; set; }
        public string phoneShop { get; set; }
        public string mailShop { get; set; }
        public double longitude { get; set; }
        public double latitude { get; set; }
        public string fromHour { get; set; }
        public string toHour { get; set; }
        public string addressString { get; set; }
        public virtual List<CategoryDTO> Categories { get; set; }
       }
```

מחלקת משתמש:

```
public class UserDTO
    {
        public int codeUser { get; set; }
        public string nameUser { get; set; }
        public string phoneUser { get; set; }
        public string mailUser { get; set; }
        public string passwordUser { get; set; }
        public virtual List<SearchDTO> Searches { get; set; }
       }
```

מחלקת חיפוש:

```
public class SearchDTO
    {
        public int codeSearch { get; set; }
        public int codeUser { get; set; }
        public string nameProduct { get; set; }
        public int status { get; set; }
        public int? codeShop { get; set; }
        public int codeCategory { get; set; }
       }
```

מחלקת קטגוריה:

```
public class CategoryDTO
    {
        public int codeCategory { get; set; }
        public string nameCategory { get; set; }
       }
```

מחלקות ב-BL:

BL היא השכבה הלוגית.

מחלקות ה-Casting:

אלו מחלקות שכל מחלקה ב– Entities יכולות להמיר למחלקה ב-DAL וכן להיפך. נביא כאן דוגמה של מחלקה כזו:

```
class ShopCast
    {
        public static ProjectEntities db = new ProjectEntities();
        //Convert from shop to shopDTO
        public static ShopDTO GetShopDTO(Shop shop)
        {
            var cat = CategoryCast.GetCategoriesDTO(shop.Category_to_shop.Select(s =>
s.Category).ToList());
            return new ShopDTO()
            {
                codeShop = shop.codeShop,
                nameShop = shop.nameShop,
                passwordShop = shop.passwordShop,
                phoneShop = shop.phoneShop,
                longitude=shop.longitude,
                latitude=shop.latitude,
                fromHour=shop.fromHour,
                toHour=shop.toHour,
                mailShop=shop.mailShop,
                addressString=shop.addressString,
                Categories=cat
            };
        }
        //Convert from ShopDTO to Shop
        public static Shop GetShop(ShopDTO shop)
        {
            return new Shop()
            {
                codeShop = shop.codeShop,
                nameShop = shop.nameShop,
                passwordShop = shop.passwordShop,
                phoneShop = shop.phoneShop,
                mailShop=shop.mailShop,
                latitude=shop.latitude,
                longitude=shop.longitude,
                fromHour=shop.fromHour,
                toHour=shop.toHour,
                addressString=shop.addressString
            };
        }
        //Conver a list of shops to list of shopDTOs
        public static List<ShopDTO> GetShopsDTO(List<Shop> shops)
        {
            List<ShopDTO> shopDTOs = new List<ShopDTO>();
            foreach (var item in shops)
```

```
    {
        shopDTOs.Add(new ShopDTO()
        {
            codeShop = item.codeShop,
            nameShop = item.nameShop,
            passwordShop = item.passwordShop,
            phoneShop = item.phoneShop,
          longitude=item.longitude,
          latitude=item.latitude,
            mailShop=item.mailShop,
            fromHour=item.fromHour,
            toHour=item.toHour,
            addressString=item.addressString
        });
    }
    return shopDTOs;
}
   }
```

מחלקת Shop :

מטפלת בכל הלוגיקה הקשורה לחנויות:

```
public class Shops
{
    public static ProjectEntities db = new ProjectEntities();
    //Register
    public async static Task<WebResult<LoginData<ShopDTO>>> Register(ShopDTO shopDto, Uri request)...
    //Login
    public static async Task<WebResult<LoginData<ShopDTO>>> Login(string mail, string password, Uri requestUri)...
    //Update shop with categories
    public static WebResult<ShopDTO> Update(ShopDTO shopDTO)...
    //Send email for request category
    public static WebResult<string> SendEmailForNewCategory(string categoryName, [UserLogged] ShopDTO shopDTO)...
    //Returns the searches that found in that shop
    public static WebResult<SearchesForShop> getSearchesForShop([UserLogged] ShopDTO shopDTO)...

    //Returns the categories for choosing
    public static WebResult<List<CategoryDTO>> GetAllCategories()...

    //Logout
    public static WebResult<string> Logout([UserLogged] ShopDTO shopDTO)...
    //Saves who is the shop now
    private static async Task<string> GetTokenDataAsync(string username, string password, Uri req)...
}
```

מחלקת Search :

מטפלת בכל הלוגיקה הקשורה לחיפושים של המשתמשים:

```
public class Searches
{
    public static ProjectEntities db = new ProjectEntities();
    //Returns the categories for choosing
    public static WebResult<List<CategoryDTO>> GetCategories()...
    //Create search
    public static WebResult<SearchDTO> Create(SearchDTO searchDTO)...
    //Delete search- status changes to 2
    public static WebResult<SearchDTO> Delete(int code)...
    //Search is found- user bought the product
    public static WebResult<SearchDTO> Found(int codeSearch, int codeShop)...
    //Returns history of the searches, even thouse the user found
    public static WebResult<List<SearchDetailsForUser>> GetHistory(string uuid)...
    //Returns user searches that have not yet been found
    public static WebResult<List<SearchDetailsForUser>> GetHistoryNotFound(string uuid)...
    //Returns user searches that have been found
    public static WebResult<List<SearchDetailsForUser>> GetHistoryFound(string uuid)...
    //Returns all stores that sell a particular category
    public static WebResult<List<ShopDetailsForUsers>> GetShopsForCategory(int codeCategory)...

}
```

Helpers:

מחלקות עזר.

מחלקות ב-WebService:

זוהי השכבה שמתקשרת עם צד ה-client, מקבלת נתונים ומחזירה, מחכה לקריאות.

מחלקת ShopsController, מאזינה לקריאות מהאתר לחנויות ומפנה לפונקציות המתאימות ב-BL:

```
public class ShopsController : ApiController
    {
        [Route("Register")]
        [HttpPost]
        public async Task<IHttpActionResult> Register(ShopDTO shop)
        {
            return Ok(await Shops.Register(shop, Request.RequestUri));
        }

        [Route("Login")]
        [HttpGet]
        public async Task<IHttpActionResult> Login(string mail, string password)
```

```
        {
            return Ok(await Shops.Login(mail, password, Request.RequestUri));
        }
        //Who is the current shop
        [Route("getLoggedShop")]
        [HttpGet]
        public IHttpActionResult GetLoggedShop([UserLogged] ShopDTO shopDTO)
        {
            return Ok(shopDTO);
        }

        [Route("Logout")]
        [HttpGet]
        public IHttpActionResult Logout([UserLogged] ShopDTO shopDTO)
        {
            return Ok(Shops.Logout(shopDTO));
        }

        [Route("Update")]
        [HttpPost]
        public IHttpActionResult Update(ShopDTO shop)
        {
            return Ok(Shops.Update(shop));
        }

        //Get searches for shop, for statistics
        [Route("GetSearches")]
        [HttpGet]
        public IHttpActionResult GetSearches([UserLogged] ShopDTO shopDTO)
        {
            return Ok(Shops.getSearchesForShop(shopDTO));
        }

        [Route("GetAllCategories")]
        [HttpGet]
        public IHttpActionResult GetAllCategories()
        {
            return Ok(Shops.GetAllCategories());
        }

        //Request for new category
        [Route("NewCategory")]
        [HttpGet]
        public IHttpActionResult NewCategory(string newCategory, [UserLogged] ShopDTO
shopDTO)
        {
            return Ok(Shops.SendEmailForNewCategory(newCategory, shopDTO));
        }
       }
```

כך, יש גם מחלקת SearchesController- מאזינה לקריאות של משתמשים מהאפליקציה.

3. מדריך למשתמש

3.1 הוראות

מדריך לבעל/ת החנות:

שלום לך בעל/ת חנות יקר/ה!

אנו שמחים שבחרת להצטרף למאגר שלנו.

ראשית, עליך להירשם, עם כל הפרטים על חנותך. הפרטים החשובים ביותר: הכתובת, והקטגוריות הנמכרות בחנות. אם אין ברשימת הקטגוריות הכללית את הקטגוריה המסוימת שהחנות מוכרת- ניתן לשלוח מייל בקשת הוספת קטגוריה, ונענה בהקדם.

לאחר מכן, בכל זמן ניתן לעדכן ולשנות.

בדף הבית, תוכל לראות את הנתונים שלך, וכן תרשים של אחוזי המכירות שלך לפי הקטגוריות (שנמכרו דרך אתר זה). כך אפשר לדעת אילו קטגוריות הן הנצרכות ביותר והנמכרות ביותר.

מדריך למשתמש:

שלום לך משתמש/ת יקר/ה!

אחרי שהורדת את האפליקציה, הורד את פרטיך האישיים, ומיד תוכל להתחיל לחפש: במסך הבית כתוב טקסט של שם המוצר, בחר מתוך רשימת הקטגוריות באיזו קטגוריה המוצר, בחר מאיזה מרחק מהחנות תקפוץ לך ההודעה על המציאה, ו—הפעל חיפוש. מעתה, ברגע שתעבור קרוב לחנות שמוכרת את המוצר- תקפוץ לך הודעה מתאימה. תוכל להחליט אם אינך קונה עכשיו- ולהמשיך את החיפוש לפעם אחרת, או לסמן שקנית, ואז החיפוש כבר לא יהיה מופעל.

אפשר להפעיל כמה חיפושים בו זמנית.

כמו כן, אפשר לראות את היסטוריית החיפושים על פי סינונים שונים.

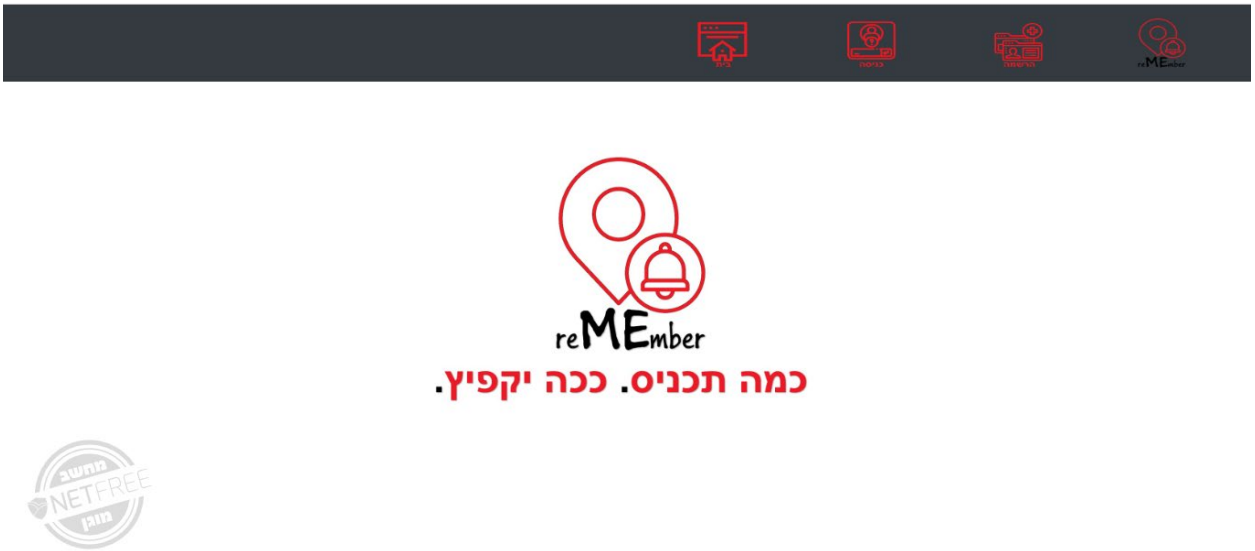
כמובן, תוכל לשנות את הפרטים האישיים שלך בכל עת.

כך לא תשכח לקנות את המוצרים החשובים לך, ותחסוך זמן.

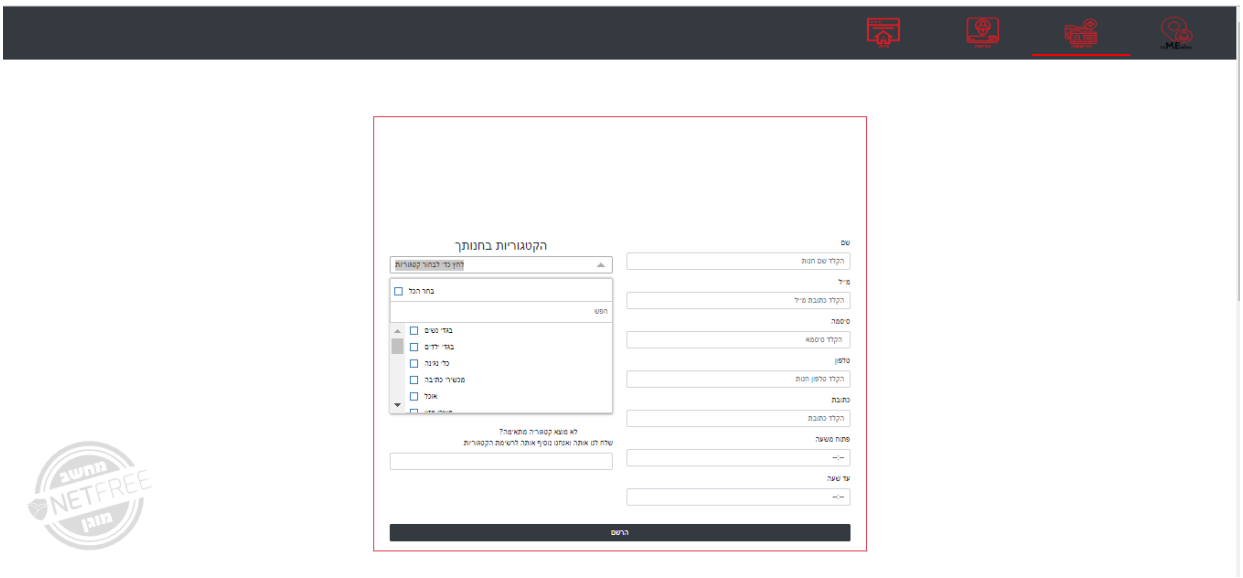
3.2 מסכים

מסכים לבעלי חנויות- באתר:

דף כניסה לאתר

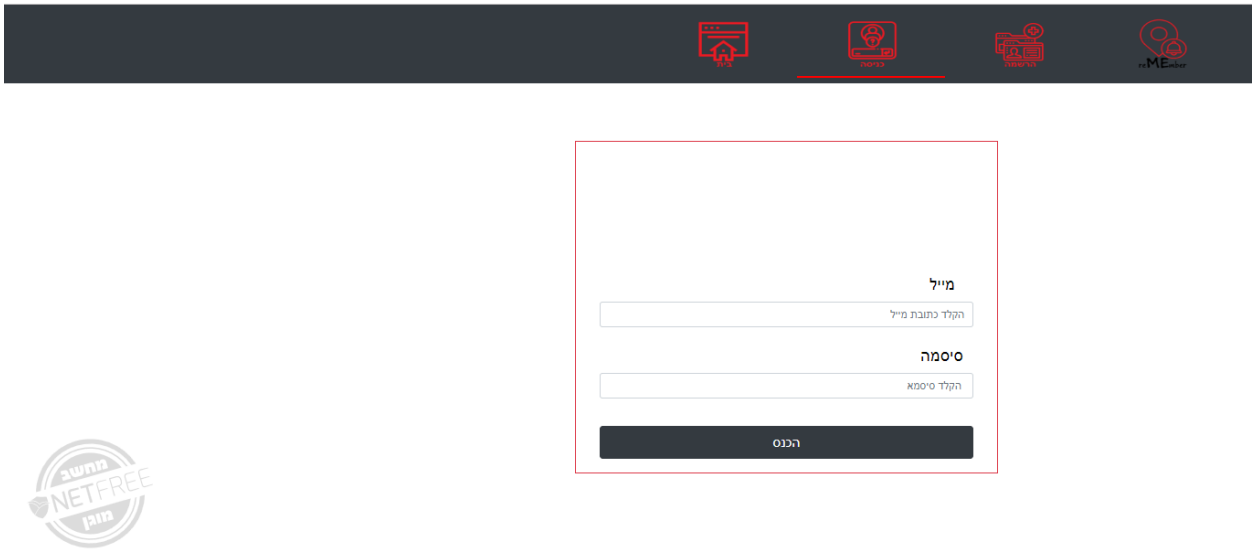


דף REGISTER

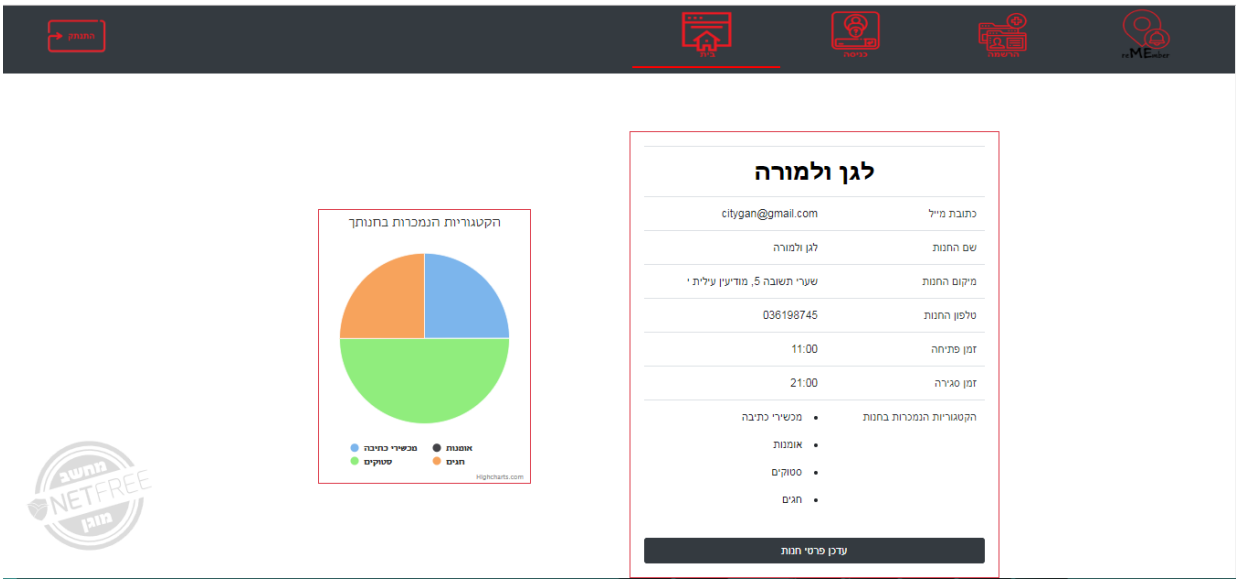


דף LOGIN

בעל חנות נכנס לאתר עם שם וסיסמה.



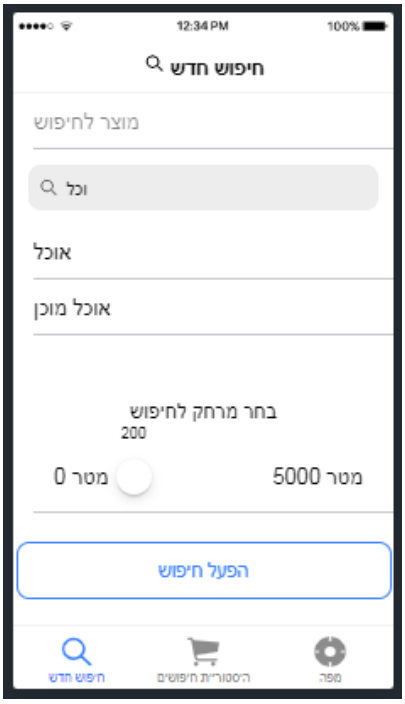
דף בית



מסכים למשתמשים- באפליקציה:

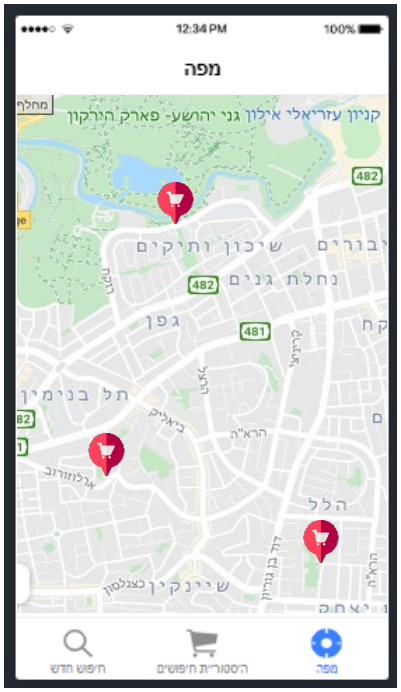
דף הפעלת חיפוש

המשתמש מכניס מוצר לחיפוש בוחר מהרשימה קטגוריה בוחר מרחק ומפעיל חיפוש.

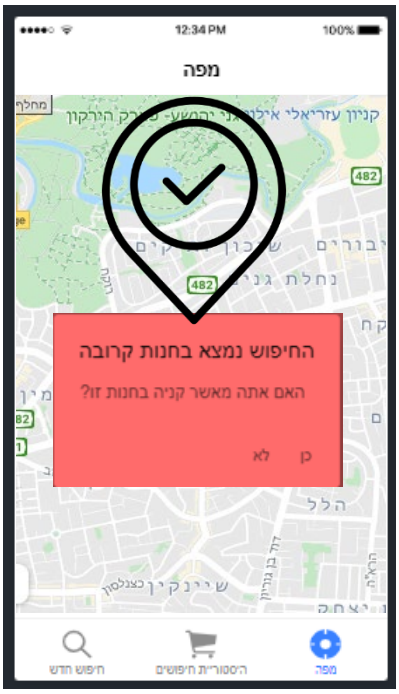


דף התוצאה

כאשר המשתמש עומד על מרקר זה מציג לו את פרטי החנות הנוכחית.

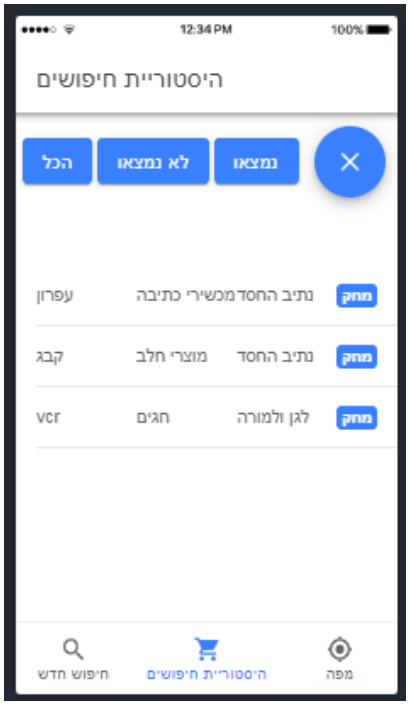


דף מציאת החיפוש



דף היסטורית חיפושים

המשתמש צופה בהיסטוריית חיפושים ושם יכול לסנן את הרשימה לפי קריטריונים מסוימים.



reMEmber

4. סיכום ומסקנות

סיום הוא עניין מרגש, וכאן- יצאנו מהפרויקט עם רווחים גדולים.

מה הקנה לנו הפרויקט:

- ניתוח מערכות יעיל- כשנגשנו לתכנון פרויקט הגמר, שרטטנו אותו תחילה על נייר, תכננו מחלקות, מסכים ורכיבים נוספים. פעילות זו הרחיבה את ידיעותינו בתחום ניתוח מערכות ותכנון פרויקטים בהיקף גדול.
- חשיבה לוגית גבוהה
- אלתור ומציאת פתרונות לבעיות שונות.
- למידה עצמית של נושאים רבים מתוך אתרי אינטרנט.
- התחברות למשאבים חיצוניים- API.
- ידע, איך אמור להיראות אתר, איך אמורה להיראות אפליקציה.
- ידע מעמיק ב- #C, Angular ו- Ionic.
- נסיון בGITHUB
- עמידה בלוחות זמנים.

למדנו לפתח מוצר- מתחילה ועד סוף.

5. נספחים

הפונקציה שמקבלת את המיקום, ומחזירה למשתמש האם יש חנות קרובה עבור אחד החיפושים שלו:

```
// Help function
        private static double rad(double x)
        {
            return x * Math.PI / 180;
        }
        // Returns distance between two locations, in meters
        private static double getDistance(double p1Lat, double p1Long, double p2Lat,
double p2Long)
        {
            int R = 6378137;// Earth’s mean radius in meter
            double dLat = rad(p2Lat - p1Lat);
            double dLong = rad(p2Long - p1Long);
            double a = Math.Sin(dLat / 2) * Math.Sin(dLat / 2) +
                Math.Cos(rad(p1Lat)) * Math.Cos(rad(p2Lat)) *
                Math.Sin(dLong / 2) * Math.Sin(dLong / 2);
            double c = 2 * Math.Atan2(Math.Sqrt(a), Math.Sqrt(1 - a));
            double d = R * c;
            return d;// returns the distance in meter
        }

        //Checks whether this location has a nearby store for one of the categories
        public static WebResult<ShopDetailsForUsers> CheckDistance(UserIdWithLocation
userIdWithLocation)
        {
            double lat = userIdWithLocation.Lat;
            double lng = userIdWithLocation.Lng;
            var codeUser = db.Users.First(f => f.passwordUser ==
userIdWithLocation.Uuid).codeUser;
            foreach (var search in db.Searches)
            {
                //only if the search is from this user and its status is 0, to find
                if (search.codeUser == codeUser && search.status == 0)
                {
                    foreach (var shop in db.Shops)
                    {
                        //if distance is less than 1000 meter
                        if (getDistance(lat, lng, shop.latitude, shop.longitude) < 1000)
                        {
                            //if there is the category that the user search in that shop
                            if
(Casting.ShopCast.GetShopDTO(shop).Categories.FirstOrDefault(f => f.codeCategory ==
search.codeCategory) != null)
                            {
                                return new WebResult<ShopDetailsForUsers>()
                                {
                                    Status = true,
                                    Message = "found shop",
                                    Value = new ShopDetailsForUsers()
                                    {
                                        AddressString = shop.addressString,
                                        NameShop = shop.nameShop,
```

```
                                PhoneShop = shop.phoneShop,
                                FromHour = shop.fromHour,
                                ToHour = shop.toHour,
                                Latitude = shop.latitude,
                                Longitude = shop.longitude
                            }
                        };
                    }

                }
            }
        }
    }
    return new WebResult<ShopDetailsForUsers>()
    {
        Status = false,
        Message = "not found close shop",
        Value = null
    };
}
```

6. שונות

7. ביבליוגרפיה

Stack overflow

http://stackoverflow.com

Google developers- API google places

https://developers.google.com/places/web-service/autocomplete

Bootstrap

https://getbootstrap.com/docs/4.3/getting-started/introduction

Angular

https://angular.io

Ionic

https://ionicframework.com

w3schhols

https://www.w3schools.com

GitHub

https://github.com