



מגישות:

חסיה גרינהוט

לאה ווייס

מנחה:

גב' מלצר

תש"פ – 2019



## תודות

בראש ובראשונה תודה לבורא עולם על כל הטוב אשר גמלנו ועל הסיעתא דשמיא המרובה שליוותה אותנו בכל שנות הלימודים ובפרט במהלך הפרויקט, החל מהעלאת הרעיון ועד גמר הביצוע.

- ❖ להנהלת סמינר מעלות שליווה את צעדינו לאורך כל השנים, ודאג לנו למיטב הכלים והאפשרויות ללמוד, לגדול ולהצליח. ועימו במלאכה הגב' נ. הורביץ, מרכזת המגמה, אשר לא חסכה כל מאמץ ליעל את המסלול ולהופכו למקצועי ומכובד, על המסירות, ההדרכה וההשקעה הרבה.
- ❖ למרכזת הפרויקטים הגב' יהודית מלצר ולכלל המורות על מסירות והשקעה בכל שנות הלימודים.
- ❖ למנחות על העזרה האינסופית, העידוד והתמיכה בשמחה ובמאור פנים.
- ❖ לכל חברותינו במסלול על שעות של שמחה, עידוד ואוירה טובה בכל הזמנים יחד.
- ❖ לבני משפחותינו האהובים תודה שהייתם שם בשבילינו בכל התחומים ובכל הזמנים.



## תוכן ענינים

5	1	מדריך למשתמש
	1.1	כניסה
	1.1.1	כניסת משתמש
	1.1.2	דף הבית
	1.1.3	אזור האישי
	1.2	אזור ניהול
	1.2.1	ניהול לקוחות
8	2	מדריך למתכנת
	2.1	צד שרת
	2.2	צד לקוח
9	3	עקרונות התכנון והבניה :
	3.1	מידול
	3.1.1	מתודולוגית Agile
	3.1.2	אבטחה
10	4	מבנה הפרויקט
	4.1	תיאור הפרויקט
	4.1.1	טבלאות
	4.1.2	תקשורת עם databasen
	4.2	שכבת האפליקציה
	4.2.1	צד Client
	4.2.2	צד Server
20	5	פונקציונאליות
21	6	סיכום ומסקנות
22	7	ביבילוגרפיה



## מבוא

בבואינו להגיש פרויקט גמר פנתה אלינו צלמת סטודיו בבקשת שדרוג והנגשת האתר ללקוחות. רצינו להקל על תהליך ההתקשרות בין הצלם/ת ללקוחותיו ע"י חשבון אישי באתר. כל לקוח חדש פותח חשבון אישי ע"י הכנסת שם משתמש וסיסמא. לאחר יום צילומים, המנהל מעלה לאזור האישי של הלקוח את התמונות שצולמו, ומשם הלקוח יכול לבצע פעולות שונות. ביכולתו של הלקוח לדפדף בין התמונות שלו, לבחור את התמונות שיודפסו, לבחור את סוג האלבום שיודפס ואף להוריד את התמונות למחשבו האישי. לאחר בחירת תמונות ואלבום להדפסה ע"י הלקוח יופיע אצל המנהל משימה חדשה ואז הוא יוכל לראות את בחירת הלקוח. ניתן להיכנס לאתר ולצפות בתמונות ואלבומים המוצגים כדוגמאות ולקבל השראה אף ללא חשבון אישי.

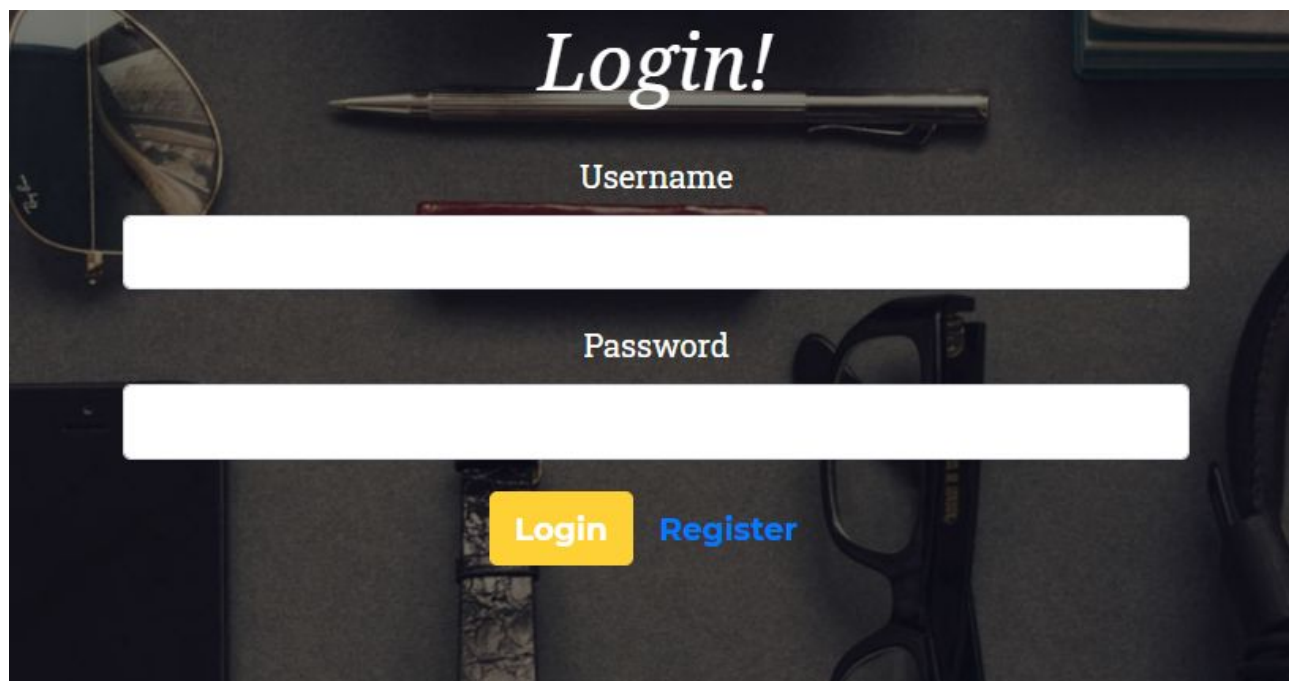




## מדריך למשתמש

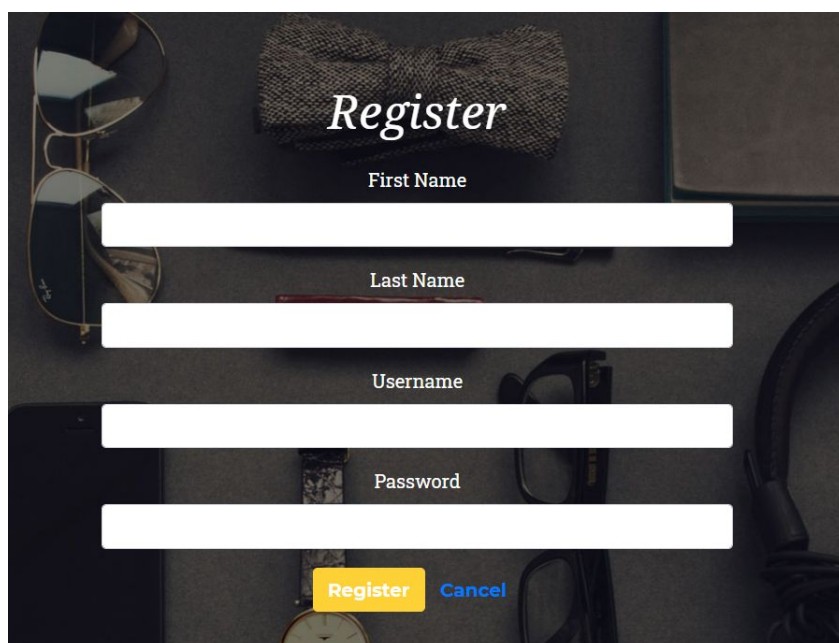
### 1.1 כניסה

#### 1.1.1 כניסת משתמש



The image shows a login form titled "Login!" in a large, white, serif font. Below the title are two white input fields. The first field is labeled "Username" and the second is labeled "Password". Below the input fields are two buttons: a yellow "Login" button and a blue "Register" button. The background is a dark, textured surface with various objects like a pen, glasses, and a watch.

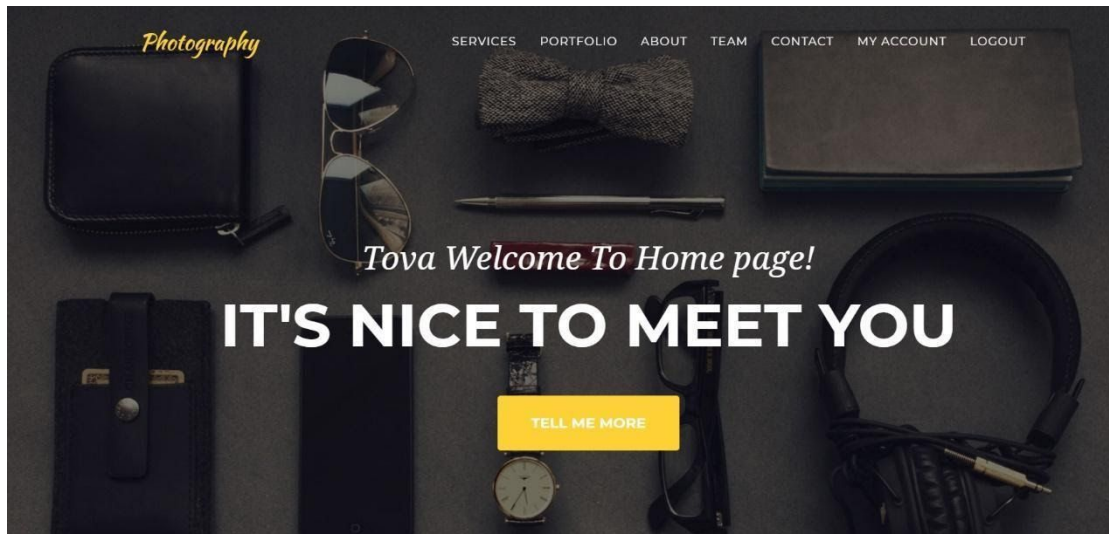
במידה והמשתמש הינו משתמש רשום, על המשתמש להכניס שם משתמש וסיסמא כדי שיוכל להתחבר למערכת על ידי לחיצה על ה[login](#).-



The image shows a register form titled "Register" in a large, white, serif font. Below the title are four white input fields. The first field is labeled "First Name", the second "Last Name", the third "Username", and the fourth "Password". Below the input fields are two buttons: a yellow "Register" button and a blue "Cancel" button. The background is a dark, textured surface with various objects like a bow tie, glasses, and a watch.

במידה והמשתמש אינו רשום במערכת, עליו להירשם על ידי לחיצה על ה-**register** ובדף שיעלה יהיה עליו להכניס את פרטיו האישיים על מנת להתחבר למערכת כמשתמש רשום.

## 1.1.2 דף הבית



בחלקו העליון של המסך מופיע התפריט.

בלחיצה על **services** יופיעו סוגי החבילות הקיימות.

בלחיצה על **portfolio** ניתן לעיין בקטלוגים השונים ולקבל השראה.

בלחיצה על **about** יופיע מידע על חברת הצילום.

בלחיצה על **contact** יופיע מידע שבו יהיה ניתן ליצור קשר.

## 1.1.3 אזור האישי

על מנת להקל ולייעל התקשורת בין הצלמת ללקוח.

אחרי רישום לאתר, ללקוח יש אפשרות להכנס לאזור האישי שלו, באיזור האישי הוא יוכל לבצע הזמנות, לצפות בהזמנות קיימות ולהוריד את כל התמונות שהזמין כולל צפיה באלבומים המוכנים. פעילות זו משפרת את התקשורת בין הצלמת ללקוח וגורמת לחויית הצילום להיות מושלמת.

### 1.1.3.1 צפיה באלבומים ותמונות

לאחר עריכת יום הצילומים ללקוח מסוים, הצלמת מעלה את התמונות (דרך אזור הניהול) לאזור האישי של הלקוח על מנת שיהיה לו אפשרות לראות אותם.

### 1.1.3.2 בחירת תמונות

כאשר התמונות כבר הועלו לאיזור האישי של הלקוח על הלקוח יהיה לבחור את התמונות שאותם הוא רוצה, ואת התמונות שהוא רוצה שיכנסו לאלבום.



### 1.1.3.3 הורדת תמונות

באזור האישי של הלקוח, תהיה אפשרות להוריד את כל התמונות באיכות נמוכה למחשבו האישי.

## 1.2 אזור ניהול

### 1.2.1 ניהול לקוחות

#### 1.2.1.1 צפיה בלקוחות האתר

בדף זה המנהל רואה את כל פרטי הלקוחות שלו.

#### 1.2.1.2 ניהול הזמנות ללקוחות

בלחיצה על כפתור ה orders שליד כל לקוח המנהל יועבר לאזור ניהול הזמנות עבור הלקוח באזור זה המנהל יוכל לעשות את הפעולות הבאות:

- ❖ יצירת הזמנה חדשה
- ❖ העלאת תמונות להזמנה קיימת ברזולוציה נמוכה
- ❖ צפיה בתמונות אותן בחר הלקוח
- ❖ העלאת תמונות שנבחרו ע"י הלקוח ברזולוציה גבוהה



## מדריך למתכנת

### 2.1 צד שרת

הפרויקט נכתב בטכנולוגיות חדשות תוך שימת דגש על ארכיטקטורה נכונה והטמעת Design patterns במקומות המתאימים לכך.

- ❖ שפת תכנות - #C מעל .Net Framework.
- ❖ בניית WebAPI להנגשת הנתונים ממסדי הנתונים לצד הלקוח, הדאטה מוחזר לצד הלקוח בפורמט Json ובפרוטוקול HTTPS/HTTP.

### 2.2 צד לקוח

- ❖ ספרייה ראשית - Angular8 זו הגרסה העדכנית ביותר של ה Framework של גוגל לבניית יישומים מורכבים בדפדפן.
- ❖ **שפת תכנות - Typescript המבוססת על javascript** ומאפשרת כתיבה יפה ונקייה של קוד ויותר אפשרויות תכנותיות.
- ❖ עיצוב UI - בניית מסכים ב HTML ושימוש בספריית bootstrap ע"מ להנגיש את האתר למגוון מכשירים ו CSS-נוספים להרחבת של העיצוב.

3



## עקרונות התכנון והבניה :

### 3.1 מידול

את צד הסרבר חלקנו לשלוש שכבות:

**שכבה ראשונה:** שכבת ה-DAL המכילה את ה-entity framework ואת החיבור ל-DB.

**שכבה שנייה:** שכבת ה-BL מכילה את הפונקציות הנדרשת לביצוע, וכן מבצעת שליפות, אחסונים ועדכונים.

**שכבה שלישית:** שכבת ה-API מכילה את הקונטרולים של צד השרת המכילים פניות לשכבה השנייה.

### 3.1.1 מתודולוגית Agile

במהלך פיתוח הפרויקט עבדנו במתודולוגית Agile ע"מ למקסם את היכולת שלנו לדלוור את הפרויקט בצורה יעילה ומהירה.

#### 3.1.1.1 שימוש במערכת ניהול גרסאות GIT

פיתחנו את הפרויקט תוך שימוש ב-GIT לניהול גרסאות ע"מ לשמור על סדר והיסטורית שינויים בפרויקט, כמו כן התהליך הקל עלינו בתקשורת אחת עם השניה ואיפשר סינכרון שינויים בינינו בצורה יעילה ותקינה, כמו כן המהלך הכין אותנו לעבודה בתעשייה.

#### 3.1.1.2 שימוש בשרות אחסון מבוסס רשת GitHub

השתמשנו בשרות אחסון מבוסס רשת GitHub ע"מ לאפשר לשנינו לעבוד כל אחת מביתה ובמחשב שלה על העדכונים האחרונים.

לאורך כל הדרך נצמדנו לעקרונות המידול בשכבות-דבר המאפשר קוד נקי ויפה המאפשר לכל מתכנת להבין ולהתמצא בקוד. כמו כן, כאשר יש צורך בשינוי כלשהו, ע"י המידול ניתן לעשות אותו ביתר קלות ע"י ביצוע שינויים רק בשכבה אחת בלי לפגוע ולשנות את שאר השכבות כלל.

### 3.1.2 אבטחה

בכל שלבי הפרויקט ניתן דגש מיוחד על תקינות ואבטחת המידע. רק משתמשים רשומים יכולים לגשת למערכת, המערכת תוודא בכל גישה אליה שהמשתמש מוכר ורשום למערכת ורק אז יוכל המשתמש לבצע את הפעולות הרצויות.

המידע שהוכנס לבסיס הנתונים עובר בדיקות תקינות הן בצד הלקוח והן בצד השרת על מנת לוודא שלא מוכנס מידע שגוי. בדיקות תקינות מונעות במידה רבה בעיות נוספות של אבטחת מידע כגון (Buffer Overflow) - דריסת זיכרון. דריסת זיכרון הוא מצב בו קלט לתוכנית נכתב מחוץ למקום שיועד לו, ולכן דורס חלקים אחרים בזיכרון מה שמשפיע על המשך פעולת התוכנית. בדיקת תקינות הקלט היא דרך פשוטה וקלה להפחית את הסיכון לפריצת המערכת ולשלון ממנה פרטים אישיים של משתמשים, או מידע של מנהל המערכת.



## מבנה הפרויקט

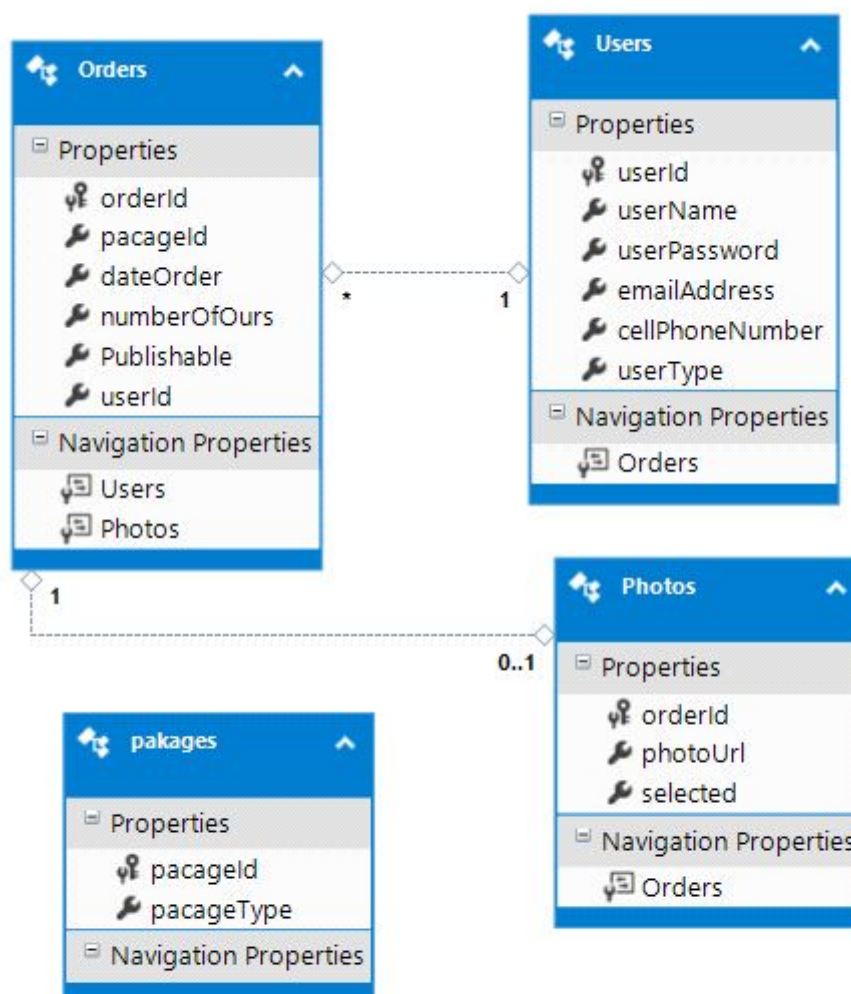
### 4.1 תיאור הפרויקט

בסעיף זה נפרט את השכבות מהם מורכב הפרויקט.

שכבת ה-DAL

#### 4.1.1 טבלאות

להלן תרשים הטבלאות במסד הנתונים:



בבניית המסד הושקעה מחשבה רבה ותכנון. על מנת להפיק יעילות מקסימלית. מהאפשרויות שמציעה ולקבל נתונים תקינים והגיוניים DB.

מסד הנתונים מכיל טבלאות עם קשרי גומלין – דבר שמאפשר מעקב אחר חוקיות הנתונים ותאימות בין כל החלקים

כמו"כ מכיל המסד שדות מפתח – שמונעים כפילויות אסורות ומזרזים את שליפת הנתונים.

במסד קיימות טבלאות עבור: משתמשים, הזמנות, תמונות וחבילות.

להלן פירוט של שדות הטבלאות וסוגיהם:

#### טבלת Packages - טבלת חבילות

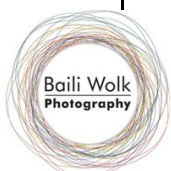
שם העמודה	סוג	הסבר
PackageId	Int	קוד חבילה
PackageType	Int	סוג חבילה

#### טבלת Pictures - טבלת תמונות

שם העמודה	סוג	הסבר
orderId	Int	קוד הזמנה
photoUrl	string	כתובת תמונה
selected	bit	תמונה שנבחרה

#### טבלת Users - טבלת משתמשים

שם העמודה	סוג	הסבר
userId	int	קוד משתמש
userName	string	שם משתמש
userPassword	string	סיסמת משתמש
emailAddress	string	כתובת מייל
cellPhoneNumber	string	מס' פל'



userType	string	סוג משתמש
----------	--------	-----------

#### טבלת orders - טבלת הזמנות

שם העמודה	סוג	הסבר
orderId	int	קוד הזמנה
packageId	int	קוד חבילה
dateOrder	Date/time	תאריך הזמנה
NumberOfOurs	string	מס' שעות
publishable	bit	ניתן לפרסום
userId	int	קוד משתמש

#### 4.1.2 תקשורת עם databasen

את הקישור ל database ואת התקשורת אתו עשינו באמצעות Entity framework סביבת עבודה נוחה ופשוטה אשר בונה אוטומטית לכל טבלה class החופף בשדותיו לשדות הטבלה אותה מייצג. באופן זה קל מאוד לבצע פונקציונאליות הקשורה ל-database.

עם ה DB תקשרנו בשפת LINQ אשר דומה בתחבירה לשפת SQL ומאפשרת גישה נוחה אל נתוני ה-database דבר שאפשר לנו להתרכז בלוגיקה ובמורכבות ויעילות האלגוריתמים מבלי לבזבז זמן על התעסקות עם קישור מסובך ל-database.

התקשורת הישירה מול ה DB נעשתה בשכבה הראשונה - שכבת ה DAL כדלהלן:

שכבת ה BL :

בשכבה זו נמצאות כל הפונקציות הפונות ל DB התקשורת ל DB נעשית באמצעות שפת C#Linq.

עבור כל טבלה יצרנו מחלקה נפרדת כדי לשמור על סדר ונקיות הקוד.

רשימת המחלקות ב BL

**UserRepository.cs**

מחלקה זו מטפלת בכל הלוגיקה הקשורה למשתמשים, כגון: כניסה, הוספת משתמש חדש, שליפת רשימת משתמשים.

**OrdersRepository.cs**



מחלקה זו מטפלת בכל הלוגיקה הקשורה לטיפול בהזמנות עבור לקוחות. כגון: שמירת הזמנה, עידכון הזמנה.

### PackageRepository.cs

מחלקה זו מטפלת בכל הלוגיקה הקשורה לחבילות. כגון: הוספת חבילה, עידכון חבילה קיימת.

### ImageRepository.cs

מחלקה זו מטפלת בכל הלוגיקה הקשורה לתמונות. כגון: הוספת תמונה חדשה, הסרת תמונה, עידכון סטטוס תמונה.

כפי הנכתב לעיל התקשורת עם DB נעשיית בשפת Linq להלן כמה דוגמאות קוד:

דוגמא לשליפת לקוח:

```
// דוגמא לשליפה של משתמש
public Users GetUser(string email, string password)
{
    try
    {
        return Data.DB.Users.FirstOrDefault(u => u.emailAddress == email && u.userPassword == password);
    }
    catch (Exception e)
    {
        throw new Exception(e.Message);
    }
}
```

דוגמא להוספת לקוח:

```
public static void add(Users user)
{
    Data.DB.Users.Add(user);
    Data.DB.SaveChanges();
}
```



דוגמא לשליחת מייל:

```

16 public HttpResponseMessage SendContact(ContactFormModel contactForm)
17 {
18     SendMail sendMail = new SendMail();
19     string result = sendMail.SendEmailAsync(contactForm);
20     if(result == "succses")
21     {
22         return Request.CreateResponse(HttpStatusCode.OK);
23     }
24     else
25     {
26         return Request.CreateErrorResponse(HttpStatusCode.ExpectationFailed, result);
27     }
28 }

```

דוגמא להעלאת תמונה ל-DB.

```

public HttpResponseMessage UploadImage()
{
    string imageName = null;
    var httpRequest = HttpContext.Current.Request;
    var postedFile = httpRequest.Files["Image"];
    imageName = new string(Path.GetFileNameWithoutExtension(postedFile.FileName).Take(10).ToArray().Replace(" ", "-"));
    imageName = imageName + DateTime.Now.ToString("yy-mm-dd") + Path.GetExtension(postedFile.FileName);
    var filePath = HttpContext.Current.Server.MapPath("~/Image/" + imageName);
    postedFile.SaveAs(filePath);
    using (PhotoGiftEntities db = new PhotoGiftEntities ())
    {
        Image image = new Image()
        {
            ImageCaption = httpRequest["ImageCaption"],
            ImageName = imageName
        };
        db.Image.Add(image);
        db.SaveChanges();
    }
    return Request.CreateResponse(HttpStatusCode.Created);
}

```

דוגמא למחיקת לקוח:

```

public void Delete(Users user)
{
    Users user1 = Data.DB.Users.Where(u => u.userId == user.userId).First();
    Data.DB.Users.Remove(user1);
    Data.DB.SaveChanges();
}

```

## 4.2 שכבת האפליקציה

בשכבה זו יש את כל הפונקציונאליות של הclient וכן את כל הפונקציונאליות של ה-server אשר לא דורשת התנהלות ישירה מול ה-DB.

### 4.2.1 צד Client

את צד הקליינט כתבנו באמצעות אנגולר 8 – תשתית פיתוח שבשונה מכל תשתית פיתוח אחרת מכתיבה לנו גם איך לפתח, מעבר לטכנולוגיה היא מכתיבה לנו צורת פיתוח לצד לקוח המתבססת על Typescript. Typescript זוהי שפת פיתוח בצד לקוח הנותנת לנו כלים



עבודה עם types ואז ממילא ניתן לבצע קומפילציה. בשלב זה typescript מבצע המרה של הקוד לקוד JS מכיון שרוב הדפדפנים אינם תומכים כרגע בתקן החדש של שפה זו.

את דפי HTML עיצבנו באמצעות Bootstrap, אשר גם הם מאד פופולארים בקרב עולם האינטרנט. רכיבי Bootstrap פועלים עם רכיבי ממשק משתמש מקיפים ומודרניים ברחבי האינטרנט, ניידים ושולחן העבודה. יש להם מגוון רחב של קווי עיצוב שניתנים להתאמה אישית ומועדפת למשתמש. וכן הם מספקים פתרונות למגוון מכשירים שונים ע"י כך שהתצוגה משתנה דינמית לפי גודל המסך.

מתכנתי אנגולר יצרו עבורנו את האפשרות ליצור אפליקציה (SPA (Single Page Application) כלומר - רק חלק מסוים בדף מתרענן ומתחלף וכל שאר החלקים נשארים אותו הדבר. דבר זה מאפשר גלישה מהירה וחלקה אשר משפרת את חווית המשתמש (user experience).

צד הלקוח מחולק לקומפוננטות שונות, קומפוננטה (component) - רכיב המיועד לתצוגה המכיל את כללי התצוגה ואת הניהול שלה והיא מורכבת **מכמה חלקים**:

- 1) קבצי Css - בהם נכתב העיצוב של הקומפוננטה.
- 2) קבצי Html - בהם נכתב קוד הנראות של הקומפוננטה.
- 3) קבצי ts - בהם נכתב הקוד והלוגיקה של הקומפוננטה בשפת Typescript.
- 4) קבצי spec.ts - בהם קוד הקומפוננטה נבדק.

HTML - שאומר לAngular איזה תוכן להציג על המסך עבור קומפוננטה זו עפ"י הURL - הנובי שרשום בה.

ה template Url מכיר את הקומפוננטה ואת כל השדות הכתובים בה.

לדוגמא:

```
import { AlertService } from '../_services
```

```
})Component@
```

```
, 'selector: 'alert
```

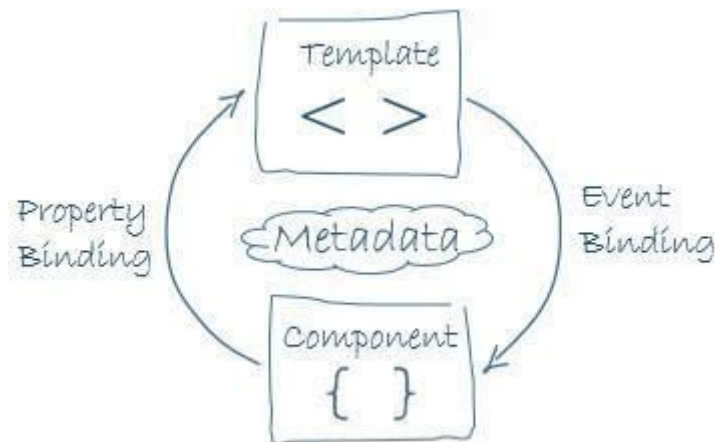
```
'templateUrl: 'alert.component.html
```

```
{
```

אנגולר משתמש בשיטת two way data binding. כלומר שכל שינוי הנעשה בקוד (ts) משפיע ישירות על הנראות שלו בטופס הנראה למשתמש (html) וכן להיפך.







דוגמא לחיבור של קומפוננטה ל template עליו היא אחראית:

```
import { AlertService } from '../_services';

@Component({
  selector: 'alert',
  templateUrl: 'alert.component.html'
})

export class AlertComponent implements OnInit, OnDestroy {
```

דוגמא להטמעת הנתונים ב template :

באמצעות המאפיין ngModel שנוסיף לכל שדה קלט בhtml יודע ה view איזה שדה בקומפוננטה יכיל את הערך שיוזן בשדה, בדוגמא הנ"ל הקומפוננטה יודעת לחבר בין השדה Name לבין מה שיוזן לתוך

```
<header class="masthead">
  <div class="container">
    <div class="intro-text">
      <div class="intro-lead-in">Welcome To Our Studio!</div>
      <div class="intro-heading text-uppercase">It's Nice To Meet You</div>
      <a class="btn btn-primary btn-xl text-uppercase js-scroll-trigger" href="#services">Tell Me More</a>
    </div>
  </div>
</header>
<section class="page-section" id="Users">
  <div class="container">
    <div class="row">
      <div class="col-lg-12 text-center">
        <h2 class="section-heading text-uppercase">Users</h2>
        <h3 class="section-subheading text-muted">Lorem ipsum dolor sit amet consectetur.</h3>
      </div>
    </div>
    <li *ngFor="let user of users">
      {{user.username}} {{user.firstName}} {{user.lastName}}
      - <a (click)="deleteUser(user.id)" class="text-danger">Delete</a>
    </li>
  </div>
</section>
```

שדה הקלט.



אנגולר מאפשר לנו להשתמש ב service - מעין אובייקט המכיל פונקציות. זאת על מנת לסדר את הקוד ולאפשר להרבה קומפוננטות לממש את אותן הפונקציות מבלי ליצור אותם. מנגנון זה ממומש ע"י הזרקת ה service ב constructor של כל קומפוננטה שצורכת את ה service ואז אפשר לגשת למחלקת ה server ולהשתמש בפונקציות שלו ע"י פניה אליו, שם המחלקה. שם הפונקציה() בקובץ ה typescript של הקומפוננטה.

דוגמא ל service:

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

import { User } from '../_models';
import { environment } from 'src/environments/environment';

@Injectable({ providedIn: 'root' })
export class UserService {
  constructor(private http: HttpClient) { }

  getAll() {
    return this.http.get<User[]>(`${environment.apiUrl}/users`);
  }

  getById(id: number) {
    return this.http.get(`${environment.apiUrl}/users/${id}`);
  }

  register(user: User) {
    return this.http.post(`${environment.apiUrl}/api/Account/Register`, user);
  }

  update(user: User) {
    return this.http.put(`${environment.apiUrl}/users/${user.id}`, user);
  }

  delete(id: number) {
    return this.http.delete(`${environment.apiUrl}/users/${id}`);
  }
}
```

הזרקת ה service ב constructor:

```
export class ContactComponent implements OnInit {
  constructor(private baseService: BaseService) { }
```

כאן ניתן לראות שהקומפוננטה הזריקה service בשם UploadImageService. הזרקת זו מאפשרת לו להשתמש בפונקציות הקיימות ב service הזה בצורה הבאה:



```

17  onSubmit() {
18      this.baseService.Contact("http://localhost:49261/api/Contact/SendContact",this.model).subscribe((res)=>{
19          alert('SUCCESS!! :-)\n\n' + JSON.stringify(this.model)+ res);
20      }),(error => {
21          alert('ERROR!! :-)\n\n' + JSON.stringify(this.model)+ error);
22      })))
23  }

```

דוגמא ל: get

```

getAll() {
    return this.http.get<User[]>(`${environment.apiUrl}/users`);
}

```

דוגמא ל: post

```

register(user: User) {
    return this.http.post(`${environment.apiUrl}/api/Account/Register`, user);
}

```

הנתיב המתקבל כפרמטר בתוך הפונקציה הוא אותו נתיב הנמצא בserver בקונטרולים של שכבת האפליקציה. כאשר מתקבל הנתיב, הפונקציה בסרבר לה הולכים הנתונים היא אותה פונקציה הנמצאת מתחת לנתיב זהה.  
 כעת נעבור לפרט את צד הserver-.

## 4.2.2 צד Server

כפי שנכתב, מה שקובע איזו פונקציה תתבצע בserver אשר אליה ישלחו הנתונים או ממנה ישלפו הנתונים זהו הנתיב.

לדוגמא: עבור הדוגמא שבעמוד הקודם, הפונקציה שתתבצע בסרבר תהיה:

```

// POST api/Account/Register
[AllowAnonymous]
[Route("Register")]
public async Task<IHttpActionResult> Register(Model.User model)
{
    UserRepo.Save(new Users { emailAddress = model.Email, userName = model.FirstName });

    return Ok();
}

```

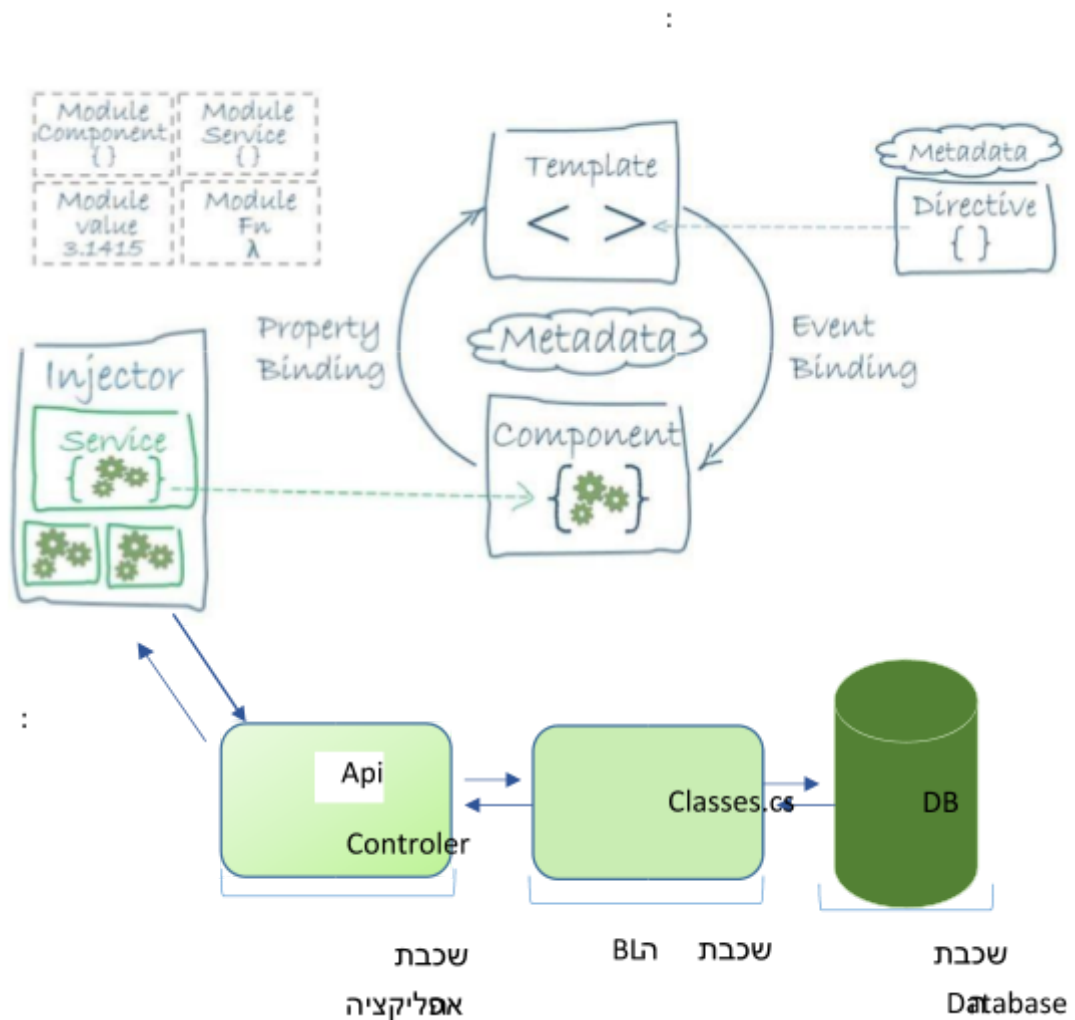
כי הנתיב שלה זהה לנתיב שנשלח לפונקציית ה post בקליינט.



הקונטרולרים של שכבת האפליקציה כוללים בד"כ רק פניות למחלקות של שכבת ה BL וזאת והם אלו שמתקשרים עם ה DB כפי שכבר ראינו ופרטנו לעייל, וזאת שוב על מנת לשמור על עקרונות המידול.

לסיכום : תהליך תעבורת הנתונים בפרויקט:

Client:



## פונקציונאליות

- ❖ שימוש ב-Authentication ע"מ לוודא שהמשתמש מוכר ורשום למערכת ורק אז תהיה לו גישה לנתונים.
- ❖ בדיקות תקינות לנתונים שמכניס המשתמש.
- ❖ שליפה, הוספה, מחיקה ועדכון נתונים בטבלאות.
- ❖ הורדת תמונות באיכות נמוכה למחשב האישי.
- ❖ ממשק משתמש מעוצב, יפה ונוח לשימוש.

6



## סיכום ומסקנות

העבודה על פיתוח אתר Photo הקנתה לנו ידע רב באנגולר 6, בתכנון נכון של מסד הנתונים, ופונקציות יעילות.

העבודה על פיתוח האתר היתה אינטנסיבית ומחייבת סטנדרטים גבוהים בשל העובדה שהאתר צריך לעלות לאויר ולפעול ככל אתר בצורה התקינה ביותר מה שחייב אותנו בשימת לב מרובה והשקעה רבה וחשיבה על כל פרט ותהליך.

השקענו בפונקציות נכונות ומסד נתונים טוב ותקין שיכיל את כל הנתונים בצורה היעילה ביותר כמו כן יצרנו ממשק משתמש נוח ביותר לשימוש ללקוח.

וממשק משתמש לניהול האתר לבעלת העסק שזה כולל העלאת תמונות לאתר ויצירת קטגוריות חדשות, הכנסת פריטים חדשים, ומבצעים, וכן רישום משתמשים באתר וכו'.

7



## ביבילוגרפיה

[www.stackoverflow.com](http://www.stackoverflow.com) ❖  
[www.bootstrap.com](http://www.bootstrap.com) ❖  
[www.angulario.com](http://www.angulario.com) ❖  
<https://reshetech.co.il> ❖  
<https://jasonwatmore.com> ❖  
<https://github.com> ❖  
<https://www.npmjs.com> ❖

