

ספר פרויקט

2Dots

שירה דגני (רייזמן)

ונועה רדליך



שלמי תודה

בראש ובראשונה תודה לבורא עולם שהביאנו עד הלום.

תודה לרב י. ליברמן שליט"א, מנהל הסמינר.

שליווה את צעדינו לאורך כל השנים, ודאג לנו למיטב הכלים והאפשרויות ללמוד, לגדול ולהצליח.

עמו במלאכה הגב' ה. גרוס, מרכזת המגמה אשר לא חסכה כל מאמץ להכשיר אותנו בצורה הטובה ביותר, על המסירות, ההדרכה וההשקעה הרבה.

למנחת הפרויקט : הגב' גולדמינץ, על חשיבה על כל פרט ופרט ועזרה מעשית,

על הסבלנות והייעוץ בכל זמן ובכל מקום מתוך רצון להביאו לידי גמר בהרגשה טובה ובמאור פנים.

וכמובן מעל לכל לבני משפחתנו היקרים

שלולא הבנתם, עצותיהם החכמות, ותמיכתם לאורך כל הדרך לא היינו מגיעות לאן שהגענו.

תוכן עניינים

3	מבוא
4	תקציר
5	הצעת פרויקט
16	תיאור הפרויקט
17	מפרט טכני
17	דרישות המערכת
17	דרישות פונקציונליות
18	יעדים ומטרות
18	יעדים
18	מטרות
19	מבנה נתונים
19	ארגון קבצים
20	אבטחת מידע
21	תרשימי מערכת מרכזיים
22	תיאור ופרוט המסכים
27	זרימת המידע במערכת
28	טכנולוגיה נבחרת
28	תכנון
28	עקרונות התכנות
29	תיאור האלגוריתם העיקרי ומחלקות בצד לקוח ושרת
53	בדיקות המערכת
54	בקרת גירסאות
54	מה תרם לנו הפרויקט
54	בביליוגרפיה
55	נספחים – דוגמאות לקטעי קוד עיקריים

מבוא:

כאשר חיפשנו רעיון לפרויקט, הקווים שהנחנו אותנו בבחירת הרעיון היו – מחד ליצור פרויקט מקצועי העושה שימוש בטכנולוגיות המתקדמות ביותר, ומאידך פרויקט מעשי ויעיל שניתן לעשות בו שימוש מיטבי.

רצינו פרויקט שלא יהיה רק בגדר מטלה שצריך לבצע אלא יתן ויוסיף ידע ופרקטיקה מעשית וכן לפתח יישום שיביא תועלת ועזרה בכנפיו.

לבסוף החלטנו לפתח אתר שמנהל שליחת הזמנות במייל.

לאחר בירור מקיף על החסרים הקיימים כיום בשוק העבודה, החלטנו על פיתוח בתחום אשר יקל רבות על בעלי השמחות.

על בעלי השמחות מוטלת כיום עבודה מרובה שכוללת: חיפוש כתובות, עיצוב הזמנה, שליחה להדפסה, הפקת ההזמנה והכנתה לשליחה, בדיקה ומעקב אחר הגעת ההזמנות ליעדן.

תהליך זה מוטל על כתפיהם של בעלי השמחה שגם כך עמוסים בהכנות הנדרשות לשמחה המתקרבת, התהליך ארוך ומייגע כאשר הוא נעשה באופן ידני, וכולל ניירת מרובה.

כמו"כ מטבע הדברים כאשר תהליך מורכב שכזה מתנהל באופן ידני הרי שקיימים מקרים רבים של אובדן מידע אי סדר וטעויות אנוש.

מטרת הפרוייקט הינה לנהל את כל התהליך הנ"ל תחת קורת גג אחת בצורה פשוטה ומונגשת ביותר אשר תתן מענה אוטומטי לכל פניה ותנהל בעצמה את כל התהליך כמעט ללא צורך בהתערבות בעלי השמחה.

אתר יחסוך זמן עבודה, כסף, ומשאבים יקרים.

ע"י המנהל יתבצע מעקב אחר פתיחת המיילים, ובתאריך נקוב לפני האירוע ישלח למשתמש פירוט מצב פיתחת המיילים אצל כל המוזמנים.

המערכת מנהלת את כל התהליך הנ"ל, שומרת נתונים, מבצעת בדיקות חוקיות.

תקציר:

במהלך הפרויקט הקמנו אתר יעודי לבעלי שמחה הכולל ניהול שליחת הזמנות במייל. האתר נכתב בסביבת visual studio. הטכנולוגיה הראשית בה השתמשנו היא אנגולר, שהיא הטכנולוגיה המתקדמת ביותר כיום, ואליה מצטרפות טכנולוגיות נוספות כמו web.api וכד'. כמו כן לאתר הוספנו ומימשנו פונקציות בסיסיות כמו כניסה מזוהה, הוספה מחיקה עידכון וכו', וכן התממשקנו לאתר מסר 10 שבאמצעותו המערכת שולחת את ההזמנות. מטרתו של הספר היא להציג בפניכם את הפרויקט מהיבטיו השונים. האתר **dots2** הוא אתר ידידותי המסייע לנהל שליחת הזמנות במייל בצורה חדשנית. ראשית המשתמש מכניס את אנשי הקשר אליו הוא רוצה לשלוח הזמנות והמערכת מבצעת התאמת כתובות מייל לכל איש קשר ושולחת את ההזמנה באמצעות אתר מסר 10.

תאריך: _____

הצעה לפרויקט גמר

יש להדפיס את כל הנתונים הנדרשים

פרטי הסטודנט

שם הסטודנט	ת.ז. 9 ספרות	כתובת	טלפון נייד	תאריך ס.הלימודים
נועה רדליך	314906090	עוזיאל 106	0527669394	2020
שירה רייזמן	206500522	זרח ברנט 4	0583246977	2020

שם המכללה מרכז בית יעקב סמל המכללה: 7220
מסלול ההכשרה: הנדסאים מוסמכים.

מגמת לימוד: מחשבים – הנדסת תוכנה מקום ביצוע הפרויקט: מרכז בית יעקב

פרטי המנחה האישי

שם המנחה	כתובת	טלפון נייד	תואר	מקום עבודה/תפקיד
גולדמן רחל	בית יעקב 17	0548468600	B.A.	מנכ"ל בית יעקב

עבור מנחה אישי חדש יש לצרף קורות חיים, ניסיון מקצועי ותעודות השכלה לאישור מה"ט.

שירה רייזמן
חתימת הסטודנט

ג. רייזמן
חתימת הסטודנט

חתימת הגורם המקצועי מטעם מה"ט

גולדמן רחל
חתימת המנחה האישי

1. שם הפרויקט:

2.DOTS

2. רקע:

2.1. תיאור ורקע כללי –

האתר מיועד לציבור החרדי, והוא בא ליעל לציבור את שליחת ההזמנות לאירועים. האתר מאגד בתוכו את ניהול את עיצוב ההזמנה ואופן שליחתה במייל. הלקוח נדרש רק לבחור את ההזמנה הרצויה ולהטעין את קובץ הכתובות, מכאן והלאה הכל נעשה באופן אוטומטי.

2.2. מטרות המערכת –

- להוזיל את עלות שליחת ההזמנות.
- להקל במציאת כתובות המיילים.
- לאפשר בחירת דוגמא של הזמנה מתוך מגון דוגמאות.

3. סקירת מצב קיים בשוק, אילו בעיות קימות:

ישנם אתרים ליצירת הזמנות אך ללא שליחתם. כיום אין מערכת שמנהלת גם את יצירת ההזמנה, גם את ניהול הכתובות וגם את שליחת המיילים.

4. מה הפרויקט אמור לחדש או לשפר:

הפרויקט יציע יצירת הזמנה אינטראקטיבית ניהול כתובות ע"י מציאת כתובת מייל לכל אדם שליחת ההזמנות ומעקב אחר קבלתן.

5. דרישות מערכת ופונקציונליות:

- בחירת דוגמת הזמנה מתוך הדוגמאות הניתנות.
- טעינת קובץ כתובות והתאמת הפרטים לכתובת המייל.

6. בעיות צפויות במהלך הפיתוח ופתרונות:

6.1. תיאור הבעיות –

- * מציאת שמות שדות זהים בחיפוש הכתובות.
- * שליחת המיילים באופן מרכז.

6.2. פתרונות אפשריים –

- * התאמת השדות ע"י המשתמש.
- * שימוש באתר נוסף שמציע את השרות.

7. פתרון טכנולוגי נבחר:

7.1 טופולוגית הפתרון – פריסת המערכת:

המערכת מורכבת מ:
-אתר אינטרנט
-SQL server (DB)
-ממשק טעינת קבצים
-ממשק שליחת מיילים

7.2 טכנולוגיות בשימוש -

SPA- single page application

7.3 שפות פיתוח –

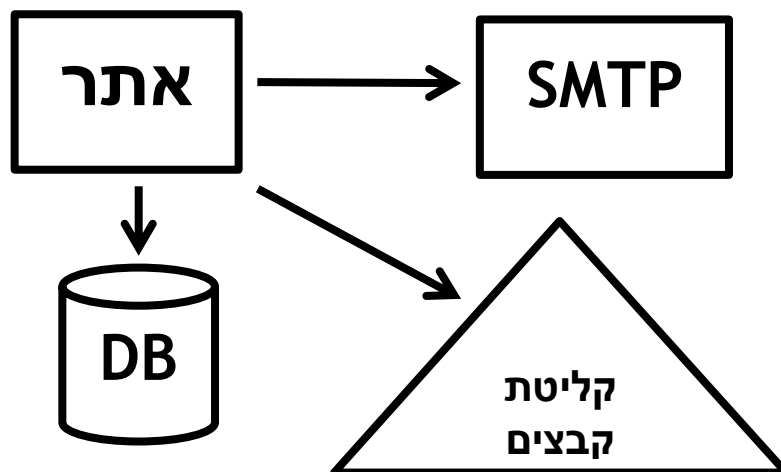
שפות אלו נבחרו משום שהן מציעות מגוון כלים בסביבת הפיתוח, וביחד עם בסיס הנתונים שנבחר – מאפשרות לנו להשיג את מטרות הפרויקט.

C#-

Html-

Angular-

7.4 הארכיטקטורה הנבחרת –



7.5 חלוקה לתוכניות ומודלים –

UI*

API*

BL*

DL*

MODEL*

פונקציות: לכל טבלה תהינה הפונקציות select insert update לפי הצורך.

פונקציות נוספות: MatchContact- התאמת מייל לכל נמען.

SendMail- שליחת מייל לכל הנמענים.

LoadExcelFile- טעינת קובץ excel ושפיכתו לתוך הDB.

ExsportSqlToCsv- יצוא רשומות מהsql לקובץ csv

7.6. סביבת השרת – מקומי.

7.7. ממשק המשתמש/לקוח – GUI

*login:

Web page title

http://2dots.url.com

מילוי פרטים פרטי הזמנה פרטי הכתובות תשלום

משתמש חדש מנוי

שם פרטי שם משפחה טלפון מייל סיסמא

שמירה והמשך

Web page title

http://2dots.url.com

מילוי פרטים פרטי הזמנה פרטי הכתובות תשלום

משתמש חדש מנוי

מייל סיסמא

שמירה והמשך

* מילוי פרטי ההזמנה:

Web page title

http://2dots.url.com

מילוי פרטים פרטי הזמנה פרטי הכתובות תשלום

הזן את פרטי ההזמנה

פרטי ההזמנה...

בחר דוגמא של הזמנה מתוך המאגר

שמירה והמשך

* טעינת קובץ כתובות:

Web page title

http://2dots.url.com

מילוי פרטים פרטי הזמנה פרטי הכתובות תשלום

טען את קובץ הכתובות

טעינה

מחזיר שאלה למשתמש על אנשים שההתאמה לא ברורה

שמירה והמשך

* תשלום:

Web page title

http://2dots.url.com

מילוי פרטים פרטי הזמנה פרטי הכתובות תשלום

הכנס פרטי כרטיס אשראי:

פרטי האשראי...

פרטי האשראי...

פרטי האשראי...

אישור התשלום
וסיום

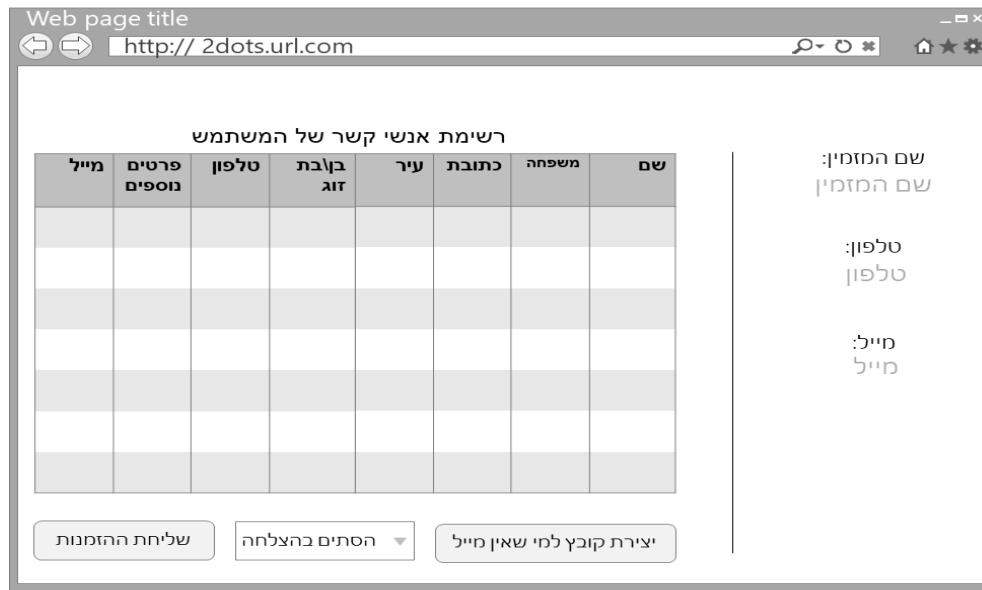
* מסך מנהל:

Web page title

http://2dots.url.com

מספר הזמנה	שם הלקוח	פרטי ההזמנה
1	A	מסך פרטי הזמנה
2	B	מסך פרטי הזמנה
3	C	מסך פרטי הזמנה
....

* מסך פרטי הזמנה:



Web page title

http:// 2dots.url.com

רשימת אנשי קשר של המשתמש

שם	משפחה	כתובת	עיר	בן/בת זוג	טלפון	פרטים נוספים	מייל

שם המזמין:
שם המזמין

טלפון:
טלפון

מייל:
מייל

יצירת קובץ למי שאין מייל

הסתים בהצלחה

שליחת ההזמנות

7.8 שימוש בחבילות תוכנה –

.Kendo, Bootstrap,meser10

8. שימוש במבני נתונים וארגון קבצים: 8.1. צילומי טבלאות data base –

Order

Column Name	Data Type	Allow Nulls
orderId	int	<input type="checkbox"/>
userId	int	<input type="checkbox"/>
invitationId	int	<input type="checkbox"/>
invitationDetails	nvarchar(MAX)	<input type="checkbox"/>
status	nvarchar(50)	<input type="checkbox"/>

Contact

Column Name	Data Type	Allow Nulls
contactId	int	<input type="checkbox"/>
contactFirstName	nvarchar(50)	<input type="checkbox"/>
contactLastName	nvarchar(50)	<input type="checkbox"/>
contactSpouse	nvarchar(50)	<input checked="" type="checkbox"/>
contactAddress	nvarchar(50)	<input type="checkbox"/>
contactCity	nvarchar(50)	<input type="checkbox"/>
contactTelephone	nvarchar(10)	<input type="checkbox"/>
contactDetails	nvarchar(50)	<input checked="" type="checkbox"/>
contactMail	nvarchar(50)	<input type="checkbox"/>

Users_contact

Column Name	Data Type	Allow Nulls
userId	int	<input type="checkbox"/>
contactId	int	<input type="checkbox"/>
contactFirstName	nvarchar(50)	<input type="checkbox"/>
contactLastName	nvarchar(50)	<input type="checkbox"/>
contactSpouse	nvarchar(50)	<input checked="" type="checkbox"/>
contactAddress	nvarchar(50)	<input type="checkbox"/>
contactCity	nvarchar(50)	<input type="checkbox"/>
contactTelephone	nvarchar(10)	<input type="checkbox"/>
contactDetails	nvarchar(50)	<input checked="" type="checkbox"/>
contactMail	nvarchar(50)	<input type="checkbox"/>
contactTitle	nvarchar(50)	<input checked="" type="checkbox"/>

Users

Column Name	Data Type	Allow Nulls
userId	int	<input type="checkbox"/>
userFirstName	nvarchar(50)	<input type="checkbox"/>
userLastName	nvarchar(50)	<input type="checkbox"/>
userTelephone	nvarchar(50)	<input type="checkbox"/>
userMail	nvarchar(50)	<input type="checkbox"/>
userPassword	nvarchar(10)	<input type="checkbox"/>

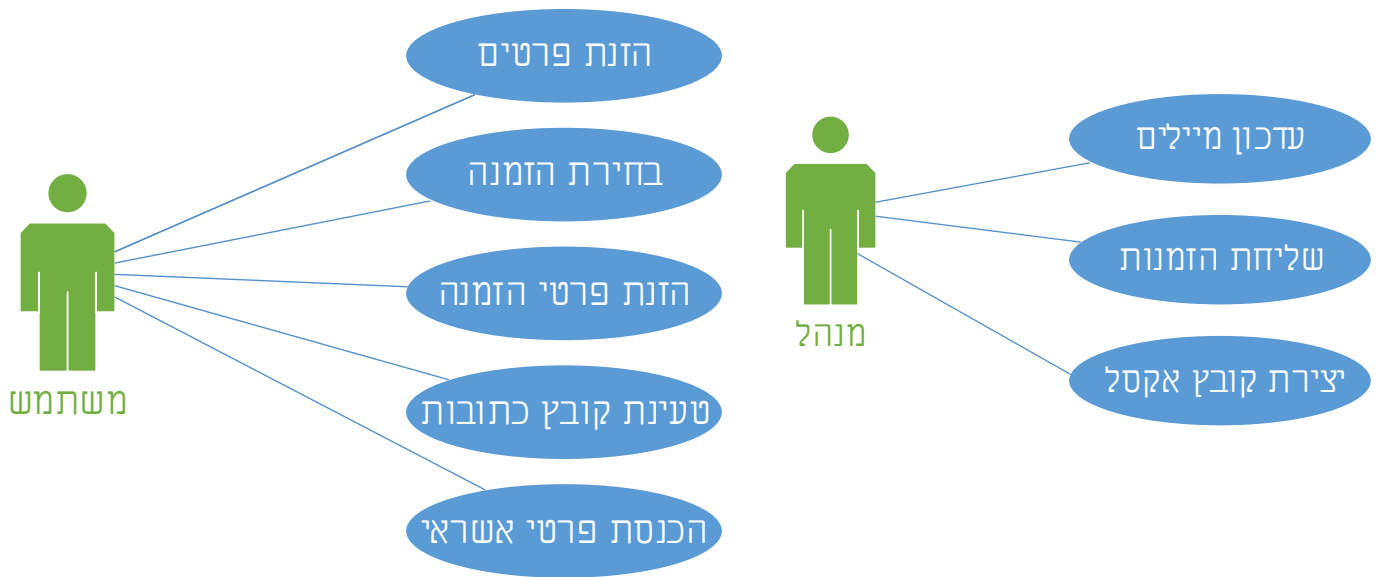
8.2. שיטת האחסון –

בסיס נתונים של SQL.

8.3. מנגנוני התאוששות מפילה/קריסה/תמיכה בטרנזקציות –

מנגנוני התאוששות מובנים של SQL.

9. תרשימי מערכת מרכזיים:



10. תיאור/התייחסות לנושאי אבטחת מידע:

- הזדהות.
- הרשאות לפי רמת משתמש.

11. משאבים הנדרשים לפרויקט:

11.1. מספר שעות המוקדש לפרויקט, חלוקת עבודה בין חברי הצוות –

700 שעות.

11.2. חלוקת עבודה בין חברי הצוות -

700 העבודה תחולק בין המגישות בצורה מאוזנת

11.3. ציוד נדרש –

מחשב ורשת

11.4. תוכנות נדרשות –

.Visual studio (code)

.Sql server

11.5. ידע חדש שנדרש ללמוד לצורך ביצוע הפרויקט –

* ספריות ליצירת הזמנה.

11.6. ספרות ומקורות מידע –

אינטרנט.

12. תכנית עבודה ושליבים למימוש הפרויקט:

- * תכנון מבנה הפרויקט.
- * בניית DB.
- * כתיבת אלגוריתם.
- * בניית מסכים.
- * בדיקות תקינות.

13. תכנון הבדיקות שיבוצעו:

- הכנסת שם ואיתורו במאגר כתובות.
- הוספת משתמש ובדיקת הימצאותו ב-DB.
- שליחת מייל ובדיקה האם נשלח כראוי.
- בדיקה האם התאמת המיילים פועלת כראוי.

14. בקרת גרסאות (version control):

גרסאות קודמות ינוהלו ב – TFS.

ש.ר. נ.ר.

חתימת המנחה האישי
חתימת המנחה האישי

חתימת הסטודנט
חתימת הסטודנט

ג. הערות ראש המגמה במכללה

ד. אישור ראש המגמה

שם: ד"ר גל חתימה: ד"ר גל תאריך: 24.6.19

ה. הערות הגורם המקצועי מטעם מה"ט

1/2/19

ו. אישור הגורם המקצועי מטעם מה"ט

שם: ד"ר גל חתימה: ד"ר גל תאריך: 29/7/19

תיאור הפרויקט:

הפרויקט נועד לאפשר למשתמש לנהל את שליחת ההזמנות בצורה ממוחשבת, עי ממשק webi נעים למראה ונח לתפעול.

המערכת נותנת ניהול מרוכז של כל הפעולות הנדרשות לתהליך שליחת הזמנות לשמחה.

בציבור החרדי ענין שליחת ההזמנות הפך למעמסה כבדה ביותר על בעלי השמחה. על בעל השמחה נדרשת עבודה רבה בחיפוש אחר כתובות המוזמנים, התרוצצות סביב עיצוב ההזמנה הדפסתה והכנתה לשליחה. וכן נגרם עוגמת נפש מרובה בשל אי הגעה של הזמנות ליעדן במועד הרצוי.

מטרת הפרויקט שיצרנו להקל על בעלי השמחה ולגרום להם להגיע לשמחה ברוגע ושלווה בנוגע לנושא ההזמנות. המערכת עובדת באופן יעיל ביותר וידידותי למשתמש, והכל נעשה במינימום זמן, במינימום התרוצצויות ובמעקב צמוד אחר הגעת ההזמנות ליעדן.

בניית הפרויקט נעשתה במס' שלבים:

ייזום רעיון הפרויקט

תכנון וארגון שלבי פיתוח התוכנה על פי הדרישות

לימוד החומר לצורך פיתוח הפרויקט

תכנון אפשרויות מתקדמות

כתיבת הקוד

ניהול מסד נתונים

עיצוב Design של המערכת

ניסוי התוכנה ובדיקות

עיקר הביצוע של הפרויקט התמקד בשני תחומים - החלק של ניהול מסד הנתונים, וכתיבת הקוד.

את מסד הנתונים יצרנו ב-SQL -

הוא זה שיכיל את כל הנתונים אותם נרצה להציג למשתמש באתר.

הביצוע של שמירת הנתונים נעשה ע"י פרוצדורות ששומרות בטבלאות לפי נושאים, נתוני כל שאילתא ופעילות בשרת של המערכת עליה מתבסס האתר.

מפרט טכני:

סביבת פיתוח

עמדת פיתוח- מחשב PC

מערכת הפעלה windows10

שפות תכנות- web API, angular6

עמדת משתמש מינימאלית

חומרה- מחשב

חיבור לאינטרנט- נדרש

למשתמש בשרת

כלי התוכנה לפיתוח התוכנה-

Microsoft Visual Studio Code

דרישות המערכת:

דרישות משתמש- משתמש נכנס לאתר, מבצע הזנת שם משתמש וסיסמא, על המשתמש לבחור את סוג ההזמנה מבין הדוגמאות הניתנות וכן לכתוב את נוסח ההזמנה הרצוי לו ולאחר ביצוע ההזדהות עליו להעלות את קובץ אנשי הקשר שלו בהתאם לתבנית המערכת. לאחר מכן עליו לאשר את התאמת כתובות המייל לכל איש קשר שהתאמתו לא מוחלטת, ואישור שליחת ההזמנות.

דרישות המנהל- מעקב אחר הזמנות המשתמשים, שליחת ההזמנות דרך אר מסר10, ושליחת קובץ אקסל הכולל את כל אנשי הקשר שלא נמצאה התאמה למייל ללקוח.

דרישות פונקציונאליות:

על המשתמש לבחור דוגמת הזמנה בין הדוגמאות הניתנות.

טעינת קובץ כתובות והתאמת הפרטים לכתובת המייל.

יעדים ומטרות:

יעדים:

- האתר יהיה מעוצב וידידותי למשתמש, המנחה בצורה ברורה את אופן השימוש בהליך שליחת ההזמנה.
- ניהול מרוכז של כל הפעולות הקשורות לתהליך שליחת ההזמנה, מבחירת בעיצוב עד לשליחה.
- מעקב אחר שליחת ההזמנות ופתיחתן ע"י המוזמן.
- סינכרון בין איש קשר לכתובת המייל הנמצאת במאגר.

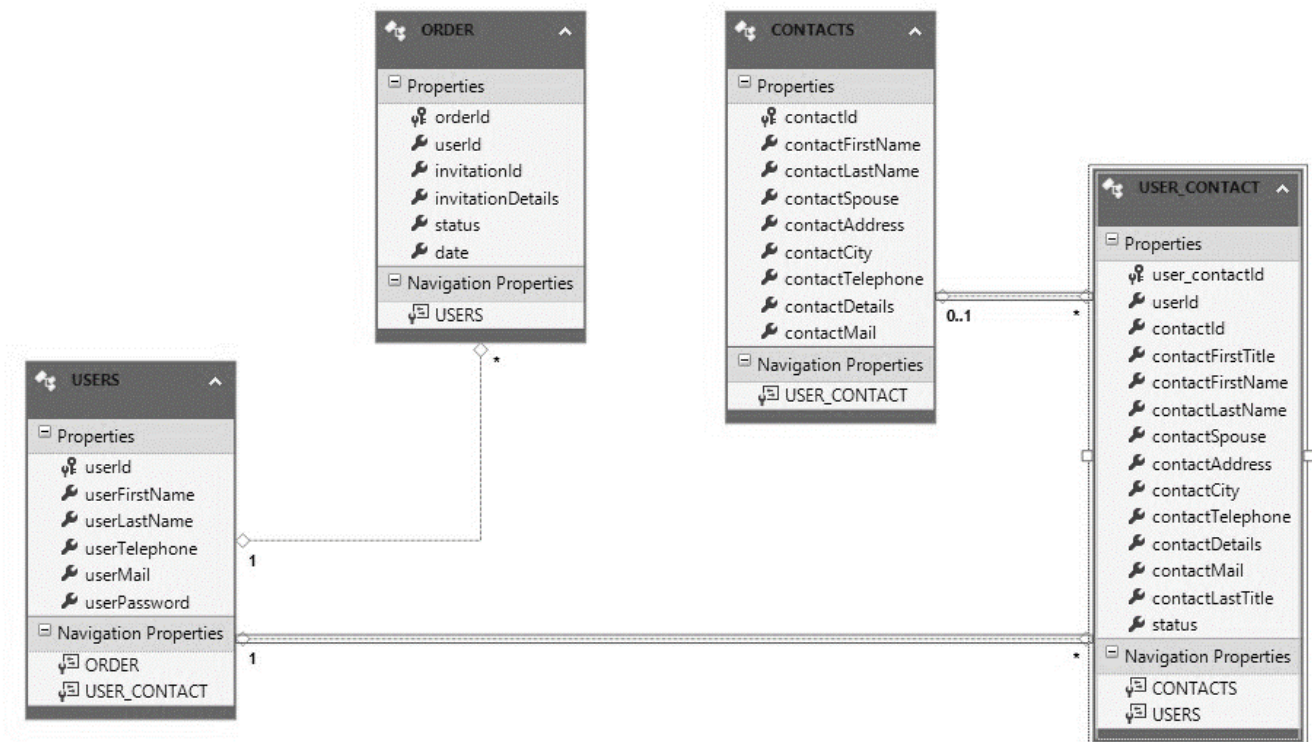
מטרות:

- להוזיל את עלות שליחת ההזמנות.
- להקל במציאת כתובות המיילים.
- לזרז משמעותית את אופן שליחת ההזמנות.
- לחסוך בניירת להשליט סדר.
- בניית ממשק פשוט וברור לביצוע תהליך שליחת ההזמנות.

מבנה נתונים:

המערכת מתבססת על שרת, השואב נתונים מבסיס נתונים של sql server . ניתן לגלוש לאתר מכמה דפדפנים בו זמנית.

בצד שרת מתבצעים העיבוד והשליפות של הנתונים, ובצד לקוח מתבצעת התצוגה.



ארגון קבצים:

צד שרת-

המערכת בנויה במודל השכבות: שכבת תצוגה (presentation layer),

שכבת הלוגיקה העסקית (BL-Buisness Logic),

שכבת נתונים (DAL-Data Access Layer).

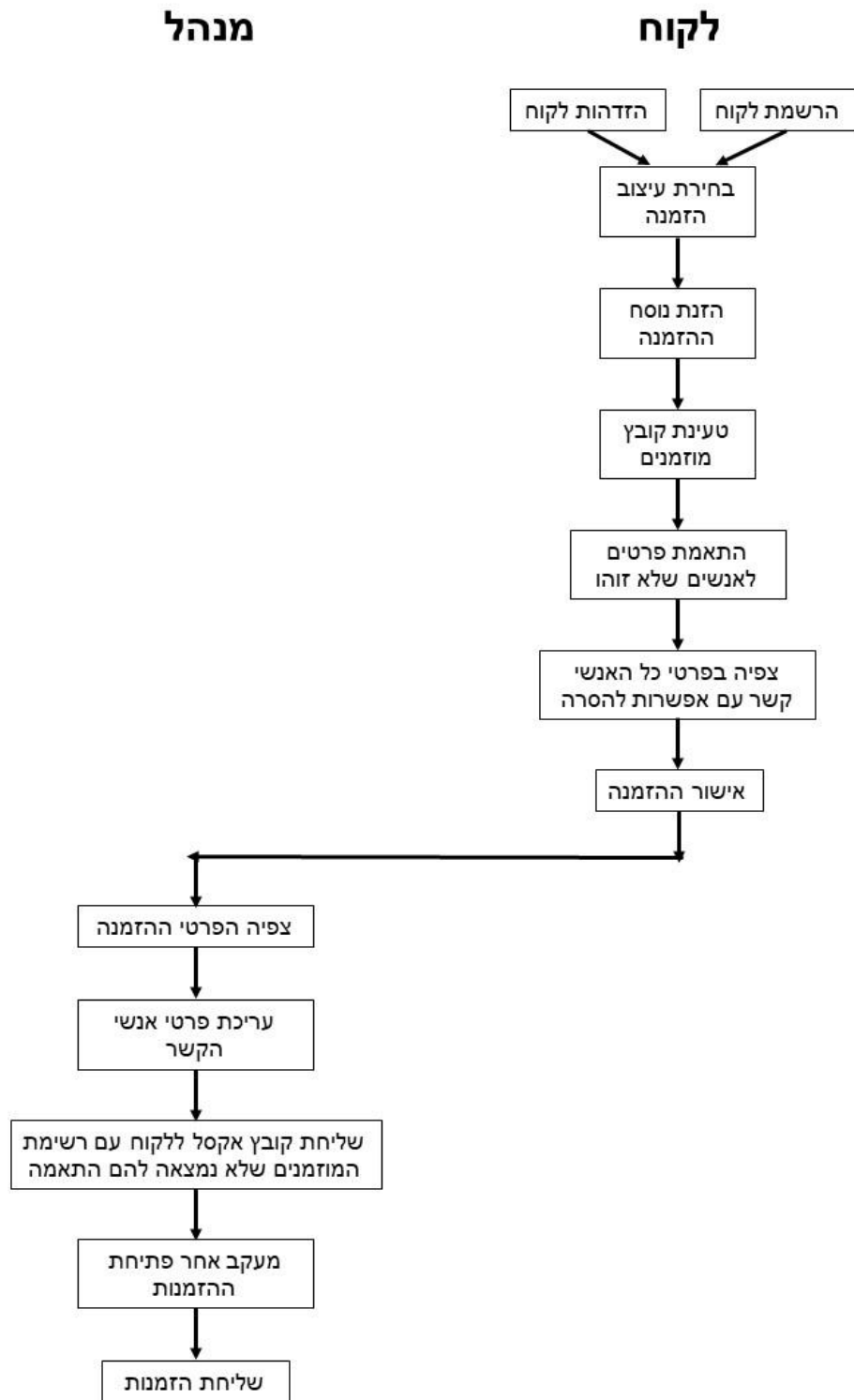
צד לקוח –

את צד הלקוח – הכתוב באנגלית 6 יצרנו בפרויקט נפרד משום המודולריות שבהפרדה ונוחות בסביבת עבודה שונה מצד השרת. צד שרת – VisualStudio וצד לקוח – VSCode.

אבטחת מידע:

בכדי למנוע טעויות אנוש בקלט מהמשתמש, ביצענו בדיקות תקינות הן בצד לקוח והן בצד שרת, על כל קלט חיוני. בדיקות התקינות יעילות גם לבעיות נוספות של אבטחת מידע כמו דריסת זיכרון, כלומר הקלט בגודל המתאים לו שהוא הגודל המוקצב לו בזיכרון, בדיקות התקינות מונעות גם אפשרות של גניבת מידע אישי על לקוחות וגניבת מידע מהמערכת.

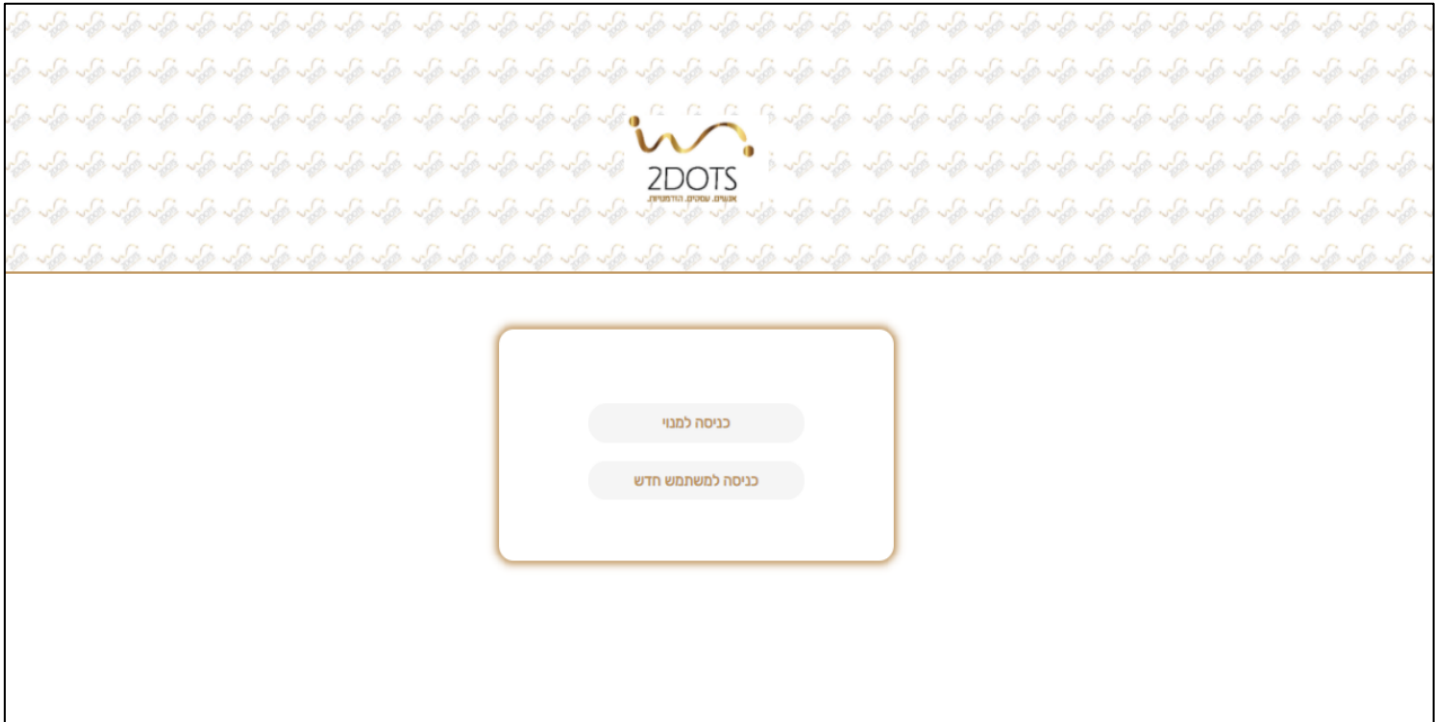
תרשימי מערכת מרכזיים:



תיאור ופירוט המסכים:

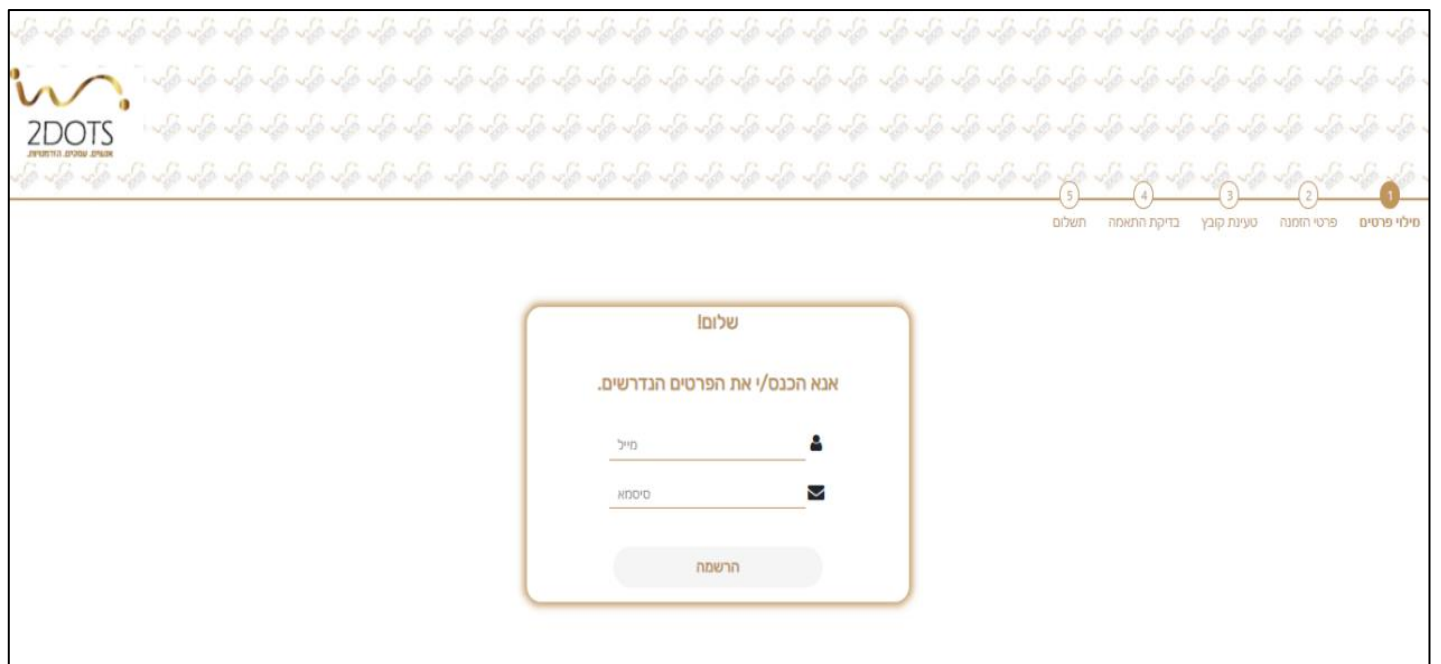
כניסה למערכת:

כאשר המשתמש מכניס שם משתמש וסיסמא, בלחיצה על "כניסה" מתבצע אימות מול DB.



The login screen features a repeating pattern of small logos in the background. At the top center is the 2Dots logo with the text "אנשים נעים הנדרשים" below it. In the center, there is a white rounded rectangle containing two buttons: "כניסה למנוי" (Login) and "כניסה למשתמש חדש" (Login as new user).

אם המשתמש קים הוא נכנס למנוי.



The registration screen features a repeating pattern of small logos in the background. At the top left is the 2Dots logo with the text "אנשים נעים הנדרשים" below it. At the top right is a progress bar with five steps: 1. מילוי פרטים (filled), 2. פרטי חשונה, 3. טעינת קובץ, 4. בדיקת התאמה, and 5. תשלום. In the center, there is a white rounded rectangle with the heading "שלום!" and the text "אנא הכנס/י את הפרטים הנדרשים." Below this are two input fields: "מיל" (with a person icon) and "סיסמא" (with an envelope icon). At the bottom of the rectangle is a button labeled "הרשמה".

אם המשתמש אינו קיים הוא נכנס למשתמש חדש.

2DOTS
אנשים עושים הדמיות

5 תשלום 4 בדיקת התאמה 3 טעינת קובץ 2 פרטי הזמנה 1 פרטי פרטים

שם פרטי
שם משפחה
טלפון
מייל
סיסמא

הרשמה

פרטי ההזמנה:

על המשתמש להכניס את תוכן ההזמנה וכן לבחור סוג הזמנה.

2DOTS
אנשים עושים הדמיות

5 תשלום 4 בדיקת התאמה 3 טעינת קובץ 2 פרטי הזמנה 1 פרטי פרטים


הזן פרטי הזמנה

הכנס טקסט...

בחר את ההזמנה הרצויה

שמירה והמשך

העלאת קובץ אקסל של אנשי קשר:



5

4

3

2

1

תשלום

בדיקת התאמה

טעינת קובץ

פרטי החמנה

מילוי פרטים

סעו את קובץ הכתובות

לא נבחר קובץ

הצג את אנשי הקשר שלי

התאמת אנשי קשר:



5

4

3

2

1

תשלום

בדיקת התאמה

טעינת קובץ

פרטי החמנה

מילוי פרטים

נועה דליך עו"אל 1066 ירושלים
האם התכונת לי ☐

נועה דליך עו"אל 106 ירושלים
האם התכונת לי ☐

נועה דליך עו"אל 1066 ירושלים
האם התכונת לי ☐

נועה דליך עו"אל 1066 ירושלים
האם התכונת לי ☐

נועה דליך עו"אל 106 ירושלים
האם התכונת לי ☐

נועה דליך סמינר עו"אל 106
האם התכונת לי ☐

נועה דליך עו"אל 106 ירושלים
האם התכונת לי ☐

שמירה

הצגת אנשי קשר:



אנשי הקשר שלי

חוקר/ת	תואר ראשון	שם פרטי	שם משפחה	בן זוג	תואר שני	תחנות	עיר	סלפון	פרטים	מייל
<input type="checkbox"/>	gbg	נסה	רדליח	to	106	jerusalem	ירושלים	noa9394@gmail.com		
<input type="checkbox"/>	vbvg	שרה	רזמן	התכנדה	4 זרע	ירושלים	ירושלים	shulir8229@gmail.com		
<input type="checkbox"/>	gbg	נסה	רדליח	to	106	jerusalem	ירושלים	noa9394@gmail.com		
<input type="checkbox"/>	gbg	נסה	רדליח	to	106	jerusalem	ירושלים	noa9394@gmail.com		
<input type="checkbox"/>	gbg	נסה	רדליח	to	106	jerusalem	ירושלים	noa9394@gmail.com		
<input checked="" type="checkbox"/>	לאה	לאה	בפורת	לם		ירושלים	ירושלים	eah benporat@gmail.com		
<input type="checkbox"/>	לאה	נסה	רדליח	to	106	jerusalem	ירושלים	noa9394@gmail.com		
<input checked="" type="checkbox"/>	לאה	לאה	בפורת	לם		ירושלים	ירושלים	eah benporat@gmail.com		
<input type="checkbox"/>	gbg	נסה	רדליח	to	106	jerusalem	ירושלים	noa9394@gmail.com		
<input checked="" type="checkbox"/>	לאה	לאה	בפורת	לם		ירושלים	ירושלים	eah benporat@gmail.com		
<input type="checkbox"/>	gbg	נסה	רדליח	to	106	jerusalem	ירושלים	noa9394@gmail.com		

שמירה והמשך

חזור

מנהל:

רשימת הזמנות:



טבלת לקוחות פרטי המזמין

תחנה ראשונה	פרטים	שם משפחה	בן	תאריך	סוג	כתובת	עיר	טלפון	פרטים	מייל
gogco	noa	redlich	to	usiel	106	jerusalem	ירושלים			noa9394@gmail.com
bgvby	שרה	ירמון	התנבדה	4	רח	ירושלים	ירושלים			shulir8229@gmail.com
לכבוד	נעה	דניאל	החמודה	עוזיאל	106	ירושלים	ירושלים			
gogco	noa	redlich	to	usiel	106	jerusalem	ירושלים			noa9394@gmail.com
gogco	noa	redlich	to	usiel	106	jerusalem	ירושלים			noa9394@gmail.com
gogco	noa	redlich	to	usiel	106	jerusalem	ירושלים			noa9394@gmail.com
לאה	לאה	בפורת	לם	ירושלים		ירושלים	ירושלים			ah_benporat@gmail.com
gogco	noa	redlich	to	usiel	106	jerusalem	ירושלים			noa9394@gmail.com
לאה	לאה	בפורת	לם	ירושלים		ירושלים	ירושלים			ah_benporat@gmail.com
gogco	noa	redlich	to	usiel	106	jerusalem	ירושלים			noa9394@gmail.com
לאה	לאה	בפורת	לם	ירושלים		ירושלים	ירושלים			ah_benporat@gmail.com
gogco	noa	redlich	to	usiel	106	jerusalem	ירושלים			noa9394@gmail.com

סייל

סלפון

שליחת הזמנות

יצירת קובץ אקסל לאנשי קשר ללא מייל

שמירת השינויים בטבלה

צפיה בפרטי הזמנה ושליחתה:



אנשים נעים ונחמדים

1 סבלת לקוחות

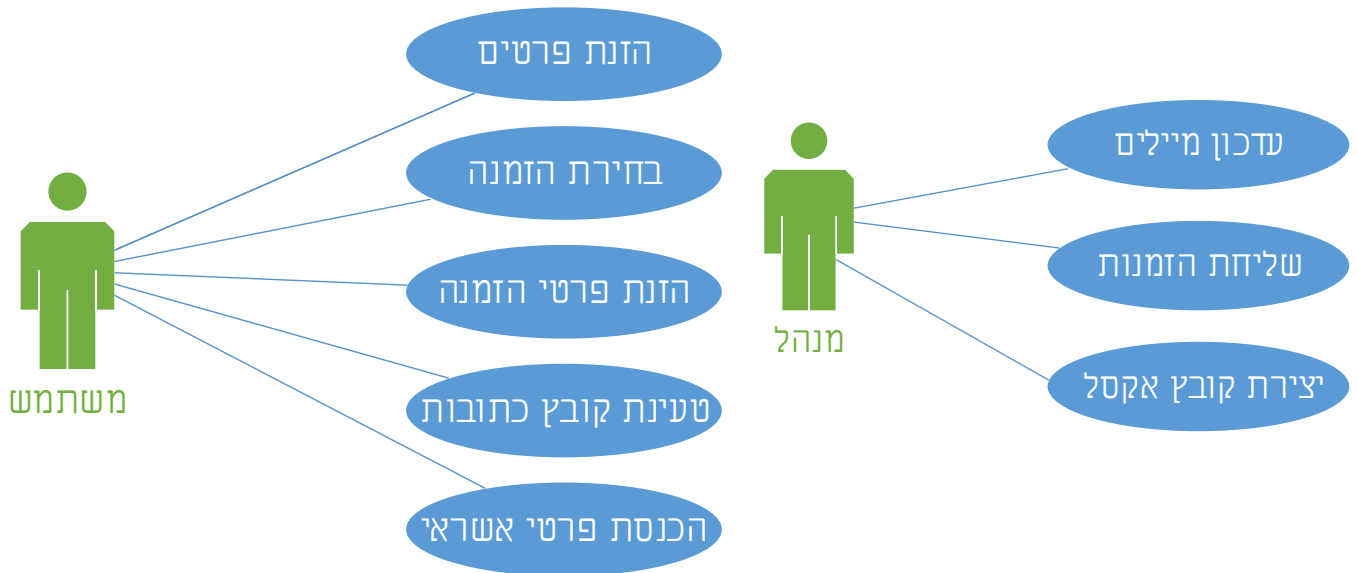
2 פרטי הזמנה

הוסף איש קשר חדש

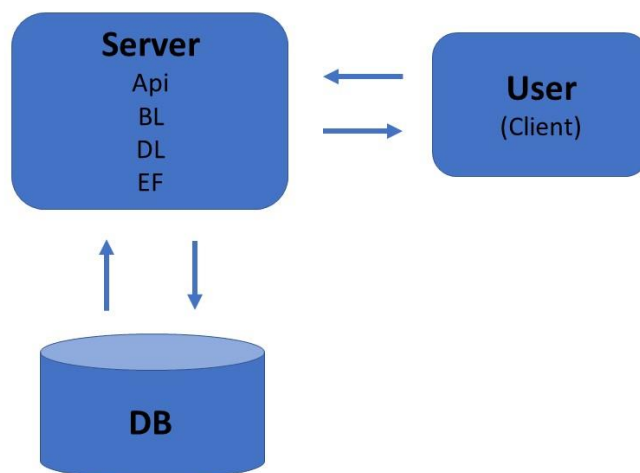
שם הלקוח	פרטי ההזמנה
סב סב	פרטי הזמנה
סב סב	פרטי הזמנה
סב סב	פרטי הזמנה
סב סב	פרטי הזמנה
סב סב	פרטי הזמנה
סב סב	פרטי הזמנה
סב סב	פרטי הזמנה
סב סב	פרטי הזמנה
סב סב	פרטי הזמנה
סב סב	פרטי הזמנה
סב סב	פרטי הזמנה
סב סב	פרטי הזמנה

זרימת המידע במערכת:

• Use case :



• Data flow :



טכנולוגיה נבחרת:

Client side – מומש ב – angular 6 , Type script

Server side – מומש ב – WebAPI

תכנון:

מטרתנו בכתיבת הפרויקט היתה לספק מערכת נוחה ונעימה למשתמש, התואמת את עקרונות אתרי אינטרנט חדישים כיום, ומאפשרת לנהל את המידע בקלות מקסימלית. דאגנו לעיצוב נעים לעין וחדיש, לזרימת מידע קלה וברורה ולהפשטה ככל האפשר של התהליכים ואופן השימוש. תכנון הפרויקט התבצע עפ"י בקשות הלקוח.

עקרונות התכנות:

❖ כללי

צד הלקוח בפרויקט נכתב בשפת typescript ו- html תוך כדי שימוש בטכנולוגיות מתקדמות:

✓ Angular 6

✓ 3 Css – לעיצוב הפרויקט.

וצד שרת בפרויקט נכתב ב C#, ASP.NET מסוג WEB.API.

❖ חלוקה לשכבות

חלוקה ברורה בין ממשק המשתמש כפי שנכתב לעיל. הפרויקט נכתב תוך התייחסות רבה לשימוש בטכנולוגיות המתקדמות ביותר. הקוד קצר ומתועד, מחולק למחלקות נכונות וללא כפילויות.

המטרה העיקרית שעמדה בפנינו במהלך הפיתוח היתה ליצור מוצר שיענה באופן מלא על דרישות הלקוח, משימה שהתגלתה מסובכת לעיתים עקב חוסר הרגילות בהגדרת מוצר ע"י הלקוח.

בנוסף היה עלינו להעביר תהליך ידני לחלוטין לתהליך ממוחשב, ללא פריבילגיה בהעזרות בחלקים ממוחשבים קיימים.

בנוסף הקפדנו במהלך הפיתוח ליצור מוצר קל לשימוש וכמה שיותר מקביל למהלך העבודה הנוכחי היום בעסק (ללא כל מערכת ממוחשבת). לשם כך נעזרנו במאגרים שונים שסופקו לנו ע"י הלקוח, וכמובן בקשר ישיר עם הלקוח שעזר לנו בהבהרות שונות על התהליך האמור להתבצע.

תיאור האלגוריתם העיקרי ומחלקות בצד לקוח ושרת:

❖ אלגוריתם:

המשתמש טוען קבוצת אנשי קשר:

```
uploadFile(event) {  
  debugger;  
  this.myFile = event.target.files[0];  
  this.uploadForm.get('profile').setValue(this.myFile);  
  debugger;  
  let frmData = new FormData();  
  frmData.append("file", this.uploadForm.get('profile').value);  
  this.orderService.upload(frmData, this.userId).subscribe(succses => {  
    debugger;  
    let navigationExtras: NavigationExtras = {  
      queryParams: {  
        "id": this.userId,  
        "match": succses  
      }  
    };  
    if (succses != null) {  
      this.router.navigate(["/match"], navigationExtras).then((e) => {  
        if (e) {  
          console.log("Navigation is successful!");  
        } else {  
          console.log("Navigation has failed!");  
        }  
      });  
    }  
  });  
} });
```

המערכת מתאימה בין איש הקשר לכתובת:

```
1 reference | 0 changes | 0 authors, 0 changes
public static List<DictDto> MatchContacts(int userId)//בדיקת התאמה
{
    List<DictDto> list = new List<DictDto>();
    List<User_ContactDto> user_Contacts = User_ContactManager_BL.GetUser_ContactsByIdWithStatus1(userId);
    foreach (User_ContactDto user_contact in user_Contacts)
    {
        list.Add(MatchContactsHelp(user_contact, userId));
    }
    list.RemoveAll(item => item == null);
    return list;
}
```

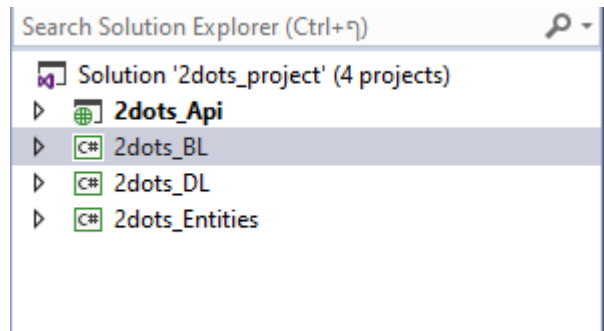
```
1 reference | 0 changes | 0 authors, 0 changes
public static DictDto MatchContactsHelp(User_ContactDto user_contact, int userId)
{
    ContactDL d = new ContactDL();
    User_ContactDL du = new User_ContactDL();
    List<ContactDto> contacts = GetAllContacts();
    List<User_ContactDto> updateUser_Contacts = new List<User_ContactDto>();
    List<ContactDto> returnContacts = new List<ContactDto>();
    List<ContactDto> helpContacts = new List<ContactDto>();
    User_ContactDto returnUser_Contacts = new User_ContactDto();
    User_ContactDto helpUser_Contacts = new User_ContactDto();
    int numOfMatch;
    bool flag = false;
    numOfMatch = 0;
    flag = false;
    foreach (ContactDto contact in contacts)
    {
        numOfMatch = 0;
        if (user_contact.contactTelephone != null && contact.contactTelephone != null && user_contact.contactTelephone != "")//אם לשתיים יש טלפון
        {
            if (user_contact.contactTelephone == contact.contactTelephone)//אם הטלפון שווה
            {
                user_contact.contactMail = contact.contactMail;//עדכון
                user_contact.contactId = contact.contactId;//עדכון
                updateUser_Contacts.Add(user_contact);
                break;
            }
            if (user_contact.contactFirstName == contact.contactFirstName)
                numOfMatch++;
            if (user_contact.contactLastName == contact.contactLastName)
                numOfMatch++;
            if (user_contact.contactCity == contact.contactCity)
                numOfMatch++;
            if (user_contact.contactAddress == contact.contactAddress)
                numOfMatch++;
            if (numOfMatch == 4)//אם כל הפרטים תואמים
            {
                user_contact.contactMail = contact.contactMail;//עדכון
                user_contact.contactId = contact.contactId;//עדכון
                updateUser_Contacts.Add(user_contact);
                break;
            }
            else if (numOfMatch >= 2)
            {
                helpUser_Contacts = user_contact;
                helpContacts.Add(contact);
                flag = true;
            }
        }
    }
    if (flag == true)//אם לא נמצא שום התאמה להכניס לרשימה שמוזרת
    {
        DictDto keyValuePairs = new DictDto();
        keyValuePairs.user_contact = helpUser_Contacts;
        keyValuePairs.listContact=helpContacts;
        return keyValuePairs;
    }
    helpContacts.Clear();
    foreach (User_ContactDto update_user_contact in updateUser_Contacts)
    {
        USER_CONTACT user_contact = new USER_CONTACT();
        user_contact.user_contactId = update_user_contact.user_contactId;
        user_contact.userId = update_user_contact.userId;
        user_contact.contactId = update_user_contact.contactId;
        user_contact.contactFirstTitle = update_user_contact.contactFirstTitle;
        user_contact.contactFirstName = update_user_contact.contactFirstName;
        user_contact.contactLastName = update_user_contact.contactLastName;
        user_contact.contactSpouse = update_user_contact.contactSpouse;
        user_contact.contactAddress = update_user_contact.contactAddress;
        user_contact.contactCity = update_user_contact.contactCity;
        user_contact.contactTelephone = update_user_contact.contactTelephone;
        user_contact.contactDetails = update_user_contact.contactDetails;
        user_contact.contactMail = update_user_contact.contactMail;
        user_contact.contactLastTitle = update_user_contact.contactLastTitle;

        du.UpdateUser_Contact(user_contact);
    }
    return null;
}
```

צד שרת

המחלקות לפי שכבות:

כאמור לעיל חילקנו את המחלקות לשכבות:

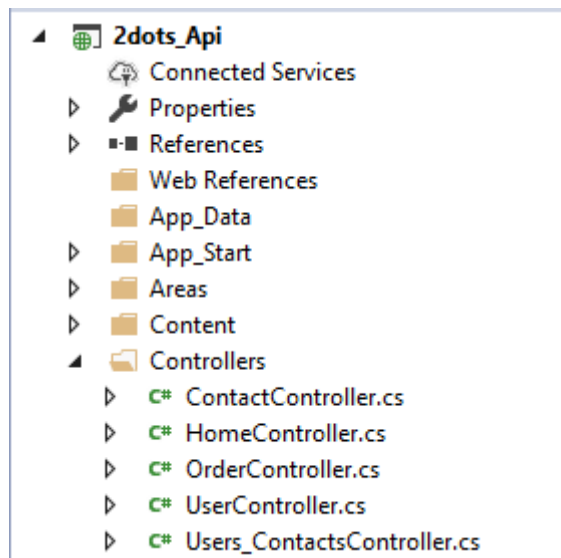


WebAPI Service:

בשכבה זו ישנו שימוש בשירות webAPI המאפשר התממשקות לסביבת הלקוח דרך הדפדפן.

השירות מאפשר קריאות מסוג GET, PUT, POST, DELETE

Controllers:



```
public class ContactController : ApiController ❖
```

```
    public List<DictDto> MatchContacts(int id)
```

מחזיר רשימה של אנשי קשר שלא נמצאה להם התאמה מוחלטת. .

```
    public void Post(ContactDto contact)
```

מוסיף איש קשר חדש.


```
public class OrderController : ApiController ❖
```

```
public object[] GetAllOrders()  
מחזיר רשימה של כל ההזמנות.
```

```
public object GetAllOrderDetails(int id)  
מחזיר את פרטי המזמין.
```

```
public List<User_ContactDto> GetTableUserContacts(int id)  
מחזיר את רשימת ה- user_contact.
```

```
public OrderDto AddOrder(OrderDto order)  
מוסיף הזמנה.
```

```
public void UploadFiles(int id)  
מעלה את קובץ הכתובות ובודק התאמה
```

```
public class UserController : ApiController ❖
```

```
public List<UserDto> Gett()  
מחזיר את כל רשימת המשתמשים.
```

```
public UserDto GetByMail(string mail, string password)  
מחזיר את כל פרטי המשתמש לפי מייל וסיסמא.
```

```
public UserDto AddUser(UserDto userDto)  
מוסיף משתמש חדש.
```

```
public class Users_ContactsController : ApiController ❖
```

```
public List<User_ContactDto> Get()  
מחזיר את כל ה- user_contact.
```

```
public String ExsportExelWhithOutMail(int id, String userName)  
יוצר קובץ אקסל עם האנשי קשר ללא מייל.
```

```
public List<User_ContactDto> GetUser_ContactsById(int id)  
מחזיר איש קשר לפי id
```

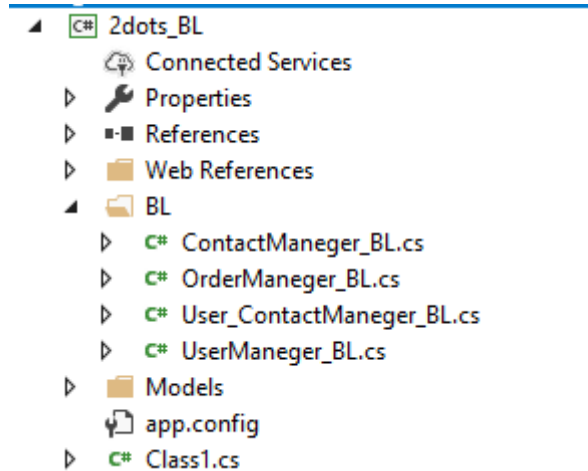
```
public Boolean sentInvationEmails(int id)  
שולח את ההזמנות לאנשי הקשר.
```

```
public List<User_ContactDto> UpdateUser_ContactByUserContactId  
(List <User_ContactDto> user_ContactsDto)  
עידכון מייל של איש קשר אצל הלקוח.
```

:BL Business Layer

פרוט :

בשכבה זו ישנה התאמת אובייקטים המתקבלים משירות ה webAPI ונשלחים אל ה DAL ולהיפך – אובייקטים המקבלים משכבת Data ונשלחים לשירות.



```
public class ContactManeger_BL ❖  
  
public static List<ContactDto> GetAllContacts()  
{  
    ContactDL d = new ContactDL();  
    List<CONTACTS> listContacts = d.GetContacts();  
    List<ContactDto> listContactsDto = new List<ContactDto>();  
    foreach (CONTACTS contact in listContacts)  
    {  
        ContactDto contactDto = new ContactDto();  
        contactDto.contactId = contact.contactId;  
        contactDto.contactFirstName = contact.contactFirstName;  
        contactDto.contactLastName = contact.contactLastName;  
        contactDto.contactSpouse = contact.contactSpouse;  
        contactDto.contactAddress = contact.contactAddress;  
        contactDto.contactCity = contact.contactCity;  
        contactDto.contactTelephone = contact.contactTelephone;  
        contactDto.contactDetails = contact.contactDetails;  
        contactDto.contactMail = contact.contactMail;  
        listContactsDto.Add(contactDto);  
    }  
    return listContactsDto;  
}
```

```

public static ContactDto GetContactById(int id)
{
    ContactDL d = new ContactDL();
    CONTACTS contact = d.GetContactById(id);
    ContactDto contactDto = new ContactDto();
    contactDto.contactId = contact.contactId;
    contactDto.contactFirstName = contact.contactFirstName;
    contactDto.contactLastName = contact.contactLastName;
    contactDto.contactSpouse = contact.contactSpouse;
    contactDto.contactAddress = contact.contactAddress;
    contactDto.contactCity = contact.contactCity;
    contactDto.contactTelephone = contact.contactTelephone;
    contactDto.contactDetails = contact.contactDetails;
    contactDto.contactMail = contact.contactMail;
    return contactDto;
}

public static void InsertContact(ContactDto contactDto)
{
    ContactDL d = new ContactDL();
    CONTACTS contact = new CONTACTS();
    contact.contactId = contactDto.contactId;
    contact.contactFirstName = contactDto.contactFirstName;
    contact.contactLastName = contactDto.contactLastName;
    contact.contactSpouse = contactDto.contactSpouse;
    contact.contactAddress = contactDto.contactAddress;
    contact.contactCity = contactDto.contactCity;
    contact.contactTelephone = contactDto.contactTelephone;
    contact.contactDetails = contactDto.contactDetails;
    contact.contactMail = contactDto.contactMail;
    d.InsertContact(contact);
}

public static List<DictDto> MatchContacts(int userId) //בדיקת התאמה
{
    List<DictDto> list = new List<DictDto>();
    List<User_ContactDto> user_Contacts =
    User_ContactManeger_BL.GetUser_ContactsByIdWithStatus1(userId);
    foreach (User_ContactDto user_contact in user_Contacts)
    {
        list.Add(MatchContactsHelp(user_contact, userId));
    }
    list.RemoveAll(item => item == null);
    return list;
}

```

```

public static DictDto MatchContactsHelp(User_ContactDto user_contact, int
userId)
{
    ContactDL d = new ContactDL();
    User_ContactDL du = new User_ContactDL();
    List<ContactDto> contacts = GetAllContacts();
    List<User_ContactDto> updateUser_Contacts = new
List<User_ContactDto>();
    List<ContactDto> returnContacts = new List<ContactDto>();
    List<ContactDto> helpContacts = new List<ContactDto>();
    User_ContactDto returnUser_Contacts = new User_ContactDto();
    User_ContactDto helpUser_Contacts = new User_ContactDto();
    int numOfMatch;
    bool flag = false;
    numOfMatch = 0;
    flag = false;
    foreach (ContactDto contact in contacts)
    {
        numOfMatch = 0;
        if (user_contact.contactTelephone != null &&
contact.contactTelephone != null&& user_contact.contactTelephone != "")// האם
לשתיהם יש טלפון
        {
            if (user_contact.contactTelephone ==
contact.contactTelephone )// הטלפון שווה
            {
                user_contact.contactMail = contact.contactMail;// עדכון
                user_contact.contactId = contact.contactId;// עדכון
                updateUser_Contacts.Add(user_contact);
                break;
            }
            if (user_contact.contactFirstName == contact.contactFirstName)
                numOfMatch++;
            if (user_contact.contactLastName == contact.contactLastName)
                numOfMatch++;
            if (user_contact.contactCity == contact.contactCity)
                numOfMatch++;
            if (user_contact.contactAddress == contact.contactAddress)
                numOfMatch++;
            if (numOfMatch == 4)// אם כל הפרטים תואמים
            {
                user_contact.contactMail = contact.contactMail;// עדכון
                user_contact.contactId = contact.contactId;// עדכון
                updateUser_Contacts.Add(user_contact);
                break;
            }
            else if (numOfMatch >= 2)
            {
                helpUser_Contacts = user_contact;
                helpContacts.Add(contact);
                flag = true;
            }
        }
    }
    if (flag == true)// אם לא נמצא שום התאמה להכניס לרשימה שחוזרת
    {
        DictDto keyValuePairs = new DictDto();
        keyValuePairs.user_Contact = helpUser_Contacts;
        keyValuePairs.listContact=helpContacts;
        return keyValuePairs;
    }
    helpContacts.Clear();
}

```

```

        foreach (User_ContactDto update_user_contact in
updateUser_Contacts)
        {
            USER_CONTACT user_Contact = new USER_CONTACT();
            user_Contact.user_contactId =
update_user_contact.user_contactId;
            user_Contact.userId = update_user_contact.userId;
            user_Contact.contactId = update_user_contact.contactId;
            user_Contact.contactFirstTitle =
update_user_contact.contactFirstTitle;
            user_Contact.contactFirstName =
update_user_contact.contactFirstName;
            user_Contact.contactLastName =
update_user_contact.contactLastName;
            user_Contact.contactSpouse = update_user_contact.contactSpouse;
            user_Contact.contactAddress =
update_user_contact.contactAddress;
            user_Contact.contactCity = update_user_contact.contactCity;
            user_Contact.contactTelephone =
update_user_contact.contactTelephone;
            user_Contact.contactDetails =
update_user_contact.contactDetails;
            user_Contact.contactMail = update_user_contact.contactMail;
            user_Contact.contactLastTitle =
update_user_contact.contactLastTitle;

            du.UpdateUser_Contact(user_Contact);
        }
        return null;
    }

```

```

public class OrderManeger_BL ❖

public static object[] GetAllOrders()
{
    OrderDL d = new OrderDL();
    object[] listOrders = d.GetOrders();
    return listOrders;
}

public static object GetAllOrdersDetails(int orderId)
{
    OrderDL d = new OrderDL();
    object listOrdersDetails = d.GetOrdersDetails(orderId);
    return listOrdersDetails;
}

public static List<User_ContactDto> GetTableUserContacts(int userId)
{
    User_ContactDL d = new User_ContactDL();
    List<USER_CONTACT> listTableUserContact = d.GetAllUser_ContactsById(userId);
    List<User_ContactDto> listUser_ContactDto = new List<User_ContactDto>();
    foreach (USER_CONTACT user_Contact in listTableUserContact)
    {
        User_ContactDto user_ContactDto = new User_ContactDto();
        user_ContactDto.userId = user_Contact.userId;
        user_ContactDto.user_contactId = user_Contact.user_contactId;
        user_ContactDto.contactId = user_Contact.contactId;
        user_ContactDto.contactFirstTitle = user_Contact.contactFirstTitle;
        user_ContactDto.contactFirstName = user_Contact.contactFirstName;
        user_ContactDto.contactLastName = user_Contact.contactLastName;
        user_ContactDto.contactSpouse = user_Contact.contactSpouse;
        user_ContactDto.contactAddress = user_Contact.contactAddress;
        user_ContactDto.contactCity = user_Contact.contactCity;
        user_ContactDto.contactTelephone = user_Contact.contactTelephone;
        user_ContactDto.contactDetails = user_Contact.contactDetails;
        user_ContactDto.contactMail = user_Contact.contactMail;
        user_ContactDto.contactLastTitle = user_Contact.contactLastTitle;
        user_ContactDto.contactLastTitle = user_Contact.contactLastTitle;
        listUser_ContactDto.Add(user_ContactDto);
    }
    return listUser_ContactDto;
}

public static OrderDto InsertOrder(OrderDto orderDto)
{
    OrderDL d = new OrderDL();
    ORDER order = new ORDER();
    order.orderId = orderDto.orderId;
    order.userId = orderDto.userId;
    order.invitationId = orderDto.invitationId;
    order.invitationDetails = orderDto.invitationDetails;
    order.status = orderDto.status;
    order = d.InsertOrder(order);
    orderDto.orderId = order.orderId;
    orderDto.userId = order.userId;
    orderDto.invitationId = order.invitationId;
    orderDto.invitationDetails = order.invitationDetails;
    orderDto.status = order.status;
    return orderDto;
}
}

```

public class User_ContactManeger_BL ❖

```
private static int meser10Id = 5827;

public static User_ContactDto conversionToDto(USER_CONTACT user_Contact)
{
    User_ContactDto user_ContactDto = new User_ContactDto();
    user_ContactDto.user_contactId = user_Contact.user_contactId;
    user_ContactDto.userId = user_Contact.userId;
    user_ContactDto.contactId = user_Contact.contactId;
    user_ContactDto.contactFirstTitle = user_Contact.contactFirstTitle;
    user_ContactDto.contactFirstName = user_Contact.contactFirstName;
    user_ContactDto.contactLastName = user_Contact.contactLastName;
    user_ContactDto.contactSpouse = user_Contact.contactSpouse;
    user_ContactDto.contactAddress = user_Contact.contactAddress;
    user_ContactDto.contactCity = user_Contact.contactCity;
    user_ContactDto.contactTelephone = user_Contact.contactTelephone;
    user_ContactDto.contactDetails = user_Contact.contactDetails;
    user_ContactDto.contactMail = user_Contact.contactMail;
    user_ContactDto.contactLastTitle = user_Contact.contactLastTitle;
    user_ContactDto.status = user_Contact.status;
    return user_ContactDto;
}

public static USER_CONTACT conversionFromDto(User_ContactDto user_ContactDto)
{
    USER_CONTACT user_Contact = new USER_CONTACT();
    user_Contact.user_contactId = user_ContactDto.user_contactId;
    user_Contact.userId = user_ContactDto.userId;
    user_Contact.contactId = user_ContactDto.contactId;
    user_Contact.contactFirstTitle = user_ContactDto.contactFirstTitle;
    user_Contact.contactFirstName = user_ContactDto.contactFirstName;
    user_Contact.contactLastName = user_ContactDto.contactLastName;
    user_Contact.contactSpouse = user_ContactDto.contactSpouse;
    user_Contact.contactAddress = user_ContactDto.contactAddress;
    user_Contact.contactCity = user_ContactDto.contactCity;
    user_Contact.contactTelephone = user_ContactDto.contactTelephone;
    user_Contact.contactDetails = user_ContactDto.contactDetails;
    user_Contact.contactMail = user_ContactDto.contactMail;
    user_Contact.contactLastTitle = user_ContactDto.contactLastTitle;
    user_Contact.status = user_ContactDto.status;
    return user_Contact;
}

public static List<User_ContactDto> GetAllUser_Contact()
{
    User_ContactDL d = new User_ContactDL();
    List<USER_CONTACT> listUsers_Contacts = d.GetUsers_Contacts();
    List<User_ContactDto> listUsers_ContactsDto = new List<User_ContactDto>();
    foreach (USER_CONTACT user_Contact in listUsers_Contacts)
    {
        listUsers_ContactsDto.Add(conversionToDto(user_Contact));
    }
    return listUsers_ContactsDto;
}
```

```

public static List<User_ContactDto> GetUser_ContactsById(int userId)
{
    User_ContactDL d = new User_ContactDL();
    List<USER_CONTACT> listUser_Contacts = d.GetUser_ContactsById(userId);
    List<User_ContactDto> listUser_ContactsDto = new List<User_ContactDto>();
    foreach (USER_CONTACT user_Contact in listUser_Contacts)
    {
        listUser_ContactsDto.Add(conversionToDto(user_Contact));
    }
    return listUser_ContactsDto;
}

public static List<User_ContactDto> GetUser_ContactsByIdWithStatus1(int userId)
{
    User_ContactDL d = new User_ContactDL(); List<USER_CONTACT>
listUser_Contacts = d.GetUser_ContactsByIdWithStatus1(userId);
    List<User_ContactDto> listUser_ContactsDto = new List<User_ContactDto>();
    foreach (USER_CONTACT user_Contact in listUser_Contacts)
    {
        listUser_ContactsDto.Add(conversionToDto(user_Contact));
    }
    return listUser_ContactsDto;
}

public static void InsertUser_Contact(User_ContactDto user_ContactDto)
{
    User_ContactDL d = new User_ContactDL();
    d.InsertUser_Contact(conversionFromDto(user_ContactDto));
}

public static List<User_ContactDto> UpdateUser_Contact(List<User_ContactDto>
userContacts)
{
    User_ContactDL d = new User_ContactDL();
    foreach (User_ContactDto user_ContactDto in userContacts)
    {
        d.UpdateUser_Contact(conversionFromDto(user_ContactDto));
    }
    return userContacts;
}

public static string ExsportExelWhithOutMail(int userId, String userName)
{
    User_ContactDL d = new User_ContactDL();
    return d.SQLToCSV(userId, userName);
}

public static mesereser.CLoginInfo LoginInfo()
{
    mesereser.Services services = new mesereser.Services();
    mesereser.CLoginInfo info = new mesereser.CLoginInfo();
    info.UserName = "314906090@mbby.co.il";
    info.Password = "noa314";
    return info;
}

```



```

public static void AddGroup(int userId)
{
    mesereser.Services services = new mesereser.Services();
    UserDto user = UserManeger_BL.GetUserById(userId);
    OrderDL dL = new OrderDL();
    services.AddGroup(LoginInfo(), meser10Id, "order:" + dL.GetOrderByUserId
(userId).orderId + " name:" + user.userFirstName + " " + user.userLastName);
}

public static void CreateContact(int userId)
{
    List<User_ContactDto> user_contacts = GetUser_ContactsById(userId);
    UserDto user = UserManeger_BL.GetUserById(userId);
    foreach (User_ContactDto user_contact in user_contacts)
    {
        mesereser.Services services = new mesereser.Services();
        services.CreateContact (LoginInfo(), meser10Id, user.userMail, user_contact
.contactMail, user_contact.contactFirstName, user_contact.contactLastName,
user_contact.contactTelephone, user_contact.contactAddress, user_contact
.contactCity, null, user_contact.contactFirstTitle,
user_contact.contactLastTitle, null, null, null);
    }
}

public static Boolean CreateAndSendEMail(int userId)
{
    mesereser.Services services = new mesereser.Services();
    string[] a = new string[1];
    UserDto user = UserManeger_BL.GetUserById(userId);
    a[0] = user.userMail;
    services.CreateAndSendEMail(LoginInfo(), meser10Id, "invatation",
"<html>< head ></ head >< body dir = 'rtl' ><h5> aaa</ h5 ><span
class='clsTemplateField' fld='ContactName'>שם מלא של הנממן</span></ body ></ html
>< br />< title ></ title >", a, null);
    return true;}

public static List<DictDto> LoadExelFile(string path, int userId)
{
    @"M:\FrontEnd\file1.csv"
    List<User_ContactDto> listToInsert;
    Encoding.GetEncoding(38598)
using (var reader = new StreamReader(@path, Encoding.GetEncoding(38598)))using
(var reader = new StreamReader(@"C:\Users\NOA\Desktop\project\abc2008.csv"))
{
    listToInsert = new List<User_ContactDto>();
    var firstLine = reader.ReadLine();
    while (!reader.EndOfStream){
        var line = reader.ReadLine();
        var values = line.Split(',');
        User_ContactDto user_contact = new User_ContactDto();
        user_contact.userId = userId;
        user_contact.contactFirstTitle = values[0];
        user_contact.contactFirstName = values[1];
        user_contact.contactLastName = values[2];
        user_contact.contactSpouse = values[3];
        user_contact.contactLastTitle = values[4];
        user_contact.contactAddress = values[5];
        user_contact.contactCity = values[6];
        user_contact.contactTelephone = values[7];
        user_contact.contactDetails = values[8];
        user_contact.contactMail = values[9];
        listToInsert.Add(user_contact);
    }
}

```

```

public class UserManeger_BL ❖

public static List<UserDto> GetAllUsers()
{
    UserDL d = new UserDL();
    List<USERS> users = d.GetUsers();
    List<UserDto> listUsers = new List<UserDto>();
    foreach (USERS u in users)
    {
        UserDto user = new UserDto();
        user.userId = u.userId;
        user.userFirstName = u.userFirstName;
        user.userLastName = u.userLastName;
        user.userMail = u.userMail;
        user.userTelephone = u.userTelephone;
        user.userPassword = u.userPassword;
        listUsers.Add(user);
    }
    return listUsers;
}

public static UserDto GetUserById(int userId)
{
    UserDL d = new UserDL();
    USERS user = d.GetUserById(userId);
    UserDto userDto = new UserDto();
    userDto.userId = user.userId;
    userDto.userFirstName = user.userFirstName;
    userDto.userLastName = user.userLastName;
    userDto.userMail = user.userMail;
    userDto.userTelephone = user.userTelephone;
    userDto.userPassword = user.userPassword;
    return userDto;
}

public static UserDto GetUserByMail(string mail, string password)
{
    UserDL d = new UserDL();
    USERS user = d.GetUserByMail(mail, password);
    if (user == null)
        return null;
    UserDto userDto = new UserDto();
    userDto.userId = user.userId;
    userDto.userFirstName = user.userFirstName;
    userDto.userLastName = user.userLastName;
    userDto.userMail = user.userMail;
    userDto.userTelephone = user.userTelephone;
    userDto.userPassword = user.userPassword;
    return userDto;
}

public static UserDto InsertUser(UserDto userDto)
{
    UserDL d = new UserDL();
    USERS user = new USERS();
    user.userId = userDto.userId;
    user.userFirstName = userDto.userFirstName;
    user.userLastName = userDto.userLastName;
    user.userMail = userDto.userMail;
    user.userTelephone = userDto.userTelephone;
    user.userPassword = userDto.userPassword;
}

```

```

user=d.InsertUser(user);

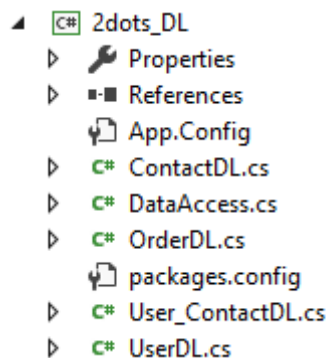
userDto.userId = user.userId;
userDto.userFirstName = user.userFirstName;
userDto.userLastName = user.userLastName;
userDto.userMail = user.userMail;
userDto.userTelephone = user.userTelephone;
userDto.userPassword = user.userPassword;
return userDto;

```

}

DAL

בשכבה זו מתנהלים שליפה, מחיקה, הוספה ועדכון של מסד הנתונים ע"י שימוש ב EntityFramework.



Contact ❖

```

public class ContactDL
{
    //Select Contact
    public List<CONTACTS> GetContacts()
    {
        var db = new TowDotsEF();
        return (from c in db.CONTACTS
                select c).ToList();
    }

    //Select Contact by id
    public CONTACTS GetContactById(int id)
    {
        var db = new TowDotsEF();
        return (from c in db.CONTACTS
                where c.contactId == id
                select c).FirstOrDefault();
    }

    //Insert Contact
    public void InsertContact(CONTACTS contact)
    {
        var db = new TowDotsEF();
        db.CONTACTS.Add(contact);
        db.SaveChanges();
    }
}

```

Order ❖

```
public class OrderDL
{
    //Select Order
    public object[] GetOrders()
    {
        var db = new TowDotsEF();
        return (from o in db.ORDER
                join u in db.USERS
                on o.userId equals u.userId
                where o.status == 1
                select new { o.orderId, userName = u.userLastName + " " +
u.userFirstName }).ToArray();
    }

    public object GetOrdersDetails(int orderId)
    {
        var db = new TowDotsEF();
        return (from o in db.ORDER
                join u in db.USERS
                on o.userId equals u.userId
                where o.orderId==orderId
                select new {u.userId,userName = u.userLastName + " " +
u.userFirstName ,u.userMail,u.userTelephone}).ToArray();
    }
    public ORDER GetOrderByUserId(int userId)
    {
        var db = new TowDotsEF();
        List<ORDER> lisrOrders = (from o in db.ORDER
                                where o.userId == userId
                                select o).ToList();

        return lisrOrders.FirstOrDefault(a => a.orderId ==
lisrOrders.Select(o => o.orderId).Max());
    }

    //Insert Order
    public ORDER InsertOrder(ORDER order)
    {
        var db = new TowDotsEF();
        db.ORDER.Add(order);
        db.SaveChanges();
        return GetOrderByUserId(order.userId);
    }
}
```

User_contact ❖

```
public class User_ContactDL
{
    //Select Users_Contacts
    public List<USER_CONTACT> GetUsers_Contacts()
    {
        using (var db = new TowDotsEF())
        {
            return (from u in db.USER_CONTACT
                    select u).ToList();
        }
    }

    //Select User_Contact by id
    public List<USER_CONTACT> GetUser_ContactsById(int userId)
    {
        var db = new TowDotsEF();
        return (from u in db.USER_CONTACT
                where u.userId == userId & u.contactMail !=
                null & u.contactMail != "" select u).ToList();
    }

    public List<USER_CONTACT>GetUser_ContactsByIdWithStatus1(int userId)
    {
        var db = new TowDotsEF();
        return (from u in db.USER_CONTACT
                where u.userId == userId & u.contactMail ==
                null | u.contactMail == "" & u.status==1
                select u).ToList();
    }

    public List<USER_CONTACT> GetAllUser_ContactsById(int userId)
    {
        var db = new TowDotsEF();
        return (from u in db.USER_CONTACT
                where u.userId == userId
                select u).ToList();
    }

    public USER_CONTACT GetUser_ContactByUser_ContactId(int userContactId)
    {
        var db = new TowDotsEF();
        return (from u in db.USER_CONTACT
                where u.user_contactId == userContactId
                select u).First();
    }

    //Insert User_Contact
    public void InsertUser_Contact(USER_CONTACT User_Contact)
    {
        var db = new TowDotsEF();
        db.USER_CONTACT.Add(User_Contact);
        db.SaveChanges();
    }

    //update User_Contact
    public void UpdateUser_Contact(USER_CONTACT user_contact)
    {
        var db = new TowDotsEF();
        db.USER_CONTACT.AddOrUpdate(user_contact);
        db.SaveChanges();
    }

    //delete User_Contact
}
```

```

public void DeleteUser_Contact(int userContactId)
{
    var db = new TowDotsEF();
    db.USER_CONTACT.Remove(GetUser_ContactByUser_ContactId(userContact
Id));
    db.SaveChanges();
}
//GetUser_ContactsByIdWhithOutMail
public List<USER_CONTACT> GetUser_ContactsByIdWhithOutMail(int userId)
{
    var db = new TowDotsEF();
    return (from u in db.USER_CONTACT
            where u.userId == userId && u.contactMail ==null
            select u).ToList();
}
//SQLToCSV
public String SQLToCSV(int userId,String userName)
{
    const string conStr = @"Data Source =
(localDb)\MSSQLLocalDB; Initial Catalog = 2dots_project;
Integrated Security = True;";
    SqlConnection conn = new SqlConnection(conStr);
    conn.Open();
    SqlCommand cmd = new SqlCommand("select * from
USER_CONTACT where USER_CONTACT.userId=@userId AND
USER_CONTACT.contactMail is null", conn);
    SqlParameter p = new SqlParameter("@userId", userId);
    cmd.Parameters.Add(p);
    SqlDataReader dr = cmd.ExecuteReader();
    using (System.IO.StreamWriter fs = new
System.IO.StreamWriter("C:\\Users\\NOA\\Desktop\\project\\"+
userName + ".csv"))
    {
        // Loop through the fields and add headers
        for (int i = 0; i < dr.FieldCount; i++)
        {
            string name = dr.GetName(i);
            if (name.Contains(","))
                name = "\"" + name + "\"";

            fs.Write(name + ",");
        }
        fs.WriteLine();
        // Loop through the rows and output the data
        while (dr.Read())
        {
            for (int i = 0; i < dr.FieldCount; i++)
            {
                string value = dr[i].ToString();
                if (value.Contains(","))
                    value = "\"" + value + "\"";

                fs.Write(value + ",");
            }
            fs.WriteLine();
        }
        fs.Close();
    }
    conn.Close();
    return "C:\\Users\\NOA\\Desktop\\project\\" + userName
+ ".csv";
}

```

User ❖

```
public class UserDL
{
    //Select User
    public List<USERS> GetUsers()
    {
        var db = new TowDotsEF();
        return (from u in db.USERS
                select u).ToList();
    }

    //Insert User
    public USERS InsertUser(USERS user)
    {
        var db = new TowDotsEF();
        db.USERS.Add(user);
        db.SaveChanges();
        return GetUserByMail(user.userMail, user.userPassword);
    }

    //Select User by Mail
    public USERS GetUserByMail(string mail, string password)
    {
        var db = new TowDotsEF();
        return (from u in db.USERS
                where u.userMail == mail && u.userPassword == password
                select u).FirstOrDefault();
    }

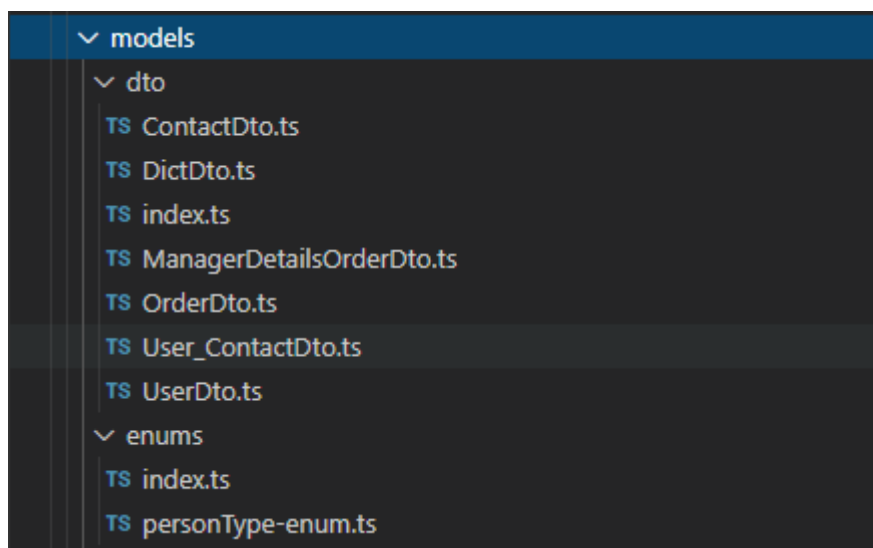
    //Select User by Id
    public USERS GetUserById(int userId)
    {
        var db = new TowDotsEF();
        return (from u in db.USERS
                where u.userId == userId
                select u).FirstOrDefault();
    }
}
```

צד לקוח

צד הלקוח נכתב באנגולר 6.

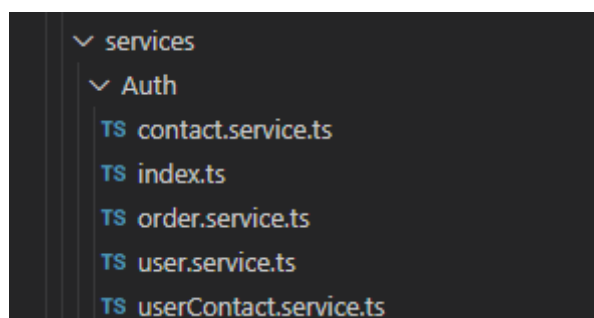
מבנה המערכת: המערכת בנויה ממחלקות, SERVICES, models-קומפוננטות-חלקי התצוגה ו-CSS קבצי סטייל.

Models



Services:

ה SERVICES הינם מחלקות שרותים המשמשות את הקומפוננטות. מכילות מידע משותף ומבצעות חישובים וגישה לשרת.



ContactService

```
@Injectable()
export class ContactService extends BaseApiService {
  constructor(private baseHttpService: BaseHttpService) {
    super('Contact');
  }
  matchContacts(userId:number):Observable<Array<DictDto>>{
    let url = this.actionUrl('MatchContacts');
    return this.baseHttpService.get<Array<DictDto>>(url,userId);
  }
}
```

OrderService

```
addOrder(orderDto: OrderDto): Observable<OrderDto> {
  let url = this.actionUrl('AddOrder');
  if (typeof orderDto.status === "undefined" ||
    typeof orderDto.invitationId === "undefined" || typeof orderDto.invitationDetails === "undefined") {
    return null;
  }
  return this.baseHttpService.post<OrderDto>(url, orderDto);
}
getAllOrders(): Observable<any[]> {
  let url = this.actionUrl('GetAllOrders');
  return this.baseHttpService.get<any[]>(url);
}

getAllOrderDetails(orderId: number): Observable<any[]> {
  let url = this.actionUrl('GetAllOrderDetails');
  return this.baseHttpService.get<any[]>(url, orderId);
}
getTableUserContacts(userId: number): Observable<any[]> {
  let url = this.actionUrl('GetTableUserContacts');
  return this.baseHttpService.get<any[]>(url, userId);
}
upload(frmData: FormData, userId: Number) {
  debugger;
  return this._http.post("http://localhost:10082/api/Order/UploadFiles?id="+userId, frmData);
}
```

UserService

```
getUsers(): Observable<UserDto[]> {
    let url = this.actionUrl('GetUsers');
    return this.baseHttpService.get<UserDto[]>(url);
}

getUserByMail(userMail: string, password: string): Observable<any>
{
    let url = this.actionUrl('GetByMail');
    let params: URLSearchParams = new URLSearchParams();
    if (typeof userMail === "undefined" || typeof password === "undefined")
    {
        userMail = "";
        password = "";
    }
    params.set('mail', userMail);
    params.set('password', password);
    return this.baseHttpService.get<any>(url,params);
}

addUser( userDto:UserDto) :Observable<UserDto>{
    let url = this.actionUrl('AddUser');
    if (typeof userDto.userFirstName === "undefined" || typeof userDto.userPassword === "undefined" || typeof userDto.userFirstName === "undefined" ||
        typeof userDto.userLastName === "undefined" || typeof userDto.userTelephone === "undefined")
    {
        userDto.userFirstName = "";
        userDto.userLastName="";
        userDto.userTelephone="";
        userDto.userPassword = "";
        userDto.userMail="";
    }
    return this.baseHttpService.post<UserDto>(url,userDto);
}
```

UserContactService

```
getUserContacts(userId: number): Observable<User_ContactDto[]> {
    let url = this.actionUrl('GetUser_ContactsById');
    return this.baseHttpService.get<User_ContactDto[]>(url, userId)
;
}
UpdateUserContact(userContacts: User_ContactDto[]): Observable<User_ContactDto[]> {
    let url = this.actionUrl('UpdateUser_ContactByUserContactId');
    return this.baseHttpService.put<User_ContactDto[]>(url, userContacts);
}
craetFileWithOutMail(userId: number, userName: string): Observable<string> {
    let url = this.actionUrl('ExsportExelWhithOutMail');
    let params: URLSearchParams = new URLSearchParams();
    params.set('id', userId.toString());
    params.set('userName', userName);
    return this.baseHttpService.get<string>(url,params);
}
sentInvationEmails(userId: number):Observable<any>{
    let url = this.actionUrl('sentInvationEmails');
    return this.baseHttpService.get<any>(url,userId);
}

updateMail(contactToUpdate:Array<User_ContactDto>):Observable<Array<User_ContactDto>>{
    debugger;
    let url = this.actionUrl('UpdateMail');
    return this.baseHttpService.put<Array<User_ContactDto>>(url,contactToUpdate);
}
```

Components:

```

  ✓ check
    # check.component.css
    <> check.component.html
    TS check.component.ts
  ✓ home-page
    # home-page.component.css
    <> home-page.component.html
    TS home-page.component.ts
  ✓ login-page
    # login-page.component.css
    <> login-page.component.html
    TS login-page.component.ts
  ✓ management-user-contact
    # management-user-contact.component.css
    <> management-user-contact.component.html
    TS management-user-contact.component.ts
  ✓ manager
    # manager.component.css
    <> manager.component.html
    TS manager.component.ts
  ✓ match
    # match.component.css
    <> match.component.html
    TS match.component.ts
  ✓ new-user
    # new-user.component.css
    <> new-user.component.html
    TS new-user.component.ts
  ✓ order
    # order.component.css
    <> order.component.html
    TS order.component.ts
  ✓ order-details
    # order-details.component.css
    <> order-details.component.html
    TS order-details.component.ts
  > services
  > shared
  ✓ subscription
    # subscription.component.css
    <> subscription.component.html
    TS subscription.component.ts
  ✓ table-user-contact
    TS products.ts
    # table-user-contact.component.css
    <> table-user-contact.component.html
    TS table-user-contact.component.ts

```

את צד הלקוח בנינו בטכנולוגיה חדשנית של PAGE ONE הקומפוננטות כולן נטענות לדף אחד, כך שמתבצעת טעינת רכיבי תצוגה ולא דף שלם.

```
TS app-routing.module.ts
<> app.component.html
TS app.component.ts
TS app.module.ts
```

כל קומפוננטה מורכבת מדף HTML וממחלקה ב typescript שמריצה את הקוד לתפעול ה HTML וכן דף CSS שמכיל את ה style של הדף.

בדיקות המערכת:

סוגי הבדיקות:	פרוט:	תיאור:
בדיקת ממשקים	בדיקת התממשקות עם ממשק לצורך קבלת נתונים. בדיקה דרך הקוד כי ההתחברות תקינה כולל קבלת נתונים נכונים.	בדיקה עברה בהצלחה
בדיקת פונקציונאליות	בדיקה כי כל המסכים פועלים כשורה, כגון: ולידציות על נתונים, ניתוב נכון בין המסכים וכו'.	בדיקה עברה בהצלחה
בדיקת תאימות	בדיקה כי האתר נתמך בשני הדפדפנים: Exploreri Chrom .	בדיקה עברה בהצלחה
בדיקות תוכן	בדיקת מלל תקין באתר, עקביות באיות מלל וכן תקינות ניסוח מילים.	בדיקה עברה בהצלחה
בדיקות אבטחת מידע	גם כאשר העברת הנתונים לצורך הדפסת התוצאות, הנתונים מועברים בגוף הבקשה ולא ב URL.	בדיקה עברה בהצלחה
בדיקת שימושיות	בדיקות נוחות השימוש ויעילות העיצוב של האתר. לדוגמא: תצוגת שלבי ההזמנה	בדיקה עברה בהצלחה

על מנת לבדוק את תקינות המערכת ראשית בדקנו אחרי כתיבת כל פונקציה, בין בצד השרת ובין בצד הלקוח אם הפונקציה מבצעת את האמור עליה, אף במקרי קצה, שליחת פרמטרים שגויים, הכנסת ערכים לא מתאימים וכו'.

בנוסף לקראת סיום ביצוע הפרויקט בדקנו נכונות תהליכים שונים, לראות שהאינטגרציה בין צד השרת והלקוח ובין הפונקציות השונות והמצבים השונים שיכולים לקרות במערכת עובדים כרצוי.

ולבסוף על מנת לבדוק את פשטות ויעילות המערכת ביקשנו מן הלקוח לנסות דרכו במערכת, ולראות האם אופן ביצוע הפעולות השונות טריוויאלי וברור, והאם הפונקציונליות עונה על כל הדרישות

בקרת גירסאות:

ניהול הגרסאות מתבצע בTFS, כאשר בכל עליה נוספת לאויר הגרסה הקודמת נשמרת בהיסטוריה, ע"מ לאפשר שיחזור וכן השוואות בין הגרסאות.

מה תרם לנו הפרויקט:

- לימוד ושליטה בסביבת NET, בשפת C# וב-SQL Server.
- התמודדות בפיתוח פרויקט בהיקף גדול.
- פיתוח הידע והשליטה בפלטפורמת Angular 6.
- ניסיון בעמידה בדרישות לקוח ובעבודה מול לקוח.
- ניסיון בניתוח מערכות.
- ניסיון בתכנון מסד נתונים מתאים לדרישות.
- ניסיון בבדיקות מקיפות.
- לימוד עצמאי של טכנולוגיות חדשות.
- יכולת איתור צורך בהוספת ספריות חיצוניות והוספתן.
- זיהוי בעיות ופיתוח היכולת להתמודד איתן באופן עצמאי.
- ידע בהקמה עצמאית של סביבה לבניית פרויקט- התקנות שונות הנצרכות והיכולות לעזור
- יכולת עבודה בצוות.
- ניסיון בהתממשקות ל API חיצוני והבנת דרך השימוש בו.

ביביליוגרפיה:

- Stackoverfolw
- primeNG
- GitHub
- השתמשנו רבות במנוע החיפוש של google ובסוגי עזרה שונים.

נספחים - דוגמאות לקטעי קוד עיקריים:

צד לקוח:

```
newUser() {
  this.flag = false;
  let navigationExtras: NavigationExtras = {
    queryParams: {
      "btn": this.btnn
    }
  };
  this.router.navigate(["/new-user"],navigationExtras).then((e) => {
    if (e) {
      console.log("Navigation is successful!");
    } else {
      console.log("Navigation has failed!");
    }
  });
}

subscription() {
  this.flag = false;
  this.router.navigate(["/subscription"]).then((e) => {
    if (e) {
      console.log("Navigation is successful!");
    } else {
      console.log("Navigation has failed!");
    }
  });
}
```

```
getUserContacts() {
  let navigationExtras: NavigationExtras = {
    queryParams: {
      "id": this.userId
    }
  };
  debugger;
  this.router.navigate(["/table-user-
contact"], navigationExtras).then((e) => {
    if (e) {
      console.log("Navigation is successful!");
    } else {
      console.log("Navigation has failed!");
    }
  });
}
```



```

orderDetails(lod){
  let navigationExtras: NavigationExtras = {
    queryParams: {
      "orderId": lod.orderId
    }
  };
  this.router.navigate(["/order-
details"],navigationExtras).then((e) => {
    if (e) {
      console.log("Navigation is successful!");
    } else {
      console.log("Navigation has failed!");
    }
  });
});

```

```

saveAndContinue() {
  if (this.contactToUpdate.length > 0) {
    this.userService.updateUserContact(this.contactToUpdate).s
    ubscribe(
      data => {
        let navigationExtras: NavigationExtras = {
          queryParams: {
            "id": this.userId
          }
        };
        this.router.navigate(["/management-user-
contact"], navigationExtras).then((e) => {
          if (e) {
            console.log("Navigation is successful!");
          } else {
            console.log("Navigation has failed!");
          }
        });
      }
    );
  } else {
    let navigationExtras: NavigationExtras = {
      queryParams: {
        "id": this.userId
      }
    };
    this.router.navigate(["/management-user-
contact"], navigationExtras).then((e) => {
      if (e) {
        console.log("Navigation is successful!");
      } else {
        console.log("Navigation has failed!");
      }
    }
  }
}

```

```

login() {
  this.items1 = [
    { label: 'משתמש חדש', icon: 'fa fa-fw fa-user', url: 'http://localhost:4200/new-user' },
    // { label: 'manager', icon: 'fa fa-fw fa-user', url: 'http://localhost:4200/manager' },
    { label: 'מנוי', icon: 'fa fa-fw fa-user', url: 'http://localhost:4200/subscription' }
  ];
  this.user.userFirstName = this.userFirstName
  this.user.userLastName = this.userLastName
  this.user.userPassword = this.userPassword
  this.user.userTelephone = this.userTelephone
  this.user.userMail = this.userMail

  this.userService.addUser(this.user).subscribe(
    data => {
      if (data != null) {
        alert("ברוכה הבאה שמחים להצטרפותך" + " " + data.userFirstName)

        let navigationExtras: NavigationExtras = {
          queryParams: {
            "id": data.userId
          }
        };

        this.router.navigate(["/order"], navigationExtras).then((e) =
> {
          if (e) {
            console.log("Navigation is successful!");
          } else {
            console.log("Navigation has failed!");
          }
        }
      );
    }
    else {
      alert("הוספת המשתמש נכשלה")
      this.user.userFirstName = "";
      this.user.userLastName = "";
      this.user.userPassword = "";
      this.user.userTelephone = "";
      this.user.userMail = "";
    }
  });
}

```

```

craetFileWithOutMail() {
    this.userContactService.craetFileWithOutMail(this.orderDetails[0].userId, this.orderDetails[0].userName).subscribe(
        data => {
            alert("the file craeted on path " + data)
        }
    );
}

sentInvationEmails() {
    this.userContactService.sentInvationEmails(this.orderDetails[0].userId).subscribe(
        data => {
        }
    );
}

changeMail(uc) {
    if (this.contactToUpdate.includes(uc))
        this.contactToUpdate.forEach((item, index) => {
            if (item.user_contactId === uc.user_contactId)
                this.contactToUpdate.splice(index, 1);
        });
    this.contactToUpdate.push(uc);
}

```

צד שרת:

כתוב בצורה מפורטת בעמוד 31