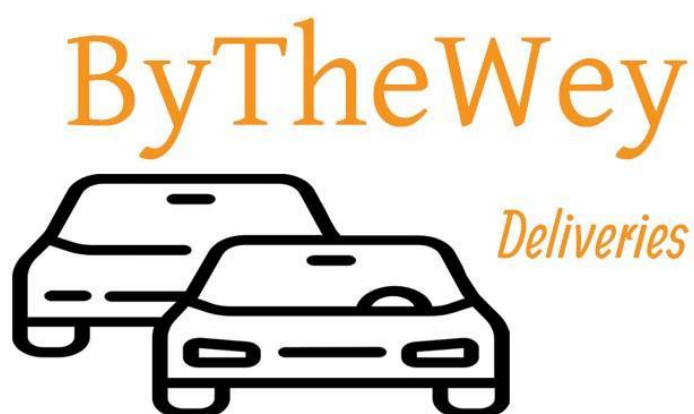


# ספר פרויקט



שם הפרויקט: BYTHEWAY deliveries

שם התלמידה: שירה בורה

ת.ז: 212423693

סמינר: גברא

מנחה: יעל עמר

תאריך הגשה: יולי

## תוכן

4	הצעת פרויקט
6	מבוא / תקציר
6	הרקע לפרויקט
7	תהליך המחקר
7	סקירת ספרות
8	מטרות ויעדים
	אתגרים 8
9	מדדי הצלחה
9	תיאור המצב הקיים
10	רקע תאורטי
10	ניתוח חלופות מערכת
11	תיאור החלופה הנבחרת והנימוקים לבחירה
12	אפיון המערכת
12	ניתוח דרישות המערכת
13	מודול המערכת
13	אפיון פונקציונאלי
14	ביצועים עיקריים
14	אילוצים
15	תיאור הארכיטקטורה
15	הארכיטקטורה של הפיתרון המוצע בפורמט של Design level Down-Top
16	תיאור הרכיבים בפתרון
18	ארכיטקטורת רשת
18	תיאור פרוטוקולי התקשורת
18	שרת – לקוח
19	ניתוח ותרשים use case של המערכת המוצעת
19	
20	רשימת use case
20	תיאור ה-use case העיקריים של המערכת
20	מבני נתונים בהם משתמשים בפרויקט
20	תרשים מחלקות
22	תיאור המחלקות
25	תיאור התוכנה
25	אלגוריתמים מרכזיים
29	קוד האלגוריתם

	בס"ד
36	תיאור מסד הנתונים
36	פירוט הטבלאות ב- Data Base
42	מדריך למשתמש
42	תיאור המסכים
43	מדריך למשתמש
43	צילומי מסכים
46	בדיקות והערכה
46	ניתוח יעילות
46	אבטחת מידע
47	מסקנות
48	פיתוח עתידי
49	ביבליוגרפיה

## הצעת פרויקט

סמל מוסד: 560342

שם מכללה: סמינר תורני בית יעקב.

שם הסטודנט: שירה בורה

ת.ז. הסטודנט: 212423693

שם הפרויקט: by the way

תיאור הפרויקט:

אפליקציה שתאפשר שירות משלוחים שיתופי.

המשתמש שנכנס למערכת צריך להגדיר פרופיל (משלוחן או לקוח)

אם בחר בפרופיל משלוחן - הוא ימלא טופס ובו יגדיר נסיעה אפשרית שלו (תאריך, כתובת מקור, כתובת יעד, שעה וכו')

אם בחר בפרופיל לקוח - הוא ימלא טופס ובו יגדיר פרטים על משלוח שרוצה שיעשו (תאריך, כתובת מקור, כתובת יעד, שעה וכו')

כמו כן בבחירת פרופיל לקוח, תעמוד בפניו האפשרות להגיב על משלוח שנעשה לו.

לאחר שליחת הטופס (או מצד המשלוחן או מצד הלקוח) המערכת תציג למשתמש את התוצאות - בהתאם לחישובים שלה.

אופן חישוב המערכת :

המערכת מחשבת תחילה התאמה בסיסית בין משלוחן ללקוח (כתובת מקור ויעד, תאריך, שעה) לאחר שיש התאמה בסיסית המערכת מתחשבת ב:

1. המערכת מחשבת לפי רייטינג - ככל שרייטינג של משלוחן יותר גבוה כך הוא יקבל רשימה יותר גדולה של לקוחות (משלוחים) שמתאימים לדרישותיו וכן להפך.

במידה והרייטינג של משלוחן נמוך מהרף הקבוע במערכת הוא לא יוכל לקיים עוד משלוחים.

כאשר יהיה עודף משלוחים (בלי מישולחנים) המערכת לא תתחשב ברייטינג

2. המערכת תתחשב גם בבקשות הקודמות של המשלוחן כדי שלא ייווצר מצב שהוא ביקש כמה פעמים לבצע משלוח ולא קיבל.

התשלום - חישוב הסכום לתשלום יעשה לפי קילומטרים ותעריף קבוע. הפרויקט לא מטפל באופן העברת התשלום.

לאחר החישוב ובדיקת ההתאמות המערכת תהיה אחראית לקשר בין המשלוחן ללקוח. ע"י צ'אט שנפתח בין המשלוחן ללקוח.

הגדרת הבעיה האלגוריתמית:

\*חיפוש וחישוב המישלוחן המתאים מבין כל מאגר המישלוחנים החישוב יעשה לפי:

1. התאמה בסיסית של כתובת ותאריך.

2. הבקשות הקודמות שלו (כמה ביקש כמה קיבל) כדי למנוע מצב של הרעבה

בס"ד

3. הרייטינג של המישלוחן - ככל שהרייטינג יותר גבוה יהיה לו יותר סיכויים וכן להפך

ידע תאורטי בתחום הפרויקט: כיום אין מענה לבעיית המשלוחים - כלומר אין דרך יעילה וזמינה לשלוח משלוחים לכל מקום מכל מקום לכן המערכת תיעל את שירות המשלוחים כך שכל אחד יוכל לעשות משלוחים לכל רחבי הארץ בזמן הקצר ביותר.

תהליכים עיקריים בפרויקט:

1. הזנת נתונים במערכת ושמירתם

2. חישוב

3. הצגת התאמה בין משלוחן ללקוח או להיפך ופתיחת צ'אט להמשך תקשורת עצמאי.

תיאור הטכנולוגיה:

צד שרת: C#

צד לקוח: angular

מוד נתונים: SQL

פרוטוקולי תקשורת :

לוחות זמנים:

1. חקר המצב הקיים -
2. הגדרת הדרישות -
3. אפיון המערכת -
4. אפיון בסיס הנתונים -
5. עיצוב המערכת -
6. בנית התוכנה -
7. בדיקות -
8. הכנת תיק פרויקט -
9. הטמעת המערכת -
10. הגשת פרויקט סופי -

חתימת הסטודנט: שירה בורה

חתימת הרכז המגמה:

אישור משרד החינוך:

## מבוא / תקציר

### הרקע לפרויקט

חיפשתי רעיון לפרויקט שיהא שימושי ויעל ויוסיף לחיי היום של האנשים, חשבתי על דברים שלא קיימים היום, דברים חסרים שיכולים להועיל. כמו כן חיפשתי משהו חדש שיגרום לי לפתח מיומנויות חשיבה חדשות ולהכיר דברים חדשים תוך כדי כתיבת הפרויקט.

בחרתי ברעיון של שירות משלוחים שיתופי, מה שעזר לי להגיע לרעיון הזה הוא החסר- קרה לי הרבה פעמים שרציתי להעביר משהו ולא יכולתי להעביר - והיו מקרים ממש דחופים, ותמיד חיפשתי איזה אתר או אפליקציה שיתנו את השירות הזה. הרעיון שעומד מאחורי הפרויקט זה לעזור גם ללקוח וגם למשלוחן(הנהג), הלקוח ירוויח מזה שיוכל לשלוח משלוח מכל מקום לכל מקום בכל זמן נתון – מה שלא קיים היום, וכן גם המשלוחן (הנהג) יצא מורווח כאשר הוא יקבל תשלום על נסיעה שממילא הוא עושה, יוצא ששתי הצדדים יצאו מורווחים.

אני חושבת שזה רעיון טוב בתור רעיון לפרויקט גמר משום שיש בו תועלת רבה וזה יכול לתרום לחיי היום יום. מה גם שהשירות הזה לא קיים, כיום יש אפשרות לעשות משלוחים רק מעסקים מסוימים ללקוחות שלהם אבל לאזרח הפשוט אין באמת שירות של משלוחים מסודר שהוא יכול להעביר משהו למישהו אחר. לאחר הפיתוח כל אזרח פשוט יכול להעביר משלוח(חבילה) מכל מקום לכל מקום במחיר סמלי.

את האלגוריתם עצמו כתבתי לבד. ועל מנת לחשב מרחקים התממשקתי עם גוגל מפות והשתמשתי בפונקציות כתובות שלהם.

כיון שהאפליקציה שלי מחייבת שימוש נרחב בתחומים שונים מעולם התכנות, בהם תחומים אליהם לא נחשפתי במהלך הלימודים, נדרשה ממני יכולת למידה עצמית גבוהה מאד והיא אכן הלכה והשתפרה במהלך הפיתוח כאשר נדרשתי להתגבר על בעיות שונות ולמצוא להן פתרונות מגוונים.

הפרויקט מכיל מספר חלקים עיקריים:

האפליקציה: אפליקציה חדשנית לסיוע בשירות המשלוחים בצורה יעילה ונעימה האפליקציה פותחה בטכנולוגיה המתקדמת ביותר ובממשק משתמש נוח וידידותי כדי למקסם את חווית המשתמש.

הספר: המעיין בספר זה ימצא תיאור נרחב של הפרויקט הן בפן הלוגי והן בפן הגרפי-חיצוני שישתלב במערכת הכללית. במהלך הספר ניתן להבחין בעקרונות התכנון של המערכת, שלבי החישוב, מדגם של אלגוריתמים הנכללים בפרויקט, תהליכים שקיימים במערכת, תרשימים ועוד. בנוסף מכיל הספר מדריך למשתמש בו נמצאים צילומי מסך והסברים כיצד ניתן להשתמש במגוון האפשרויות אותן מספקת האפליקציה.

חלק משמעותי נוסף בספר הוא הלמידה הרבה שהושקעה בפרויקט. לאלגוריתם המוגמר ודרך הפעולה הסופית שנבחרה קדמו למידה עמוקה של החומר ואפשרויות פתרון שונות שנוסו ונשללו במהלך העבודה עד לבחירת האלגוריתם המתאים ביותר.

הממשק: האפליקציה פותחה באנגולר והושקעה מחשבה רבה בעיצוב נקי ונעים לעין, בנוחות השימוש ובפשטות ההפעלה כדי לספק למשתמש חוויית שימוש מושלמת. בין השאר שמנו דגש על ניצול פונקציונליות מעניינת ועיצוב נעים ומרענן.

בחרתי בשם bytheway deliveries כי הרעיון הוא שירות משלוחים על הדרך(של הנהג)

כולי תקווה שהאפליקציה באמת תהפוך לשימושית ותתרום לכולנו.

## תהליך המחקר

הדבר הראשון שעשיתי כשרציתי להתחיל לפתח את הפרויקט זה עבודת מחקר, חקרתי את הנושא לעומק בדקתי את החסרים העלויות איך עובד גוגל מפות ועוד...

בררתי אצל אנשים מה הם עושים כיום בלי השירות הזה, ומה היו רוצים שיהא, מה יתרום ויוסיף להם, חיפשתי חומרים לקריאה בנושא וכן חיפשתי אלגוריתם שאוכל להתבסס עליו-בגלל שלא מצאתי ישבתי וכתבתי את האלגוריתם לבד תוך שימוש בשירותי גוגל מפות -בפונקציות של חישוב מרחקים. כמו כן קראתי המון על גוגל מפות איך זה עובד, איך מתממשקים לגוגל מפות, מה החיובים שלהם ועוד.

מצאתי ב API של גוגל מפות פונקציה של מטריצת מרחקים. מטריצה זו מחשבת מרחקים וזמן בין מספר מקורות ויעדים שהלקוח מכניס. לקחתי את הפונקציה והתאמת את הפרויקט שלי ע"י שאפשרתי למשתמש להכניס למטריצה רק כתובת מקור אחת וכתובת יעד אחת וכן הצגתי למשתמש רק את חישובי המרחק ולא את חישובי הזמן -מה שלא רלוונטי לפרויקט שלי. לשם השימוש בגוגל מפות API פתחתי חשבון, יצרתי מפתח ייחודי לשימוש, והטמעת את ה API של גוגל מפות אצלי בפרויקט.

## סקירת ספרות

### האתרים שבהם נעזרתי:

- אלגוריתם-כתבתי את רובו לבד ונעזרתי בגוגל מפות API לצורך חישוב מרחקים האתר שבו השתמשתי לצורך זה הוא google maps platform
- שפות-נעזרתי ב stack overflow על מנת לשאול שאלות על כל מיני באגים שהיו לי בצד שרת. בשביל צד לקוח השתמשתי באתר של angular
- לצורך עיצוב האפליקציה נעזרתי באתר של bootstrap כדי להשתמש בעיצובים שלהם.
- כדי לשמור את השינויים שעשיתי על הפרויקט וכדי שאוכל לעבוד עליו ממקומות שונים נעזרתי בGitHub כמו כן לקחתי משם קצת קודים.



## מטרות ויעדים

מאחורי הרעיון של הפרויקט עומדות מספר מטרות: ראשית כמו שהזכרתי קודם המטרה המרכזית שלי היא לפתח אפליקציה שתועיל לחיי היום ויום ותשפר את איכות החיים של כל אחד מאיתנו.

מטרה נוספת היתה להרחיב אופקים ולהרוויח ידע נוסף בתחום התכנות וההייטק תוך כדי צבירת ניסיון על פרויקטים ועבודה מעשית, במהלך הפרויקט צברתי המון ידע ומיומנויות שאי אפשר להכיר בלימודים רגילים אלא רק על ידי התנסות בפועל. הרגשתי שהפרויקט נתן לי התנסות מקצועית שתעזור לי בהמשך.

הפרויקט שלי בעצם יפשט את כל שירותי המשלוחים. אם עד היום אדם שרצה לשלוח איזה חבילה למקום כלשהו ברחבי הארץ-ולא הייתה לו אפשרות כזו עכשיו תהיה לו אפשרות לעשות זאת בצורה הכי קלה ונוחה ע"י הרשמות בסיסית לאפליקציה ושם הוא יוכל לבקש לשלוח חבילה או להירשם כמשלוחן. כדי שהאפליקציה תהיה נגישה לכל שכבות האוכלוסייה יצרתי עיצוב נח, פשוט וברור כדי להקל כמה שיותר על המשתמש ולאפשר לכל אחד להשתמש בשירות האפליקציה. השימוש באפליקציה הוא חינמי ונגיש.

בבניית המערכת וידאתי שתהיה נוחה, פשוטה, וקלה לשימוש. בנוסף בניתי אותה כך שתהיה ברורה ומובנית למתכנת כך כשירצה להמשיך בפיתוח התכנה יוכל בקלות לעשות זאת. כמובן חשוב שתהיה יעילה ותעבוד כראוי.

היעד שאליו אני שואפת להגיע הוא להטמיע את האפליקציה בחיי היום יום שלנו.

## אתגרים

במהלך הפרויקט נתקלתי באתגרים. אפרט כמה נקודות:

- בהתממשקות עם גוגל מפות הייתי צריכה להשקיע מחשבה, ראשית נדרשתי להבין איך להתממשק עם גוגל מפות, איך להטמיע את API בפרויקט שלי בכפוף לנהלי ההתחייבות של גוגל מפות, הייתי צריכה לפתוח חשבון וליצור מפתח לשימוש google maps platform . והחלק הכי קשה בשלב הזה היה למצוא את הפונקציה של חישוב מרחקים ולהתאים אותו לפרויקט הספציפי שלי.
  - בחלק של צד הלקוח נתקלתי בקשיים כשרציתי לחבר בין הפונקציות של הצד לקוח לצד שרת, וכן כשנדרשתי לעדכן נתונים שהוכנסו על ידי הלקוח אל מסד הנתונים, גם בחלק של הניתובים בתוך האפליקציה נתקלתי באתגרים אבל לאחר התייעצות וקריאת חומרים בנושא צלחתי אותם.
  - בכל מהלך הפרויקט החל מהיגוי הרעיון דרך הצעת הפרויקט, העבודה עצמה, וכלה בהגשה - נדרשתי להתייעץ, להשקיע מחשבה לקרוא וללמוד על חומרים מסוימים ואפילו לשנות כיוון לפעמים
  - אחד החלקים הכי מורכבים היה להתחשב בכל המקרי קצה ולקחת בחשבון את 2 הצדדים, כלומר לחשוב על אותה פעולה גם מצד המשלוחן וגם מצד הלקוח לדוגמא: פונקציית ההתאמה שמופעלת ברגע שנכנסת נסיעה אפשרית חדשה למערכת -שונה מפונקציית ההתאמה שמופעלת ברגע שנכנסת בקשה למשלוח חדשה למערכת.
  - עוד מקום שבו נתקלתי באתגר היה בחלק של חישוב העדיפות. התלבטתי באיזה שיקולים להתחשב, באיזה סדר, באיזה תדירות ועוד. בתחילה חשבתי להתחשב קודם כל בחישובי רייטינג ואחר כך התאמה, לאחר מכן מכן הבנתי שיותר יעיל יהיה לחפש קודם כל התאמה, אחר כך לחפש מי המשלוחן שקיבל הכי פחות משלוחים ורק לבסוף-אם ייווצר מצב שלאחר כל הסינונים יישארו כמה באותו מצב -נחשב שקלולי רייטינג.
- גם בשלב שלאחר האפיון נתקלתי בקשיים רבים- בעצם מימוש האלגוריתם, האם עדיף להשתמש בתור, אולי עדיף ברשימה -היות ונדרשתי לכתוב הכל לבד ולא יכולתי להתבסס על משהו קיים.

## מדדי הצלחה

הפרויקט שלי יצליח אם אכן האפליקציה תצליח לקשר בין משולחנים ולקוחות בהתחשב בכל הנתונים:

האפליקציה תצליח למצוא התאמה בסיסית - מיקום מדויק וחשוב מרחקים-כולל בתוכו התממשקות עם גוגל מפות (החלק היותר מורכב) וכן התאמה של תאריך ושעה

האפליקציה תבחר מבין כל המועמדים את המועמד שהכי מגיע לו בהתחשב בנתונים קודמים וכן ברייטינג של המועמד.

עוד חלק שהוא מדד הצלחה בפרויקט שלי זה הצד לקוח:

צד לקוח שנוח וקליל למשתמש כולל בתוכו :

הרשמה והתחברות(כולל הפקת סיסמאות ובדיקות תקינות)

כמו כן יעמדו בפני המשתמש אפשרויות מגוונות : אודות החברה, איך משתמשים באפליקציה, פרופיל לקוח - כולל שליחת משלוח, תגובה על משלוח ועוד. פרופיל משלוחן- למשתמש יהיה אפשרות לצפות בהיסטורית המשלוחים שלו, ברייטינג שלו, וכן לרשום נסיעה אפשרית.

החלק המורכב בצד לקוח זה בעצם לחבר בין כל הפונקציות בצד לקוח לצד שרת.

## תיאור המצב הקיים

כיום המצב הוא שאין שירות משלוחים קליל ונוח לכל אחד מכל מקום לכל מקום, בכל נקודת זמן

לאחר בירור אצל אנשים שונים הגעתי למסקנה שזה שירות נצרך, שירות שיועיל מאד ויקדם את חיי היום יום של כל אחד מאיתנו.

אחרי חיפוש של תוכנות המספקות את השירות הנ"ל גיליתי שיש תוכנות שמספקות שירות דומה אבל מאד מוגבל כגון חברת wolt שמספקת שירות משלוחים אבל רק בתחום המזון ורק למסעדות הספציפיות שרשומות אצלם במאגר.

חישוב מרחקים ושימוש בגוגל מפות או בבניג מפות קיים ונפוץ מאוד, כמעט בכל אפליקציה, אבל מציאת המרחק הוא רק השלב הראשון בסדרת חישובים שביצעתי, בלי להסתמך על שום קוד כתוב אלא חישוב מותאם אישית לצורך הפרויקט הספציפי הזה.

## רקע תיאורטי

אפרט את האלגוריתם העיקרי בפרויקט.

החלק הכי מורכב בפרויקט זה החלק של החישוב וההתאמה בין המשלוחן ללקוח (לאחר כל החישובים)

כמו כן עוד חלק מרכזי בפרויקט זה החלק של התממשקות לגוגל מפות -להטמיע את API של גוגל מפות אצלי בפרויקט להפעיל את הפונקציות הנדרשות ולהתאים אותם ספציפית לפרויקט שלי . בעצם הבעיה שהאלגוריתם שלי פותר זה הבעיה של חוסר שירות כזה כיום. הפרויקט שלי אחראי להתאים בין משלוחן ללקוח, לבדוק שאכן הנתונים מתאימים זה לזה ולדאוג ששתי הצדדים יצאו מורווחים-גם הלקוח המערכת תדאג להתאים לו משלוחן על מנת שלא יצטרך להמתין זמן רב למענה וכל זה כמובן במחיר סימלי, וכן גם למשלוחן -המערכת תציג לו רשימה של משלוחים שמתאימים לדרישותיו והוא יאשר את מה שרוצה לבצע.

במידה ויש משלוחים שלא אושרו, המערכת תבצע חיפוש נוסף של משלוחים שמתאימים לדרישות ותציע להם את השירות -כך בעצם המערכת מכסה את כל האפשרויות ודואגת לתת מענה לכל הצדדים.

לא היה לי שום אלגוריתם שיכולתי להתבסס עליו כדי לפתח את האפליקציה ולכן כתבתי את כל האלגוריתם לבד -חוץ מפונקציות חישוב המרחקים.

## ניתוח חלופות מערכתי

אפשרויות שונות לפתרון הבעיה:

1. שימוש ב bing מפות -בחרתי להשתמש בגוגל מפות מפני שאני מכירה את המוצר הזה, איך זה עובד, והיה לי יותר קל להתממשק עם גוגל מפות.
2. אלגוריתם מבוסס התאמה מוחלטת ורייטינג- בתחילה תכננתי לחפש התאמה מוחלטת בלבד - במידה ומצאתי, תכננתי להשתמש בשקלולי רייטינג וככל שהרייטינג של המשלוחן גבוה יותר כך יגברו סיכוייו לקבל את המשלוח. כך נוצר מצב שאם לא הייתה התאמה מוחלטת, לא היה ללקוח מענה.
- על פי אלגוריתם זה -פונקציית בדיקת ההתאמה הייתה מופעלת רק כאשר הוכנסה למערכת בקשה חדשה למשלוח, ולא כאשר הוכנסה נסיעה אפשרית-כך נוצר מצב שהמשלוחן לא היה מקבל מענה מיד.
3. אלגוריתם מבוסס התאמה מוחלטת, התאמה בקירוב, נתוני רייטינג, מספר משלוחים מינימלי- אלגוריתם שנותן מענה לשתי הצדדים ומכסה את כל מקרי הקצה

## תיאור החלופה הנבחרת והנימוקים לבחירה

הרעיון שפיתחתי הוא פיתוח טכנולוגי שמחבר בין לקוח למשלוחן בהתחשב בהרבה נתונים וחשובים, האלגוריתם נותן מענה גם ללקוח וגם למשלוחן ומכסה את כל מקי הקצה הרעיונות האחרים שהצעתי לגבי המערכת של לא היו רלוונטים מפני שהם מאד מגבילים ולא נותנים מענה לכל המשתמשים כמו כן הם לא מכסים מקי קצה ולא מתחשבים בכל הנתונים, לכן בחרתי באפשרות טכנולוגית חכמה ונחה.

אז איך האלגוריתם עובד בעצם?

בצד הלקוח- המשתמש יכנס לאפליקציה ויכל לעיין בה, לקרוא אודות האפליקציה, איך היא עובדת ועוד. כאשר הוא ירצה לעשות [פעולה כלשהי בפרופיל משלוחן או לקוח, הוא יהיה חייב להתחבר. אם הוא משתמש ישן הוא יזין את שם המשתמש שלו ואת הסיסמא, ואם הוא משתמש חדש הוא יכניס פרטיים בסיסים והמערכת תפיק לו סיסמא לשימוש באתר. המטרה בחיבור ע"י סיסמא למערכת זה על מנת שהמערכת תשמור נתונים חיוניים על המשתמש. לאחר ההתחברות תעמוד בפני המשתמש 2 אפשרויות או משלוחן או לקוח. בפרופיל משלוחן הוא יוכל להכניס נסיעה אפשרית, לצפות בהיסטוריית המשלוחים שלו, וכן את נתוני הרייטינג שלו.

בפרופיל לקוח יעמדו בפניו האפשרויות להכניס בקשה למשלוח או להגיב על משלוח.

בצד השרת- השרת יקבל את הנתונים מהצד לקוח הנתונים ישמרו במסד הנתונים אם נבחר פרופיל הלקוח עי המשתמש והוא הכניס בקשה למשלוח המערכת תעבור על כל רשימת הנסיעות האפשריות שלא נעשו ותחפש התאמה מוחלטת-כתובת מקור=לכתובת מקור, כתובת יעד=לכתובת יעד, התאריכים, וכו.

לאחר הסינון תתקבל רשימה של משלוחנים מתאימים ובשלב הזה המערכת תעשה סינון נוסף הפעם בהתחשב במדד המשלוחים שכבר ביצע -המערכת תבחר את המינימלי מבין כל הרשימה, במידה ויהיו כמה מינימלים המערכת תבחר לפי דרוג רייטינג-למי שיש דירוג רייטינג הכי גבוה מבין כל המינימלים הוא יקבל את המשלוח לביצוע.

במידה ולא נמצאה התאמה מוחלטת המערכת תפעיל את פונקציית ההתאמה בקירוב-הפונקציה תבחר את כל מי שהכי קרוב לנתונים שהוכנסו.

לאחר הסינון תתקבל רשימה של משלוחנים מתאימים ובשלב הזה המערכת תעשה סינון נוסף הפעם בהתחשב במדד המשלוחים שכבר ביצע -המערכת תבחר את המינימלי מבין כל הרשימה, במידה ויהיו כמה מינימלים המערכת תבחר לפי דרוג רייטינג-למי שיש דירוג רייטינג הכי גבוה מבין כל המינימלים הוא יקבל את המשלוח לביצוע.

המערכת תציג למשלוחן את פרטי המשלוח והוא יאשר. במידה ולא אישר המערכת תחפש משלוחן אחר. לפני שהמערכת תפעיל שוב את החיפוש היא תעדכן במסד הנתונים את המשלוחן שסרב לבקשה, על מנת שלא יופיע שוב בסינון החוזר.

לאחר פונקציה זו ייוצר מצב של התאמה בן משלוחן ללקוח (יכול להיות שהמשלוחן עדיין לא אישר).  
אם בחר באפשרות של תגובה על משלוח המערכת תציג לו את רשימת המשלוחים האחרונים שנעשו לו והוא יוכל להגיב על השירות(אם כבר הגיב על משלוח מסוים אפשרות זו תחסם).  
אם נבחר פרופיל משלוחן, המערכת תפעיל את פונקציית ההתאמה של המשלוחן - המערכת תבדוק את כל נתוני הנסיעה החדשה למול כל הבקשות המתאימות שהותאמו ולא אושרו עדיין. לאחר כל הסינונים והחיפושים תמצא נסיעה מתאימה והמערכת תחשב את הניקוד של הנסיעה החדשה למול הניקוד של הנסיעה הישנה, במידה והניקוד של הנסיעה החדשה יהיה יותר גבוהה - הוא יקבל את הנסיעה.

הניקוד משכלל את נתוני התאריך, השעה, הכתובות והרייטינג.

לדוגמא: לקוח מכניס בקשה למשלוח בתאריך 03.05 מתל אביב לחיפה.

שובץ לו משלוחן מספר 111 שנוסע בתאריך 08.05. (ע"י שיבוץ מספר 1)

המשלוחן עדיין לא אישר את הבקשה.

משלוחן מספר 333 מוסיף נסיעה חדשה מתל אביב לחיפה בתאריך 04.05.

הבקשה משובצת לנסיעה החדשה (ע"י שיבוץ מספר 2)

כשמשלוחן יבחר באפשרות צפה בהיסטוריית המשלוחים שלי, או צפה בנתוני הרייטינג, המערכת תציג לו רשימה של משלוחים שעשה וכן את דירוג הרייטינג שלו.

## אפיון המערכת

סביבת פיתוח :

חומרה-מעבד: i7-97 Intel(R) core(TM)

עמדת פיתוח:

מחשב : DELL Optilex 7070

מערכת ההפעלה: Windows 10

שפות תוכנה: #C , , תוך שימוש בטכנולוגיית WebApi , אנגולר .

כלי תוכנה לפיתוח המערכת: Microsoft Visual Studio2019, vs code

מסד נתונים: SQLserver

עמדת משתמש מינימלית:

-חומרה: מעבד i3 RAM 4GB

-מערכת ההפעלה: windows : 10 ומעלה

-חיבור לרשת: נדרש

-תוכנות: chrome

## ניתוח דרישות המערכת

### דרישות בהן המערכת צריכה לעמוד:

- כתיבה בסטנדרטים מקצועיים.
- מחשוב השרות ללקוח.
- כתיבת הקוד בסיבוכיות היעילה ביותר.
- ממשק נוח וידידותי למשתמש.
- תגובה מהירה ככל שניתן למשתמש.

## מודול המערכת

- התחברות למערכת
- הכנסת הנתונים ע"י המשתמש ושליחתם לשרת.
- הכנסת הנתונים ל data base.
- הפעלת האלגוריתם- הפעלת הפונקציה המתאימה בהתאם לבחירת המשתמש.
- מציאת התאמה בין משלוחן ללקוח.
- הזנת פרטי הנסיעה והמשלוח ב data base.
- חיבור בין המשלוחן ללקוח על ידי הצגת מספרי הטלפון של האחד לשני.



## אפיון פונקציונאלי

### פירוט פונקציות עיקריות ותפקידן:

`Absolutefit(List<dtoPOSSIBLEDRIVE> request, dtoDELIVERY p)` - הפונקציה מקבלת רשימה של נסיעות ובקשה ומחזירה רשימה של משלוחים שיש להם התאמה מוחלטת עם הנתונים של הנסיעה

`Partialfit(List<dtoPOSSIBLEDRIVE> AllOpenRequest, dtoDELIVERY p)` - במקרה שלא נמצא אף משלוח עם התאמה מוחלטת נפעיל את פונקציה זו שמחפשת התאמה חלקית ומביאה את הכי קרובים לנתונים של הבקשה

`Minimumnumber(List<dtoPOSSIBLEDRIVE> Partialfitlist)` - במידה ולאחר ההתאמה המוחלטת או החלקית נשיג כמה משלוחים באותה רמה - נבצע בדיקת הסתברות חיפוש כל הנסיעות עם מספר משלוחים מינימלי

`CalculatePoint(long tz)` - הפונקציה מקבלת משתמש מסוים ומחזירה את חישוב הרייטינג שלו בנקודות

`CheckMatchBetweenDeliverAndCustomer()` - קלט: משלוח ובקשה פלט: ניקוד רמת התאמה

`CheckRatingBetweenDeliverAndCustomer()` - קלט: משלוח ובקשה פלט: ניקוד רמת הסתברות.

`MainDelivery.main(DELIVERIES d)` - הפונקציה הראשית שמופעלת ברגע שנכנסת בקשה חדשה למערכת, הפונקציה מפעילה תתי פונקציות בהתאם לערך שהוזר מהפונקציות הקודמות. בסוף הפונקציה הזו (או במהלכה) ישובץ משלוח מתאים ללקוח והוא יצטרך לאשר את הנסיעה.

`checkpoint(dtoPOSSIBLEDRIVE p, dtoDELIVERY d)` - הפונקציה בודקת את סך הנקודות של הנסיעה החדשה אל מול הנסיעה הישנה אם הניקוד של החדשה קטן מהניקוד של הישנה אז הנסיעה החדשה תשובץ לבקשה

`MainpossibledriveBL.main(DELIVERIES d)` - הפונקציה מופעלת ברגע שנכנסת נסיעה חדשה למערכת- הפונקציה תפעיל את תתי הפונקציות הנדרשות בהתאם לערך המוזר.

### ביצועים עיקריים

כל משתמש יכול להירשם לאתר ע"י הכנסת פרטים אישיים.  
לאחר ההתחברות הוא יכניס לפעולה שרוצה לבצע פרטים שידרשו ממנו.  
לאחר שהמערכת תחפש לו את השרות שביקש היא תקשר אותו לצד השני באמצעות הצגת מספר  
טלפון של הצד השני.

### אילוצים

המערכת תיתן ללקוח מענה רק אם יהא משלוחן מתאים וכן להפך.  
המערכת מחייבת התחברות למערכת עי סיסמא על מנת להשתמש בה.  
המערכת פועלת ע"פ המקומות של גוגל מפות והמשתמש יהיה חייב לבחור כתובת מתוך המאגר.

## תיאור הארכיטקטורה

### הארכיטקטורה של הפתרון המוצע בפורמט של Design level Down-Top

צד השרת - server side פותח במודל 3 השכבות ומתחלק ל-4 פרויקטים

החלוקה לשכבות נועדה להפריד באופן מוחלט בין הלוגיקה של הפרויקט לבין הנתונים עצמם. הפרדה זו מאפשרת לבצע שינויים בכל אחת מהשכבות בלי תלות ובלי זעזועים בשכבות האחרות.

API – שכבת ה Controller – חיבור בין צד השרת והלקוח.

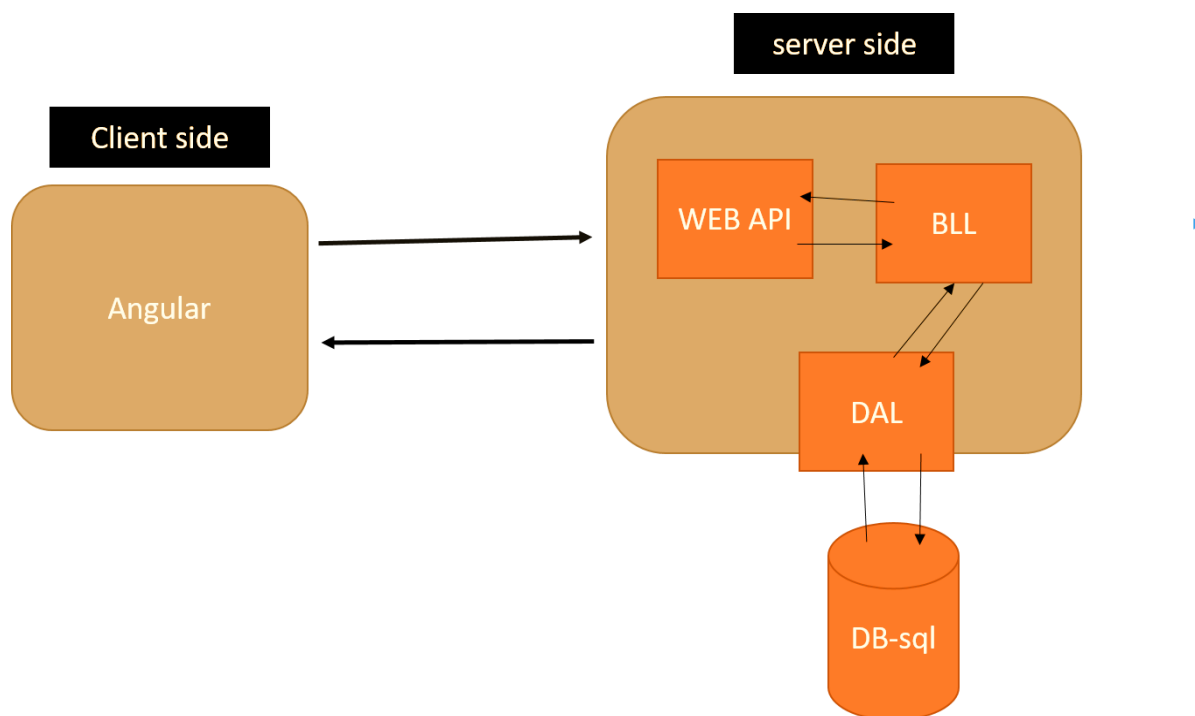
BL – הלוגיקה של המערכת.

DAL – מכיל את הפונקציונאליות הנדרשת לכל התקשורת עם Data Base.

Models – מכילה מחלקות המתארות את הנתונים ובמבנה זה מעבירים את הנתונים בין השכבות.

מטרת שכבה זו היא למנוע תלות של שכבת ה BL במבנה בסיס הנתונים. שכבת ה BL מכילה פונקציות המרה מטיפוס הנתונים של בסיס הנתונים לטיפוס הנתונים של שכבת ה Models ולהיפך, וכך מיוצגים הנתונים בכל הפרויקט.

### תיאור הרכיבים בפתרון



## הפרויקט מחולק ל-2 חלקים:

- צד שרת - הנכתב בשפת C# ובטכנולוגיית WebApi.
  - צד לקוח - נכתב בשפת Angular ובטכנולוגיית TypeScript, Html.
- בחרתי לכתוב צד לקוח ב - אנגולר שהינה טכנולוגיה מתקדמת ועדכנית בעלת מאפייני Angular 8 חדשניים ופונקציונאלית ביותר.
- אנגולר הינה סביבת עבודה שפותחה על ידי גוגל. מאפשרת לפתח אפליקציות Framework אינטרנט בקלות ומהירות. במקור היא באה לתת מענה לבניית Applications Page Single בצורה מושלמת ומהירה. מהיתרונות הבולטים והעיקריים של אנגולר אפשר למנות: חיסכון במשאבים, מהירות ביצוע, קוד קצר יותר, רוב העבודה מתבצעת בצד הלקוח ופחות בשרת ויכולת התמודדות טובה של תוכן המתקבל מהשרת לפי מספר רב של פרמטרים.
- צד שרת בחרתי לכתוב ב-C#. #C# היא שפת תכנות עילית מרובת-פרדיגמות, מונחית עצמים בעיקרה המשלבת רעיונות כמו טיפוסיות חזקה, אימפרטיביות, הצהרתיות, פונקציונאליות פרוצדוראליות וגנריות.
- #C היא שפה מעניינת, נוחה ומלאה פונקציונאליות למתכנת. שימוש בשפה זו נפוץ כיום, וכתוצאה מכך, ניתן היה למצוא בה קודים שונים שנדרשו לפיתוח.
- ל-SQL Server יש כלים נרחבים לגיבוי כל המידע של המערכת, כולל מערכת ההפעלה, חשבונות המשתמשים והרשאותיהם, הגדרות ההתקנים, תוכניות וכן של שאר הרכיבים המסופקים עם השרת ואובייקטי המשתמש.

## דוגמא לזרימת מידע במערכת

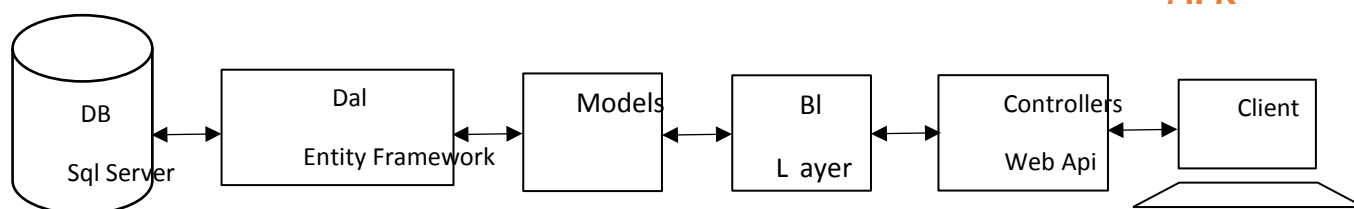
- שליפת היסטוריית המשלוחים של משתמש
- ברצוננו לקבל את כל המשלוחים של משתמש מסוים מ DB ולכן יתבצעו השלבים הנ"ל:
- המשתמש יחפוץ לראות את כל המשלוחים שעשה, הוא ילחץ על כפתור היסטוריית המשלוחים שלי בתצוגה html (ובקשתו תפנה לTypeScript).
  - תתבצע קריאה לפונקציה ViewHistory ב-TypeScript אשר תפנה לשרת url ותתבצע בקשת services.
  - השרת מקבל את הבקשה ומנווט ל Controller שנמצא ב-API.
  - ה Controller יזמן את הפונקציה ViewHistory() שנמצאת ב - PossibleDriveBL

o ה BL מעוניין לקבל נתונים מהDB ולכן הוא פונה ל-DAL- דרך ה

Entityframework

- o ה-DAL שואב את הנתונים הרצויים ממסד הנתונים וכעת מתבצע שלב החזרה.
- o ה DAL מחזיר את רשימת המשלוחים לשכבת הBL בה מתבצעת פונקצית הסינון של הבאת המשלוחים של משתמש מסוים.
- o הפונקצייה ViewHistory מחזירה את הנתונים מה- controller ל- BL.
- o הנתונים מוחזרים ל controller- מה service.
- o מהservice חוזרת הרשימה לtypeScript.
- o הרשימה מוצגת בHTML.

## איור:



1. מסד הנתונים הבנוי מטבלאות וקשרי גומלין ביניהם.
2. שכבת הגישה לנתונים באמצעות Entity Framework.
3. שכבת הישויות.
4. שכבת ה- BL בה כתובים האלגוריתמים.
5. Web Api פרוטוקול התקשורת בינן צד הלקוח וצד השרת.
6. angular, TypeScript. צד לקוח

## ארכיטקטורת רשת

לא קיים.

## תיאור פרוטוקולי התקשורת

### שרת – לקוח

HTTP-ראשי תיבות **Hypertext Transfer Protocol** הוא פרוטוקול תקשורת שנועד להעברת דפי html ואובייקטים שהם מכילים (כמו תמונות, קובצי קול, סרטוני פלאש וכו') ברשת האינטרנט.

בס"ד

הפרוטוקול פועל בשכבת הישום של מודל ה osi ובשכבת הישום של מודל tcp שרתי http הם שרתי התוכן המרכזיים ברשת האינטרנט ודפדפנים הם תוכנות הלקוח הנפוצות ביותר לפרוטוקול HTTP.

צד השרת נכתב בטכנולוגיית WebApi ובשפת #c.

בטכנולוגיית -typescript, css, Html צד הלקוח נכתב בשפות . angular

## ניתוח ותרשים use case של המערכת המוצעת



### רשימת use case

התחברות-המשתמש יתחבר למערכת באמצעות שם משתמש וסיסמא

הרשמה-אם המשתמש לא רשום במערכת, הוא יזין פרטים אישיים שישמרו במערכת

תגובה על משלוח-הלקוח יכול להגיב על שירות של משלוח שנעשה לו

הכנסת נסיעה-משלוחן יכול להכניס נסיעה אפשרות שלו

אישור משלוח-המשלוחן צריך לאשר משלוח שרוצה לעשות

צפייה בנתוני רייטינג-המשלוחן יכול לצפות בנתוני הרייטינג שלו

צפייה בתוצאות-המשלוחן או הלקוח ילחצו על כפתור זה על מנת לצפות בתוצאות חישוב המערכת

צפייה בהיסטורית משלוחים- המשלוחן יצפה בהיסטורית המשלוחים שלו

בקשה למשלוח-הלקוח יכניס פרטים בסיסיים אודות משלוח שרוצה לשלוח

## תיאור ה-use case העיקריים של המערכת

### Uc1

Indetifier :Uc1

Name: בקשה למשלוח

Description: המשתמש מזין פרטים בסיסיים אודות משלוח שרוצה שיעשו לו, הפרטים כוללים: כתובות מקור ויעד, תאריך ושעה.

Actor: לקוח

Frequency: בעת הכנסת משלוח למערכת

Extend use case: התחברות למערכת

Condition-pre: פרטים בסיסיים

Condition-post: פרטי הבקשה נשמרים במערכת, והמערכת מחפשת משלוח מתאים.

Basic course of action: הלקוח צריך להתחבר למערכת קודם לכן.

Alternet course of action: אם לא התחבר למערכת אין לו אפשרות לשלוח בקשה.

### Uc2

Indetifier :Uc2

Name: הכנסת נסיעה

Description: המשתמש מזין פרטים בסיסיים אודות נסיעה עתידית שלו, הפרטים כוללים: כתובות מקור ויעד, תאריך ושעה.

Actor: משלוחן

Frequency: בעת הכנסת נסיעה אפשרית למערכת

Extend use case: התחברות למערכת

Condition-pre: פרטים בסיסיים

Condition-post: פרטי הבקשה נשמרים במערכת, והמערכת מחפשת לקוח מתאים.

Basic course of action: המשלוחן צריך להתחבר למערכת קודם לכן.



בס"ד

Alternet course of action: אם לא התחבר למערכת אין לו אפשרות להכניס נסיעה .

### Uc3

Indetifier :Uc3

Name: צפייה בתוצאות

Description: המשתמש ילחץ על כפתור לתוצאות -כדי לראות את תוצאות חישוב המערכת

Actor: משלוחן או לקוח

Frequency: לאחר הכנסת נסיעה אפשרית למערכת או לאחר הכנסת בקשה למערכת

Extend use case: התחברות למערכת, הכנסת נסיעה או בקשה

Condition-pre: הכנסת נסיעה או בקשה

Condition-post: פרטי הבקשה או הנסיעה נשמרים במערכת, והמערכת מחפשת משתמש מתאים.

Basic course of action: המשלוחן או הלקוח צריכים להתחבר למערכת קודם לכן.

Alternet course of action: אם לא התחבר למערכת אין לו אפשרות להכניס נסיעה או לשלוח בקשה .

### Uc4

Indetifier :Uc4

Name: תגובה על משלוח

Description: הלקוח יגיב על משלוח שנעשה לו

Actor: לקוח

Frequency: לאחר קיום משלוח

Extend use case: התחברות למערכת

Condition-pre: משלוח שנעשה ללקוח

Condition-post: פרטי הנסיעה נשמרים במערכת והמערכת תתן ללקוח אפשרות להגיב על שירות שנעשה לו

Basic course of action: הלקוח צריך להתחבר למערכת קודם לכן.

Alternet course of action: אם לא התחבר למערכת אין לו אפשרות להגיב.

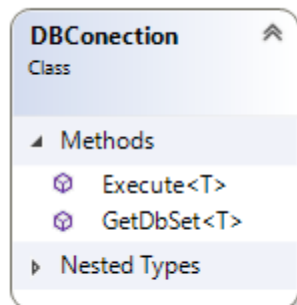
### מבני נתונים בהם משתמשים בפרויקט

List - הרשימה מאפשרת שמירה של נתונים בצורה סדרתית בלי חובה להגדיר את הגודל, על כן השתמשתי ברשימה כדי לשלוף רשימה של ישויות שמתאימות לתנאים מסוימים בלי להגדיר את הגודל מראש לדוגמא: רשימה של משלוחים שמתאימים לתנאים של נסיעה מסוימת.

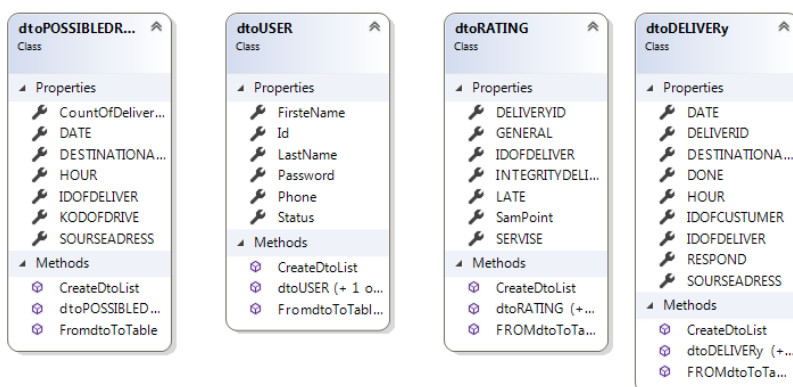
Queue - כדי לשמור על סדר מסוים כמו: לסדר את הבקשות לפי זמן כניסתם למערכת בתור.

## תרשים מחלקות

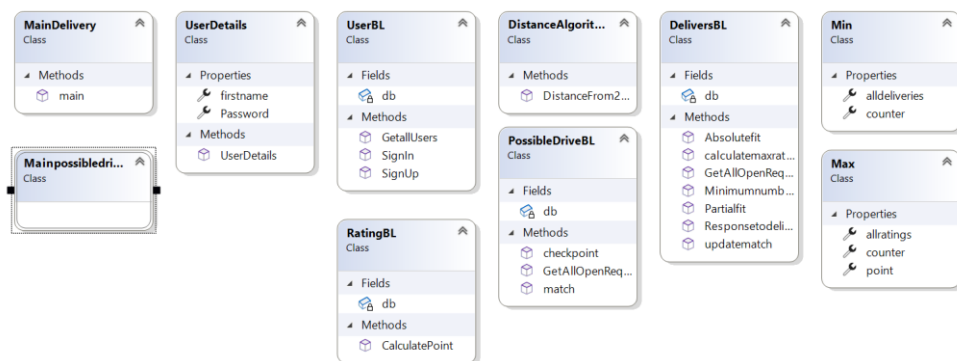
### DAL-ה שכבת ה-



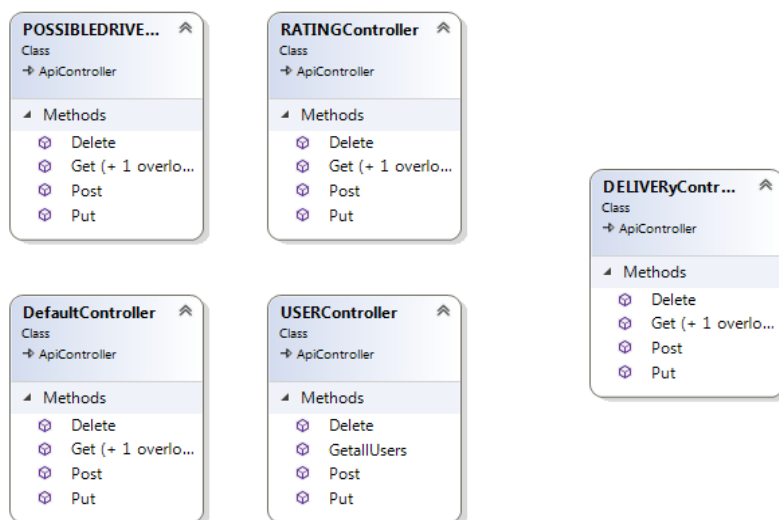
### DTO - שכבת ה-



## שכבת ה-BLL



## שכבת ה-API



## תיאור המחלקות

הצד שרת מחולק ל- 3 שכבות. כל שכבה אחראית על תחום מסוים בפרויקט.

### DAL

השכבה הנמוכה ביותר. שכבה זו אחראית על התקשורת עם ה-Data Base. בשכבה זו פונקציות שונות המפעילות את ההתקשרות.

המחלקות הקיימות בשכבה זו הן:

**DBConnection**-מחלקה זו משמשת להתקשרות בפועל ל- Data Base. המחלקה מכילה את הניתוב ל- Data Base, שם ה- Data Base, ושם ה- Collection.

### BLL

השכבה שמעל ה- DAL היא מקשרת בין ה- DAL ל- API. השכבה אחראית על כל החלק הלוגי של המערכת.

המחלקות הקיימות בשכבה זו הן:

#### -CalculateDistanceBL

דרך מחלקה זו התממשקתי לגוגל מפות. במחלקה יש פונקציה של חישוב מרחקים -הפונקציה בעצם קוראות לפונקציות חיצוניות של גוגל מפות.

`DistanceFrom2PointsInMinutes(string point1, string point2)*`

#### -DeliversBL

מחלקה זו אחראית בעצם על כל מה שקשור לצד של הבקשות למשלוחים(פרופיל לקוח) הפונקציות במחלקה זו הן:

`public static List<dtoPOSSIBLEDRIVE> GetAllOpenRequest(dtoDELIVERY p)*`

פונקציה שמקבלת בקשה מסוג תצוגה ומסננת עבורה את כל המשלוחים שלא סרבו לבקשה זו.

`public static List<dtoPOSSIBLEDRIVE> AbsoluteFit(List<dtoPOSSIBLEDRIVE> request,dtoDELIVERY p)*`

מקבלת רשימה של נסיעות אפשריות מהסינון הקודם ומקבלת בקשה -הפונקצייה מחזירה רשימה של נסיעות שמתאימות בהתאמה מוחלטת לנתוני הנסיעה

`public static List<dtoPOSSIBLEDRIVE> Partialfit(List<dtoPOSSIBLEDRIVE> AllOpenRequest, dtoDELIVERY p)*`

מקבלת רשימת נסיעות ובקשה ומחזירה רשימה של התאמה חלקית

`public static List<dtoPOSSIBLEDRIVE> Minimumnumber(List<dtoPOSSIBLEDRIVE> Partialfitlist)*`

מחפשת את המשלוחן שמספר המשלוחים שקיבל עד היום - הוא מינימלי ומחזירה רשימה של מינימלים

`public static dtoPOSSIBLEDRIVE calculatemaxrating(List<dtoPOSSIBLEDRIVE> mincounter)*`

במידה ולאחר הסינון הקודם יחזור יותר מנסיעה אחת אז יבדקו שיקלולי רייטינג, פונקצייה זו מקבלת רשימת נסיעות ומחזירה משלוחן שנתוני הרייטינג שלו הוא הגבוה ביותר

`public static void updatematch(dtoDELIVERY p, dtoPOSSIBLEDRIVE match)*`

הפונקציה מעדכנת במסד הנתונים את המשלוחן המתאים לנסיעה

`()*Responsetodelivery-Aפשרות תגובה על משלוח`

- Max

מחלקה שנעזר בה לצורך חישובי רייטינג

- Min

מחלקה שנעזר בה לצורך חישובי רייטינג

-PossibleDriveBL

מחלקה זו אחראית בעצם על כל מה שקשור לצד של המשלוחנים-הנסיעות האפשריות, הפונקציות במחלקה זו הן:

`public static List<dtoDELIVERY> GetAllOpenReqwest(dtoPOSSIBLEDRIVE p)*`

-מסננת את כל הבקשות שעוד לא אושרו (עבור נסיעה חדשה שהוכנסה)

`public static void match(List<dtoDELIVERY> AllOpenRequest, dtoPOSSIBLEDRIVE p)*`

הפונקצייה בודקת את סך הנקודות של הנסיעה החדשה אל מול הנסיעה הישנה

`public static float checkpoint(dtoPOSSIBLEDRIVE p, dtoDELIVERY d)*`

פונקצייה שמחשבת ניקוד למשלוחן בהתאם לבקשה מסוימת

-Viewhistory

צפייה בנתוני המשלוחים הקודמים של משתמש מסוים

-Viewratingצפייה בנתוני רייטינג של משתמש

-RatingBL

במחלקה זו יתבצעו כל חישובי הרייטינג הפונקציות במחלקה זו הן:

בס"ד

```
public static long CalculatePoint(long tz)*-
```

פונקציה שמחשבת את סך הנקודות למשתמש מסוים

```
Responsetodelivery*-
```

פונקציה שמעדכנת את הנתונים שהלקוח הכניס כתגובה על משלוח מסוים- ל. data base.

-UserBL

מחלקה זו אחראית על כל מה שקשור לצד של המשתמש-התחברות,הרשמה,שמירת נתונים ועוד.הפונקציות במחלקה זו הם:

```
public static List<dtoUSER> GetAllUsers()*-
```

שליפת כל המשתמשים

```
public static Object SignUp(dtoUSER user)*-
```

שומרת נתונים חדשים אודות משתמש חדש במסד הנתונים

```
public static Object SignIn(UserDetails ud)*
```

בודקת את ההתחברות של המשתמש למערכת

-UserDetails

מחלקה שמסייעת בבדיקת התחברות משתמש למערכת

MainpossibledriveBL-

-פונקציה ראשית שמפעילה את כל מה שקשור לצד של המשלוחנים הפונקציות הקיימות במחלקה זו הם:

```
public static void main(DELIVERIES d)*
```

הפונקציה מקבלת בקשה ומפעילה את כל הפונקציות הנדרשות כדי למצוא לה משלוחן מתאים

MainDelivery-

-פונקציה ראשית שמפעילה את כל מה שקשור לצד של הלקוח והבקשות הפונקציות הקיימות במחלקה זו הם:

```
public static void main(POSSIBLEDRIVE d)*
```

הפונקציה מקבלת נסיעה ומפעילה את כל הפונקציות הנדרשות כדי להתאים לו נסיעה

**API**

שכבה זו אחראית על החיבור בין צד השרת והלקוח. בשכבה זו קיימים קבצי מערכת רבים, קבצי התקנות, סקריפטים וכו'. בנוסף בשכבה זו קיימים Controllers – בקרים האחראים על ניתוב התקשורת בין השרת והלקוח.

המחלקות בשכבה זו הם:

**DELIVERyController** במחלקה זו קיימות הפעולות: GET, POST, הרשמה, התחברות, הוספה, מחיקה ועוד.

**POSSIBLEDRIVEController** במחלקה זו קיימות הפעולות: GET, POST, הרשמה, התחברות, הוספה, מחיקה ועוד.

**RATINGController** במחלקה זו קיימות הפעולות: GET, POST, הרשמה, התחברות, הוספה, מחיקה ועוד.

**USERController** במחלקה זו קיימות הפעולות: GET, POST, הרשמה, התחברות, הוספה, מחיקה ועוד.

**:DTOclass**

ה data transfeere object הם אובייקטים הנושאים נתונים בין תהליכים על מנת לצמצם את מספר הקריאות לפונקציות. הם משמשות מחלקות תצוגה ומונעות כפילויות.

המחלקות בחלק זה הם:

**dtoDELIVERy**

מחלקת dto שאחראית על כל מה שקשור לבקשות למשלוח, הפונקציות במחלקה זו הן:

**public dtoDELIVERy(DELIVERIES d)\***

פעולה בונה שהופכת רשומה מסוג הטבלה לסוג תצוגה

**public DELIVERIES FROMdtoToTable(dtoDELIVERy d)\***

הופכת רשומה ממחלקת תצוגה לרשומה מסוג הטבלה

**public static List<dtoDELIVERy> CreateDtoList(List<DELIVERIES> LIST)\***

מקבלת רשימה מסוג הטבלה והופכת אותה לרשימה מסוג תצוגה



## dtoPOSSIBLEDRIVE

-מחלקת dto שאחראית על כל מה שקשור למשלוחנים (לנסיעות אפשריות), הפונקציות במחלקה זו הן:

```
public dtoPOSSIBLEDRIVE(POSSIBLEDRIVE p)*
```

-פעולה בונה שהופכת רשומה מסוג הטבלה לסוג תצוגה

```
public POSSIBLEDRIVE FromdtoToTable(dtoPOSSIBLEDRIVE p)*
```

-הופכת רשומה ממחלקת תצוגה לרשומה מסוג הטבלה

```
public static List<dtoPOSSIBLEDRIVE> CreateDtoList(List<POSSIBLEDRIVE> LIST)*
```

-מקבלת רשימה מסוג הטבלה והופכת אותה לרשימה מסוג תצוגה

## dtoRATING

-מחלקת dto שאחראית על כל מה שקשור לנתוני רייטינג של משתמש, הפונקציות במחלקה זו הן:

```
public dtoRATING(RATING r)*
```

-פעולה בונה שהופכת רשומה מסוג הטבלה לסוג תצוגה

```
public RATING FROMdtoToTable(dtoRATING u)*
```

-הופכת רשומה ממחלקת תצוגה לרשומה מסוג הטבלה

```
public static List<dtoRATING> CreateDtoList(List<RATING> LIST)*
```

-מקבלת רשימה מסוג הטבלה והופכת אותה לרשימה מסוג תצוגה

## dtoUSER

-מחלקת dto שאחראית על כל מה שקשור לנתוני משתמש, הפונקציות במחלקה זו הן:

```
public dtoUSER(USERS u)*
```

-פעולה בונה שהופכת רשומה מסוג הטבלה לסוג תצוגה

```
public USERS FromdtoToTable(dtoUSER u)*
```

-הופכת רשומה ממחלקת תצוגה לרשומה מסוג הטבלה

```
public static List<dtoUSER> CreateDtoList(List<USERS> LIST)*
```

-מקבלת רשימה מסוג הטבלה והופכת אותה לרשימה מסוג תצוגה

## תיאור התוכנה

### סביבת עבודה:

- Visual Studio Code
- Visual Studio

### שפות תכנות:

צד השרת נכתב בטכנולוגית WebApi ובשפת #c.

angular. בטכנולוגית typescript, css, Html- צד הלקוח נכתב בשפות

## אלגוריתמים מרכזיים

האפליקציה במשפט אחד:

שירות משלוחים שיתופי.

מה האלגוריתם המרכזי?

1. התאמה בין בקשה לנסיעה של משלוחן בעל התאמה מקסימלית.

לכל בקשה שנכנסת למערכת מתבצע שיבוץ למשלוחן המתאים עם הסיכויים המרביים לאשר את הבקשה

בשלב הראשון מתבצע סינון של המשלוחנים שכבר סירבו לבקשה ( אם קיימים)

בשלב השני מתבצע חיפוש המשלוחן המתאים ביותר:

לכל משלוחן מתבצעת בדיקת התאמה ובדיקת הסתברות לאישור המשלוח:

יש 2 סוגי בדיקות התאמה:

- התאמה מוחלטת:

עיר ורחוב מקור עיר ורחוב יעד תאריך שעה.

- התאמה חלקית:

חישוב מרחקים ע"פ google maps. מהמקור ומהיעד. חישוב טווח ביטחון של שעה

אם יש יותר ממשלוחן אחד שמתאימים באותה רמה, מתבצע חיפוש של המשלוחן עם הסיכויים הרבים ביותר להיענות לבקשה.

יש בדיקת הסתברות לאישור המשלוח:

- מספר משלוחים .

- רייטינג גבוה.

אם לאחר כל הסינונים עדיין קיימים מספר משלוחנים:

תתבצע הגרלה.

2. התאמה של הבקשות שעוד לא אושרו לנסיעה החדשה שנוספה.

בס"ד

מסננת בקשות שלא אושרו.

לכל בקשה שלא אושרה כבר יש משלוחן שקיבל אותה.

נבדוק מה הניקוד של ההתאמה לבקשה עם הנסיעה של המשלוחן הקודם.

נבדוק מה הניקוד של ההתאמה לבקשה עם הנסיעה של המשלוחן החדש.

מתבצעת בדיקת התאמה ובדיקת הסתברות לאישור המשלוח:

יש 2 סוגי בדיקות התאמה:

- התאמה מוחלטת:

עיר ורחוב מקור עיר ורחוב יעד תאריך שעה.

- התאמה חלקית:

חישוב מרחקים ע"פ googlemaps. מהמקור ומהיעד. חישוב טווח ביטחון של שעה

יש בדיקת הסתברות לאישור הבקשה:

- מספר משלוחים .

- רייטינג גבוה.

אם הניקוד גבוה יותר הבקשה תועבר למשלוחן החדש.

דוגמא:

לקוח מכניס בקשה למשלוח בתאריך 03.05 מתל אביב לחיפה.

שובץ לו משלוחן מספר 111 שנוסע בתאריך 08.05. (ע"י שיבוץ מספר 1)

המשלוחן עדיין לא אישר את הבקשה.

משלוחן מספר 333 מוסיף נסיעה חדשה מתל אביב לחיפה בתאריך 04.05.

הבקשה משובצת לנסיעה החדשה (ע"י שיבוץ מספר 2)

משלוחן מספר 555 מוסיף נסיעה חדשה מתל אביב לחיפה בתאריך 06.05

הבקשה אינה משובצת למשלוחן 555 אלא נשארת משויכת למשלוחן 333

(למרות שהתבצע ניסיון לשיבוץ מספר 2)

הלקוח –

- תפקידו להכניס את הבקשה.
- יכול לצפות בבקשות ולכלל בקשה - במשלוחן המתאים – אם המשלוחן אכן אישר את הבקשה.

המשלוחן –

- מכניס פרטי נסיעה.
- צופה ברשימת בקשות לא מאושרות.
- מאשר בקשה או מסרב לבקשה.
- צופה בבקשות מאושרות שלו ועוד לא התבצעו – משלוחים שהוא אמור לעשות.
- צופה בבקשות מאושרות שהתבצעו בעבר.
- צופה ברשימת משלוחים.
- צופה בפרטי לקוח שקשור למשלוח.

מתי נפעיל את השיבוע?

שיבוע מספר 1 – משלוחן לבקשה:

- כאשר לקוח מכניס בקשה.
- כאשר משלוחן מסרב לבקשה.

שיבוע מספר 2 – בקשות למשלוחן (לנסיעה):

פונקציה בדיקה:

CheckMatchBetweenDeliverAndCustomer()

קלט: משלוחן ובקשה

פלט: ניקוד רמת התאמה. //כאן יש גישה לגוגל מפיס בינתיים תתעלמי מזה תגירלי מרחק.

פונקציה הסתברות לאישור המשלוח:

CheckRatingBetweenDeliverAndCustomer()

קלט: משלוחן ובקשה

פלט: ניקוד רמת הסתברות.

פונקציית שיבוץ בקשה לנסיעה בעלת התאמה מקסימלית:

קלט: בקשה

פלט: אין פלט

הפונקציה עוברת בלולאה פעם ראשונה על המשלוחנים לכל משלוחן בודקת התאמה ושומרת את הניקוד המקסימלי.

עוברת בלולאה נוספת ושומרת ברשימה את של המשלוחנים בעלי הניקוד הגבוה ביותר.

אם יש רק משלוחן אחד ברשימה – מעדכנים בבקשה קוד משלוחן.

עוברת בלולאה פעם שניה ושומרת את ההסתברות הכי גבוהה.

עוברת בלולאה נוספת ושומרת ברשימה את כל המשלוחנים בעלי ההסתברות הגבוהה ביותר.

אם יש רק משלוחן אחד ברשימה – מעדכנים בבקשה קוד משלוחן.

מגרילה משלוחן מהרשימה - מעדכנים בבקשה קוד משלוחן.

פונקציית שיבוץ של הבקשות שעוד לא אושרו לנסיעה החדשה שנוספה.

קלט: נסיעה חדשה

פלט: אין פלט

הפונקציה מסננת לרשימה החדשה בקשות שלא אושרו.

הפונקציה יוצרת רשימה ריקה לבקשות שיתאימו לנסיעה החדשה.

הפונקציה עוברת בלולאה על הבקשות שלא אושרו ולכל בקשה:

שולפת פרטי נסיעה שמקושרת ולא אושרה.

בודקת התאמה והסתברות לנסיעה שלא אושרה.

בודקת התאמה והסתברות לנסיעה החדשה.

אם הניקוד של הנסיעה החדשה טוב יותר – נוסיף את הבקשה לרשימת בקשות לנסיעה.

עוברת על הרשימה של הבקשות המתאימות ומעדכנת לכל בקשה קוד של הנסיעה החדשה.

רשימת מונחים להבנה של ההקדמה:

-משלוחן-משתמש המכניס פרטים למערכת אודות נסיעה עתידית שלו(נסיעה שבה הוא רוצה לקחת משלוח)

-לקוח-משתמש המכניס פרטים אודות בקשה של משלוח שרוצה שיעשו לו.

-נסיעה-ביצוע המשלוח

-בקשה למשלוח-הלקוח מכניס בקשה לשליחת משלוח ממקום כלשהו למקום כלשהו

-בדיקת התאמה-המערכת בודקת אם הפרמטרים (כתובת מקור, כתובת יעד, תאריך, שעה)מתאימים בין 2 הצדדים(משלוחן ולקוח)

-בדיקת התאמה בקרוב- המערכת בודקת אם הפרטים הנ"ל מתאימים בקרוב (כתובות קרובות אחת לשנייה, תאריך קרוב)

-פרמטרים להתאמה: כתובת מקור, כתובת יעד, תאריך, שעה

-בדיקת הסתברות לאישור הבקשה-המערכת בודקת פרמטרים שונים לאחר ההתאמה האישית.

פרמטרים להסתברות: מספר משלוחים מינימלי-ככל שמספר המשלוחים יותר מינימלי כך הסיכויים של המשלוחן יגברו (במידה ויש מספר משלוחים עם ניקוד מינימלי אז יתחשבו בנתוני רייטינג)

-רייטינג-על כל נסיעה, יש ללקוח שנעשה לו המשלוח אפשרות להגיב על המשלוח, שירות המשלוחן, שלמות המשלוח ועוד, הנתונים שיכנסו ישתקללו לניקוד שישמר באזור האישי של המשתמש ככל שניקוד הרייטינג יותר גבוה יש למשלוחן יותר סיכוי לקבל משלוח

במידה וגם מהסינון הזה יצא יותר מאחד יתקיים ביניהם הגרלה.

## DistanceAlgorithm

```

public class DistanceAlgorithm
{
    0 references
    public static int DistanceFrom2PointsInMinutes(string point1, string point2)
    {
        string uri = "https://maps.googleapis.com/maps/api/distancematrix/xml?origins="
            + point1 + "&destinations=" + point2 + "&mode=driving&units=imperial&sensor=false&key=AIzaSyDN6hmLe4mB_RT43pToR3PtvXch2xeeBxQ";
        WebClient wc = new WebClient();
        try
        {
            string geoCodeInfo = wc.DownloadString(uri);
            XmlDocument xmlDoc = new XmlDocument();
            xmlDoc.LoadXml(geoCodeInfo);

            string duration = xmlDoc.DocumentElement.SelectSingleNode("//DistanceMatrixResponse/row/element/duration/value").InnerText;
            return Convert.ToInt32(duration) / 60;
        }
        catch (Exception)
        {
            return -1;
        }
        //return TravelingTime
    }
}

```



## DeliversBL

```

14 references
public class DeliversBL
{
    static DBConection db = new DBConection();
    //בשלב הראשון מתבצע סינון של המשלוחים שכבר סירבו לבקשה (אם קיימים)
    1 reference
    public static List<dtoPOSSIBLEDRIVE> GetAllOpenRequest(dtoDELIVERY p)
    {
        List<dtoPOSSIBLEDRIVE> AllOpenRequest = DTOclass.dtoPOSSIBLEDRIVE.CreateDtoList((db.GetDbSet<POSSIBLEDRIVE>().ToList()));
        List<NOTCONFIRM> NOTCONFIRMS = db.GetDbSet<NOTCONFIRM>();
        List<dtoPOSSIBLEDRIVE> request = new List<dtoPOSSIBLEDRIVE>();
        bool flag = false;
        foreach (var i in AllOpenRequest)
        {
            foreach (var item in NOTCONFIRMS)
            {
                if (item.PossibleDriveId == i.KODOFDRIVE && item.DeliveryId == p.DELIVERID)
                {
                    flag = true;
                }
            }
            if (flag == false)
            {
                request.Add(i);
            }
        }
        return request;
    }
    //בשלב השני מתבצע חיפוש המשלוחים המתאים ביותר:
    //ראשית, מתבצע בדיקת התאמה מוחלטת
    1 reference
    public static List<dtoPOSSIBLEDRIVE> AbsoluteFit(List<dtoPOSSIBLEDRIVE> request, dtoDELIVERY p)
    {
        List<dtoPOSSIBLEDRIVE> AllOpenRequest = new List<dtoPOSSIBLEDRIVE>();
        foreach (var item in request)
        {
            if (item.DATE == p.DATE && item.HOUR == p.HOUR)
            {
                //if()
                //כאן תהיה פונקציה של גוגל מפות שתבדוק את התאמת המקומות של כתובות המקור אל מול כתובות היעד
                AllOpenRequest.Add(item);
            }
        }
    }
}

```

```

44         return AllOpenRequest;
45     }
46     //שנית מתבצע בדיקת התאמה חלקית
    1 reference
47     public static List<dtoPOSSIBLEDRIVE> PartialFit(List<dtoPOSSIBLEDRIVE> AllOpenRequest, dtoDELIVERY p)
48     {
49         List<dtoPOSSIBLEDRIVE> Partialfitlist = new List<dtoPOSSIBLEDRIVE>();
50         foreach (var item in AllOpenRequest)
51         {
52             //כאן תהיה פונקציה חישוב מרחקים של גוגל מפות
53             //את כל הנסיעות המתאימות עם המרחק המינימלי Partialfitlist נכניס לרשימה של
54         }
55         return Partialfitlist;
56     }
57     //במידה ולאחר ההתאמה המוחלטת או החלקית נשיג כמה משלוחים באותה רמה - נבצע בדיקת הסתברות
58     //חיפוש כל הנסיעות עם מספר משלוחים מינימלי
59     2 references
60     public static List<dtoPOSSIBLEDRIVE> Minimumnumber(List<dtoPOSSIBLEDRIVE> Partialfitlist)
61     {
62         Min mincounter = new Min();
63         foreach (var i in Partialfitlist)
64         {
65             if (i.CountOfDeliveries < mincounter.counter)
66             {
67                 mincounter = new Min();
68                 mincounter.counter = i.CountOfDeliveries;
69             }
70             if (i.CountOfDeliveries == mincounter.counter)
71             {
72                 mincounter.alldeliveries.Add(i);
73             }
74         }
75         return mincounter.alldeliveries;
76     }
}

```

```

// חישוב רייטינג מקסימלי מבין כל הנסיעות עם מספר המשלוחים המינימלי
2 references
public static dtoPOSSIBLEDRIVE calculatemaxrating(List<dtoPOSSIBLEDRIVE> mincounter)
{
    Max maxrating = new Max();
    int w;
    foreach (var i in mincounter)
    {
        if (RatingBL.CalculatePoint(i.IDOFDELIVER) > maxrating.counter)
        {
            maxrating = new Max();
            maxrating.counter++;
            maxrating.point = RatingBL.CalculatePoint(i.IDOFDELIVER);
        }
        if (RatingBL.CalculatePoint(i.IDOFDELIVER) == maxrating.counter)
        {
            maxrating.counter++;
            maxrating.allratings.Add(i);
        }
    }
    // במידה וגם לאחר כל הסינונים הנל ישארו מספר נסיעות באותה הרמה-תתבצע ביניהם הגרלה
    if (maxrating.counter > 1)
    {
        Random r = new Random();
        w = r.Next(maxrating.counter);
        return maxrating.allratings[w];
    }
    return maxrating.allratings[0];
}
// פונקציית השיבוץ (מעדכנים אצל הבקשה את הקוד נסיעה של המשלוח המתאים)
7 references
public static void updatematch(dtoDELIVERY p, dtoPOSSIBLEDRIVE match)
{
    DELIVERIES d = new DELIVERIES();
    d=p.FROMdtoToTable(p);
    p.IDOFDELIVER =match.KODOFDRIVE;
    db.Execute<DELIVERIES>(d, DBConection.ExecuteActions.Update);
}
// המשתמש יכול להגיב על משלוח שנעשה לו
0 references
public static void Responsetodelivery()
{
}

```

## MainDelivery

```
class MainDelivery
{
    0 references
    public static void main(DELIVERIES d)
    {
        //פונקציה קיבלה רשומה מסוג הטבלה והופכת לסוג תצוגה
        dtoDELIVERY dtoDELIVERY = new dtoDELIVERY(d);
        List<dtoPOSSIBLEDRIVE> request = new List<dtoPOSSIBLEDRIVE>();
        //מפעילה את פונקציית הסינון של כל מי שסרב לבקשה
        request = DeliversBL.GetAllOpenRequest(dtoDELIVERY);
        //מפעילה את פונקציית בדיקת ההתאמה המוחלטת
        List<dtoPOSSIBLEDRIVE> Absolutefit = DeliversBL.Absolutefit(request, dtoDELIVERY);
        //אם חזר מהסינון רק רשומה אחת אז נעדכן את הנסיעה שהותאמה בדאטה בייס
        if (Absolutefit.Count == 1)
        {
            DeliversBL.updatematch(dtoDELIVERY, Absolutefit[0]);
            //אם חזר מהסינון יותר מרשומה אחת אז נפעיל שקלולי רייטנג ומספר משלוחים
            if (Absolutefit.Count > 1)
            {
                //הפעלת פונקציית בדיקת מספר משלוחנים מינמלי מבין כל הנסיעות המתאימות
                List<dtoPOSSIBLEDRIVE> Partialfitlist = DeliversBL.Minimumnumber(Absolutefit);
                //אם חזרה בדיוק נסיעה אחת -נעדכן אותה בדאטה בייס
                if (Partialfitlist.Count == 1)
                {
                    DeliversBL.updatematch(dtoDELIVERY, Partialfitlist[0]);
                    //אם חזר יותר מאחד-נפעיל פונקציית הרייטינג ונבחר מבין כולם את המקסימלי
                    if (Partialfitlist.Count > 1)
                    {
                        dtoPOSSIBLEDRIVE a = DeliversBL.calculatemaxrating(Partialfitlist);
                        //נעדכן בדאטה בייס
                        DeliversBL.updatematch(dtoDELIVERY, a);
                    }
                }
                if (Partialfitlist.Count == 0)
                {
                    Console.WriteLine("לא נמצאו נסיעות מתאימות במערכת");
                }
            }
        }
        //אם מהסינון של ההתאמה המוחלטת לא חזרו בכלל רשומות מתאימות אז נפעיל את פונקציית ההתאמה החלקית
        if (Absolutefit.Count == 0)
        {
            List<dtoPOSSIBLEDRIVE> Partialfit = DeliversBL.Partialfit(request, dtoDELIVERY);
            //אם חזר מהסינון רק רשומה אחת אז נעדכן את הנסיעה שהותאמה בדאטה בייס
            if (Partialfit.Count == 1)
            {
                DeliversBL.updatematch(dtoDELIVERY, Partialfit[0]);
                //אם חזר מהסינון יותר מרשומה אחת אז נפעיל שקלולי רייטנג ומספר משלוחים
                if (Partialfit.Count > 1)
                {
                    //הפעלת פונקציית בדיקת מספר משלוחנים מינמלי מבין כל הנסיעות המתאימות
                    List<dtoPOSSIBLEDRIVE> Partialfitlist = DeliversBL.Minimumnumber(Absolutefit);
                    //אם חזרה בדיוק נסיעה אחת -נעדכן אותה בדאטה בייס
                    if (Partialfitlist.Count == 1)
                    {
                        DeliversBL.updatematch(dtoDELIVERY, Partialfitlist[0]);
                        //אם חזר יותר מאחד-נפעיל פונקציית הרייטינג ונבחר מבין כולם את המקסימלי
                        if (Partialfitlist.Count > 1)
                        {
                            dtoPOSSIBLEDRIVE a = DeliversBL.calculatemaxrating(Partialfitlist);
                            //נעדכן בדאטה בייס
                            DeliversBL.updatematch(dtoDELIVERY, a);
                        }
                    }
                }
                if (Partialfitlist.Count == 0)
                {
                    Console.WriteLine("לא נמצאו נסיעות מתאימות במערכת");
                }
            }
        }
    }
}
```

## PossibleDriveBL

```
public class PossibleDriveBL
{
    // חיפוש כל הבקשות המתאימות שעוד לא אושרו-לנסיעה החדשה שנוספה
    static DBConnection db = new DBConnection();
    // מסננת בקשות שהוחמרו ועוד לא אושרו
    0 references
    public static List<dtoDELIVERY> GetAllOpenRequest(dtoPOSSIBLEDRIVE p)
    {
        List<dtoDELIVERY> AllOpenRequest = dtoDELIVERY.CreateDtoList((db.GetDbSet<DELIVERIES>().Where(r => r.IDOFDELIVER != null && r.DONE = false))).ToList();
        return AllOpenRequest;
    }
    // הפונקציה בודקת את סך הנקודות של הנסיעה החדשה אל מול הנסיעה הישנה
    0 references
    public static void match(List<dtoDELIVERY> AllOpenRequest, dtoPOSSIBLEDRIVE p)
    {
        foreach (dtoDELIVERY d in AllOpenRequest)
        {
            dtoPOSSIBLEDRIVE dto = new dtoPOSSIBLEDRIVE();
            dto = db.GetDbSet<POSSIBLEDRIVE>().Where(POSSIBLEDRIVE.KODOFDRIVE = d.IDOFDELIVER);
            float prepoint = 0;
            float newpoint = 0;
            newpoint = checkpoint(p, d);
            prepoint = checkpoint(dto, d);
            // אם סכום הנקודות של הנסיעה הישנה יותר קטן אז הדאטה בייס יעודכן בפרטי הנסיעה החדשה
            if (prepoint > newpoint)
            {
                DeliversBL.updatematch(d, p);
            }
        }
    }
}
```

// פונקציה שמחשבת ניקוד למשלוחן בהתאם לבקשה מסוימת

2 references

```
public static float checkpoint(dtoPOSSIBLEDRIVE p, dtoDELIVERY d)
{
    float point = 0;
    float date = 0;
    float hour = 0;
    float Sourceaddress = 0;
    float Destinationaddress = 0;

    date = d.DATE - p.DATE;
    hour = d.HOUR - p.HOUR;
    //Sourceaddress= מפות גוגל
    //Destinationaddress= מפות גוגל
    point = date + hour + Sourceaddress + Destinationaddress;
    return point;
}
```

## RatingBL

```

class RatingBL
{
    static DBConnection db = new DBConnection();
    3 references
    public static long CalculatePoint(long tz)
    {
        long point = 0;
        List<RATING> AllRating = db.GetDbSet<RATING>().Where(m => m.IDOFDELIVER == tz).ToList();
        foreach (var i in AllRating)
        {
            point += (long)i.INTEGRITYDELIVER;
            point += (long)i.LATE;
            point += (long)i.SERVICE;
            point += (long)i.GENERAL;
        }

        foreach (var i in AllRating)
        {
            i.SamPoint = point;
        }

        foreach (var i in AllRating)
        {
            db.Execute<RATING>(i, DBConnection.ExecuteActions.Update);
        }
        return point;
    }
}

```

## UserBL

```

public class UserBL
{

    static DBConection db = new DBConection();

    0 references
    public static List<dtoUSER> GetallUsers()
    {
        List<USERS> list = db.GetDbSet<USERS>().ToList();
        List<dtoUSER> dtoList = dtoUSER.CreateDtoList(list);
        return dtoList;
    }

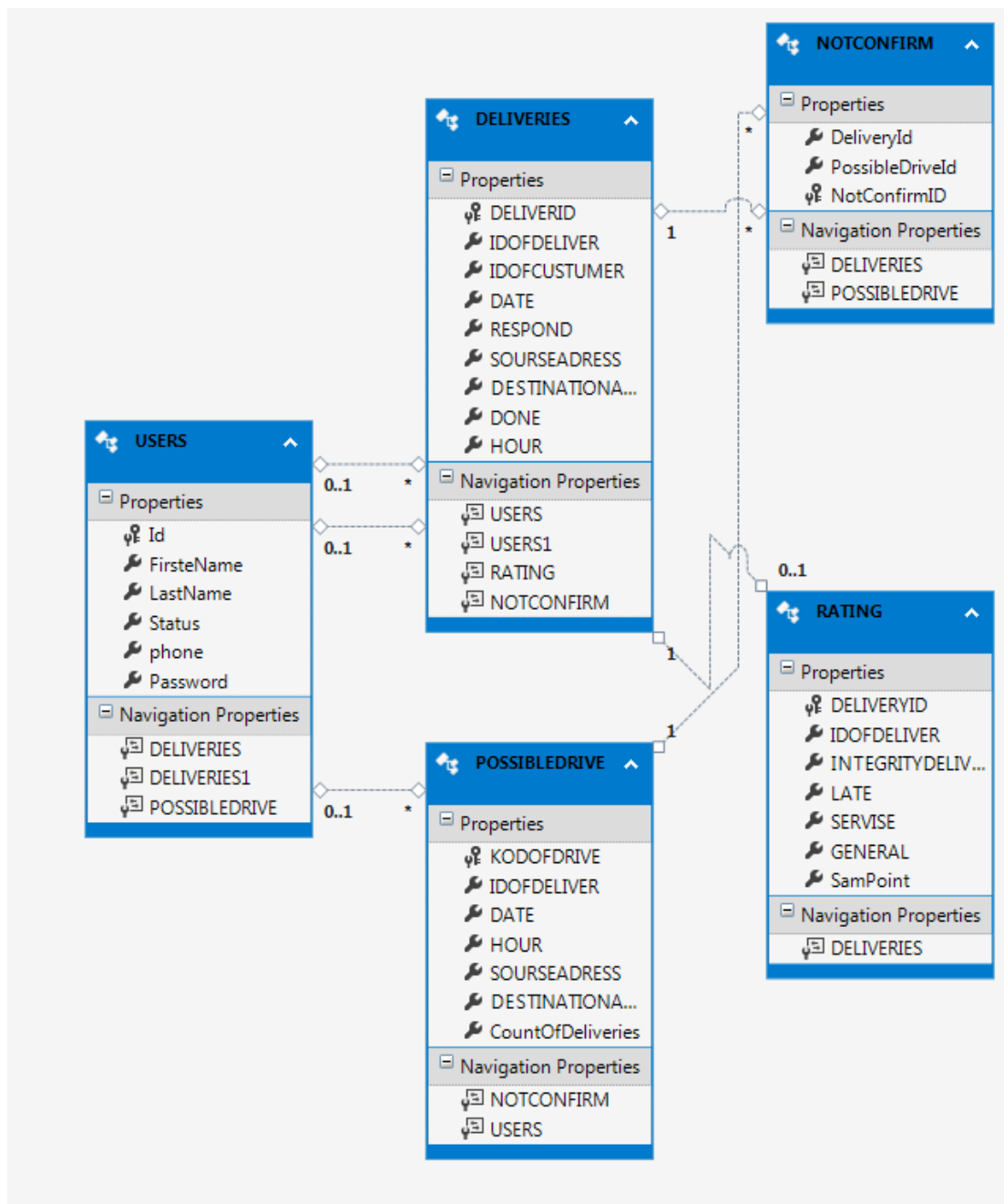
    0 references
    public static Object SignUp(dtoUSER user)
    {
        USERS u = user.FromdtoToTable();
        db.Execute<USERS>(u, DBConection.ExecuteActions.Insert);
        return u;
    }

    0 references
    public static Object SignIn(UserDetails ud)
    {
        USERS u = db.GetDbSet<USERS>().FirstOrDefault(u1 => u1.FirsteName == ud.firstname);
        if (u == null)
            return new { success = false, message = "user does not exist" };
        else if (!u.Password.Equals(ud.Password))
            return new { success = false, message = "Password is wrong" };
        else
            return new { success = true, user = new dtoUSER(u), message = "Login success" };
    }
}

```

## תיאור מסד הנתונים

פירוט הטבלאות ב- Data Base



**שם הטבלה: DELIVERIES**

**תפקיד הטבלה:** בטבלה זו ישמרו כל הבקשות למשלוח אלו שואשרו כבר ואלו שלא (מה שאושר  
השדה DONE יהיה שווה ל false).

שם שדה	טיפוס	תיאור	שדה שאינו חובה	מפתח
DELIVERID	bigint	קוד משלוח	כן	PK
IDOFDELIVER	bigint	תז משלוחן	לא	FK
IDOFCUSTOMER	bigint	תז לקוח	לא	FK
DATE	date	תאריך שליחת המשלוח	לא	
RESPOND	bit	האם הגיבו על המשלוח	לא	
SOURCEADDRESS	varchar	כתובת מקור	לא	
DESTINATIONADDRESS	varchar	כתובת יעד	לא	
DONE	bit	האם אושר	לא	
HOUR	time	שעה	לא	



**שם הטבלה: NotConfirm**

**תפקיד הטבלה:** טבלת קשר - בטבלה זו ישמרו כל הבקשות שלא אושרו ולצדם מיהו הנהג שלא אישר כך שלחיפוש הבא הוא לא יכנס.

שם שדה	טיפוס	תיאור	שדה שאינו חובה	מפתח
DeliveryId	bigint	קוד משלוח	כן	PK,FK
PossibleDriveId	bigint	קוד נסיעה	כן	PK,FK

**שם הטבלה: POSSIBLEDRIVE**

**תפקיד הטבלה:** טבלה ובה ישמרו כל הנסיעות האפשריות (המשלוחנים יכניסו פרטים אודות נסיעה)

שם שדה	טיפוס	תיאור	שדה שאינו חובה	מפתח
KODOFDRIVE	bigint	קוד נסיעה	כן	PK
IDOFDELIVER	bigint	תז משלוחן	לא	FK
DATE	date	תאריך נסיעה	לא	
HOUR	time	שעה	לא	
SOURCEADDRESS	varchar	כתובת מקור	לא	
DESTINATIONADDRESS	varchar	כתובת יעד	לא	
CountOfDeliveries	int	מספר משלוחים שקיבל	לא	

**RATING: שם הטבלה:**

**תפקיד הטבלה:** טבלת דירוג רייטינג הלקוח יתן ניקוד מ-1-הכי נמוך ל-5-הכי גבוה כתגובה על משלוח שנעשה לו

שם שדה	טיפוס	תיאור	שדה שאינו חובה	מפתח
DELIVERYID	bigint	קוד משלוח/נסיעה	כן	PK,FK
IDOFDELIVER	bigint	תז משלוחן	לא	
INTEGRITYDELIVER	int	שלמות המשלוח	לא	
LATE	int	האם איחר	לא	
SERVISE	int	שירות	לא	
GENERAL	int	ניקוד כללי	לא	
SamPoint	bigint	סהכ	לא	

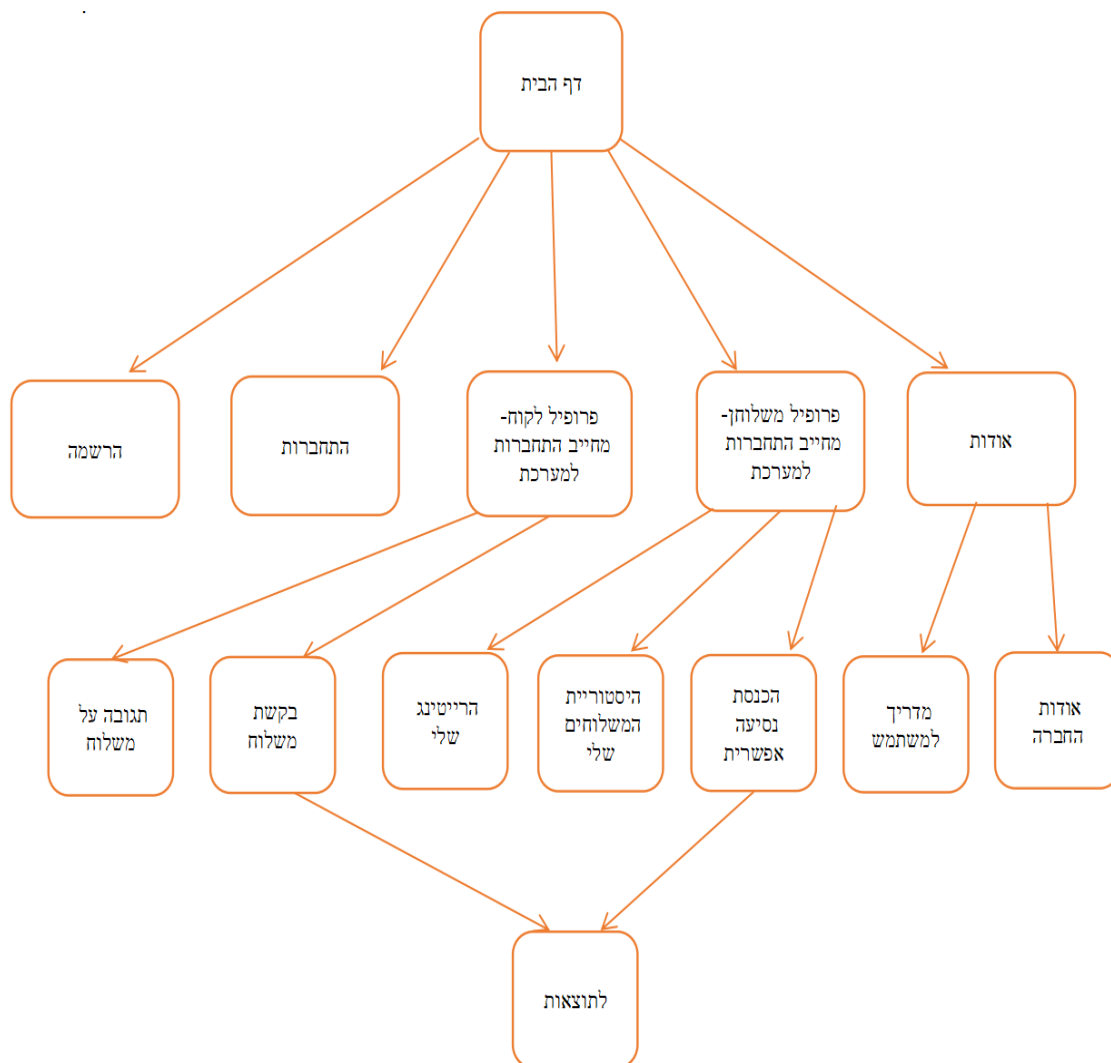
**שם הטבלה: USERS:**

תפקיד הטבלה: טבלת משתמשים שישמרו בה פרטים בסיסיים

שם שדה	טיפוס	תיאור	שדה שאינו חובה	מפתח
Id	bigint	תז משתמש	כן	pk
FirstName	varchar	שם פרטי	לא	
LastName	varchar	שם משפחה	לא	
Status	varchar	סטטוס	לא	
phone	varchar	מס טלפון	לא	
Password	varchar	סיסמא	לא	

## מדריך למשתמש

### תיאור המסכים



## מדריך למשתמש

בכניסה לאתר יוצג דף הבית כברירת מחדל. דף הבית מכיל תפריט, האפשרויות בתפריט הם: אודות-כאן המשתמש יוכל לקרוא אודות החברה וכן לקרוא את המדריך לשימוש באפליקציה. פרופיל משלוחן-בחירה זו מחייבת התחברות למערכת, לאחר ההתחברות המשתמש יכול לבחור להכניס נסיעה אפשרית או לקבל את רשימת המשלוחים שעשה עד היום או לצפות בנתוני הרייטינג שלו.

אם בחר באפשרות הכנסת נסיעה אפשרית הוא יקבל טופס ובו ימלא פרטים בסיסיים אודות הנסיעה ולאחר השליחה הוא ילחץ על אפשרות "לתוצאות" ויוצגו בפניו תוצאות המערכת (רשימה של משלוחים שמתאימים לדרישותיו ואפשרות ליצירת קשר עם הלקוחות).

פרופיל לקוח- גם מחייב התחברות למערכת, לאחר ההתחברות המשתמש יוכל להכניס בקשה למשלוח וכן להגיב על משלוח.

אם בחר בהכנסת משלוח הוא יקבל טופס ובו ימלא פרטים בסיסיים אודות הבקשה (כתובת מקור יעד וכו') לאחר האישור הוא ילחץ על אפשרות "לתוצאות" ויוצגו בפניו תוצאות המערכת (מספר טלפון ושם של נהג שמוכן לעשות לו את הנסיעה).

-הרשמה- המשתמש יזין פרטים בסיסיים כמו: שם פרטי, משפחה, מספר טלפון, תעודת זהות ועוד. -התחברות- המשתמש יתחבר למערכת באמצעות שם המשתמש והסיסמא שלו.

## צילומי מסכים

### מסך הבית:



**חודש שכחתם בו?**  
חברת המשלוחים הגדולה בארץ ובעולם  
ני אפליקציה זו תוכלו לשלוח משלוחים וכן לעשות נסיעות לכל חלקי הארץ בדרך הפשוטה והיעילה  
האפליקציה פותחה בשנת 2022 בטכנולוגיה מתקדמת וממשק נוח לשימוש יש לכם מה למד? הצעות לשדרוג? טקסט? לט  
@שיירה בורה

### אפשרות אודות:

### אודות החברה:



**קצת עלינו**  
ByTheWay deliveries היא אפליקציית משלוחים שיתופית  
דרך אפליקציה זו תוכלו לשלוח משלוחים מכל סוג, בכל מקום ולכל מקום בדרך פשוטה ונוחה  
בזמן קצר. להכנס פרטי נסיעה אפשרית - שאותם בכל אופן תעשו - להוראות עליה  
האפליקציה פותחה בשנת 2022 בטכנולוגיה מתקדמת וממשק נוח וידידותי לשימוש  
יש לכם מה למד? הצעות לשדרוג? טקסט? לט  
כל הזכויות שמורות לשיירה בורה - אין להעתיק או לשכפל חלקים מהמאמר ללא רשות



## מדריך למשתמש:



### מדריך למשתמש

בכניסה לאחר יישוג דרך הבית בבירור מחולל, דף הבית מפיל תפריט האפשרויות בתפריט הב:

#### אודות

כאן המשתמש יוכל לקרוא אודות החברה וכן לקרוא את המדריך לשימוש באפליקציה

#### פרופיל משלוח

בחירה זו מחייבת התחברות למערכת.

לאחר ההתחברות המשתמש יוכל לבחור להכניס נסיעה אפשרית או לקבל את רשימת המשלוחים שעשה עד היום או לצפות בתזונו הרייטינג שלו. אם בחר באפשרות הכנסת נסיעה אפשרית הוא יקבל טופס ובו ימלא פרטים בסיסיים אודות הנסיעה לאחר השליחה הוא ילחץ על אפשרות "לחצאנא" ויצג בפניו חוזאות המערכת (רשימה של משלוחים שמתאימים לרשותו ואפשרות לצריר קשר עם הלקוחות).

#### פרופיל לקוח

גם מחייב התחברות למערכת. לאחר ההתחברות המשתמש יוכל להכניס בקשה למשלוח וכן להציג על השליח

אם בחר בהכנסת משלוח הוא יקבל טופס ובו ימלא פרטים בסיסיים אודות הבקשה (כמות מקור יעד וכו'). לאחר האישור הוא ילחץ על אפשרות "לחצאנא" ויצג בפניו חוזאות המערכת (מספר טלפון ואם של נרג שמות לעשות לו את הנסיעה).

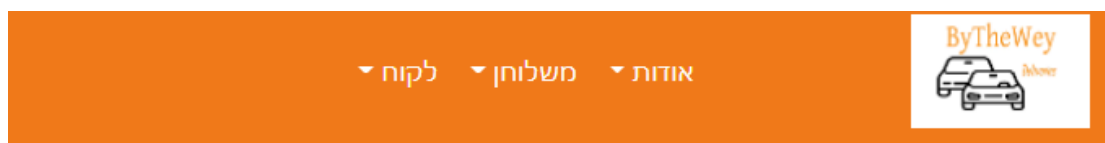
#### הרשמה

המשתמש יצין פרטים בסיסיים כמו: שם פרטי, משפחה, מספר טלפון, תעודת זהות ועוד.

#### התחברות

המשתמש יתחבר למערכת באמצעות שם המשתמש והסימא שלו.

## פרופיל משלוח:



- הכנס נסיעה אפשרית
- היסטורית המשלוחים שלי
- הרייטינג שלי

## הכנסת נסיעה אפשרית:

החשבון

הרשמה

אודות

משלוח

לקוח

למכשאות



מלא פרטים בסיסיים אודות נסיעה עתידית שלך

תעודת זהות

תאריך

שעה

כתובת מקור

כתובת יעד


אישור

## פרופיל לקוח:

אודות

משלוח

לקוח



- הכנס בקשה למשלוח
- הגב על משלוח



## בקשה למשלוח:

התחברות

הרשמה

אודות • משלוח • לקוח • למעצמות

מלא פרטים אודות בקשה למשלוח שלך

תעודת זהות

תאריך

שעה

כתובת מקור

כתובת יעד

אישור

## הרשמה:

התחברות

הרשמה

אודות • משלוח • לקוח • למעצמות

תודה שבחרת להצטרף אלינו! נא מלא פרטים אישיים

תעודת זהות

שם פרטי

שם משפחה

סטטוס

משלוח

מספר תלפון

סיסמא

אימות סיסמא

אישור

## התחברות:



הכנס שם משתמש וסימא

שם פרטי

סימא

אישור

### תיאור מסכים:

כשנכנסים בוחרים פרופיל משלוח או לקוח

משלוח – נכנס ורואה חבילות מתאימות

לקוח – מראה משלוח מתאים

### תיאור פעילות האפליקציה:

- משלוח יכול לבחור כמה לעשות, האם לעשות, מאיפה, לאן.
- המערכת בוחרת אם להציע לו בכלל – מתחשבת באילוצים.
- כל משלוח שמקבל בקשה יכול לאשר ויש שדה של כמות משלוחים שמתעדכן.
- כשלקוח נכנס ומכניס הזמנה (מקור ויעד).
- המערכת מפעילה שיקול דעת של משלוחנים שהכי פחות תפוסים.

## בדיקות והערכה

לאחר הרצת האלגוריתם נבחנו כל האילוצים שדרושים כדי להביא לשירות משלוחים יעיל ואופטימלי. כאשר הופיעו טעויות ובאגים בביצוע של האלגוריתם נבדק הקוד שוב עד שתוקנו הבעיות. לאחר בדיקות רבות אחר כל מקרי הקצה שעלו בדעתי, והרצת האלגוריתם מספר פעמים על נתונים שונים, האלגוריתם הגיע לקירוב האפשרי ביותר בכלים העומדים לרשותי.

## ניתוח יעילות

בבואי לתכנן את האלגוריתם נתקלתי רבות בשאלת היעילות. ביצועי הפרויקט חייבים להיות יעילים שכן זה קשור באופן ישיר לזמן התגובה של המשתמש, ברגע שהביצועים גרועים, זמן התגובה למשתמש מתארך והאתר חושב הרבה זמן. דבר המעצבן את המשתמשים ומוריד את האחוזים לשימוש באתר ובנוסף מכביד גם על המעבד וגרם לביצועים איטיים ומסורבלים יותר, על כן נתתי חשיבות רבה ליעילות בביצועים ואף לעיתים העדפתי את היעילות על פני דברים אחרים.

## אבטחת מידע

כדי שהשימוש במערכת יהיה בטוח ואזור המשתמשים יהיה אישי וחסוי על כל משתמש שרוצה לעשות פעולה כשלהי במערכת חלה חובה להתחבר באמצעות שם משתמש וסיסמא אישיים. במידה ושם המשתמש או הסיסמא לא נכונים תוקפץ לו הודעת שגיאה והוא לא יוכל להיכנס למערכת.

## מסקנות

מסקנותיי בעת סיום הפרויקט :

שצברתי ידע רב מקיף בכל מיני נושאים וצברתי המון ניסיון ועבודה מעשית בעת פיתוח האפליקציה למדתי איך לכתוב קודים בצורה יעילה וחכמה, איך לפתור באגים לבד, להתממשק עם אתרים חיצוניים ועוד.

יותר מהכול - הגעתי למסקנה שהרצונות השאיפות, המשמעת העצמית, והעמידה בלוח זמנים הם אלו שהובילו אותי להצלחה.

אני מלאת אופטימיות לגבי הפרויקט שלי, אני בטוחה שהוא יתרום לחיי היום של כולנו

מה שלדעתי הכי חשוב זה לקחת אחריות ולקבוע זמן מסוים כל יום - כדי לעבוד על הפרויקט

סוף כל סוף הגעתי לזמן חתימת הספר, בו חשפתי טפח ממה שמאחורי הפרויקט וסקרתי במעט את תהליך ההתלמדות המופלא, רצוף העמל וההשקעה, עמקני ועשיר בגילויים כה מרתקים, בו הוספתי ידע והתנסות בתחומים רבים ומגוונים.

תחילה, תכנן הפרויקט ושלבי העבודה, נראתה בעיני כמשימה בלתי אפשרית. זאת בשל ההיקף, המורכבות ולחץ הזמן. חלקים מסוימים שתוכננו נפסלו מראש כי חששתי שלא אצליח לבצע אותם, אך בסופו של תהליך ניתן לומר שהכול אפשרי. כל שהיה נראה בלתי עביר, הפרויקט בכללותו, וכן חלקים מסוימים שנפסלו- הכול התברר כבר ביצוע. השכלתי להבין שעם הרבה מוטיבציה ומוסר עבודה, ניתן להשלים כל משימה, קשה ומורכבת ככל שתהיה.

בנוסף כמעט על כל צעד ושעל בפיתוח נתקלתי בתחומי ידע לא מוכרים, מורכבים, הרבה מעבר לרמת הידע שיש לנו. היתקלויות אלה שדרשו פתרונות, גרמו לי להתנסות המון בלמידה עצמית.

ערכתי הכרות עם מגוון כלים וטכנולוגיות, גיליתי שאין בעיה חסרת פתרון וגם לסטודנטית חסרת ידע רחב ומקיף כמוני ישנה אפשרות להיכנס לנושא שהיא לא מכירה להתאמץ להבין ולמצוא פתרון.

ניתן לומר שכסטודנטית ומתכנתת, הפרויקט הכניס אותי לעולם היוצרים בכלל והמתכנתים בפרט. למדתי לתכנן מערכת ע"י ניתוח ואפיון הצרכים והאתגרים, למדתי לחשוב על כל הפרטים ופרטי הפרטים ועל כל הבעיות שיכולות לצוץ ולמצוא את הדרך היעילה ביותר להתמודד איתן. למדתי המון על דרכי חקירה, פיתוח וכתיבה, והרבה על הסיפוק העצום שביצירה.

את הפרויקט אפיינתי באופן בולט בכתיבה עצמאית. את התוצר הנפלא יצרתי בעשר אצבעות, בנחישות ובאומץ. שברתי את הראש, כמעט כפשוטו: "לכלכתי" הרבה את הידיים בכתיבה ומחיקה וכתיבה חוזרת. נאבקתי קשות במלחמת הבהירות ונטרול השגיאות. ובעצם, פיתחתי אפליקציה בעצמי, מא' ועד ת'. וזו התנסות מדהימה. פיתוח עצמי הוא אולי לא הדרך הכי טובה להגיע למוצר בקלות ובלי

להזיע, אולם במבט לאחור, את מטרת הלימדה השגתי גם השגתי, והכלים שרכשתי הם נכס עוצמתי ששווה את הכל ואותו קנינו לתמיד.

כאן המקום להודות גם לצוות לחברות, למשפחה וכמובן לבורא העולם.

## פיתוח עתידי

אם יהיה לי זמן ומשאבים נוספים הייתי רוצה להוסיף מנגנוני הבטחה נוספים כדי שהמערכת תהיה בטוחה לשימוש לדוגמא: להוסיף מנגנון שיוודא שאכן המשלוח הגיע ליעדו ובשלמותו, להבטיח שלא יהיו גנבות ועוד.

כמו כן הייתי רוצה להוסיף לאפליקציה את הסדרי התשלום ולקבוע תעריף מסוים.

## ביבליוגרפיה

- Stackoverflow-<https://stackoverflow.com/> •
- Angular material -<https://material.angular.io/> •
- Github-<https://github.com/> •
- Bootstrap-<https://getbootstrap.com/> •
- w3schools-<https://www.w3schools.com/> •
- developer.mozilla-<https://developer.mozilla.org/en-US/> •
- GeeksForGeeks-<https://www.geeksforgeeks.org/> •