ספר פרויקט



BYTHEWAY deliveries שם הפרויקט:

שם התלמידה: שירה בורה

תז:212423693

סמינר: גברא

מנחה: יעל עמר

תאריך הגשה: יולי

תוכן

[1. הצעת פרויקט 3](#_Toc102467141)

[2. מבוא / תקציר 3](#_Toc102467142)

[2.1. הרקע לפרויקט 3](#_Toc102467143)

[2.2 תהליך המחקר 3](#_Toc102467144)

[2.3 סקירת ספרות 3](#_Toc102467145)

[3. מטרות ויעדים 3](#_Toc102467146)

[4. אתגרים 4](#_Toc102467147)

[5. מדדי הצלחה 4](#_Toc102467148)

[6. תיאור המצב הקיים 4](#_Toc102467149)

[7. רקע תאורטי 4](#_Toc102467150)

[8. ניתוח חלופות מערכתי 4](#_Toc102467151)

[9. תיאור החלופה הנבחרת והנימוקים לבחירה 4](#_Toc102467152)

[10. אפיון המערכת 5](#_Toc102467153)

[10.1. ניתוח דרישות המערכת 5](#_Toc102467154)

[10.2. מודול המערכת 5](#_Toc102467155)

[10.3. אפיון פונקציונאלי 6](#_Toc102467156)

[10.4. ביצועים עיקריים 6](#_Toc102467157)

[10.5. אילוצים 6](#_Toc102467158)

[11. תיאור הארכיטקטורה 6](#_Toc102467159)

[11.1. הארכיטקטורה של הפיתרון המוצע בפורמט של Design level Down-Top 6](#_Toc102467160)

[11.2. תיאור הרכיבים בפיתרון 7](#_Toc102467161)

[11.3. ארכיטקטורת רשת (לא רלוונטי ) 8](#_Toc102467162)

[11.4. תיאור פרוטוקולי התקשורת 8](#_Toc102467163)

[11.5. שרת – לקוח 8](#_Toc102467164)

[11.6. תיאור הצפנות (לא רלוונטי) 8](#_Toc102467165)

[12. ניתוח ותרשים use case של המערכת המוצעת 8](#_Toc102467166)

[12.1. רשימת use case 9](#_Toc102467167)

[12.2. תיאור ה-use case העיקריים של המערכת 9](#_Toc102467168)

[12.3. מבני נתונים בהם משתמשים בפרויקט 9](#_Toc102467169)

[12.4. תרשים מחלקות 9](#_Toc102467170)

[12.5. תיאור המחלקות 9](#_Toc102467171)

[13. תיאור התוכנה 9](#_Toc102467172)

[14. אלגוריתמים מרכזיים 10](#_Toc102467173)

[14.1. חלק מהאלגוריתם... הפיכת התמונה לשחור לבן 10](#_Toc102467174)

[14.2. חלק אחר מהאלגוריתם... פירוק התמונה לאותיות 10](#_Toc102467175)

[14.3. החלק העיקרי באלגוריתם למידת מכונה – וזיהוי האות 10](#_Toc102467176)

[15. \*קוד האלגוריתם 10](#_Toc102467177)

[16. תיאור מסד הנתונים 10](#_Toc102467178)

[16.1. פירוט הטבלאות ב- Data Base 10](#_Toc102467179)

[17. מדריך למשתמש 10](#_Toc102467180)

[17.1. תיאור המסכים 10](#_Toc102467181)

[17.2. מדריך למשתמש 11](#_Toc102467182)

[17.3. צילומי מסכים 11](#_Toc102467183)

[18. בדיקות והערכה 11](#_Toc102467184)

[19. ניתוח יעילות 11](#_Toc102467185)

[20. אבטחת מידע 11](#_Toc102467186)

[21. מסקנות 11](#_Toc102467187)

[22. פיתוח עתידי 11](#_Toc102467188)

[23. ביבליוגרפיה 11](#_Toc102467189)

# הצעת פרויקט

**סמל מוסד:**

**שם מכללה:**

**שם הסטודנט:** שירה בורה

**ת.ז. הסטודנט:** 212423693  
**שם הפרויקט:**

**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 בן 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

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

**מסד נתונים:** SQL

**פרוטוקלי 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 יום של האנשים, חשבתי על דברים שלא קיימים היום , דברים חסרים שיכולים להועיל .כמו כן חיפשתי משהו חדש שיגרום לי לפתח מימניות חשיבה חדשות ולהכיר דברים חדשים תוך כדי כתיבת האלגוריתם.

בחרתי ברעיון של שירות משלוחים שיתופי, מה שעזר לי להגיע לרעיון הזה זה החסר- קרה לי הרבה פעמים שרציתי להעביר משהו ולא יכולתי להעביר -והיו מקרים ממש דחופים, ותמיד חלמתי שיקום איזה מיזם או אפליקציה שתתן את השרות הזה ,כשהגעתי למגמה הפכתי את החלום למציאות. 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 יוכל 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, נדרש 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 והתאמתי אותה לפרויקט שלי עי 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, והטמעתי 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 overflowעל 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 יום ותשפר את איכות החיים של כל אחד מאיתנו .

מטרה נוספת היתה להרחיב אופקים ולהרוויח ידע נוסף בתחום התכנות וההיטק תוך כדי צבירת ניסיון על פרויקטים ועבודה מעשית ,במהלך הפרויקט צברתי המון ידע ומיומניות 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 שירות המשלוחים. אם עד היום אדם שרצה לשלוח איזה חבילה למקום כלשהו ברחבי הארץ-ולא היתה לו אפשרות כזו עכשיו תהיה לו אפשרות לעשות זאת בצורה הכי קלה ונוחה עי הרשמות בסיסית לאפליקציה ושם הוא יוכל לבקש לשלוח חבילה או להרשם 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 הAPI בפרויקט שלי בכפוף לנהלי ההתחיבות של גוגל מפות, הייתי צריכה לפתוח חשבון וליצור מפתח לשימוש google maps platform . והחלק הכי קשה בשלב הזה היה למצוא את הפונקצייה של חישוב מרחקים ולהתאים אותו לפרויקט הספציפי שלי.

-בחלק של הצד לקוח 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 התיעציות וקריאת חומרים בנושא צלחתי אותם.

בכל מהלך הפרויקט החל מהיגוי הרעיון דרך ההצעת פרויקט, העבודה עצמה ,וכלה בהגשה -נדרשתי להתייעץ, להשקיע מחשבה לקרוא וללמוד על חומרים מסוימים ואפילו לשנות כיוון לפעמים

# 

# מדדי הצלחה

הפרויקט שלי הצליח אם אכן האפליקציה תצליח לקשר בין משלוחנים ולקוחות בהתחשב בכל הנתונים:

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

האפליקציה בחרה 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 אפשריות מגוונות :אודות החברה ,איך משתמשים באפליקציה, פרופיל לקוח -כולל שליחת משלוח ,תגובה על משלוח וכו,וכן 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 הספצפיות שרשומות אצלהם במאגר.

# רקע תאורטי

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 הAPI 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 משלוחנים שמתאימים לדרישות ותציע להם את השירות -כך בעצם המערכת מכסה את כל האפשריות 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. באופן אנושי -פשוט להתקשר לקרובים ובני משפחה ולבדוק מי רוצה לעשות משלוח

2. בקבוצות מסודרות של קבוצות וואצפ -להזין את הנונים כל פעם מחדש ולקוות שאכן יהיה אדם מתאים

3. להמציא אלגוריתם שיפתור את הבעיה

אני בחרתי באפשרות השלישית מפני שהיא הכי מועילה והכי מתקדמת

# 

# תיאור החלופה הנבחרת והנימוקים לבחירה

הרעיון שפיתחתי הוא פיתוח טכנולוגי שמחבר בין לקוח למשלוחן בהתחשב בהרבה נתונים וחישובים.

הרעיונות האחרים שהצעתי לגבי המערכת של לא היו רלוונטים מפני שהם מאד מסובכים ומיושנים וגם רוב הזמן לא יהיה מענה 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 אפשריות או משלוחן או לקוח.

בפרופיל משלוחן הוא יוכל להכניס נסיעה אפשרית ,לצפות בהיסטוריית המשלוחים שלו, וכן את נתוני הרייטינג שלו.

בפרופיל לקוח יעמדו בפניו האפשריות 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 בסיסית -כתובת מקור=לכתובת מקור, כתובת יעד=לכתובת יעד ,התאריכים,וכו.

המערכת מחשיבה כל כתובת שנמצאת ברדיוס של עד 10 קילומטרים מהכתובת שהוזנה על ידי המשתמש כמתאימים.

לאחר הסינון תתקבל רשימה של משלוחנים מתאימים ובשלב הזה המערכת תעשה סינון נוסף הפעם בהתחשב במדד המישלוחים שכבר ביצע -המערכת תבחר את המינימלי מבין כל הרשימה ,בסמידה ויהו כמה מינימלים המערכת תבחר לפי דרוג רייטינג-למי שיש דירוג ריטינג 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.

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 תעשה את אותם פעולות רק להפך.

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 RAM 32GB i7 עמדת פיתוח: מחשב Lenovo מערכת ההפעל ה: 10 Windows

שפות תוכנה: #C , תוך שימוש בטכנולוגי ת WebApi, אנגולר .

כלי תוכנה לפיתוח המערכת: .Microsoft Visual Studio2019, vs code:

מסד נתונים:.SQL Server עמדת משתמש מינימאלי ת :

* + חומרה: מעבדRAM 4GB i5 .

o מערכת ההפעל ה: Windows 7ומעלה.

* + חיבור לרש ת: נדרש .
  + . chrome :תוכנות

## 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 לדאטה בייס .
* הפעלת האלגוריתם- הפעלת הפונקציה המתאימה בהתאם לבחירת המשתמש
* מציאת התאמה בין משלוחן ללקוח
* הזנת פרטי הנסיעה והמשלוח בדאטה בייס
* חיבור בין המשלוחן ללקוח על ידי הצגת המספרים של האחד לשני.

## אפיון פונקציונאלי

פירוט פונקציות עיקריות ותפקידן:

GetAllOpenRequest(dtoDELIVERy p)-פונקצציה שמקבלת בקשה מסוג תצוגה ומסננת עבורה את כל המשלוחנים שלא סרבו לבקשה זו.

Absolutefit(List<dtoPOSSIBLEDRIVE> reqest,dtoDELIVERy p)-הפונקציה מקבלת רשימה של נסיעות ובקשה ומחזירה רשימה של משלוחנים שיש להם התאמה מוחלטת עם הנתונים של הנסיעה

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

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

CalculatePoint-הפונקצייה מקבלת משתמש מסוים ומחזירה את חישוב הרייטינג שלו בנקודות

CheckMatchBetweenDeliverAndCustumer-קלט: משלוחן ובקשה פלט: ניקוד רמת התאמה

CheckRatingBetweenDeliverAndCustumer() קלט: משלוחן ובקשה פלט: ניקוד רמת הסתברות.

SignIn-התחברות למערכת עי שם משתמש וסיסמא .כשהמשתמש יכניס את הנתונים המערכת תבצע בדיקת תקינות ותאשר לו לבצע פעולות אם לא יהיה התאמה תקפוץ לו הודעת שגיאה

Signup -הרשמה למערכת עי הכנסת פרטים בסיסיים כמו שם פרטי ,משפחה,תעודת זהות ועוד .לאחר שהמשתש ילחץ על אישור ,הפונקצציה תשלח לשרת והנתונים ישמרו במסד הנתונים.

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

GetAllOpenReqwest(dtoPOSSIBLEDRIVE p-מסננת בקשות שהותאמו ועוד לא אושרו

Match- הפונקצייה בודקת את סך הנקודות של הנסיעה החדשה אל מול הנסיעה הישנה אם הניקוד של החדשה קטן מהניקוד של הישנה אז הנסיעה החדשה תשובץ לבקשה

Checkpoint-בודקת את הניקוד של משלוחן

MainpossibledriveBL.main-הפונקצייה מופעלת ברגע שנכנסת נסיעה חדשה למערכת-הפונקציה תפעיל את תתי הפונקציות הנדרשות בהתאם לערך המוחזר.

Viewhistory-צפייה בנתנוי המשלוחים הקושמים של משתמש מסוים

Viewrating-צפייה בנתנוני רייטינג של משתמש

Responsetodelivery-פונקצייה שמעדכנת את הנתונים שהלקוח הכניס כתגובה על משלוח מסוים-לדאטה בייס.

## ביצועים עיקריים

כל משתמש יכול להרשם לאתר עי הכנסת פרטים אישיים

לאחר ההתחברות הוא יכיניס 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 תתן 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 הפיתרון 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.

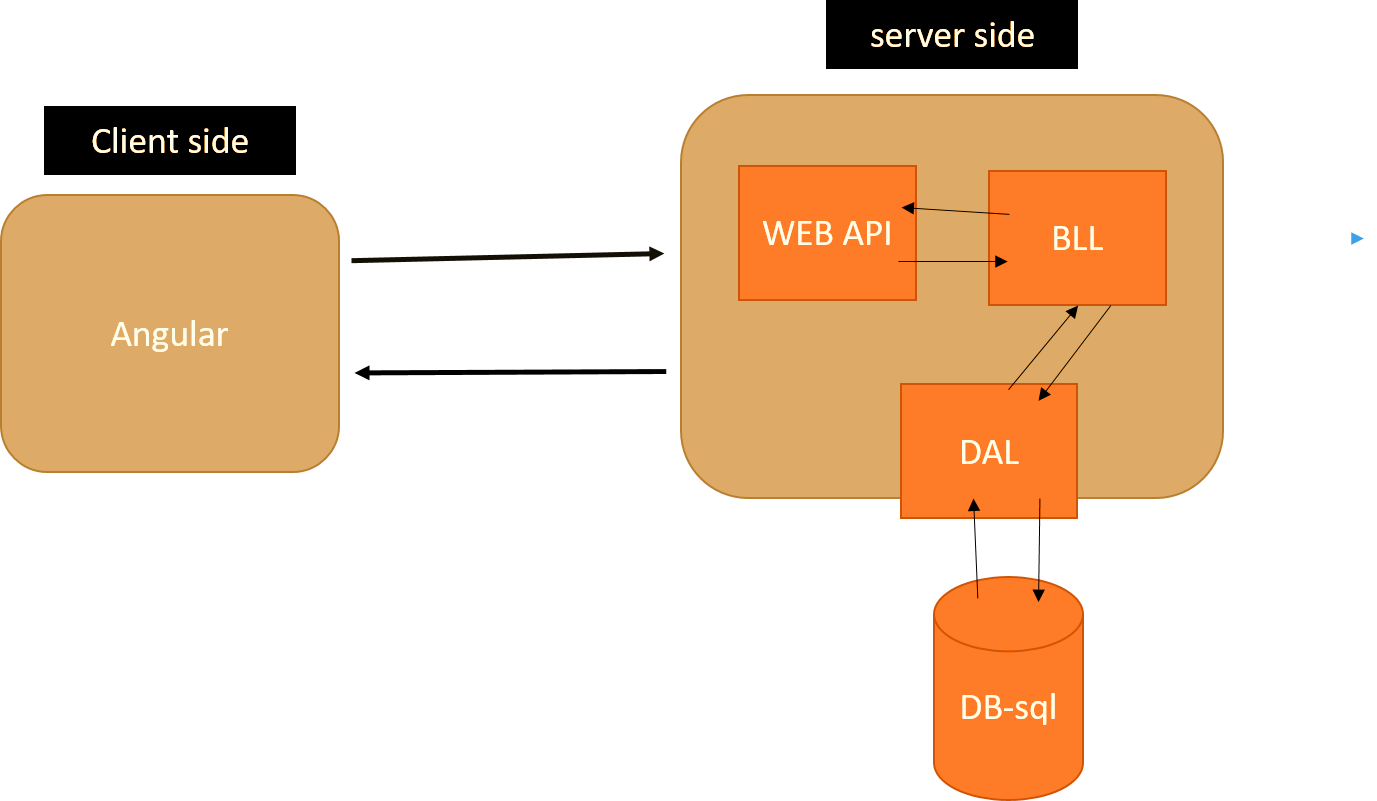
DAL – מכיל את הפונקציונאליות הנדרשת לכל התקשורת עם הData Base.

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

מטרת שכבה זו היא למנוע תלות של שכבת הBL במבנה בסיס הנתונים. שכבת הBL מכילה פונקציות המרה מטיפוס הנתונים של בסיס הנתונים לטיפוס הנתונים של שכבת הModels 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 Html, TypeScript.

בחרתי לכתוב צד לקוח ב - אנגולר שהינה שפה מתקדמת ועדכנית בעלת מאפייניAngular8 חדשניים ופונקציונאלית ביות ר.

אנגולר הינה סביבת עבודה שפותחה על ידי גוגל. מאפשרת לפתח אפליקציותFramework אינטרנט בקלות ומהירות. במקור היא באה לתת מענה לבניי תApplications 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 )סינון מהיר ופשוט לביצוע( של תוכן המתקבל מהשרת לפי מספר רב של פרמטרים.

צד שרת בחרתי לכתוב בC# .C# היא שפת תכנות עילית מרוב ת-פרדיגמות, מונחית עצמים בעיקרה המשלבת רעיונו ת כמו טיפוסיות חזקה, אימפרטיביות, הצהרתיו ת, פונקציונאליו ת פרוצדוראליות וגנריות .

C# היא שפה מעניינ ת, נוחה ומלאה פונקציונאליות למתכנת. שימוש בשפה זו נפוץ כיום , וכתוצאה מכך, ניתן היה למצוא בה קודים שונים שנדרשו לפיתוח.

בנוסף ,בחרתי להשתמש ב - EntityFramework טכנולוגית עבודה מתקדמת של מיקרוסופט.

הEntityFramework מאפשר לטעון את הנתונים מהDB -ולעשות להם השמה בצורה ישירה ואוטומטית לתוך אובייקטים בקוד הממפים את מאגר הנתונים בצורה מידית.

.Sql Server שנכתב בשפת DataBaseקורא נתונים מ ה EntityFrameworkה

למסד הנתונים של ה-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 הDB 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 ViewHistoryב- TypeScript אשר תפנה לשרת url ותתבצע בקשת services.
* השרת מקבל את הבקשה ומנווט ל Controller שנמצא בAPI.
* הController יזמן את הפונקציה ()ViewHistoryשנמצאת ב PossibleDriveBL

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

Entityframework

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

*איור:*

DB

Sql Server

Dal

Entity Framework

Bl

L

ayer

Controllers

Web Api

Client

Models

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 בי ן צד הלקוח וצד השרת.
6. .angular, TypeScript צד לקוח

## 

## ארכיטקטורת רשת

## 

## תיאור פרוטוקולי התקשורת

http – תמצאו משפט הסבר

## שרת – לקוח

צד השרת נכת בבטכנולוגית WebApi ובשפת c#.

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

## 

# ניתוח ותרשים use case של המערכת המוצעת

## 

## רשימת use case

כל משתמש שצריך להתחבר צריך להכניס שם משתמש וסיסמא

משלוחן יכול להכניס נסיעה אפשרית, לצפות בנתוני רייטינג ובהיסטורית משלוחים

לקוח יכול להגיב על משלוח וכן להכניס בקשה למשלוח

## תיאור ה-use case העיקריים של המערכת

Identifier , Name, Description , Actors, Frequency , pre-condition ,   
תחפשו כיצד מנסחים תיאור של use case.

## מבני נתונים בהם משתמשים בפרויקט

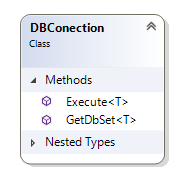
-רשימה-כדי לשלוף רשימה של ישיות שמתאימות לתנאים מסוימים לדוגמא:רשימה של משלוחינים שמתאימים לתנאים של נסיעה מסוימת.

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

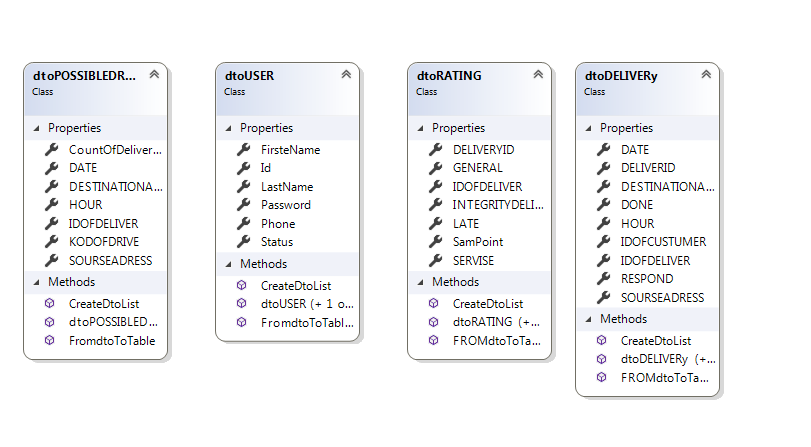
## תרשים מחלקות

לכל שכבה תיאור המחלקות שלה.

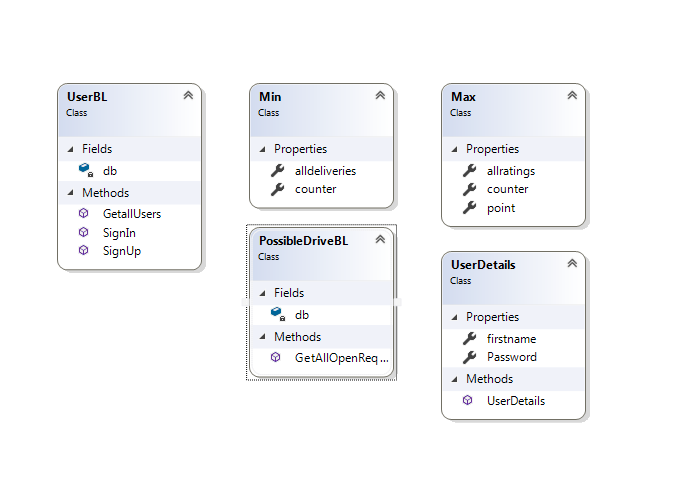
שכבת ה-DAL

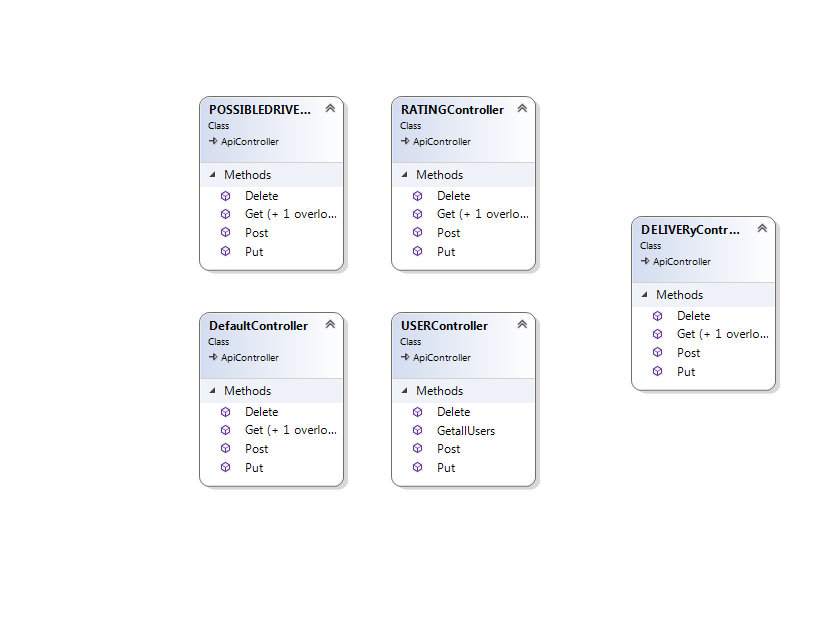


שכבת ה dto



שכבת ה-BL



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:

DBConection-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:

-CalculateDistanceBLדרך מחלקה זו התממשקתי לגוגל מפות.במחלקה ישנם מספר פונקציות של חישוב מרחקים -הפונקציות בעצם קוראות לפונקציות חיצוניות של גוגל מפות.

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

GetAllOpenRequest(dtoDELIVERy p)

Absolutefit(List<dtoPOSSIBLEDRIVE> reqest,dtoDELIVERy p)

Partialfit(List<dtoPOSSIBLEDRIVE> AllOpenRequest, dtoDELIVERy p)

Minimumnumber(List<dtoPOSSIBLEDRIVE> Partialfitlist)

calculatemaxrating(List<dtoPOSSIBLEDRIVE> mincounter)

updatematch(dtoDELIVERy p, dtoPOSSIBLEDRIVE match)

Responsetodelivery()

-Maxמחלקה שנעזר בה לצורך חישובי רייטינג

-Minמחלקה שנעזר בה לצורך חישובי רייטינג

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

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

match(List<dtoDELIVERy> AllOpenRequest, dtoPOSSIBLEDRIVE p)

checkpoint(dtoPOSSIBLEDRIVE p, dtoDELIVERy d)

Viewhistory()

Viewrating()

-RatingBLבמחלקה זו יתבצעו כל חישובי הרייטינג הפונקציות במחלקה זו הן

CalculatePoint- פונקציה שמחשבת את סך הנקודות לכל משתמש.

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

GetallUsers-שליפת כל המשתמשים

SignUp-שומרת נתונים חדשים אודות משתמש חדש בדאטה בייס

SignIn-בודקת את התחברות המשתמש למערכת(שם משתמש וסיסמא זהים)

-UserDetailsמחלקה שמסייעת בבדיקת התחברות משתמש למערכת

MainpossibledriveBL-פונקצייה ראשית שמפעילה את כל מה שקשור לצד של המשלוחנים הפונקציות הקימות במחלקה זו הם:

Main()

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

Main()

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

המחלקות בשכבה זו הם:

DELIVERyController-

POSSIBLEDRIVEController-

RATINGController-

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

חלק נוסף בצד שרת הוא ה **:DTOclass**

ה data transfere object הם אוביקטים הנושאים נתונים בין תהליכים על מנת לצמצם את מספר הקריאות לפונקציות.הם משמשות מחלקות תצוגה ומונעות כפיליות.

המחלקות בחלק זה הם:

dtoDELIVERy-מחלקתdto שאחראית על כל מה שקשור לבקשות למשלוח,הפונקציות במחלקה זו הן:

dtoDELIVERy-פעולה בונה שהופכת רשומה מסוג הטבלה לסוג תצוגה

FROMdtoToTable-הופכת רשומה ממחלקת תצוגה לרשומה מסוג הטבלה

CreateDtoList-מקבלת רשימה מסוג הטבלה והופכת אותה לרשימה מסוג תצוגה

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

dtoPOSSIBLEDRIVE-פעולה בונה שהופכת רשומה מסוג הטבלה לסוג תצוגה

FromdtoToTable-הופכת רשומה ממחלקת תצוגה לרשומה מסוג הטבלה

CreateDtoList-מקבלת רשימה מסוג הטבלה והופכת אותה לרשימה מסוג תצוגה

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

dtoRATING-פעולה בונה שהופכת רשומה מסוג הטבלה לסוג תצוגה

FromdtoToTable-הופכת רשומה ממחלקת תצוגה לרשומה מסוג הטבלה

CreateDtoList-מקבלת רשימה מסוג הטבלה והופכת אותה לרשימה מסוג תצוגה

dtoUSER-מחלקתdto שאחראית על כל מה שקשור לנתוני משתמש ,הפונקציות במחלקה זו הן:

dtoUSER-פעולה בונה שהופכת רשומה מסוג הטבלה לסוג תצוגה

FromdtoToTable-הופכת רשומה ממחלקת תצוגה לרשומה מסוג הטבלה

CreateDtoList-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 -Html, css ,typescript 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.**

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 ולכל בקשה -   
  במשלוחן המתאים – אם המשלוחן אכן אישר את הבקשה.
* (((((((טופס שינוי בקשה (שינוי מיקום) )))))))

*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 השיבוץ?

שיבוץ מספר 1 – משלוחן לבקשה:

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

שיבוץ מספר 2 – בקשות למשלוחן (לנסיעה):

**פונקציה בדיקה:**

CheckMatchBetweenDeliverAndCustumer()

קלט: משלוחן ובקשה

פלט: ניקוד רמת התאמה. //כאן יש גישה לגוגל מפס בינתיים תתעלמי מזה תגרילי מרחק.

**פונקציה הסתברות לאישור המשלוח:**

CheckRatingBetweenDeliverAndCustumer()

קלט: משלוחן ובקשה

פלט: ניקוד רמת הסתברות.

**פונקציית** **שיבוץ בקשה לנסיעה בעלת התאמה מקסימלית:**

קלט: בקשה

פלט: אין פלט

הפוקנציה 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 וששומרת 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.

# קוד האלגוריתם

public class PossibleDriveBL

{

//חיפוש כל הבקשות המתאימות שעוד לא אושרו-לנסיעה החדשה שנוספה

static DBConection db = new DBConection();

//מסננת בקשות שהותאמו ועוד לא אושרו

public static List<dtoDELIVERy> GetAllOpenReqwest(dtoPOSSIBLEDRIVE p)

{

List<dtoDELIVERy> AllOpenRequest = dtoDELIVERy.CreateDtoList((db.GetDbSet<DELIVERIES>().Where(r => r.IDOFDELIVER != null && r.DONE = false))).ToList();

return AllOpenRequest;

}

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

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 הניקוד של הנסיעה הישנה יותר קטן אז הדאטה בייס יעודכן בפרטי הנסיעה החדשה

if (prepoint > newpoint)

{

DeliversBL.updatematch(d, p);

}

}

}

//פונקצייה שמחשבת ניקוד למשלוחן בהתאם לבקשה מסוימת

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=חישוב גוגל מפות

//Destinationaddress=חישוב גוגל מפות

point = date + hour + Sourceaddress + Destinationaddress;

return point;

}

}

public class DeliversBL

{

static DBConection db = new DBConection();

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

public static List<dtoPOSSIBLEDRIVE> GetAllOpenRequest(dtoDELIVERy p)

{

List<dtoPOSSIBLEDRIVE> AllOpenRequest = dtoPOSSIBLEDRIVE.CreateDtoList((db.GetDbSet<POSSIBLEDRIVE>().ToList()));

List<NOTCONFIRM> NOTCONFIRMs =db.GetDbSet<NOTCONFIRM>();

List<dtoPOSSIBLEDRIVE> reqest=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)

reqest.Add(i);

}

return reqest;

}

//בשלב השני מתבצע חיפוש המשלוחן המתאים ביותר:

//ראשית,תתבצע בדיקת התאמה מוחלטת

public static List<dtoPOSSIBLEDRIVE> Absolutefit(List<dtoPOSSIBLEDRIVE> reqest,dtoDELIVERy p)

{

List<dtoPOSSIBLEDRIVE> AllOpenRequest=new List<dtoPOSSIBLEDRIVE>();

foreach (var item in reqest)

{

if(item.DATE==p.DATE&&item.HOUR==p.HOUR)

//if() כאן תהיה פונקצייה של גוגל מפות שתבדוק את התאמת המקומות של כתובות המקור אל מול כתובות היעד

AllOpenRequest.Add(item);

}

return AllOpenRequest;

}

//שנית תתבצע בדיקת התאמה חלקית

public static List<dtoPOSSIBLEDRIVE> Partialfit(List<dtoPOSSIBLEDRIVE> AllOpenRequest, dtoDELIVERy p)

{

List<dtoPOSSIBLEDRIVE> Partialfitlist = new List<dtoPOSSIBLEDRIVE>();

foreach(var item in AllOpenRequest)

{

//כאן תהיה פונקצצית חישוב מרחקים של גוגל מפות

// נכניס לרשימה של Partialfitlist את כל הנסיעות המתאימות עם המרחק המינימלי

}

return Partialfitlist;

}

//במידה ולאחר ההתאמה המוחלטת או החלקית נשיג כמה משלוחנים באותה רמה -נבצע בדיקת הסתברות

//חיפוש כל הנסיעות עם מספר משלוחים מינימלי

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

{

Min mincounter = new Min();

foreach (var i in Partialfitlist)

{

if (i.CountOfDeliveries < mincounter.counter)

{

mincounter = new Min();

mincounter.counter = i.CountOfDeliveries;

}

if (i.CountOfDeliveries == mincounter.counter)

{

mincounter.alldeliveries.Add(i);

}

}

return mincounter.alldeliveries;

}

//חישוב רייטינג מקסימלי מבין כל הנסיעות עם מספר המשלוחים המינימלי

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 בינהם 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 המשלוחן המתאים)

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

public static void Responsetodelivery()

{

}

}

}

class MainDelivery

{

public static void main(DELIVERIES d)

{ //הפונקציה קיבלה רשומה מסוג הטבלה והופכת לסוג תצוגה

dtoDELIVERy dtoDELIVERy = new dtoDELIVERy(d);

List<dtoPOSSIBLEDRIVE> reqest = new List<dtoPOSSIBLEDRIVE>();

//מפעילה את פונקציית הסינון של כל מי שסרב לבקשה

reqest = DeliversBL.GetAllOpenRequest(dtoDELIVERy);

//מפעילה את פונקציית בדיקת ההתאמה המוחלטת

List<dtoPOSSIBLEDRIVE> Absolutefit = DeliversBL.Absolutefit(reqest, 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 הרייטיג 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(reqest, 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 הרייטיג 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");

}

}

} }

}

class RatingBL

{

static DBConection db = new DBConection();

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.SERVISE;

point += (long)i.GENERAL;

}

foreach (var i in AllRating)

{

i.SamPoint = point;

}

foreach (var i in AllRating)

{

db.Execute<RATING>(i, DBConection.ExecuteActions.Update);

}

return point;

<<<<<<< Updated upstream

// ואז לשאול על העמודה הזאת מי המקסימום עי תור עדיפיות

=======

// איך מכניסים את שקלול הריטינג פויינט לטבלה ואז לשאול על העמודה הזאת מי המקסימום

//איך המשתמש מכניס נתוני ריטינג

>>>>>>> Stashed changes

}

}

}

public class UserBL

{

static DBConection db = new DBConection();

public static List<dtoUSER> GetallUsers()

{

List<USERS> list = db.GetDbSet<USERS>().ToList();

List<dtoUSER> dtoList = dtoUSER.CreateDtoList(list);

return dtoList;

}

public static Object SignUp(dtoUSER user)

{

USERS u = user.FromdtoToTable();

db.Execute<USERS>(u, DBConection.ExecuteActions.Insert);

return u;

}

public static Object SignIn(UserDetails ud)

{

USERS u = db.GetDbSet<USERS>().FirstOrDefault(u1 => u1.FirsteName == ud.firstname);

if (u == null)

return new { success = false, massage = "user does not exist" };

else if (!u.Password.Equals(ud.Password))

return new { success = false, massage = "Password is wrong" };

else

return new { success = true, user = new dtoUSER(u), massage = "Login success" };

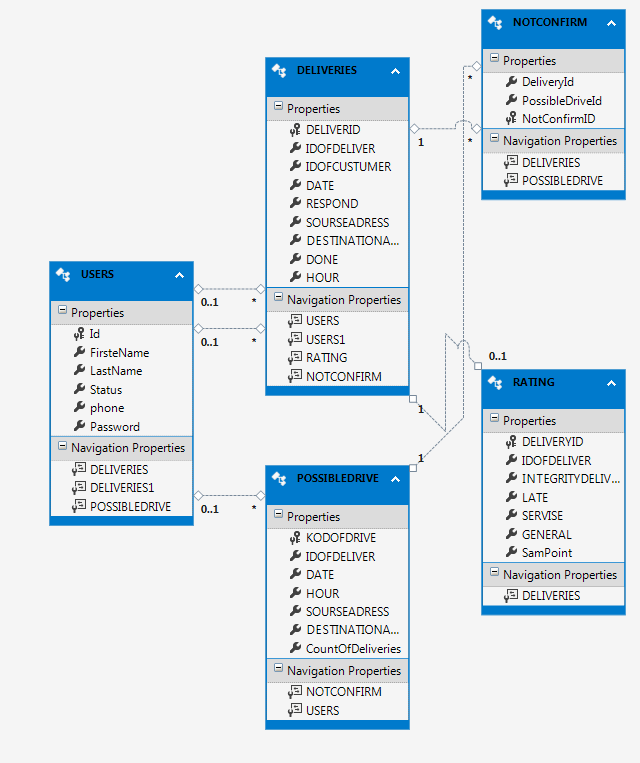
}

}

}

# 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 |
| IDOFCUSTUMER | bigint | תז לקוח | לא | FK |
| DATE | date | תאריך שליחת המשלוח | לא |  |
| RESPOND | bit | האם הגיבו על המשלוח | לא |  |
| SOURSEADRESS | varchar | כתובת מקור | לא |  |
| DESTINATIONADRESS | 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 ולצידם 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 |  |
| SOURSEADRESS | varchar | כתובת מקור | לא |  |
| DESTINATIONADRESS | varchar | כתובת יעד | לא |  |
| CountOfDeliveries | int | מספר משלוחים שקיבל | לא |  |

שם הטבלה: RATING

תפקיד הטבלה:טבלת דרוג רייטינג הלקוח יתן ניקוד מ1-הכי נמוך ל5-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 |  |

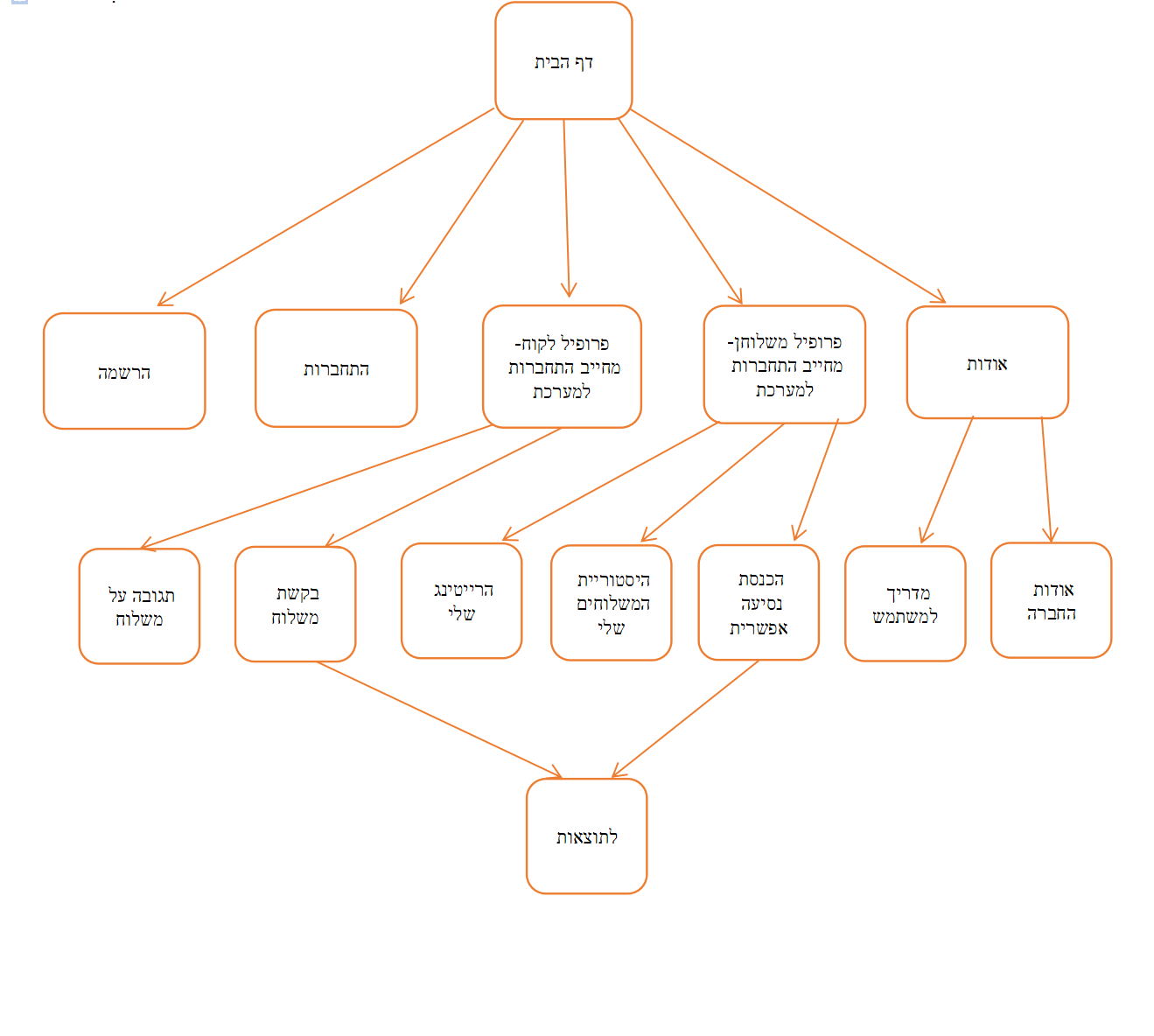
שם הטבלה: USERS

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

תפקיד הטבלה:טבלת משתמשים שישמרו בה פרטים בסיסים

# מדריך למשתמש

## תיאור המסכים



הסטורית משלוחים?

נתוני רייטינג?

רשימת משלוחים לתגובה?

## מדריך למשתמש

בכניסה לאתר יוצג דך הבית כברירת מחדל .דף הבית מכיל תפריט,האפשריות 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

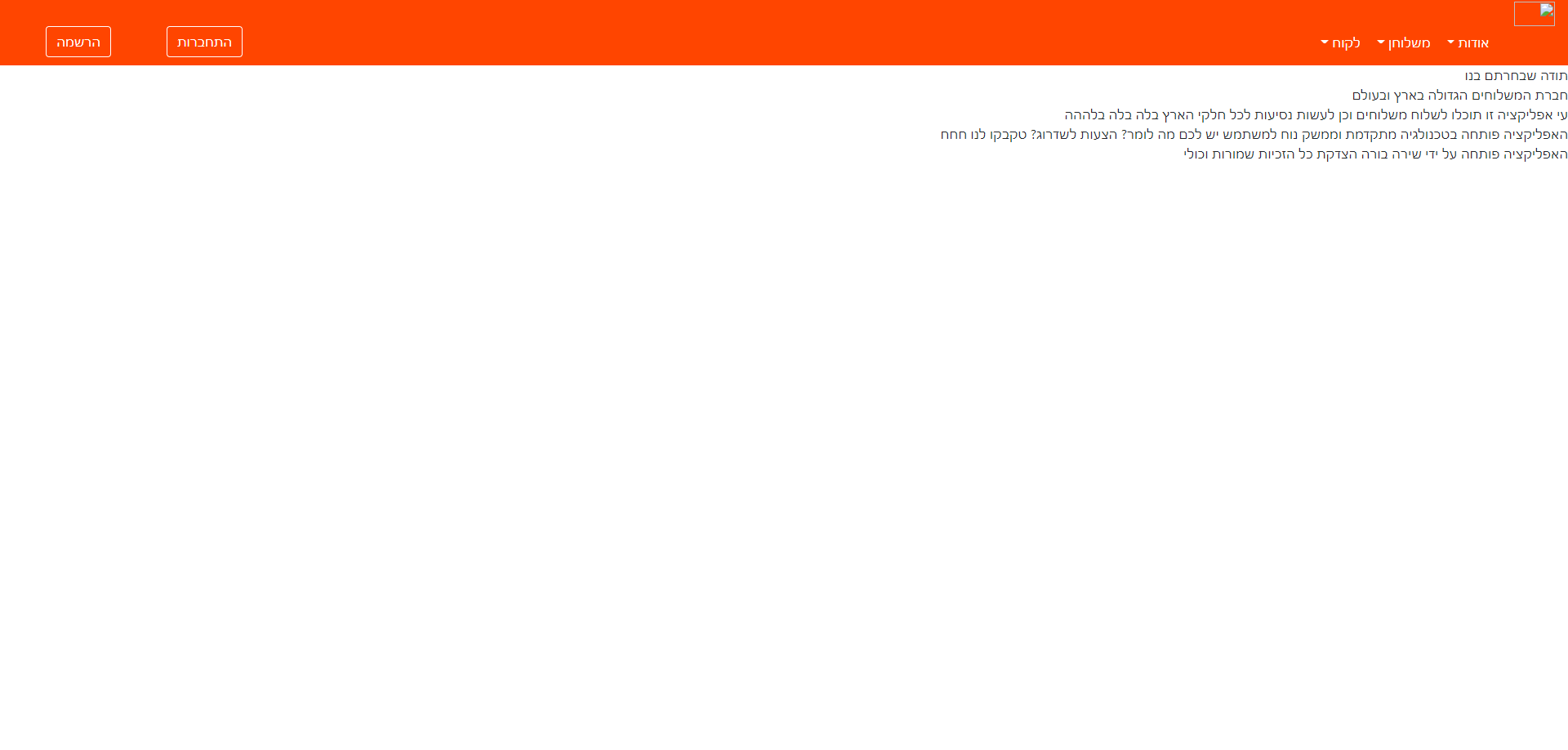
אם בחר בהכנסת משלוח הוא יקבל טופס ובו ימלא פרטים בסיסיים אודות הבקשה(כתובת מקור יעד וכו) לאחר האישור הוא ילחץ על אפשרות “לתוצאות” ויצגו בפניו תוצאות המערכת(מספר טלפון ושם של נהג שמוכן לעשות לו את הנסיעה)

-הרשמה- המשתמש יזין פרטים בסיסיים כמו :שם פרטי ,משפחה, מספר טלפון ,תעודת זהות ועוד

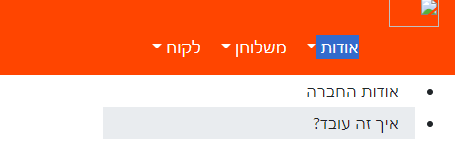
-התחברות- המשתמש יתחבר למערכת באמצעות שם המשתמש והסיסמא שלו

## צילומי מסכים

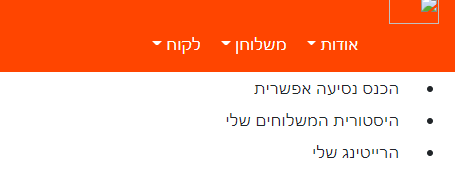
מסך הבית:



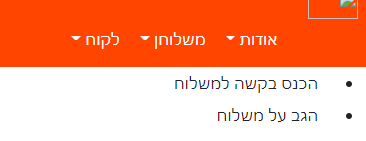
אפשרות אודות:



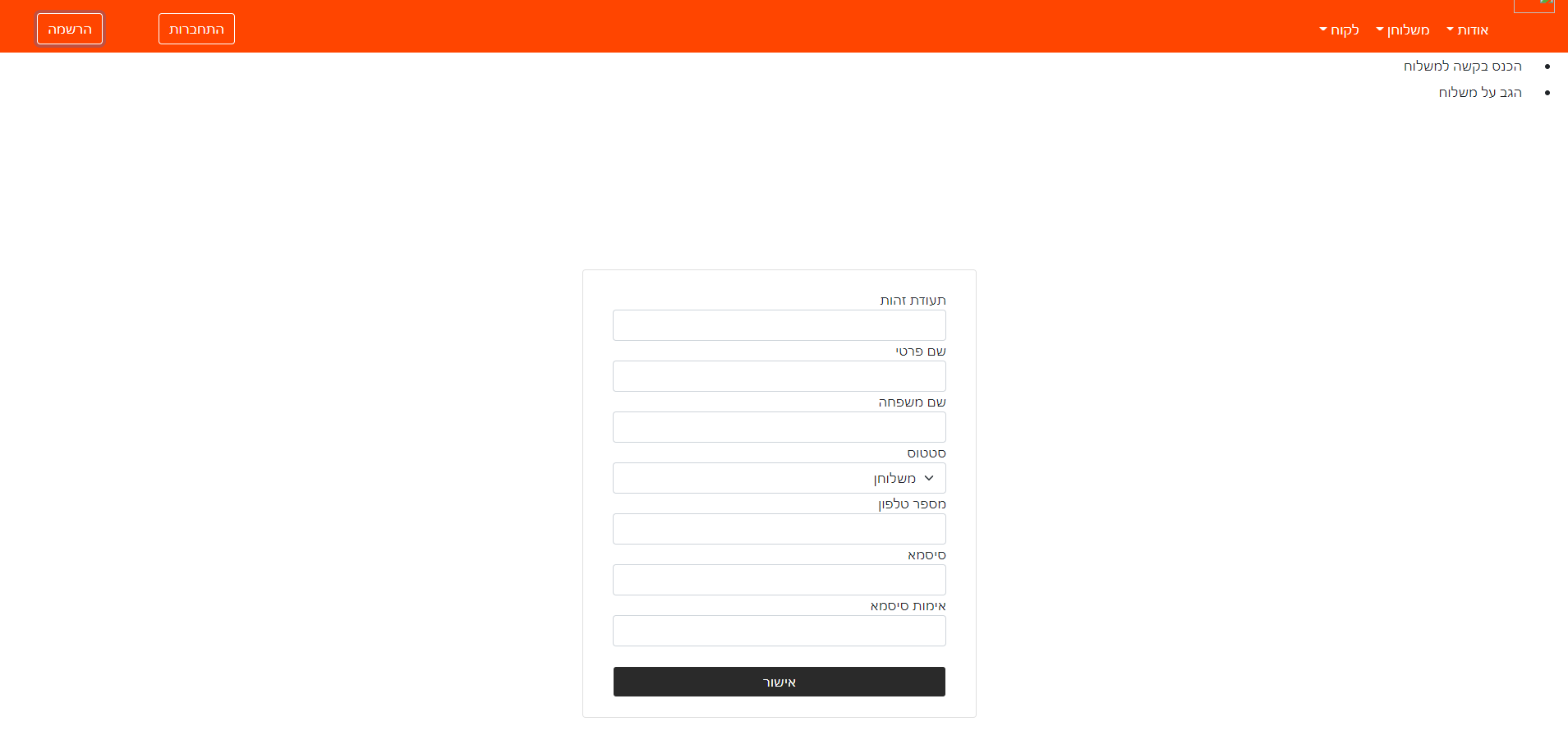
פרופיל משלוחן:



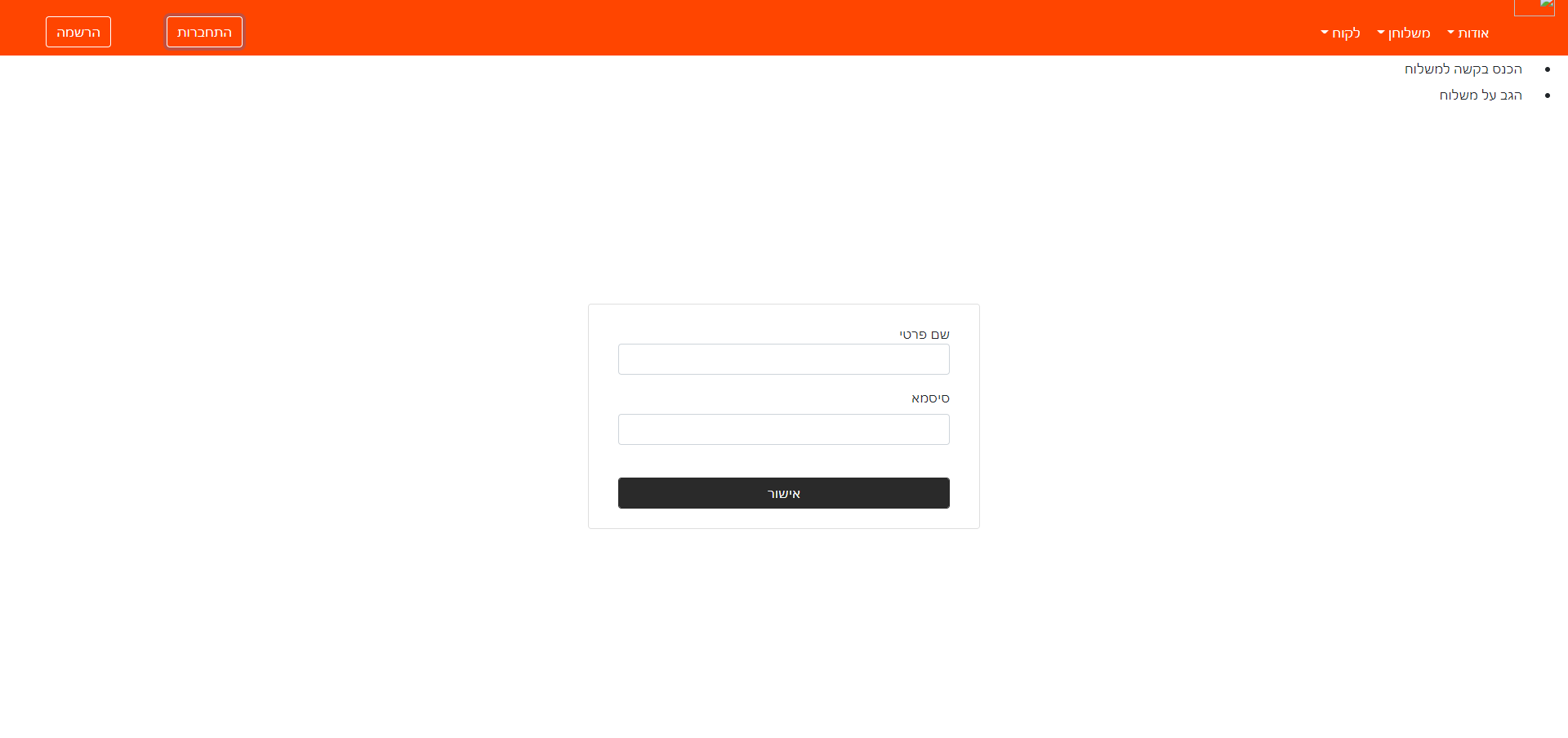
פרופיל לקוח:



הרשמה:



התחברות:



**תיאור מסכים:**

כשנכנסים בוחרים פרופיל משלוחן או לקוח

משלוחן – נכנס ורואה חבילות מתאימות

לקוח – מראה משלוחן מתאים

**תיאור פעילות האפליקציה:**

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

לכל מסך: שם המסך ותמונה

# 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

(תסבירי מה זה יעילות למה היא הייתה חשובה לך וכמה הסיבוכיות של האלגוריתם שלך)

# 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 מתשתמש וסיסמא אישיים שלו במידה ושם המשתמש או הסיסמא לא נכונים תוקפץ לו הודעת שגיאה והוא לא יוכל להכנס למערכת

# 

# מסקנות

מסקנותי בעת סיום הפרויקט :

שצברתי המון ידע 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 מהכל -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 יום של כולנו

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

יחד עם זאת יש מקום לשפר את צורת העבודה שלדעתי לא היתה יעלה ככ (כאשר היינו כל הקבוצה ביחד עם מתרגלת אחת)

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, תכנון 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. למדנתי 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. למדנתי המון על דרכי חקירה, פיתוח וכתיבה, והרבה על הסיפוק העצום שביצירה.

את הפרויקט אפיינו באופן בולט בכתיבה עצמאית. את התוצר הנפלא יצרתי בעשר אצבעות, בנחישות ובאומץ. שברתי את הראש, כמעט כפשוטו:) לכלכתי נ הרבה את הידיים בכתיבה ומחיקה וכתיבה חוזרת. נאבקתי קשות במלחמת הבהירות ונטרול השגיאות. ובעצם, פיתחי 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 הלמידה 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 שיודא שאכן המשלוח הגיע ליעדו ובשלמותו ,להבטיח שלא יהיו גנבות ועוד.

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

# ביבליוגרפיה

Stackoverflow

Angular material

Github

Bootstrap

w3schools

developer.mozilla

GeeksForGeeks