SimplyTravel

מאת : אסתר זלוצ'בסקי

מספר זהות : 207426974

ביה"ס : סמינר רכסים

מנחה : רבקה נילסון

תאריך הגשה : יולי 2021

תוכן עניינים

1.הצעת פרויקט

סמל מוסד : 340695

שם מכללה :סמינר בית יעקב רכסים.

שם הסטודנט :אסתר זלוצ'בסקי.

ת.ז. הסטודנט :207426974.

שם הפרויקט :SimplyTravel.

תיאור הפרויקט:

אפליקציה לתכנון טיול מותאם אישית. בתחילה המשתמש ימלא טופס של מספר אנשים וגילאים, רמת קושי, שעות ,מחיר ואזור מבוקש..

האפליקציה תבצע חיפוש באזור המבוקש, ותציע מספר אפשרויות למסלולים שונים (בנויים מקטגוריות שונות) , המשתמש יבחר את המסלול שמתאים להעדפותיו.

כך מתחיל להתבצע תהליך למידה של המשתמש :.

עפ"י בחירת המסלול שיתאים הן מבחינת מחיר והן מבחינת שעות של הטיול המבוקש.

הבחירות יתנו אפשרויות שונות : אטרקציות, מוזיאונים, מסלולים, קברי צדיקים ומסעדות.

בכל אחת מהאפשרויות יהיו שלוש אפשרויות, לפי בחירות הלקוח במשך נזמן המערכת "תלמד" את העדפותיו. ההצעות לטיולים יהיו בהתאמה שתגדל לפי השימוש מבחירותיו הקודמות.

המערכת תציע לו את המסלול הקצר ביותר בין האתרים שהכי מתאימים ללקוח.

תהיה אפשרות להוספת אתר ע"י המשתמשים, ההוספה תתבצע ע"י google maps, שם בנוסף לאיכון המיקום, תתבצע ההעתקה של כתובת האתר.

על מנת לתת מענה מדויק, אנו רוצים ששעות הפתיחה של האתר יותאמו לשעות הטיול, לכן עבור כל אתר נתממשק עם כתובת אתר האינטרנט לצורך הוצאת שעות הפעילות ועלות האתר.

העלות של האתר תופחת מהמחיר אותו הלקוח רוצה לשלם עבור השירות. כך שבאפשרויות שיתקבלו יתקבל מענה של ההיבט הכלכלי, המיקום , לוחות הזמנים והעדפות הלקוח.

הגדרת הבעיה האלגוריתמית:

ישנם מספר בעיות אלגוריתמיות שצריכות מענה באפליקציה זו :

1. מציאת האתרים שעונים על צורכי הלקוח.
2. מציאת המסלול הקצר ביותר בין מגוון האתרים שעונים על הקריטריונים שהלקוח רוצה , לבין במקום ממנו הלקוח מגיע.
3. מציאת המסלול הקצר ביותר בין האתר הראשון לבא אחריו וכן הלאה, על מנת לחסוך זמן של נסיעות מיותרות.
4. הוספת אתר שקיים וביצוע איכון למיקומו
5. הוצאת מידע על כתובת אתר האינטרנט של האתר בו אנו רוצים לבקר לצורך הוצאת פרטים נוספים : שעות פתיחה, מחיר, טווח גילאים- חשוב לציין שאם המשתמש שהוסיף את האתר מילא את הפרטים הנ"ל החיפוש לא יתבצע.
6. תהליך למידת לקוח הבנוי ע"י חישוב של בחירותיו מתוך כלל האפשרויות והסקת מסקנות שיובילו להצעה מדויקת יותר של טיול להבא.
7. הצגת האזור האישי של הלקוח, בו ייוצגו טיוליו הקודמים. והן הצגת הצעות טיול מותאמות אליו אישית.

רקע תיאורטי בתחום הפרויקט :

1. מציאת האתרים המתאימים תתבצע לאחר ייבוא כל הפרטים והמידע שאנו זקוקים לו, ביצענו בדיקה האם יש לנו צורך להתממשק ל google ול google maps ולהשוות את המידע על מנת לוודא שמדובר באותו אתר. אך ראינו שבאתר google maps יש את כתובת אתר האינטרנט של האתרים. נוכל להוסיף את כתובת האתר ל DB , ולבצע את הבדיקה באתר על שעות פתיחה , מחיר ואם יש גם טווח גילאים.
2. בתחילה, אנו צריכים למצוא את האתר שבאזור המבוקש שהכי קרוב לנקודת ההתחלה של הטיול. נשתמש באלגוריתם דייקסטרה על מנת לבחור מאיזה אתר הכי טוב להתחיל.
3. הבעיה שצריכה התייחסות בסעיף זה, היא מעבר בין מספר אתרים, הפתרון האופטימלי (לא מבחינת יעילות , אך כן מבחינת רעיון) הינו אלגוריתם הסוכן הנוסע. הסיבה לכך שאנו רוצים לעבור במספר אתרים בכל מסלול. מכיוון שהיעילות הינה n! , החלטנו להשתמש באלגוריתם דייקסטרה בצורה חמדנית , כלומר, בכל פעם לתת את השכן הקרוב ביותר. וכך נבקר באתרים שאנו רוצים. כמובן שנדאג שמכל קטגוריה תתבצע בחירה אחת. והן שיוצגו אתרים שונים בכל אחת מהאפשרויות .
4. לצורך הוספת אתר , תתבצע התממשקות ל google maps, יועתק איכון המיקום בצורת קורדינציה . והן כתובת האתר.
5. תתבצע התממשקות עם כתובת האתר שהוכנסה ל DB, ותתבצע שליפה של הנתונים הרצויים.
6. עבור כל לקוח תתבצע שמירה של האתרים אותם הוא בחר, יתקבל משקל שונה עבור המיקום שלהם בטיול : אם בחר ראשון : קטגוריה *3 , שני : קטגוריה*2 וכן הלאה.
7. יתבצע לימוד עפ"י ניתוח סטטיסטי של בחירותיו, ותהיה התחשבות בבחירותיו הקודמות בהצעות הבאות.

תהליכים עיקריים בפרוייקט :

1. הלקוח יוריד את האפליקציה.
2. הלקוח יוכל לבחור בין חיפוש רגיל לפי מרחק לבין חיפוש מותאם אישית.

3. הלקוח יבצע הצטרפות. התחברות למערכת.
4. הלקוח ימלא טופס של פרטים שיעזרו בתכנון מותאם אליו.
5. יוצגו ללקוח חמש אפשרויות למסלולים שעונים על ההגדרות שלו.
6. תתבצע בחירה של טיול מועדף.
7. באיזור האישי יוצגו הטיולים הקודמים של הלקוח.
8. תתבצע למידה של האלגוריתם עפ"י בחירות הלקוח.
9. תהיה אפשרות של הוספת אתר: הן מיקומו והן כל הפרטים הנצרכים.

תיאור הטכנולוגיה :

הפרויקט מחולק לשני חלקים: צד שרת וצד לקוח.

צד השרת נכתב בשפת #C וצד הלקוח נכתב בשפות Css ,Html, TypeScript

ובשימוש בטכנולוגיית Angular וממשקי תכנות יישומים של .Chrome

התוכנה מחולקת למספר שכבות:

- שכבת מסד הנתונים ()Server SQL
- שכבת הישויות ( ) Entity Frameworkוהגישה לבסיס הנתונים- ()Dal
- שכבת הקוד ()BL

שכבת הקוד נעזרת בספרית ה- Common שמכילה את מבנה הנתונים כדי

לתקשר עם שכבת ה-Dal וה-.webApi

- שכבת הקישור ()WebAPI
- ממשק משתמש UI(– angular)

בשיטה זו קיימת הפרדת רשויות מוחלטת וכל שכבה עומדת באופן עצמאי

לחלוטין ומתקשרת רק עם השכבה הסמוכה לה .

החלוקה לשכבות נועדה להפריד באופן מוחלט בין הלוגיקה של הפרויקט לבין

הנתונים עצמם. כך ששינוי בשכבה אחת לא יגרור אחריו צורך בשינויים נוספים

בשכבות האחרות.

צד שרת

פרטי לקוח, העדפותיו, אתרים שביקר (על מנת לא להציע אותם שוב), אתרים מסוגים שונים וכל הפרטים, מיקומי האתרים. פרופיל מותאם אישית ללקוח שנבנה עפ"י ניתוח בחירותיו :

.

שפת תכנות בצד השרת :#C, web API.

צד לקוח : יוצגו למשתמש טיוליו הקודמים, תהיה אפשרות להוספת אתר חדש, חיפוש מסלול. והן יוצגו הצעות לאתרים שמתאימים לפרופיל משתמש של הלקוח.

שפת תכנות בצד הלקוח: - html ו- TypeScript בטכנולוגית angular

מסד נתונים : SQL SERVER.

פרוטוקולי תקשורת :

לוחות זמנים :*סיום DB :01/11/2020.

*חיבור ל- entity frame work :01/12/2020.

*בניית מודל שלושת השכבות :01/12/2020.

*סיום שכבת ה-DAL : 01/12/2020.

*סיום שכבת ה-BL : 01/02/2021.

*חיבור לאנגולר ויצירת ה-GUI :01/04/2021.

*בדיקות :01/05/2021.

חתימת הסטודנט :

חתימת רכז המגמה :

אישור משרד החינוך :

2.מבוא/תקציר

הפרויקט עוסק בתכנון טיולים ללקוח בהתאמה אישית.

האתר שאותו אני מפתחת הוא אתר חברתי חינמי המציע אופציות טיולים ללקוח ובכך מאפשר לו לתכן את טיוליו בצורה הטובה ביותר. האתר מורכב מאזור אישי וכן אזור כללי. השימוש באתר לא מותנה בהרשמות ואפשר להשתמש בשירותיו בלי להכנס לאזור האישי –דרך האזור הכללי.

הרשמות לאתר מקנה ללקוח פלוסים רבים. באזור האישי נשמרים טיוליו הקודמים של הלקוח וכן ישנה האפשרות לשמור את תמונותיו ורשמיו מהטיולים הקודמים שערך וכן את רשימת התזכורות שהכניס. המידע משמש לצורך שקלול דירוגיו מהטיולים הקודמים לתכנון הטיולים הבאים.

מלבד האזור האישי יש את האזור הכללי שם ישנן מספר אופציות:

- חיפוש-המשתמש מכניס פרטים מסוימים כמו אזור בארץ, גילאים ואפשרות הגעה (מכונית/תחבורה ציבורית). התוכנה מעבדת את הנתונים ומחפשת אחר 5 אופציות של טיולים המושלמים ללקוח ומתאימים לפרטיו שהכנס. כל אופציה של טיול שייכת לסגנון שונה, לדוגמא: טיול רגוע/טיול בטבע וכו...וכך כל אופציה מורכבת מכמה אתרי טיולים המתאימים לסגנון המסוים הזה.
- הוספת אתר: כל משתמש יכול להכניס אתר חדש למאגר ע"י מילוי הכתובת במפה וכן הכנסת פרטים בסיסיים עליו. כמובן שישנה אפשרות לעדכן את פרטי האתרים כמו שעות פתיחה, משך זמן, סוג וכו' .

2.1הרקע לפרויקט

תדיר בבואי לארגן טיול המורכב מאתרים שונים נוכחתי לראות עד כמה הדבר לא פשוט. נתחיל מהנקודה בה צריך לחפש רעיונות לטיול. בתור בחורה שלא תרה עדין את הארץ הדבר קשה מאד ותמיד מותיר אותי בהרגשה חמוצה של "אולי יכולתי לחפש ואף למצוא יותר?". בנוסף, לאחר שישנן רעיונות צריך לחפש ולבדוק האם הם מתאימים מבחינת מספר אנשים, גילאים וכן מיקום בארץ.

נקודה בעייתית נוספת היא הנסיעות. קשה מאד למצוא אטרקציות ומסלולים ולדאוג שיהיו בקרבה אחד לשני דבר שחשוב מאד שכן אף אחד לא חפץ לבזבז את זמן הטיול היקר בנסיעות הלוך ושוב. בקיצור תמיד חלמתי על גמד שיעשה לי בלילה את העבודה השחורה וישאיר לי בבוקר לעשות את השורה התחתונה: בחירת טיול מתוך כמה אופציות וארגון החפצים (אומנם יש אתרים שונים הנותנים מידע על אתרי תיירות שונים, אך לא נמצא אתר שיאגד את המידע אל כל האתרים ביחד וידאג לחפש ולתאם בין הקריטריונים

לתוצאות.) ועל כן בבואי לבחור רעיון לפרויקט גמר ישר צץ לי הרעיון בראש. אני אצור אתר שיעשה את כל העבודה בחיפוש הטיול ויגיש את התוצאות למשתמש על מגש של כסף.

בתחילה הדבר היה נראה לי כהר עצום בלתי עביר. אך לאט לאט פרקתי את הפרויקט למשימות, התוודעתי לספריות שונות שיעזרו לי בתכנות (כמו גוגל מפס)והתקדמתי ליעד הנכסף.

אני שמחה להציג את האתר המושלם שיעזור לאלפי משתמשים לתכנן את טיוליהם ולשמור להם את החוויות בצורה הטובה ביותר.

בתחילה רציתי להשתמש באלגוריתם דייקסטרה לצורך בחירת האתרים עם המרחק הקצר ביותר בינהם אך ראיתי שאלגוריתם זה לא רלוונטי עבורי שכן אני רוצה את המרחק לפי מפת הכבישים העדכנית ולא מרחק אוירי וכן זמן הריצה שלו גבוה מידי ואני רוצה שהפרויקט יתן מענה לזמן תגובה מהיר למשתמש. על כן מצאתי את השימוש בגוגל מפס כמתאים ביותר עבורי.

דבר נוסף שהחלפתי במהלך עבודתי על הפרויקט הינו צורת התוצאות ללקוח: בתחילה חפצתי להציג ללקוח אופציית טיול בודדת המתאימה לקריטורינו ומיקומו. אך בהמשך חשבתי שמספר אופציות מגוונות יתנו מענה טוב הרבה יותר.

עוד דבר בו חשבתי להשתמש אך הבנתי שהוא לא מתאים זה האלגוריתם ההונגרי לצורך שיבוץ המסלולים בטיולים.

2.2תהליך המחקר

בכדי לממש את האלגוריתם למציאת אתרי טיולים השומרים על מרחק מינימלי בינהם חשבתי בתחילה על אלגוריתם דייקסטרה-אלגוריתם זה מוצא את המרחק הקצר ביותר בין כל האתרים ועונה למה שאני מחפשת אך זמן הריצה שלו הוא גבוה (תלוי במבני הנתונים בו נשתמש אבל בכל מקרה זמן הריצה יהיה גבוה)לצורך כך ניתן להשתמש בדייקסטרה חמדני שמקטין את זמן הריצה אך נותן תוצאה שלא מתאימה לצרכי האתר. ישנה עוד בעיה לגבי השימוש בגרפים לצורך חישוב המרחק הקצר ביותר והיא העניין של המשקל-איזה משקל אקבע לכל קשת בגרף? המשקל בעצם מציין את המרחק בין האתרים ולמדוד זאת לפי מרחק אווירי זה לא נכון שכן צריך להתחשב במפת הכבישים. ולכן פניתי לכיוון השימוש ב google maps –נושא זה חקרתי היטב והשתמשתי בו כמה פעמים בפרויקט.(בהוספת אתר, בחישוב המרחק הקטן ביותר בין האתרים וכו..)חוץ מהענין של המרחקים יש את האלגוריתם של תכנון הטיול אותו תכנתתי בעצמי כדי לעשותו הכי טוב שאפשר(מפורט בהרחבה בסעיפים הבאים).

2.3סקירת ספרות

האתרים בהם נעזרתי לכתיבת הפרויקט הן ברקע התאורטי של הנושא והן בנושאים

המקצועיים והתכנותיים:

ברקע התאורטי:

http://he.wikipedia.org

https://www.geeksforgeeks.org/how-to-add-google-locations-autocomplete-to-your-angular-application/

בכתיבת האלגוריתם:

http://stackoverflow.com

http://docs.microsoft.com

http://www.github.com

http://www.google.co.il/maps/

https://cloud.google.com/

3.מטרות הפרויקט

המטרה העיקרית שעמדה מול עיני היתה רכישת הידע והנסיון ולמידת הטכנולוגיות והספריות השונות:
Google Maps , Web Api , Angular, Angular Material, Github, Entity Framework111111וכו...

מטרת העל בפרויקט:

לסייע לאנשים לתכן את טיוליהם בצורה הקלה ביותר תוך חיסכון בזמן ובמשאבים עם תוצאות הטובות ביותר.

מטרות נוספות:

- לספק תצוגת נוחה ונעימה לעין של אופציות של טיולים מסוגים שונים לבחירה תוך מתן מענה לקריטריונים שהציג הלקוח.
- לדאוג למאגר אתרי טיולים ואטרקציות שונים ומגוונים, מעודכנים ואמינים.
- לספק ענן לשמירת פרטי הלקוח והיסטוריה.

3.1יעדים

- המערכת תקבל מהמשתמש פרטים ותציג לפניו אטרקציות העומדות בקריטריונים.
- המערכת תציג בעזרת האלגוריתם מספר אופציות של טיולים המורכבים מאתרים מסוגים שונים המשולבים ביניהם בהתאם לסוג הטיול.
- המערכת תבחר בכל פעם את האתר המתאים לדרישות הלקוח ושהמרחק בניו לבין האתר הקודם הוא מינימלי.
- המערכת אף תחשיב את תגובות הלקוח על טיוליו הקודמים ותיקח אותם בחשבון בבחירת טיול נוסף.

- המערכת תאפשר להגדיל את מאגר האתרים ע"י הוספת אתרים חדשים.
- המערכת תאפשר עדכון אתרים.

4.אתגרים

האתגרים שעמדו בפני בעת העבודה על הפרויקט היו רבים מאד אך בסופו של דבר הם הקנו לי נסיון רב, מיומניות, ידע והצלחה.

- לימוד חומר חדש.
- למידת החיפושים באינטרנט.
- חשיבה ובניית האלגוריתם שלי .
- קושי בהתמקדות בדברים מסוימים מתוך שלל האפשרויות.
- פתירת באגים.
- לימוד והבנת קודים קיימים ושימוש בהם.

5.מדדי הצלחה

- הצגת תוצאות התואמות לפרטים שהכניס הלקוח ולרשמיו מטיוליו הקודמים וכן תוצאות המשלבות בתוכם אתרים המתאימים לסגנון הטיול ושומרים על מרחק קצר יחסית ביניהם.
- הכנסת האתרים בצורה נכונה תוך שמירת מיקומם הנכון (full name ו-coordination) ב-DB.
- כניסה מאובטחת לאזור האישי.
- הוספת רשמים ותגובות לטיולים בצורה נכונה.

6.רקע תאורטי

הבעיה האלגוריתמית איתה התמודדתי בפרויקט היא הצורך להציע ללקוח הצעות לטיולים המורכבים מאתרים שונים המתאימים לקריטריונים שהכניס וכן ששומרים על מרחק מינימלי ביניהם. הפתרון צריך לכלול: הצגת תוצאות המכילות טיולים טובים תוך שמירה על קריטריוני הלקוח בזמן הקצר ביותר וכן הצעת מספר תוצאות שונות למשתמש. פתרון בעיה זו יכול להיעשות בעזרת כמה אופציות של אלגוריתמים ישנה את האפשרות לפתור זאת ע"י אלגוריתם שיבוץ כלשהוא וכן ע"י אלגוריתם של קריטריונים.

7.מצב קיים

את בעיית סינון האתרים לפי אילוצים קיימים פתרונות שיבוץ רבים כגון אלגוריתם 8 המלכות : אלגוריתם חידת שמונה המלכות:

חידת שמונה המלכות היא חידת שחמט שבה יש למקם שמונה מלכות שחמט על לוח שחמט כך שאף אחת מהן לא מאיימת על אף אחת מחברותיה .החידה היא מקרה פרטי של בעיית , המלכות n בעיה דומה בה יש להציב מלכות על לוח בגודל n x n .

חידת שמונה המלכות משמשת כדוגמה נפוצה לבעיה הניתנת לפתרון בעזרת מחשב ,בטכניקה "כוח גס "כלומר בדיקת כל המצבים האפשריים ,ומציאת אלה מתוכם העונים על תנאי .

המשבצות שעל הלוח64 משבצות מתוך 8לשם מיקום המלכות עלינו לבחור ,כשאין חשיבות לסדר הבחירה .יוצא שנבחר יותר מארבעה מיליארד אפשרויות .מספר רב מדי של מצבים לבדיקה בידי אדם ,אך מספר סביר בהחלט לבדיקה באמצעות מחשב. משום שניסוח החידה פשוט ניתן לתאר את האלגוריתם בקווים כלליים באופן הבא) והציבו מלכה 8 (:התחילו מהשורה העליונה בתחילת השורה (א) .עברו לשורה הבאה ,התחילו מתחילת השורה והתקדמו לסופה עד שתמצאו את המקום הראשון שאינו מאוים על ידי המלכה הקודמת .המשיכו כך שורה אחרי שורה ,בכל אחת ,מצאו את המקום הראשון בשורה שלא מאוים על ידי המלכות מהשורות הקודמות .אם הגעתם לשורה שבה לא קיים מקום לא מאוים ,חזרו לשורה הקודמת והציבו את המלכה במקום הבא שאינו מאוים באותה השורה .יש להמשיך בביצוע האלגוריתם (להתקדם ולסגת במידת הצורך) עד למילוי כל השורות..

8.ניתוח חלופות מערכתי

הפתרון של חידת 8 המלכות אומנם פותרת את בעיית האילוצים אך יש לה 2 חסרונות :

1.נותנת פתרונות המורכבים מתבנית בודדת של סידור האתרים.

2.דורשת מהמשתמש להכניס המון פרטים דבר המעייף ומתיש את המשתמש.

9.תיאור החלופה הנבחרת

החלטתי לכתוב אלגוריתם שלא עובד על הרעיון של האילוצים מצד אחד ומצד שני הוא נותן מענה של הצגת האופציות לטיול לפי קריטריוני המשתמש.

10.אפיון המערכת

10.1סביבת פיתוח

חומרה : מעבדi5 RAM 12GB

עמדת פיתוח : מחשב Intel

מערכת הפעלה10 windows

שפות תוכנה : C# תוך שימוש בטכנולוגית webApi, Angular.

כלי תוכנה לפיתוח המערכת: Microsoft ,visual studio 2019,vs code

מסד נתונים : SqlServer.

עמדת משתמש מינימלית :

- חומרה : מעבד i3 4GB RAM.
- מערכת הפעלה : windows 10 ומעלה.
- חיבור לרשת :נדרש.
- תוכנות :chrome.

10.2תיחום המערכת

נושאים באחריות המערכת :

- המערכת תבנה מסלולים מהאטרקציות הנמצאות במאגר המערכת או שנוספו ע"י המשתמשים האחרים.
- המערכת מחפשת את השילובים הטובים ביותר על מנת להציע אותם למטייל.
- המערכת תאפשר הכנסת אתרים חדשים וכן עדכון אתרים קיימים.
- המערכת תשמור עבור הלקוח את ההסטוריה שלו : טיוליו, תגובותיו, ותמונותיו.

נושאים שאינם באחריות המערכת :

- עוסקת בדרכי הגעה לאזור הטיול (תחבורה ציבורית / דרכי גישה).
- המערכת מציגה מידע לגבי שעות פתיחה אך אינה מוודאת מול מנהלי האתרים השונים את רמת העדכון של לוח הזמנים המוצג.
- המערכת לא אחראית לבדיקת תקינותם של הפרטים הנוספים ע"י המשתמשים.

10.3מודול המערכת

- הזנת פרטי המשתמשים .
- אופציונלי : העלאת קבצים של המשתמש לאזורו האישי.
- תכנון הטיול : בחירת מיקום הטיול, מספר שעות וגילאים.
- מציאת תוצאות הטיולים ע"י האלגוריתם.
- צפיה בפרטי הטיולים.
- בחירת טיול מסוים מתוך 5 האופציות ע"י המשתמש.
- הזנת פרטי הטיול לדטה בייס.
- צפיה באופציות נוספות.

10.4אפיון פונקציונלי

CreateTravels-הפונקציה מאתחלת נתונים עבור האלגוריתם.

SitesRandom -הפונקציה המרכזית של האלגוריתם-צירוף האתרים הנבחרים לתוצאות.

ChooseTheShortDistance-הפונקציה מחזירה את המרחק הקצר ביותר בין האתר הנוכחי לאתר הבא.

GetDistance-הפונקציה מחזירה את המרחק בין 2 אתרים.

BuildUrlForLocationId-הפונקציה בונה קורדינציה לאתר הטיול.

BuildUrlForDistance -הפונקציה בונה url ל-Api של GoogleMaps שמחזיר את המרחק בין כל הכתובת לכתובת המקור.

AddSite-הפונקציה מוספיתה אתר חדש שכתובתו מתקבלת כקלט ועוברת בדיקת נכונות.

SignUp/SignIn-התחברות/הרשמות לאתר.

10.5ביצועים עיקריים:

<u>האלגוריתם המרכזי</u>

הבעיה האלגוריתמית איתה התמודדתי בפרויקט היא הצורך להציע ללקוח הצעות לטיולים המורכבים מאתרים שונים המתאימים לקריטריונים שהכניס וכן ששומרים על מרחק מינימלי ביניהם.

פתרתי בעיה זאת ע"י אלגוריתם שכתבתי שאחראי על יצירת הרכב טיולים בהתאמה ללקוח.

האלגוריתם עובד בצורה כזאת:

בתחילה נתונים 2 משתנים:1.מערך שמות סגנונות הטיולים שהם טבע, רגוע, קלאסי, היסטורי ואתגרי.

2.מטריצת אחוזים בה מחושבים כמה זמן מכל אתר יהיה לכל סגנון טיול.

המטריצה:

	בתי קפה	מסעדות	בתי קברות	מסלולים יבשים	מסלולים רטובים	אטרקציות יבשות	אטרקציות רטובות	שמורות טבע	מוזאונים
קלאסי	5%	5%	18%	9%	9%	9%	9%	18%	18%

טבע	0%	15%	1%	22%	22%	0%	0%	40%	0%
הסטוריה	0%	10%	17%	0%	0%	0%	0%	28%	45%
אתגרי	0%	15%	0%	17.5% רק מראת* אותך בבוקר	17.5% רק מראת* אותך בבוקר	25%	25%	0%	0%
רגוע	10%	10%	10%	15% רק מראת* אותך נאוכל	15% רק מראת* אותך נאוכל	0%	0%	20%	20%

הלקוח מכניס את פרטיו הבסיסיים שהם : גיל המשתתפים בטיול, דרכי ההגעה (רכב/אוטובוס), חצי יום/יום שלם ואזור רצוי בארץ.(במידה ולא מוכנסים פרטים מסוימים, הטיולים יכללו תוצאות של כל האתרים ללא סינון הפרט שלא הוכנס).

האלגוריתם מסנן קודם כל סינון ראשוני של הפרטים שהוכנסו.

אח"כ האלגוריתם מחשב את הזמן הממוצע של שהיה באתר(לצורך ההמשך).

וכן הוא מחשב את הכמות אתרים הממוצעת לכל סגנון טיול מכל אתר(לפי הממוצע זמן שהייה באתר ומטריצת האחוזים).

האלגוריתם מכין את התוצאות בצורה של בחירת אתר מהסוג הנצרך שמרחקו מהאתר הקודם היה הקצר ביותר שזה נעשה בעזרת התממשקות לגוגל מפס לצורך בדיקת המרחק הקצר לפי מפות הכבישים העדכניות והוספתו לתוצאות. (האתר הראשון יבחר לפי המרחק הקצר ביותר מהבית של המשתמש).

10.6אילוצים

המערכת מסתמכת על נכונות הנתונים שהוכנסו ע"י המשתמשים.

11.תיאור האריטקטורה

11.1הארכיטקטורה של הפתרון המוצע בפורמט של :Top-Down level Design:

הפרויקט מחולק לשכבות :

הפרויקט מחולק ל 2 חלקים :

- צד שרת :הנכתב בשפת #c. בטכנולוגיית WebApi.
- צד לקוח :הנכתב בשפת Angular ובטכנולוגית Html, Type Script.

תיאור צד השרת :

צד השרת מחולק כמקובל לשכבות :

שכבת ה - DAL

שכבת ה - Models

שכבת ה - BL

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.

פירוט :

שכבת ה DAL : היא השכבה דרכה ניגשים לנתונים היושבים ב DB היא מכילה מחלקות המייצגות את בסיס הנתונים וכן פונקציות נחוצות לתפעולו . פעולות ההתקשרות עם בסיס הנתונים נעשו בטכנולוגית Entity framework.טכנולוגיה זו היא המקובלת לטיפול בבסיס הנתונים כאשר היא מנתחת את בסיס הנתונים ,בונה מחלקות לייצוג הטבלאות ומספקת רשימות מלאות בנתוני בסיס הנתונים.

שכבת הBL : שכבה שבה כתובה כל הלוגיקה של הפרויקט.

שכבת הModels : שכבה זו 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 הBL מכילה פונקציות המרה מטיפוס הנתונים של בסיס הנתונים לטיפוס הנתונים של שכבת ה-Models

ולהיפך,וכך מיוצגים הנתונים בכל הפרויקט.

בנוסף קימות מתודות השרת המחצינות את הפעולות שניתן לבצע בשרת .מתודות אלו משתמשות ב BL ומופעלות ע"י שכבת ה GUI בצד הלקוח. מודל זה אמנם גורם טרחה טכנית לא מעטה בכתיבת הקוד ובתכנון הפונקציות אבל מספק קוד נקי, קל להבנה ונוח לשינויים ולשדרוגים. בפרויקט הושקעה עבודה רבה בכתיבה נכונה ויעילה כמקובל בעולם התכנות ,דבר שאמנם דרש זמן רב למדי אך הקנה לנו ניסיון אמתי מעין כמוהו.

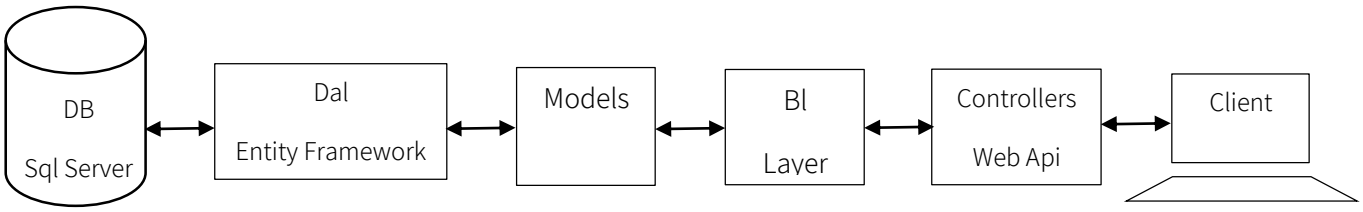
<u>דוגמא לזרימת מידע במערכת</u>

שליפת כל האטרקציות.

ברצוננו לקבל את כל הטיולים של לקוח מסויים מה -DB ולכן יתבצעו השלבים הנ"ל :

- המשתמש יחפוץ לראות את כל הטיולים שלו.הוא ילחץ על הכפתור ובקשתו תפנה ל- Type script.
- ב- Type script תתבצע קריאה לפונקציה () GetTripsToUser אשר תפנה ל- Service.
- ב-Service תתבצע בקשת url לשרת.
- השרת מקבל את הבקשה ומנווט ל -Controller שנמצא ב- SimplyTravelAPI.
- ה Controller יזמן את הפונקציה ()GetTripsToUser שנמצאת ב-TripsBl.
- ה-BL מעוניין לקבל נתונים מהDB ולכן הוא פונה ל-DAL(דרך ה-Entity framework).
- ה-DAL שואב את הנתונים הרצויים ממסד הנתונים וכעת מתבצע שלב החזרה.
- ה DAL מחזיר את הרשימה של הטיולים לשכבת ה bl בה מתבצעת פונקציה הסינון(Linq)של הבאת הטיולים של לקוח מסויים.
- הפונקציה ()GetTripsToUser מחזירה את הנתונים לController.
- מה- Controllerהנתונים מוזרקים ל-Service.
- מה-Service חוזרת הרשימה לType script.
- הרשימה מוצגת ב-html.

11.2תיאור הרכיבים בפיתרון



1. מסד הנתונים הבנוי מטבלאות וקשרי גומלין בינהם.

2.שכבת הגישה לנתונים באמצעות Entity Framework.

3.שכבת הישויות.

4.שכבת ה -bl בה כתובים האלגוריתים.

5.Web Api-פרוטוקול התקשורת בין צד הלקוח וצד השרת.

6.צד לקוח Angular, Type Script.

11.3ארכיטקטורת רשת

לא רלוונטי

11.4תיאור פרוטוקולי התקשורת

https

11.5שרת-לקוח

צד השרת נכתב בטכנולוגית Web Apiובשפת C#.

צד הלקוח נכתב בשפות html, Css ו– Type script בטכנולוגית Angular.

11.6תיאור הצפנות

לא רלוונטי

12.ניתוח ותרשים UML / Use cases של המערכת המוצעת

12.1רשימת ה- Use cases

- העלאת תמונות טיולים לאתר.

- עדכון פרטי לקוח.
- תכנון טיול.
- צפיה בתוצאות הטיולים.
- צפיה בטיולים קודמים.
- הכנסת תגובות.
- הרשמות/התחברות לאתר.
- הוספת אתר חדש.
- תכנון טיול.
- עדכון אתר.
- הצגת קישור לתחנות דלק במיקום הטיול.
- הצגת רשימת המצרכים.
- הוספת אזור בארץ.
- הוספת תת אזור בארץ.
- הוספת טיול חדש ללקוח.
- הוספת אתרים לטיול ללקוח.

12.2 Use Cases עיקריים:

1.תכנון הטיול:

UC1:

- UC1: Indentifier
- Name: תכנון הטיול
- Desscription: המשתמש מכניס פרטים בסיסיים: אזור בארץ, גילאי המטיילים, חצי יום/יום שלם ורכב/אוטובוס והמערכת תכננת לו 5 אופציות לטיולים.
- Actors: משתמש.
- Status:
- Frequency: בעת תכנון טיול.
- Conditions-Pre: פרטים בסיסיים.
- Conditions-Post: תוצאות התכנון מוצגות למשתמש והטיול המובחר נשמר בהסטוריה.

- Extended use cases : *קישור לתחנות דלק באזור.

  *הצגת רשימת מצרכים.

  *צפיה בתוצאות הטיולים.

  *הוספת אתר חדש.

  *הוספת תת אתר חדש.

- Assumptions: פרטי המשתמש נכונים.
- Basic course of action: המשתמש פונה מתוך האזור האישי שלו לתכנון הטיול.
- Alternate course of action: משתמש שלא רשום במערכת פונה לתכנון הטיול.
- Change history: גרסא ראשונה. אסתי זלוצ'בסקי.
- Issues: *הכנסת נתונים

  * אתחול הנתונים

  *סינון ראשוני

  *סינון מורחב

  *הגרלת האתרים

  *בחירת הקרוב ביותר ע"י google maps

- Decisions : הרצה נוספת של האלגוריתם יכולה להביא לתוצאות שונות(וזה בכונה תחילה כדי שהמשתמש יוכל לקבל תוצאות רבות יותר במידה ויחפוץ בכך).

2.הוספת אתר חדש :

UC2 :

- UC2: Indentifier
- Name: הוספת אתר חדש.
- Desscription: המשתמש מכניס כתובת אתר המערכת ופרטים בסיסיים.
- Actors: משתמש.
- Status :
- Frequency: בעת הוספת אתר.
- Conditions-Pre: פרטים בסיסיים על האתר.
- Conditions-Post: האתר מוכנס ל-DB.
- Extended use cases : *עדכון אתר.

- Assumptions: פרטי האתר נכונים.
- Basic course of action: המשתמש פונה מתוך האזור האישי שלו להכנסת אתר.
- Alternate course of action: משתמש שלא רשום במערכת פונה להכנסת אתר.
- Change history: גרסא ראשונה. אסתי זלוצ'בסקי.
- Issues: *הכנסת הכתובת.

  * וידוא נכונות הכתובת ע"י פניה ל-google maps.

  *הכנסת פרטים נוספים לפי בחירת המשתמש.

- Decisions: אנו מסתמכים על נכונות הפרטים.

האזור האישי של המשתמש:

UC3:

- UC3: Indentifier
- Name: טיפול באיזור האישי.
- Desscription: המשתמש יכול להעלות לאתר תמונות שלו מהטיולים, תזכורות ודירוגים וכן יש לו אפשרות לצפות בטיוליו הקודמים.
- Actors: משתמש.
- Status:
- Frequency: בעת התחברות/הרשמות.
- Conditions-Pre: תעודת זהות וסיסמא.
- Conditions-Post: שמירת הפריטים שהזין.
- Extended use cases: * העלאת תמונות טיולים לאתר.

  * עדכון פרטי לקוח.

  * הכנסת תגובות.

  * הוספת אתר חדש.

- Assumptions:
- Basic course of action: המשתמש נרשם.
- Alternate course of action: המשתמש מתחבר.
- Change history: גרסא ראשונה. אסתי זלוצ'בסקי.
- Issues: *התחברות/הרשמות.

* וידוא נכונות הסיסמא.

*צפיה באזור.

- Decisions :

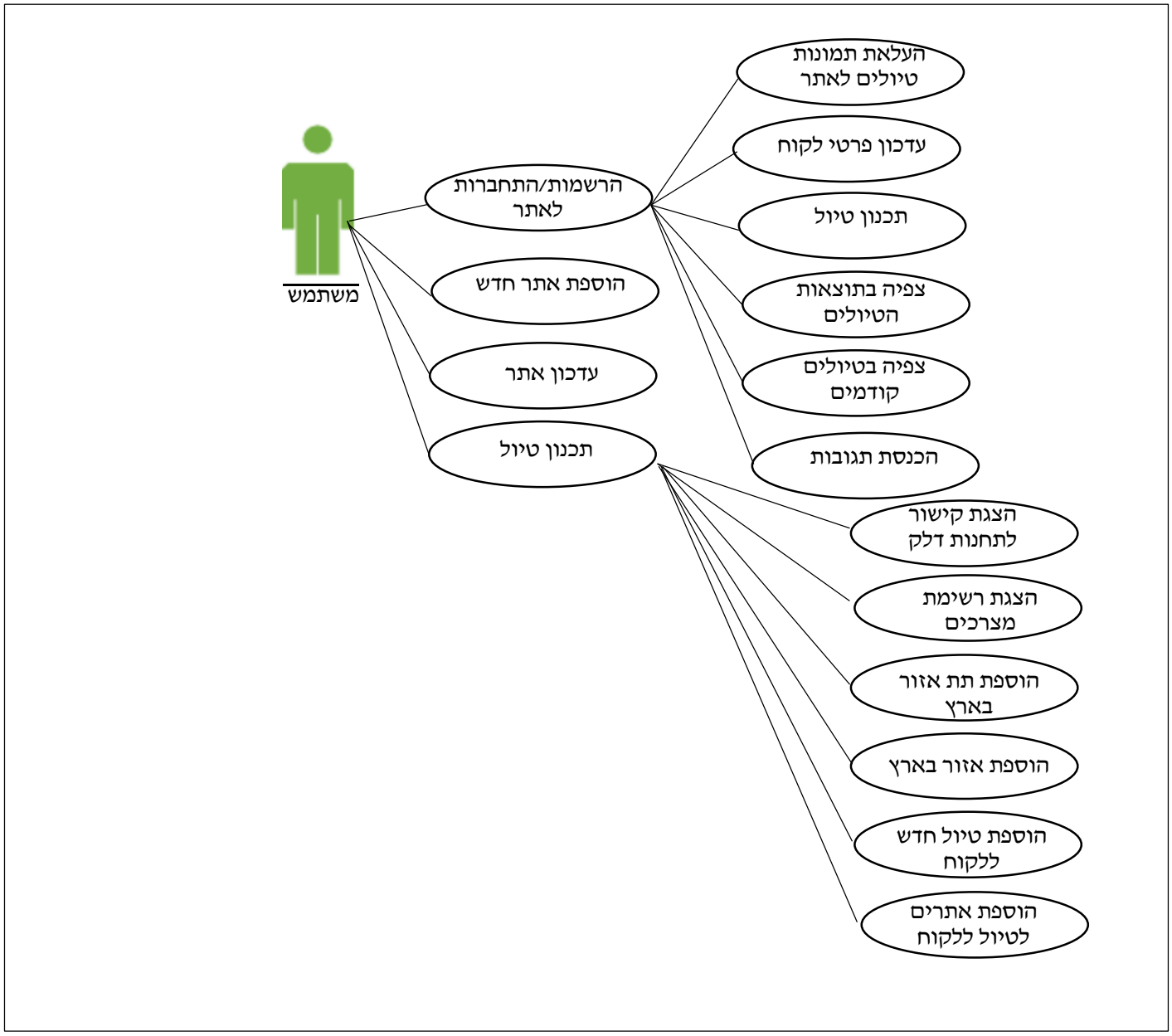
Use cases לפונקציות העיקריות

CreateTravels-קליטת נתונים מהמשתמש.

AddSite-קליטת מיקום האתר ופרטים נוספים עליו.

SignUp/SignIn-קליטת ת.ז. וסיסמא מהמשתמש.

12.3: Use case Diagram



12.4מבני הנתונים בפרויקט

מילון-המילון הוא מבנה נתונים המגדיר אוסף של מפתחות וערכים- הדבר סייע לי בשמירה על תוצאות החיפוש כאשר המפתח הוא מספר הטיול והערך הוא רשימת האתרים לטיול זה –צורה זו עוזרת לגשת לכל אתר במהירות תוך שמירה על הסדר.

רשימות- הרשימה מאפשרת שמירה של נתונים בצורה סדרתית בלי חובה להגדיר את הגודל. ישנן גם פעולות רבות שמאד קל להפעיל אותם על רשימה בשפת #c.על כן השתמשתי ברשימות בכל פעם כשרציתי לשמור אוסף של ערכים ללא מספר קבוע מראש וכן כשרציתי להפעיל על האוסף פונקציות שונות כגון פונקציות linq וכו' לדוגמא: שמירת רשימות האתרים מכל סוג,שמירת תוצאות הסינון וכו'.

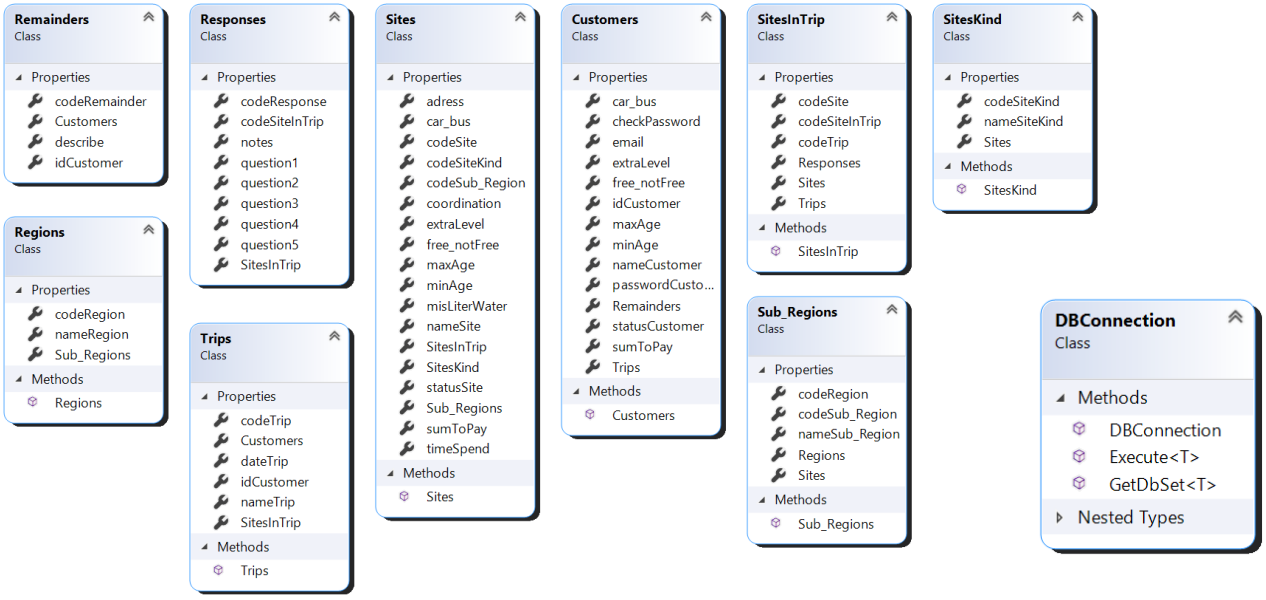
מערכים-המערך הוא אוסף של איברים השתמשתי בו רבות כאשר רציתי לשמור את אוסף עם גודל מוגדר מראש(במקרים אלו העדפתי על פני רשימה מבחינת ניצול הזיכרון ועיקרון הלוקליות).

12.5עץ מודולים

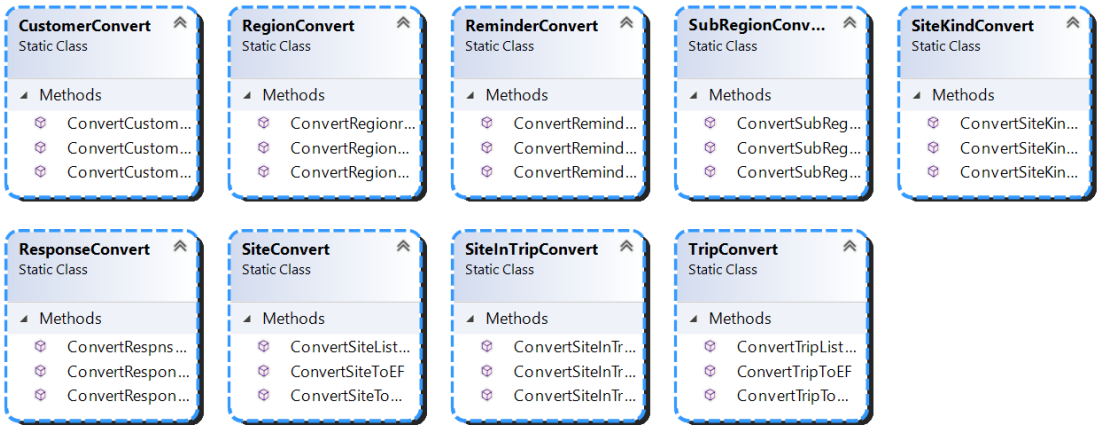
לא רלוונטי

12.6תרשים מחלקות

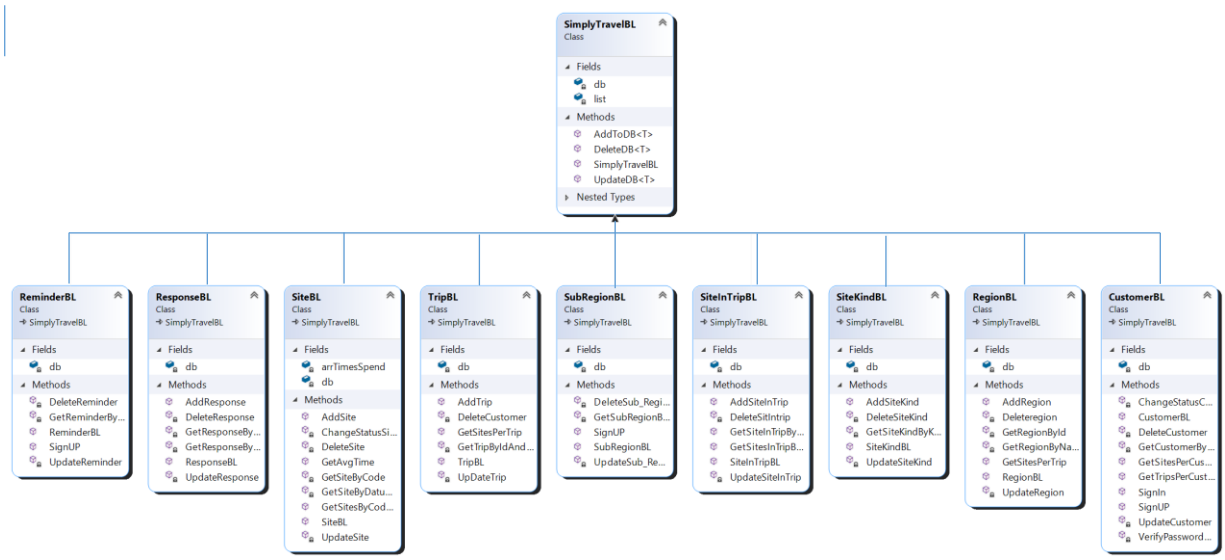
שכבת הגישה למסד הנתונים DAL.



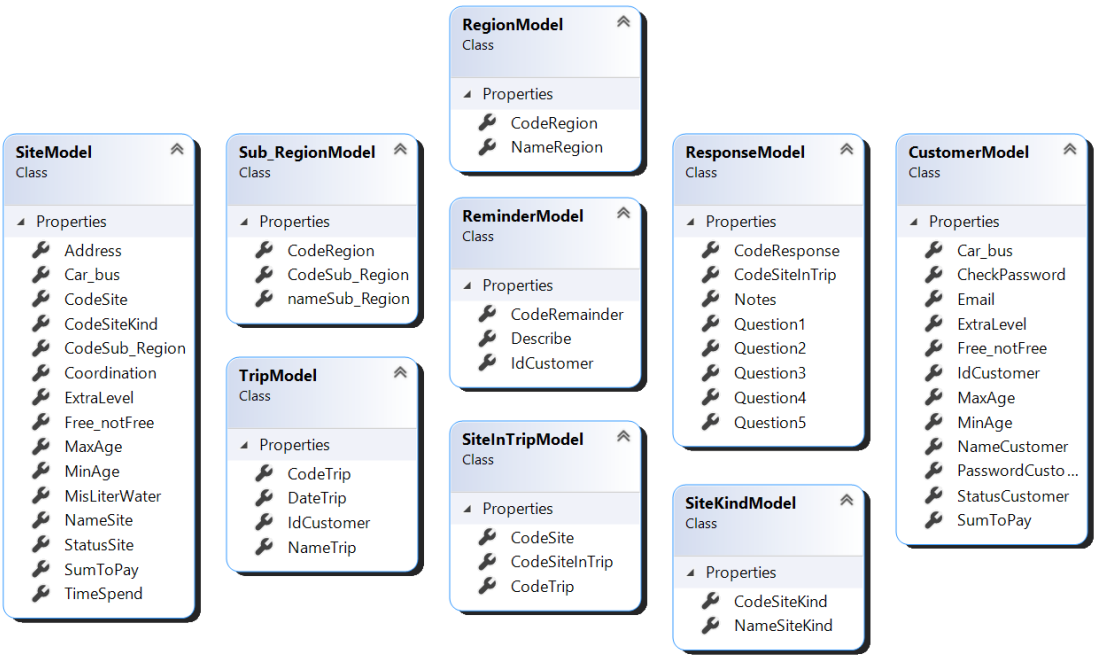
פונקציות המרה:



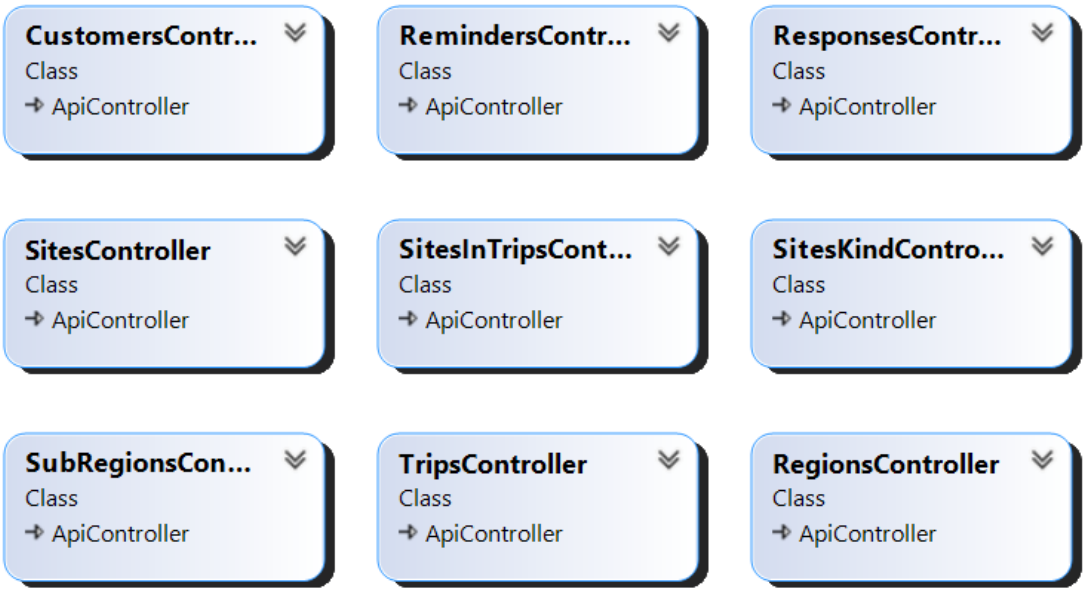
שכבת האלגוריתם BLL



שכבת ה-Models:



שכבת הApi:



12.7תיאור המחלקות המוצעות

1.שכבת הdata base - שכבה זו היא בסיס נתונים שישמור את הנתונים עבור הפרויקט. בנוי כקובץ של SQL DATABASE ומכיל את כל הטבלאות הנחוצות עבור הנתונים שישמשו עבור הפרויקט. בדטה בייס נשמרים הנתונים הבאים לפי טבלה:

טבלת משתמשים: בה נשמרים המשתמשים שנרשמו למערכת- עבור המשתמשים נשמר המידע הרלוונטי של המשתמש כמו תעודת זהות , שם, , כתובת מדויקת ,כתובת מייל וסיסמא לשימוש במערכת. וכן נשמר עבור המשתמש פרטים נוספים הנדרשים לתכנון הטיול: גיל מינימלי וגיל מקסימלי, רמת אתגר שהוא רוצה, דרכי הגעה לטיול וסכום גג לתשלום.

טבלת תזכורות : עבור כל משתמש (רשום) נשמרים התזכורות אותם הכניס באזור האישי עבור כל תזכורת נשמר: קוד תזכורת, תעודת זהות לקוח והתזכורת.

טבלת דירוגי התגובות לטיול: עבור כל משתמש (רשום) תגובותיו לאתרים שטייל בהם בעבר. מידע זה עוזר למערכת בתכנון טיוליו הבאים וכן בתכנון הטיולים לכלל המשתמשים. עבור כל תגובה נשמר: קוד מזהה, תעודת זהות לקוח, הערות ו-5 דירוגים לשאלות.

טבלת הטיולים: עבור כל משתמש (רשום) נשמרים הטיולים הקודמים שטייל עבור כל טיול נשמר: קוד מזהה, תעדות זהות לקוח, תאריך ושם הטיול.

טבלת האתרים בטיולים: עבור כל טיול שטייל המשתמש נשמרים האתרים של הטיול לכל אתר בטיול נשמר: קוד מזהה, קוד טיול(מטבלת טיולים) וקוד אתר(מטבלת אתרים).

טבלת סוגי האתרים: בה שמורים סוגי האתרים לכל סוג נשמר: קוד מזהה ושם הסוג.

טבלת אתרים: בה נשמרים האתרים טיולים שמוכנסים ע"י המשתמשים-לכל אתר נשמרים פרטיו כמו: קוד מזהה, קוד סוג אתר(מטבלת סוגי האתרים),כתובת, קורדינציה, גילאים מתאימים, סכום לתשלום וזמן בילוי באתר .

טבלת אזורים: בה נשמרים האזורים בארץ-קוד מזהה ושם האזור.

טבלת תתי אזורים: בה נשמרים תתי האזורים בארץ לכל תת אזור נשמר: קוד מזהה, קוד אזור(מטבלת אזורים) ושם תת אזור.

2. שכבת הגישה לנתונים: שכבה זו היא השכבה האחראית על הגישה לנתונים השמורים במסד הנתונים ולהעבירם לשכבה המבצעת את האלגוריתמים. על ידי הפקודה Db-Scaffold נוצרות

ישויות -מחלקות עבור כל אחת מהטבלאות שבמסד הנתונים שדרכם, עם טכנולוגיית ה - Framework Entity ניתן לגשת לנתוני מסד הנתונים.

המחלקות:

DBConnection

תכונות המחלקה:

public enum ExecuteActions –המכיל את 3 אופציות הגישה למסד הנתונים: Insert/Update/Delete.

הפונקציות:

- public DBConnection()

פעולה בונה היוצרת עצם מסוג המחלקה.

- public void Execute<T>(T entity, ExecuteActions exAction) where T : class

פונקציה גנרית המקבלת Entity ושם הפעולה שיש לבצע (ע"י ה-Enum- ExecuteActions ש"מכריח" לבחור פעולה מתוך השלוש) ומופעלת על שם מחלקה מסוים.

הפונקציה מבצעת את הפעולה ומרעננת את ה-DB.

- public List<T> GetDbSet<T>() where T : class

פונקציה גנרית ששולפת נתוני טבלה מסויימת מה-DB.פונקציה זו מופעלת על שם המחלקה שממנה רוצים לשלוף ומחזירה את רשימת העצמים של המחלקה.

בשכבה זו ישנן 9 מחלקות נוספות-מחלקות סטטיות שהן אחראיות להמיר בין העצמים של ה Entity -Framework לבין העצמים של פרויקט ה- Models.

הסיבה לעשיית מחלקות אלו היא: החילול האוטומטי (Entity Framework)יוצר מחלקות כנגד כל טבלה שבמסד הנתונים אך הוא מוסיף למחלקות אלו פרטים נוספים שמפריעים לריצת הפרויקט. ולכן עשיתי את שכבת הישויות בה יש מחלקות כנגד כל מחלקה של ה-Entity Framework . כך במחלקות החדשות משתמשים לאורך כל הפרויקט וכאשר רוצים לבוא במגע עם הדטה בייס (אם להוצאת נתונים ואם להכנסת הנתונים ושינוים)ממירים בין המחלקות.לצורך כל יש בשכבת הDal תיקיה שנקראת Convert בה יש מחלקה להמרה כנגד כל מחלקה ב-DB. וכך כאשר רוצים לקבל את טבלת המשתמשים מהדטה בייס, משתמשים בפונקציה<GetDBSet <Customers שב-DBConnection ומקבלים את הרשימה של המשתמשים מסוג Customers, ממירים אותה לרשימה של עצמים מהסוג CustomersModels ומבצעים על הרשימה החדשה מה שרוצים. ואותו הדבר גם בצורה ההפוכה לדוגמא כאשר רוצים להוסיף משתמש חדש לדטה בייס, יוצרים עצם מסוג מחלקת CustomerModel ומכניסים את התכונות. אח"כ ממירים את העצם לעצם מסוג Customers ומכניסים ל-DB.

CustomerConvert

הפונקציות:

- public static Customers ConvertCustomerToEF(CustomerModel customer)

מקבלת אוביקט מסוג CustomerModel וממירה אותו (מחזירה) מסוג Customers.

- public static CustomerModel ConvertCustomerToModel(Customers customer)

מקבלת אוביקט מסוג Customers וממירה אותו (מחזירה) מסוג CustomerModel.

- public static List<CustomerModel> ConvertCustomerListToModel(IEnumerable<Customers> customers)

מקבלת רשימה של אוביקטים מסוג Customers וממירה אותה (מחזירה) לרשימה מסוג CustomerModel.

RegionConvert

הפונקציות:

- public static Regions ConvertRegionToEF(RegionModel region)

מקבלת אוביקט מסוג RegionModel וממירה אותו (מחזירה) מסוג Regions.

- public static RegionModel ConvertRegionToModel(Regions regions)

מקבלת אוביקט מסוג Regions וממירה אותו (מחזירה) מסוג RegionModel.

- public static List<RegionModel> ConvertRegionListToModel(IEnumerable<Regions> regions)

מקבלת רשימה של אוביקטים מסוג Regions וממירה אותה (מחזירה) לרשימה מסוג RegionModel.

SubRegionConvert

הפונקציות:

- public static SubRegions ConvertSubRegionToEF(SubRegionModel subRegion)

מקבלת אוביקט מסוג SubRegionModel וממירה אותו (מחזירה) מסוג SubRegions.

- public static SubRegionModel ConvertSubRegionToModel(SubRegions . subRegions)

מקבלת אוביקט מסוג SubRegions וממירה אותו (מחזירה) מסוג SubRegionModel.

- public static List<SubRegionModel> ConvertSubRegionListToModel(IEnumerable<SubRegions> subRegions)

מקבלת רשימה של אוביקטים מסוג SubRegions וממירה אותה (מחזירה) לרשימה מסוג SubRegionModel.

TripsConvert

הפונקציות:

- public static Trips ConvertTripToEF(TripModel customer)

מקבלת אוביקט מסוג TripModel וממירה אותו (מחזירה) מסוג Trips.

- public static TripModel ConvertTripToModel(Trips trip)

מקבלת אוביקט מסוג Trips וממירה אותו (מחזירה) מסוג TripModel.

- public static List<TripModel> ConvertTripListToModel(IEnumerable<Trips> trips)

מקבלת רשימה של אוביקטים מסוג Trips וממירה אותה (מחזירה) לרשימה מסוג TripModel.

SiteConvert

הפונקציות:

- public static Sites ConvertSiteToEF(SiteModel site)

מקבלת אוביקט מסוג SiteModel וממירה אותו (מחזירה) מסוג Sites.

- public static SiteModel ConvertSiteToModel(Sites site)

מקבלת אוביקט מסוג Site וממירה אותו (מחזירה) מסוג SiteModel.

- public static List<SiteModel> ConvertSiteListToModel(IEnumerable<Sites> sites)

מקבלת רשימה של אוביקטים מסוג Sites וממירה אותה (מחזירה) לרשימה מסוג SiteModel.

SiteKindConvert

הפונקציות:

- public static SiteKinds ConvertSiteKindToEF(SiteKindModel siteKind)

מקבלת אוביקט מסוג SiteKindModel וממירה אותו (מחזירה) מסוג SiteKinds.

- public static SiteKindModel ConvertSiteKindToModel(SiteKinds siteKind)

מקבלת אוביקט מסוג SiteKinds וממירה אותו (מחזירה) מסוג SiteKindModel.

- public static List<SiteKindModel> ConvertSiteKindListToModel(IEnumerable<siteKinds> SiteKinds)

מקבלת רשימה של אוביקטים מסוג SiteKinds וממירה אותה (מחזירה) לרשימה מסוג SiteKindModel.

ReminderConvert

הפונקציות:

- public static Reminders ConvertReminderToEF(ReminderModel reminder)

מקבלת אוביקט מסוג ReminderModel וממירה אותו (מחזירה) מסוג Reminders.

- public static ReminderModel ConvertReminderToModel(Reminders reminder)

מקבלת אוביקט מסוג Reminders וממירה אותו (מחזירה) מסוג ReminderModel.

- public static List<ReminderModel> ConvertReminderListToModel(IEnumerable<Reminders> reminders)

מקבלת רשימה של אוביקטים מסוג Reminders וממירה אותה (מחזירה) לרשימה מסוג ReminderModel.

ResponseConvert

הפונקציות:

- public static Responses ConvertResponseToEF(ResponseModel response)

מקבלת אוביקט מסוג ResponseModel וממירה אותו (מחזירה) מסוג Responses.

- public static ResponseModel ConvertResponseToModel(Responses response)

מקבלת אוביקט מסוג Responses וממירה אותו (מחזירה) מסוג ResponseModel.

- public static List<ResponseModel> ConvertResponseListToModel(IEnumerable<Responses> responses)

מקבלת רשימה של אוביקטים מסוג Responses וממירה אותה (מחזירה) לרשימה מסוג ResponseModel.

SiteInTripConvert

הפונקציות:

- public static SiteInTrips ConvertSiteInTripToEF(SiteInTripModel siteInTrip)

מקבלת אוביקט מסוג SiteInTripModel וממירה אותו (מחזירה) מסוג SiteInTrips.

- public static SiteInTripModel ConvertSiteInTripToModel(SiteInTrips siteInTrip)

מקבלת אוביקט מסוג SiteInTrips וממירה אותו (מחזירה) מסוג SiteInTripModel.

- public static List<SiteInTripModel> ConvertSiteInTripListToModel(IEnumerable<SiteInTrips> sitesInTrip)

מקבלת רשימה של אוביקטים מסוג SiteInTrips וממירה אותה (מחזירה) לרשימה מסוג SiteInTripModel.

קוד הפונקציות:

פונקציות המחלקה DBConnection :

```
public List<T> GetDbSet<T>() where T : class
{
    using (SimplyTravelEntitiesNew s = new SimplyTravelEntitiesNew())
    {
        return s.Set<T>().ToList();
    }
}
public enum ExecuteActions
{
    Insert,
    Update,
    Delete
}
public void Execute<T>(T entity, ExecuteActions exAction) where T : class
{
    using (SimplyTravelEntitiesNew s = new SimplyTravelEntitiesNew())
    {
        var model = s.Set<T>();
        switch (exAction)
        {
            case ExecuteActions.Insert:
                model.Add(entity);
                break;
            case ExecuteActions.Update:
                model.Attach(entity);
                s.Entry(entity).State = System.Data.Entity.EntityState.Modified;
                break;
            case ExecuteActions.Delete:
                model.Attach(entity);
                s.Entry(entity).State = System.Data.Entity.EntityState.Deleted;
                break;
            default:
                break;
        }
        s.SaveChanges();
    }
```

פונקציות המחלקה CustomerConvert :

```
public static Customers ConvertCustomerToEF(CustomerModel customer)
{
    return new Customers
    {
        idCustomer = customer.IdCustomer,
        passwordCustomer = customer.PasswordCustomer,
        checkPassword = customer.CheckPassword,
        email = customer.Email,
        nameCustomer = customer.NameCustomer,
        extraLevel = customer.ExtraLevel,
        free_notFree = customer.Free_notFree,
        sumToPay = customer.SumToPay,
        minAge = customer.MinAge,
        maxAge = customer.MaxAge,
        car_bus = customer.Car_bus,
        statusCustomer = customer.StatusCustomer
    };
}
public static CustomerModel ConvertCustomerToModel(Customers customer)
{
    return new CustomerModel
    {
        IdCustomer = customer.idCustomer,
        PasswordCustomer = customer.passwordCustomer,
        CheckPassword = customer.checkPassword,
        Email = customer.email,
        NameCustomer = customer.nameCustomer,
        ExtraLevel = customer.extraLevel,
        Free_notFree = customer.free_notFree,
        SumToPay = customer.sumToPay,
        MinAge = customer.minAge,
        MaxAge = customer.maxAge,
        Car_bus = customer.car_bus,
        StatusCustomer = customer.statusCustomer
    };
}

public static List<CustomerModel> ConvertCustomerListToModel(IEnumerable<Customers> customers)
{
    return customers.Select(c => ConvertCustomerToModel(c)).OrderBy(n => n.IdCustomer).ToList();
}
```

3. שכבת הישויות(Models) -בשכבה זו ישנן מחלקות כנגד כל מחלקה שנוצרה ע"י ה Entity Framework (כנגד כל טבלה שבדטה בייס).

4. שכבת הbl- בשכבה זו מתבצעים האלגוריתמים וקטעי הקוד הקריטיים והמשמעותיים בפרויקט.

שכבה זו מחולקת גם היא למחלקות35 : 35

SimplyTravelBl שהיא מחלקת בסיס למחלקות האחרות :

תכונות המחלקה : db- מופע ממחלקת DBConnection שמאפשר לתקשר בעזרתו עם מסד הנתונים.

הפונקציות :

- public SimplyTravelBL()

פעולה בונה המאתחלת את המופע db.

- public void AddToDB<T>(T entity) where T: class

פונקציה גנרית המקבלת עצם ומוסיפה אותו לדטה בייס ע"י זימון הפונקציה הגנרית שב-db.היא מופעלת על שם של המחלקה בה רוצים להוסיף.

- public void DeleteDB<T>(T entity) where T : class

פונקציה גנרית המקבלת עצם ומוחקת אותו מהלדטה בייס ע"י זימון הפונקציה הגנרית שב-db .היא מופעלת על שם של המחלקה בה רוצים למחוק.

- public void UpdateDB<T>(T entity) where T : class

פונקציה גנרית המקבלת עצם ומעדכנת אותו בדטה בייס ע"י זימון הפונקציה הגנרית שב-db .היא מופעלת על שם של המחלקה בה רוצים לעדכן.

פונקציות המחלקה SimplyTravelBl :

```
public class SimplyTravelBL
{
    DBConnection db;
    public SimplyTravelBL()
    {
        db = new DBConnection();
    }
    public enum Result
    {
        IncorrrectDetails,
        NotFound,
        Found
    }
    public List<T> GetDbSet<T>()where T:class
    {
        return db.GetDbSet<T>();
    }
    public void AddToDB<T>(T entity) where T:class
    {
        db.Execute<T>(entity, DBConnection.ExecuteActions.Insert);
    }
    public void DeleteDB<T>(T entity) where T : class
    {
        db.Execute<T>(entity, DBConnection.ExecuteActions.Delete);
    }
    public void UpdateDB<T>(T entity) where T : class
    {
        db.Execute<T>(entity, DBConnection.ExecuteActions.Update);
    }
}
```

CustomerBl

מחלקה זו יורשת ממחלקת SimplyTravelBl ומשתמשת בפונקציות שלה.

הפונקציות :

- public CustomerBL() פעולה בונה היוצרת עצם מהמחלקה.
- public List<TripModel> GetTripsPerCustomer(int id)

פונקציה המקבלת מספר תעודת זהות ומביאה את רשימת הטיולים השייכים למשתמש זה.

- public List<SiteInTripModel> GetSitesPerCustomer(int id)

פונקציה המקבלת מספר תעודת זהות ומחזירה רשימת אתרים בהם ביקר המשתמש הזה.

- private CustomerModel GetCustomerById(int id)

פונקציה המקבלת מספר תעודת זהות ומחזירה את המשתמש שמספרו =למספר תעודת הזהות.

public int SignUP(CustomerModel customerNew)
פונקציה המקבלת משתמש חדש ומנסה להוסיף אותו לדטה בייס היא מחזירה-במקרה ולא נמצא כבר בdb או שנמצא והסיסמא שלו תקינה, את מספר תעודת הזהות שלו(לצורך המשך השימוש) .במקרה אחר מחזירה 0.

- public int SignIn(int id, string password)

פונקציה המקבלת מספר תעודת זהות וסיסמא ומוודאת אם הסיסמא נכונהפונקציה מחזירה - 1במקרה ולא נמצא בDB , ו-0 במקרה ונמצא אך סיסמתו שגויה. ובמקרה ונמצא וסיסמתו נכונה, את מספר תעודת הזהות שלו(לצורך המשך שימוש).

- private string ChangeStatusCustomer(int id, string status)

פונקציה המקבלת מספר תעודת זהות וסטטוס משתמש, משנה את הסטטוס ומחזירה את הסטטוס הקודם.

private int DeleteCustomer(int id)
פונקציה המקבלת מספר תעודת זהות ומוחקת את משתמש מהדטה בייס הפונקציה מחזירה 0 במקרה והיה קיים, במקרה ולא היה קיים, 1.

- private int UpdateCustomer(CustomerModel c)

פונקציה המקבלת משתמש, ומעדכנת משתמש זה בדטה בייס הפונקציה מחזירה 0 במקרה והיה קיים, במקרה ולא היה קיים, 1.

RegionBl

מחלקה זו יורשת ממחלקת SimplyTravelBl ומשתמשת בפונקציות שלה.

הפונקציות:

- public RegionBL() פעולה בונה היוצרת עצם מהמחלקה.

- private RegionModel GetRegionByName(string name)

פונקציה המקבלת שם אזור ומחזירה את האזור הזה.

- public List<Sub_RegionModel> GetSubRegionOfRegion(int codeR)

פונקציה המקבלת קוד אזור ומחזירה רשימה ש כל תתי האזורים השייכים לאזור הזה.

- private RegionModel GetRegionById(int code)

פונקציה המקבלת קוד אזור ומחזירה אזור זה.

- public int AddRegion(string name)

פונקציה המקבלת שם אזור בונה עצם מסוג אזור עם שם זה ומוסיפה אותו לדטה בייס.
הפונקציה מחזירה במקרה והאזור כבר קיים, 0.אחרת, את קוד האזור.

- private int Deleteregion(string name)

פונקציה המקבלת שם אזור ומוחקת אותו מהדטה בייס. הפונקציה מחזירה 1 במקרה ולא קיים, אחרת 0.

- private int UpdateRegion(string name)

פונקציה המקבלת שם אזור לעדכון ומעדכנת בדטה בייס את שם האזור הזה.הפונקציה מחזירה 1 במקרה ולא קיים, אחרת 0.

ReminderBl

מחלקה זו יורשת ממחלקת SimplyTravelBl ומשתמשת בפונקציות שלה.

הפונקציות :

- public ReminderBL() פעולה בונה היוצרת עצם מהמחלקה.
- private ReminderModel GetReminderById(int id)

פונקציה המקבלת תעודת זהות משתמש ומחזירה את התזכורת שלו.

- public int AddReminder(int id,string des)

פונקציה המקבלת תעודת זהות לקוח ותיאור ומכניסה את התזכורת לדטה בייס. הפונקציה מחזירה 0 במקרה וכבר קיימת תזכורת זו וקוד התזכורת במקרה ולא היתה קיימת והוספה בהצלחה.

- private int DeleteReminder(int id)

פונקציה המקבלת תעודת זהות משתמש ומוחקת את התזכורת שלו מהדטה בייס. הפונקציה מחזירה 1 במקרה ולא קיימת תזכורת למשתמש, אחרת 0.

- private int UpdateReminder(ReminderModel c)

פונקציה המקבלת תזכורת לעדכון ומעדכנת בדטה בייס את התזכורת הזו הפונקציה מחזירה 1 במקרה ולא קיים, אחרת 0.

ResponseBl

מחלקה זו יורשת ממחלקת SimplyTravelBl ומשתמשת בפונקציות שלה.

הפונקציות:

- public ResponseBL() פעולה בונה היוצרת עצם מהמחלקה.
- private bool GetResponseByCodeToSiteToSpecifiecCustomer(int code,int id)

הפונקציה מקבלת קוד אתר מסויים ותעודת זהות משתמש. הפונקציה מוצאת את תגובתו של הלקוח לאתר הזה ומחזירה true במקרה ותשובתו לשאלה "האם ירצה לחזור על אתר זה שנית" היתה כן, אחרת יחזור false.

- private ResponseModel GetResponseByCode(int code)

פונקציה המקבלת קוד אתר בטיול ומחזירה את התגובה לטיול (אם היתה תגובה).

- public int AddResponse(int code,int q1,int q2,int q3,int q4,string note)

פונקציה המקבלת פרטי תגובה ומוסיפה אותה לדטה בייס. הפונקציה מחזירה 0 במקרה שכבר קיים ואת קוד התגובה במקרה והוספה בהצלחה.

- private int UpdateResponse(ResponseModel r)

פונקציה המקבלת תגובה לעדכון ומעדכנת בדטה בייס את התגובה הזו הפונקציה מחזירה 1 במקרה ולא קיים, אחרת 0.

- private int DeleteResponse(int code)

פונקציה המקבלת קוד אתר בטיול ומוחקת את התגובה שלו מהדטה בייס. הפונקציה מחזירה 1 במקרה ולא קיימת תגובה לאתר, אחרת 0.

SiteBl

מחלקה זו יורשת ממחלקת SimplyTravelBl ומשתמשת בפונקציות שלה.

תכונות המחלקה: arrTimesSpend -מערך בגודל 9 המכיל את הזמן בילוי הדפולטיבי בכל סוג אתר. המערך מאותחל מיד בהגדרה.

<u>האיתחול לפי שמות סוגי האתרים:</u>

מוזאון	שמורת טבע	אטרקציה רטובה	אטרקציה יבשה	מסלול רטוב	מסלול יבש	קבר	מסעדה	בית קפה

0.5	1	0.5	3	3	2	2	3	4

הפונקציות :

- public SiteBL() פעולה בונה היוצרת עצם מהמחלקה.
- private SiteModel GetSiteByDatum_point(string location)

הפונקציה מקבלת קורדינציה של אתר ומחזירה אותו.

- public int GetAvgTime()

פונקציה המחשבת את ממוצע זמן הבילוי באתר ומחזירה אותו.

- private SiteModel GetSiteByCode(int code)

פונקציה המקבלת קוד אתר ומחזירה אותו .

- public List<SiteModel> GetSitesByCodeKindSite(int codeKindSite)

פונקציה המקבלת קוד סוג אתר ומחזירה את רשימת האתרים של סוג זה .

- public int AddSite(SiteModel site)

פונקציה המקבלת אתר ומוסיפה אותו לדטה בייס. הפונקציה מחזירה 0 במקרה שכבר קיים ואת קוד האתר במקרה והוסף בהצלחה.

- private string ChangeStatusSite(int code, string status)

פונקציה המקבלת קוד אתר וסטטוס אתר, משנה את הסטטוס ומחזירה את הסטטוס הקודם.

- private int UpdateSite(SiteModel c)

פונקציה המקבלת אתר לעדכון ומעדכנת בדטה בייס את האתר הזה הפונקציה מחזירה 1 במקרה ולא קיים האתר, אחרת 0.

- private int DeleteSite(int code)

פונקציה המקבלת קוד אתר ומוחקת אותו מהדטה בייס. הפונקציה מחזירה 1 במקרה והאתר לא קיים, אחרת 0.

SubRegionBl

מחלקה זו יורשת ממחלקת SimplyTravelBl ומשתמשת בפונקציות שלה.

הפונקציות :

- public SubRegionBL() פעולה בונה היוצרת עצם מהמחלקה.

- private Sub_RegionModel GetSubRegionByNameAndCode(string name,int code)

הפונקציה מקבלת שם תת אזור וקוד אזור ומחזירה את תת האזור.

- public int AddSubRegion(string name, int code)

פונקציה המקבלת שם תת אזור וקוד אזור ומוסיפה את תת האזור לדטה בייס. הפונקציה מחזירה 0 במקרה שכבר קיים ואת קוד תת האזור במקרה והוסף בהצלחה.

- private int UpdateSub_Region(Sub_RegionModel c)

פונקציה המקבלת תת אזור לעדכון ומעדכנת בדטה בייס .הפונקציה מחזירה 1 במקרה ולא קיים, אחרת 0.

- private int DeleteSub_Region(string name,int CodeRegion)

פונקציה המקבלת שם תת אזור וקוד אזור ומוחקת את תת האזור מהדטה בייס. הפונקציה מחזירה 1 במקרה ולא קיים תת האזור, אחרת 0.

TripBl

מחלקה זו יורשת ממחלקת SimplyTravelBl ומשתמשת בפונקציות שלה.

הפונקציות :

- public TripBL() פעולה בונה היוצרת עצם מהמחלקה.
- private TripModel GetTripByIdAndDate(int id,DateTime date)

הפונקציה מקבלת תעודת זהות משתמש ותאריך ומחזירה הטיול למשתמש זה ביום זה.

- public List<SiteInTripModel> GetSitesPerTrip(int CodeTrip)

פונקציה המקבלת קוד טיול ומחזירה את רשימת האתרים בו.

- public int AddSubRegion(string name, int code)

פונקציה המקבלת שם תת אזור וקוד אזור ומוסיפה את תת האזור לדטה בייס. הפונקציה מחזירה 0 במקרה שכבר קיים ואת קוד תת האזור במקרה והוסף בהצלחה.

- private int UpDateTrip(TripModel t)

פונקציה המקבלת טיול לעדכון ומעדכנת בדטה בייס .הפונקציה מחזירה 1 במקרה ולא קיים, אחרת 0.

- private int DeleteTrip(int id,DateTime date)

פונקציה המקבלת תעודת זהות משתמש ותאריך ומוחקת את הטיול מהדטה בייס. הפונקציה מחזירה 1 במקרה ולא קיים הטיול, אחרת 0.

SendMail

מחלקה זו היא מחלקה סטטית (כלומר א"א ליצור ממנה עצמים) ומשמשת לשליחת מייל למשתמש במקרה ושכח את הסיסמא. במייל תופיע סיסמא חדשה אותה הוא יצטרך להזין במערכת וכך יוכל להכנס(לאחר כניסתו הוא יוכל לשנות את הסיסמא למשהו אחר).

במחלקה יש פונקציה אחת והיא פעולת שליחת המייל:

public static void SendMailToConfirmPassword(CustomerModel c)

פונקציה זו מקבל עצם מסוג משתמש ודואגת להגריל סיסמא אחרת ולשלוח לכתובת המייל של המשתמש את הסיסמא.

Validition

במחלקה זו ישנן פונקציות שדואגות לבדיקת תקינות הנתונים שנכנסים -DB.

הפונקציות:

- public static bool LegalId(int id)

  בדיקת תקינות תעודת זהות(מכל הבחינות).

- public static bool IsHebrew(string word)

  בדיקה שהטקסט הינו בעברית בלבד.

- public static bool IsTelephone(string tel)

  בדיקת תקינות טלפון.

- public static bool IsPassword(string pass,int id)

  בדיקת תקינות סיסמא.

- public static bool IsCellPhone(string tel)

  בדיקת תקינות פלאפון.

- public static int GetAge(DateTime d)

  חישוב הגיל לפי תאריך לידה.

- public static bool CheackMail(string t)

  בדיקת תקינות מייל.

- public static bool IsNumber(string num)

  בדיקה שהטקסט מכיל רק מספרים.

- public static bool VerifyNotNumbers(string name)

בדיקה שהטקסט לא מכיל מספרים.

- public static bool VerifyLettersAndNumbers(string add)

בדיקה שהטקסט מכיל גם אותיות וגם מספרים.

- public static bool VerifyNotSpeciall(string add)

בדיקה שהטקסט מכיל רק אותיות או מספרים.

5. שכבת הWeb Api-התקשרות עם צד לקוח:

בשכבה זו ישנם Controllers כנגד כל מחלקה שבדטה בייס.

Controller יש את הנתיב אליו ופונקציות כנגד הפונקציות שבBL.

6.צד הלקוח-צד זה בנוי ע"י Angular ב-Type scrtipt. צד הלקוח מקבל פרטים ובקשות מהמשתמש, והוא שולח אותם לשכבת הרשת, דרך ה -Web Apiמגיעות הבקשות, והתוכנה מחזירה לאחר מכן את התשובה למשתמש.בצד הלקוח ישנן גם מחלקות כנגד כל טבלה של הדטה בייס וכן Services.

13.תיאור התוכנה

- סביבת עבודה:

Visual Studio Code ו Visual Studio

- שפות תכנות:

צד השרת נכתב בטכנולוגית Web Api ובשפת C# .

צד הלקוח נכתב בשפות html, Css ו– Type script בטכנולוגית Angular.

14.אלגוריתמים מרכזיים

14.1האלגוריתם המרכזי

הבעיה האלגוריתמית איתה התמודדתי בפרויקט היא תכנון הטיול ואתריו. ישנן מגוון אפשרויות ודרכים לעשות זאת אך אני רציתי להשתמש באלגוריתם שמצד אחד יסנן את האתרים לפי קריטריוני הלקוח ומצד שני ייתן ללקוח את הרעיונות לטיולים. כלומר הפרויקט אותו רציתי לפתח יעזור הן מהבחינה של בחירת אתרים המותאמים ללקוח והן מהבחינה שהוא זה שיעלה רעיונות לטיולים שונים המורכבים משילובים מוצלחים של אתרים. חקרתי רבות את הנושא ולאחר העלאה ופסילה של רעיונות אלגוריתמים רבים החלטתי שאני רוצה לפתור בעיה זו ע"י אלגוריתם שאכתוב בעצמי.

האלגוריתם יסנן בסינון ראשוני את האתרים בהתאם לפרטים שהכניס הלקוח ואח"כ יכין הצעות לטיולים שונים לפי סגנונות טיולים.

חלק נוסף באלגוריתם הוא חלק בו בודקים שהאתרים שומרים על מרחק קטן ביניהם(לצמצום הנסיעות) וזה נעשה ע"י התממשקות לGoogle Maps .

האלגוריתם עובד בצורה כזאת:

נתונים כמה מבני נתונים ראשוניים:

טבע	רגוע	קלאסי	היסטורי	אתגרי

1. arrNamesKinds -מערך שמות סגנונות הטיולים שהם טבע, רגוע, קלאסי, היסטורי ואתגרי.
2. matPercents -מטריצת אחוזים בה מחושבים כמה זמן מכל אתר יהיה לכל סגנון טיול, כלומר לכל סגנון טיול מתאים שילוב שונה של האתרים זה בזה. לדוגמא בטיול רגוע מתאים שילוב לדוגמא של 2 שמורות טבע, מסלול לא מסובך, מוזאון ובית קפה.

לעומת זאת בטיול אתגרי מתאים שילוב של: 2 מסלולים של רמת אתגר גבוהה, אטרקציה רצינית, מסעדה וקבר אחד.

לצורך כך הכנתי מטריצה של אחוזים בה מחושבים כמות האחוזים מכל אתר בכל סוג טיול.

(המטריצה הוכנה לפי חישובים רבים וסקירת הענין ע"י שאילת שאלות לאנשים).

המטריצה:

	בתי קפה	מסעדות	בתי קברות	מסלולים יבשים	מסלולים רטובים	אטרקציות יבשות	אטרקציות רטובות	שמורות טבע	מוזאונים
קלאסי	5%	5%	18%	9%	9%	9%	9%	18%	18%
טבע	0%	15%	1%	22%	22%	0%	0%	40%	0%
הסטוריה	0%	10%	17%	0%	0%	0%	0%	28%	45%
אתגרי	0%	15%	0%	17.5% *רק מרמת אתגר גבוהה	17.5% *רק מרמת אתגר גבוהה	25%	25%	0%	0%

רגוע	10%	10%	10%	15%	15%	0%	0%	20%	20%
				רק מראת* אתר נאוכה	רק מראת* אתר נאוכה				

לאור המטריצה לדוגמא בטיול אתגרי לא יופיעו כלל בתי קפה, שמורות טבע ומוזאונים כי סטטיסטית אתרים אלו לא מתאימים לטיול אתגרי. וכן בטיול היסטוריוני לא יהיו אטרקציות ולא מסלולים אלא בתי קברות שמורות טבע ומוזאונים.

3. misHours-מילון המכיל את מספר השעות של הטיול. מאותחל ב-2 ערכים-אופציות:

*7 שעות(חצי יום) *14 שעות (יום שלם).

4.dictOfTrips -מילון התוצאות של הטיולים.מורכב מ-key:מספר סידורי של הטיול ו-value :רשימת האתרים בטיול לפי הסדר אחד אחרי השני.

בתחילה מתבצע סינון ראשוני של האתרים. לתוך הרשימה listOfSites נכנסים כל האתרים מסוננים לפי פרטי המשתמש : גיל מינימלי ומקסימלי, אזור מתאים בארץ ודרכי הגעה המתאימים ליכולות המשתמש(אוטו/אוטובוס). במידה ולא מוכנסים פרטים מסוימים, הרשימה תכלול תוצאות של האתרים ללא סינון הפרטים שלא הוכנסו.

אח"כ מחלקים את תוצאות רשימת האתרים ל-9 רשימות כנגד 9 סוגי האתרים. כל רשימה מכילה את האתרים לפי הסוג שלה .הרשימות הן : tombs, restaurants, coffeeShops , natureReserves, wetAttractions, dryAttractions, wetTrails,dryTrails ו-museuns . מתבצע חישובים של זמן ממוצע שהייה באתר לתוך המשתנה avg החישוב נעשה ע"י זימון הפונקציה GetAvgTime ממחלקת SiteBL שעוברת על כל האתרים ומחשבת את הממוצע.

חישוב נוסף לתוך המשתנה numOfSiteInTrip - ממוצע האתרים לכל סוג טיול:

ממוצע זמן שהיה באתר/מספר השעות בטיול.

תכנון הטיול:

אחרי שהמשתנים ומבני הנתונים מאותחלים, מתחילים בתכנון : קודם כל מחשבים עבור כל סוג טיול את כמות האתרים מכל סוג אתר שמתאימה לו. החישוב נכנס לתוך numOfSitesOfType-מערך בגודל 9 המכיל את כמות האתרים מכל סוג אתר המחושב לפי מטריצת האחוזים.

אתן דוגמא שתסביר יותר את הרעיון : מספר שעות הטיול=14 ממוצע זמן השהייה באתר=1.8 שעות. מספר אתרים בכל טיול=7(14/1.8)

numOfSitesOfType של טיול טבע:

בתי קפה	מסעדות	בתי קבחות	מסלולים בשים	מסלולים טובים	אטרקציות בשות	אטרקציות טובות	שמורות טבע	מוזאונים
0	1	0	1	1	0	0	2	0

ואז פונים לבחירת האתרים בכל סוג טיול : בכל פעם שנבחר אתר, מוסיפים אותו לרשימת התוצאות של סוג הטיול הזה ואותו מחזירים בסופו של דבר למילון התוצאות.בתחילה מוסיפים את האתר הראשון שהוא הבית של המשתמש(בשביל לחשב אח"כ את המרחק)

ואז כל עוד מספר השעות לא עולה על מספר השעות ההתחלתי של הטיול, ממשיכים להוסיף אתרים לטיול ההוספה מתבצעת בצורה כזאת : בכל פעם מגרילים 20 אתרים מכל סוג טיול מרשימת האתרים בסוג. שולחים את רשימת 20 האתרים לפונקציה

ChooseTheShortDistance שמחזירה את האתר שהכי קרוב לאתר הקודם בטיול.

(הרעיון בהגרלה הוא שבהרצה נוספת של התכנון התוצאות יוכלו להיות שונות. דבר זה חשוב מכיון שהוא נותן אינספור של אופציות טיולים למשתמשים. ברגע שלמשתמש הוצעו טיולים בהרצה ראשונית והם לא מצאו חן בעיניו , הוא יוכל להריץ שוב את החיפוש ולמצוא תוצאות שונות).

פונקציה זו מזמנת את הפונקציה GetDistance שמחזירה את האתר ע"י התממשקות ל-Google Maps.הפונקציה ממירה את כל הכתובות לקורדינציות ומעבירה אותם כפרמטר ל-Api של Google Maps הדורש key שרכשתי Google Cloud Platform

(https://console.cloud.google.com/) הוא מחשב את המרחק מכל 20 האתרים לאתר הראשון והפונקציה עוברת על התוצאה ומחזירה את האתר עם המרחק המינימלי.

אתר זה מוסיפים לסוף הרשימה.

בסיום הריצה הפונקציה מחזירה את מילון התוצאות לצד לקוח שם מפרקים אותו ושותלים אותו בתוך התצוגה הגרפית.

15.קוד התוכנית

הגדרת הנתונים ואתחולם:

```
public class Algurithm
{
    #region data
    DBConnection dbCon;
    Random random = new Random();
    //lists cf data
    List<Sites> listOfSites;
    #endregion
    public Algurithm()
    {
        dbCon = new DBConnection();
    }
    //dictionary of the time spend
    Dictionary<bool, int> misHours = new Dictionary<bool, int>()
    {
        { true, 7 },
        { false, 14 }
    };
    //arr of num of the site each type in a trip
    double[] numOfSitesOfType;
```

יצירת הרכב הטיולים -הפונקציה CreateTravels:

```
//יצירת הרכב טיולים ללקוח
public Dictionary<int, List<Sites>> CreateTravels(int MinAge, int MaxAge, bool Car_bus,
    bool halfDay_allDay, int codeSubRegion,string myAddress)
{
    listOfSites = dbCon.GetDbSet<Sites>().Where(i => i.minAge <= MinAge && i.maxAge >= MaxAge
    && i.car_bus == Car_bus
    && i.codeSub_Region == codeSubRegion&& i.statusSite=="active").ToList();
    //dictionary of 5 options to trips
    Dictionary<int, List<Sites>> dictOfTrips = new Dictionary<int, List<Sites>>();
    //list of coffee shops
    List<Sites> coffeeShops = listOfSites.Where(i => i.codeSiteKind == 1).ToList();
    //list of restaurants
    List<Sites> restaurants = listOfSites.Where(i => i.codeSiteKind == 2).ToList();
    //list of tombs-קברים
    List<Sites> tombs = listOfSites.Where(i => i.codeSiteKind == 3).ToList();
    //list of dry trails extra level
    List<Sites> dryTrails = listOfSites.Where(i => i.codeSiteKind == 4 ).ToList();
    //list of wet trails extra level
    List<Sites> wetTrails = listOfSites.Where(i => i.codeSiteKind == 5 ).ToList();
    //list of dry attractions extra level
    List<Sites> dryAttractions= listOfSites.Where(i => i.codeSiteKind == 6 ).ToList();
    //list of wet attractions extra level
    List<Sites> wetAttractions = listOfSites.Where(i => i.codeSiteKind == 7 ).ToList();
    //list of nature reserves
    List<Sites> natureReserves = listOfSites.Where(i => i.codeSiteKind == 8).ToList();
    //list of museuns
    List<Sites> museuns = listOfSites.Where(i => i.codeSiteKind == 9).ToList();
    //mat to calculation number of sites from all of kind
    double[,] matPercents = new double[5, 9] { { 0, 15, 1, 22, 22, 0, 0, 40, 0 },
        { 10, 10, 10, 15, 15, 0, 0, 20, 20 }, { 0, 10, 17, 0, 0, 0, 0, 28, 45 },
        { 0, 15, 0, 17.5, 17.5, 25, 25, 0, 0 }, { 5, 5, 18, 9, 9, 9, 9, 18, 18 } };
    //arr of the names trips kind
    string[] arrNamesKinds = new string[5] { "טבע", "רגוע", "הסטוריה", "אתגרי", "קלאסי" };
    //arr of the list trips kind
    List<Sites>[] arrListKinds = new List<Sites>[9] {coffeeShops,restaurants,tombs,dryTrails,
        wetTrails,dryAttractions,wetAttractions,natureReserves,museuns };
    //number of sites in a trip
    //get the average time to a site
    SiteBL s = new SiteBL();
    int avg = s.GetAvgTime();
```

```
        //local variable
        double misLeft;
        //mis sites in trip
        int numOfSiteInTrip = misHours[halfDay_allDay] / avg;
        for (int i = 0; i < 5; i++)
        {
            //number of sites per type
            numOfSitesOfType = new double[9];
            for (int j = 0; j < 9; j++)
            {
                numOfSitesOfType[j] = Math.Floor(numOfSiteInTrip * (matPercents[i, j] * 0.01));
            }
            //Refresh
            if (i > 0)
            {
                coffeeShops = listOfSites.Where(c => c.codeSiteKind == 1).ToList();
                restaurants = listOfSites.Where(c => c.codeSiteKind == 2).ToList();
                tombs = listOfSites.Where(c => c.codeSiteKind == 3).ToList();
                natureReserves = listOfSites.Where(c => c.codeSiteKind == 8).ToList();
                museuns = listOfSites.Where(c => c.codeSiteKind == 9).ToList();
                if (i == 1)
                {
```

```
                dryTrails = listOfSites.Where(c => c.codeSiteKind == 4
                && c.extraLevel==1).ToList();
                wetTrails = listOfSites.Where(c => c.codeSiteKind == 5
                && c.extraLevel == 1).ToList();
                dryAttractions = listOfSites.Where(c => c.codeSiteKind == 6
                && c.extraLevel == 1).ToList();
                wetAttractions = listOfSites.Where(c => c.codeSiteKind == 7
                && c.extraLevel == 1).ToList();
            }
            if (i == 3)
            {
                dryTrails = listOfSites.Where(c => c.codeSiteKind == 4
                && c.extraLevel == 3).ToList();
                wetTrails = listOfSites.Where(c => c.codeSiteKind == 5
                && c.extraLevel == 3).ToList();
                dryAttractions = listOfSites.Where(c => c.codeSiteKind == 6
                && c.extraLevel == 3).ToList();
                wetAttractions = listOfSites.Where(c => c.codeSiteKind == 7
                && c.extraLevel == 3).ToList();
            }
```

```
                dryTrails = listOfSites.Where(c => c.codeSiteKind == 4 ).ToList();
                wetTrails = listOfSites.Where(c => c.codeSiteKind == 5 ).ToList();
                dryAttractions = listOfSites.Where(c => c.codeSiteKind == 6 ).ToList();
                wetAttractions = listOfSites.Where(c => c.codeSiteKind == 7 ).ToList();
            }
            misLeft = misHours[halfDay_allDay];
            dictOfTrips[i] = new List<Sites>();
            dictOfTrips[i] = SitesRandom(misLeft, numOfSitesOfType, arrListKinds,myAddress);
        }
        return dictOfTrips;

    }
```

הגרלת האתרים -הפונקציה SitesRandom :

```
public List<Sites> SitesRandom(double misHours,double[]arrNumOfSitesToType,
    List<Sites>[] arrListOfSitesToType,string myAddress)
{
    List<Sites> listToRandom = new List<Sites>();
    List<Sites> result = new List<Sites>();
    result.Add(new Sites() { adress = myAddress });
    while(misHours>0)
    {
        for (int i = 0; i < arrNumOfSitesToType.Length; i++)
        {
                if (arrNumOfSitesToType[i] > 0)
                {
               for(int k=0;k<20;k++)
                {
                    listToRandom.Add(arrListOfSitesToType[i][random.Next(arrListOfSitesToType[i].Count)]);
                    arrListOfSitesToType[i].Remove(result[result.Count - 1]);
                }
                result.Add(ChooseTheShortDistance(listToRandom,result[result.Count-1].adress));
                    misHours -= result[result.Count - 1].timeSpend.GetValueOrDefault();
                    arrNumOfSitesToType[i]--;
                }
        }
    }
    return result;
}
```

בחירת המרחק הקצר- הפונקציה ChooseTheShortDistance :

```
private Sites ChooseTheShortDistance(List<Sites> sites,string address)
{
    Task<Sites> s= GetDistance(address, sites);
    return s.Result;
}
```

הבאת האתר בעל המרחק הקצר- הפונקציה GetDistance:

```
public async Task<Sites> GetDistance(string origin, List<Sites> sites)
{
    Sites s = sites[0];
    string[] locationUrls = { BuildUrlForLocationId(origin.Split()) }, idLocations = new string[20];
    foreach (Sites site in sites)
        locationUrls.Append(BuildUrlForLocationId(site.adress.Split()));
    HttpClient http = new HttpClient();

    for (int i = 0; i < idLocations.Length; i++)
    {
        var responseId = await http.GetAsync(locationUrls[i]);

        if (responseId.IsSuccessStatusCode)
        {
            var result = await responseId.Content.ReadAsStringAsync();

            RootLocationBase root = JsonConvert.DeserializeObject<RootLocationBase>(result);
            idLocations[i] = root.results[0].place_id;
        }
    }
    string url = BuildUrlForDistance(idLocations[0], idLocations[1]);
    for (int i = 2; i < 20; i++)
    {
        url += "|place_id:" + idLocations[i];
    }
    var responseDistance = await http.GetAsync(url);

    if (responseDistance.IsSuccessStatusCode)
    {


        RootDistanceBase root = JsonConvert.DeserializeObject<RootDistanceBase>(result);
        string directionMin = root.rows[0].elements[0].distance.text;
        directionMin = directionMin.Replace("mi", "");
        double minDistance = double.Parse(directionMin),min;
        string direction;

        for (int i = 1; i < 20; i++)
        {
            direction = root.rows[0].elements[i].distance.text.Replace("min","");
            min = double.Parse(direction);
            if (min < minDistance)
            {
                minDistance = min;
                s = sites[i];
            }
        }
    }
    return s;
}
```

הפונקציה BuilderForLocationId:

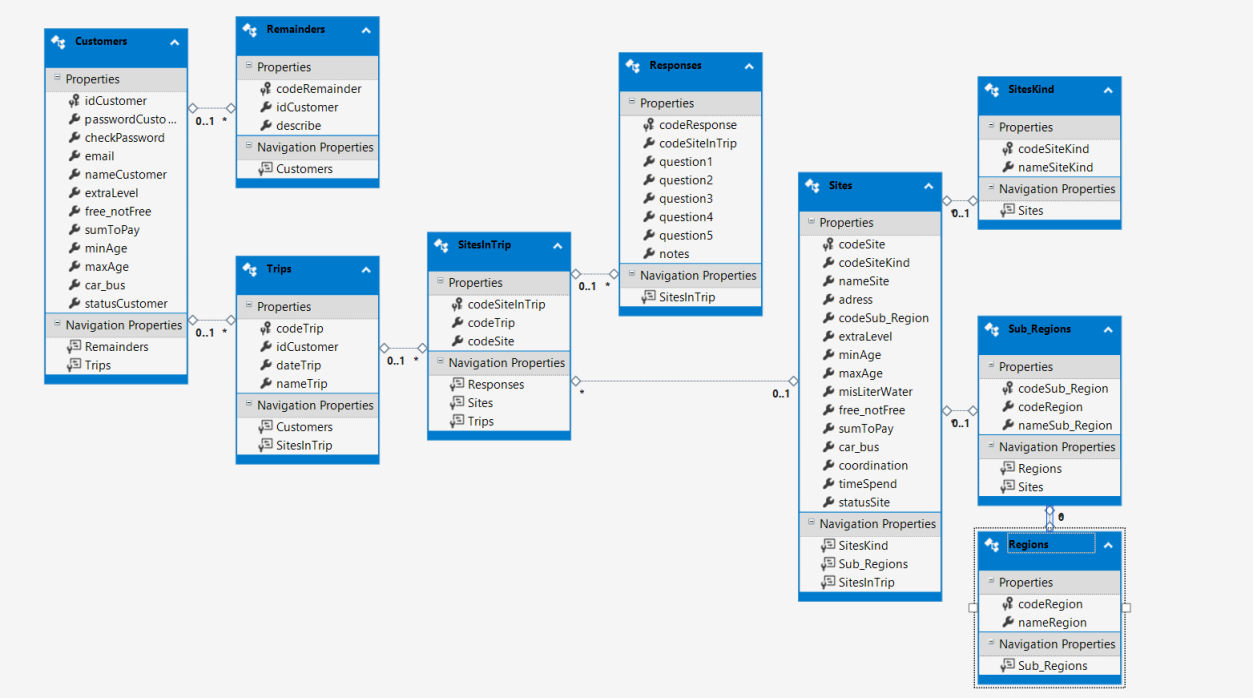
```
static string BuildUrlForLocationId(string[] address)
{
    string location = "";
    string[] locationAsArray;
    locationAsArray =address;

    for (int i = 0; i < locationAsArray.Length; i++)
    {
        if (i < locationAsArray.Length - 1)
            location += locationAsArray[i] + "+";
        else
            location += locationAsArray[i];
    }

    return "https://maps.googleapis.com/maps/api/place/textsearch/json?key=&query=" + location + "&mode=driving&units=imperial&sensor=true";
}
```

הפונקציה BuildUrlForDistance:

```
static string BuildUrlForDistance(string place1, string place2)
{
    string url = "https://maps.googleapis.com/maps/api/distancematrix/json?key=&units=imperial&origins=";
    return url + "place_id:" + place1 + "&destinations=place_id:" + place2;
}
```

:

16.תיאור מסד הנתונים



16.1פירוט הטבלאות בדטה בייס

Customers :

מכילה את פרטי המשתמשים.

פרטי המשתמש נקלטים למערכת בעת כניסת משתמש חדש ומתעדכנים מאזורו האישי על ידו.

פרטי המשתמש הכרחיים בעת תכנון טיול.

CheckPassword-צריך להיות שווה ל- PasswordCustomer.

ExtraLevel-מכיל מספר בין 1 ל-3 (1 :רמת אתגר נמוכה,3 :רמת אתגר גבוהה).

Free_notFree-חינם=true לא בחינם=false.

SumToPay-במקרה והשדה הנ"ל אמת, השדה הזה יהיה מאותחל ב-0.

Car_bus- אוטו=true אוטובוס=false

שם השדה	טיפוס השדה	תאור	NullAble?	מפתח	ברירת מחדל
IdCustomer	int	תעודת זהות		PK	
PasswordCustomer	string	סיסמא			
CheckPassword	string	בדיקת סיסמא			
Email	string	מייל			
NameCustomer	string	שם משתמש			
ExtraLevel	int	רמת אתגר	Nullable		
Free_notFree	bool	חינם/לא בחינם	Nullable		
SumToPay	int	סכום גג לתשלום			
MinAge	int	גיל מינימלי			
MaxAge	int	גיל מקסימלי			
Car_bus	bool	אוטו/אוטובוס			
StatusCustomer	string	סטטוס			active

Regions :

מכילה את האזורים בארץ.

הטבלה מאותחלת באזורים מסוימים ומתאפשר הכנסה של אזורים נוספים דרך תכנון הטיול.

CodeRegion-מספור רץ.

שם השדה	טיפוס השדה	תאור	NullAble?	מפתח	ברירת מחדל
CodeRegion	int	קוד אזור		PK	
NameRegion	string	שם אזור			

Sub_Regions :

מכילה את תתי האזורים בארץ.

הטבלה מאותחלת בתתי אזורים מסוימים ומתאפשר הכנסה של תתי אזורים נוספים דרך תכנון הטיול.

CodeSub_Region -מספור רץ.

שם השדה	טיפוס השדה	תאור	NullAble?	מפתח	ברירת מחדל
CodeSub_Region	int	קוד תת-אזור		PK	
CodeRegion	int	קוד אזור	Nullable	FK	
nameSub_Region	string	שם אזור			

Reminders :

מכילה את תזכורות הטיולים של הלקוח.

התזכורות נקלטות במערכת ע"י המשתמש מאזורו האישי.

CodeRemainder-מספור רץ.

-מחרוזת תיאור של התזכורות-בין כל תזכורת יש פסיק.

שם השדה	טיפוס השדה	תאור	NullAble?	מפתח	ברירת מחדל
CodeRemainder	int	קוד תזכורת		PK	
IdCustomer	int	תעודת זהות לקוח	Nullable	FK	
Describe	string	תוכן התזכורת			

Responses:

מכילה את תגובות הלקוח לטיולים.

התגובות נקלטות במערכת ע"י המשתמש מאזורו האישי.

CodeResponse-מספור רץ.

CodeSiteInTrip-קוד האתר בטיול לו מכניס המשתמש את התגובה.

Question1,2,3…-מספר בין 1 ל-5 שהוא תשובת דירוג לשאלה הראשונה, שניה וכו....

Question5-רוצה לחזור לטייל באתר זה שוב=true לא רוצה=false.

שם השדה	טיפוס השדה	תאור	NullAble?	מפתח	ברירת מחדל
CodeResponse	int	קוד תגובה		PK	
CodeSiteInTrip	int	קוד אתר טיול		FK	
Question1	int	תשובה לשאלה 1	Nullable		
Question2	int	תשובה לשאלה 2	Nullable		
Question3	int	תשובה לשאלה 3	Nullable		
Question4	int	תשובה לשאלה 4	Nullable		
Question5	bool	תשובה לשאלה 5	Nullable		
Notes	string	הערות			

SitesInTrip

מכילה אתרים בטיול מסוים.

הטבלה מתמלאת באתרי הטיול בעת בחירת טיול מסויים ע"י המשתמש.

CodeSiteInTrip-מספור רץ.

שם השדה	טיפוס השדה	תאור	NullAble?	מפתח	ברירת מחדל
CodeSiteInTrip	int	קוד אתר בטיול	Nullable	PK	
CodeTrip	int	קוד טיול	Nullable	FK	
CodeSite	int	קוד אתר	Nullable	FK	

SiteKind

מכילה את סוגי האתרים.

הטבלה מאותחלת בכמה סוגים ולא יכולה להתעדכן.

טבלת האתחול:

מסעדה	1
בית קפה	2
קבר	3
מוזאון	4
שמורת טבע	5
אטרקציה יבשה	6
אטרקציה רטובה	7
מסלול יבש	8
מסלול רטוב	9

CodeSiteKind-מספור רץ.

שם השדה	טיפוס השדה	תאור	NullAble?	מפתח	ברירת מחדל
CodeSiteKind	int	קוד סוג אתר		PK	
NameSiteKind	string	שם הסוג			

Sites

מכילה את פרטי האתרים.

הטבלה מתעדכנת בעת הכנסת אתר חדש/בעת עדכון אתר קיים.

CodeSite-מספור רץ.

ExtraLevel-מכיל מספר בין 1 ל-3 (1 :רמת אתגר נמוכה,3 :רמת אתגר גבוהה).

Free_notFree-חינם=true לא בחינם=false

SumToPay-במקרה והשדה הנ"ל אמת, השדה הזה יהיה מאותחל ב-0.

Car_bus- דרכי הגעה לאתר :באוטו=true באוטובוס=false

Coordination-נקודת הארוך והרוחב של האתר.

שם השדה	טיפוס השדה	תאור	NullAble?	מפתח	ברירת מחדל
CodeSite	int	קוד אתר		PK	
CodeSiteKind	int	קוד סוג אתר		FK	
NameSite	string	שם אתר			
Address	string	כתובת			
CodeSub_Region	int	קוד תת אזור	Nullable	FK	
ExtraLevel	int	רמת אתגר	Nullable		
MinAge	int	גיל מינימלי	Nullable		0
MaxAge	int	גיל מקסימלי	Nullable		120
MisLiterWater	int	ליטרים מים	Nullable		2
Free_notFree	bool	חינם/לא בחינם	Nullable		
SumToPay	int	סכום לתשלום	Nullable		
Car_bus	bool	אוטו/אוטובוס	Nullable		
Coordination	string	קורדינציה			
TimeSpend	double	זמן בילוי	Nullable		*לפי המערך ב-SiteBl
StatusSite	string	סטטוס			active

Trips

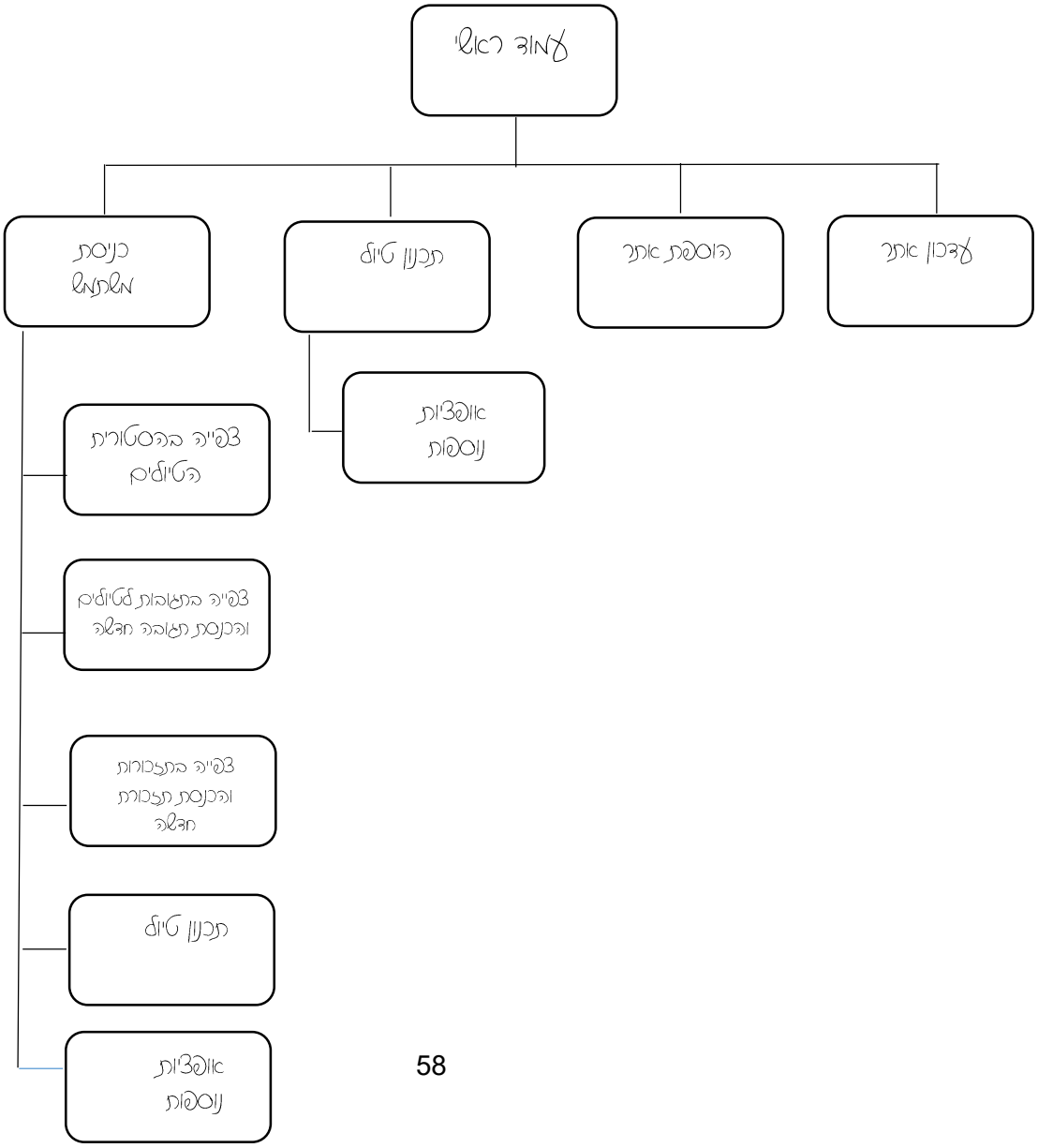
מכילה את פרטי הטיולים ללקוח.

מתווסף טיול לטבלה בעת תכנון טיול.

CodeTri-מספור רץ.

שם השדה	טיפוס השדה	תאור	NullAble?	מפתח	ברירת מחדל
CodeTrip	int	קוד אתר		PK	
IdCustomer	int	קוד סוג אתר	Nullable	FK	
DateTrip	Datetime	שם אתר	Nullable		תאריך של היום
NameTrip	string	כתובת			

17.תיאור מסכים:



18.מדריך למשתמש

בכניסה לאתר, יש למשתמש 2 אפשרויות: או להזדהות ולהיכנס לאזורו האישי או לגשת לתכנון טיול בלי להזדהות. במידה והוא בוחר להכנס לאזורו האישי נדרשת ממנו סיסמא ואם שכח ישנה אפשרות למתן סיסמא חדשה דרך המייל. לאחר ההזדהות יעלה לו הדף הראשי ממנו תהיה לו מספר אופציות: *מעבר לתכנון הטיול מתוך האזור האישי דבר החוסך ממנו להכניס פרטים בעת התכנון.*עדכון פרטיו האישיים.*צפייה בהיסטורית הטיולים שלו והכנסת תגובות לאתרים בטיולים אלו.*צפייה בתזכורות שהכניס והכנסת תזכורת חדשה. בתכנון הטיול (בין אם נכנס דרך האזור האישי ובין אם לא) הוא ממלא פרטים ולוחץ על החיפוש. לאחר החיפוש התוצאות מוצגות לו וישנן אופציות נוספות. ישנה אפשרת נוספת שניתנת בין אם למשתמשים רשומים ובין אם לא והיא הכנסת אתר חדש: המשתמש מכניס כתובת במפה ופרטים נוספים. וכן אפשר לעדכן אתרים קיימים.

19.ניתוח יעילות

בבואי לתכנן את האלגוריתם בפרויקט נתקלתי רבות בשאלות על היעילות. ביצועי הפרויקט חייבים להיות יעילים שכן זה קשור באופן ישיר לזמן התגובה למשתמש: ברגע שהביצועים גרועים, זמן התגובה למשתמש מתארך והאתר "חושב" הרבה זמן-דבר המעצבן את המשתמשים ומוריד את האחוזים בשימוש באתר. על כן נתתי חשיבות רבה ליעילות בביצועים ואף לעיתים העדפתי את היעילות על פני דברים אחרים.

*הפעם הראשונה בה נתקלתי בשאלה זו היתה בתחילת תכנון האלגוריתם: בתחילה רציתי להשתמש באלגוריתם דייקסטרה לצורך חישוב במרחקים הקצרים ביותר בין אתרי הטיול(למידע נוסף- https://he.wikipedia.org/wiki/%D7%90%D7%9C%D7%92%D7%95%D7%A8%D7%99%D7%AA%D7%9D_%D7%93%D7%99%D7%99%D7%A7%D7%A1%D7%98%D7%A8%D7%94)

אך משום שזמן הריצה שלו גבוה מאד והשימוש שהייתי צריכה בו באתר יכול להגיע לתדירות רבה -לא רציתי שזמן הריצה יהיה גדול ומשום כך גם זמן התגובה למשתמש יתארך והביצועים יורעו. ולכן חשבתי להשתמש בדייקסטרה חמדני (שזמן הריצה שלו טוב יותר)אך הוא לא התאים לדרישות של הפרויקט שלי ועל כן שיניתי את האלגוריתם שישתמש ב- Google maps לצורך חישוב במרחקים.

*דבר נוסף ששיפרתי ביעילות של האלגוריתם הוא השימוש ב-API של Google maps:בתכנון הטיול ללקוח אני בעצם מתכננת לו 5 טיולים-מחמישה סוגים שוני כאשר כל סוג מורכב מכמה אתרי טיולים שמחושבים לפי מטריצת אחוזים מיוחדת(פרטים באלגוריתם המרכזי)הבחירה של האתרים מתבצעת לפי

כמה קריטריונים וביניהם קריטריון המרחק לצורך כך השתמשתי ב-API של Google maps כאשר בכל פעם כשרוצים לבחור אתר בודקים מתוך 20 אתרים שעונים לקריטריונים האחרים את האתר שנמצא הכי קרוב לאתר של אותו טיול הקודם. בתחילה זימנתי את הApi שוב ושוב בין כל אתר מתוך ה-20 לאתר הקודם בטיול כדי למצוא את האתר הכי קרוב. אך דבר זה מאט מאד את הביצועים שכן כל פניה לApi לוקחת זמן. ועל כן נברתי במידע על הApi's של Google maps ומצאתי אחד שמאפשר לשלוח את כל 20 האתרים בפעם אחת. דבר זה עוזר לשפר את היעילות בפרויקט.

*פעם נוספת שנתקלתי ביעילות היתה כאשר העדפתי להביא את הרשימה פעם אחת בתחילת כל מחלקה ועליה לבצע את כל השינויים וכך העדפתי את שיפור הביצועים על פני הזיכרון.

20.אבטחת מידע

המידע של המשתמשים מוגן ע"י סיסמא השמורה במערכת.

21.פיתוחים עתידיים

בשלב זה הפרויקט מציע הצעות לטיולים המורכבות רק מאתרים בארץ וכן הוא מתכנן את הטיולים ליום אחד בלבד ללא אפשרויות לינה ואופציות של טיול לכמה ימים.השאיפה שלי היא להגדיל עוד את הפרויקט שיקיף גם נושאים אלו וכן לשכלל את ממשק המשתמש-להוסיף פונקציונליות ונוחות בשימוש.

22.סיכום ומסקנות

ההשוואה בין התכנון הראשוני לתוצר הסופי מגלה כי האתר עומד בדרישות וזהה לתכנון.

ניתן לומר כי הפרויקט תרם לי רבות כסטודנטית וכמהנדסת את היכולת לדעת לבצע עבודת מחקר באופן מעמיק ,למצוא חומרים ,לעיין בהם ,ללמוד את הנושא בצורה מקיפה ובאופן עצמאי , ולדעת ליישם אותו. פעמים רבות עבודת מחקר ארוכה מסתכמת בשורת קוד קצרה ,למדנו לתמצת מחקר רב למספר מועט של שורות קוד פשוט קריא וקצר. הגם שבמהלך הפיתוח עלו שאלות רבות לגבי ביצוע טכני ותכנותי בפרויקט דברים מסוימים שנראו בלתי אפשריים ,התגלו כברי ביצוע על ידי דרכים יצירתיות ומגוונות ובאמצעות שימוש בספריות וחומרים שהתפרסמו בנושא .דבר שהקנה לנו את היכולת להתמודד מול בעיות ולנסות למצוא פתרונות בכוחות עצמינו. במבט לאחור עקב היקפו הגדול של הפרויקט ומגבלות הזמן ,למדו אותנו לחלק את הזמן ואת העבודה באופן יעיל כך שההשקעה תניב פירות ,נוכחנו לדעת עד כמה חשוב לתכנן ולעבוד על פי סדר נכון למדנו ורכשנו כלים רבים במהלך העבודה על הפרויקט :ידע נרחב בשפת C#,ידע נרחב בטכנולגית WebApi ובספריות Angular וAngular Material. מיומנות בבניית אלגוריתם, ניתוח יעילות ומבט רחב על מערכת ממוחשבת. יש לציין כי כתיבת ספר זה תרמה רבות לביסוס הידע שצברנו במהלך הפרויקט.

לסיכום :בהגיעי למועד זה של סיום הפרויקט ובמבט לאחור על הדרך אותה עברתי, שהיה בה רצון לפתח תוכנה ברמה המקצועית ביותר ,ניתן לומר כי בס"ד הגעתי להבנה מעמיקה ביסודות התכנות ולידע כיצד

ללמוד ולהכיר טכנולוגיות חדשות. הפרויקט קידם אותי באופן ניכר מבחינה מקצועית ואישית. פרויקט זה אכן היווה אבן דרך משמעותית בהתמקצעות בתחום הנדסת תוכנה.

23.ביבליוגרפיה

ויקיפדיה

Microsoft

StackOverFlow

GitHub

Angular

GeeksToGeeks