**­­מצא לי**

1. מבוא

חפשנו רעיון מקורי לפרויקט. שימושי, יעיל וחכם.

רעיון שיביא תועלת לאנושות ויקל על האנשים בחיי היום יום . ומאידך מבחינה מקצועית, רצינו פרויקט חדשני המשתמש בטכנולוגיות המתקדמות והחדשניות ביותר.

הלכנו ברחוב חפשנו רעיונות,

ובין לבין גם אבדות.

כי אם כבר מחפשים רעיונות –

ננצל את הזן לחיפוש אבדות.

חפשנו רעיונות ואבדות –

ומצאנו...

מצאנו אבדה או מצאנו רעיון?

מצאנו רעיון איך למצוא את האבדות...

נכין ממשק המקשר בין בעל האבדה ומוצא האבדה. בכך נקל הן על הצד המחפש – שיקבל סיוע בחיפוש והן על המוצא – שיוכל להשיב האבדה לבעליה בקלות ובנוחות.

1. מדריך למתכנת
   1. תיאור המערכת

האתר 'מצא לי' הוא אתר המקשר בין בעל האבדה למוצא האבדה.

מוצא –

המערכת תסייע למוצא אבדה להשיב את האבדה לבעליה. משתמש יכול להיכנס למערכת כמוצא אבדה, להשאיר את פרטיו האישיים כולל כתובת אימייל ליצירת קשר – בכדי שבעל האבדה יוכל ליצור עמו קשר ולקבל אבדתו.

כאשר מאבד מכניס את פרטיו האישיים, פרטיו נקלטים במערכת בטבלה זמנית. המערכת תשלח הודעה עם שם משתמש לכתובת שהמשתמש עדכן – בכדי לאמת את פרטי ההתקשרות, שבעל האבדה יוכל ליצור קשר עם מוצא אבדתו.

לאחר שהמשתמש קיבל שם משתמש באפשרותו להיכנס למערכת, כעת פרטיו יעברו לטבלת מוצאים והוא יוכל להשאיר את פרטי אבדתו. כאשר אבדה זו תתבקש יקבל המאבד את פרטי המוצא. כך יוכל המוצא להשיב את האבדה לבעליה בקלות ובנוחות.

מאבד –

המערכת תסייע למאבד למצוא את אבדתו בקלות. משתמש יכול להיכנס למערכת כמאבד, להשאיר את פרטיו האישיים כולל כתובת אימייל ליצירת קשר – בכדי שמוצא אבדתו יוכל ליצור עמו קשר ולהחזיר לו את אבדתו.

כאשר מאבד מכניס את פרטיו האישיים, פרטיו נקלטים במערכת בטבלה זמנית. המערכת תשלח הודעה עם שם משתמש לכתובת שהמשתמש עדכן – בכדי לאמת את פרטי ההתקשרות, שמוצא האבדה יוכל ליצור קשר עם המאבד.

לאחר שהמשתמש קיבל שם משתמש באפשרותו להיכנס למערכת, כעת פרטיו יעברו לטבלת מאבדים והוא יוכל להשאיר את פרטי אבדתו. כאשר אבדה זו תתבקש יקבל המאבד את פרטי המוצא. כך יוכל המוצא להשיב את האבדה לבעליה בקלות ובנוחות.

חיפוש אבדות –

המערכת תאפשר לציבור הרחב לחפש את אבדותיהם. ניתן לחפש את האבדה במפת האבדות או על פי סימנים.

מפת האבדות – מפת האבדות מציגה למשתמש את כל האבדות שנמצאו על פי מיקומן במפה. האבדות מחולקות על פי קטגוריות, לכול קטגוריה icon שונה. בכדי להקל על המשתמש את החיפוש במפה המשתמש יכול לבחור עיר ולקבל ב'זום' את האזור המבוקש. בלחיצה על אבדה מסוימת ניתן לצפות בפרטי האבדה ובמידת הצרך לקבל את פרטי המוצא.

כאשר אזור האבדה לא ידוע למשתמש ניתן לחפש על פי סימנים. בחיפוש על פי סימנים נדרש המשתמש להשלים את פרטי אבדתו: קטגוריה, תיאור, צבע, תאריך האבדה והערות. בלחיצה על 'חיפוש' יופעל מנגנון החיפוש של המערכת.

כאן נדרשנו לחשיבה מרובה היות ובני האדם שונים בטבעם ובראייתם לא ניתן להשוות באופן מוחלט בין פרטי האבדה שהגדיר המוצא לאלו של המאבד. לדוגמא צבע האבדה – כחול כהה / בהיר / טורקיז / תכלת כול אחד מתאר אחרת. גם החלוקה לקטגוריות שנויה במחלוקת. תאריכים – לא תמיד זוכרים דיוק מתי נאבד וכולי.

לכן מנגנון החיפוש מפלטר באופן כזה שלכול צבע מוגדרת קבוצה – אם הצבע המבוקש מופיע ברשימה נוסיף זאת כאופציה. תאריך נחשב מטווח של שבוע קודם. ובשקלול הנתונים – מספיקה התאמה של 60% בין הגדרות המוצא למאבד בשביל לתאם ביניהם.

האזור האישי –

כניסה לאזור האישי נעשית באמצעות מספר זהות. באזור האישי ישנה אפשרות לעריכת פרטים אישיים, צפייה במציאות שמצאתי ובאבדות שנאבדו לי.

* 1. זרימת המידע במערכת:
     1. Sequence Diagram

TODO//

* + 1. Use Case



מערכת

העברה לארכיון

מחיקת אבדות שנמצאו

קישור בין מאבד למוצא

שליחת קוד משתמש



מוצא

<<include>>

רישום מציאה

פרטי מציאה

פרטי מוצא

<<include>>

>>Uses>>



מאבד

<<include>>

רישום אבדה

פרטי אבדה

פרטי מאבד

<<include>>

<<include>>

<<extend>>

חיפוש אבדה

<<include>>

צפייה באזור המבוקש במפה

סינון לפי קריטריונים

<<include>>

צפייה במפת האבדות

בחירת אזור חיפוש

* + 1. Data flow

TODO//

* 1. תכנות
     1. תיאור

המערכת נכתבה בשפות typescript, C# בסביבת .visual studio 2017

המערכת מתחברת ל - Google Maps לצורך קבלת מיקומי אבדות ומציאות והצגת מפת מציאות.

שימוש בטכנולוגיות ובספריות חדשות ומתקדמות בתוכם:

Web API, Angular6, Bootstrap

**F**rontEnd: נכתב בAngular6 תוך שימוש בTypeScript, העיצוב נעשה בעזרת Css.Bootsrap.

BackEnd: נכתב בשפת C# תוך שימוש בטכנולוגית WebApi עבור קריאות שרת לקוח.

DataBase: ,SQLהגישה למסד הנתונים: תעשה בעזרת טכנולוגיית Entityframework.

* + 1. חלופות שפת מימוש

שפות וטכנולוגיות צד שרת:

* WCF - Windows Communication Foundation

מודל מערכת וממשקי API ב - NET Framework. ליצירת תקשורת בין יישומים ובניית יישומים מבוזרים מכווני שירותים, על ידי הגדרות בקונפיג או בקוד נוכל להחלטי מהי צורת התקשורת ובאיזה פרוטוקול ומהם התכונות שיש בתקשורת.

קיימת אבטחה מובנית. WCF תומך בפרוטוקולים שונים של רשתות מקומיות ומרוחקות אך הוא כבד מאוד.

* ASP.NET MVC

טכנולוגיה מתקדמת מבית מיקרוסופט לפיתוח אפליקציות .WEB הטכנולוגיה שמה דגש על פיתוח מהיר,  שימוש בארכיטקטורה נכונה והפרדת שכבות, שילוב ספריות קוד פתוח.

מבחינת אבטחה והגנה קיימים פילטרים שונים, יש תמיכה מובנית בבדיקות אוטומטיות וקיימת תשתית לניהול משתמשים (Membership).

* Wep Api

תוכנת שרת המספקת אינפורמציה לתוכנת לקוח. המידע שמספקת הוא על פי שימוש

במסד נתונים הממוקם בשרת. בשרת ישנם שתי חלקים: Model, Controller ואת

ה View מממשת תוכנת צד לקוח. ניתן לאכסון רק בשרת IIS ותומך רק בפרוטוקול

.HTTP

שפות וטכנולוגיות צד לקוח:

* JavaScript + jQuery

jQuery היא תשתית פיתוח עבורJavaScript שהופכת פעולות נפוצות ב JavaScript

לקלות הרבה יותר, עוזרת לשפר את חווית המשתמשים באתרים ומטפלת בהבדלים

באופן המימוש של JavaScript בין דפדפנים שונים.

הפעולות ב- jQuery מהירות ויעילות בביצועיהן כמו כן בעקבות הפופולאריות הרבה

ש jQueryצברה בשנים האחרונות קיים תיעוד רחב עבורה ומספר התוספים שנכתבו

עבורה רב.

* Angular

Angular זה ה framework של JavaScript שמשמש ליצירת אפליקציות מבוססות דף יחיד. Angular מוגדר framework בגלל שהוא מספק פונקציות מן המוכן. כלומר במקום שהמפתחים יאלצו לעמול על קוד מתוחכם בכדי לענות על הצרכים הרבים והמגוונים

של אפליקציות מודרניות הם יכולים להשתמש בפונקציות של framework שעושות

את העבודה בהרבה פחות מאמץ וזמן. ומאפשרת התנהגות חלקה וחווית משתמש

גבוהה במיוחד מאחר ולא כל הדף צריך להתרענן.

בחרנו להשתמש בצד שרת ב Web Api ולא ב WCF, מכיוון שהתוכנה מספקת שירות דרך רשת האינטרנט וממילא תעבוד רק בפרוטוקול HTTP. ואין צורך להפעיל מנוע חזק כל כך כמו .WCF Web Api עונה על הצרכים שלנו ומספק אבטחה מקסימלית.

בצד לקוח בחרנו להשתמש ב Angular בגלל הפופולאריות שלה, רצינו לרכוש מיומנות וידע רחב

בשפה שנפוצה ובעלת יתרונות תכנותיים ואוסף פונקציות ענק שהיא מספקת.

* + 1. טכנולוגיות מתקדמות:
* Web Api – טכנולוגיה עבור צד server במערכת, ממשק API של אינטרנט הוא ממשק תכנות יישומים עבור שרת אינטרנט או דפדפן אינטרנט. ממשק API של אינטרנט בצד ה – server הוא ממשק תכנותי המורכב מנקודת קצה אחת או יותר של נקודות קצה חשופות למערכת. הודעה מוגדרת לבקשה המתבטאת בדרך כלל ב – JSON או XML הנחשפת דרך האינטרנט.

קטעי קוד לדוגמא:

* קבלת בקשה לקבלת רשימת הערים:

[Route("api/Losty/GetCity")]

[HttpPost]

public List<City> GetCity()

{

SqlCommand cmd = ConnectSql("Select \* from City");

SqlDataReader reader = cmd.ExecuteReader();

List<City> resultCity = new List<City>();

City city = new City();

while (reader.Read())

{

resultCity.Add(city.Initialization(reader));

}

connection.DisConnectSql();

return resultCity;

}

* קבלת בקשה להוספת מאבד:

[Route("api/Losty/InsertLoss")]

[HttpPost]

public string InsertLoss(Loss loss)

{

loss.LoseID = GetUserID(loss.LoseID);

SqlCommand cmd = ConnectSql("Insert into Loss Values(@LoseID , @CategoryCode , @LossDesc , @LossColor , @LossDate , @Loss\_X , @Loss\_Y , @Remarks, @StatusCode , @Date)");

try

{

cmd.Parameters.AddWithValue("@LoseID", loss.LoseID);

cmd.Parameters.AddWithValue("@CategoryCode", loss.CategoryCode);

cmd.Parameters.AddWithValue("@LossDesc", loss.LossDesc);

cmd.Parameters.AddWithValue("@LossColor", loss.LossColor);

cmd.Parameters.AddWithValue("@LossDate", loss.LossDate);

cmd.Parameters.AddWithValue("@Loss\_X", loss.Loss\_X);

cmd.Parameters.AddWithValue("@Loss\_Y", loss.Loss\_Y);

cmd.Parameters.AddWithValue("@Remarks", loss.Remarks);

cmd.Parameters.AddWithValue("@StatusCode", loss.StatusCode);

cmd.Parameters.AddWithValue("@Date", loss.Date);

cmd.ExecuteNonQuery();

return "Inserting Loss Seccessfuly";

}

catch (Exception e)

{

return "Exception Occre while inserting loss:" + e.Message + "\t" + e.GetType();

}

finally

{

connection.DisConnectSql();

}

}

* קבלת בקשה לעדכון סטטוס של אבדה:
* Angular

[HttpPost]

[Route("api/Losty/ChangeStatus")]

public string ChangeStatus(Found \_found)

{

int status = 3;

SqlCommand cmd = ConnectSql(string.Format("UPDATE Found SET StatusCode = '{0}' Where FoundCode = '{1}'", status, \_found.FoundCode));

try

{

cmd.Parameters.AddWithValue("@StatusCode", 3);

cmd.Parameters.AddWithValue("@FoundCode", \_found.FoundCode);

cmd.ExecuteNonQuery();

return \_found.FoundCode + "Data updated!";

}

catch (Exception ex)

{

return ex.Message;

}

finally

{

connection.DisConnectSql();

}

}

תשתית פיתוח שבשונה מול תשתית פיתוח אחרת מכתיבה לנו גם איך לפתח. מעבר לטכנולוגיה היא מכתיבה לנו צורת פיתוח לצד לקוח המתבססת על TypeScript.

TypeScript זוהי שפת פיתוח בצד לקוח הנותנת לנו כלים של עבודה עם types, ואז ממילא ניתן לבצע קומפילציה – שלב בו TypeScript מבצע המרה של הקוד לקוד JS מכיוון שרוב הדפדפנים אינם תומכים כרגע בתקן החדש של שפה זו.

מתכנתי Angular יצרו עבורנו את האפשרות ליצור אפליקציות של S.P.A – single page application כלומר רק חלק מסוים בדף מתרענן ומתחלף וכל שאר החלקים נשארים אותו הדבר. דבר זה מאפשר גלישה מהירה וחלקה אשר משפרת את חווית המשתמש.

בכול קומפוננטה (רכיב המיועד לתצוגה המכיל את כללי התצוגה ואת הניהול שלה)

בה רוצים להציג תוכן כלשהו במאפיין template – מאפיין זה אומר ל Angular איזה תוכן HTML להציג עבור קומפוננטה זו.

ה template מכיר את הקומפוננטה ואת כל השדות והפונקציות הכתובים בה. אנגולר משתמש בשיטת ה two way data binding. כלומר כל שינוי שנעשה ב template משפיע ישירות על הקופוננטה ולהיפך.

דוגמא לחיבור של הקומפוננטה ל template עליו היא אחראית:

import { Component, OnInit, Input } from '@angular/core';

import { Found, Category, Color, WebApiService, Person } from '../../Service/web-api.service';

@Component({

selector: 'app-personal-found',

templateUrl: './personal-found.component.html',

styleUrls: ['./personal-found.component.scss']

})

export class PersonalFoundComponent implements OnInit {

@Input()

userId: string;

person: Person = new Person();

category: Array<Category> = new Array();

Colors: Array<Color> = new Array<Color>();

founds: Array<Found> = new Array();

constructor(private \_WebApiService: WebApiService) { }

דוגמא להטמעת הנתונים ב template:

<table class="table">

<thead>

<tr>

<th scope="col">הערות</th>

<th scope="col">תאריך המציאה</th>

<th scope="col">צבע המציאה</th>

<th scope="col">תיאור</th>

<th scope="col">קטגוריה</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let f of founds" style="text-align: center">

<td>{{f.Remarks}}</td>

<td>{{f.FoundDate | date:'yyyy/MM/dd'}}</td>

<td>{{f.color}}</td>

<td>{{f.FoundDesc}}</td>

<td>{{f.Category}}</td>

<td>

<span class="oi oi-trash"></span>

</td>

</tr>

<tr \*ngIf="loading">

<td colspan="5" align="center">Fetching Employees</td>

</tr>

</tbody>

</table>

* BootStrap

ספרייה שנועדה לעיצוב ומאפשרת רזולוציה נכונה לכל סוג של מסך.

* Google Maps

שימוש ב service של google החושף את ה API של המפות שלו. מאפשר לנו להציג מפות, לקבל נקודות מהמפה, לעבד את המידע בשרת, להחזיר ל client ולהציג את המיקום במפה. שימוש של ה – service נעשה אצלנו בצד ה – client.

* + 1. עקרונות התכנות
* חלוקה לשכבות - Tier Architecture-4 - בפרויקטים שמבוססים על Database מומלץ להשתמש בגישה של מודל השכבות, בארכיטקטורה זו קיימת הפרדה בין השכבות השונות בפרויקט.

היתרון העיקרי בחלוקה לשכבות הוא תחזוקה קלה, במיוחד במערכות גדולות, כך שניתן יחסית בקלות להחליף כל שכבה בלי לגעת בשכבות האחרות כאשר נרצה לעשות שינויים בתוכנית.

* כיום הגישה לגבי אפליקציות web היא כמה שיותר כתיבה בצד ה client מכיוון שכל פניה לשרת מכבידה על המערכת ומאריכה את זמן פעילותה, מה שגורם לקוצר רוח מצד המשתמש. לכן השתדלנו שרוב הנתונים שלא צריכים עיבוד בשרת ימומשו בצד ה client.
* מסכי המערכת נכתבו בצורה פשטנית וברורה ביותר לעין, כך שקלים מאד לשימוש, ונותנים חווית משתמש.
  + 1. בעיות ופתרונן

סימנים, שינוי סטטוס

TODO//

* + 1. אבטחה
* לא ניתן להשאיר פרטי אבדה שנמצאה ללא אימות כתובת המייל - במקרה שמוצא אבדה נותן כתובת מייל לא מדויקת הוא לא יוכל להשאיר את פרטי האבדה במערכת כי לא יקבל סיסמת משתמש.
* במקרה שמשתמש חדש מנסה להיכנס כמשתמש רשום, המערכת תפנה אותו לדף ההרשמה לאתר.
* במקרה שבעת כניסת משתמש רשום למערכת הסיסמא אינה תואמת לכתובת המייל המערכת תציג הודעת שגיאה ולא תאפשר לו כניסה.
* הסיסמא תהיה מוסתרת.
  + 1. שרידות המערכת ותיעוד

שרידות, ביצועים התמודדות עם עומסים: צד השרת מריץ IIS Express המסוגל להתמודד עם מספר קריאות בו זמנית. גם עומס על שרת ה- SQL אינו צפוי בסדר גודל כזה של אתר כי הוא בנוי להתמודדות בהצלחה עם עומסים כבדים בהרבה.

* 1. ארכיטקטורה ובניה
     1. תיאור הפרויקט
* **שכבת הנתונים (Data Access Layer - DAL) -** שכבה זו מורכבת ממקור נתונים –מסד הנתונים שלנו, וממערכת תוכנה entity framework אשר תפקידה לקרוא את המידע הנדרש למערכת, לשמור את העדכונים, ולהוסיף מידע חדש או למחוק פרטי מידע קיימים.
* **שכבת הלוגיקה העסקית (BL - Business Logic) –** השכבה שאמונה על הלוגיקה של המערכת, עוסקת בעיבוד המידע, בחישובים שונים ושליחת המידע לשכבת התצוגה.

בשכבה זו נממש את הפונקציונאליות של המערכת.

* **BE – Entities Business** **–** בשכבה זו קיימות רק מחלקות של אובייקטים שמקבילות למחלקות בשכבת ה DAL – לצורכי אבטחה, הגנה על שכבת ה DAL מפני חשיפה ולצורך וולידציות שונות על מידע המגיע מה -client .
* **שכבת תצוגה (Presentation Layer) –** שכבת ממשק המשתמש (UI- User Interface).

שכבת התצוגה מדפי HTML באתר אינטרנט.

השכבה מתקשרת בין המשתמש לבין שאר השכבות של המערכת.

* + 1. תיאור קוד פונקציות ומחלקות

מחלקות ה – Entity framework: מחלקות מקבילות לטבלאות שב Database.

Connection: מחלקה המכילה את פונקציות החיבור ל – DataBase:

public class Connection

{

SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["SQLConn"].ConnectionString);

/// <summary>

/// Connect to SQL

/// </summary>

/// <param name="query">Query</param>

/// <returns>command result</returns>

public SqlCommand connectToSql(string query)

{

conn.Open();

SqlCommand command = new SqlCommand(query, conn);

return command;

}

/// <summary>

/// Disonnect from SQL

/// </summary>

public void DisConnectSql()

{

conn.Close();

}

}

LostyController: ­­­מחלקת ה -  **Business Logic –** אמונה על הלוגיקה של המערכת, עוסקת בשליפת נתונים מהשרת, בעיבוד המידע, בחישובים שונים ושליחת המידע לשכבת התצוגה. המחלקה מורכבת מפונקציות, כל פונקציה מקבלת בקשה מה client מעבדת את הנתונים ושולחת בקשה - SqlCommand לשרת. מקבלת תגובה מהשרת מעבדת את הנתונים ושולחת תגובה ל – client.

* שליחת אימייל – הפונקציה מקבלת Person שנרשם למערכת, בכדי לאמת את כתובת האיממיל המערכת תשלח לו שם משתמש שישמש אותו לכניסה למערכת כךיוכל להשאיר פרטי מציאות ואבדות.

הפונקציה משתמשת בפונקציית עזר ליצירת שם המשתמש. שם משתמש מורכב מאות ראשונה של שם פרטי + אות ראשונה של שם משפחה + מספר רנדומלי. (בכדי לשמור על שם משתמש ייחודי לכל אחד.

המשתמש מכניס את שמו בעברית ושם המשמש מורכב מאותיות באנגלית ומספרים בלבד. את ההמרה מעברית לאנגלית עשינו באמצעות קוד אסקי.

///פונקציה ליצירת שם משתמש:

/// <summary>

/// GetUserName

/// </summary>

/// <param name="reciveName">reciveName</param>

/// <returns>User Name</returns>

[HttpGet]

[Route("api/Losty/GetUserName")]

public string GetUserName(string reciveName)

{

string[] characters = { "a", "b", "g", "d", "h", "v", "z", "ch", "t", "i", "c", "c", "l", "m", "m", "n", "n", "s", "a", "p", "p", "th", "th", "k", "r", "s", "t" };

string[] fullName = reciveName.Split(' ');

string userName = "";

int i = 0, ascci;

foreach (string word in fullName)

{

ascci = (int)Convert.ToChar(word[0]);

userName += characters[ascci - 1488];

i++;

}

Random random = new Random();

userName += random.Next().ToString();

return userName;

}

שליחת האימייל מתבצעת באמצעות שרת SmtpServer.

///פונקציה שליחת אימייל.

/// <summary>

/// SendEmail

/// </summary>

/// <param name="person">person</param>

/// <returns>answer</returns>

[HttpPost]

[Route("api/Losty/SendEmail")]

public string SendEmail(Person person)

{

mail\_core mail = new mail\_core();

string userName = GetUserName(person.PersonName);

mail.NewMail(person.PersonEmail, person.PersonName, "RS.Losty@gmail.com", "Losty", "LOSTY שם משתמש כניסה למערכת ", "שלום ל" + person.PersonName + " שם המשתמש שלך: " + userName, "");

mail.smtpServerSettings("smtp.gmail.com", 587, "RS.Losty@gmail.com", "losty.1234", true);

InsertUserName(person.PersonID, person.PersonEmail, userName);

return mail.sendMail();

}

* פונקציה לאימות שם משתמש: הפונקציה מקבלת User ובודקת האם שם המשתמש קיים בטבלת User – אם כן -   
  אם ה – User מופיע בטבלת Person ימחקו נתוני ה – User מטבלת Person ויעברו לטבלה המתאימה (Find / Lose).

אם המשתמש לא מופיע בטבלת Person אזי אם הוא מופיע בטבלה המבוקשת – שם המשתמש תקין והפונקציה תחזיר true. אחרת תעתיק את הנתונים לטבלה הרלוונטית.

///אימות שם משתמש

/// <summary>

/// VerifyUserName

/// </summary>

/// <param name="user">user</param>

/// <returns>is insert?</returns>

[Route("api/Losty/VerifyUserName")]

[HttpPost]

public bool VerifyUserName(string[] user)

{

string userID = "";

Person person = new Person();

//Get User ID

string searchUser = string.Format("Select UserID From [User] Where UserName = '{0}' AND UserEmail = '{1}' ", user[1], user[2]);

SqlCommand cmd = ConnectSql(searchUser);

SqlDataReader reader = cmd.ExecuteReader();

if (reader.Read())

{

userID = reader["UserID"].ToString();

}

connection.DisConnectSql();

//Search if User appear in Person Table? Delete & Add To Find/Lose Table : Search in Find/Lose Tables

string searchInPersonTable = string.Format("Select \* From Person Where PersonID = '{0}' ", userID);

bool checkInsert = false;

cmd = ConnectSql(searchInPersonTable);

reader = cmd.ExecuteReader();

if (reader.Read())

{

person = person.Initialization(reader, PERSON);

connection.DisConnectSql();

if (DeleteUser(person) == true)

{

if (user[0].Equals(\_FIND))

checkInsert = InsertFind(person);

else

checkInsert = InsertLose(person);

}

}

//Search in Find / Lose Tables if User Find/Lose appear in the different table copy to the relevant table

else

{

connection.DisConnectSql();

string searchInFindTable = string.Format("Select \* From Find Where FindID = '{0}' ", userID);

cmd = ConnectSql(searchInFindTable);

reader = cmd.ExecuteReader();

if (reader.Read())

{

if (user[0].Equals(\_FIND))

{

checkInsert = true;

}

else

{

person = person.Initialization(reader, FIND);

}

}

connection.DisConnectSql();

if (checkInsert == false)

{

string searchInLoseTable = string.Format("Select \* From Lose Where LoseID = '{0}' ", userID);

cmd = ConnectSql(searchInLoseTable);

reader = cmd.ExecuteReader();

if (reader.Read())

{

if (user[0].Equals(\_LOSE))

{

checkInsert = true;

}

else

{

person = person.Initialization(reader, LOSE);

connection.DisConnectSql();

checkInsert = InsertFind(person);

}

}

else

{

if (person != null)

{

connection.DisConnectSql();

checkInsert = InsertLose(person);

}

}

}

}

return checkInsert;

}

* חיפוש אבדות – הפונקציה מקבלת סימנים של האבדה ומחזירה רשימה של מציאות רלוונטיות בהתאמה של 70%.

///search

/// <summary>

/// GetRelevantColors

/// </summary>

/// <param name="color">color</param>

/// <returns>relevant colors</returns>

public string GetRelevantColors(int color)

{

List<int> resultColors = new List<int>();

SqlCommand cmd = ConnectSql(string.Format("Select \* From Color Where ColorCode = '{0}' ", color));

SqlDataReader reader = cmd.ExecuteReader();

Color \_color = new Color();

\_color.Initialization(reader);

connection.DisConnectSql();

return \_color.RelevantColors;

}

/// <summary>

/// Get Founds by signs

/// </summary>

/// <param name="signs">lose's signs for search</param>

/// <returns>list of relevant founds</returns>

[Route("api/Losty/GetFounds")]

[HttpPost]

public List<Found> GetFounds(Signs signs)

{

SqlCommand cmd = ConnectSql("Select \* From Found");

SqlDataReader reader = cmd.ExecuteReader();

List<Found> resultFounds = new List<Found>();

Found found = new Found();

while (reader.Read())

{

resultFounds.Add(found.Initialization(reader));

}

connection.DisConnectSql();

int flag, i, j;

for (i = resultFounds.Count() - 1; i >= 0; i--)

{

flag = 0;

if (resultFounds[i].CategoryCode == signs.Category)

flag += 20;

if (signs.Description != " ")

{

string[] description = signs.Description.Split(' ');

j = 0;

foreach (string word in description)

{

string[] FoundDesc = resultFounds[i].FoundDesc.Split(' ');

foreach (string desc in FoundDesc)

{

j += word.Contains(desc) == true ? 1 : 0;

}

}

if (j != 0)

flag += 20;

}

else

{

flag += 10;

}

if(GetRelevantColors(signs.Color).Contains(resultFounds[i].FoundColor.ToString()))

flag += 20;

if (resultFounds[i].FoundDate >= signs.Date.AddDays(-7))

flag += 20;

}

* + 1. מבנה נתונים

מבנה הנתונים נשמר ב – SQL SERVER.

צורת התקשרות עם ה – DB:

את הקישור ל – Database ואת התקשורת אתו עשינו באמצעות Entity framework.

עם ה – Database תקשרנו באמצעות SqlCommand לשליחת שאילתות SQL, קריאת הנתונים ב SqlDataReader

---צילומי מסך של הטבלאות

* 1. בדיקות תוכנה
     1. טבלת בדיקות מסכמת

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **סוג הבדיקה** | **מספר הבדיקה** | **תיאור** | **עבר / נכשל** | **באגים** |
| בדיקת שפיות - sanity | 1 | בדיקת תהליך של כניסה למערכת | עבר | -- |
| בדיקת GUI -  ממשק משתמש | 2 | בדיקת הכנסת נתונים לשדות – שימוש בתווים מיוחדים. | עבר | -- |
| בדיקות פונקציונליות - Functional | 3 | אפשרות לחפש אבדה עפ"י סימנים. | עבר | -- |
| בדיקות שימושיות - Usability | 4 | מעבר תקין לדפים מהתפריט שבממשק המשתמש. | עבר | -- |

* + 1. Test Case
       1. בדיקת שפיות – sanity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bugs** | **Pass / Fail** | **Actual Result** | **Excepted Result** | **Step** |
| -- | Pass | open | Drop-down open | Click on 'Find' |
| -- | Pass | show | Show form with user details | Check register |
| -- | Pass | confirmed | Get confirmation message that the data saved | Complete user's details & click save |
| -- | Pass | Get email | Drop-down open | Waiting to get user name to your email |
| -- | Pass | Open | Drop-down open | Click on 'Find' |
| -- | Pass | show | Show entry form | Check entry |
| -- | Pass | enable | Verify user name - enable to add found details | Enter user name and email address |
| -- | Pass | confirmed | Get confirmation message that the data saved | Complete found's details & click save |

* + - 1. בדיקת GUI – ממשק משתמש

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bugs** | **Pass / Fail** | **Actual Result** | **Excepted Result** | **Step** |
| -- | Pass | Disable save | Disable to save data | Enter '+' |
| -- | Pass | Disable save | Disable to save data | Enter '1' |
| -- | Pass | Disable save | Disable to save data | Enter '@' |
| -- | Pass | Disable save | Disable to save data | Enter '12@' |
| -- | Pass | Disable save | Disable to save data | Enter '.co' |
| -- | Pass | Disable save | Disable to save data | Enter 'dw@' |
| -- | Pass | Confirmation message | Get confirmation message that cannot send email | Enter 'abd@gmail.com' |

* + - 1. בדיקת פונקציונליות - Functional

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bugs** | **Pass / Fail** | **Actual Result** | **Excepted Result** | **Step** |
| -- | Pass | open | Drop-down open | Click on search |
| -- | Pass | show | Show form signs details | Choose click by signs |
| -- | Pass | Get result | Get list of optional losses | Enter your signs and click search |
| -- | Pass | Get result | Get more optional list | Try to change the signs and search again |

* + - 1. בדיקת שימושיות – Usability

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bugs** | **Pass / Fail** | **Actual Result** | **Excepted Result** | **Step** |
| -- | Pass | open | Drop-down open | Click on Find |
| -- | Pass | show | Open entry form | Choose entry |
| -- | Pass | open | Drop-down open | Click on search |
| -- | Pass | show | Show losses map | Choose map |

* + 1. טבלת באגים

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Is correct?** | **Test type** | **Test number** | **Description** | **Bug number** |
| -- | -- | -- | -- | -- |

1. מדריך למשתמש (--- צילום מסך)

ברוכים הבאים למערכת "מצא לי" בכניסה למערכת תבחרו לאיזה ממשק ברצונכם להיכנס.

מאבד - מוצא

---

לכניסה פעם ראשונה בחרו ב"הרשמה"

---

אימייל ששלחה המערכת עם שם משתמש:

---

באם הינכם רשומים במערכת "מצא לי" בחרו ב"כניסה"

---

חיפוש אבדה – יש להשלים סימנים לחיפוש אבדה

---

חיפוש במפת האבדות

חיפוש על פי סימנים

כניסה לאזור האישי – הכניסה לאזור האישי נעשית באמצעות תעודת זהות. באזור האישי יש למשתמש אפשרות לצפייה במציאות שמצא, באבדות שאבד ואפשרות לעריכת פרטים אישיים.

---

מציאות

---

אבדות

---

עריכת פרטים

---

1. סיכום ומסקנות

בעבודה על פיתוח האתר השקענו רבות בבניה נכונה ויעילה של המערכת, על המודלים שירכיבו אותו וצורת הכתיבה שתאפיין אותו.

* במהלך הפרויקט למדנו הרבה נושאים חדשים, ואנו מרגישות שצברנו ניסיון רב . תחומים שהכרנו בשטחיות או שלא הכרנו בכלל, נעשו ידידותיים בהרבה ממה שחשבנו. גילינו שמהנה ללמוד דברים חדשים ולנסות דרכי חשיבה מגוונות ומפתיעות ממה שנראה במבט ראשון.
* למדנו להכיר לעומק את סביבת הפיתוח של ,WEB API.NET SQL Serverו ,Angular6 - את יכולותיהם הרבות ושיטות חדשות ושונות בתכנות. ולפני הכל - הפנמנו שסבלנות והתמדה – אין כמותם בכל משימה הנראית עד בלתי אפשרית...
* רכשנו ניסיון בתכנון מערכת החל משלב ניתוח המערכת, תכנון נתונים ועיבוד יעיל, פיתוח המערכת ובדיקות מקיפות.
* צברנו ידע והתמקצעות בעיצוב אתרים והבנה מעמיקה בהתממשקות לתוכנות אחרות.

1. ביבילוגרפיה

* [www.w3school.com](http://www.w3school.com)
* [www.codeproject.com](http://www.codeproject.com)
* [www.stackoverflow.com](http://www.stackoverflow.com)
* [www.developers.google.com](http://www.developers.google.com/)
* [www.angular-maps.com](http://www.angular-maps.com)
* www.csharp.net-informations.com