Readme-BG-Usafe

האפליקציה מאפשרת לסטודנטים לבצע דיווחים בזמן אמת וצפייה בדיווחים של אחרים בדומה לWAZE ובכך לעזור לסטודנטים להימנע מאזורים בהם לא הרחובות חשוכים/יש רכב או אדם שיכור או נרקומן אשר גורם לדרך להיות פחות סימפטית. בנוסף בעזרת הדיווחים יהיה ניתן להפיק דוח לפי שם רחוב ולראות את מספר הדיווחים באותו רחוב וזה ייתן אינדיקציה לבחירת רחוב כאשר סטודנט חדש עובר לגור בב"ש ורוצה להבין באיזה אזור מומלץ לגור.

**יש לבצע את סדר הפעולות הבא לאחר פתיחת כל הקבצים ושמירה בVS :**

1. לשים את הסיסמה של הdb בתוך הקובץ של הdbconfig
2. להריץ בcommand node index.js
3. לגשת לlocalhost3000

להעתיק בכל פעם url אחד :

1. localhost:3000/CreateTable
2. localhost:3000/alterTable\_reports\_primeryKey
3. . localhost:3000/alterTable\_reports\_AUTO\_INCREMENT
4. localhost:3000/InsertReportsFromCSVfile
5. ניתן לחזור לlocalhost3000 – הדף הראשי של האפליקציה

**סדר פעולות שממחישות את הפעילות האתר:**

1. ללחוץ לservices
2. ללחוץ על send a report למלא ולשלוח אותו.
3. לחזור לservices ללחוץ לreal time map – בו יופיעו כל הדיווחים שנטענו מקובץ הנתונים + הדיווח שדיווחת משלב 2.
4. לחזור לservices ללחוץ על statistical reports כדי לראות את הדיווחים ברחוב יש לחפש את הכתובת הרלוונטית **וללחוץ עלייה בהשלמה אוטומטית**! ( אם לא ילחץ יתכן ולא יופיע הדיווח ) ניתן גם לחפש כל כתובת אחרת שנמצא בקובץ DB

**סרטון שמציג את כל הפיצ'רים באתר (באורך דקה)**

<https://www.loom.com/share/12c32e441d3e4793b6b5d3531d6e7f34>

**סעיף 1 מבנה הפרויקט:**

**חלק ב:**

1.Views:

2. static:

קובץ CSS ותיקיית Imeags

* send-report JS
* statistical JS
* real-time-map JS
* Jquery JS – ספרייה חיצונית שייבאתי אותה.
* App JS

**חלק ג:**

3. index.js – זה העמוד בו יצרתי את הserver

4.node:

* קובץ db - חיבור
* dbconfig – פרטי DB
* Data.csv

5. routes – ריכזתי את כל הroutes במקום אחד

6. reportsController – ( כמו CRUD )

סעיף 2 – טיפול בבקשת לקוח

בקשות הלקוח –

|  |  |  |
| --- | --- | --- |
| בקשה | קובץ JS | סוג מתודה |
| שליחת דיווח | Send report | post |
| הצגת כל הדיווחים על המפה | Real time map | get |
| הלוקח מחפש רחוב וצופה בכל הדיווחים שהיו ברחוב זה. | Statistical report | Post |

סעיף 3 – בסיס נתונים

שם הבסיס נתונים :mqsql

שם הטבלה : reports מרכזת את כל הדיווחים

קובץ נתונים : data.cvs יש בו כ10 דיווחים שונים.

שאילתות : נמצאות בקובץ – reportsController

פירוט השאילתות ותפקידם:

|  |  |
| --- | --- |
| שם | תפקיד |
| InsertReportFromClient | מכניסה דיווח חדש לטבלה reports מהלקוח |
| GetReportsToMap | בזמן לחיצה על המפה לוקחת את כל הדיווחים מהבסיס ובעזרת מימוש בJS כל דיווח מוצג כמרקר על המפה |
| DropTable | מחיקת הטבלה בעת הצורך |
| GetReportsBySearch | לכל דיווח לוקחים טווח של הקורדינטות כדי להציג את כל הדיווחים מאותו רחוב ולכן צריך מינימום ומקסים לlat ולlangעושים סינון לפי המדדים האלו    const query =      "SELECT \* FROM reports WHERE lat between " +      req.body.latMin +      " and " +      req.body.latMax +      " AND lng between " +      req.body.lngMin +      " and " +      req.body.lngMax; |

סעיף 4 + 5 –

**הערה:** לא מימשתי בטפסים ACTION וMETHOD אלא עשיתי קליטה בעזרת jQuery-post/get בקבצי JS הרלוונטיים לכל טופס. הסיבה שבחרתי לממש כך כי כבר השתמשתי בjQuery וכי זה עוזר לי במימוש פונקציונליות יחד עם פונקציות נוספות שלקחתי מספריות google API. אין הבדל מהותי בין השניים עדיין יש שימוש של method וpath רלוונטי.

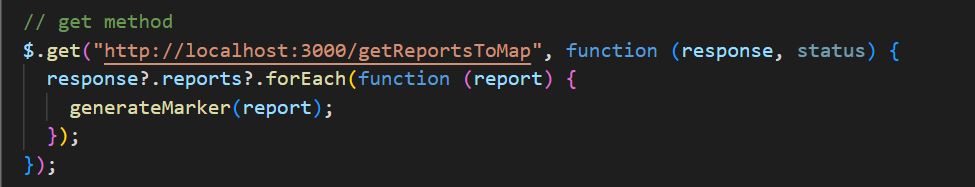
בעמוד הבא פירטתי לגבי זה:

* העברת הדיווח לDB :
  + האזנה לכפתור submit של הטופס למילוי דיווח ->

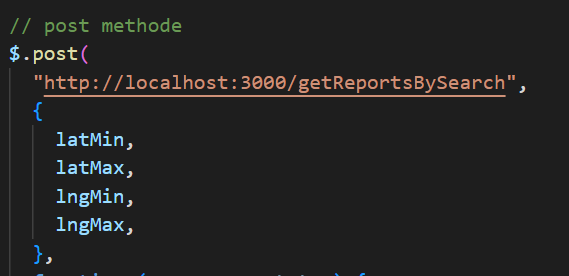
קליטה של הנתונים , המרה של הקורדינטות לטקסט שניתן לקרוא בעזרת geocoder. העברת הנתונים לבקאנד בעזרת המתודה post

 $.post("http://localhost:3000/sendReport",

שליפת הדיווח מהDB והפיכתו למרקר על המפה

* real-time-map JS
  + function initialize ->
    - קבלת כל הדיווחים מהבקאנד בעזרת המתודה GET + כל דיווח שנמצא בטבלה בבסיס נתונים להפוך למרקר על המפה

שליפת הדיווחים הרלוונטים מהDB והצגתם בטבלה למשתמש

* Statistical –
  + initAutocomplete – פונקציה שעושה השלמה אוטומטית של המיקום לפי הזנת המשתמש בinput החיפוש
  + האזנה לבחירת הAutocomplete ברגע שנבחרת כתובת מהשתמש קורא לפונקציה fillInAddress:
    - שליחה לבקאנד - המשמעות של latmin/mac lng min/max הם קורדינטות