שיעור 2

אג'נדה

- ✔ שימוש בNode ליצירת דפים דינמיים ע"י ejs.
- ✔ קצת מושגים: Statelessness, REST APIs, Auth, Jwt
- ✔ Middleware- שימוש נכון והיתרונות שמבנה זה מספק לנו
- ✔ מימוש API פשוט
- ✔ היכרות ובניית פרויקט התחלתי ב mongoDB

ejs

ejs זו דרך להחזיר נתונים דינמיים מnode לדפי html.

נציין בקוד (app.set('view engine' , 'ejs' , נוסיף לפרוייקט תיקייה view שבה יהיו כל הקבצים הדינמיים שזה בעצם קבצים עם סיומת של ejs -> מבנה רגיל של html בתוספת למשתנים שניתן להעביר.

Hands On!!!

נרים שרת express, ניצור תיקייה public המכילה קבצי html וcss וכן routes לכל קובץ.

נקנפג את השרת עבור ejs, ניצור תיקיית views עם קבצי ejs ונעביר נתונים לדפים בצורה דינמית.

REST APIs

REpresentational
State
Transfer

Application
Programming
Interface

REST APIs מאפשר לתוכנה לדבר עם תוכנה אחרת.
בקשה נשלחת לשרת פנימי או לשרת אחר על מנת לקבל נתונים ממקור חיצוני.

שימוש נכון בAPI מאפשר לאפליקציה להיות פרודאקטיבית משום שיהיה ניתן להשתמש במספר אפליקציות שונות עבור אותו מודול של API (scalability) הנתונים מתקבלים כdata בצורת json-> אובייקט המורכב מ name:value.

REST API's

פעולות ה http האפשריות עבור קריאות REST הן (מיוצגות ע"י CRUD):

Post-> יצירה או שליחת מידע
Get-> קבלת מידע יחיד או מערך
Put-> יצירה או החלפה של מידע (שליחת כל המידע בשינוי השדות הרצויים)
Patch -> עדכון או שינוי של שדה מסוים בתוך המידע
Delete -> מחיקת יחידת מידע.

לדוג' : { user_name:"shalom", user_email:"shalom@gmail.com" }
באיזו פעולה נשתמש כדי לעדכן את השדה user_name ?
באיזו פעולה נשתמש כדי להכניס משתמש חדש?
באיזו פעולה נשתמש כדי לעדכן את כל הפרטים?

REST API's

ע״י שימוש פעולות הנ״ל (בתוספת כמובן לעוד מספר הגדרות ע״מ לבצע את הפעולה עצמה), נוכל ליצור בשרת שלנו endpoint עבור כל פעולה, להחזיר לfront את הערכים המבוקשים או להשתמש בנתונים שקיבלנו למטרות הרצויות.

Hands On!!!
Weather API

Statelessness

המושג Statelessness מאפיין תכנה שפועלת רק על סמך הקלט הנוכחי שהיא מקבלת ולא מסתמכת על data מבקשות קודמות.

כל בקשה משמשת כ"טרנזקציה"- תנועה, קריאה נפרדת לגמרי שלא בנויה על הבקשה הקודמת.

ע"מ ליצור את המבנה הזה, ה client יצטרך בכל בקשה לשלוח את כל הנתונים מחדש, וה server לא אמור לשמור אצלו כלום (גם לא הרשאות ואימותים).

JWT

Json **W**eb **T**oken

כאשר אנו מעוניינים לאבטח (ולהבטיח) ניהול הרשאות נכון ווליד‌ציה תקינה,
לכל משתמש שנכנס למערכת על ידי שם משתמש וסיסמה תקינים, ניצור jwt שתפקידו לוודא את זהות המשתמש במערכת בכל פעולה (במקום לבקש ממנו להזדהות בכל פעם).

JWT מורכב מצירוף של פרטים בסיסיים על המשתמש (בצורת אובייקט) ומחרוזת סודית אשר רק לשרת יש אפשרות לגשת אליה (נשמרת בקובץ env.).
בדרך כלל, הtoken כולל claims נוספים כמו זמן תפוגה וכן התרעננות אוטומטית.
בכל בקשת api שמתבצעת, הjwt נשלח לשרת, השרת מחלץ את נתוני המשתמש ובודק אם המחרוזת הסודית לא נפתחה (משווה את המחרוזת בtoken למחרוזת השמורה אצלו) אזי המשתמש מאומת והבקשה מתבצעת בהתאם להרשאות.

middleware

פונקצייה או תכנית שהולכת לרוץ בין הזמן שהשרת מקבל את ה req
ועד לזמן שבו הוא מחזיר res ל client.

ז"א, כאשר לדוגמא יש route: /user ,
פונקציית ה middleware המתאימה תתבצע בין הזמן שבו הroute נשלח, לבין הזמן שבו חזר
העמוד המדובר.

[לדוג': res.send זה פונקציית middleware משום שהיא מתבצעת לפני שהתשובה נשלחת למסך].

באופן כללי כשנדבר על moddleware,
נדבר על הפעולות שקורות בין שליחת הבקשה לביצוע שלה.

middleware

פונקציות אילו מאפשרות לנו לגשת לבקשה ולתשובה וכך לשמור על בקרה בין העמודים, נוכל לשנות את req בהתאם לתוצאות הפונצקייה.

לדוגמא, אם נרצה לבדוק האם משתנה הוא אדמין באתר:

```
if(req.query.admin=='true')
    next()
else
    res.send('no auth')
```

וב endpoint המתאים נכתוב :

```
/users?admin=true
```

אם התנאי הפוך- לא לשכח לעשות return()

MongoDB