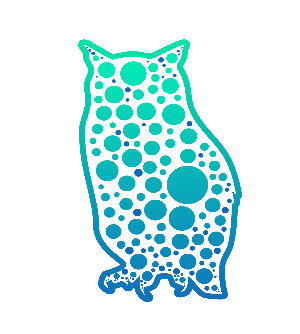
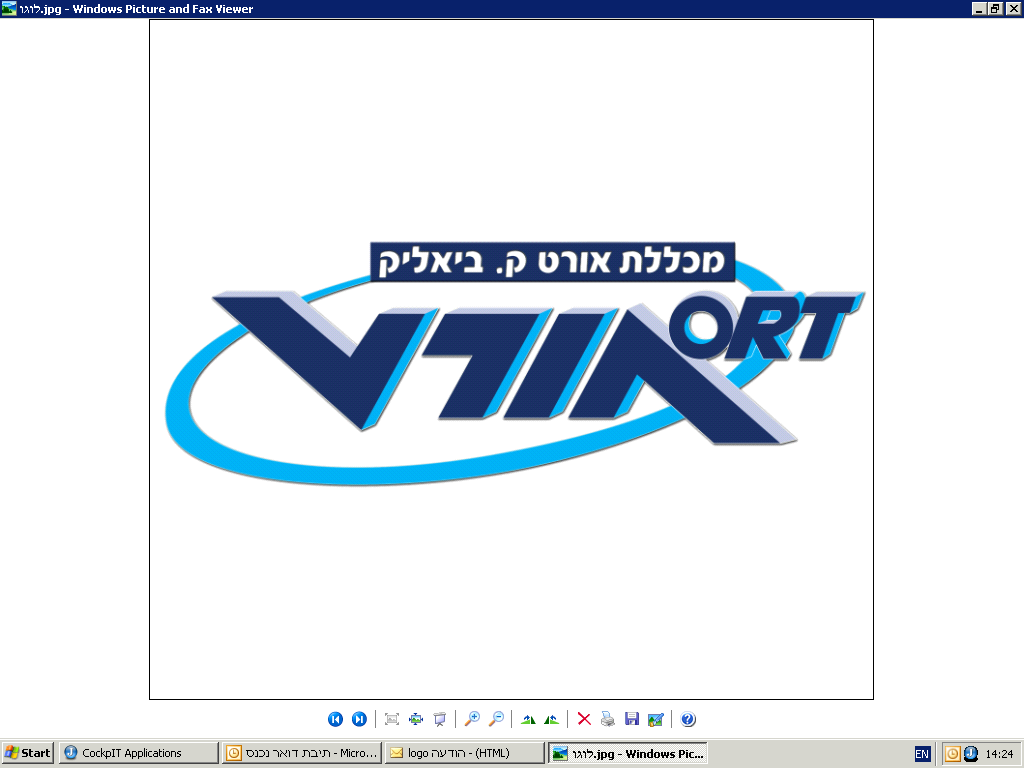
****

****

**פרויקט גמר**

-פרוייקטנט ממר''ם-

ה נ ד ס א י

ה נ ד ס ת - ת ו כ נ ה

בהתמחות: מ ח ש ב י ם

**נושא:** "פתיחת ציר" וניטור מערכת NetApp E-Series

**מגיש:** ארד גונן

**תעודת זהות:** 206394579

שנה"ל תשע"ח 2018

תוכן

[*מטרת המערכת:* 2](#_Toc513025814)

[ציוד חומרה ותוכנה הנדרשים לביצוע הפרויקט 3](#_Toc513025815)

[תהליכים עיקריים: 4](#_Toc513025816)

[כלים תוכנתיים ששולבו בפרוייקט: 5](#_Toc513025817)

[שלבים בפיתוח הפרוייקט: 6](#_Toc513025818)

[הפעלת המערכת: 7](#_Toc513025819)

[*מבנה* שרת הREST*:* 10](#_Toc513025820)

# מטרת המערכת:

מערכת ניטור ופתיחת ציר עבור מכונת אחסון מסוג E-Series (NETAPP). מערכת זאת באה לפתור בעיה של חוסר יכולת ביצוע פתיחת ציר במכונות אלו, ובנוסף ומאפשרת ניטור קל בעזרת ממשק גרפי נעים לשימוש.

פתיחת ציר הוא תהליך צבאי בו מציגים תמונת מצב של כל התקלות במכונות הרלוונטיות בפתיחה. במכונות כמו ה E-Series  תהליך זה הוא ידני לגמרי ולוקח מספר שעות. בעזרת מערכת זו אני שואף לקצר את זמן פתיחת הציר לדקות בודדות.

מערכת זו אוספת נתונים מהלוגים של המכונה, מנתחת אותם ומציגה בצורה גרפית תקלות חומרה וניהול של המכונה.

מטרת המערכת, היא "פתיחת ציר" – מושג המתאר תהליך של חיבור בין שני שרתים והעברת מידע ביניהם. הצורך במערכת כזו נובע מכך שהמערכת E-Series של חברת NetApp לא נותנת אפשרות מובנת להצגת הודעות ממערכת האחסון, אלא רק קבצי לוג שאותם צריך לנתח. המערכת פותחה במטרה לתת דרך קלה ומהירה לבדוק את תקינותה של המערכת, המשמשת את היחידה בצורה נרחבת. המערכת נבנתה בסביבת Web (אזכיר שלא נלמד פיתוח בסביבה זו במכללה, אלא אני למדתי לבד כיצד לפתח בה). השרת המריץ את המערכת מבוסס Node.js שזוהי מעטפת javaScript המשמשת לתכנות צד שרת. צד הלקוח, מבוסס Angular 5, (הדרך המודרנית ביותר כיום לפיתוח Web), העיצוב מבוסס Bootstrap, Angular-Material,ng-bootstrap. מסד הנתונים המשמש אותי הוא MongoDB , מכיוון שיזרום מידע רב דרך המערכת ולכן צריך מסד המיועד לשימוש Big Data.

# ציוד חומרה ותוכנה הנדרשים לביצוע הפרויקט

מבני נתונים- רשימות וטבלאות.

קבצים- קבצי קונפיגורציה (יכילו מידע על המערכות שצריך להתחבר אליהם).

בסיס נתונים-מבוסס MongoDB , ישמור את תוצאות פתיחת הציר האחרונה על מנת לזרז גישת משתמשים.

אלגוריתמים-ניתוח קבצי JSON  שנקבל אותם מהמערכת - בעזרת REST api  .

תכנות רשת- חיבור מרחוק בעזרת HTTP  וRESTAPI לשרת הניהול של המכונה. חיבור לדומיין לצורך זיהוי משתמש בכניסתו לאתר (SSO) – מעבר מידע בין שרת ללקוח ולהיפך בעזרת SocketIO.

עמדת פיתוח - Windows  עם NodeJs וMongoDB.

עמדת משתמש - כל עמדה עם Chrome

# תהליכים עיקריים:

-התחברות לשרת RESTapi לשם קבלת מידע מהמערכות.

- התחברות למסד נתונים MongoDB

-לכל לקוח שמתחבר, המערכת תנסה לקבל את שם המשתמש המחובר לwindows, במידה ולא תצליח הוא ייקרא Guest .

- המערכת תקבל רשימה של כל התקלות במערכת האחסון ותייצר טבלאות המייצגות כל מערכת ומערכת ברשת.

* התחברות למערכת – התחברות בצורת SSO
* צפייה במצבן הכללי של כל המכונות – חלון ה Dashboard
* צפייה בתקלות חומרה בקטגוריה מסויימת – (Disks , Mgmt , Hardware , FC , Pools&Vols)
* ייצוא של פרטי התקלה והרכיב התקול והמערכת אליה הוא שייך בCSV וHTML
* צפייה בתקלות השוטפות במכונות
* עדכון הנתונים – בצורה ידנית (פתיחת ציר עלולה לקחת מספר דקות – כדי למנוע מצב של כניסה למערכת ולא מוצג שום מידע

# כלים תוכנתיים ששולבו בפרוייקט:

כל הפרוייקט מבוסס בבסיסו על JavaScript + HTML. צד השרת מבוסס מעטפת JavaScript בשם Node.js - זוהי מעטפת מאוד יעילה לתכנות צד שרת בזכות זה שהיא אסינכרונית כלומר אין מצב שתהיה שורה שלוקח לה הרבה זמן להתבצע כי במקום להמתין הקומפיילר יעבור לשורה הבאה לביצוע.

צד הלקוח מבוסס Angular 5 - בנוסף השתמשתי בכמה מודולים קיימים המוסיפים פונקציונליות משופרת.

Charts.js - מודול המציג דיאגרמות, איתו כלול pieceLabel.js המאפשר לכתוב על הדיאגרמות.

Angular Material – מודול לעיצוב של גוגל לAngular.

Ng-bootstrap - מודול עיצוב בוטסטראפ לממשק Angular של צד לקוח

Ngx-json-viewer – מודול להצגה יפה ונוחה של קבצי JSON

Node-sspi – מודול להטמעת sso - כניסה חד פעמית למערכת, המודול מזהה את המשתמש המחובר לווינדוס ומעביר למערכת את שם המשתמש

Express - מודול לשרת

# שלבים בפיתוח הפרוייקט:

*עד סוף נובמבר 2017 – סיום תהליך הוצאת תקלות מכל קטגוריה וייצוא המידע.*

*עד סוף 2017 – שאיבת הנתונים מהמכונות ב RestAPI  ותחילת עבודה על MongoDB.*

*עד סוף ינואר 2018 – הרצת \פתיחת ציר על פי לוז ואחסון התוצאות בMongoDB.*

*עד סוף פברואר 2018 – עבודה על מנגנון SSO.*

*מרץ 2018 – ליטושים \ תיקוני באגים.*

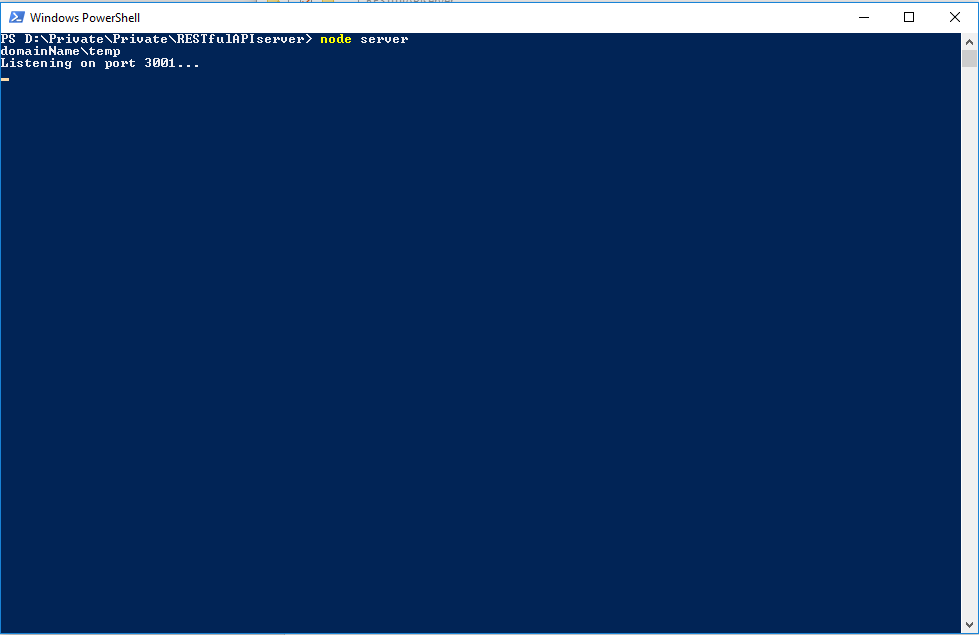
*עד סוף מרץ 2018 –  מערכת עובדת 90-100% מהתכנון המקורי.*

# הפעלת המערכת:

-קודם כל בגלל שזה פרוייקט צבאי שנועד לפעול בממר''ם, נצטרך להפעיל שרת דמה הידמה את פעולת מכונת האחסון ושיתמוך בrestapi.

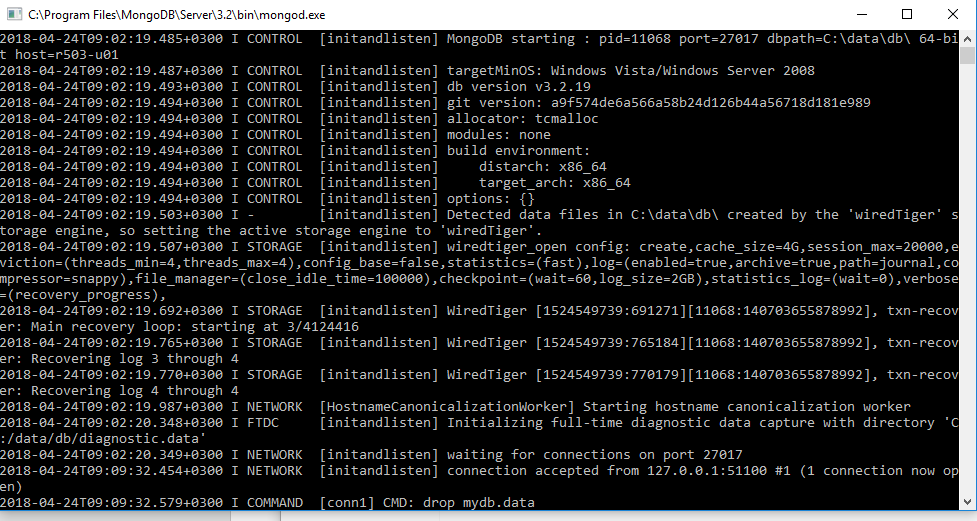
כך נפעיל את השרת:

נפתח חלון powershell בתיקייה בה נמצאים קבצי השרת -> נכתוב את הפקודה node server



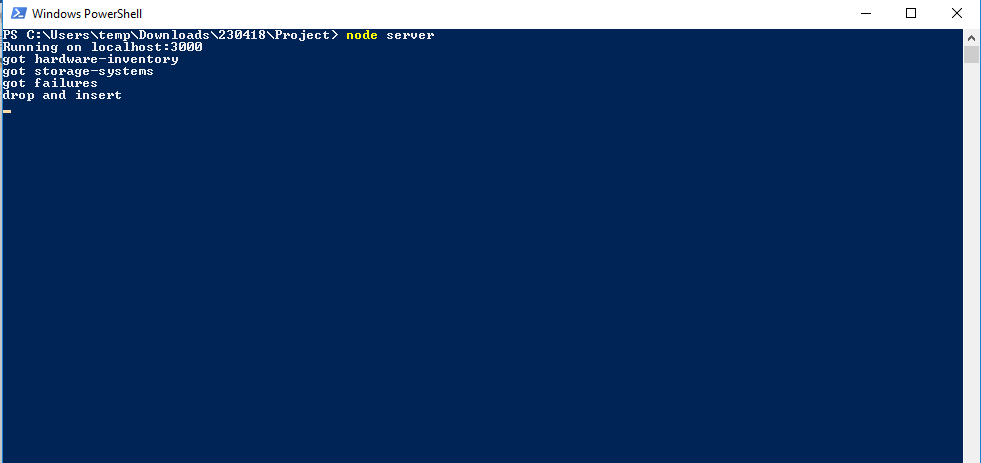
לאחר מכן, נצטרך להפעיל את מסד הנתונים (במקרה שהוא לא מותקן כservice)

נכנס לכונן C: -> לתקייה program files - > לתיקייה MongoDB -> לתקייה בשם server - > לתקייה בשם 3.2 -> לתקייה bin - > נריץ קובץ בשם mongod



ועכשיו ניתן להריץ את השרת הראשי

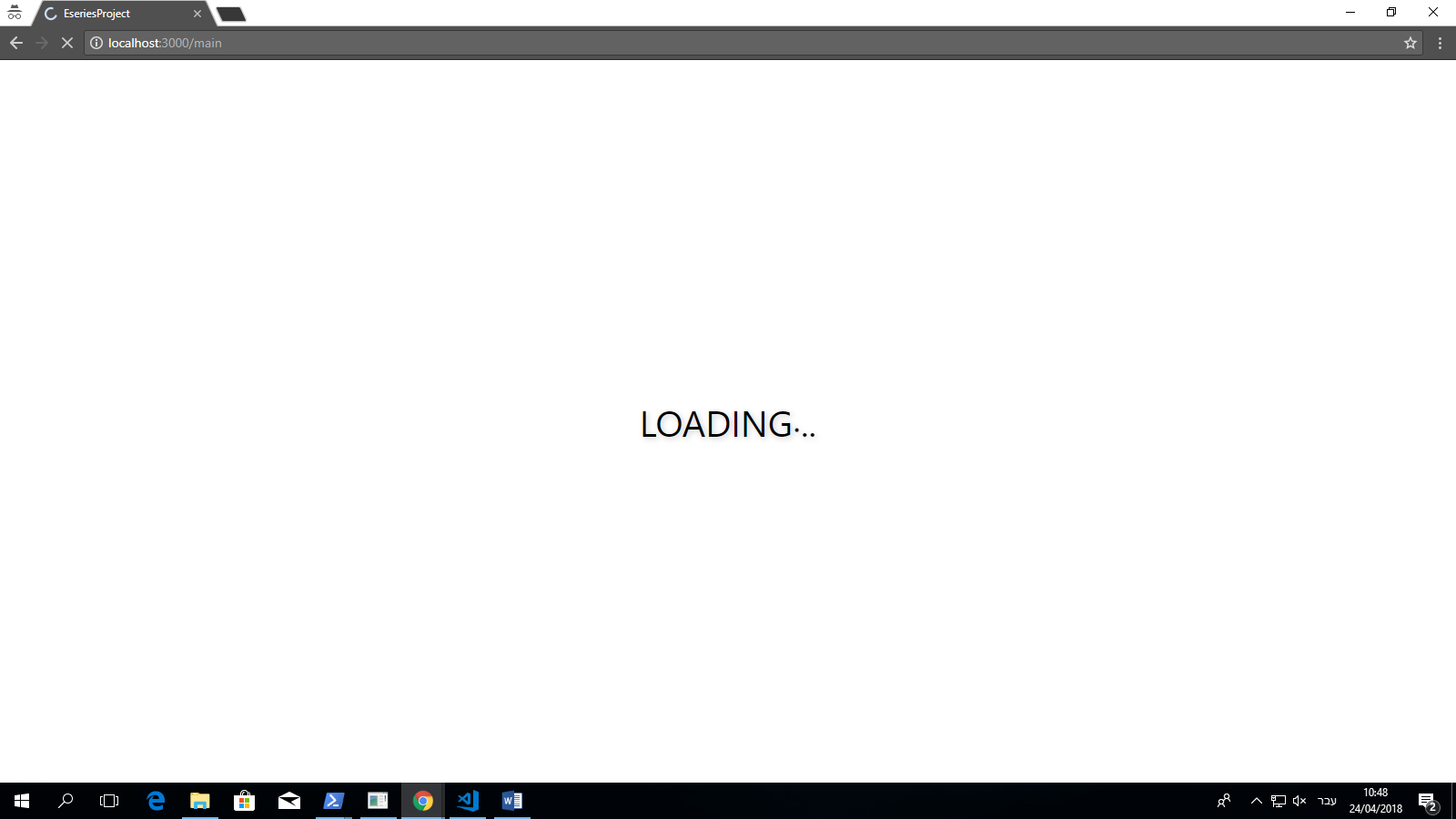
ניכנס לתיקייה של הפרויקט, נפתח חלון powershell ונכתוב את הפקודה node server (במידה ונעשה שינוי במערכת נריץ לפני את פקודה ng build).



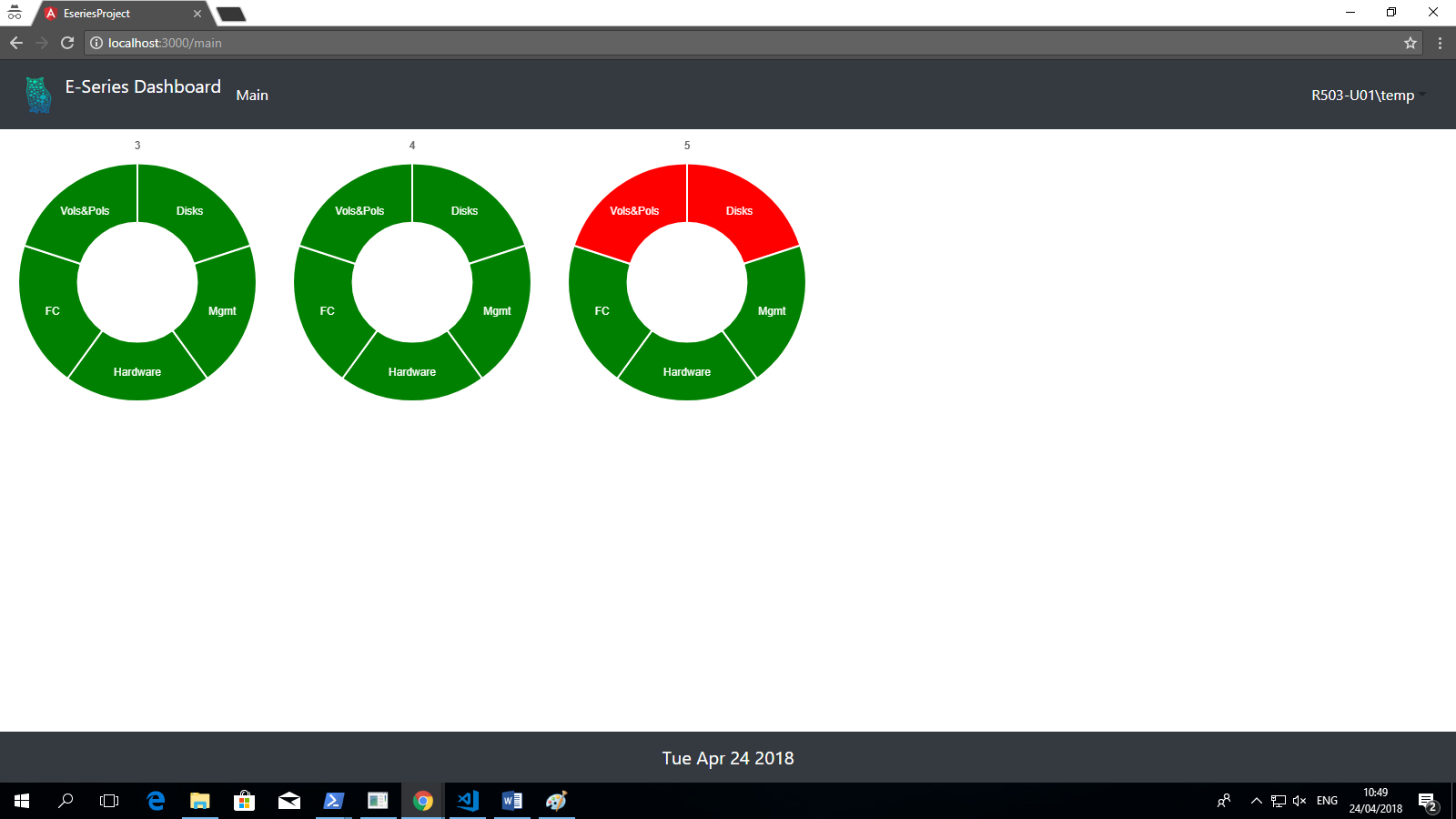
עכשיו ניכנס לדפדפן (כרום) ונכתוב: אם אנחנו במחשב שבו רץ השרת, localhost:3000

במידה ואנחנו על מחשב אחר ברשת, נכתוב את כתובת הip של השרת לדוגמא 192.168.1.123:3000

בהתחלה נראה מסך טעינה:



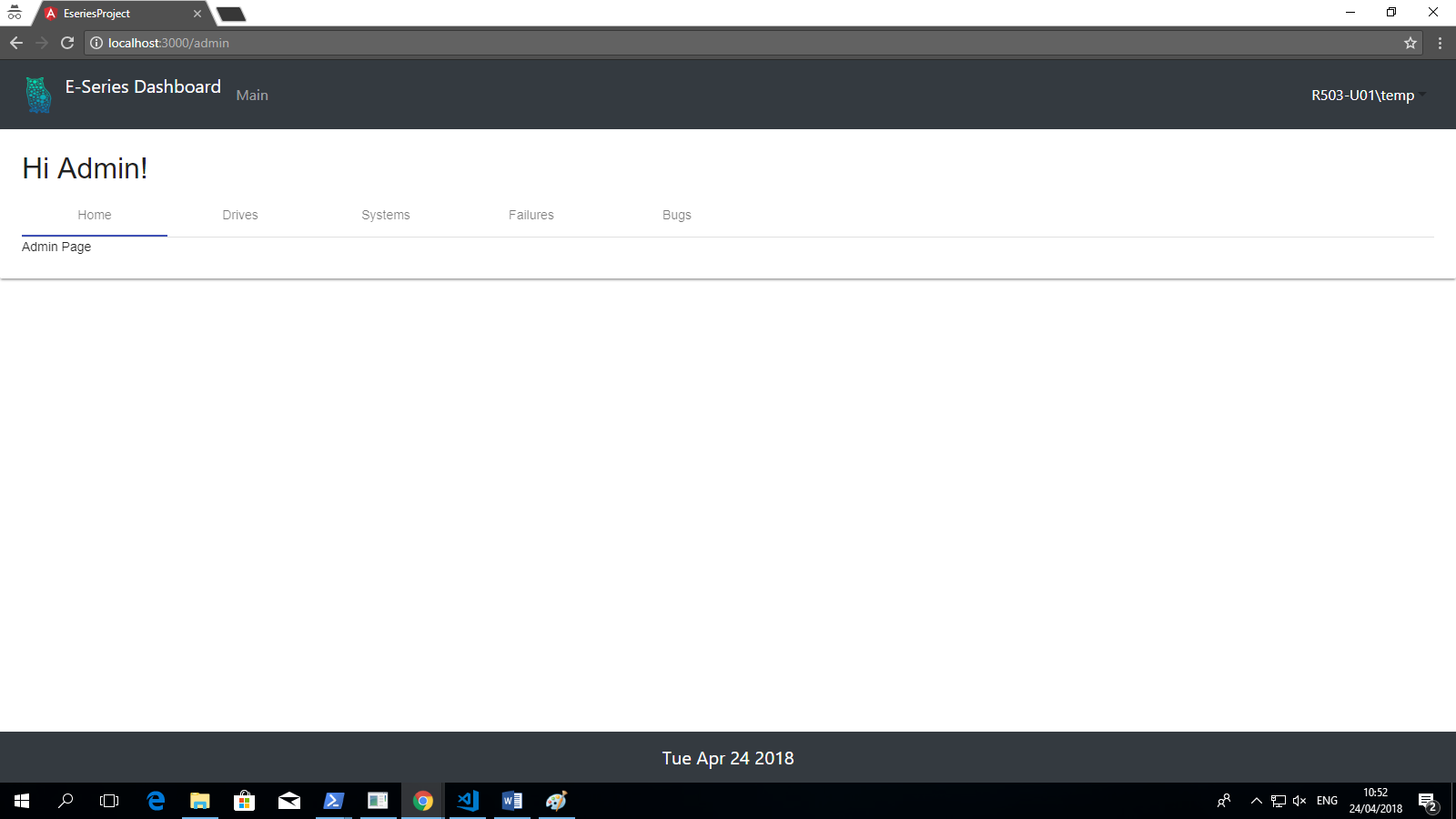
ואז יעלה המסך הראשי:



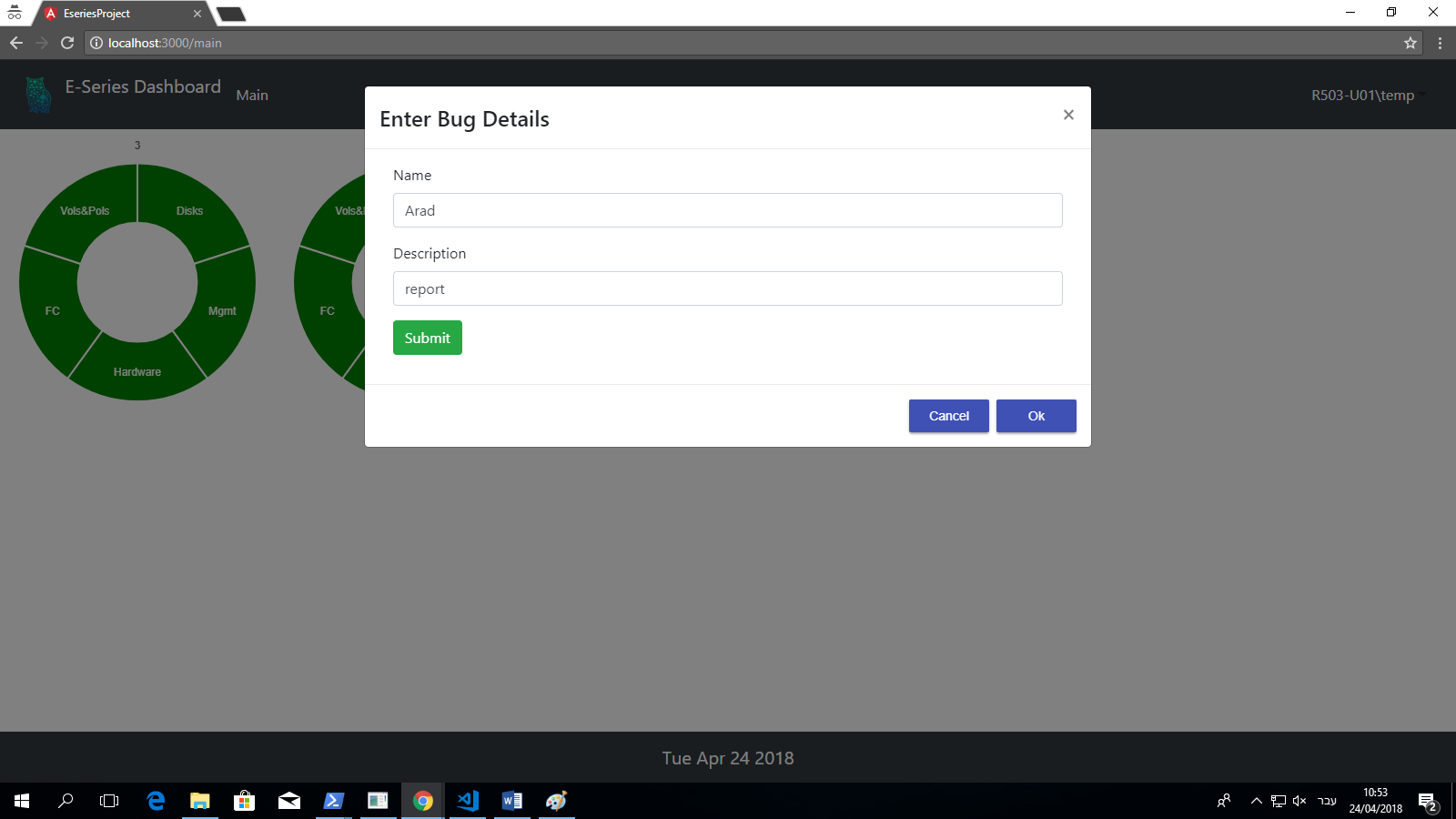
אם נלחץ על שם המשתמש, ייפתח תפריט משתמש – במידה והמשתמש המחובר מוגדר במערכת כמנהל, יופיע לו האפשרות להגיע לדף הניהול של המערכת



דף מנהל המערכת נראה כך:



במידה ונרצה לשלוח הודעה על באג, נלחץ על bug report ונקבל מסך כזה:



# מבנה שרת הREST:

* + בתיקייה Arad נמצאים קבצי log זמניים שקיבלתי מממר''ם
  + בתיקייה node\_modules נמצאים קבצי הגדרה של node.js
  + בתיקייה routes קיים קובץ routes.js

שהוא הקובץ שבו נמצא הקוד העונה לhttp request

* + Package.json קובץ הגדרה של node.js
  + Server.js קובץ השרת

Server.js:

**var** express **=** require**(**'express'**);**//express module

**var** app **=** express**();**//assign the app variable to express

**var** fs **=** require**(**'fs'**);**//import the filesystem module

**var** bodyParser **=** require**(**"body-parser"**);**//import body-parser

**var** path **=** require**(**'path'**);**//import path module

**var** userName **=** process**.**env**[**'USERPROFILE'**].**split**(**path**.**sep**)[**2**];**

**var** loginId **=** path**.**join**(**"domainName"**,**userName**);**

console**.**log**(**loginId**);**

app**.**use**(**express**.static(**'Arad'**));**//set main folder for assets

app**.**use**(**bodyParser**.**json**());**//assign to use the module

app**.**use**(**bodyParser**.**urlencoded**({** extended**:** **true** **}));**

**var** routes **=** require**(**"./routes/routes.js"**)(**app**);**//initialize the routes

**var** server **=** app**.**listen**(**3001**,** **function** **()** **{**

console**.**log**(**"Listening on port %s..."**,** server**.**address**().**port**);**

**});**//start the server on specific port

Routes.js:

**var** appRouter **=** **function(**app**)** **{**

app**.**get**(**"/storage-systems"**,** **function(**req**,** res**)** **{**//when http request like this arrive, send the file

res**.**sendFile**(**'storage-systems.json'**,** **{**root**:**'Arad'**});**

**});**

app**.**get**(**"/storage-systems/5/hardware-inventory"**,** **function(**req**,** res**)** **{**//when http request like this arrive, send the file

res**.**sendFile**(**'hardware-inventory.json'**,** **{**root**:**'Arad'**});**

**});**

app**.**get**(**"/storage-systems/5/failures"**,** **function(**req**,** res**)** **{**//when http request like this arrive, send the file

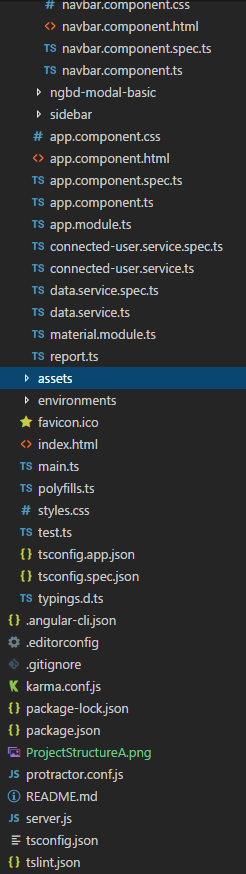
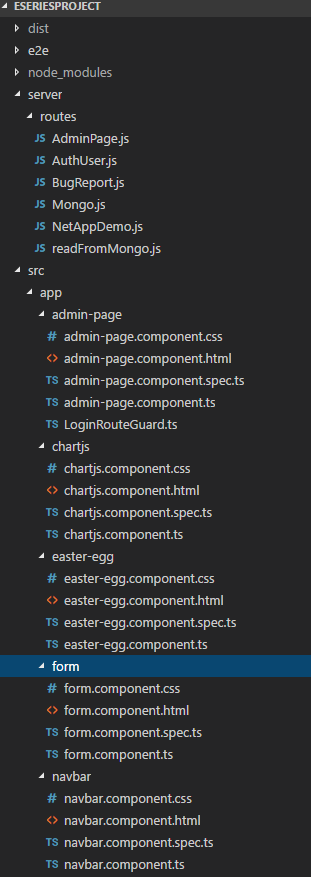
res**.**sendFile**(**'failures.json'**,** **{**root**:**'Arad'**});**

**});**

**};**

module**.**exports **=** appRouter**;**//export the router to use in other file, server file

# מבנה השרת הראשי:



תיקיית Routes:

מכילה קבצי הגדרה לשרת לטיפול בבקשות HTTP.

AdminPage.js:

AuthUser.js: יחזיר את המידע שמתקבל מהמודול node-sspi ויחזיר תגובת http עם פרטי המשתמש המחובר

BugReport.js : שומר למסד הנתונים פרטי תקלות ממשתמשים וגם מושך מהמסד כדי להציג את כל התקלות

Mongo.js: מכיל פונקציות תקשורת עם מסד הנתונים

NetAppDemo.js : מבוסס פרוטוקול RESTapi לתקשורת עם שרת הדמה, בעתיד יתקשר עם המערכת עצמה.

readFromMongo.js: מכיל פונקציות לקריאה ממסד הנתונים

Server.js: קובץ ההגדרה של השרת

קוד:

AdminPage.js:

**const** express **=** require**(**'express'**);**//express module

**const** router **=** express**.**Router**();**//express-router module

**const** path **=** require**(**'path'**);**

router**.**get**(**'/Info'**,** **(**req**,** res**)** **=>** **{**

res**.**sendFile**(**path**.**join**(**\_\_dirname**,** '../views/adminPage.html'**));**

**});**

module**.**exports **=** router**;**

AuthUser.js:

**const** express **=** require**(**'express'**);**//express module

**const** router **=** express**.**Router**();**//express-router module

**var** path **=** require**(**'path'**);**

**var** userName **=** process**.**env**[**'USERPROFILE'**].**split**(**path**.**sep**)[**2**];**

**var** loginId **=** path**.**join**(**"domainName"**,**userName**);**

**const** MongoClient **=** require**(**'mongodb'**).**MongoClient**;**//mongo module and client

**const** ObjectID **=** require**(**'mongodb'**).**ObjectID**;**//object ID-useful for data sorting

**let** response**={**

**status:** 200**,**

data**:** **[],**

message**:** **null**

**};**

**const** connection **=** **(**closure**)** **=>** **{**

**return** MongoClient**.**connect**(**'mongodb://localhost:27017/mydb'**,** **(**err**,** client**)** **=>** **{**

**if** **(**err**)** **return** console**.**log**(**err**);**

**let** db **=** client**.**db**(**'mydb'**);**

closure**(**db**);**

**});**

**};**

router**.**get**(**'/UserInfo'**,** **(**req**,** res**,**next**)** **=>** **{**

**var** nodeSSPI **=** require**(**'node-sspi'**);**

**var** nodeSSPIObj **=** **new** nodeSSPI**({**

retrieveGroups**:** **true**

**});**

nodeSSPIObj**.**authenticate**(**req**,** res**,** **function(**err**){**

res**.**finished **||** next**()**

**});**

**var** userName**=**req**.**connection**.**user**;**

**if(**userName**==**undefined**)**

**return;**

**var** userSid**=**req**.**connection**.**userSid**;**

**if(**userSid**==**undefined**)**

**return;**

**var** user **=** **[];**

**var** groups**=[];**

**if** **(**req**.**connection**.**userGroups**)** **{**

**for** **(var** i **in** req**.**connection**.**userGroups**)** **{**

**if(**req**.**connection**.**userGroups**[**i**]==**undefined**)**

**return;**

groups**.**push**(**req**.**connection**.**userGroups**[**i**]);**

**}**

**}**

user**.**push**({**UserName**:**userName**,**UserSID**:**userSid**,**userGroups**:**groups**});**

response**.**data **=** user**;**

**if(**res**.**statusCode**==**200**)**

{

console.log(res.statusCode);

res.json(response);

}

console.log(this.username);

});

router.get('/UserInfo/Admins', (req, res) => {

connection((db) => {

db.collection('users')

.find()

.toArray()

.then((users) => {

res.json(users);

})

.catch((err) => {

sendError(err, res);

});

});

});

module.exports = router;

BugReport.js :

**const** express **=** require**(**'express'**);**//express module

**const** router **=** express**.**Router**();**//express-router module

**const** MongoClient **=** require**(**'mongodb'**).**MongoClient**;**//mongo module and client

**const** ObjectID **=** require**(**'mongodb'**).**ObjectID**;**//object ID-useful for data sorting

//connect block

**const** connection **=** **(**closure**)** **=>** **{**

**return** MongoClient**.**connect**(**'mongodb://localhost:27017/mydb'**,** **(**err**,** client**)** **=>** **{**

**if** **(**err**)** **return** console**.**log**(**err**);**

**let** db **=** client**.**db**(**'mydb'**);**

closure**(**db**);**

**});**

**};**

router**.**post**(**'/SaveReport'**,** **(**req**,** res**)** **=>** **{**

**var** newReport **=** req**.**body**;**

console**.**log**(**newReport**);**

connection**((**db**)=>{**

db**.**collection**(**'BugReports'**).**insert**(**newReport**,function(**err**,**res**){**

**if(**err**)** **throw** err**;**

console**.**log**(**"New Report In DB"**);**

**});**

**})**

**});**

**let** response**={**

**status:** 200**,**

data**:** **[],**

message**:** **null**

**};**

router**.**get**(**'/AllBugs'**,** **(**req**,** res**)** **=>** **{**

connection**((**db**)** **=>** **{**

db**.**collection**(**'BugReports'**)**

**.**find**()**

**.**toArray**()**

**.**then**((**users**)** **=>** **{**

response**.**data **=** users**;**

res**.**json**(**response**);**

**})**

**.catch((**err**)** **=>** **{**

sendError**(**err**,** res**);**

**});**

**});**

**});**

module**.**exports **=** router**;**

Mongo.js:

**const** express **=** require**(**'express'**);**//express module

**const** router **=** express**.**Router**();**//express-router module

**const** MongoClient **=** require**(**'mongodb'**).**MongoClient**;**//mongo module and client

**var** dataToClient**;**

**var** url **=** 'mongodb://localhost:27017/mydb'**;**

**function** writeToDB **(**DataToDB**,**collection\_name**)**

**{**

**if(**DataToDB**!=**undefined**)**

**{**

MongoClient**.**connect**(**url**,** **function** **(**err**,** db**)** **{**

**if** **(**err**)** **throw** err**;**

db**.**collection**(**collection\_name**).**count**(function(**err**,**count**){**

**if(**count**!=**0**)**

**{**

db**.**collection**(**collection\_name**).**drop**(function(**err**,**reply**){**

**if(**err**)** **throw** err**;**

**});**

db**.**collection**(**collection\_name**).**insert**(**DataToDB**,function(**err**,**res**){**

**if(**err**)** **throw** err**;**

**});**

console**.**log**(**"drop and insert"**);**

**}**

**else**

**{**

db**.**collection**(**collection\_name**).**insert**(**DataToDB**,function(**err**,**res**){**

**if(**err**)** **throw** err**;**

**});**

console**.**log**(**"insert"**);**

**}**

**});**

**});**

**}**

**else**

**{**

console**.**log**(**"ERROR - GOT UNDEFINED"**);**

**}**

**}**

module**.**exports **=** **{**

toDB**:** **function(**DataToDB**)**

**{**

writeToDB**(**DataToDB**,**"data"**);**

}

};

NetAppDemo.js :

**const** express **=** require**(**'express'**);**//express module

**const** request **=** require**(**'request'**);**

**var** data**={};**

**const** mongo **=** require**(**'./Mongo'**);**

**function** get\_hardware\_inventory**(**store\_hardware\_inventory**)**

**{**

request**(**'http://localhost:3001/storage-systems/5/hardware-inventory'**,** **{**json**:** **true},** **(**err**,** res**,** body**)** **=>** **{**

**if** **(**err**)** **{**

**return** console**.**log**(**err**+**"Im HereA"**);**

**}**

console**.**log**(**"got hardware-inventory"**);**

**return** store\_hardware\_inventory**(**body**);**

**});**

**}**

**function** get\_storage\_systems**(**store\_storage\_systems**)**

**{**

request**(**'http://localhost:3001/storage-systems'**,** **{**json**:** **true},** **(**err**,** res**,** body**)** **=>** **{**

**if** **(**err**)** **{**

**return** console**.**log**(**err**+**"Im HereB"**);**

**}**

console**.**log**(**"got storage-systems"**);**

**return** store\_storage\_systems**(**body**);**

**});**

**}**

**function** get\_failures**(**store\_failures**)**

**{**

request**(**'http://localhost:3001/storage-systems/5/failures'**,** **{**json**:** **true},** **(**err**,** res**,** body**)** **=>** **{**

**if** **(**err**)** **{**

**return** console**.**log**(**err**+**"Im HereC"**);**

**}**

console**.**log**(**"got failures"**);**

store\_failures**(**body**);**

**});**

**}**

**function** store\_hardware\_inventory**(**value**)**//callback function

**{**

data**[**0**]=**value**;**

get\_storage\_systems**(**store\_storage\_systems**);**

**}**

**function** store\_storage\_systems**(**value**)**//callback function

**{**

data**[**1**]=**value**;**

get\_failures**(**store\_failures**);**

**}**

**function** store\_failures**(**value**)**//callback function

**{**

data**[**2**]=**value**;**

print**();**

}

function print()

{

if(!data)

console.log("DATA EMPTY");

}

module.exports={

storeData: function()

{

var myPromise = new Promise(function(resolve,rejected){

get\_hardware\_inventory(store\_hardware\_inventory);

setTimeout(()=> resolve(data),2000);

});

myPromise.then((res)=>{

res=data;

mongo.toDB(res);

});

}

}

readFromMongo.js:

**const** express **=** require**(**'express'**);**//express module

**const** router **=** express**.**Router**();**//express-router module

**const** MongoClient **=** require**(**'mongodb'**).**MongoClient**;**//mongo module and client

**const** ObjectID **=** require**(**'mongodb'**).**ObjectID**;**//object ID-useful for data sorting

//connect block

**const** connection **=** **(**closure**)** **=>** **{**

**return** MongoClient**.**connect**(**'mongodb://localhost:27017/mydb'**,** **(**err**,** client**)** **=>** **{**

**if** **(**err**)** **return** console**.**log**(**err**);**

**let** db **=** client**.**db**(**'mydb'**);**

closure**(**db**);**

**});**

**};**

//error handling

**const** sendError **=** **(**err**,** res**)=>{**

response**.status** **=** 501**;**//status code

response**.**message **=** **typeof** err **==**'object' **?** err**.**message **:** err**;** //the error massage that include the error information

res**.status(**501**).**json**(**response**);**//return the status code and message

**};**

//response handling - the structure

**let** response**={**

**status:** 200**,**

data**:** **[],**

message**:** **null**

**};**

//get users

router**.**get**(**'/data'**,** **(**req**,** res**)** **=>** **{**

connection**((**db**)** **=>** **{**

db**.**collection**(**'data'**)**

**.**find**()**

**.**toArray**()**

**.**then**((**users**)** **=>** **{**

response**.**data **=** users**;**

res**.**json**(**response**);**

**})**

**.catch((**err**)** **=>** **{**

sendError**(**err**,** res**);**

**});**

**});**

**});**

module**.**exports **=** router**;**

Server.js:

**const** express **=** require**(**'express'**);**

**const** bodyParser **=** require**(**'body-parser'**);**

**const** path **=** require**(**'path'**);**

**const** http **=** require**(**'http'**);**

**const** app **=** express**();**

**const** NetAppData **=** require**(**'./server/routes/NetAppDemo'**);**

// API file for interacting with MongoDB

**const** MongoDBapi **=** require**(**'./server/routes/Mongo'**);**

**const** readFromMongo **=** require**(**'./server/routes/readFromMongo'**);**

**const** AuthUser **=** require**(**'./server/routes/AuthUser'**);**

**const** AdminPage **=** require**(**'./server/routes/AdminPage'**);**

**const** BugReport **=** require**(**'./server/routes/BugReport'**);**

**var** StillNeedUserInfo **=** **true;**

app**.**use**(function(**req**,** res**,** next**)** **{**

res**.**header**(**"Access-Control-Allow-Origin"**,** "\*"**);**

res**.**header**(**"Access-Control-Allow-Headers"**,** "Origin, X-Requested-With, Content-Type, Accept"**);**

next**();**

**});**

// Parsers

app**.**use**(**bodyParser**.**json**());**

app**.**use**(**bodyParser**.**urlencoded**({** extended**:** **false}));**

// Angular DIST output folder

app**.**use**(**express**.static(**path**.**join**(**\_\_dirname**,** 'dist'**)));**

app**.**use**(**'/readFromMongo'**,** readFromMongo**);**

app**.**use**(**'/AuthUser'**,** AuthUser**);**

app**.**use**(**'/AdminPage'**,**AdminPage**);**

app**.**use**(**'/BugReport'**,**BugReport**);**

// Send all other requests to the Angular app

app**.**get**(**'/snake'**,(**req**,**res**)=>{**

res**.**sendFile**(**path**.**join**(**\_\_dirname**,**'dist/assets/snake.html'**));**

**})**

app**.**get**(**'\*'**,** **(**req**,** res**)** **=>** **{**

res**.**sendFile**(**path**.**join**(**\_\_dirname**,** 'dist/index.html'**));**

**});**

//Set Port

**const** port **=** process**.**env**.**PORT **||** '3000'**;**

app**.**set**(**'port'**,** port**);**

**const** server **=** http**.**createServer**(**app**);**

server**.**listen**(**port**,** **()** **=>** **{**

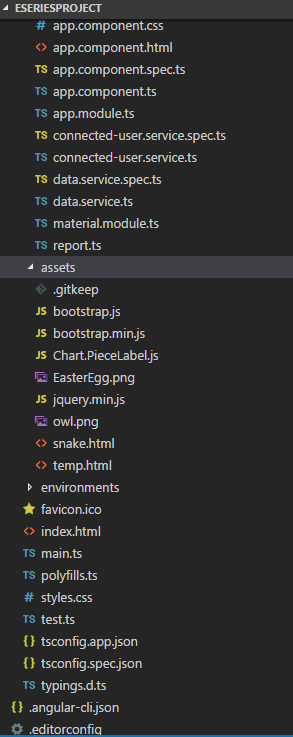
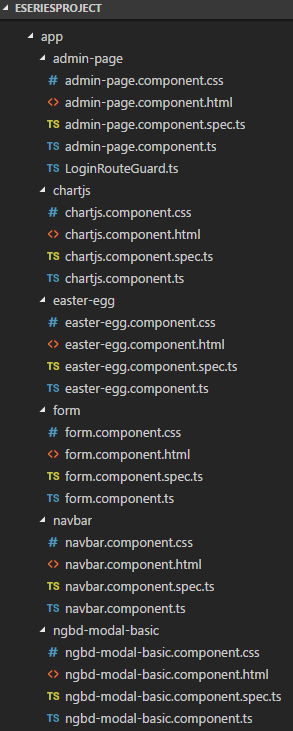
console**.**log**(**`Running on localhost:${port}`**);**

NetAppData**.**storeData**();**

**});**

# מבנה צד לקוח:

צד הלקוח מבוסס Angular 5 , שהיא מבוססת TypeScript וHTML. (TypeScript מבוסס JavaScript)זוהי הפלטפורמה המתקדמת ביותר כיום לפיתוח אפליקציות WEB. הפלטפורמה מורכבת מModules המוסיפים פונקציונליות ומComponents שהם בעצם רכיבים הבונים את ממשק המשתמש. כל Component מכיל קובץ .ts שהוא המחלקה, קובץ html שהוא המבנה, וקובץ css של עיצוב.



בפרוייקט שלי, צד הלקוח בנוי מפס ניווט (NavBar) קבוע, עליון ותחתון, וביניהם מסך המציג Components משתנים , Router-Outlet. בעצם לפי הניווט המערכת משנה את מה שמוצג על המסך מבלי לטעון מחדש את העמוד.

Admin-page: דף ניהול של המערכת, רק למשתמש שהוגדר אדמין יש גישה אליו, הקובץ LoginRouteGuard.ts מטפל באבטחת הדף.

Chartjs: Component המציג טבלאות פאי המייצגות כל אחת מערכת אחסון

Form: Component להצגת טופס

Navbar : Component המציג שורת ניווט קבועה

Ngbd-modal-basic: חלון קובץ גלובאלי של המערכת

App.component.ts: המחלקה הראשית

App.module.ts: קובץ המכיל imports למערכת

Connected-user.service.ts: שירות מערכת המנהל את המידע על המשתמש המחובר

Data.service.ts: שירות מערכת המנהל העברת מידע במערכת

Material.module.ts: imports למודולי עיצוב

Report.ts : מחלקה לאובייקט של דיווח

בתיקייה assets נמצאים קבצי עזר למערכת:

Bootstrap.js: עיצוב בוטסטראפ

Chart.pieceLabel.js: מאפשר כתיבה על פאי

EasterEgg+Owl: קבצי תמונה

Snake.html: משחק מוסתר

מחוץ לתיקייה זאת,

Index.html: הדף הראשוני של המערכת, שם נטענים כל הרכיבים ומשם מופעל מסך הloading