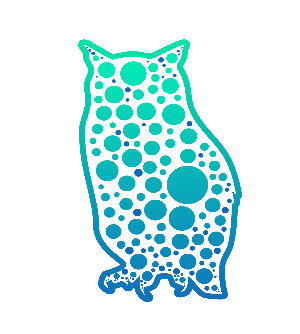
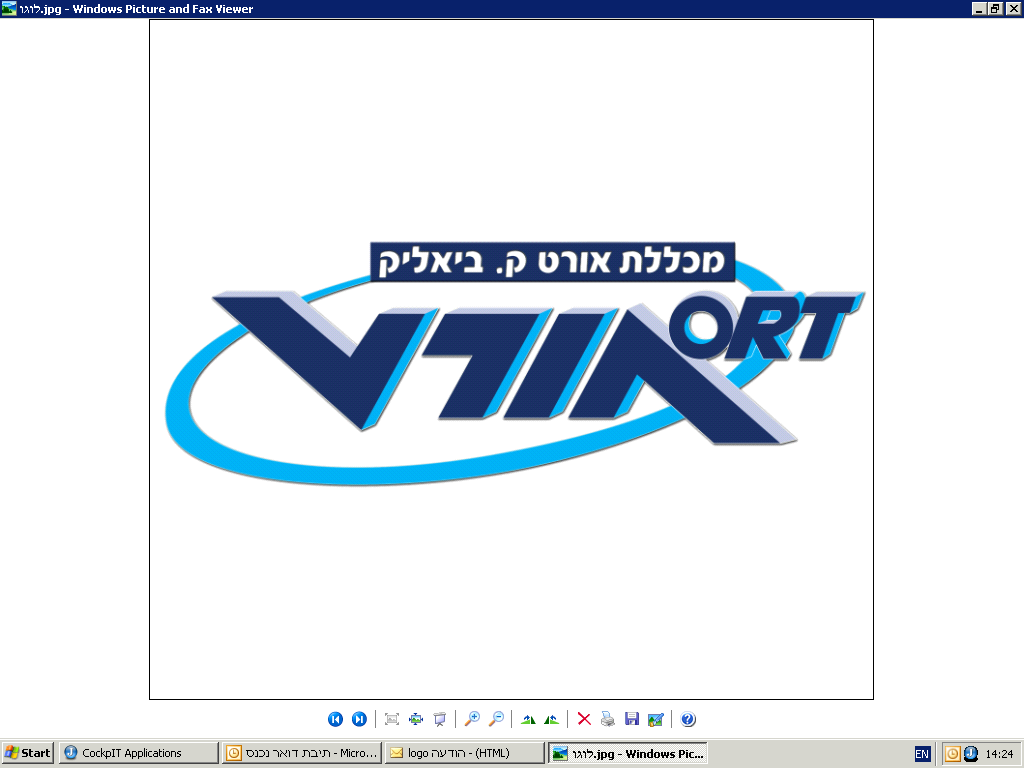
****

****

**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.*

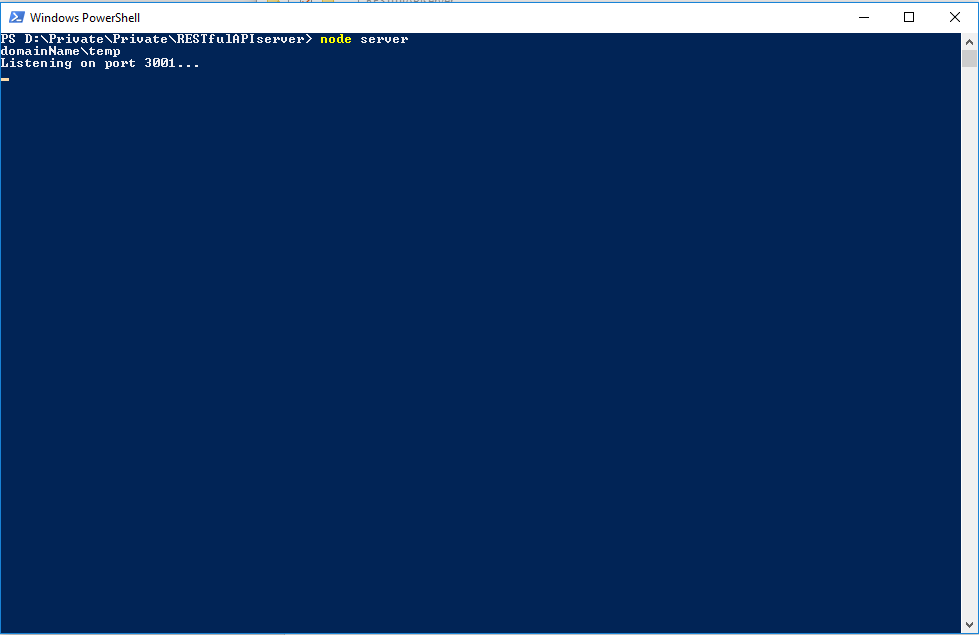
*עד סוף מרץ 2018 –  מערכת עובדת 90-100% מהתכנון המקורי.*

# הפעלת המערכת:

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

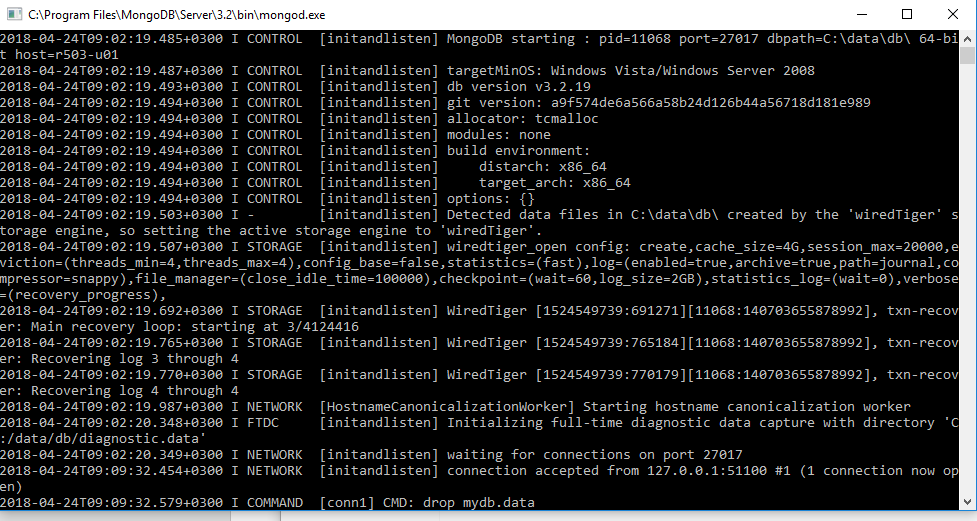
כך נפעיל את השרת:

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



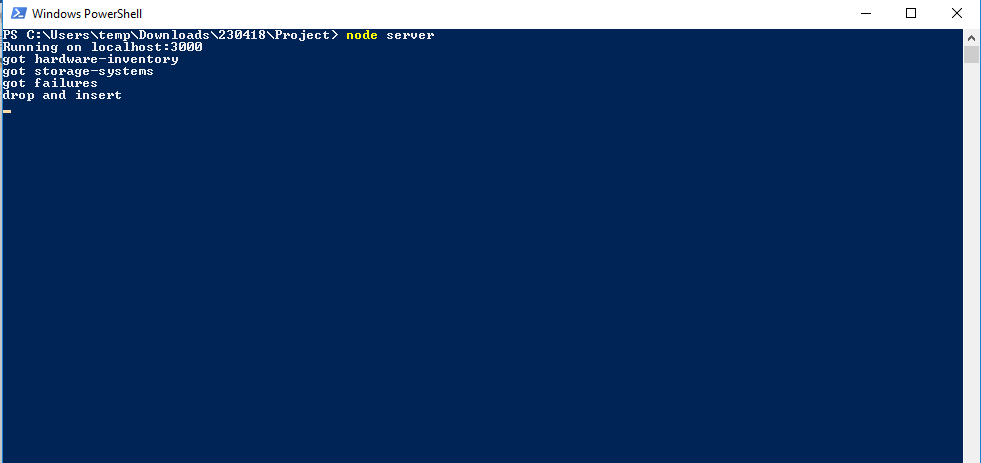
לאחר מכן, נצטרך להפעיל את מסד הנתונים (במקרה שהוא לא מותקן כservice)

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

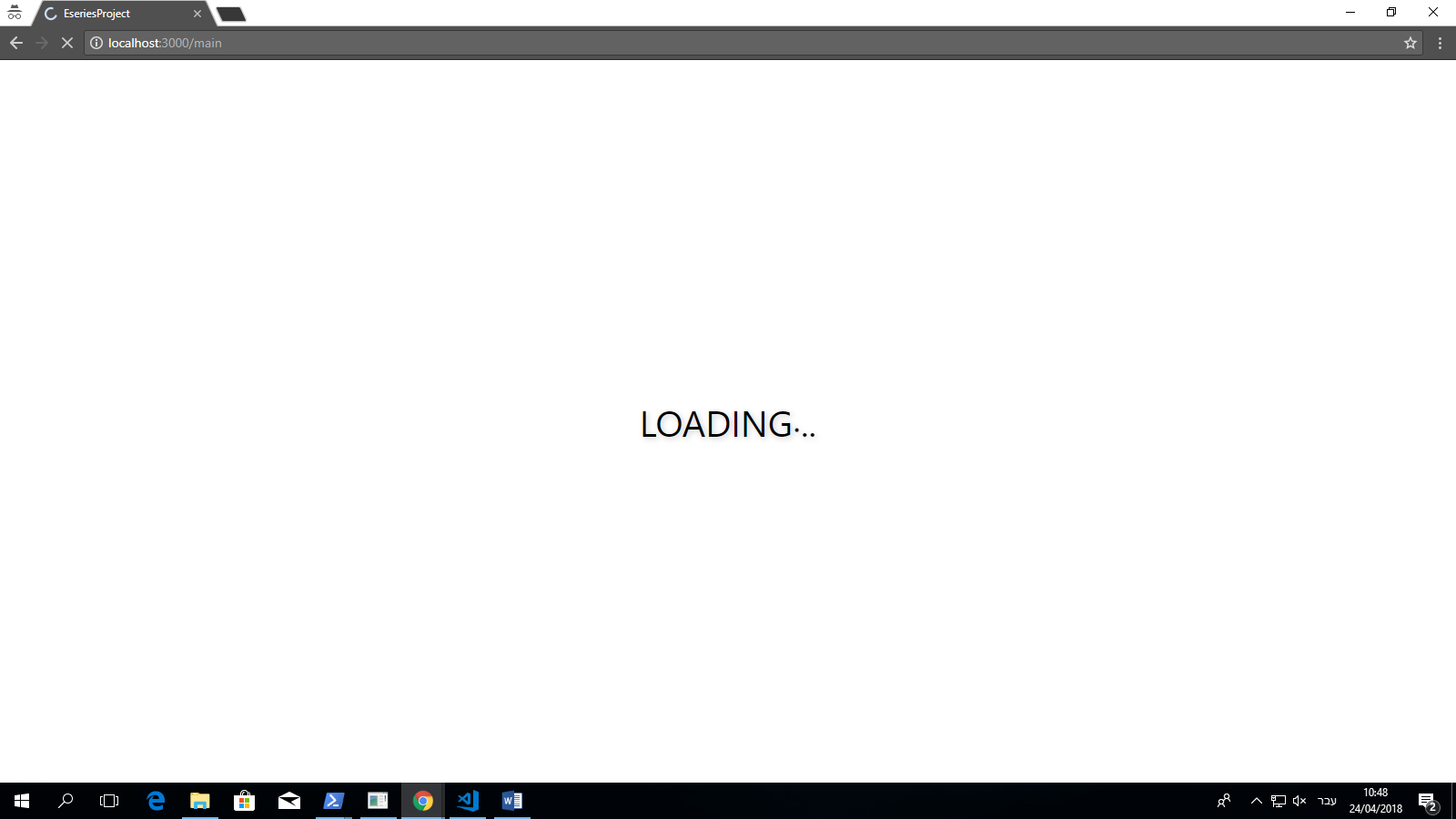
ניכנס לתיקייה של הפרויקט, נפתח חלון powershell ונכתוב את הפקודה node server (במידה ונעשה שינוי במערכת נריץ לפני את פקודה ng build).



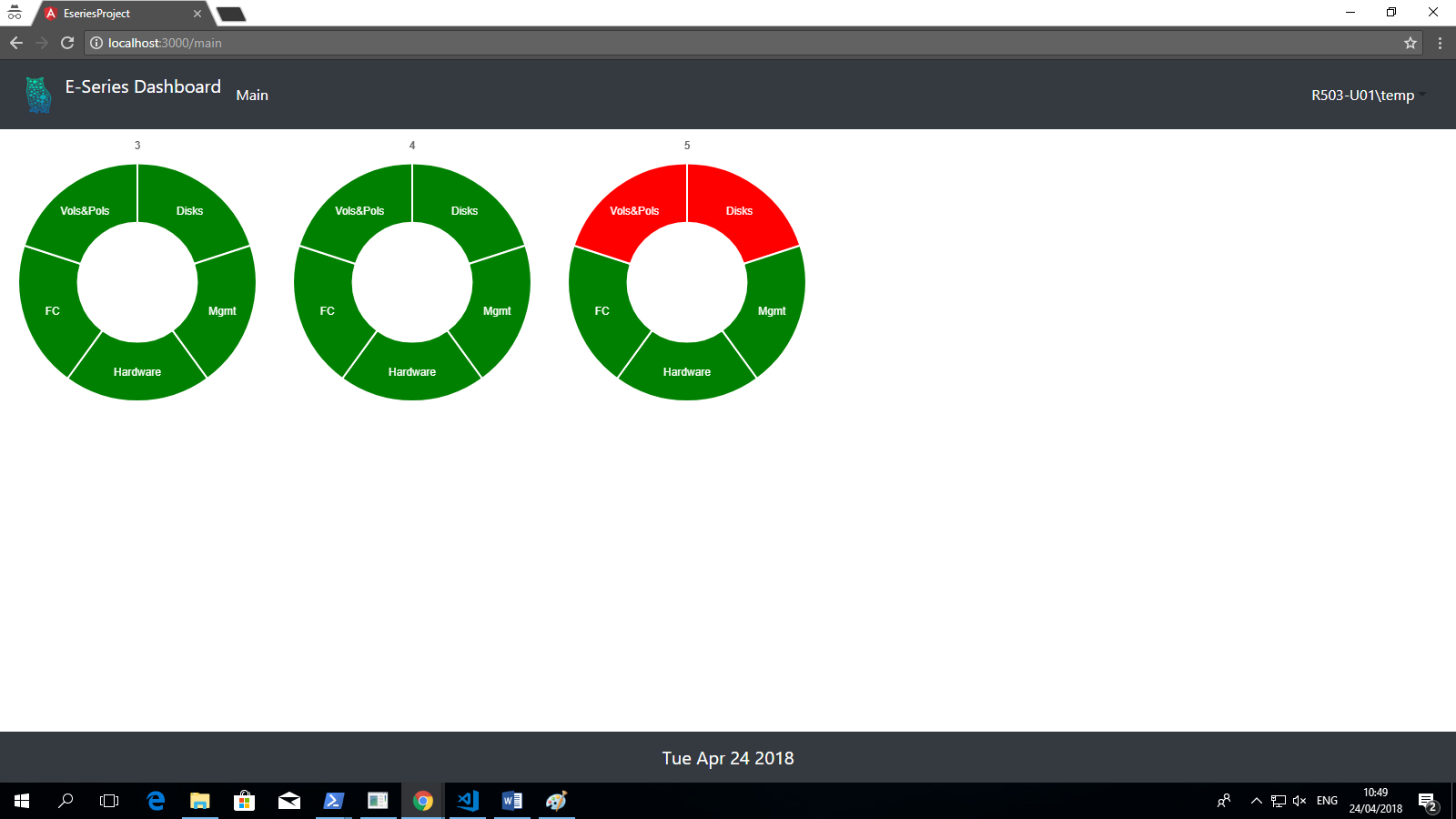
עכשיו ניכנס לדפדפן (כרום) ונכתוב: אם אנחנו במחשב שבו רץ השרת, localhost:3000

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

בהתחלה נראה מסך טעינה:



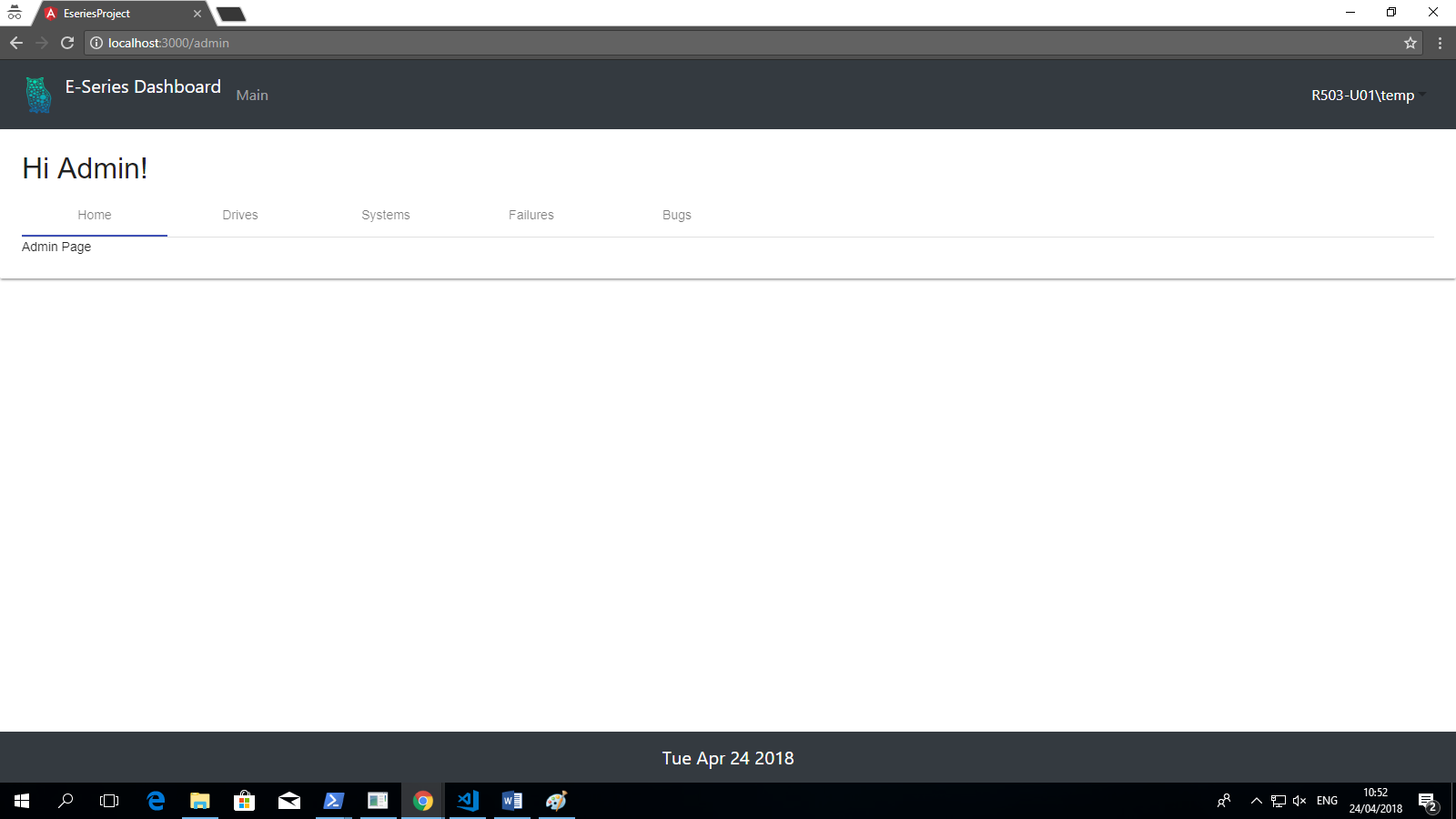
ואז יעלה המסך הראשי:



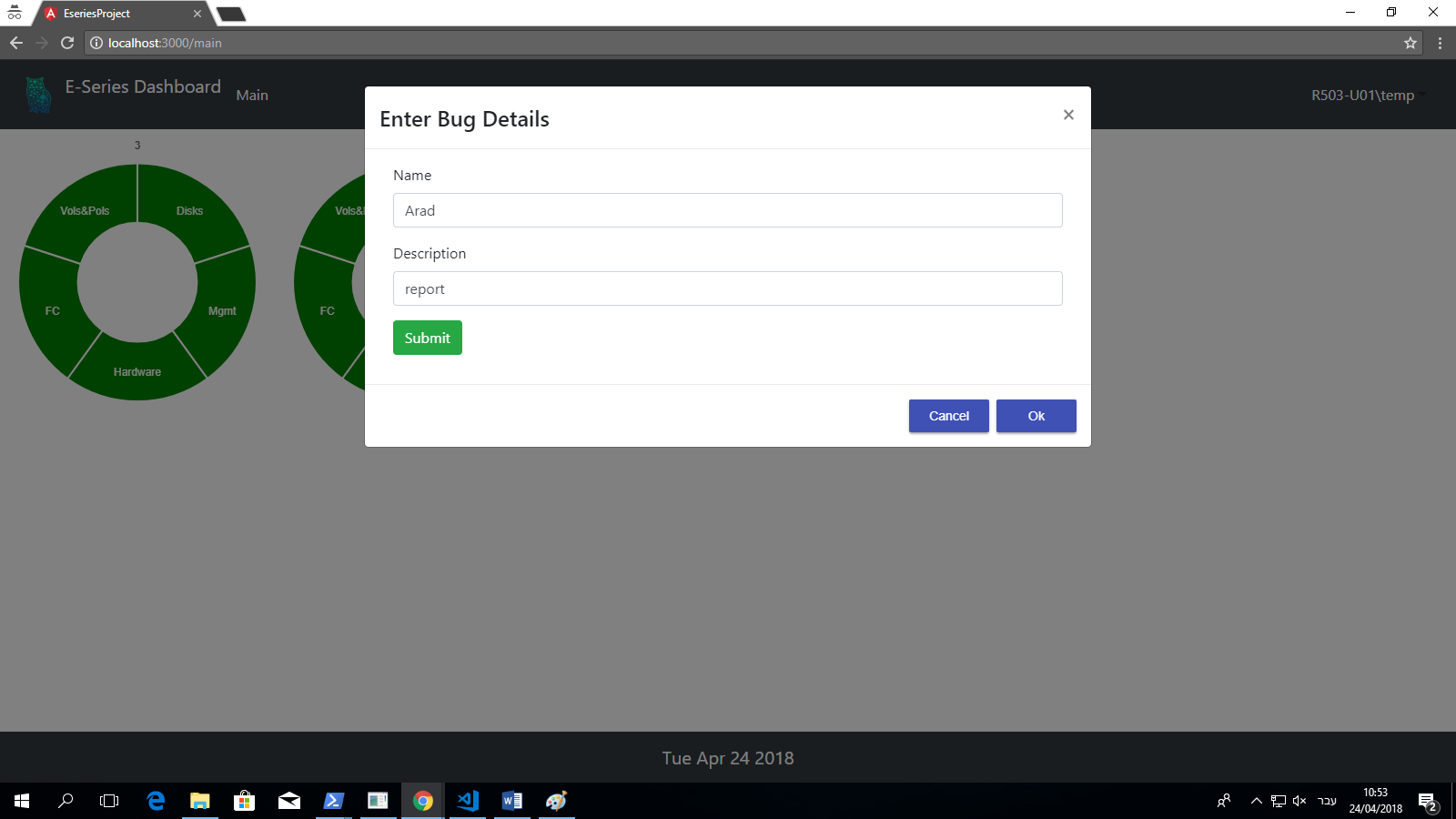
אם נלחץ על שם המשתמש, ייפתח תפריט משתמש – במידה והמשתמש המחובר מוגדר במערכת כמנהל, יופיע לו האפשרות להגיע לדף הניהול של המערכת



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