

## Abstract

The internet of things is one of the most discussed subject in IT, which brings a revolution in our interaction with the environment. In last years in this domain significant progress has been made: intelligent houses/ cities, pollution factors and production lines monitoring etc.

This work is about realizing an application that allows data acquisition from a set of IoT type sensors and following the processing of these data to generate the information in a graphic form. To demonstrate the flexibility of the platform data will be purchased from a device for monitoring physical activity and from an intelligent house, the platform functionality doesn't stop here and can be extended through the connection of other IoT devices.

Because the platform is a virtual connection environment of IoT devices it must ensure the communication of IoT devices that use as communication medium different standards. Thus the essential problem consists in standardizing the data transmitted by IoT systems to the central server, the problem being solved through a communication protocol offered by the application.

For the research side the application offers mechanisms for validating and aggregating data, and a mechanism of data presentation. Because the application follows the concept of BI (Business Intelligence) it offers graphical display tools like graphs, diagrams, schemes etc. thus giving the user the information in the desired form to be flexible in the research process.

The application offers an APP ( Application Programming Interface) to ensure further integration with:

1. Libraries for development platforms
2. Libraries for programming languages
3. Query- based applications

Because the user operates with data purchased from Io T systems in real time and wants a graphic result on a web platform, the application is developed with Node.JS technology for the implementation of the server and as a MongoDB document- oriented database.

The essential purpose of the application is to offer the necessary tools to the research of weather conditions and to be a support to the community that investigates this field.

At the moment the internet of things is in a continuous development and day by day offers more and more possibilities of interaction with the environment. In the last years there appeared more and more solutions that realize IoT devices interconnection, one of them being the IoT platforms.

The developed platform doesn't represent a direct competitor to Amazon Web Services (AWS) IoT, Microsoft Azure IoT suite, IBM Watson IoT, because it doesn't have an entire ecosystem but accomplishes the main functionalities:

- Collecting and preparing data
- Connectivity, through API
- Device monitoring
- The security implemented at authentication, authorization and data security
- Processing and analyzing data with BI elements

An ample study was effectuated in the paper on IoT domain and Business Intelligence Industry, following which it was demonstrated that between these 2 domains exists a close relationship and it will continue to develop.

There were identified and analyzed software technologies needed for the development of the IoT domain. As a result these technologies were used in the realization of the platform.

From a technological point of view the resulted platform is a very good support to start the study of IoT domain and it can be used in more serious applications. The platform is "young" and it is at the beginning, but for the future I plan its development through the publication of the source code on GitHub what will allow any developer to use and develop its function. Another method of development would be by creating a team which will improve the platform further. With the collaboration of the developers a premium solution, besides the standard one, can be achieved, a solution which will offer payable ways of IoT devices integration.

## Cuprins

Introducere .....	3
1. Fundamente ale Internet of Things .....	5
1.1. Istorie IoT.....	5
1.2. IoT ”rețea de rețele” .....	6
1.3. Ieri, astăzi și mâine pentru IoT.....	7
1.4. Avantajele IoT.....	8
1.5. Tehnologii necesare pentru dezvoltarea IoT .....	8
1.5.1. Identificarea.....	8
1.5.2. Măsurarea.....	9
1.5.3. Transmisia datelor .....	9
1.6. Probleme în dezvoltare.....	9
1.7. Securitatea .....	9
1.8. Așteptări privind conceptul IoT .....	10
2. IoT și Industria Business Intelligence .....	11
2.1. Analiza predictivă .....	12
2.2. Analiza descriptivă.....	12
3. Aplicații utilizate în cadrul sistemului .....	13
3.1. Instalare distributiv Linux in Windows 10.....	13
3.2. Sisteme de versionare a aplicațiilor.....	14
3.2.1. Instalarea Git .....	14
3.2.2. Configurarea.....	14
3.2.3. Inițializarea proiectului .....	15
3.2.4. Comenzi utilizate pe parcursul dezvoltării proiectului.....	15
3.3. Tehnologia Node.js .....	16
3.3.1. Instalarea versiunii stabile.....	17
3.3.2. Instalarea cu PPA .....	18
3.3.3. Instalarea cu ajutorul nvm .....	18
3.4. Managerul de pachete Yarn .....	19
4. Structura aplicației de BI.....	21
4.1. Aplicații Server și API (Back-end) .....	21
4.1.1. Sistemul de gestiune a bazelor de date MongoDB.....	23

4.1.2.	Conceptul aplicației.....	27
4.1.3.	Structura aplicației server.....	27
4.1.4.	Utilizarea instrumentului Postman .....	34
4.2.	Aplicații Client (front-end) .....	36
4.2.1.	Framework-ul Nuxt.js .....	36
4.3.	Interfața grafică .....	38
Concluzii .....		44
Bibliografie .....		45
Anexe - Documentația detaliată API.....		46

## Introducere

În ultimii 50 de ani omenirea are un vector foarte înalt de dezvoltare lucru care nu face alt ceva decât să ne bucure și să ne stimuleze să participăm și noi activ la acest proces. Internetul obiectelor este una din cele mai discutate teme din domeniul IT, care aduce o revoluție în interacțiunea noastră cu mediul înconjurător. Ultimii ani în domeniul dat apar progrese semnificative cum ar fi: orașe/case inteligente, monitorizarea factorilor de poluare a mediului, monitorizarea liniilor de producție și multe altele.

Lucrarea are în vedere realizarea unei aplicații care permite achiziționarea datelor de la un set de senzori de tip IoT și în urma prelucrărilor acestor date să genereze informație într-o formă grafică. Pentru a demonstra flexibilitatea platformei se va achiziționa date de la un dispozitiv pentru monitorizarea activităților fizice și de la o casă inteligentă, funcționalul platformei nu se oprește aici și poate fi extins prin conectarea și altor dispozitive IoT.

Deoarece platforma este un mediu virtual de conectare a dispozitivelor IoT trebuie să asigure comunicarea dispozitivele IoT care folosesc ca mediu de comunicare standarde diferite. Prin urmare problema esențială constă în standardizarea datelor transmise de sistemele IoT către serverul central, problema soluționându-se printr-un protocol de comunicare oferit de aplicație.

Pentru partea de cercetare aplicația oferă mecanisme de validare și agregare a datelor, și un mecanism de prezentare a datelor. Deoarece aplicația urmează conceptul de BI (Business Intelligence) oferă instrumente de reprezentare grafică cum ar fi grafice, diagrame, scheme, etc. astfel oferind utilizatorului informația în forma dorită pentru a fi flexibil în procesul de cercetare.

Aplicația oferă și un API<sup>1</sup> pentru a asigura o integrare ulterioară cu:

1. librăriile destinate platformelor de dezvoltare (embedded),
2. librării pentru limbaje de programare
3. Aplicații bazate pe interogări

---

<sup>1</sup> API - Application Programming Interface – un set de comenzi, funcții, protocoale utilizate de programator pentru a crea aplicații software.

Deoarece utilizatorul operează cu date achiziționate în timp real din sisteme IoT și dorește un rezultat grafic pe o platformă web, aplicația este dezvoltată cu tehnologia Node.JS pentru implementarea serverului iar ca bază de date orientată pe documente MongoDB.

Scopul esențial al aplicației este de a oferi instrumente necesare cercetării condițiilor meteorologice și de a fi un suport comunității care cercetează domeniul dat.

## 1. Fundamente ale Internet of Things

Internet of Things (IoT) – astăzi acest termen îl auzim la fiecare pas, tot mai des companiile mari se alătură la crearea și dezvoltarea conceptului prin crearea unor noi tipuri de CPU sau GPU pentru noile generații de dispozitive. Dar nu fiecare înțelege până la capăt ce reprezintă IoT și cât de departe vom ajunge datorită acestuia.

Definiții ale termenului IoT astăzi exista multe dar printre cele mai populare sunt următoarele: IoT – reprezintă o rețea comună de dispozitive fizice care sunt în stare să modifice parametrii proprii sau externi, să colecteze informație și să o transmită către alte dispozitive. IoT – reprezintă o rețea cu fir sau fără la care sunt conectate dispozitive autonome gestionate de sisteme inteligente, cu sistem de operare înalt cu conexiune autonomă la Internet, care pot folosi aplicații din Cloud cât și proprii, achiziționează și prelucreză date. Pe lângă acestea sunt în stare de a capta, analiza și transmite informațiile către alte sisteme. IoT – reprezintă o rețea de obiecte cu tehnologii de interacțiune între ele și mediul înconjurător [3].

### 1.1. Istorie IoT

IoT ca și alte multe concepte științifice a apărut la Institutul Tehnologic Massachusetts (MIT), în anul 1999 în cadrul lui a fost creat centrul Auto-ID, care se ocupa de identificarea prin radiofrecvență și tehnologii noi de senzori. Conceptul și termenul a fost formulat pentru prima dată de către fondatorul grupului de cercetare Kevin Ashton la prezentarea pentru conducerea Procter & Gamble (P&G). În prezentare se vorbea despre cum identificarea prin radiofrecvență va modifica sistemul de gestiune a lanțurilor logistice din companie [13].

După părerea Cisco IBSG (Internet Business Solutions Group), IoT – reprezintă un moment de timp când numărul ”obiectelor” conectate la Internet a depășit numărul de oameni. În anul 2003 populația globului era de 6.3 miliarde iar la Internet erau conectate doar 500 milioane de dispozitive. Dacă împărțim aceste 2 numere obținem următorul rezultat 0,08 de dispozitive care revin unei persoane, astfel după definiția propusă de Cisco IBSG în anul 2003 IoT nu exista. Smartphone-ul abia își făceau apariția iar Iphone a fost prezentat abia după 4 ani la 9 ianuarie 2007.

În anul 2004 în jurnalul Scientific American apare un articol dedicat IoT care demonstra cum aparatele de uz casnic, sistemele pentru casă, senzorii și alte dispozitive interacționează între ele prin diferite tipuri de comunicație și asigură o autonomie a proceselor. În sine metodele automatizării nu erau noi, dar accentul s-a pus mai mult pe conectarea dispozitivelor și "obiectelor" într-o rețea comună gestionată de protocoale Internet.

În anul 2010 datorită răspândirii agresive a smartphone-ului și tabletelor numărul dispozitivelor conectate la rețea a crescut la 12,5 miliarde, atunci când populația era de 6,8 miliarde. Astfel pentru prima dată în istorie fiecărei persoane îi revenea mai mult de un dispozitiv conectat (1,8 per persoana).

După validarea cifrelor de mai sus cercetătorii de la Cisco IBSG au concluzionat că IoT a apărut între anii 2008-2009. Această perioadă este considerată de ei nașterea IoT, deoarece în acea perioadă numărul de dispozitive conectate la rețeaua globală a depășit numărul de oameni. [3]

## 1.2. IoT "rețea de rețele"

Astăzi IoT este alcătuit din rețele slab interconectate, fiecare fiind creată pentru a rezolva o problemă. Un exemplu cunoscut pentru toți îl putem observa la oricare mașină, în care funcționează mai multe rețele pentru funcționarea motorului, securitate, semnal radio, etc. După cum se poate vedea în figura 1.1, IoT poate fi privit ca o rețea de rețele [13].

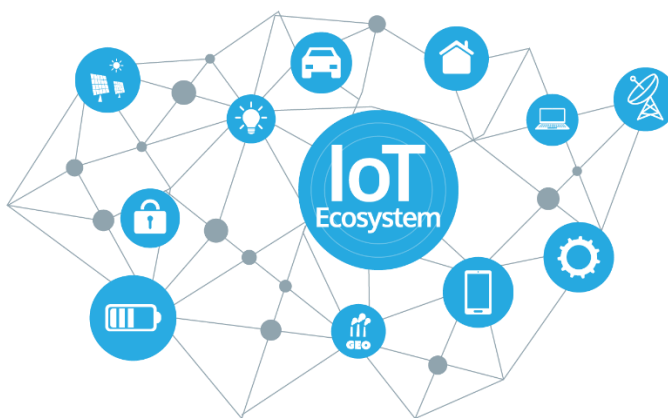


Figura 1.1 Structura IoT



În birouri și locuințe la fel există multe rețele cum ar fi sistemul de ventilare, căldură, telefonie, securitate, iluminare, etc. Odată cu dezvoltarea IoT toate aceste ”obiecte” și multe altele vor fi interconectate și vor căpăta mai multe posibilități în asigurarea securității, analizării și controlului.

Este demn de remarcat faptul că această tendință reflectă ceea ce s-a observat în primele etape ale dezvoltării tehnologiilor de rețea. La sfârșitul anilor 1980 și începutul anilor 1990, Cisco sa format ca o companie mare, tocmai datorită eforturilor sale de a conecta rețele eterogene prin rutare multiprotocol, ceea ce a făcut ca în cele din urmă IP să fie standardul de rețea comună. În ceea ce privește Internetul lucrurilor, istoria se repetă, însă pe o scară mult mai mare.

### 1.3. Ieri, astăzi și mâine pentru IoT

Înainte de a analiza semnificația IoT, este necesar să ne amintim diferențele dintre Internet și WWW(Web). Acești termeni sunt deseori utilizați ca sinonime cu toate că Internet reprezintă mai întâi nivelul fizic. Principala funcție a Internet-ului este însă asigurarea unei transmisiuni rapide și sigure dintr-un punct în altul. Web-ul însă reprezintă nivelul aplicație care funcționează datorită Internetului, scopul lui constă în crearea unei interfețe pentru a obține beneficii reale din transmisia de informație prin Internet. În dezvoltarea sa, Web-ul a trecut prin mai multe etape distincte: Prima etapă a fost cercetarea, în acea perioadă Web se numea ARPANET (Advanced Research Projects Agency Network) de care se foloseau în general doar universitățile în scopuri de cercetare. Etapa a doua poate fi numită ”broșura”, deoarece atunci fiecare companie dorea să afirme de existența ei în Internet pentru a informa oamenii despre produsele și serviciile lor. În etapa a treia s-a realizat trecerea de la informații statice la tranzacții dinamice astfel puteai să procuri produse și servicii nu doar să citești despre ele. Etapa a patra (în care suntem astăzi) este o etapă ”socială”. În această etapă companiile precum Facebook, Twitter și Google au devenit foarte populare prin faptul că au oferit utilizatorilor platforme de interacționare [16].

Spre deosebire de tehnologiile Web, Internetul nu sa dezvoltat enorm de mult și la funcționalități a rămas aproape identic ca pe timpul ARPANET. În acea perioadă existau mai multe protocoale de comunicație (AppleTalk, Token Ring și IP), dintre care până în zilele noastre a rămas doar IP.

În contextul dat IoT obține o semnificație diferită deoarece observăm modificări majore la nivelul fizic al Internetului. Acest salt calitativ ar trebui să aducă la viață aplicații noi care ar modifica rapid modul de cum trăim, învățăm, lucrăm și ne distrăm. Deja astăzi IoT a cauzat o utilizare pe scară largă a senzorilor de diferite tipuri care ne ajută să prezicem problemele dar nu să acționăm în ”mod de incendiu”.

#### 1.4. Avantajele IoT

Principalul avantaj poate fi ilustrat prin locuințe smart, care deschid ușile când stăpânul ajunge pe scări, încălzesc mâncarea, mențin temperatura optimă și multe altele. În industria automobilelor IoT va permite în primul rând o dirijare mai eficientă a traficului. Automobilele vor fi echipate cu dispozitive de poziționare care vor permite urmărirea mașinii în timp real anticipând în prealabil ambuteiajele. Pe lângă altele un automobil ”inteligent” va putea anunța poliția singur dacă va fi furat, iar în unele situații automobilul nici nu va avea nevoie de șofer, va merge singur condus de calculator [11].

IoT a ajuns și în domenii noi precum ar fi medicina, în care pacienții înghit dispozitive care permit diagnosticarea și identificarea cauzei unor anumitor boli. Senzorii microscopici pot fi fixați pe plante, animale și formațiuni geologice.

Pe de altă parte Internetul începe să meargă și în spațiul cosmic, de exemplu ca parte a programului Cisco IRIS (Internet Routing in Space).

#### 1.5. Tehnologii necesare pentru dezvoltarea IoT

Desigur pentru implementarea unei rețele globale IoT este necesar crearea unor tehnologii care să asigure: identificarea, măsurarea și transmiterea datelor [11].

##### 1.5.1. Identificarea

Odată ce un dispozitiv IoT este conectat în rețea este obligat să ofere informații asupra identității sale. Ca mijloace de identificare se pot folosi identificatori vizuali QR-cod sau chiar și dispozitiv de localizare în timp real. Cel mai important este asigurarea unicității fiecărui identificator, ceea ce ne induce cu gândul la crearea unui standard în domeniul dat. Astăzi tradițional se utilizează adresa MAC a plăcii de rețea. Adresa MAC reprezintă un șir unic din 12 de caractere în format hexazecimal, acest cod este înscris de către producătorul plăcii de rețea în stadiul de fabricare și nu mai poate fi modificat.

### 1.5.2. Măsurarea

Chiar dacă un dispozitiv primește informație de la mediu înconjurător, el trebuie să o transforme într-un format accesibil și altor dispozitive. Astăzi pentru măsurări se folosesc o clasă largă de senzori, începând de la cei mai simpli (temperatură, presiune, umiditate, etc ), până la sisteme sofisticate de calcul.

### 1.5.3. Transmisia datelor

Gama de tehnologii posibile de transmisie a datelor acoperă toate mijloacele posibile wireless și prin cablu. Astăzi cel mai interesant este standardul IEEE 802.15.4 care oferă o configurare și mentenanță ușoară. Pe baza lui există deja multe protocoale care pot fi viitorul IoT. Din tehnologiile cu cablu un rol important îl au soluțiile PLC (Power Line Communication) – tehnologia permite transmiterea informației prin intermediu liniilor electrice, soluție destul de des întâlnită datorită faptului ca în aplicațiile dorite exista deja linii electrice [9].

## 1.6. Probleme în dezvoltare

Dezvoltarea IoT va întâmpina încă multe probleme pe parcursul dezvoltării, două dintre care vor trebui soluționate în viitorul apropiat. Se are în vedere crearea unui limbaj comun de comunicare pentru senzori și dispozitive. Fără un limbaj comun rețelele nu vor putea interacționa între ele. A doua problemă este dezvoltarea de standarde comune în domeniul dat. Fără crearea acestora interconectarea rețelelor este imposibilă. Din fericire tehnologiile astăzi se dezvoltă destul de rapid și mulți producători deja sau interesat de IoT așa că este puțin probabil ca toate acestea să dureze mult.

## 1.7. Securitatea

Una din problemele importante rămâne protecția datelor. Dacă rețeaua nu va fi în totalitate sigură, nimeni nu va dori să o folosească. De exemplu persoana nu ar vrea ca șosete "smart" care să-i spună infractorului când el se culcă sau când pleacă la muncă. Pe lângă acestea folosirea tehnologiilor wireless ca mod de comunicare între dispozitive ar oferi infractorului un spectru larg de atacuri.

### 1.8. Așteptări privind conceptul IoT

În primul rând o dezvoltare extrem de rapidă a Internetului în viitorul apropiat. Cu timpul dispozitivele vor comunica între ele și cu oamenii astfel vor apărea tot mai des în toate domeniile, inclusiv în business și domeniile sociale.

Rob Van Kranenburg, teoretician și designer, afirmă că IoT va fi un tort din 4 straturi:

- primul strat va reprezenta identificarea fiecărui dispozitiv
- al doilea strat va satisface nevoile umane (ex. casa inteligentă)
- al treilea strat va fi reprezentat de achiziția și prelucrarea datelor, organizarea serviciilor și gestionarea societății pe baza informațiilor primite (ex. oraș inteligent)
- al patrulea strat – planeta senzorilor, mai multe rețele de tip MAN conectate la una singură ”globală”

Următoarea etapă de dezvoltare a conceptului dat va fi Internet of Everything (IoE) care va oferi conectarea a tot ce este posibil la rețeaua globală. Rețeaua planetară se va dezvolta singură și va lua decizii după algoritmi stabiliți de programatori.

Putem spune sigur că IoT va influența semnificativ viața unora dintre noi. Deci cum ar arăta ea peste 5 ani? Pe drumuri vor merge mașini dirijate de sisteme de control al traficului. Dimineața după trezire casa va anunța știrile, va pregăti micul dejun și va reaminti lista de acțiuni planificate pe ziua respectivă. Sistema de medicină va analiza starea persoanei bolnave va transmite datele către doctor iar apoi va primi medicamentele necesare acasă. La intrarea în magazin sistemul va anunța unde se găsesc produsele din rețetă salvată pentru mâine sau va analiza dieta prescrisă de medic [13].

La prima vedere unele exemple pot fi considerate fantastice, dar dacă analizăm mai bine un lucru este clar tehnologiile intră încet dar sigur în viața noastră de zi de zi. A rămas doar un lucru mic de unificat pe toate în IoT, ca rezultat IoT va avea mai multe oportunități pentru a deschide omenirii noi perspective.

## 2. IoT și Industria Business Intelligence

Adevărata puterea a IoT apare atunci când valorificăm puterea analizelor predictive și descriptive. În lumea BI trebuie să fim pregătiți pentru creșterea volumului de date și să ne confruntăm cu gestiunea acestora [2].

Tot mai des auzim despre modul în care IoT va transforma modul în care trăim conectându-ne la fiecare aspect din zilele noastre prin dispozitive noi, indiferent dacă este vorba de un tracker fitness, un smartwatch, un frigider sau un termostat care tocmai a fost instalat în casa noastră. Companiile mari precum Google, GE, Bosch, Samsung inclusiv și companiile de telecomunicații în ultima perioadă se concentrează pe stabilirea afacerilor IoT concentrându-se pe dispozitive, standardizări, platforme, și infrastructuri care ar permite dezvoltarea IoT.

Să luăm ca exemplu un frigider smart, în calitate de consumator putem fi liniștiți știind că frigiderul va comanda automat un nou filtru de apă la momentul potrivit înlocuirii, astfel oferindu-ne mai mult timp pentru a viziona Mr. Robot sau pur și simplu am scăpat de un lucru în plus de care putem să nu ne mai facem griji.

Din punctul de vedere al producătorului, un dispozitiv de tip IoT este benefic. Acest scenariu asigură producătorul că toate componentele de tip consumator vor fi procurate direct de la producător, sporind numărul vânzărilor.

Acest exemplu este unul banal și nu reflectă adevărata puterea IoT care constă în analiza predictivă și descriptivă.

Dispozitivele conectare generează date – o mulțime de date. De exemplu într-un termostat conectat se presupune că va măsura și înregistra continuu temperatura curentă, umiditatea, setările, nivelul bateriilor și așa mai departe. Toate aceste date sunt trimise înapoi la producător și trebuie stocate undeva, ba chiar mai mult datele trebuie procesate, dacă dorim să fie analizate. Când luăm în considerare milioanele de termostate care creează sute de puncte de date pe zi, putem înțelege rapid mărimea acestei provocări. Pentru a soluționa problema dată furnizorii precum Amazon Web Services, Microsoft Azure și Hadoop vor juca un rol important în IoT, deoarece oferă soluții de stocare, scalabilitate și procesarea cantităților mari de date. De exemplu Amazon, oferă servicii precum Kinesis (serviciu de

streaming în timp real), S3 (Simple Storage) și DynamoDb, care este o bază de date scalabilă NoSQL. Companiile IoT investesc în prezent în infrastructură, astfel în următorii ani după ce fundația va fi pusă la dispoziție oricărui doritor vor apărea cereri de analiză și prelucrare a datelor furnizate de dispozitivele IoT [12].

### 2.1. Analiza predictivă

Să ne imaginăm că suntem manager într-o instalație, prioritatea mea este să mă asigur ca timpul de exploatare a instalației este de 100%, costurile de întreținere sunt menținute la nivel minim și echipa mea are un mediu de lucru sigur. Într-o instalație în care liniile de asamblare, motoarele și alte componente sunt conectate la Internet și datele privind performanțele lor sunt transmise înapoi la producătorul echipamentului.

El le poate utiliza pentru o analiză de precizie pe baza a milioane de puncte de acces de date de la alte dispozitive similare ca să-și dea seama care dispozitive în viitorul apropiat se vor defecta. Mai mult, poate identifica un timp exact al defecțiunii în cazul în care dispozitivul funcționează la parametrii actuali. Astfel, știind când se va întâmpla ceva cu dispozitivele mele, voi putea planifica mai bine verificarea/înlocuirea lor și pierderile vor fi minime [7].

### 2.2. Analiza descriptivă

Analiza descriptivă merge cu un pas mai departe prin recomandarea unei acțiuni de prevenire a defecțiunii dispozitivului. Acest lucru ar putea însemna reducerea RPM mașinii în timp ce informează operatorul că este necesară intervenție, dacă se prevede o situație gravă mașina se poate opri pentru a micșora costurile de înlocuire sau răni personalul care operează.

Cu toate acestea analiza descriptivă nu se bazează doar pe ce s-a întâmplat în trecut sau este probabil să se întâmple în viitor, dar ia în considerare și rezultatele dorite, scenariile specifice, precum și toate datele actuale. Acest lucru permite determinarea modului optim pentru a trata un scenariu viitor, astfel vom depăși nu doar problemele de întreținere și defecțiuni.

În cazul unei fabrici de producție, analizele descriptive ar putea deduce o soluție optimă la o mașină pentru a modifica parametrii de funcționare astfel măbind randamentul de lucru [7].

### 3. Aplicații utilizate în cadrul sistemului

#### 3.1. Instalare distributiv Linux in Windows 10

Datorită faptului ca cei de la Microsoft au făcut posibilă integrarea subsistemului Linux direct in Windows viața dezvoltatorilor de aplicații s-a simplificat enorm. Acum dezvoltatorii pot instala subsisteme de Linux pe care le pot utiliza în propriile proiecte astfel eliminând mașinile virtuale și alte aplicații de virtualizare [5].

Începând cu actualizarea Fall Creators Update (1709) instalarea subsistemului Linux pentru Windows a ieșit din stadiul beta iar în pașii de instalare au survenit unele modificări [17].

1. Mai întâi de toate trebuie pornită componenta ”Windows Subsystem for Linux” din ”Control panel” -> ”Turn Windows features on or off” (figura 3.1), sau executând în PowerShell comanda:

```
Enable - WindowsOptionalFeature - Online- FeatureName Microsoft - Windows Subsystem - Linux
```

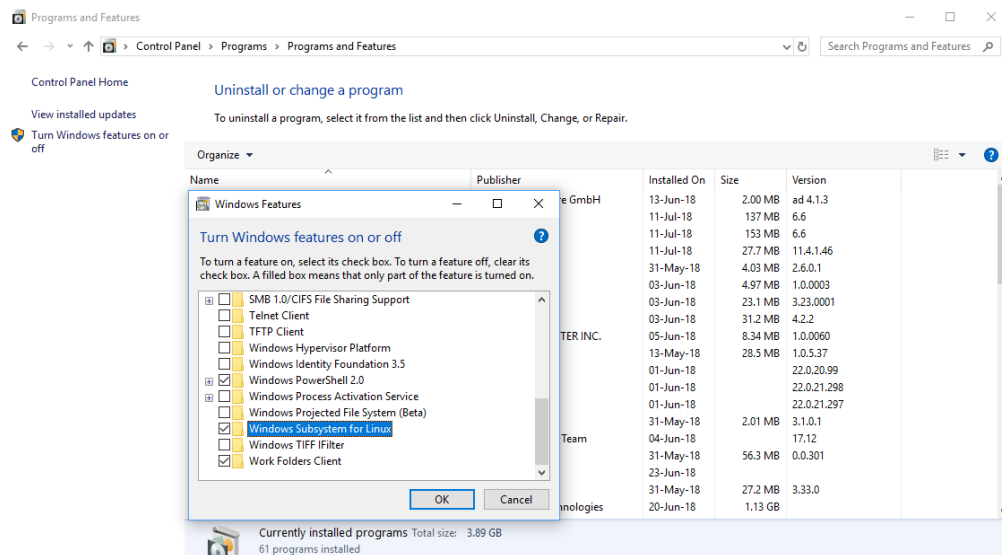


Figura 3.1 Instalarea subsistemului Linux din fereastra Turn Windows features on or off

2. După instalarea componentei și repornirea calculatorului, în aplicația Microsoft Store se identifică distributivul Linux dorit, de exemplu Ubuntu, openSUSE, Debian urmată de descărcarea cu doar un click. În aplicație a fost instalat Ubuntu 18.04.

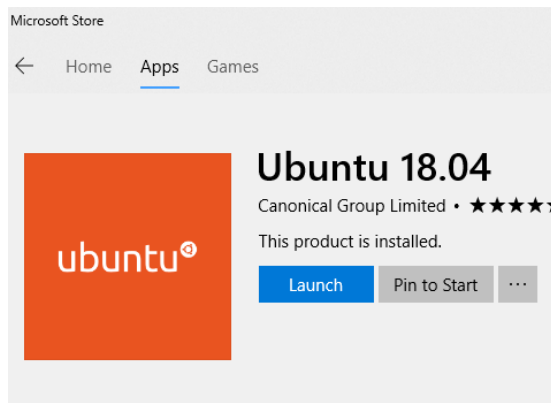


Figura 3.2 Distribuția Ubuntu 18.04 din Windows Store

3. Apoi se pornește ca o simplă aplicație iar la primul start se configurează utilizatorul root.

### 3.2. Sisteme de versionare a aplicațiilor

Sistemele de versionare se utilizează foarte des în ultima perioadă în dezvoltarea oricărui tip de software indiferent dacă dezvoltarea se face de unul singur sau în echipă. Aplicațiile de tipul dat permit supravegherea modificărilor în codul sursă și dacă se dorește se poate reveni la starea precedentă [14].

Una din cele mai cunoscute aplicații de tipul dat este Git. Mulți utilizează această aplicație pentru dezvoltarea proiectelor iar site-urile GitHub și Bitbucket au făcut partajarea codului sursă între dezvoltatori foarte simplă.

#### 3.2.1. Instalarea Git

Cea mai simplă metoda de instalare a aplicației Git pentru Ubuntu este cu ajutorul managerului de pachete apt. Aceasta este cea mai simplă metodă însă versiunea aplicației nu va fi ultima. Pentru instalarea Git mai întâi este necesară actualizarea pachetelor din repo, prin executarea următoarei comenzi [19]:

```
sudo apt update  
iar pentru instalare
```

```
sudo apt install git
```

După ce s-a terminat instalarea urmează configurarea aplicației respective.

#### 3.2.2. Configurarea

Înainte de a începe utilizarea Git trebuie să se realizeze autorizarea, altfel Git va da eroare. Pentru autorizarea globală pentru toate proiectele trebuie setate numele și adresa de email. Parametrii pot fi setați prin următoarele comenzi [19]:



```
$ git config --global user.name "Nume Prenume"  
$ git config --global user.email "adresaEmail@domain.com"
```

Pentru a vedea dacă setările au fost aplicate se execută comanda `config --list`, iar rezultatul trebuie să fie identic cu valorile introduse mai sus:

```
$ git config --list  
user.name "Nume Prenume"  
user.email "adresaEmail@domain.com"
```

### 3.2.3. Inițializarea proiectului

Site-ul utilizat pentru încărcarea codului sursă este Bitbucket deoarece permite crearea de repertorii (repository) private.

Se creează un repository cu titlul "licenta", apoi în linia de comandă se creează dosarul de lucru cu comanda `cd`, după accesarea acestuia se execută comanda de mai jos care va inițializa și Git local:

```
$ git clone git@bitbucket.org:picaso133/licenta.git
```

### 3.2.4. Comenzi utilizate pe parcursul dezvoltării proiectului

#### a) Vizualizarea stării curente

```
$ git status
```

#### b) Adăugarea tuturor fișierelor, inclusiv din subdirectoare

```
$ git add .
```

#### c) Salvarea modificărilor în baza locală, unde mesaj reprezintă un scurt mesaj informativ

```
$ git commit -a -m "mesaj"
```

#### d) Pentru sincronizarea proiectului local cu serverul extern sunt necesare următoarele comenzi:

```
$ git remote add origin git@bitbucket.org:picaso133/licenta.git  
$ git push -u origin master
```

#### e) Iar pentru actualizarea proiectului de pe server cu cel local

```
$ git pull
```

f) Pentru salvarea tuturor modificărilor locale pe server

```
$ git push
```

g) De asemenea se permite operații cu ramuri

```
$ git branch nume // creează o ramură cu numele specificat  
$ git branch // afișează toate ramurile  
$ git checkout nume // schimbarea ramura  
$ gitk --all // vizualizarea log-urilor pe toate ramurile  
$ git merge NUME // combină doua ramuri
```

### 3.3. Tehnologia Node.js

Programarea Web la momentul actual se află într-o creștere continuă, iar la începutul creării unui proiect trebuie să alegem dintre limbajele universale cu o istorie în spate cum ar fi Perl, C++ sau Java ori limbajele noi apărute orientate spre Web cum ar fi Ruby, GO, JavaScript. Mulți ani la rând dezvoltatorii web clasificau JS mai mult pentru partea de realizare a aplicației client și nu îi preziceau un viitor favorabil, totul până în momentul când Google a prezentat motorul V8 pentru JS. După apariția motorului apare Node.js care aduce ideea de a utiliza JS ca și limbaj pentru realizarea aplicațiilor de tip server [1].

Node.js se află într-o dezvoltare foarte dinamică și agresivă. Astfel în decursul a câtorva ani de dezvoltare au apărut peste 200000 de module pentru Node.js, ceea ce depășește tempo-urile dezvoltării limbajelor deja existente precum Perl care are mai puține module. Tehnologia Node.js prinde popularitate printre rândurile dezvoltatorilor și datorită acestui fapt ajunge și în companiile mari ca Yahoo, Microsoft, PayPal, LinkedIn, mai ales Google.

Node.js este proiectat pentru crearea aplicațiilor web sau mobil, în care pe partea back-end sunt necesare modificări în timp real, folosind o infrastructură pe bază de micro-servicii. Node.js poate reduce în mod semnificativ timpul de dezvoltare a aplicației fără să afecteze logica aplicației.

În acest caz timpul poate fi considerat o resursă esențială și critică atât pentru servere cât și pentru dispozitivele conectate la acesta. Timpul real reprezintă timpul de răspuns al sistemului la totalitatea modificărilor apărute. Există o legătură importantă între comportarea în timp real și inerția procesului [8].

Încă un punct în plus pentru Node.js este consumul mic de resurse CPU, RAM și putere de procesare, acest fapt se datorează infrastructurii de tip micro-servici. Acest lucru poate fi vizibil la un proiect cu multe date intrare/ieșire deoarece Node.js permite simularea serviciilor paralele, bazându-se pe un flux de execuție care nu necesită tehnici complicate de programare paralelă (figura 3.3) [15].

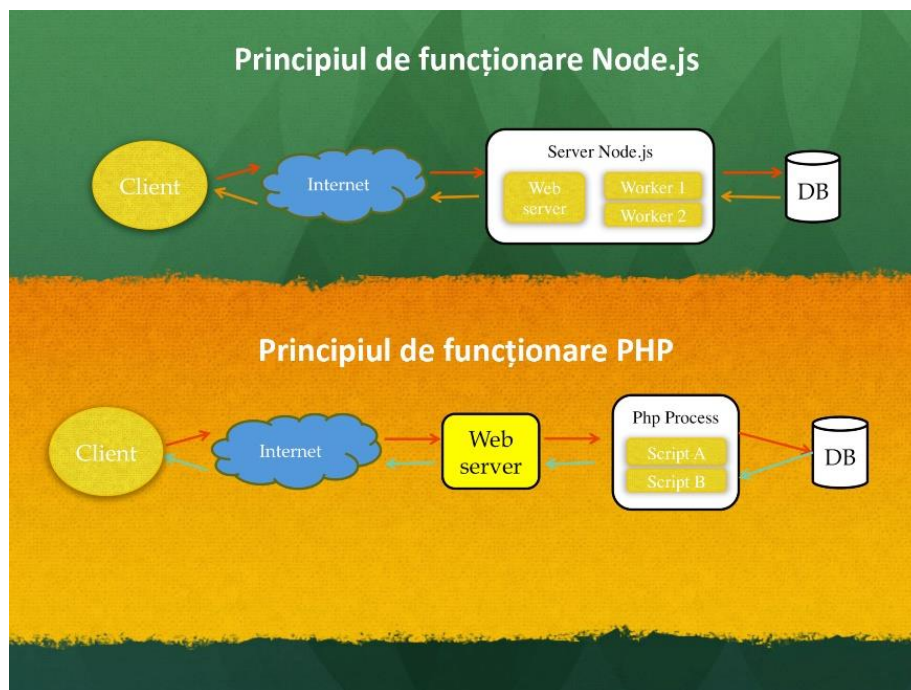


Figura 3.3 Flux de lucru comparative între un server simplu și unul cu node.js

Pentru construcția aplicației pot fi utilizate module deja existente care urmează doar a fi adaptate la cerințele aplicației. De exemplu unul dintre cele mai cunoscute framework-uri este "Express", baza de date poate fi folosită una relațională precum MySQL sau ne-relațională MongoDB, și multe alte module.

### 3.3.1. Instalarea versiunii stabile

Ubuntu 18.04 oferă din repository standard versiunea 8.10.0 (la momentul scrierii lucrării). Desigur versiunea dată nu este ultima, dar această versiune este stabilă și ușor de instalat. Pentru început actualizăm pachetele și apoi executăm comanda de instalare [18]:

```
$ sudo apt update
$ sudo apt install nodejs
```

Pentru a se asigura că Node.js a fost instalat, un programator execută comanda utilizată pentru afișarea versiunii instalată de Node.js

```
$ nodejs -v
```

### 3.3.2. Instalarea cu PPA

Utilizând arhiva PPA<sup>2</sup> acceptată de NodeSource se poate obține o versiune mai recentă a Node.js. În acest caz se poate alege una din următoarele v4.x(valabilă până în aprilie 2019), v8.x(LTS cu suport până în decembrie 2019) și v10.x(ultima versiune cu suport până în aprilie 2021) [18].

Pentru a instala PPA în catalogul local, se folosește curl<sup>3</sup> pentru a descărca script-ul de instalare și bash pentru executarea lui.

```
$ cd ~  
$ curl -sL https://deb.nodesource.com/setup_10.x -o nodesource_setup.sh  
$ sudo bash nodesource_setup.sh
```

După executarea script-ului de instalare arhiva PPA va fi adăugată în setări și pachetele se vor actualiza automat. Astfel programatorului nu îi rămâne decât să instaleze Node.js

```
$ sudo apt install nodejs  
$ nodejs -v  
$ v10.0.0
```

### 3.3.3. Instalarea cu ajutorul nvm

Managerul nvm<sup>4</sup> permite instalarea mai multor versiuni de Node.js pe un singur server. Astfel se poate seta versiuni diferite fiecărui proiect [18].

Pentru instalarea nvm trebuie la fel sa se descărce scriptul de instalare. Pentru a descărca ultima versiune de nvm se accesează repository-ul oficial de pe GitHub : <https://github.com/creationix/nvm>

```
$ curl -sL https://raw.githubusercontent.com/creationix/nvm/v0.33.11/install.sh -o  
install_nvm.sh  
$ bash install_nvm.sh
```

Lista comenzilor utile nvm:

```
$ nvm ls-remote // afișează lista accesibilă a versiunilor Node.js  
$ nvm install 10.1.0 //instalează versiunea dorită
```

---

<sup>2</sup> PPA - Personal Package Archives – permite încărcarea pachetelor sursă pentru Ubuntu ce pot fi publicate ca repository aptitude.

<sup>3</sup> Curl – comandă utilizată pentru transfer de date

<sup>4</sup> Nvm - Node Version Manager

```
$ nvm use 10.1.0 // selectează versiunea dorită  
$ node -v //afișează versiunea curentă care se folosește
```

Din cele prezentate mai sus există diferite metode de instalare a Node.js pe un server cu sistem de operare Ubuntu 18.04, și fiecare are puncte forte. Metoda de instalare depinde de cerințe și specificații, dar nvm oferă cea mai mare flexibilitate.

### 3.4. Managerul de pachete Yarn

Yarn este un nou manager de pachete, creat împreună de Facebook, Google, Exponent și Tilde [20]. Conform documentației oficiale, Yarn a fost creat să rezolve o serie de probleme întâlnite de dezvoltatori la dezvoltarea proiectelor cu ajutorul npm și anume:

- Instalarea pachetelor nu era foarte rapidă și continuă
- Existau probleme de securitate, deoarece npm oferă posibilitatea pachetelor de a executa cod în timpul instalării

Yarn este un client nou care descarcă aceleași module din repository-uri npm, prin urmare nu avem de ce să ne facem griji că viitoare module nu vor mai fi publicate în repository-urile npm.

După prima vedere Yarn nu se deosebește foarte mult de npm, dar există deosebiri la nivel de implementare.

#### 1. Fișierul yarn.lock

În acest fișier se înregistrează modulele care se instalează. Astfel se garantează ca pe alt calculator se va instala aceeași versiune a modulului. La npm lucrurile merg mai diferit, el lucrează cu diapazoane de versiuni luate din fișierul package.json unde se poate întâmpla ca pe calculatoare diferite să fie instalate versiuni diferite ale modulelor ceea ce poate duce la bug-uri [20].

#### 2. Instalarea paralelă

Indiferent dacă se instalează un modul cu npm sau Yarn rezultatul constă în efectuarea aceleași operații. În npm toate operațiile se execută secvențial și separat pentru fiecare modul, însemnând faptul că până dacă un modul se instalează celălalt așteaptă. În Yarn aceste operații se efectuează paralel, lucru care influențează performanța [20].

Pentru comparație s-a instalat modulul express cu npm și Yarn, modulul conține 42 fișiere: npm: 9s Yarn: 1.37s. Modulul gulp cu 195 de dependențe: npm: 11s Yarn: 8.81s.

### 3. Comenzile Yarn

Pe lângă diferențele funcționale, în Yarn comenzile diferă. Unele comenzi npm au fost ș terse, unele modificate și au fost adăugate câteva noi [20].

```
$ yarn global
$ yarn global //efectuează setări globale
$ yarn install //instalează toate dependențele din fișierul package.json
$ yarn add [-dev] //adaugă și instalează dependențele parametrul
$ yarn licenses [ls|generate-disclaimer] //afișează lista licențelor sau generează un fișier
$ yarn upgrade //actualizează pachetele compatibile cu diapazonul din package.json
$ yarn generate-lock-entry //generează fișierul yarn.lock pe baza dependențelor din package.json
```

## 4. Structura aplicației de BI

### 4.1. Aplicații Server și API (Back-end)

În anul 2009 platforma Node.js a făcut primii pași în domeniul back-end-ului, această a fost prima încercare de a folosi JavaScript în aplicațiile de tip server. Astăzi însă este destul de dificil să găsești un dezvoltator care nu a auzit despre Node.js, însă câțiva ani în urmă încercările prin care a trecut platforma nu îi garantau un succes sigur. Platforma data a supraviețuit divizării, a fost subiectul multor ”războaie” pe forumuri și a adus mari bătăi de cap dezvoltatorilor [21].

Node.js nu doar că a făcut o revoluție în back-end dar și în front-end deoarece a adus un interes enorm pentru V8<sup>5</sup> și joacă un rol semnificativ în extinderea întregului ecosistem JavaScript prin îmbunătățirea framework-urilor cum ar fi React, Angular sau Vue.

De-a lungul timpului Node.js a reușit să schimbe părerile apărute la început de drum, cum ar fi:

- **Codul JS pentru Node.js este dificil de depanat.**

Pentru a depana aplicațiile JS de pe server, se pot utiliza aceleași tehnici ca la depanarea codului client, utilizând node-inspector, sau folosirea unui program universal de depanare ca DTrace.

- **Node.js nu poate fi utilizat la dezvoltarea aplicațiilor de tip enterprise.**

Această părere la fel nu corespunde realității. Cu Node.js se pot dezvolta aplicații enterprise, dificultatea constă însă în faptul că nu există instrumente încorporate care ar simplifica crearea acestora.

- **JavaScript nu a fost creat ca limbaj de programare pentru back-end**

În apărare putem spune că JS deja putea lucra pe servere din momentul când Netscape a încorporat suportul acestui limbaj în browser-ul propriu. Acest lucru sa întâmplat în 1995. JS este considerat limbaj pentru front-end, dar acest lucru este datorat doar faptului că a acaparat acest domeniu.

---

<sup>5</sup> V8 – un motor care compilează direct JavaScript în cod mașină

Acum, trebuie discutat despre scenariile de utilizare a Node.js și limitările sale pentru a înțelege mai bine locul acestei tehnologii în lumea modernă.

### **Avantaje și caracteristici generale**

- Dacă partea ce ține de front-end este scrisă în JS, atunci pentru a obține un cod universal ideal ar fi să folosim JS și în back-end
- Instrumentele precum webpack ajută la reutilizarea codului atât pe client cât și pe server, ceea ce duce la uniformitatea codului în toate nivelurile sistemului
- Folosind JS pe client și server aplicația creată poate fi randată în browser cât și pe server.
- Apariția `async/await` a schimbat complet abordarea de scriere a codului asincron.

Unii dezvoltatori văd un minus faptul că atât serverul cât și clientul folosesc JS astfel obligându-i să folosească JS. Dar nu este chiar așa, dacă dorim putem accesa biblioteci specializate ale altor limbaje de programare, printr-un API.

Însumând cele spuse mai sus putem aduce 3 scenarii de utilizare în care Node.js își arată adevărata putere.

#### **• Scenariul 1: Aplicații în timp real**

Aplicații pentru lucru în echipă, chat-uri interactive, sisteme de schimb instant de informație și jocuri online, toate sunt exemple de aplicații în timp real, în dezvoltarea cărora Node.js este instrumentul perfect pentru crearea lor. Pentru o funcționare adecvată a aplicațiilor din domeniul dat trebuie asigurată o viteză mare de răspundere la acțiuni și întârzieri minime, caracteristici pe care Node.js le deține [12].

#### **• Scenariul 2: Aplicații pe o singură pagină**

O aplicație pe o singură pagină este reprezentată de o pagină web încărcată în browser, a cărei conținut se actualizează dinamic în timpul interacțiunii cu utilizatorul. Cea mai mare parte a volumului de lucru pentru astfel de aplicație îi revine clientului.

Chiar dacă aplicațiile de o singură pagină reprezintă un pas important în evoluția dezvoltării web, acestea au de asemenea unele probleme care privesc asupra modului de randare. În special această problemă poate afecta grav motorul de căutare care nu poate indexa paginile randate de browser. O soluție populară pentru problema dată este randarea pe server [12].



- **Scenariul 3: Scalabilitatea**

Serverele de Node nu vor fi niciodată mai puternice decât este nevoie. Frumusețea arhitecturii Node constă în minimalism, astfel aplicația back-end poate fi scalată în funcție de nevoile proiectului. Conceptul oferit de Node ne permite să avem mai multe noduri distribuite care pot face schimb de date între ele. Astfel putem eficient mări resursele cerute de aplicație sau micșora când este cazul. Divizarea aplicației în mai multe noduri este un proces care necesită atenție maximă iar în mâinile unui programator începător poate duce la urmări fatale [12].

#### 4.1.1. Sistemul de gestiune a bazelor de date MongoDB

MongoDb implementează o abordare nouă a construcției bazelor de date, în care nu există tabele, scheme, interogări SQL, chei străine și multe alte lucruri prezente în bazele de date relaționale. De când există dinozaurii suntem obișnuiți să stocăm toate datele în baze de date relaționale (MS SQL, MySQL, Oracle, PostgreSQL) și ne-am pus întrebarea dacă bazele de date relaționale sunt potrivite pentru aceste date [6].

Spre deosebire de bazele de date relaționale, MongoDB oferă un model de date orientat pe documente, ceea ce face MongoDB mai rapid, are o scalabilitate mai bună și este mai ușor de utilizat.

Luând în considerare toate minusurile bazelor de date relaționale și avantajele MongoDB, este important de înțeles că cerințele pot fi diferite și metodele de rezolvare sunt diferite. În unele situații MongoDB poate îmbunătăți performanțele aplicației spre exemplu când trebuie să salvăm date cu structură dificilă, iar în alte situații bazele de date relaționale ar fi o soluție mai bună. În unele situații putem să le folosim pe ambele.

Întreg sistemul MongoDB poate reprezenta nu numai o singură bază de date, situată pe același server fizic. Facilitățile MongoDB permit plasarea mai multor baze de date pe servere fizice diferite care pot ușor schimba date între ele păstrând integritatea.

#### **Formatul de date în MongoDB**

Unul din cel mai popular standard de schimb și stocare a datelor este JSON (JavaScript Object Nation). JSON descrie eficient structuri de date complexe. Metodă de stocare a datelor în MongoDB este asemănătoare cu JSON dar nu se aplică formal JSON. Pentru salvarea datelor în MongoDb se utilizează formatul BSON (binary JSON) [10].

BSON permite operarea mai rapidă a datelor: căutarea și editarea se efectuează mai repede. Singurul minus al BSON față de JSON este că ocupă mai mult spațiu, însă minusul este compensat de performanța oferită.

MongoDB este scris în C++, deci este ușor de portat pe diferite platforme. MongoDB poate fi instalată pe platformele Windows, Linux, Solaris, MacOS

Dacă bazele de date relaționale salvează rânduri, MongoDB salvează documente. Spre deosebire de rânduri, documentul permite salvarea datelor cu structură complexă. O altă deosebire constă în faptul că un document poate fi reprezentat ca un depozit de chei și valori. O cheie reprezintă o etichetă simplă cu care este asociată o anumită parte din date. Cu toate acestea, în ciuda diferențelor, există o asemănare între MongoDB și bazele de date relaționale. În MongoDB fiecare document are un identificator unic denumit `_id`. Dacă nu este specificată valoare sa MongoDB va genera automat o valoare pentru el [10].

Dacă în bazele de date relaționale există tabele, atunci în MongoDB există colecții. Dacă în bazele de date relaționale se salvează obiecte stabile, colecțiile permit salvarea diferitor obiecte cu structuri și proprietăți flexibile.

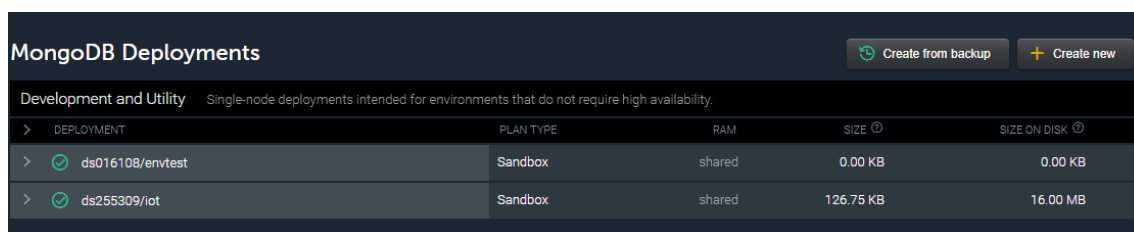
Structura păstrării datelor în MongoDB este reprezentată de replici. În acest set există un nod principal sau pot exista mai multe noduri secundare. Toate nodurile secundare mențin integritatea și se actualizează automat cu nodul principal, astfel dacă nodul principal se defectează un nod secundar îi ia rolul.

Absența unei scheme stabile permite o mobilitate enormă, astfel nu trebuie să modificăm schema la fiecare modificare a structurii datelor salvate. În acest fel se obține o scalabilitate enormă și nu trebuie să recreăm baza de date împreună cu interogări noi.

#### *4.1.1.1. mLab*

Pentru crearea bazei de date s-a utilizat serviciul de hosting gratuit mLab. Lucrul cu acest serviciu este foarte ușor și pentru proiecte mici este gratuit. Pentru început este nevoie de un cont, lucru care se face într-un singur click. Apoi tot ce rămâne de făcut e să se creeze baza de date, efectuând click pe "Create New", după cum se vede în figura 4.1.

În fereastra care apare rămâne de selectat furnizorul și tipul planului. Baza de date a aplicației este găzduită la furnizorul Amazon deoarece permite selectarea unui server din Europa (Irlanda) pentru a obține o viteză mai bună.



MongoDB Deployments				
Development and Utility <small>Single-node deployments intended for environments that do not require high availability.</small>				
DEPLOYMENT	PLAN TYPE	RAM	SIZE	SIZE ON DISK
ds016108/envtest	Sandbox	shared	0.00 KB	0.00 KB
ds255309/iot	Sandbox	shared	126.75 KB	16.00 MB

Figura 4.1 MongoDB Deployments

Planul de costuri este momentan gratuit, în momentul când aplicația va trece din stadiul de dezvoltare în producție se va opta pentru servicii cu plată cu noduri secundare de replicare pentru a asigura performanțe mai bune și scalabilitate pe viitor.

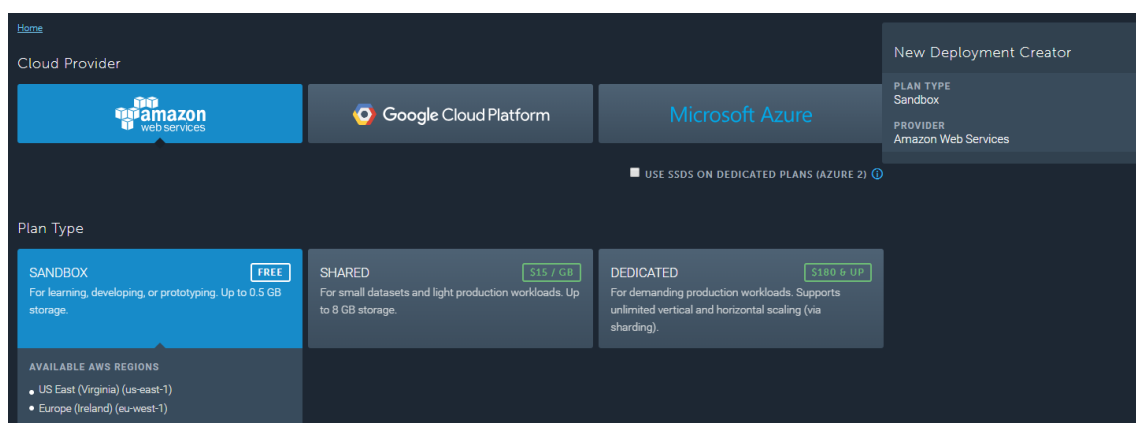


Figura 4.2 Selectarea furnizorului și a planului

Tot ce mai rămâne de făcut este un utilizator care va avea acces la baza de date. Utilizatorul se adaugă în rubrica "Users" accesând butonul "Add database user" (figura 4.3).

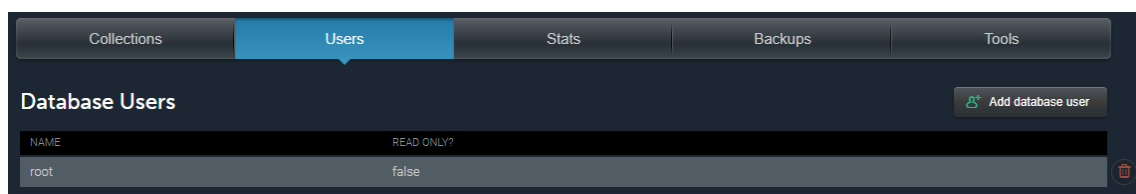


Figura 4.3 Pagina de creare a utilizatorului

Conectarea la baza de date se poate realiza prin 2 metode:

- mongo shell:

```
% mongo ds255309.mlab.com:55309/iot -u <dbuser> -p <dbpassword>
```

- MongoDB URI:

```
mongodb://<dbuser>:<dbpassword>@ds255309.mlab.com:55309/iot
```

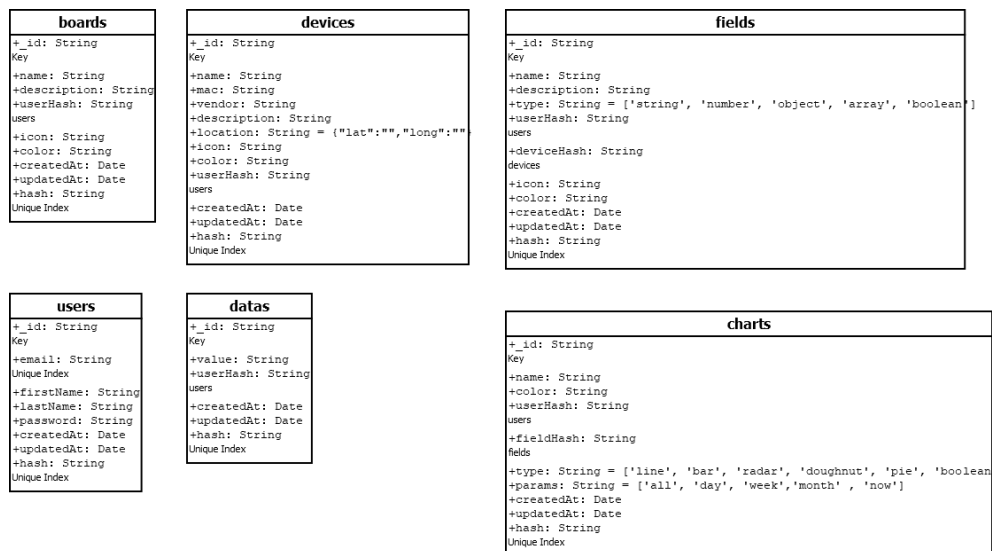


Figura 4.4 Schema bazei de date

#### 4.1.1.2. Robo 3T

Robo 3T (fostul Robomongo) este un GUI gratuit pentru entuziaștii care utilizează ca bază de date MongoDB în proiectele proprii. Aplicația este disponibilă pe platformele Windows, Linux, MacOS.

Instalarea aplicației se desfășoară ca oricare instalare a aplicației [23]. La fiecare pornire a aplicației ne întâmpină fereastra cu conexiunile active (figura 4.5). În aceeași fereastră se poate de adăugat/ editat/ șters/ clonat o conexiune .

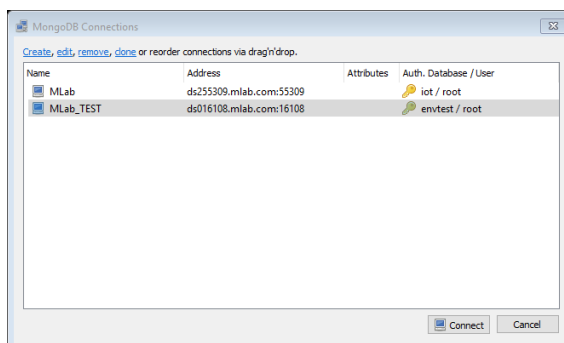


Figura 4.5 Fereastra conexiunilor active

Interfața aplicației este divizată în 3 secțiuni: bara de navigare, caseta pentru interogări și rezultatul interogării. Bara de navigare este reprezentată sub forma unui arbore, nodul primar fiind conexiunea → baza de date → Colecții / Funcții / Utilizatori.

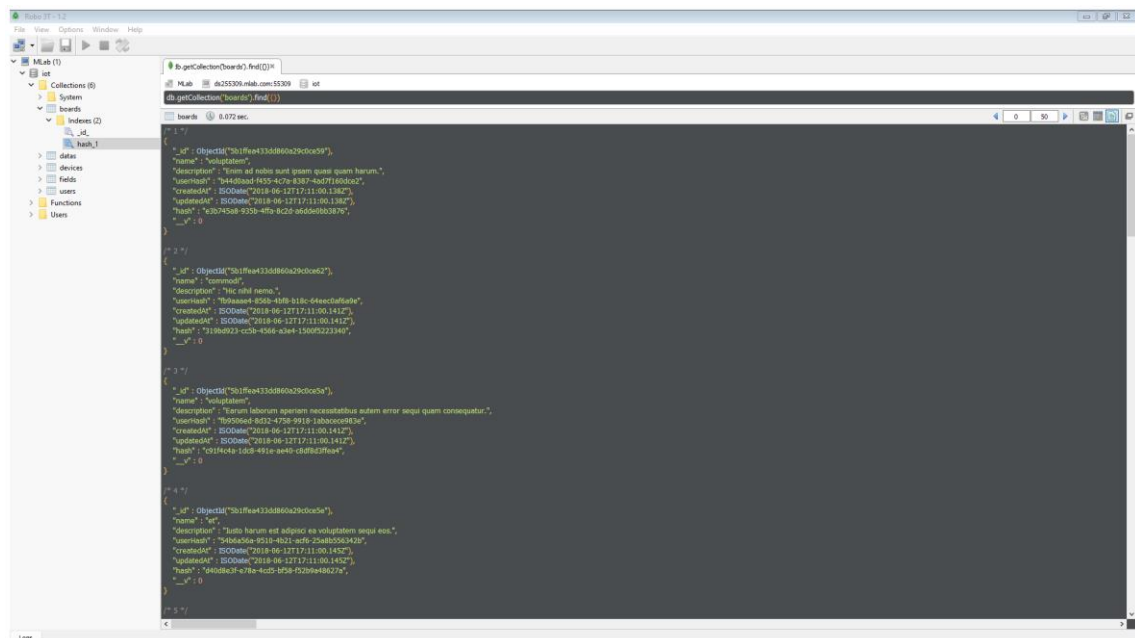


Figura 4.6 Vizualizarea unei colecții

#### 4.1.2. Conceptul aplicației

După cum a fost prezentat și în introducere aplicație reprezintă o platformă de achiziție a datelor de la dispozitive IoT și datorită tehnicilor de BI afișează grafice. Inițial după autentificare utilizatorul creează un tablou de bord, acesta este folosit pentru a grupa mai multe dispozitive în interfața căruia pot fi configurate grafice asociate câmpurilor dispozitivelor. După adăugarea tabloului de bord utilizatorul creează dispozitive căruia îi asociază câmpuri. Câmpul reprezintă un colector de date, în limbaj de programare este analog cu variabila în care se păstrează datele. După ce utilizatorul configurează platforma și obține hash-urile fiecărei componente poate configura dispozitivul pentru transmiterea datelor [22].

#### 4.1.3. Structura aplicației server

Pentru început, s-a instalat ultima versiune de Node.js 9.10.1 (la momentul prezentării aplicației versiunea de Node.js a fost actualizată la v10.3.0) cu ajutorul nvm, următoarea acțiune am creat directorul de lucru și am configurat managerul de pachete Yarn.

```
$ mkdir licenta
$ yarn init --yes
```

După instalarea managerului de pachete trebuie instalate dependențele necesare pentru crearea serverului.

#### 4.1.3.1. Structura directoarelor

Directorul "connectors" este predestinat conexiunilor cu bazele de date, în cazul de față fișierul "mongoose-connector.js" descrie conexiune cu baza de date MongoDB. În folderul "constants" fișierul "envs.js" sunt declarate mediile de dezvoltare utilizând librăria "keymirror".

Directorul "handlers" conține funcții asincrone de validare cum ar fi:

- "checkUser.js" verifică dacă utilizatorul dat are drepturi de acces
- "jwt.js" verifică dacă token-ul autorizării corespunde cu token-ul utilizatorului

Fișierul index.js importă funcțiile de mai sus în aplicație. Directorul "helpers" conține funcții ajutătoare, în cadrul proiectului am dezvoltat funcția AppError care permite captarea și afișarea erorilor Node.js într-un format mai accesibil.

Directorul "seeds" conține funcții care populează baza de date, ele sunt create pentru a popula baza de date din mediul de dezvoltare pentru a testa funcționalul API. Folderul "services" include serviciile utilizare în cadrul proiectului, în cazul dat proiectul utilizează librăria jsonwebtoken care permite generarea/validare token-urilor utilizate pentru autentificare utilizatorului.

Pentru a opera mai flexibil cu baza de date în dosarul "utils" se conțin funcții care permit realizarea conexiunii, ștergerii și închiderii conexiunii. Pe lângă acestea și o funcție care exportă mediul de dezvoltare setat la pornirea serverului.

Dosarul "config" conține fișiere json în care sunt specificați configurațiile mediilor aplicației. La pornirea serverului se accesează configurație specificată astfel obținem o configurare flexibilă și permite crearea de servere cu diferite opțiuni de start.

Exemplu: default.json

```
{
  "port" : 4000,
  "jwt": {
    "secret": "secret-key"
  }
}
```

Dosarul "node\_modules" conține toate modulele proiectului. Fișierului ".babelrc" este unul în care se specifică plugin-urile și preset-urile care se vor folosite de către

modulul babel. Babel este un compilator source-to-source care rescrie codul ECMAScript<sup>6</sup> în standardul anterior și compilarea în JS simplu.

Conținutul fișierului:

```
{
  "presets": [
    "env",
    "stage-0"
  ],
  "plugins": ["transform-async-to-generator"]
}
```

Instalarea modulului babel:

```
$ yarn add babel-core babel-cli babel-preset-env babel-preset-stage-0 babel-plugin-transform-async-to-generator --dev
```

Fișierul .env conține variabile globale ca linkul de acces al bazei de date:

```
MONGO_URI=mongodb://<user>:<password>@ds255309.mlab.com:55309/iot
```

Fișierul package.json conține dependențele proiectului scripturile de execuție ajutatoare în cadrul dezvoltării și parametrilor unor librării.

```
{
  "name": "API",
  "version": "1.0.0",
  "main": "index.js",
  "license": "MIT",
  "scripts": {
    "start": "babel-node app/server",
    "dev": "nodemon --exec 'yarn lint && yarn start'",
    "lint": "eslint app",
    "lint:fix": "eslint --fix app",
    "seed": "babel-node app/seeds",
    "test": "NODE_ENV=test jest app"
  },
  "jest": {
    "testEnvironment": "node",
    "browser": false
  },
  "devDependencies": {
    ...
  },
  "dependencies": {
    ...
  }
}
```

---

<sup>6</sup> ECMAScript – un standard specific utilizat pentru generarea fișierelor de scripturi

Comenzi implementate în cadrul proiectului.

```
"start": "babel-node app/server",
"dev": "nodemon --exec 'yarn lint && yarn start'",
"lint": "eslint app",
"lint:fix": "eslint --fix app",
"seed": "babel-node app/seeds",
"test": "NODE_ENV=test jest app"
```

- "start" pornește serverul.
- "dev" pornește serverul și la fiecare salvare a fișierelor modificate va recompila proiectul, pentru aceasta mai este nevoie de nodemon:

```
$ yarn add nodemon
```

- "test" rulează testele asociate fiecărui modul din cadrul aplicației.
- "seed" populează baza de date din mediul de teste cu seed-urile definite în cadrul aplicației
- "lint" va testa dacă codul corespunde stilului de formatare iar "lint:fix" va modifica automat codul pentru al aduce la un stil comun.

```
$ yarn add eslint eslint-config-google babel-eslint --dev
```

ESLint se folosește pentru a verifica codul scris și al aduce unui stil de formatare comun, în cadrul proiectului am optat pentru stilul de la Google. Ca și Babel ESLint are nevoie de un fișier de configurare.

Conținutul fișierului: .eslintrc.json

```
{
  "parser": "babel-eslint",
  "env": {
    "node": true,
    "es6": true,
    "jest": true
  },
  "parserOptions": {
    "sourceType": "module"
  },
  "globals": {
    "AppError": true
  },
  "extends": ["eslint:recommended", "google"],
  "rules": {
    "no-console": 0,
    "max-len": 0,
    "require-jsdoc": 0,
  }
}
```



```
"no-invalid-this": 0,
"comma-dangle": 1,
"object-curly-spacing": ["error", "always"]
}
```

Fișierul ”app.js” este unul din cele mai importante fișiere ale aplicației, în cadrul lui se importă restul fișierelor implicate ale serverului. Ca framework al aplicației s-a utilizat Koa. Pentru adăugarea în proiect este necesară comanda:

```
$ yarn add koa --save
```

Conținutul fișierului: app.js

```
import Koa from 'koa';
import connectorsInit from './connectors';
import initHandlers from './handlers';
import modules from './modules';
import AppError from './helpers/appError';

connectorsInit();
global.AppError = AppError;

const app = new Koa();

initHandlers(app);
app.use(modules);

app.use(async (ctx) => {
  ctx.body = '<h1>API start!</h1>';
});

export default app;
```

Fișierul ”config.js” determină mediul curent al aplicației și setează portul/adresa de acces a DB/ și cheia secretă pentru generarea token-urilor.

În fișierul ”server.js” se inițializează serverul cu portul specificat în configurațiile mediului.

#### 4.1.3.2. Structura modulelor

Funcțiile API realizate în cadrul serverului sunt divizate în module distincte și se găsesc în dosarul ”modules”. În general fiecare modul are o structură stabilă.

Dosarul ”\_tests\_” conține funcții de testare a modelului. Testele sunt realizate cu ajutorul librăriei supertest. La fiecare modul se testează helpers, services și rutele. Fișierele de test au extensia .spec.js. Crearea testelor pentru fiecare componentă oferă

posibilitatea de a testa tot funcționalul aplicației printr-o singură comandă astfel se poate determina cu mult mai ușor problemele apărute pe parcursul dezvoltării aplicației.

Dosarul "constants" conține constantele necesare modulului, în cadrul proiectului au fost definite constante pentru realizarea paginării.

```
export const MAX_SIZE = 20; //numărul maxim de componente în pagină
export const PAGE = 1; //pagina inițială
```

Directorul "handlers" conține funcții asincrone de validare, în cadrul modulului date se validează dacă hash-ul board-ului există în baza de date.

Folderul "helpers" conține funcții ajutătoare modulului, în cazul dat este implementată o funcție care generează filtrul de căutare pe baza interogărilor.

În dosarul "models" se află schema colecției MongoDB. Directorul "services" conține funcții asincrone care realizează logica modulului. Ele sunt apelate în controller.

Directorul "controllers" conține și el la fel funcții asincrone care se apelează în momentul accesării rutelor definite. Fișierul "index.js" din cadrul modulului definește rutele existente ale modulului, care apoi sunt incluse în modulul principal.

De exemplu pot fi definire rute generice pentru tabelul de bord, cum ar fi ruta pentru adăugare de conținut prin ruta *post* sau ștergerea conținutului prin ruta *delete*.

```
const router = new Router({ prefix: '/boards' });
router
  .post('/', checkUser(), boardController.create)
  .get('/', boardController.searchBoards)
  .param('hash', checkBoard())
  .put('/:hash', checkUser(), boardController.update)
  .delete('/:hash', checkUser(), boardController.delete)
  .get('/:hash', checkUser(), boardController.getBoard);
```

Lista modulelor dezvoltate:

- auth - răspunde de autentificare și înregistrarea utilizatorilor
- boards - gestionează tablourile de bord
- devices - gestionează dispozitivele
- fields - gestionează câmpurile
- data - gestionează datele transmise de dispozitivul IoT
- charts - gestionează graficele incluse în tabloul de bord
- user - gestionează utilizatorul

Pentru a fi sigur că logica modulelor este corectă fiecare modul are scrise teste de validare.

În codul următor este prezentat un exemplu de testarea a componentei board-service. Acest test face o comparație a datelor din baza de date și a celor introduse prin componentele bordului.

```
import pick from 'lodash/pick';
import {
  connect,
  dropDb,
  close,
} from '../utils/mongo';
import { BoardService } from '../services'; //importarea componentei testate
import AppError from '../helpers/appError'; //import helper global

global.AppError = AppError; //declararea AppError

describe('Board Service', () => { //titlul testului
  beforeEach(async () => { //execuția funcțiilor înaintea fiecărui test
    await connect(); // conectarea cu BD
    await dropDb(); // ștergerea BD
  });
  afterEach(async () => {
    await dropDb(); // ștergerea BD
    await close(); // închiderea conexiunii cu DB
  });

  it('create board as expected', async () => { //test
    const boardData = { // declararea unui board
      userHash: 'user-hash',
      name: 'name',
      description: 'description',
    };
    // apelarea funcției din componenta testată
    const boardModel = await BoardService.createBoard(boardData);
    // convertirea răspunsului primit in JSON pentru a putea fi comparat
    const board = boardModel.toObject();

    // testează dacă datele inițiale corespund cu cele primite
    expect(pick(board, Object.keys(boardData))).toEqual(boardData);
    expect(board).toHaveProperty('hash');
    expect(board).toHaveProperty('updatedAt');
    expect(board).toHaveProperty('createdAt');

    await dropDb();
  });
});
```

Pentru a rula testele scrise în terminal se execută comanda: `yarn test` care execută fiecare test din dosarul `_tests_` al modulului. În imaginea de mai jos este prezentat rezultatul testelor asupra modulul boards.

```
igor@DESKTOP-IBR0ACF:/mnt/d/licenta/iot/api$ yarn test
yarn run v1.7.0
$ NODE_ENV=test jest app
PASS app/modules/boards/__tests__/helpers/parseQueryFromSearch.spec.js
PASS app/modules/boards/__tests__/services/board-service.spec.js
  ● Console

    console.log app/connectors/mongoose-connector.js:13
      Mongo connected

PASS app/modules/boards/__tests__/route.spec.js (7.765s)
  ● Console

    console.log app/server.js:7
      Server running on port: 4000
    console.log app/connectors/mongoose-connector.js:13
      Mongo connected

Test Suites: 3 passed, 3 total
Tests:      9 passed, 9 total
Snapshots:  0 total
Time:       8.201s
Ran all test suites matching /app/i.
Done in 9.69s.
```

Figura 4.7 Rezultatul execuției comenzii `yarn test`

În figura 4.7 se poate observa că aplicația funcționează corect, testele efectuate având o rată de succes de 100%.

#### 4.1.4. Utilizarea instrumentului Postman

”Dezvoltarea aplicațiilor API este grea, Postman o face ușoară”<sup>7</sup>

Când vezi descrierea instrumentelor Postman – ai senzația că ai putere absolută asupra aplicației API, că nimic nu-ți poate sta în cale. Scopul principal al aplicației este de a crea colecții cu interogări către API-ul dorit. Orice dezvoltator sau teste, care va deschide o colecție va înțelege cum funcționează serviciul. Dezvoltarea serverului și clientului pot începe concomitent iar testerul poate scrie teste și le poate automatiza direct în Postman [24].

Conceptele pe care se bazează Postman sunt: Colecțiile (1) și interogări (2), orice lucru începe cu o colecție și se rezumă la descrierea API cu ajutorul interogărilor (figura 4.8).

Colecția este punctul de plecare pentru un API nou. Colecția poate fi privită ca un fișier nou al proiectului. O colecție include în sine mai multe interogări. De obicei API

---

<sup>7</sup> Citat al Postdot Technologies, Inc

este descris într-o singură colecție dar dacă nu se dorește acest lucru, nu există restricții. Colecția poate avea propriile scripturi și variabile.

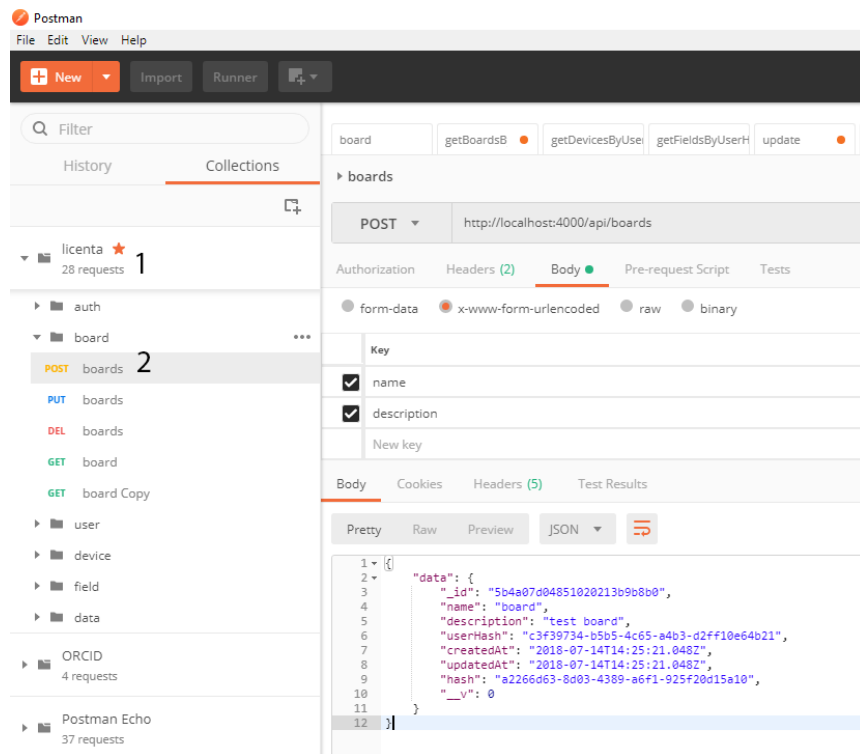


Figura 4.8 Interfața aplicației

Dosarele se utilizează pentru gruparea interogărilor în interiorul colecțiilor. De exemplu putem crea un dosar pentru prima versiune de API. În director pot fi grupate alte directoare, alegerea fiind sută la sută la dispoziția dezvoltatorului.

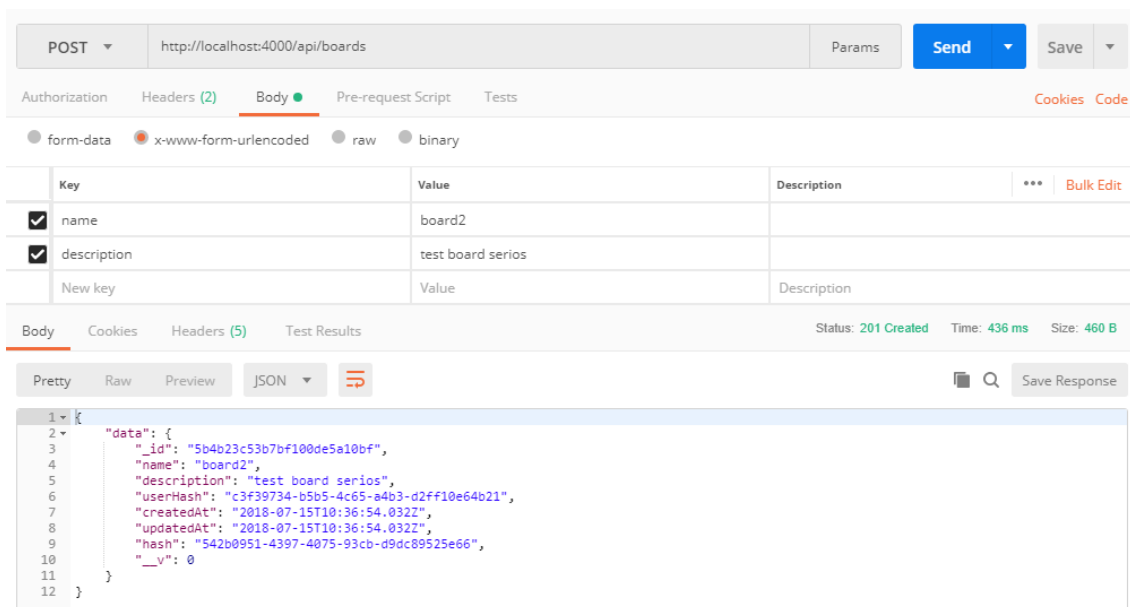


Figura 4.9 Fereastra interogării

Interogarea reprezintă componenta principală a colecției. Generatorul de interogări reprezintă cel mai important spațiu de lucru. Postmen poate executa interogări folosind toate metodele standardului HTTP. Prin 2 click-uri se pot adăuga parametri în header, cookie sau informație. Interogarea mai are și tab-uri suplimentare "Pre-request Script" și "Tests" care permit adăugarea scripturilor înainte și după executare interogării.

## 4.2. Aplicații Client (front-end)

### 4.2.1. Framework-ul Nuxt.js

Nuxt.js este un framework pentru aplicații universale bazate pe Vue.js. Principiul de bază al acestui framework constă în realizarea front-end-ului aplicației abstractizat față de arhitectura back-end-ului. Fiind un framework foarte flexibil poate fi utilizat ca baza proiectului sau ca o parte a unui proiect deja existent. Nuxt.js deține toate configurațiile necesare de a face dezvoltarea aplicațiilor cu randare pe server pentru Vue.js ușoară și plăcută [25].

Nuxt.js oferă și "nuxt generate" cu ajutorul căruia se pot genera aplicații în mod static pe Vue.js, această opțiune poate fi considerată un nou pas în dezvoltarea aplicațiilor web de tip micro-servicii. Ca și oricare alt framework Nuxt.js oferă multe posibilități de dezvoltare a aplicațiilor cum ar fi: date asincrone, middleware, șabloane și altele.

#### 4.2.1.1. Principiul de funcționare

Nuxt.js utilizează următoarele elemente pentru crearea aplicațiilor web:

- Vue 2
- Vue-Router
- Vuex
- Vue-Meta

Dimensiunea acestui pachet este de 28kb min+gzip (31kb dacă se utilizează vuex), Sub capota acestui monstru se află Webpack împreună cu vue-loader și babel-loader pentru unificarea, divizarea și minimizarea codului.

Avantajele utilizării nuxt.js:

- Fișiere Vue
- Separarea automată a codului

- Randare pe server
- Un sistem de rutare puternic cu date asincrone
- Fișiere statice
- Compilator source-to-source ES6/ES7
- Împachetarea și minimizarea JS/CSS
- Gestionarea elementelor din <header>
- Hot Module Replacement
- ESLint integrat
- Preprocesoare: SASS, LESS, Stylus ...

Instalarea Nuxt.js se face cu ajutorul comenzilor:

```
$ yarn global add vue-cli
$ vue init nux/starter
$ yarn install
```

#### 4.2.1.2. Structura dosarelor

- assets - conține fișiere cu cod sursă necompilat ca LESS, SASS
- components – conține componente Vue.js
- layouts – conține șabloanele aplicației
- middleware – conține funcții de validare
- pages – conține vederi și rute, framework-ul citește toate fișierele .vue din dosarul dat și creează rutele aplicației
- plugins – conține plugin-uri JavaScript se apelează înaintea creării componentului de bază Vue.js
- static – conține fișiere statice, accesibile prin url ”/”
- store – poate conține fișiere Vuex, este opțională
- Fișierul nuxt.config.js conține configurațiile aplicației
- Fișierul package.json conține dependențele și comenzile aplicației:

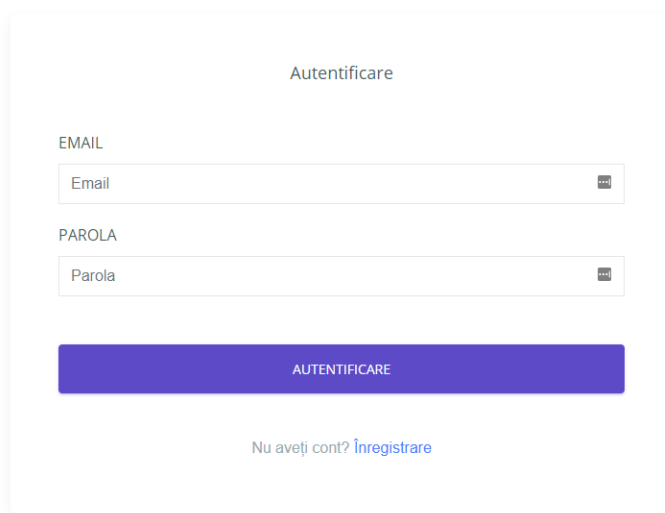
```
"scripts": {
  "dev": "nuxt",
  "build": "nuxt build",
  "start": "nuxt start",
  "generate": "nuxt generate"
}
```

Lista comenzilor disponibile:

Comanda	Descriere
<b>nuxt</b>	Pornește serverul localhost:3000
<b>nuxt build</b>	Împachetează aplicația cu webpack
<b>nuxt start</b>	Pornește serverul în mediul de producție
<b>nuxt generate</b>	Împachetează aplicația în fișiere HTML

### 4.3. Interfața grafică

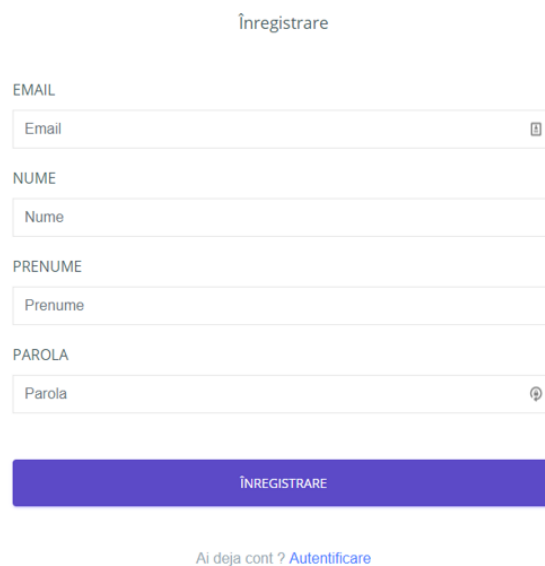
La accesarea aplicației vizitatorul este întâmpinat de formularul de autentificare (figura 4.10).



Formularul de autentificare este prezentat într-un cadru alb cu un titlu centralizat "Autentificare". Are două câmpuri de text: "EMAIL" cu placeholder "Email" și "PAROLA" cu placeholder "Parola". Ambele câmpuri au un iconiță de ochi pentru a comuta vizibilitatea conținutului. Sub câmpuri este un buton albastru cu textul "AUTENTIFICARE". În partea de jos a formularului se află textul "Nu aveți cont? [Înregistrare](#)".

Figura 4.10 Formularul de autentificare

Dacă utilizatorul nu are cont pe platformă trebuie să se înregistreze prin completarea formularului de înregistrare din figura 4.11.



Formularul de înregistrare este prezentat într-un cadru alb cu un titlu centralizat "Înregistrare". Are patru câmpuri de text: "EMAIL" cu placeholder "Email", "NUME" cu placeholder "Nume", "PRENUME" cu placeholder "Prenume" și "PAROLA" cu placeholder "Parola". Câmpurile "EMAIL" și "PAROLA" au iconițe de validare. Sub câmpuri este un buton albastru cu textul "ÎNREGISTRARE". În partea de jos a formularului se află textul "Ai deja cont? [Autentificare](#)".

Figura 4.11 Formularul de înregistrare



După autentificarea pe platformă utilizatorul este redirecționat către pagina cu tablourile de bord care grupează dispozitivele, după cum se vede în figura 4.12.

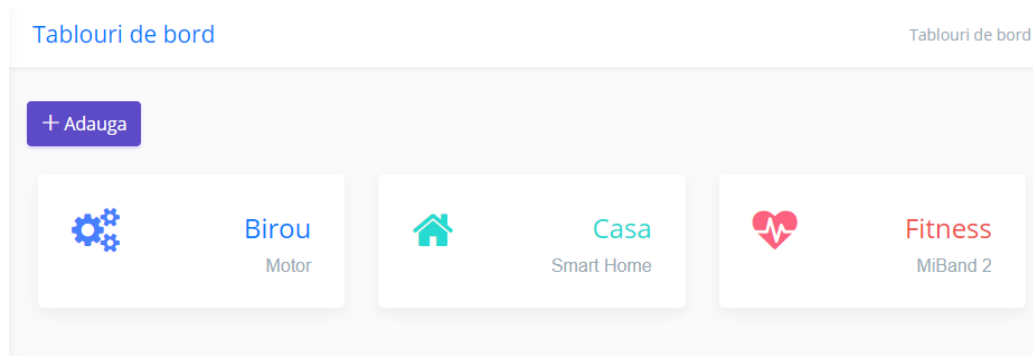


Figura 4.12 Tablouri de bord

În pagina dată sunt reprezentate pe blocuri tablourile de bord, pictograma și culoarea blocurilor sunt preluate din baza de date. Butonul ”Adaugă” creează un tablou de bord nou.

Figura 4.13 Crearea tablouri de bord

Dând click pe denumirea bordului utilizatorul este redirecționat către pagina bordului respectiv. În pagina dată sunt afișate datele tabloului de bord, numărul de dispozitive asociate și graficele generate pe baza câmpurilor dispozitivelor. Un tablou de bord poate fi editat, șters și se pot adăuga grafice (figura 4.14).

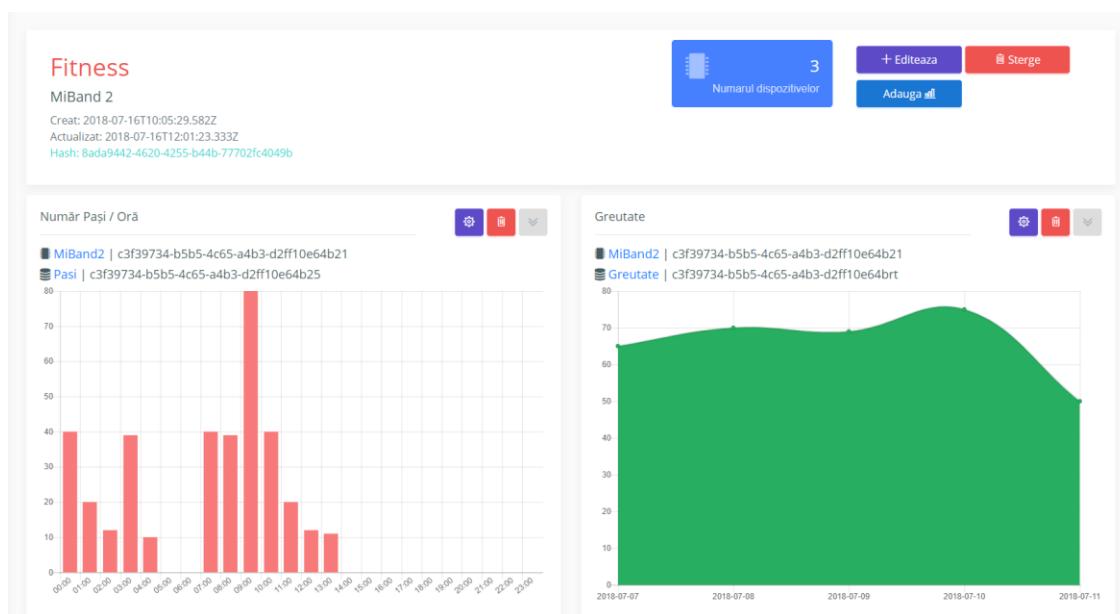


Figura 4.14 Graficele tabloului de bord

Pentru a adăuga un grafic în tabloul de bord curent se accesează butonul ”Adaugă” care deschide formularul de adăugare reprezentat în figura de mai jos. Pe lângă acțiunea de adăugare graficul mai poate fi ascuns, șters și editat.

Adaugă grafic

Denumire  
Fitness

Descriere  
MiBand 2

Câmpul  
MiBand2 | Pași

Tip  
Linie

Parametru  
Zi

Submit Cancel

Figura 4.15 Crearea unui grafic

Tipurile de grafice din aplicație:

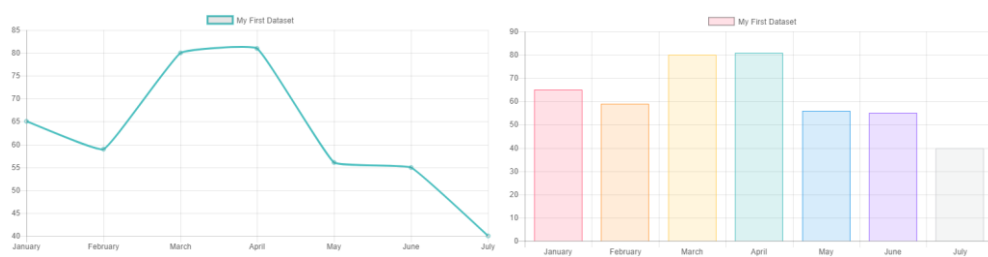


Figura 4.16 Tip linie

Figura 4.17 Tip bare

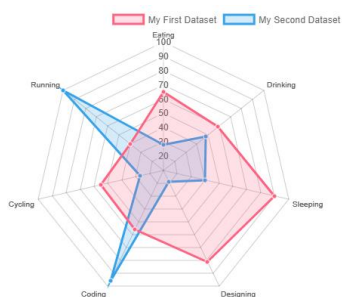


Figura 4.17 Tip radar

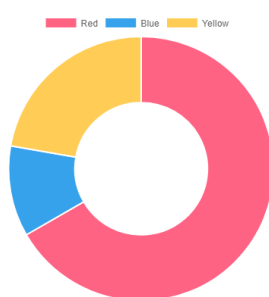


Figura 4.18 Tip covrig

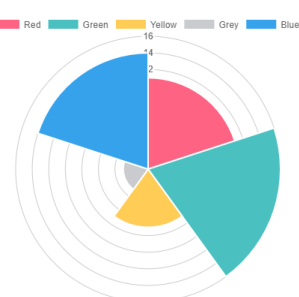


Figura 4.19 Tip polar area

Pagina "Dispozitive" afișează dispozitivele utilizatorului în blocuri separate cu Pictograma și culoarea setată de utilizator. Butonul "Adaugă" creează un dispozitiv nou.

## Dispozitive

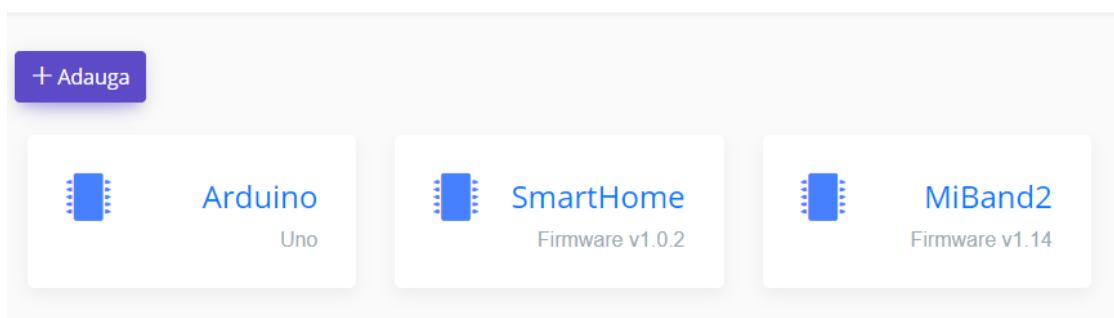


Figura 4.20 Lista dispozitivelor

La identificarea acțiunii click pe denumirea dispozitivului utilizatorul este redirectionat pe pagina dispozitivului (figura 4.20). În pagina dată este prezentat detaliat informația dispozitivului, harta, numărul de câmpuri asociate acțiunile de "Editare" și "Ștergere". Iar în parte inferioară câmpurile asociate dispozitivului dat.

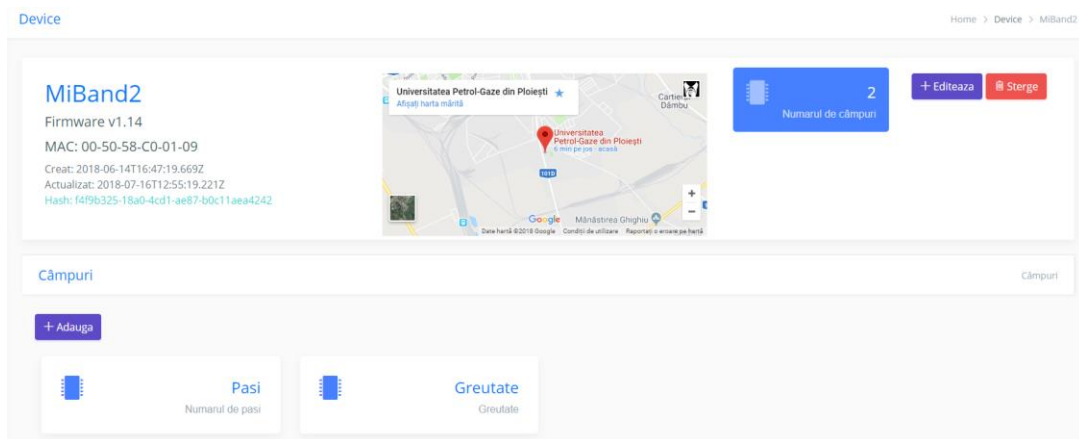


Figura 4.21 Pagina unui dispozitiv

Pentru a vizualiza câmpurile dispozitivelor există 2 metode, din pagina dispozitivului dând click pe denumirea câmpului sau din pagina de afișare a tuturor câmpurilor utilizatorului. Pagina de afișare a unui câmp conține informația sa generală și datele transmise de dispozitivul IoT, datele sunt grupate într-o tabelă care poate fi exportată în format pdf, excel sau printată. În tabelă se poate căuta o anumită valoare, se poate selecta numărul de rezultate afișate și poate fi aplicat un filtru de sortare pe coloane (figura 4.22).

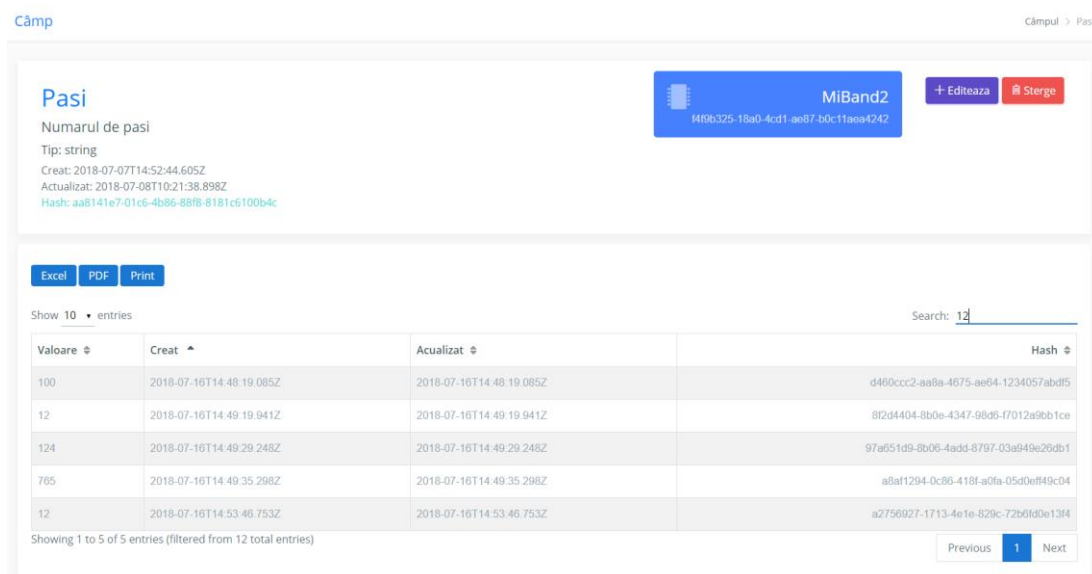


Figura 4.22 Pagina unui câmp

Dacă utilizatorul dorește să-și modifice datele trebuie să de click in colțul dreapta sus pe imagine și apoi pe "Profil" (figura 4.23).

The screenshot shows a web application interface for editing a user profile. At the top left, there is a navigation bar with the word "Utilizator" in blue. At the top right, there is a search icon and a user profile icon. Below the user profile icon, there is a dropdown menu with two options: "Profil" (with a person icon) and "Iesire" (with a power icon). The main content area is titled "Editează profilul" (Edit profile). It contains several input fields: "Hash" with the value "c3f39734-b5b5-4c65-a4b3-d2ff10e64b21", "Email" with the value "igor@gmail.com", "Nume" (Name) with the value "Igor", "Prenume" (First Name) with the value "Voloc", and "Parola" (Password) with the value "Parola". Each of the last three fields has a small icon on the right side. At the bottom of the form, there is a purple button labeled "Salvează" (Save).

*Figura 4.23 Pagina de profil*

Aplicația utilizează tehnici BI pentru gestiunea volumelor mari de date. În cadrul aplicației sau efectuat teste cu 1000 de înregistrări, timpul de execuție a funcției API care returnează înregistrările fiind de 655 ms. Efectuarea operațiilor de căutarea și sortarea în cele 1000 de înregistrări este instantă.

## Concluzii

La momentul dat internetul lucrurilor se află într-o continuă dezvoltare și pe zi ce trece oferă tot mai multe posibilități de interacționare cu mediul înconjurător. În ultimii ani au apărut tot mai multe soluții care realizează interconectarea dispozitivelor IoT, una din ele fiind platformele IoT.

Platforma dezvoltată nu reprezintă un concurent direct al gigantilor din domeniu ca Amazon Web Services (AWS) IoT, Microsoft Azure IoT suite, IBM Watson IoT, deoarece nu are un întreg ecosistem în spate dar realizează principalele funcționalități:

- culegerea și pregătirea datelor.
- conectivitate, prin intermediul API
- monitorizarea dispozitivelor
- securitatea implementată la autentificare, autorizare și securității datelor
- procesarea și analiza datelor cu elemente BI

În cadrul lucrării a fost efectuat un studiu amplu asupra domeniului IoT și Industria Bussines Inteligence, în urma căruia sa demonstrat că între aceste 2 domenii există și se va dezvolta în continuare o relație strânsă.

Au fost identificate și analizate tehnologiile software necesare pentru dezvoltarea domeniului IoT. Drept rezultat aceste tehnologii au fost utilizate în realizarea platformei.

Din punct de vedere tehnologic platforma rezultată este un suport foarte bun pentru a începe studiul domeniului IoT dar poate fi utilizată și în aplicații mai serioase. Platforma este „tânără” și se află abia la început de drum, dar în viitor planific dezvoltarea ei prin publicarea codului sursă pe GitHub ceea ce va permite oricărui dezvoltator să se folosească și să dezvolte funcționalul ei. O altă metodă de dezvoltare ar fi prin crearea unei echipe care se va ocupa de îmbunătățirea platformei în continuare. În urma modificărilor efectuate de dezvoltatori și colaboratori se poate realiza o soluție premium pe lângă cea standard care să ofere modalități de integrare a dispozitivelor IoT contracost.

## Bibliografie

1. B. Syed, „Beginning Node.js 1st ed. Edition”, Ed. Apress, 2014
2. C. Howson, „Successful Business Intelligence, Second Edition: Unlock the Value of BI & Big Data”, Ed. McGraw-Hill Education, 2014
3. C. Pfister, „Getting Started with the Internet of Things”, Ed. OREILLY, 2011
4. E. Brynjolfsson, „The Second Machine Age – Work, Progress, and Prosperity in a Time of Brilliant Technologies”, Ed. W.W. Norton & Company, 2014
5. G.Radulescu, C.Rosca, „Sisteme de operare”, Ed. UPG, Ploiesti 2015
6. K. Banker, „MongoDB in Action: Covers MongoDB version 3.0”, Ediția 2, Manning Publications Co., 2016
7. L. T. Moss, „Business Intelligence Roadmap: The Complete Project Lifecycle for Decision-Support Applications”, Ediția 2, Ed. Pearson Education, 2003
8. N. Paraschiv, „Achizitia si prelucrarea datelor”, Editura Universitatii din Ploiesti, Ploiesti, 2013
9. O.Cangea, „Transmisia si criptarea datelor”, Ed. M atrixRom, Bucuresti, 2008
10. P. Membrey, „MongoDB Basics” Ediția 2, Ed. Apress, 2014
11. P. Waher, „Learning Internet of Things”, Ed. Packt Publishing, 2015
12. R. Sherman, „Business Intelligence Guidebook: From Data Integration to Analytics”, Ed. Elsevier, 2015
13. R. Stackowiak, „Big Data and The Internet of Things”, Ed. Apress, 2015
14. S. Chacon, „Pro Git”, Ediția 2, Ed. Apress, 2014
15. S. Powers, „Learning Node: Moving to the Server-Side”, Ediția 2, Ed. OREILLY, 2016
16. \*\*\* [https://en.wikipedia.org/wiki/Internet\\_of\\_things](https://en.wikipedia.org/wiki/Internet_of_things)
17. \*\*\* <https://docs.microsoft.com/en-us/windows/wsl/install-win10>
18. \*\*\* <https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-18-04>
19. \*\*\* <https://www.atlassian.com/git/tutorials/what-is-version-control>
20. \*\*\* <https://yarnpkg.com/en/docs>
21. \*\*\* <https://medium.com/of-all-things-tech-progress/5-steps-to-build-a-rest-api-in-node-js-with-mongodb-e1f2113a39bd>
22. \*\*\* <https://github.com/vkhotey/summaries-api>
23. \*\*\* <https://robomongo.org/>
24. \*\*\* <https://www.getpostman.com/>
25. \*\*\* <https://nuxtjs.org/>

## Anexe - Documentația detaliată API

### Anexa1- Autentificare

Titlu	Înregistrarea utilizatorului
<b>URL</b>	<b>POST</b> /auth/signup
<b>Parametri URL</b>	
<b>Headers</b>	
<b>Data</b>	<pre>{   email : [string],   password : [string],   firstName : [string],   lastName : [string] }</pre>
<b>Răspunsul la succes</b>	Code: 201 Created <pre>{   "data": {     "_id": "5b49e4a34851020213b9b8a5",     "email": " email@domain.com",     "firstName": " FirstName ",     "lastName": " LastName ",     "hash": "dcf0aa2d-2e56-46be-a67c-bc2327b8fe50"   } }</pre>
<b>Răspunsul la eroare</b>	Code: 400 Bad Request <pre>{   "errors": {     ...   } }</pre>

Titlu	Autentificare utilizatorului
<b>URL</b>	<b>POST</b> /auth/signin
<b>Parametri URL</b>	
<b>Headers</b>	
<b>Data</b>	<pre>{   email : [string],   password : [string] }</pre>
<b>Răspunsul la succes</b>	Code: 200 OK <pre>{   "data":   "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Imlnb3JAZ21haWwuY29tliwiaWF0IjoxNTMxNTY5NzM0fQ.cOTkhLSVTshJDDFsg8-ILM_n M6jm7t4R92ie2Id3NWE" }</pre>
<b>Răspunsul la eroare</b>	Code: 400 Bad Request <pre>{</pre>



	<pre>"status": 400, "name": "BadRequestError", "message": "Parola este greșită!" }</pre>
--	------------------------------------------------------------------------------------------

Titlu	Datele utilizatorului curent
<b>URL</b>	<b>GET</b> /auth/ user
<b>Parametri URL</b>	
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	
<b>Răspunsul la succes</b>	Code: 200 OK <pre>{   "data": {     "_id": "5b49e4a34851020213b9b8a5",     "email": " email@domain.com",     "firstName": " FirstName ",     "lastName": " LastName ",     "hash": "dcf0aa2d-2e56-46be-a67c-bc2327b8fe50"   } }</pre>
<b>Răspunsul la eroare</b>	Code: 401 Unauthorized <pre>{   "status": 401,   "name": "UnauthorizedError",   "message": "Neautorizat. Token invalid!" }</pre>

#### Anexa 2 – Tablou de bord

Titlu	Creare tablou de bord
<b>URL</b>	<b>POST</b> /boards
<b>Parametri URL</b>	
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	<pre>{   name: [string],   description: [string],   icon: [string],   color: [string] }</pre>
<b>Răspunsul la succes</b>	Code: 201 Created <pre>{   "data": {     "_id": "5b49ecba4851020213b9b8aa",     "name": "board",     "description": "test board",     "icon": "fa-calc",     "color": "#ff0000",     "userHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b21",     "createdAt": "2018-07-14T12:29:46.784Z",     "updatedAt": "2018-07-14T12:29:46.784Z",   } }</pre>

	<pre> "hash": "c36a6f9f-f5cb-43c8-9507-6f10f849d27d", "__v": 0 } } </pre>
<b>Răspunsul la eroare</b>	<pre> Code: 400 Bad Request {   "errors": {     ...   } } </pre>

Actualizare tablou de bord	
<b>Titlu</b>	
<b>URL</b>	<b>PUT</b> /boards/:hash
<b>Parametri URL</b>	hash = [string]
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	<pre> {   name: [string],   description: [string],   icon: [string],   color: [string] } </pre>
<b>Răspunsul la succes</b>	<pre> Code: 200 OK {   "data": {     "_id": "5b49eeb94851020213b9b8ac",     "name": "board",     "description": "description board",     "icon": "fa-calc",     "color": "#ff0000",     "userHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b21",     "createdAt": "2018-07-14T12:38:17.605Z",     "updatedAt": "2018-07-14T12:38:28.289Z",     "hash": "c84abbe2-748c-48b0-89ae-18c5e30be634"   } } </pre>
<b>Răspunsul la eroare</b>	<pre> Code: 404 Not Found {   "status": 404,   "message": "Tabloul de bord cu hash-ul dat :hash nu a fost găsit" } </pre>

Șterge tablou de bord	
<b>Titlu</b>	
<b>URL</b>	<b>DELETE</b> /boards/:hash
<b>Parametri URL</b>	hash = [string]
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	
<b>Răspunsul la succes</b>	Code: 200 OK

	<pre>{   "data": {     "hash": "c84abbe2-748c-48b0-89ae-18c5e30be634"   } }</pre>
<b>Răspunsul la eroare</b>	Code: 404 Not Found <pre>{   "status": 404,   "message": "Tabloul de bord cu hash-ul dat :hash nu a fost găsit" }</pre>

Titlu	Datele board
<b>URL</b>	<a href="#">GET</a> /boards/:hash sau /boards?name=:name&description=:description
<b>Parametri URL</b>	Obigatoriu: hash = [string] Opționali: name = [string] description = [string]
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	
<b>Răspunsul la succes</b>	Code: 200 OK <pre>{   "data": {     "name": "voluptatem",     "description": "Enim ad nobis sunt ipsam quasi quam harum.",     "icon": "fa-calc",     "color": "#ff0000",     "hash": "e3b745a8-935b-4ffa-8c2d-a6dde0bb3876",     "createdAt": "2018-06-12T17:11:00.138Z",     "updatedAt": "2018-06-12T17:11:00.138Z"   } }</pre> sau <pre>{   "filter": {     "name": "",     "description": "",     "size": 20,     "page": 1   },   "boards": [ ... ],   "count": 2,   "pages": 1,   "page": 1 }</pre>
<b>Răspunsul la eroare</b>	Code: 404 Not Found

	<pre>{   "status": 404,   "message": "Tabloul de bord cu hash-ul dat :hash nu a fost găsit" }</pre>
--	-----------------------------------------------------------------------------------------------------

### Anexa 3 - Utilizatori

Titlu	Lista tablourilor de bord care aparțin utilizatorului
URL	<b>GET</b> /users/:userHash/boards
Parametri URL	userHash = [string]
Headers	{ "Authorization" : [token] }
Data	
Răspunsul la succes	Code: 200 OK <pre>{   "data": [ ... ] }</pre>
Răspunsul la eroare	Code: 404 Not Found <pre>{   "status": 404,   "name": "NotFoundError",   "message": "Utilizatorul cu hash-ul dat :userHash nu a fost găsit" }</pre>

Titlu	Lista dispozitivelor care aparțin utilizatorului
URL	<b>GET</b> /users/:userHash/devices
Parametri URL	userHash = [string]
Headers	{ "Authorization" : [token] }
Data	
Răspunsul la succes	Code: 200 OK <pre>{   "data": [ ... ] }</pre>
Răspunsul la eroare	Code: 404 Not Found <pre>{   "status": 404,   "name": "NotFoundError",   "message": "Utilizatorul cu hash-ul dat :userHash nu a fost găsit" }</pre>

Titlu	Lista câmpurilor care aparțin utilizatorului
URL	<b>GET</b> /users/:userHash/fields
Parametri URL	userHash = [string]
Headers	{ "Authorization" : [token] }
Data	
Răspunsul la succes	Code: 200 OK

	<pre>{   "data": [     {       ...     },     {       ...     }   ] }</pre>
<b>Răspunsul la eroare</b>	Code: 404 Not Found <pre>{   "status": 404,   "name": "NotFoundError",   "message": "Utilizatorul cu hash-ul dat :userHash nu a fost găsit" }</pre>

Titlu Actualizarea datelor utilizatorului	
<b>URL</b>	<b>PUT</b> /users/:hash
<b>Parametri URL</b>	hash = [string]
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	<pre>{   email : [string],   password : [string],   firstName : [string],   lastName : [string] }</pre>
<b>Răspunsul la succes</b>	Code: 200 OK <pre>{   "data": {     "_id": "5b1fff181bed1d0a4f466ac0",     "email": "igor@gmail.com",     "password": "\$2b\$10\$XJnALFLpyDgi97sZ503pdO2IHsSUEcDgF0hsM 5lEpzTnu7PRWeOqC",     "firstName": "Voloc",     "lastName": "Igor",     "createdAt": "2018-06-12T17:12:56.307Z",     "updatedAt": "2018-07-06T07:48:36.612Z",     "hash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b21",     "__v": 0   } }</pre>
<b>Răspunsul la eroare</b>	Code: 404 Not Found <pre>{   "status": 404,   "name": "NotFoundError", }</pre>

	"message": " Utilizatorul cu hash-ul dat :hash nu a fost găsit "
	}

#### Anexa 4 - Dispozitive

Titlu	Creare dispozitiv
URL	<b>POST</b> /device
Parametri URL	
Headers	{ "Authorization" : [token] }
Data	{       name: [string],       mac: [string],       vendor: [string],       description: [string],       location: [string],       icon: [string],       color: [string],       boardHash: [string]     }
Răspunsul la succes	Code: 201 Created       {         "data": {           "_id": "5b49ecba4851020213b9b8aa",           "name": "board",           "mac": "00-E0-4C-64-F6-BB",           "vendor": "Arduino",           "description": "test board",           "location": "47.1491944,28.5913391,13",           "icon": "fa-calc",           "color": "#ff0000",           "userHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b21",           "boardHash": "c36a6f9f-f5cb-43c8-9507-6f10f9d27d ",           "createdAt": "2018-07-14T12:29:46.784Z",           "updatedAt": "2018-07-14T12:29:46.784Z",           "hash": "c36a6f9f-f5cb-43c8-9507-6f10f849d27d",           "__v": 0         }       }
Răspunsul la eroare	Code: 400 Bad Request       {         "errors": {           ...         }       }

Actualizare dispozitiv	
<b>Titlu</b>	
<b>URL</b>	<b>PUT</b> /device/:hash
<b>Parametri URL</b>	hash = [string]
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	{ name: [string], mac: [string], vendor: [string], description: [string], location: [string], icon: [string], color: [string], boardHash: [string] }
<b>Răspunsul la succes</b>	Code: 200 OK { "data": { "_id": "5b49ecba4851020213b9b8aa", "name": "board", "mac": "00-E0-4C-64-F6-BB", "vendor": "Arduino", "description": "test board", "location": "47.1491944,28.5913391,13", "icon": "fa-calc", "color": "#ff0000", "userHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b21", "boardHash": "c36f9f-f5cb-43c8-9507-6f10f849d27d ", "createdAt": "2018-07-14T12:29:46.784Z", "updatedAt": "2018-07-14T12:29:46.784Z", "hash": "c36a6f9f-f5cb-43c8-9507-6f10f849d27d", "__v": 0 } }
<b>Răspunsul la eroare</b>	Code: 404 Not Found { "status": 404, "message": " Dispozitivul cu hash-ul dat :hash nu a fost găsit " }

Șterge dispozitiv	
<b>Titlu</b>	
<b>URL</b>	<b>DELETE</b> /device/:hash
<b>Parametri URL</b>	hash = [string]
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	
<b>Răspunsul la succes</b>	Code: 200 OK { "data": {

	<pre>"hash": "c84abbe2-748c-48b0-89ae-18c5e30be634" }</pre>
<b>Răspunsul la eroare</b>	<pre>Code: 404 Not Found {   "status": 404,   "message": "Dispozitivul cu hash-ul dat :hash nu a fost găsit" }</pre>

Titlu	Datele dispozitivului
<b>URL</b>	<pre>GET /device/:hash sau /device?name=:name&amp;description=:description&amp;mac=:mac&amp; vendor=: vendor</pre>
<b>Parametri URL</b>	<p>Obigatoriu:</p> <pre>hash = [string]</pre> <p>Opțional:</p> <pre>name = [string] description = [string] mac = [string] vendor = [string] boardHash = [string]</pre>
<b>Headers</b>	<pre>{ "Authorization" : [token] }</pre>
<b>Data</b>	
<b>Răspunsul la succes</b>	<pre>Code: 200 OK {   "data": {     "name": "voluptatem",     "description": "Enim ad nobis sunt ipsam quasi quam harum.",     "icon": "fa-calc",     "color": "#ff0000",     "hash": "e3b745a8-935b-4ffa-8c2d-a6dde0bb3876",     "createdAt": "2018-06-12T17:11:00.138Z",     "updatedAt": "2018-06-12T17:11:00.138Z"   } } sau {   "filter": {     "name": "",     "description": "",     "mac": "",     "vendor": "",     "size": 20,     "page": 1   },   "devices": [ ... ],   "count": 2,</pre>



	<pre>"pages": 1, "page": 1 }</pre>
<b>Răspunsul la eroare</b>	<pre>Code: 404 Not Found {   "status": 404,   "message": " Dispozitivul cu hash-ul dat :hash nu a fost găsit " }</pre>

#### Anexa 5 - Câmpuri

Titlu	Creare câmp
<b>URL</b>	<b>POST</b> /field
<b>Parametri URL</b>	
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	<pre>{   name: [string],   description: [string],   type: [string],   icon: [string],   color: [string],   deviceHash: [string] }</pre>
<b>Răspunsul la succes</b>	<pre>Code: 201 Created {   "data": {     "_id": "5b49ecba4851020213b9b8aa",     "name": "board",     "type": "String",     "description": "test board",     "icon": "fa-calc",     "color": "#ff0000",     "userHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b21",     "deviceHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b24",     "createdAt": "2018-07-14T12:29:46.784Z",     "updatedAt": "2018-07-14T12:29:46.784Z",     "hash": "c36a6f9f-f5cb-43c8-9507-6f10f849d27d",     "__v": 0   } }</pre>
<b>Răspunsul la eroare</b>	<pre>Code: 400 Bad Request {   "errors": {     ...   } }</pre>

Titlu	Actualizare câmp
<b>URL</b>	<b>PUT</b> /field/:hash
<b>Parametri URL</b>	hash = [string]
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	{ name: [string], description: [string], icon: [string], color: [string], deviceHash: [string] }
<b>Răspunsul la succes</b>	Code: 200 OK { "data": { "_id": "5b49ecba4851020213b9b8aa", "name": "board", "type": "String", "description": "test board", "icon": "fa-calc", "color": "#ff0000", "userHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b21", "deviceHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b24", "createdAt": "2018-07-14T12:29:46.784Z", "updatedAt": "2018-07-14T12:29:46.784Z", "hash": "c36a6f9f-f5cb-43c8-9507-6f10f849d27d", "__v": 0 } }
<b>Răspunsul la eroare</b>	Code: 404 Not Found { "status": 404, "message": "Câmpul cu hash-ul dat :hash nu a fost găsit" }

Titlu	Șterge câmp
<b>URL</b>	<b>DELETE</b> /field/:hash
<b>Parametri URL</b>	hash = [string]
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	
<b>Răspunsul la succes</b>	Code: 200 OK { "data": { "hash": "c84abbe2-748c-48b0-89ae-18c5e30be634" } }
<b>Răspunsul la eroare</b>	Code: 404 Not Found { "status": 404, }

	"message": "Câmpul cu hash-ul dat :hash nu a fost găsit"
	}


Titlu	Datele câmpului
<b>URL</b>	<b>GET</b> /field/:hash sau /field?name=:name&description=:description&type=:type
<b>Parametri URL</b>	Obigatoriu: hash = [string] Opționali: name = [string] description = [string] type = [string]
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	
<b>Răspunsul la succes</b>	Code: 200 OK { "data": { "name": "voluptatem", "type": "String", "description": "Enim ad nobis sunt ipsam quasi quam harum.", "icon": "fa-calc", "color": "#ff0000", "hash": "e3b745a8-935b-4ffa-8c2d-a6dde0bb3876", "deviceHash ": "c3f39734-b5b5-4c65-a4b3- d2ff10e64b24", "createdAt": "2018-06-12T17:11:00.138Z", "updatedAt": "2018-06-12T17:11:00.138Z" } } sau { "filter": { "name": "", "description": "", "type": "", "size": 20, "page": 1 }, "fields": [ ... ], "count": 2, "pages": 1, "page": 1 } }
<b>Răspunsul la eroare</b>	Code: 404 Not Found { "status": 404, "message": "Câmpul cu hash-ul dat :hash nu a fost găsit" }


#### Anexa 6 – Date


Titlu	Transmitere date
<b>URL</b>	<b>POST</b> /data
<b>Parametri URL</b>	
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	{ name: [string], fieldHash: [string] }
<b>Răspunsul la succes</b>	Code: 201 Created { "data": { "_id": "5b49ecba4851020213b9b8aa", "value": "board", "userHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b21", "fieldHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b24", "createdAt": "2018-07-14T12:29:46.784Z", "updatedAt": "2018-07-14T12:29:46.784Z", "hash": "c36a6f9f-f5cb-43c8-9507-6f10f849d27d", "__v": 0 } }
<b>Răspunsul la eroare</b>	Code: 400 Bad Request { "errors": { ... } }

Titlu	Actualizare date
<b>URL</b>	<b>PUT</b> /data/:hash
<b>Parametri URL</b>	hash = [string]
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	{ name: [string], fieldHash: [string] }
<b>Răspunsul la succes</b>	Code: 200 OK { "data": { "_id": "5b49ecba4851020213b9b8aa", "value": "board", "userHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b21", "fieldHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b24", "createdAt": "2018-07-14T12:29:46.784Z", } }

	<pre>"updatedAt": "2018-07-14T12:29:46.784Z", "hash": "c36a6f9f-f5cb-43c8-9507-6f10f849d27d", "__v": 0 } }</pre>
<b>Răspunsul la eroare</b>	<pre>Code: 404 Not Found {   "status": 404,   "message": "Data cu hash-ul dat :hash nu a fost găsit" }</pre>

Șterge date	
<b>Titlu</b>	
<b>URL</b>	 /data/:hash
<b>Parametri URL</b>	hash = [string]
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	
<b>Răspunsul la succes</b>	<pre>Code: 200 OK {   "data": {     "hash": "c84abbe2-748c-48b0-89ae-18c5e30be634"   } }</pre>
<b>Răspunsul la eroare</b>	<pre>Code: 404 Not Found {   "status": 404,   "message": "Data cu hash-ul dat :hash nu a fost găsit" }</pre>

Șterge toate datele asociate câmpului specificat	
<b>Titlu</b>	
<b>URL</b>	 /data/:fieldHash/data
<b>Parametri URL</b>	fieldHash = [string]
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	
<b>Răspunsul la succes</b>	<pre>Code: 200 OK {   "data": [ ... ] }</pre>
<b>Răspunsul la eroare</b>	<pre>Code: 404 Not Found {   "status": 404,   "message": "Câmpul cu hash-ul dat :fieldHash nu a fost găsit" }</pre>

Datele câmpului	
<b>Titlu</b>	
<b>URL</b>	 /data/:hash sau /data?value=:value&fieldHash=:fieldHash

<b>Parametri URL</b>	Obligatziu: hash = [string] Opțional: value = [string] fieldHash = [string]
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	
<b>Răspunsul la succes</b>	Code: 200 OK <pre>{   "data": {     "_id": "5b49ecba4851020213b9b8aa",     "value": "board",     "userHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b21",     "fieldHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b24",     "createdAt": "2018-07-14T12:29:46.784Z",     "updatedAt": "2018-07-14T12:29:46.784Z",     "hash": "c36a6f9f-f5cb-43c8-9507-6f10f849d27d",     "__v": 0   } }sau {   "filter": {     "name": "",     "fieldHash": "",     "size": 20,     "page": 1   },   "data": [ ... ],   "count": 2,   "pages": 1,   "page": 1 }</pre>
<b>Răspunsul la eroare</b>	Code: 404 Not Found <pre>{   "status": 404,   "message": "Data cu hash-ul dat :hash nu a fost găsit" }</pre>

#### Anexa 7 - Grafice

Titlu	Creare grafic
<b>URL</b>	<b>POST</b> /chart
<b>Parametri URL</b>	
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	<pre>{   name: [string],   description: [string],   type: [string],   params: [string], }</pre>

	<pre> color: [string], fieldHash: [string] } </pre>
<b>Răspunsul la succes</b>	<pre> Code: 201 Created {   "data": {     "_id": "5b49ecba4851020213b9b8aa",     "name": "board",     "type": "String",     "description": "test board",     "params": "all ",     "color": "#ff0000",     "userHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b21",     "fieldHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b24",     "createdAt": "2018-07-14T12:29:46.784Z",     "updatedAt": "2018-07-14T12:29:46.784Z",     "hash": "c36a6f9f-f5cb-43c8-9507-6f10f849d27d",     "__v": 0   } } </pre>
<b>Răspunsul la eroare</b>	<pre> Code: 400 Bad Request {   "errors": {     ...   } } </pre>

Titlu	Actualizare grafic
<b>URL</b>	<b>PUT</b> /chart/:hash
<b>Parametri URL</b>	hash = [string]
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	<pre> {   name: [string],   description: [string],   type: [string],   params: [string],   color: [string],   fieldHash: [string] } </pre>
<b>Răspunsul la succes</b>	<pre> Code: 200 OK {   "data": {     "_id": "5b49ecba4851020213b9b8aa",     "name": "board",     "type": "String",     "description": "test board",     "params": "all", </pre>

	<pre> "color": "#ff0000", "userHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b21", "fieldHash": "c3f39734-b5b5-4c65-a4b3-d2ff10e64b24", "createdAt": "2018-07-14T12:29:46.784Z", "updatedAt": "2018-07-14T12:29:46.784Z", "hash": "c36a6f9f-f5cb-43c8-9507-6f10f849d27d", "__v": 0 } </pre>
<b>Răspunsul la eroare</b>	<pre> Code: 404 Not Found {   "status": 404,   "message": "Graficul cu hash-ul dat :hash nu a fost găsit" } </pre>

Titlu	Șterge grafic
<b>URL</b>	<b>DELETE</b> /chart/:hash
<b>Parametri URL</b>	hash = [string]
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	
<b>Răspunsul la succes</b>	<pre> Code: 200 OK {   "data": {     "hash": "c84abbe2-748c-48b0-89ae-18c5e30be634"   } } </pre>
<b>Răspunsul la eroare</b>	<pre> Code: 404 Not Found {   "status": 404,   "message": "Graficul cu hash-ul dat :hash nu a fost găsit" } </pre>

Titlu	Datele graficului
<b>URL</b>	<b>GET</b> /chart/:hash sau /chart?name=:name&description=:description&type=:type&params=:params
<b>Parametri URL</b>	Obligatori: <pre>hash = [string]</pre> Opționali: <pre> name = [string] description = [string] type = [string] params = [string] fieldHash = [string] </pre>
<b>Headers</b>	{ "Authorization" : [token] }
<b>Data</b>	
<b>Răspunsul la succes</b>	Code: 200 OK



	<pre> {   "data": {     "name": "voluptatem",     "type": "String",     "description": "Enim ad nobis sunt ipsam quasi quam harum.",     "params": "all",     "color": "#ff0000",     "hash": "e3b745a8-935b-4ffa-8c2d-a6dde0bb3876",     "fieldHash": "c3f39734-b5b5-4c65-a4b3- d2ff10e64b24",     "createdAt": "2018-06-12T17:11:00.138Z",     "updatedAt": "2018-06-12T17:11:00.138Z"   } } sau {   "filter": {     "name": "",     "description": "",     "type": "",     "params": "",     "size": 20,     "page": 1   },   "charts": [ ... ],   "count": 2,   "pages": 1,   "page": 1 } </pre>
<b>Răspunsul la eroare</b>	<pre> Code: 404 Not Found {   "status": 404,   "message": "Graficul cu hash-ul dat :hash nu a fost găsit" } </pre>