CUPRINS

[Capitolul 1. Introducere 4](#_Toc484963423)

[1.1 Context 4](#_Toc484963424)

[1.2 Domeniul temei 4](#_Toc484963425)

[1.3Descrierea temei 5](#_Toc484963426)

[1.4 Soluții asemănătoare 5](#_Toc484963427)

[Capitolul 2. Fundamentare teoretică 7](#_Toc484963428)

[2.1 Navigator web 7](#_Toc484963429)

[2.2 Tehnologii folosite 8](#_Toc484963430)

[2.3 NodeJS 9](#_Toc484963431)

[2.4 MongoDB 14](#_Toc484963432)

[2.5 Angular 2.0 && Typescript 18](#_Toc484963433)

[2.5.1 Typescript 21](#_Toc484963434)

[2.5.6 Expres.js 23](#_Toc484963435)

[Capitolul 3. Specificațiile de proiectare ale aplicației 25](#_Toc484963436)

[3.1 Accesul la informații legate de donare 25](#_Toc484963437)

[3.2 Accesul pe nivele în aplicație 25](#_Toc484963438)

[3.2.1 Administrator de județ 26](#_Toc484963439)

[3.2.2 Administrator de centru 26](#_Toc484963440)

[3.2.3 Angajat centru 26](#_Toc484963441)

[3.2.4 Donator 27](#_Toc484963442)

[3.2.5 Vizitator 27](#_Toc484963443)

[Capitolul 4. Implementarea aplicației 28](#_Toc484963444)

[4.1 Arhitectura aplicației 28](#_Toc484963445)

[4.2. Funcțiile aplicației 29](#_Toc484963446)

[4.3 Detalii de implementare 30](#_Toc484963447)

[Capitolul 5. Utilizare, rezultate experimentale 32](#_Toc484963448)

[5.1 Utilizarea aplicației 32](#_Toc484963449)

[5.2 Accesul in aplicație pe mai multe nivele 33](#_Toc484963450)

[5.2.1 Nivelul de acces la Administrator de județ 34](#_Toc484963451)

[5.2.2 Nivelul de Administrator de centru de donare 37](#_Toc484963452)

[5.2.3 Nivelul de Angajat la un centru de donare 39](#_Toc484963453)

[5.2.4 Nivelul de Utilizator 40](#_Toc484963454)

[5.2.5 Nivelul de Vizitator 42](#_Toc484963455)

[5.3 Testarea aplicației 42](#_Toc484963456)

[Capitolul 6. Concluzii și direcții de continuare a dezvoltării 44](#_Toc484963457)

[6.1 Concluzii 44](#_Toc484963458)

[6.2 Direcții de dezvoltare 45](#_Toc484963459)

[Capitolul 7. Bibliografie 46](#_Toc484963460)

# Capitolul 1. Introducere

## 1.1 Context

În ultimul deceniu tehnologia s-a dezvoltat foarte mult, în special cea electronică  oferind  publicului larg  posibilitatea de comunicare și colaborare mult mai ușoară și rapidă iar, distanța de această dată nu mai semnifică o problemă. In urma dezvoltării au apărut și criteriile,  precum portabilitatea, viteza de transfer sau serviciile oferite de un canal de comunicare fiind tot mai exigente din partea utilizatorilor.

Această evoluție a tehnologiei influențează din ce in ce mai mult viețile noastre de zi cu zi, la orice pas este vizibilă întâlnirea omului cu tehnologia și cum acesta intervine în ușurarea activităților noastre. Cu toate astea sunt situați când tehnologia nu este folosită la întreaga capacitate.

## 1.2 Domeniul temei

Din punct de vedere al încadrării temei acesta face parte din domeniul aplicațiilor web, având la baza browser-ul care oferă interacționarea cu aplicația în ceea ce privește domeniul destinație acesta este sănătatea.

Sunt dese momentele când lipsa acestei legături dintre pacient și tehnologie nu aduc nici un beneficiu, uneori având o consecință destul de negativă pacientului, adică există situați când prezența tehnologie intru-un moment cheie, în tratamentul unui pacient ar putea avea o influentă mia mult decât pozitivă, astfel am decis să contribui cu o idee spre crearea acestei legături destul de benefică dacă este valorificată.

Scopul acestei aplicației este acela de a diminua distanța dintre utilizatorul domeniul sănătății(pacientul) și sistemul de sănătate, mai exact in ceea ce privește donarea de sânge în Timișoara, punând la dispoziție mai mult funcționalități cu ajutorul cărora donarea de sânge in Timișoara să nu mai fie așa de greu de realizat.

## 1.3Descrierea temei

Evoluția interacțiunii dintre om și tehnologie este vizibilă la tot pasul din ce în ce mai mul, însă nu tot timpul este folosită la nivelul său maxim. Privind în jurul nostru putem constate că fiecare dintre cei care i vedem au un telefon inteligent și cel mai important au acces la internet.

Din acest motiv am decis ca aplicația web Timișoara Doneză să fie dezvoltată pentru browser deoarece reduce acțiunea utilizatotului, adică nu trebuie să descarce o aplicație și să o intaleze. Mereu munca  ce o investești într-un lucru, serviciu, sau ce o fi el este important ca ideile tale sa fie clare și înțelese de cine trebuie, toate astea combinate cu evoluția tehnologiei si accesul la internet care a devenit virală in ziua de azi, te pot ajuta să ai rezultatele dorite indiferent ce loc de munca.

În această teză de disertație  m-am axat pe pacientul sistemului nostru de sănătate, deoarece să ajuți nu este așa de ușor pe cât am crede. Din propria experientă pot spune ca donarea de sânge in Timișoara e o adevărată testare a răbdării, adică de la coadă pană la donarea efectivă de sânge este ceva de așteptat.

Din acest motiv am decis să incerc sa fac acest proces de care avem mare nevoie de el, să devină mai ușor de urmat și cel mai importnat să determine câți mia mulți cetățeni sa doneaze pentru a elimina problema lipsei de sânge in Timișoara.

## 1.4 Soluții asemănătoare

În ceea ce privește soluțiile asemănătoare momentan nu există pe piață o aplicație de genul acesta, insă privind din perspectiva de implementare a acestei aplicați am luat in considerare dezvoltarea aceasteia ca aplicație mobilă aceasta necesitând alte tehnologi ca Android, instalarea unui intreg sistem de dezvoltare iar in ceea ce privește tehnologia mobilă era nevoie de dezvoltarea acesteia și in varianta pentru utilizatori OS.

Consider că navigatorul web prezintă avantajul de a nu fi nevoie de o instalare de sistem prea mare, consumul de resurse este mai mic, iar spre deosebire de varianta de aplicație mobilă nu determină utilizatorul să facă setări in plus; ca de exemplu să intaleze aplicația, ceea ce presupune ca utilizatorul să aibă spațiu de stocare pe telefon, pe când utilizatorul vrea numai sa vizualizeze centrele de donare. Un efort in plus pentru utilizatorul care având in vedere că in 7 secunde acesta i-si pierde interesul pentru orice vizualizează in mediul online, acesta nu va dedica atâta timp iar scopul aplicației nu este atins. Ca și exmeplu de aplicație in continuare ca voi prezenta aplicația numită ’Blood App’. Este o aplicație a celor de Crucea Roșie din America prin intermediul căreia utilozatri isi pot planifica intalniri, motivează donatori și nu in ultimul rand i face conștienți de puterea ce o au in a salva multe vieții prin un pic mai mult interes poți salva multe vieți.

In ceea ce privește partea de aplicații mobile există mai multe exemple insă pentru navigator web nu am găsit incă.

# Capitolul 2. Fundamentare teoretică

Dezvoltarea unei aplicatii web care iși propune sa ofere ușurarea procesului de donare de sânge atât din punct de vedere a mediciilor de acolo cat și din punct de vedere al pacienților.

După o studiere a interacțiunii oameniilor cu mediul online am decis sa fac aplicația web deoarece este usor de accesat indiferent de unde ești și nu necesită acțiuni in plus din partea utilizatorului.Câteva criterii ce trebuie îndeplinite sunt : portabilitatea, accesibilitatea și multe altele este nevoie să le luăm in considerare când decidem dezvoltarea unei aplicații pe o anumită platformă.

## 2.1 Navigator web

Un navigator web este un program pentru recepția, prezentarea și transferearea informaților in lumea larga a internetului(World Wide Web). O altă definiție: prin *browser* se înțelege un program de „navigare” (virtuală) în web. De aceea, în loc de cuvântul „*browser*” se poate folosi și termenul general „navigator”. Navigatoarele web funcționează pe baza anumitor protocoale, care îl leagă pe utilizator de paginile web stocate (definite) pe servere web specializate. Cele mai des folosite protocoale web sunt HTTP, HTTPS și FTP.[1]

Un navigator este alcătuit dintr-un set de programe care permite afișarea și manevrarea informațiilor bazate pe text, imagini și sunet precum și rularea unor programe pe care siturile web și documentele le pot include (applet-uri, scripturi). Fiecare navigator are o casetă de text unde utilizatorul poate introduce adresa documentului sau a sitului dorit, adresă care este unică pe lume (univocă), numită (*Uniform Resource Locator* sau URL). În cazul în care utilizatorul nu cunoaște adresa exactă , el poate introduce drept "cheie de căutare" o porțiune mică de text specific pe care documentul ar trebui să îl conțină. Navigatorul transmite acest text unor aplicații speciale din web numite motoare de căutare. Acestea caută în multitudinea de documente sau situri respectivul text, oferind apoi ca rezultat o listă de adrese care conțin

textul căutat. Utilizatorului nu îi mai rămâne decât sa aleagă - eventual prin mai multe încercări - locația dorită. În realitate această listă de adrese poate fi uneori extrem de lungă, de ordinul sutelor de mii de linii sau chiar și mai lungă, caz în care este nevoie de o strategie de căutare mai exactă.[1]

În general, documentele și paginile web pe care le afișează browserele sunt, la dorința autorilor lor, interconectate prin tehnologia Hipertext, care permite saltul simplu de la un document sau si la altul, printr-o simplă apăsare pe maus.[1]

## 2.2 Tehnologii folosite

În ceea ce privește tehnologie utilizate pentru a efectua această teză de disertație am încercat sa combin partea de date(backend + bază de date) cu cea vizuală(frontend) pentru a obține o aplicație cât mai bine conturată dar nu în ultimul rând reușită și ușor de utilizat, deoarece este foarte important să determini câmpul vizual al utilizatorului să urmărească toată aplicația pentru a îndeplini scopul creării acestei lucrări.

Am ales tehnologiile de mai jos pentru ca toate sunt bazate pe **JavaScript** (**JS**) care este un limbaj de programare orientat obiect bazat pe conceptul prototipurilor. Este folosit mai ales pentru introducerea unor funcționalități în paginile web, codul Javascript din aceste pagini fiind rulat de către browser. Limbajul este binecunoscut pentru folosirea sa în construirea siturilor web, dar este folosit și pentru acesul la obiecte încastrate (embedded objects) în alte aplicații. A fost dezvoltat inițial de către Brendan Eich de la Netscape Communications Corporation sub numele de Mocha, apoi LiveScript, și denumit în final JavaScript.

În ciuda numelui și a unor similarități în sintaxă, între JavaScript și limbajul Java nu există nicio legătură. Ca și Java, JavaScript are o sintaxă apropiată de cea a limbajului C, dar are mai multe în comun cu limbajul Self decât cu Java.

Până la începutul lui 2005, ultima versiune existentă a fost JavaScript 1.5, care corespunde cu Ediția a 3-a a ECMA-262, ECMAScript, cu alte cuvinte, o ediție standardizată de JavaScript. Versiunile de Mozilla începând cu 1.8 Beta 1 au avut suport pentru E4X, care este o extensie a limbajului care are de a face cu XML, definit în standardul ECMA-357. Versiunea curentă de Mozilla, 1.8.1 (pe care sunt construite Firefox și Thunderbird versiunile 2.0) suportă JavaScript versiunea 1.7.[2]

## 2.3 NodeJS

Moto:„Cine a văzut vreodată o bijuterie frumos cizelată de bijutier cu ajutorul ciocanului?” Jan Amos Comenius

In ceea ce privește prima mea interactiune cu node js acesta a apărut din pură curiozitate și anume cum ar fi să dezvolti o aplicație atât partea de interacțiune cu utilizatorul cât si partea de prelucrare a datelor, așa am decoperit ceea ce inseamna sa poți fa ce asta.În continuare vă voi prezenta câteva aspecte legate de nodeJS.

In urma cu ceva timp am descoperit combinația de programare web cu tot ce inseamnă ea și am cunoscut-o sub forma de pachet format din (Apache) +PHP însă modurile de operare dintre acest mix și lucrul cu nodeJS nu sunt chiar asemănătoare, chiar destul de indepărtate.

NodeJS spre deosebire de combinatia de mai sus acesta nu se ocupă de prelucrarea requesturilor, spre deosebire de Apache+PHP, unde Apache se ocupă de căutarea tuturor fisierelor ce sunt cerute iar PHP la randul său se ocupă cu prelucrarea requesturilor de tip GET, POST, oferind ca răspuns un vector de date ușor de interpretat.În PHP fiecare request este separat și nu se știe nimic re restul, pe când in nodeJS totul se întampla in același thread(fir de execuție) iar pe lângă asta nu ai acces la prelucrarea la nivel de request(cerere) și anume cu grijă și atenție prelucrezi fiecare cerere in parte.

Ca să vedem și un exemplu de cod in nodeJS, și anume clasicul exemplu `HelloWorld`:

// Librăriile din node.js se numesc module iar în acest exemplu incărcam   
modulul pentru HTTP  
**var** http = require('http');  
  
// Creăm un server HTTP ce răspunde la toate requesturile cu un  
header 200 și mesajul Hello World  
  
**var** server = http.createServer(**function** (request, response) {  
 response.writeHead(200, {"Content-Type": "text/plain"});  
response.end("Hello World!\n");});  
// Ascultăm pe portul 3000, adresa de IP default este 127.0.0.0(local)  
server.listen(3000);  
  
// Scriem la consolă ce facem  
console.log("Serverul tău rulează la http://127.0.0.1:8000/");

Un alt avantaj ce prezintă nodeJS este că trimite date fără a fi nevoie sa reface conexiunile, spre deosebire de PHP unde după trimiterea datelor conexiunea se inchide.În figura 1 este prezentat model de procesare al nodeJS.

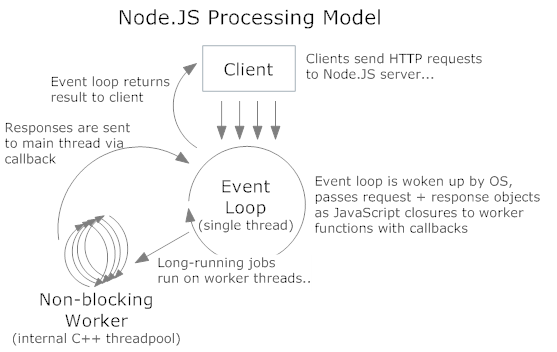


Fig. 1 NodeJS Modelul de procesare[3]

Cu toate aceste informații apar intrebări naturale ca :”Ce să construiești in nodeJS?” ,“Care sunt beneficiile?” ,“Pentru ce tipuri de proiecte se pretează a fi folosit?”. Luându-le in ordine avem așadar:

”Ce să construiești in nodeJS?”

* Aplicatii in timp real – aplicatii web, de retele sociale si instant messaging, software de chat, dashboard-uri pentru monitorizare in timp real etc.
* Aplicatii single-page
* Aplicatii event-driven
* Aplicatii care trebuie sa proceseze mii de conexiuni si fluxuri de date catre alte sisteme
* Aplicatii care asigura un schimb intensiv de date cu back-end-ul
* Aplicatii mobile in Node.js – folosind API-uri JavaScript pentru aplicatii mobile compatibile cu Node.js
* Dispozitive IoT – tehnologie “wearable”, dispozitive embedded si robotica[4]

Beneficiile folosirii nodeJS:

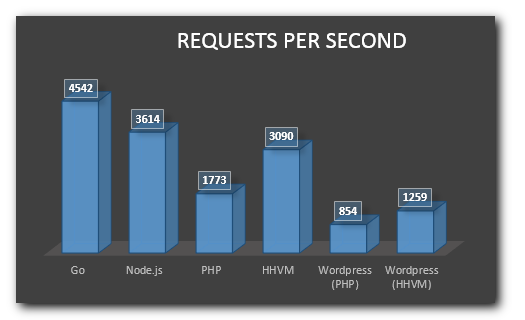
* Este foarte scalabil, datorita arhitecturii asincron, procesarii event-driven si folosirii JavaScript
* E foarte rapid: in comparatie cu alte limbaje, aplicatiile scrise in Node necesita mai putine linii de cod, mai putine fisiere, pot fi construite mai rapid si cu mai putini programatori. Aplicatiile in Node.js nu sunt doar construite mai rapid, ci si ruleaza mai repede, avand timpi de raspuns mult redusi si procesand mai multe cereri pe secunda in comparatie cu majoritatea alternativelor
* Suporta sisteme de comunicatii specifice IoT, cum ar fi MQTT, care este proiectat pentru update-uri de status rapide din partea unor dispozitive de mici dimensiuni
* Are o productivitate ridicata si, folosind JavaScript, permite implementarea a numeroase functionalitati intr-un timp scurt, construirea rapida a unui produs, obtinerea de feedback de la utilizatori si re-iterarea intr-un timp redus
* It’s inexpensive to test and deploy using pay-as-you-grow services
* Presupune costuri reduse pentru testare si lansare prin folosirea serviciilor de tip pay-as-you-grow
* Codul e scris intr-un singur limbaj, dar poate rula pe mai multe platforme
* Ofera posibilitatea de-a proiecta pe partea de client si pe cea de server intr-un fel care nu necesita oscilarea intre multiple tehnologii, datorita cuplarii stranse intre client si server
* Ofera posibilitatea de-a proiecta pe partea de client si pe cea de server intr-un fel care nu necesita oscilarea intre multiple tehnologii, datorita cuplarii stranse intre client si server
* E compatibil cu multe module, librarii si extensii open source, disponibile in cadrul comunitatilor puternice formate in jurul Node si inregistreaza o rata de adoptie in piata in continua crestere
* E o solutie perfecta pentru implementarea de proxy-uri pentru API-uri REST, indeplinind in acelasi timp toate standardele de performanta, datorita codului sau de interactiune si API-ului usor de scris, suportului pentru streaming si autentificare, si posibilitatilor de monitorizare
* Datorita operarii pe un singur thread, abordarii event-driven si modelului non-blocking I/O, Node.js practic accepta in permanenta cerinte, deoarece nu e nevoie sa astepte sa citeasca sau sa scrie operatii – ceea ce constituie o solutie eficienta pentru a face fata la sute de mii de cerinte concurente[4].

Fig.2 Compararea NodeJS cu alte limbaje in materie ce cereri pe secunda[5]

În Figura2 este prezentată o comparare între NodeJS și alte limbaje de programare in ceea ce privește evoluția în funcție de cereri pe secundă, se observă că dintr-un număr de 6 limbaje comparet NodeJS se află pe locul 2, fată de mult cunoscultul PHP.

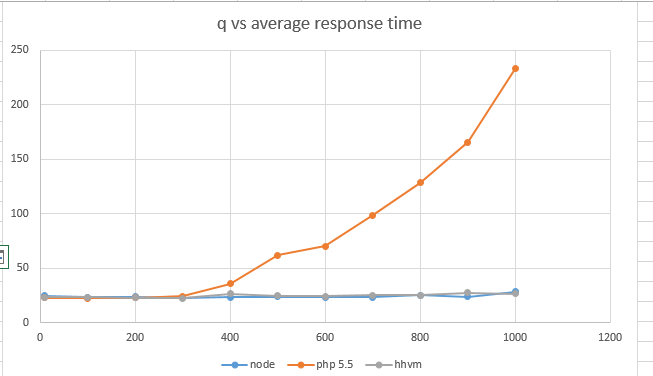


Fig.3 Comparatie cerinte HTTP +CPU [6]

În Figura 3 este prezentată o comparație directă cu PHP în cea ce privește cereriel HTTP și cum influențează activitatea procesorului, iar in cazul limbajului de programare PHP, se observă cu mult activitatea crescută a procesorului.

Ca și o mica concluzie nodeJs este folosit pentru dezvoltarea de aplicații web la nivel de server in limbajul Javascript. Toate modulele in cod menționate și alte funcționalități suplimentare se administreaza cu npm, de exemplu daca dorit sa folosit un framework in nodejs acesta se “instaleaza” la nivel de aplicație astfel: npm install <nume modul/framework>, după instalarea respectivelor module acestea se accesează in aplicație cu ajutorul funcției “require()” care specifică folosire unui anume modul.

Ca exemplu de module avem : module predefinite (built-in): privitoare la tehnologii Web – http, https, url, querystring referitoare la fișiere – fs, path văzând rețeaua – net, dns, dgram, tls,… resurse privind sistemul de operare – os, child\_process alte aspecte de interes – buffer, console, util, crypto[7]

## 2.4 MongoDB



Din puncte de vedere al construirii unei aplicații trebuie cercetat ce se pretează nevoile aplicație și nu in ultimul rând clientui. Dacă ne gandim la tipurile de baze de date expuse in lumea tehnologiei iată o poza ce descrie acestă clasificare.

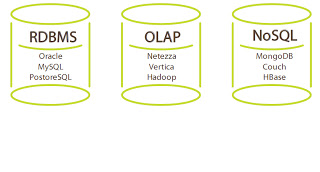


Fig. 4 Clasificare baze de date [9]

NoSql desemnează o categorie de bază de date care NU sunt construite după modelul bazelor de date relaționale(RDBMS,Relationa Database Management Systems).Bazele de date NoSQL s-au dezvoltat alături de companii de internet, cum ar fi Google, Amazon si Facebook. In cazul acestor companii, care trateaza cantităţi imense de date, solutiile traditionale RDBMS nu au putut face fata. Sistemele de baze de date NoSQL au fost dezvoltate pentru a gestiona volume mari de date care nu urmează neapărat o schemă fixă. Datele sunt împărţite între diverse masini (din motive de performanţă şi limitări de spatiu), operatiile de JOIN nu pot fi utilizate, si nici nu exista garantii ACID (Atomicitate - fiecare tranzactie va fi "totul sau nimic", in sensul in care esecul unei parti a unei tranzactii conduce la esecul intreagii tranzactii, deci la numodificarea bazei de date,

Consistenta - orice tranzactie va determina trecerea bazei de date dintr-o stare consistenta, valida in raport cu regulile definite (triggere, constrangeri, etc) intr-o alta stare tot consistenta. Izolare - executia concurenta a tranzactiilor va determina trecerea intr-o stare care poate fi obtinuta si prin executia secventiala a tranzactiilor respective.

Durabilitatea - tranzactiile finalizate (pentru care s-a dat commit) vor ramane asa si in cazul aparitiei unor probleme hard sau soft , care in cazul RDBMS-urilor garanteaza procesarea in siguranta a tranzactiilor.[9]

Principalele caracteristici ale sistemelor NoSQL sunt:

Nu folosesc SQL ca limbaj de interogare

Nu ofere garantii ACID complete: De obicei, eventual, numai coerenta este garantată pentru tranzacţiile limitate la un singur element de date. Acest lucru înseamnă că, dupa o perioadă suficient de lungă de timp în care nu au fost trimise modificările, toate update-urile (actualizarile) se vor propaga în cele din urmă prin intermediul sistemului.

Arhitectura distribuită, toleranta la defecte[9]

Sistemele NoSQL folosesc o arhitectura distribuita, deci datele sunt pastrate într-un mod redundant, pe mai multe servere. În acest fel, sistemul poate cu uşurinţă prin adăugarea de mai multe servere, iar caderea unui server poate fi tolerată.

Acest tip de bază de date de obicei, scaleaza pe orizontală şi este utilizat pentru a gestiona cantităţi mari de date, atunci când performantele în timp real sunt mult mai importante decât consistenţa (ca in cazul indexarii unui număr mare de documente, a paginilor de pe site-urile webde mare trafic, sau a livrarii stream-urilor media).

Sistemele NoSQL de baze de date sunt adesea extrem de optimizate pentru operatii de regasire/adaugare şi oferă putine functionalitati legate de memorarea inregistrarilor (memorarea inregistrarilor sub forma unor perechi, cheie-valoare). Flexibilitatea run-time redusa (faţă de sistemele SQL) este compensată prin creşteri semnificative de performanţă şi scalabilitate pentru modele de anumite date.

Practic, sistemele de gestionare a bazelor de date NoSQL isi dovedesc utilitatea atunci când se lucrează cu o cantitate mare de date, iar natura datelor nu impune un model relaţional pentru structura de date.

Datele ar putea fi structurate (relaţiile dintre elemente sunt mai putin importante), insa ceea ce contează in cazul sistemelor NoSQL este capacitatea de a stoca şi de a prelua cantitati mari de date: se pot stoca milioane de perechi cheie-valoare în una sau câteva tablouri asociative sau milioane de înregistrări de date. Acest lucru este deosebit de util pentru analizele statistice sau în timp real pentru liste tot mai mari de elemente (cum ar fi mesaje pe Twitter sau log-urile de pe serverele Internet pentru grupuri mari de utilizatori. [9]

MongoDB are printer carecteristicile de mai sus memorează datele ca documente in format BSON (asemanator JSON (JavaScript Object Notation)).

In Figura 4 sunt prezentate caracteristiciile de stocare a MongoDB in paralele cu bazele de date de tipul SQL.

|  |  |
| --- | --- |
| **Termen SQL** | **Termen MongoDB** |
| Baza de date | Baza de date |
| Tabela | Colectie |
| Rand (inregistrare) | Document |
| Index | Index |
| Coloana | Camp |
| Join | Embedding & link |

Fig.4 Paralela MongoDB cu SQL [10]

Comparativ cu bazele de date relaționale, in cele relaționale avem inregistrarea ca structură fixă specificată in crearea tabelei, în timp ce documentele dintr-o colecție pot avea câmpuri diferite, singura “schemă” necesară find aceea ca fiecare document are un camp \_id care este o cheie unică.

Un exemplu de schema in MongoDB arată astfel:

**var** userSchema = **new** mongoose.Schema({  
email: {  
type: String,  
unique: **true**,  
required: **true**,  
lowercase:**true**},  
name: {  
type: String,  
required: **true**},  
password:{  
type: String,  
required:**true**},  
role: {  
type:String,  
**enum**:['Vizitator', 'Admin\_jud', 'Admin\_centru', 'Angajat\_centru', 'Client'],  
**default**:'Vizitator'  
}  
}, { collection: 'users' });

Facilități oferite de MongoDB:

Memorarea orientată pe documente: In plus față de cele amintite anteriror, in interiorul unui document pot fi imbricate (embedding) alte documente. Intre documente se pot specifica și legături.

Posibilitati de indexare completa: Pentru fiecare colectie se pot specifica indecsi secundari si compusi. Orice "camp" (atribut) poate fi indexat.

Interogari flexibile, de orice tip, asupra documentelor: se pot realiza cautari ale unor campuri, ale unor domenii de valori, cautari pe baza unor expresii regulate sau cautari care utilizeaza functii JavaScript definite de catre utilizator.

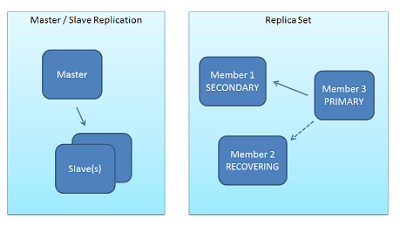
Replicare si disponibilitate: MongoDB asigura replicarea asincrona a datelor intre servere in cazul unei probleme. Practic, fiecare document va avea o multime de replici.[10]

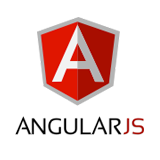
Fig.6 Replicare și disponibilitate inMongoDB [10]

Configurarea este automată, astfel incat noi servere pot fi adăugate la o baza de date care lucreaza (deci fără afectarea funcționalității).

Scalabilitate orizontală: Pot fi adaugate noi servere), fără să fie afectata functionalitatea (in timpul in care baza de date "lucreaza") [10].

Așadar in cazul alegerii bazei de date este nevoie de a lua in considerare toți termenii insă cel mai important pentru ce va fi folosit sistemul.

## 2.5 Angular 2.0 && Typescript



Angular este framework ce are la bază JavaScript, este open-source de asta atât Google cat și comunitatea de pe internet contribuie și intrețin fiind in special folosit in dezvoltarea de aplicații intr-o singura pagina(SAP).Componentele Javascript complementează Apache Cordova, care este un framework folosit aplicați destinate platformei mobile.Oferă un framework pentru client-side(MVC- model-view-controller) si (MVVM) model-view-viewmodel.

MVC(model-view controler) este reprezentat de arhitectura sa care separă partea de stocare a datelor de cea de prezentare si prelucrare.Așadar avem trei trei clase distincte:

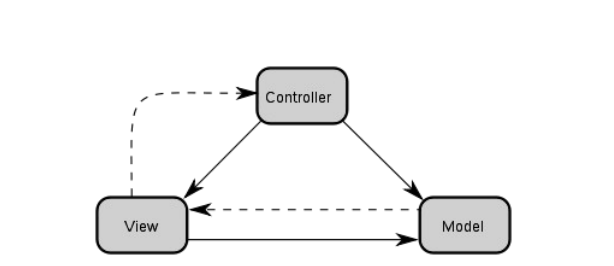
* Model-ul se ocupa de comportarea si datele aplicatiei; raspunde la cereri despre starea sistemului, la cereri de schimbare de stare si notifica utilizatorul atunci cand aceste schimbari au avut loc pentru ca acesta sa poata reactiona.
* View-ul transpune model-ul intr-o forma care permite o interactionare usoara, in mod tipic o interfata vizuala. Pot exista multiple view-uri pentru un singur model pentru scopuri diferite. Controller-ul primeste input de la utilizator si initiaza un raspuns in urma cererilor catre obiectele model.
* Controller-ul este cel care controleaza celelalte doua clase de obiecte, view si model, instructandu-le sa execute operatii pe baza input-ului primit de la utilizator[11]

Fig.7 Diagrama MVC[11]

Model-View-ViewMode(MVVM)

Model-View-ViewModel (MVVM) este un alt model arhitectural derivat din MVC. La fel ca și în cazul modelului Pasive-View nu există o dependență între view și model, dar spre deosebire de acesta view-ul nu este pasiv ci poate actualiza controller-ul care în acest caz este reprezentat de ViewModel. Cu toate acestea din punct de vedere al testării automate doar ViewModel-ul trebuie testat, întrucât comunicarea între View și ViewModel se face prin data binding, care cel puțin teoretic nu poate conține erori de logică.

Modelul arhitectural Model-View-ViewModel este un model specific platformelor .Net: Windows Presentation Foundation și Silverlight. Pentru a ușura dezvoltarea aplicațiilor MVVM au fost dezvoltate o serie de toolkit-uri care facilitează implementarea implementarea acestui model arhitectural. Unul din cele mai cunoscute astfel de toolkit-uri este MVVM Light Toolkit [12]

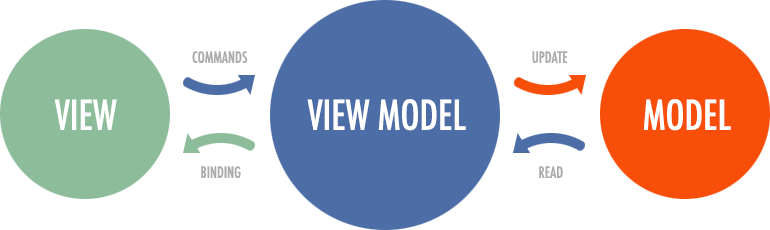


Fig.8 MVVM diagram[13]

Framework-ul Angular functionează astfel: citește pagina HTML care fost construită cu componente specifice, angular le interpretează acele atribute aparte ca și directive pentru a lega intrările sau ieșirile a unei parți de pagina cu modelul, adică iti permite să folosesti HTML ca și templete lăsându-te să sa extinzi sintaxa html pentru a exprima componentele aplicației tale cât mai clar și suncint.[15]

Data-binding-ul și Dependecy Injection oferă bucuria că elimină mult code pe care erai nevoit să il scrie iar partea cea mai bună este că totul se intâmplă in broswer-ul web ceea ce il face n partener ideal cu alte thenologi pentru partea de server.[15]

În ceea ce privește exemplu de code in Angular:

<md-sidenav-container class="example-container" fullscreen>  
  
<md-sidenav #sidenav mode="side" class="app-sidenav" md-is-locked-open="$mdMedia('gt-md')" >  
  
<app-menu></app-menu>  
  
</md-sidenav>  
  
<md-toolbar color="primary">  
<i class="material-icons" (click)="sidenav.toggle()">menu</i>  
  
Timisoara Doneaza  
<span class="app-toolbar-filler"></span>  
</md-toolbar>  
<div class="md-card\_container">  
<md-card></md-card>  
</div>  
  
</md-sidenav-container>  
  
<router-outlet></router-outlet>

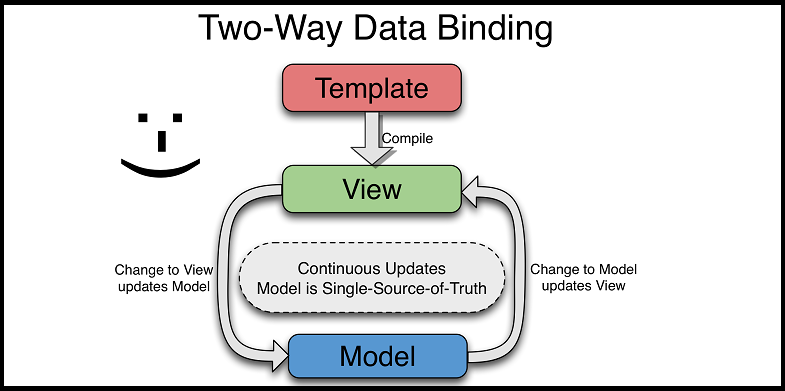


Fig.9 Cum functioneaza AngularJS[14]

De ce sa folosim Angular?

* Usor de ințeles
* Un proiect Google
* Creare de directive (taguri gandite de voi “<om></om>”)
* Documenție detaliată
* Folosit pentru REST
* Destul rapid
* Are la bază MVC
* Prezintă plusul de Dependecy Injection

## 2.5.1 Typescript



Prin definiție, "TypeScript este JavaScript pentru dezvoltarea aplicațiilor."TypeScript este un limbaj puternic compilat, orientat spre obiect și orientat spre obiect. A fost proiectat de Anders Hejlsberg (designerul C #) la Microsoft. TypeScript este atât un limbaj, cât și un set de instrumente. TypeScript este un superset tastat de JavaScript compilat în JavaScript. Cu alte cuvinte, TypeScript este JavaScript plus câteva caracteristici suplimentare.[17]

Typescript este un superset pentru Javascript care in principal oferă static typing, clase si interfețe.Spre deosebire de competitori săi(CoffeScript și Dart) codul javascript poate fi mixat cu Typescript => JavaScript este TypeScript, dar TypeScript nu este JavaScript.

Dar Typescript trebuie să fie compilat in JS inainte inainte de a putea rula in orice motor JavaScript.Asta inseamnă ca nu poți să incorporezi TypeScript intr-o pagină web direct folosind tag-ul <script> dar fisierele de tip Typescript, adică (test.ts) pot fi compilate in fișiere Javascript pentru a putea fi folosite.[16]

Exemplu de code in TypeScript:

**class** Greeter {  
greeting: **string**;  
**constructor**(message: **string**) {  
**this**.greeting= message;  
}  
greet() {  
**return** "Hello, " + **this**. greeting;  
}  
}

acesta devine transpus în simplu cod Javascript:

**var** Greeter = (**function** () {  
**function** Greeter(message) {  
**this**.greeting= message;  
}  
Greeter.prototype.greet = **function** () {  
**return** "Hello, " + **this**.greeting;  
};  
**return** Greeter;  
}());

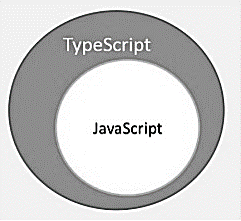
Analizând putin cele 2 parți de cod de mai sus, putem observa cateva asemănări ca și în JavaScript, acum codul este reprezentat de o clasă, asta prezintă si primul avantaj TypeScript.Typescript include suport pentru noi caracteristici cum ar fi modulele, clase, constant, interfețe, si funcțile Lambda care sunt parte din ECMAScript6.[16]

Fig.10 Ce este Typescript/Javascript[17]

În ceea ce privește legătura dintre Typescript și ECMAScript (specificațiille ECMAScript sunt specificațiile standardizate pentru limbajul de scripting).



Fig 11.Alinierea Typescript cu ECMAScript[17]

De ce să folosim TypeScript?

Compilarea – Javascript este un limbaj de interpretare, prin urmare, trebuie să fie rulat pentru a testa că este valid. Înseamnă că scrieți toate codurile doar pentru a nu găsi nici o ieșire, în cazul în care există o eroare. Prin urmare, trebuie să petreceți ore să încercați să găsiți erori în cod. Translatorul TypeScript furnizează caracteristica de verificare a erorilor. TypeScript va compila codul și va genera erori de compilare dacă va găsi un fel de erori de sintaxă. Acest lucru ajută la evidențierea erorilor înainte ca scriptul să fie rulat.[17]

Puternic Static Typing( tipul este verificat la compilare) - JavaScript nu este puternic in ceea ce privestetipurile. TypeScript este livrat cu un sistem opțional de tipare și tipare statică, prin TLS (Service Language Language). Tipul unei variabile declarate fără tip poate fi dedus de TLS pe baza valorii sale.

TypeScript acceptă definiții de tip pentru bibliotecile JavaScript existente. Fișierul DefinitionScript (cu extensia .d.ts) oferă definiții pentru bibliotecile JavaScript externe. Prin urmare, codul TypeScript poate conține aceste biblioteci. TypeScript suportă concepte de programare orientate pe obiecte, cum ar fi clase, interfețe, moștenire etc.[17]

## 2.5.6 Expres.js

Express, este un framework de aplicații web pentru Node.js, lansat ca software gratis și cu sursă deschisă, sub licența MIT. Acesta este conceput pentru a construi aplicații web și API. Acesta este frameworks de facto de server standard pentru Node.js. Express este partea din spate a pachetului MEAN, împreună cu baza de date MongoDB și cadrul frontal AngularJS.[19].

Express este un framework ce sta deasupra de serverele din NodeJs și face să fi mai ușor de folosit, printer caracteristiciile din Express se enumeră Middleware(ce reprezintă un pod de comunicare intre sistemele softeare și baza de date) și capabilitatea de rutare.Introduceți Express, un cadru care acționează ca un strat de lumină deasupra serverului web Node.js, ceea ce face mai plăcută dezvoltarea aplicațiilor web Node.js.[20]

Express este similar filozofic cu jQuery. Oamenii doresc să adauge conținut dynamic la paginile lor web, dar API-urile browserului de vanilie pot fi verbose, confuze și limitate în caracteristici. Dezvoltatorii de multe ori trebuie să scrie codul boilerplate, și o mulțime de ea. jQuery există pentru a reduce acest cod de boilerplate simplificând API-urile browserului și adăugând noi funcții utile. Aceasta este în esență.[20]

# 

# Capitolul 3. Specificațiile de proiectare ale aplicației

Aplicația Timișoara Donează intră in categoria aplicațiilor web, adică dedicată atât telefoanelor mobile, cu acces la internet dar și calculatoarelor, astfel acoperind o gamă mult mai largă de utilizatori, și oferind acces de oriunde in timp foarte scurt la informații generale legate de donare, cum ar fi centrul de are nevoie de sânge și programul de lucru.

Am ales partea de web a posibilităților deoarece, știm cu toți că nu toată lumea se descurcă să descarce o aplicație pe de Google Play, de aceea m-am gândit că voi ajuta mai mulți utilizatori si sunt sigură că se vor descurca din prima în a accesa și a interacționa cu aplicația, astfel sănătatea și tehnologia mână in mână nu mai pare un plan așa de îndepărtat.

Interfața oferă butoane cu nume sugestii dar și mici imagini, pentru o mai bună înțelegere a ceea ce poți face in cadrul acestei aplicație, și nu te poți rătăci nicicând în aplicație deoarece ai acces mereu la un meniu sugestiv, care te poate întoarce mereu la pagina de start. Așadar să începem prin a prezenta ce oferă de fapt această aplicație și cum arată intru totul.

## 3.1 Accesul la informații legate de donare

În această partea a aplicației sunt prezentate informații generale legate de donare, mai exact aplicație prezintă o hartă, a Timișoarei pe care sunt reprezentate centrele de donare din județ, iar când utilizatorul dă click pe unul dintre puncte, sunt afișate informațiile legate de situația fiecărui centru pe grupe de sânge, iar in dreptul fiecărei grupe de sânge apare un status referitor la cantitatea de sânge dintr-o anumită grupa, la un anumit centru.

M-am gândit că astfel voi reuși să conectez utilizatorul cu detalii importante și cu astfel de metode este eliminată o problemă majoră, aceea de acces la informație specifică și acea de a nu mai sta atâta timp când dorești să donezi. Din perspectiva interfeței cu utilizatorul am introdus elemente comune din harta celor de la Google, astfel utilizatorul fiind mai obișnuit cu ceea ce reprezintă acea hartă și vor înțelege mai repede ce am vrut să transmit.

## 3.2 Accesul pe nivele în aplicație

Din punct de vedere al accesului în aplicație, acesta l-am proiect să fie făcut pe nivele, mai exact exista mai multe tipuri de utilizatori, fiecare dintre acesta având acces la anumite funcționalității din aplicație. Am gândit astfel pentru o viitoare dezvoltare la nivel de țară, astfel va fi creat un alt nivel de acces, în continuare vă voi prezenta proiectarea aplicației pe nivele.

### 3.2.1 Administrator de județ

Prin intermediul acestui nivel de acces, sunt accesibile funcționalității de adăugare de noi administratori la nivel de județ, adăugare de noi centre de donare, ștergerea unui centru de donare. Sunt disponibile nu foarte multe funcționalități pentru a nu obosi utilizatorul cu foarte multă informații sau prea multă interacțiune din partea sa, ca utilizator.

La întâlnirea situațiilor de proiectare și dezvoltare de aplicați pentru utilizatorii din mediul de sănătate, este foarte important fie să reproducem o copie destul de fidelă a ceea ce folosesc dânșii din viața de zi cu zi, adică să transpunem destul de bine formularele acestora cu care prelucrează pentru a-și face treaba.

### 3.2.2 Administrator de centru

Administratorul de centru are acces la prelucrarea utilizatorilor atât din punct de vedere ca angajați cât și la utilizatori la nivel de centru, acesta poate adăuga noi conturi ce pot acces sistemul. Din punct de vedere al interfeței aplicație prezintă butoane ce prezintă clar acțiunile ce le pot face tot odată prezintă mici imagini pentru a fi cât mai sugestive și aplicația sa fie cât mai ușor de utilizat.

În tot acest timp, m-am gândit că deși este un administrator autentificat in sistem și acesta să aibă acces la două funcționalități, pe care toți cei care interacționează cu aplicația le pot folosi, opțiunea de Acasă, ce prezintă informații legate de donare, și Harta care prezintă centrele de donare din județ.

### 3.2.3 Angajat centru

Din perspectiva angajatului, am proiectat și implementat aplicația astfel încât acesta să nu fie aglomerat de toate informațiile ce putea fi afișate, ci doar să fie prezente doar cele care i pot face munca mia ușoară, în același timp păstrând tema de culoare a aplicației și coloristica lejeră. Acest tip de utilizator are doar 2 opțiuni specifice și 5 in total, fiecare dintre acestea oferind o posibilitate de a face tot procesul de donare de sânge mult mai ușor, deoarece prezintă acțiunile ce le poate face angajatul dintr-un centru de donare, și anume să adauge un donator sau să caute informații despre acesta.

### 3.2.4 Donator

Nivelul de acces ca donator reprezintă, utilizatorul care își creează un cont în aplicație și dorește să fie într-un fel sau altul activ în ceea ce privește donare. Acest nivel l-am gândit ca fiind cel cu informații foarte puține prezentate utilizatorului, deoarece este foarte important ca aplicația să determine pe majoritatea cei care o accesează să își creeze cont, deoarece opțiunile acestui tip de utilizator sunt cele de a crea notificări și de a-și vizualiza și actualiza propriul profil; astfel creânduși notificări acesta este mereu în interacțiune cu aplicația dar în același timp nu este sufocat cu foarte multă informație.

### 3.2.5 Vizitator

Vizitatorul în cadrul acestei aplicație este reprezentat ce cea/cel care accesează aplicație pentru simpla informație oferită de opțiunea Harta, cea care reprezintă centrele de donare cu informațiile aferente legate de nevoia de o anumită grupă de sânge, iar acest vizitator poate alege în funcție de grupa sa de sânge dar și de nevoia unui centru să doneze iar prin acest lucru aplicație și-a atins scopul pentru care a fost creată.

# Capitolul 4. Implementarea aplicației

## 4.1 Arhitectura aplicației

Aplicația “Timișoara Donează” am gandit-o pentru ușurarea accesului la serviciul de donare de sânge din Timișoara, fiind destinată atât cetățeniilor insă și cadrelor medicale, deaorece oferă accesul la informatie foarte utilă in acest process.

Din punct de vedere al accesului la aplicație, orice fie cu un telefon intelligent și internet sau fie cu un calculator și acces la internet poate acces și utiliza aplicația. Datorită tuturor tehnologiilor și a framewok-urilor folosite in această aplicație, adică combinația dintre MongoDB +Express+ AngularJS2+NodeJS apare noțiunea de MEAN stack, ceea ce este o anotare a celor 4 tehnologi.

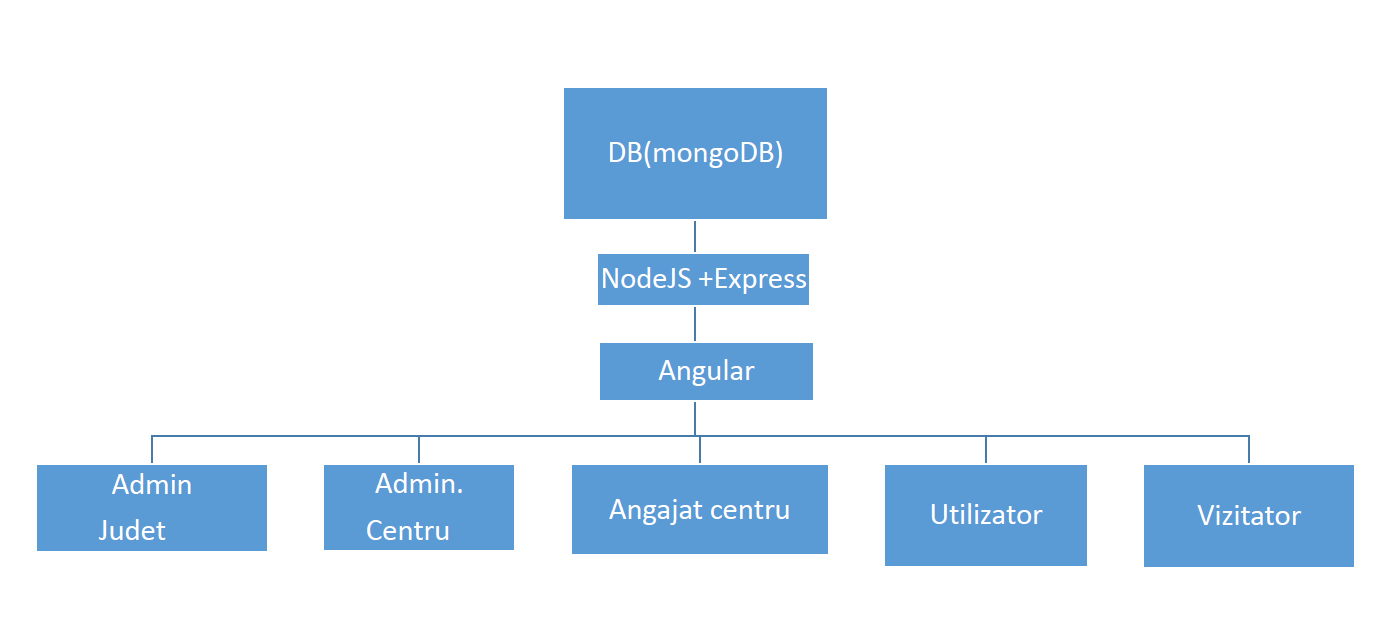


Fig 12. Arhitectura Aplicației

Utilizatori au acces la interfața aplicației prin intermediul căreia pot accesa atât informații generale legate de donare insă pe langă asta au acces la o hartă a Timișoarei unde sunt reprezentate toate centrele de donare iar in plus de asta fiecare centru de donare reprezentat pe harta are afișate informații legate de nevoia de sânge in centrul respectiv.

Interfața este simplă ușor de înțeles cu o coloristică ce nu deranjează deloc vizual utilizatorul.

Aplicația “Timișoara Donează” este compusă din mai multe module.Una dintre funcționalități, poate cea mai importantă dintre toate este posibilitatea de alege un centru de pe hartă in funcție de grupa vizitatorului și nevoile centrului respectiv.

Consider că este una dintre cele mai importante funcționalității datorită faptului că spre deosebire de alte posibilității de informare disponibile pe internet și nu numai oferă informații generale insă nimic concret care sa iți usureze tie tot acest drum și prices de donare, la urma urmei fiecare dintre noi stie cum este cu timpul, iar acest proces doar face din acest gand bun pe care il avem de a dona să fie doar mai greu de liniștit.

Când aplicația este pornită utilizatorului îi este prezentată pagina de start a aplicației aceasta este reprezentată de un fundal reprezentat de o poză din domeniu și de un header si de un meniu ce se afiseză prin glisare la apăsare butonului de meniu.

Butoanele meniului sunt reprezentative și sugestive, numele de prezentare este “Acasă”,care va duce utilizatorul pagina de start a aplicației indiferent pe unde a ajuns in navigare, “Harta” butonul care va prezenta centrele de donare cu informațiile legate de nevoie de o anumită grupă de sânge.

## 4.2. Funcțiile aplicației

În materie de folosire de tehnologii, partea vizuală este construită din ideei personale combinate cu elemente din Angular, partea cea mai interesantă în a folosi Angular este că ai la dispozitie pe lângă multe elemente predefinite ce te ajută sa nu mai te gandești la aranjarea in ecran si la tranziții ale elementelor, mai precis unul dintre elementele folosite in aplicație este Sidenav, care impreună cu side-nav-container iti oferă avantaje ca: ajustarea la redimesionarea ecranului, trazintia elementlui de navigare, fară a scrie eu toate detalii in CSS, adică fără a avea grija stilizări elementlor.

Un alt avantaj in materie de lucruri vizuale este ca poti crea elemente, numite in Angular componente, de exemplu creăm o componentă de tipul meniu in care punem elemente ce le intâlnim peste tot in aplicație, așadar pentru a nu face copiere de cod și să facem aplicație greu de înteles și mai alea de citi pentru cei care pot veni mai târziu să refacă lucruri în aplicație.

<md-sidenav-container class="example-container" fullscreen>  
  
<md-sidenav #sidenav mode="side" class="app-sidenav" md-is-locked-open="$mdMedia('gt-md')" >  
  
<app-menu></app-menu>  
  
</md-sidenav>  
  
<md-toolbar color="primary">  
<i class="material-icons" (click)="sidenav.toggle()">menu</i>  
  
Timisoara Doneaza  
<span class="app-toolbar-filler"></span>  
</md-toolbar>  
<div class="md-card\_container">  
<md-card>staff admin judet</md-card>  
</div>  
  
</md-sidenav-container>

Mai departe in aplicație utilizatorul este intampinat de un ecran cu o coloristică placuta ochiului și primitoare, dacă dicutăm despre actiuniile fiecărui buton, avem astfel:

Ecranul principal oferă un meniu către accesarea hărtii, butonul de intoarcere in pagina principal indifferent de unde este utilizatorul in aplicație, mai departe diferă in functie de nivelul de acces, al utilizatorului.

## 4.3 Detalii de implementare

Aplicația a fost dezvoltată pe etape:

* proiectarea structurii și implementarea specificațiilor
* proiectarea și implementarea modului de administrare al aplicației introducerea datelor necesare creări aplicației
* implementarea facilităților puse la dispoziția utilizatorilor și încadrarea acestora în cadrul modulelor de care aparțin
* definitivarea aspectului grafic al aplicației

Proiectarea aplicației a început ca o idee cum am dori oare atât pacientul cât și medicul să arate această aplicație, iar după ce am adunat destul de multe răspunsuri, am făcut un mic studiu legat de tehnologii, iar alegerea a avut la bază atât ideea de a nu deranja utilizatorul și de a oferi atât utilizatorului cât și medicului o aplicație ușor de folosit. În ceea ce privește accesul acesta am decis să fie pe nivele, dezvoltând în acest fel un modul de autentificare, necesar, acestui tip de implementare al aplicației, după care am revenit la opțiunile ce vor fi prezentate indifferent de nivelul de acces al utilizatorului.

Pe lângă modulul de autentificare fiecare modul din aplicație este de fapt un acces pe un anumit nivel, fiecare dintre aceste prezentând aplicație intr-un anumit mod, cu anumite opțiuni și cu o anumită interfată, păstrând tot odată nota generală a aplicației în materie de coloristica și afisare de informații, toate acestea pentru ca utilizatorul sa nu defină debusolat, dacă de exemplu are cont de donator dar și de administrator.

Implementarea facilităților este prezentată in codul aplicație, elementele vizuale fiind reprezentate de elemnete din framework-ul Angular și continunând cu apeluri către partea de date, unde se intâmplă prelucrarea datelor și trimiterea de răspunsuri inapoi către interfața utilizatorului, unde vor fi prezentatea fie rezultatele in sine fie afisarea unor anume elemente vizuale.Am definit aplicația ca fiind compusă din 5 module principale(administrator județ, administrator centru, angajat centru, utilizator, vizitator) fiecare dintre acestea prezintă anumite funcționalități și anumite opțiuni.

# Capitolul 5. Utilizare, rezultate experimentale

## 5.1 Utilizarea aplicației

Aplicația Timișoara Donează a fost concepută pentru a ușura procesul de management al donării de sânge dar și ușurarea muncii celor din domeniul medical. Acest lucru am încercat pe cât de mult posibil să-l implementez prin dezvoltarea de diferite facilități care în primul rând să ușureze accesul la informații necesare utilizatorului în momentul când dorește să doneze. În materie de beneficii aplicația oferă utilizatorului:

* O modalitate mai bună de exprimare a dorinței de a face bine
* Reducerea de materiale fizice necesare pentru management
* Acces la informație o organizare foarte eficientă
* Asigurarea unui suport în caz că apar probleme

Mai departe voi prezenta cum arată tot ce am explicat mai sus, am folosit elemente de Google Maps, însă am modificat, puțin din partea vizuală a clasicei hărții folosite de Google, oferind un punct de vedere mai vizual mai ușor de înțeles a cea ce reprezintă harta.

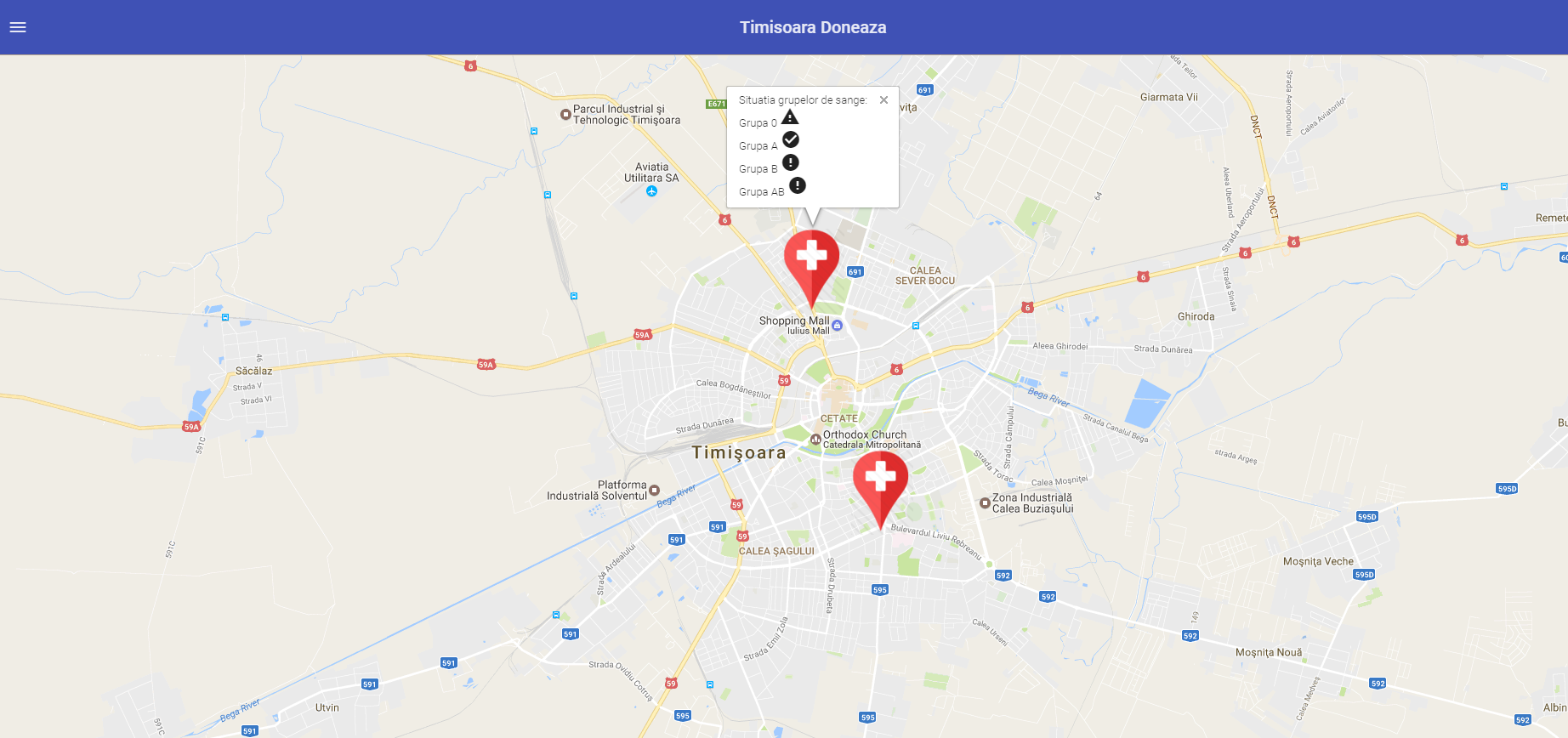


Fig. 13 Interfața opțiunii Harta

Figura 13 prezintă harta cu cele 2 puncte, reprezentând centrele de donare, iar acea mica fereastră reprezintă situația stocului de sânge la acele centru.

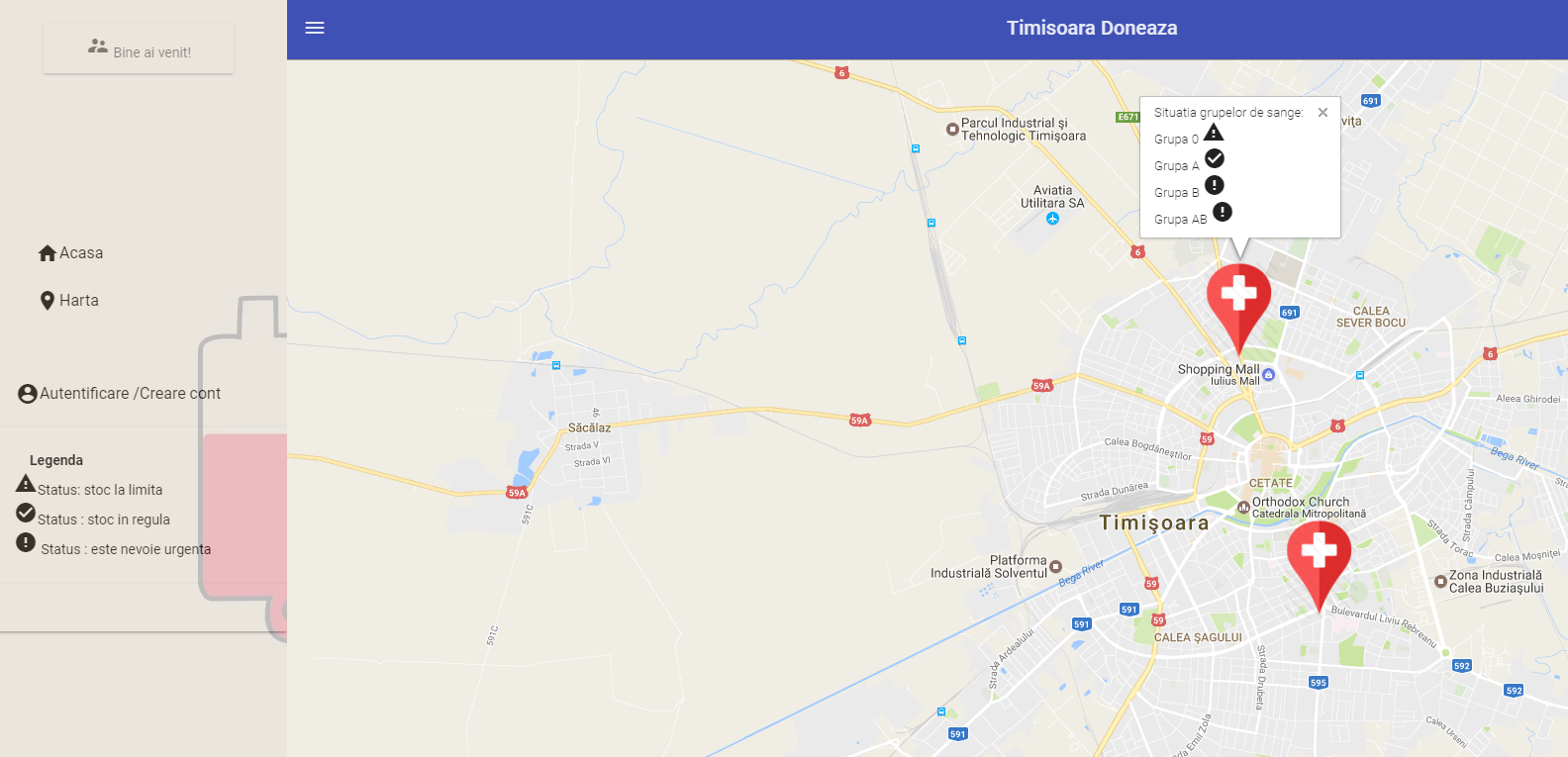


Figura 13.1 Interfata Legendei

Figura 13.1 reprezintă meniul in secțiunea de hartă cu tot cu legendă ce explică acele mici imagini din dreptul fiecărei grupe de sânge.

De remarcat este faptul că din punct de vedere vizual, centrele nu sunt reprezentate prin acea clasică iconiță ce reprezintă un punct pe hartă ci prin ceva mai diferit ce face referire directă la faptul că acele puncte au legătură cu ceva medical, pentru a fi mai sugestive, încă de la prima interacțiune cu utilizatorul, îl face să înțeleagă mult mai repede ce înseamnă de fapt toată această aplicație și cum poate influența direcția fiecăruia in materie de donare de sânge.

## 5.2 Accesul in aplicație pe mai multe nivele

Pentru a acoperi o gamă cât mai mare de opțiuni și pentru a permite cât mai multor utilizatori să o folosească am gândit un sistem pe nivele, așadar in funcție de cine ești, de exemplu administrator de județ, administrator de centru, angajat, utilizator, vizitator, ești întâmpinat de diferite opțiuni. În continuare voi prezenta si explica ce înseamnă fiecare dintre acestea.

### 5.2.1 Nivelul de acces la Administrator de județ

 Acest nivel a fost gândit, ca fiind ce mai înalt nivel de acces, după cel de sistem, tot odată acest nivel de acces prezintă cele mai multe opțiunii, fiind proiectat să folosească la managementul acestui proces de donare la nivel de județ. Mai departe voi prezenta cum arată funcționalitățile si împreună cu explicațiile aferente.

 Figura 14 Prezentarea paginii de start

Figura 14.1 Prezentarea paginii se start cu opțiuniile meniului

Figura 14 și figura 14.1 prezinta pagina de start ce conține informații generale despre donare și un scurt mesaj de motivare, in continuare după navigare prin aplicație utilizatorul este întâmpinat de meniu, de aici mai departe totul diferă in funcție de utilizator si drepturile acestuia.

În acest subcapitol prezint opțiunile pentru Administrator de județ, așadar mai departe utilizatorul merge la pagina de autentificare(Figura 15), care arată astfel



Figura 15 Interfata pentru eutentificare

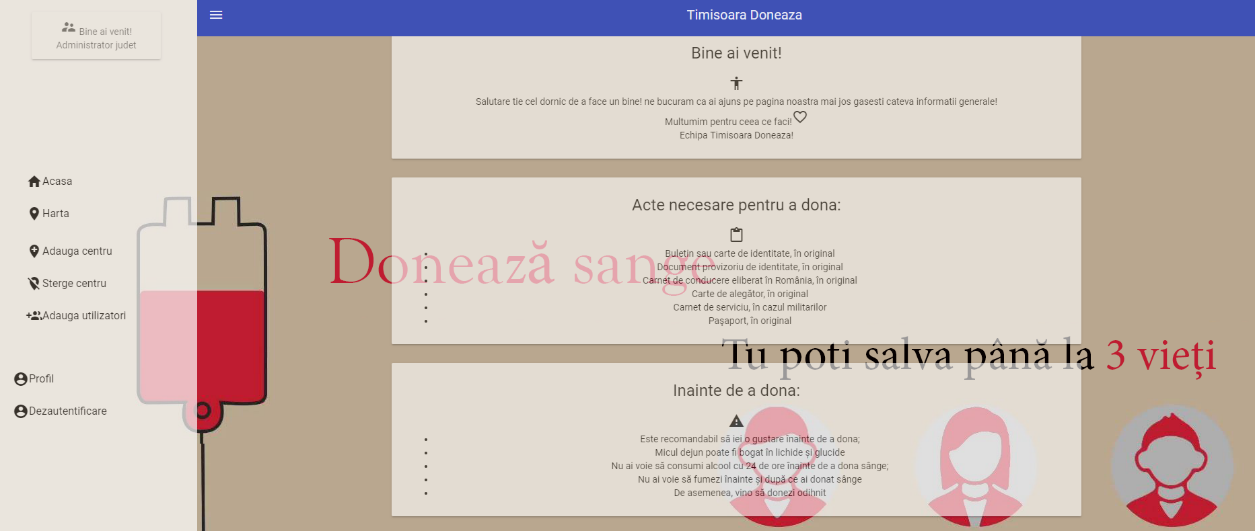
După autentificare, utilizatorul este redirecționat către pagina de strat, însă cu diferența majoră că meniul, prezentat in figura 15.1, este diferit de cel înainte, prezentând mai multe moduri de interacționare cu aplicația, oferind opțiunile:

Figura 15.1 Interfata meniului după autentificare

* Adăugare centru

În această secțiune utilizatorul are opțiunea de a adăuga noi centre de donare, în sistem și implicit pe hartă, in tot acest timp, utilizatorul are in stânga meniul pentru a nu se “rătăci” in aplicație(Figura 15.2).

Figura 15.2 Interfata de adaugare a unui centru

* Ștergere centru

În acestă secțiune utilizatorul are posibilitatea de elimina din centrele de donare, acestă opțiune a fost creată pentru cazurile în care unul dintre centre se mută, iar atunci va fi nevoie de noi coordonate să fi intoduse în sistem(Figura 15.3).

Figura15.3 Interfata de stergere a unui centru

* Adăugare de utlizatori

În acestă secțiune utlizatorul nostrum, Administrator de județ, are opțiunea de a adăuga noi utlizatori la nivel de administrator de judet(Figura 15.4). Sunt necesare introducere numelui noului utlizatorului, email-ul(pe baza caruia va fi identificat la partea de autrentificare) și o parola intială pe care utilizatorul o va scchimba după prima autentificare în secțiunea profil.

Figura 15.4 Interfata de adaugare de utilizatori la nivel de judet

### 5.2.2 Nivelul de Administrator de centru de donare

Acest nivel a fost gândit, ca fiind următorul nivel de acces, după cel de administrator de județ, tot odată acest nivel de acces prezintă opțiunii gândite să folosească la managementul acestui proces de donare la nivel de centru, în Figura16 sunt prezentate cum arată funcționalitățile împreună cu explicațiile aferente.

Figura 16 Meniul pentru utilizatorul de tip admin. centru

În ceea ce privesc opțiunile acesta, in ordinea din meniu sunt:

* Adăugarea de angajati

Această opțiune prezentată in Figura 16.1 permite adăugarea de noi angajați la nivel de centru acesteia la rândul lor vor menaja tot ce ține de donatori și datele acestora. Sunt necesare introducere numelui noului utlizatorului, email-ul(pe baza caruia va fi identificat la partea de autrentificare) și o parolă intială pe care utilizatorul o va schimba după prima autentificare în secțiunea profil.



Figura 16.1 Opțiunea de adaugare de angați

* Adăugarea de utilizatori

Acestă optiune permite adăugarea de noi utilizatori pe nivelul de administrator de centru opțiune prezentată in Figura 16.2 și la fel ca in cazul adăugării angajaților in sistem sunt necesare introducere numelui noului utlizatorului, email-ul(pe baza caruia va fi identificat la partea de autrentificare) și o parolă intială pe care utilizatorul o va schimba după prima autentificare în secțiunea profil.



Figura16.2 Adăugarea de utilizatori la nivel de centru de donare

### 5.2.3 Nivelul de Angajat la un centru de donare

Acest nivel l-am gândit astfel încât să fie ușor de folosit de cei din cadrul centrelor de donare, cei care interacționează direct cu donatori, practic cei care trebuie să consume cel mai puțin timp cu prelucrarea de informați cât și cu donarea efectivă, de aceea m-am rezumat la câteva opțiuni ce le-am considerat că o să îl ajute in a prelua cetățenii mai repede. Așadar opțiunile prezentate utilizatorului de tip Angajat centru sunt următoarele:

* Adaugă donator

În această secțiune utilizatorul are acces către un formular, prezentat in Figura 17, unde sunt necesare introducerea datelor despre donator, cum ar fi Numele, Prenumele, Vârsta, Grupa de sânge, Adresa si nu in ultimul rând CNP-ul care este identificator unic al fiecărui om.

|  |  |
| --- | --- |
|  | Figura 17 Interfata de adaugare a unui donator |

* Căutare donator

Cu această opțiune, reprezentată in Figura 17.1, angajatul respectiv, utilizatorul în aplicația noastră are posibilitatea de căuta date despre un donator doar cu ajutorul CNP-ului.

Figura 17.1 Interfata de căutare după donatori

### 5.2.4 Nivelul de Utilizator

Din punct de vedere funcțional acest nivel, oferă, deocamdată, opțiunea de a crea notificări, opțiune ce permite crearea unor așa zice alarme, adică ca utilizator cu cont activ poți să creezi alarme, fie să fi anunțat când un centru are nevoie de grupa respectivului donator, fie cât au trecut deja 6 luni de la ultima donare, toate astea pentru al determina pe donator, să rămână conectat la acest serviciu, și să nu renunțe la a mai dona după prima donare.

* Crearea de notificări

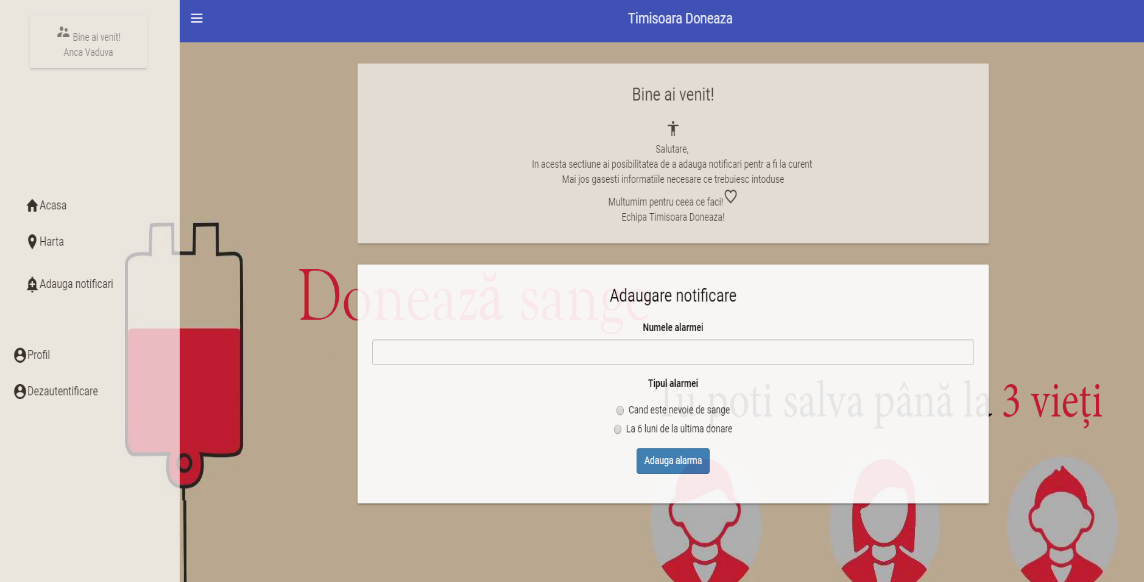


Figura 18.1 Interfata de adăugarea de notificări

In Figura 18.1 este prezentată interfața funcționalității de adăugare a notificăriilor care prezintă 2 opțiuni, una dintre ele este ca utilizatorul sa fie notificat in cazul in care este nevoie de grupa sa de sânge la un anumit centru, iar cea dea doua este acea de a fi notificat cand au trecut 6 luni de la ultima donare.

O altă funcționalitate, este acea de vizualizare a profilului, aici utilizatorul are prezentat tot felul de informați legate de profilul sau.

* Vizualizare profil

Figura 18.2 Interfata de vizualizare a ptofilului

În Figura 18.2 este prezentată opțiunea ca utilizatorul să poată modifica anumite informați legate de profilul său de donator in cadrul aplicație, acesta poate introduce date care nu există incă in sistem sau poate actualiza datele curente.

### 5.2.5 Nivelul de Vizitator

Acest nivel de utilizare permite acces la nivel general, ceea ce permite accesul la Harta(cu centrele de donare) și opțiunea de creare de cont prezentată in Figura 19.

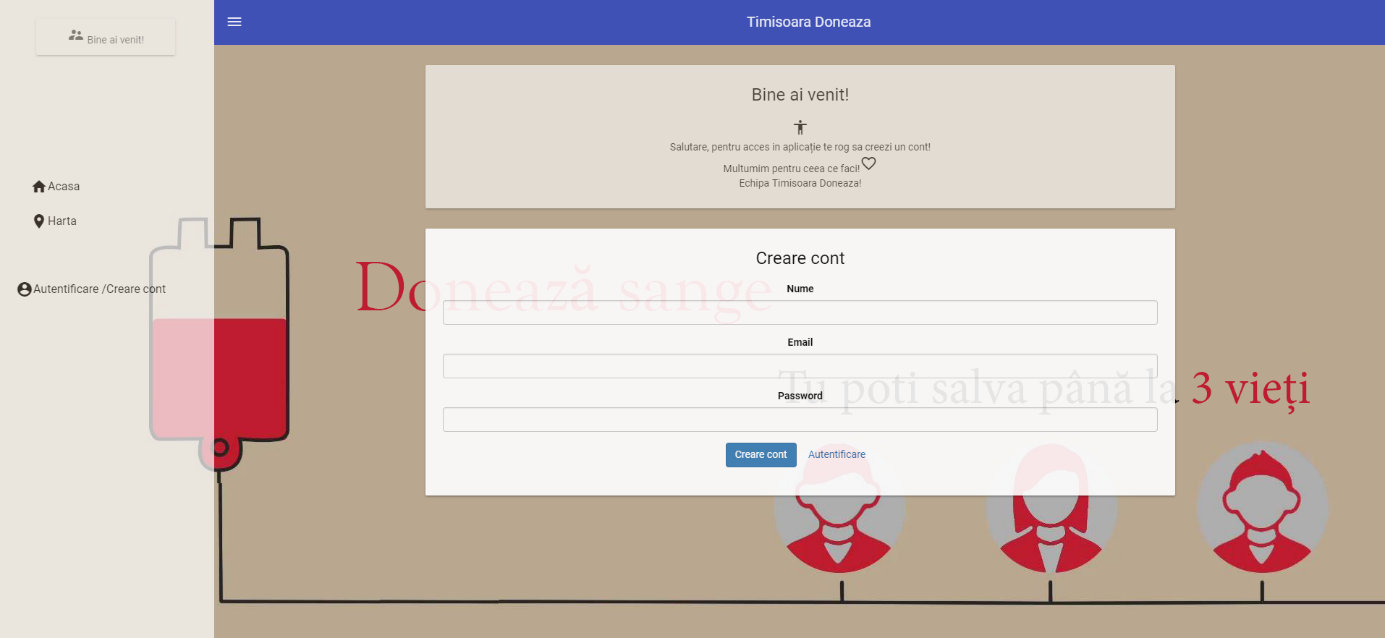


Figura 19 Interfata de creare a unui cont

Toate profilurile, din moment ce sunt accesate mai prezintă si opțiunea de a părăsi aplicația, prin opțiunea de dezauntentificare, astfel pe același calculator se poate accesa aplicația pe toate nivele de acces.

## 5.3 Testarea aplicației

Un test constă în execuția programului pentru un set de date de intrare convenabil alese, pentru a verifica dacă rezultatul obținut este corect.

Pentru a avea o situație a funcționării aplicației am decis ca aplicația să fie testate pe module după care de-a lungul creării aplicația a devenit mai mare iară cu modulele funcționale testate în prealabil a rezultat funcționalitatea finala a aplicației.

Am ales testarea pe module pentru o mai bună urmărire a erorilor ce pot apărea și totodată rezolvarea cât mai rapidă a erorilor in cazul în care apar. Totodată, testând pe module diferite, nu exista riscul sa nu funcționeze diferite componente din pricina celorlalte.

Testarea unui modul este realizată de programatorul care implementează modulul. Toate celelalte teste sunt efectuate, în general, de persoane care nu au participat la dezvoltarea programului. [21]

Scopul testării unui modul este de a se stabili că modulul este o implementare corectă a specificației sale (conforma cu specificația sa). Specificația poate fi neformală sau formală.[21].

În cursul testării unui modul, modulul este tratat ca o entitate independentă, care nu necesită prezența altor componente ale programului. Testarea izolată a unui modul ridică două probleme:

-simularea modulelor apelate de cel testat;

-simularea modulelor apelante.

Testarea fiecărui modul s-a făcut manual. Aceasta a fost realizată prin rularea aplicației pe dispozitivele disponibile și oferită utilizatorilor dornici să testeze aplicația după care s-a cerut o impresie generală.

Testarea modular oferă in alt avantaj și anume acela că greșelile pot fi urmărite și rezolvate și prin cadrul mediului de dezvoltare, iară odată ce ai întâlnit o problemă a doua oară vei observa mai rapid problema, deci vei câștiga timp.Ca dezvoltator al unei aplicații este important să urmărești aplicația la perioade de timp, pentru a observa nereguli și de ce nu pentru îmbunătățirii.

# Capitolul 6. Concluzii și direcții de continuare a dezvoltării

## 6.1 Concluzii

Prin prezenta lucrare am dorit prezentarea unei soluții pentru organizarea noastră ca cetățenii și membrii ai acestei societății pentru acest voluntariat numit donarea de sânge care deși pare o chestie de nu o foarte mare importantă sau cel puțin nu vorbim foarte des despre asta, însă avem nevoie de acest gest din partea fiecăruia dintre noi, ca cetățeni.

Din propria experiență am testat acest proces de donare și sincer vorbesc când zic că nu m-a determinat să mă mai întorc, deși știu că nu asta ar trebui să mă determine să merg să donez. Am întâlnit și situați când au fost organizate, de exemplu, la locul de munca, campanii de donare pentru a ajuta cumva in lupta asta împotriva lipsei de sânge. Nu se știe nicicând în ce situații vom ajunge fiacre dintre noi și să avem nevoie de sânge însă totuși suntem mulți oameni dar aparent rar arătăm că suntem cu adevărat. Așadar in urma experiențelor și a discuțiilor cu oameni din domeniu dar și cu simpli oameni am înțeles că nu avem disponibilă o soluție pentru rezolvarea acestei probleme.

Cum mediul online oferă acces de oriunde cu resurse puține, tot ce mai rămâne de făcut este să determin pe fiecare dintre locuitori acestui oraș, momentan, să folosească aplicația pentru a scăpa de impedimentele ca: timpul mult de așteptare până la procesul efectiv de donare, lipsa de informații specifice pentru un centru in timp real, lipsa de motivare pentru a veni la centru să doneze.

Consider că am atins destul de multe puncte specifice domeniul si mai ales rezolvării problemei in curs, însă întotdeauna este loc de mai bine, iar dezvoltarea aplicației nu va rămâne la nivelul la care este acum, urmând sa îl transform in proiect personal și doresc dezvoltarea lui la nivel național, în prezent este gândit și proiectat la nivel de Timișoara. Mai am nevoie de mai multe păreri din partea celor din domeniul medical și mai ales, supunerea la teste din partea lor, deoarece doresc să fie incantați de folosirea tehnologiei in lucrul și să descopere astfel, dacă nu au descoperit încă, avantajele tehnologiei.

## 6.2 Direcții de dezvoltare

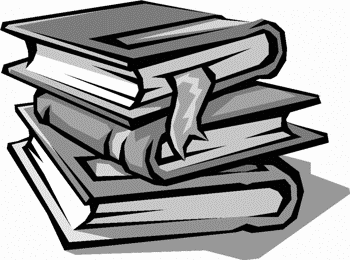
Când mă gândesc la direcțiile de dezvoltarea, am în minte următoarele, doresc să integrez mai multe funcționalități atât pentru donatori cât si pentru utilizatori de cadrul medical,una dintre funcționalităti pe care o am in plan pentru viitoare direcție de dezvoltare este ca utilizator să iși poți face o programare la centrul unde este nevoie de sângele pe care este dispus utilizatorul să il doneze pentru a ajuta pe cei care vor beneficia de acest mic ajutor dar important.Pe lângă aceasta aș dori ca pe viitor să integrez o secțiune de statistică pentru imagine de ansamblu asupra numărului de donatori si de organizarea la nivel de judet.

O funcționalitate pe care o am in plan pentru viitoare direcție de dezvoltare este ca utilizatorul să își poți face o programare la centrul unde este nevoie de sângele pe care este dispus utilizatorul să îl doneze pentru a ajuta pe cei care vor beneficia de acest mic ajutor dar important. Pe lângă aceasta aș dori ca pe viitor să integrez o secțiune de statistică pentru a avea o imagine de ansamblu asupra numărului de donatori și de organizarea donatorilor la nivel de județ, putând astfel să se face și redirecționării din anumite centre către altele în cazul in care unul dintre centre este prea aglomerat, conform statisticii.

Așadar sper ca in final să construiesc o aplicație la nivel național pentru a crea o intreaga comunitate legată de donarea de sânge, iar ce este si până acum, in ceea ce privește aplicație, a apărut in urma experințelor și a dicuțiilor cu oameni din domeniu dar și cu simpli oameni am înțeles că nu avem disponibilă o soluție pentru rezolvarea acestei probleme.

La urma urmei depinde de fiecare programator/inginer/ sau diplomat să lăsăm lumea asta măcar un pic mai bună decât am găsit-o.

# Capitolul 7. Bibliografie



[1].<https://ro.wikipedia.org/wiki/Navigator_web>, 2017

[2]. <https://ro.wikipedia.org/wiki/JavaScript> , 2017

[3].<http://www.aaronstannard.com/images/nodejs%20for%20dotnet.png>

[4].<http://www.way2web.ro/tehnologii/node-js/> , 2017

[5].http://hostingadvice.digitalbrandsinc.netdna-cdn.com/wp-content/uploads/2015/03/nodejs-vs-php-performance-requests-per-second.png

[6].<http://hostingadvice.digitalbrandsinc.netdna-cdn.com/wp-content/uploads/2015/03/nodejs-vs-php-performance-cpu-tasks.png>

[7].<https://profs.info.uaic.ro/~busaco/teach/courses/wade/presentations/web-nodejs.pdf>

[8].Node.js In action, Mike Cantelon, Nathan Rajlich, Marc Harter, T. J. Holowaychuk

[9].<http://mihaistefanescu1.blogspot.ro/p/nosql.html>

[10].http://mihaistefanescu1.blogspot.ro/p/mongodb-prezentare-generala.html

[11].<http://stst.elia.pub.ro/news/IS/IS_Teme1101/mvc.pdf>

[12].<http://mvvmlight.codeplex.com/>

[13].<https://www.devexpress.com/Products/NET/Controls/WPF/i/features/mvvm-light.png>

[14]. <http://codecondo.com/wp-content/uploads/2015/08/two-way-data-binding.png?x94435>

[15].<http://mindbowser.com/what-is-angularjs-and-how-it-works/>

[16].https://developer.telerik.com/topics/web-development/what-is-typescript/

[17].<https://www.tutorialspoint.com/typescript/typescript_overview.html>, 2017

[18].<https://image.slidesharecdn.com/slides-141203171254-conversion-gate02/95/kickstarting-nodejs-projects-with-yeoman-35-638.jpg?cb=1417632321> ,2017

[19]. <https://en.wikipedia.org/wiki/Express.js> ,2017

[20].Express in Action: Writing, Building, and Testing Node.js Applications Autor:[Evan Hahn](https://www.google.com/search?safe=strict&q=Evan+Hahn&stick=H4sIAAAAAAAAAOPgE-LVT9c3NEw2Lq4wt7A0UoJyzeLLM6qSTLRkspOt9JPy87P1y4syS0pS8-LL84uyrRJLSzLyiwDT6IngPgAAAA&sa=X&ved=0ahUKEwjih_mdraTUAhWKK5oKHTAaAlMQmxMIigEoATAQ), 2016

[21]. http://andrei.clubcisco.ro/cursuri/3ip/curs/6.1\_Testarea.doc