

CUPRINS

Capitolul 1. Introducere	4
1.1 Context	4
1.2 Domeniul temei	4
1.3 Descrierea temei	5
1.4 Soluții asemănătoare	5
Capitolul 2. Fundamentare teoretică	7
2.1 Navigator web	7
2.2 Tehnologii folosite	8
2.3 NodeJS	9
2.4 MongoDB	14
2.5 Angular 2.0 & Typescript	18
2.5.1 Typescript	21
2.5.6 Express.js	23
Capitolul 3. Specificațiile de proiectare ale aplicației	25
3.1 Accesul la informații legate de donare	25
3.2 Accesul pe nivele în aplicație	25
3.2.1 Administrator de județ	26
3.2.2 Administrator de centru	26
3.2.3 Angajat centru	26
3.2.4 Donator	27
3.2.5 Vizitator	27
Capitolul 4. Implementarea aplicației	28
4.1 Arhitectura aplicației	28
4.2. Funcțiile aplicației	29
4.3 Detalii de implementare	30
Capitolul 5. Utilizare, rezultate experimentale	32
5.1 Utilizarea aplicației	32
5.2 Accesul în aplicație pe mai multe nivele	33
5.2.1 Nivelul de acces la Administrator de județ	34
5.2.2 Nivelul de Administrator de centru de donare	37

5.2.3 Nivelul de Angajat la un centru de donare.....	39
5.2.4 Nivelul de Utilizator	40
5.2.5 Nivelul de Vizitator	42
5.3 Testarea aplicației	42
Capitolul 6. Concluzii și direcții de continuare a dezvoltării.....	44
6.1 Concluzii	44
6.2 Direcții de dezvoltare.....	45
Capitolul 7. Bibliografie	46

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. În 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 în 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 acestea sunt situații 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 situații când prezența tehnologie într-un moment cheie, în tratamentul unui pacient ar putea avea o influență mult mai mult decât pozitivă, astfel am decis să contribuim cu o idee spre crearea acestei legături destul de benefică dacă este valorificată.

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

1.3 Descrierea temei

Evoluția interacțiunii dintre om și tehnologie este vizibilă la tot pasul din ce în ce mai mult, însă nu tot timpul este folosită la nivelul său maxim. Privind în jurul nostru putem constata 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 utilizatorului, adică nu trebuie să descarce o aplicație și să o instaleze. Mereu munca ce o investești într-un lucru, serviciu, sau ce o fi el este important ca ideile tale să fie clare și înțelese de cine trebuie, toate acestea combinate cu evoluția tehnologiei și accesul la internet care a devenit virală în ziua de azi, te pot ajuta să ai rezultatele dorite indiferent de loc de muncă.

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 experiență pot spune că donarea de sânge în Timișoara e o adevărată testare a răbdării, adică de la coadă până la donarea efectivă de sânge este ceva de așteptat.

Din acest motiv am decis să încerc să fac acest proces de care avem mare nevoie de el, să devină mai ușor de urmat și cel mai important să determine câți mai mulți cetățeni să doneze pentru a elimina problema lipsei de sânge în Timișoara.

1.4 Soluții asemănătoare

În ceea ce privește soluțiile asemănătoare momentan nu există pe piață o aplicație de genul acesta, însă privind din perspectiva de implementare a acestei aplicații am luat în considerare dezvoltarea acesteia ca aplicație mobilă aceasta necesitând alte tehnologii ca Android, instalarea unui întreg sistem de dezvoltare iar în ceea ce privește tehnologia mobilă era nevoie de dezvoltarea acesteia și în 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 în plus; ca de exemplu să instaleze aplicația, ceea ce presupune ca utilizatorul să aibă spațiu de stocare pe telefon, pe când utilizatorul vrea numai să vizualizeze centrele de donare. Un efort în plus pentru utilizatorul care având în vedere că în 7 secunde acesta își pierde interesul pentru orice vizualizează în mediul online, acesta nu va dedica atâtă timp iar scopul aplicației nu este atins. Ca și exemplu de aplicație în continuare voi prezenta aplicația numită 'Blood App'. Este o aplicație a celor de Crucea Roșie din America prin intermediul căreia utilizatorii își pot planifica întâlniri, motivează donatori și nu în ultimul rând îi face conștienți de puterea ce o au în a salva multe vieți prin un pic mai mult interes poți salva multe vieți.

În ceea ce privește partea de aplicații mobile există mai multe exemple însă pentru navigator web nu am găsit încă.

Capitolul 2. Fundamentare teoretică

Dezvoltarea unei aplicații web care își propune să ofere ușurarea procesului de donare de sânge atât din punct de vedere al medicilor de acolo cât și din punct de vedere al pacienților.

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

2.1 Navigator web

Un navigator web este un program pentru recepția, prezentarea și transferarea informațiilor în lumea largă 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 să 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 și 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 să 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 că 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 încastate (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

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

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

NodeJS spre deosebire de combinația de mai sus acesta nu se ocupă de prelucrarea requesturilor, spre deosebire de Apache+PHP, unde Apache se ocupă de căutarea tuturor fișierelor ce sunt cerute iar PHP la rândul 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 în nodeJS totul se întâmplă în 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 în parte.

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

```
// Bibliotecile din node.js se numesc module iar în acest exemplu încărcăm  
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 să reface conexiunile, spre deosebire de PHP unde după trimiterea datelor conexiunea se închide. În figura 1 este prezentat model de procesare al nodeJS.

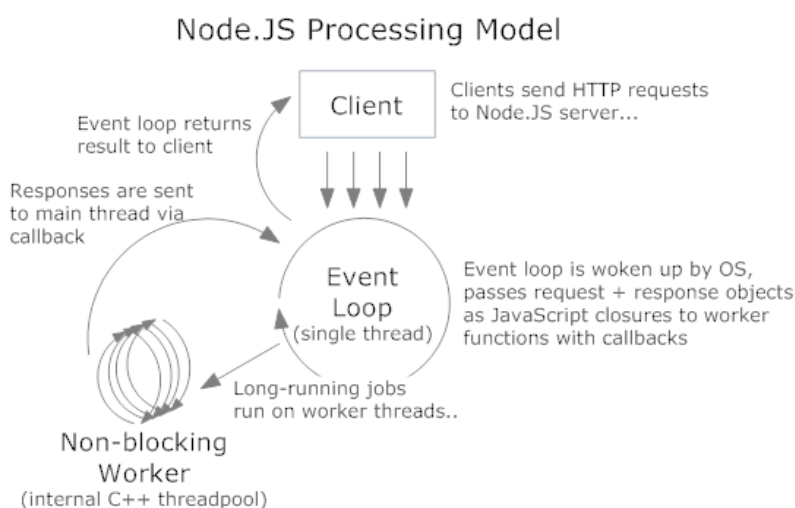


Fig. 1 NodeJS Modelul de procesare[3]

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

”Ce să construiești în nodeJS?”

- Aplicații în timp real – aplicații web, de rețele sociale și instant messaging, software de chat, dashboard-uri pentru monitorizare în timp real etc.
- Aplicații single-page
- Aplicații event-driven
- Aplicații care trebuie să proceseze mii de conexiuni și fluxuri de date către alte sisteme
- Aplicații care asigură un schimb intensiv de date cu back-end-ul

- Aplicații mobile în Node.js – folosind API-uri JavaScript pentru aplicații mobile compatibile cu Node.js
- Dispozitive IoT – tehnologie “wearable”, dispozitive embedded și robotica[4]

Beneficiile folosirii nodeJS:

- Este foarte scalabil, datorită arhitecturii asincron, procesării event-driven și folosirii JavaScript
- E foarte rapid: în comparație cu alte limbaje, aplicațiile scrise în Node necesită mai puține linii de cod, mai puține fișiere, pot fi construite mai rapid și cu mai puțini programatori. Aplicațiile în Node.js nu sunt doar construite mai rapid, ci și rulează mai repede, având timpi de răspuns mult reduși și procesând mai multe cereri pe secundă în comparație cu majoritatea alternativelor
- Suportă sisteme de comunicații 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 ridicată și, folosind JavaScript, permite implementarea a numeroase funcționalități într-un timp scurt, construirea rapidă a unui produs, obținerea de feedback de la utilizatori și re-iterarea într-un timp redus
- It's inexpensive to test and deploy using pay-as-you-grow services
- Presupune costuri reduse pentru testare și lansare prin folosirea serviciilor de tip pay-as-you-grow
- Codul e scris într-un singur limbaj, dar poate rula pe mai multe platforme
- Oferă posibilitatea de-a proiecta pe partea de client și pe cea de server într-un fel care nu necesită oscilarea între multiple tehnologii, datorită cuplării strânse între client și server
- Oferă posibilitatea de-a proiecta pe partea de client și pe cea de server într-un fel care nu necesită oscilarea între multiple tehnologii, datorită cuplării strânse între client și server
- E compatibil cu multe module, librării și extensii open source, disponibile în cadrul comunităților puternice formate în jurul Node și înregistrează o rată de adopție în piață în continuă creștere

- E o soluție perfectă pentru implementarea de proxy-uri pentru API-uri REST, îndeplinind în același timp toate standardele de performanță, datorită codului său de interacțiune și API-ului ușor de scris, suportului pentru streaming și autentificare, și posibilităților de monitorizare
- Datorită operării pe un singur thread, abordării event-driven și modelului non-blocking I/O, Node.js practic acceptă în permanentă cerințe, deoarece nu e nevoie să aștepte să citească sau să scrie operații – ceea ce constituie o soluție eficientă pentru a face față la sute de mii de cerințe concurente[4].

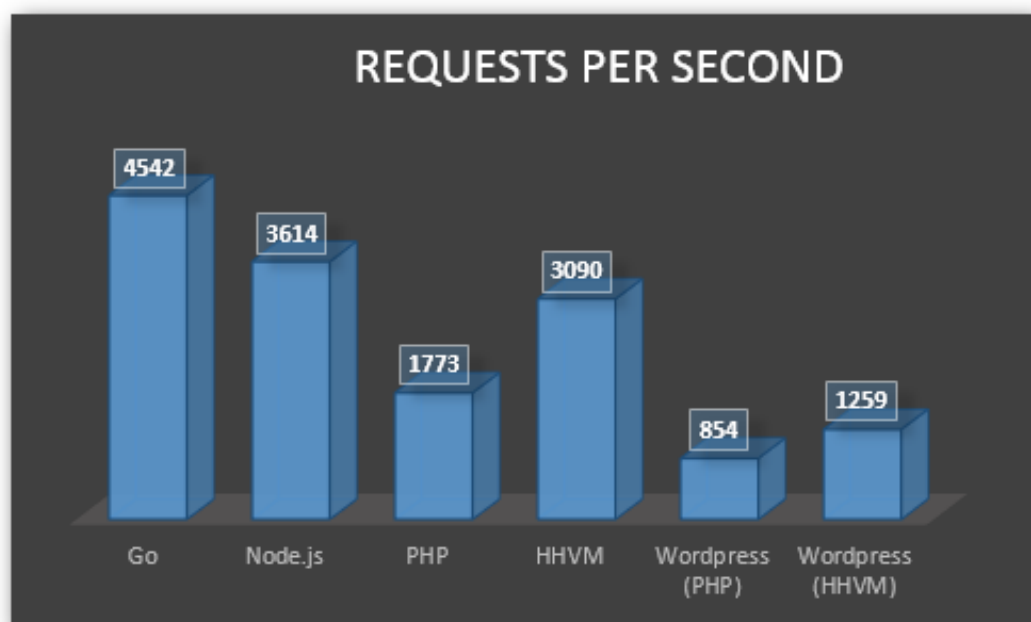


Fig.2 Compararea NodeJS cu alte limbaje în materie de cereri pe secundă[5]

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

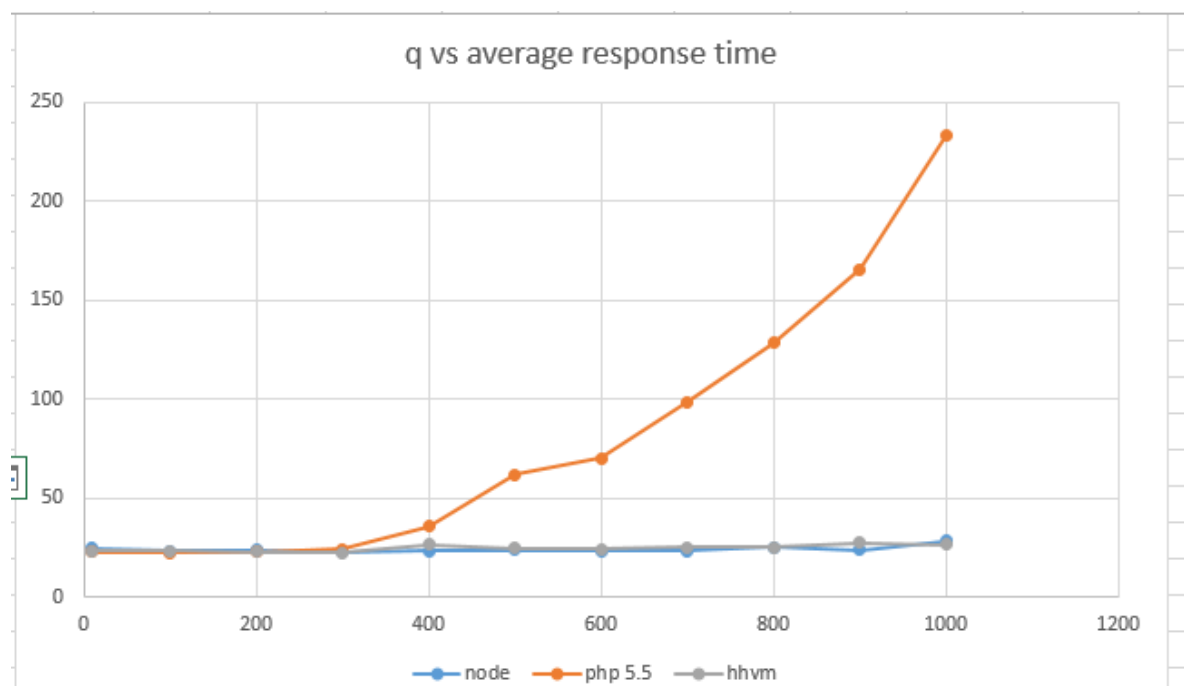


Fig.3 Comparatie cerinte HTTP +CPU [6]

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

Ca și o mică concluzie Node.js este folosit pentru dezvoltarea de aplicații web la nivel de server în limbajul Javascript. Toate modulele în cod menționate și alte funcționalități suplimentare se administrează cu npm, de exemplu dacă doriți să folosiți un framework în Node.js acesta se “instalează” la nivel de aplicație astfel: `npm install <nume modul/framework>`, după instalarea respectivelor module acestea se accesează în aplicație cu ajutorul funcției “require()” care specifică folosirea unui anumit 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 în ultimul rând clientului. Dacă ne gândim la tipurile de baze de date expuse în lumea tehnologiei iată o poză ce descrie această 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, Relational Database Management Systems). Bazele de date NoSQL s-au dezvoltat alături de companii de internet, cum ar fi Google, Amazon și Facebook. În cazul acestor companii, care tratează cantități imense de date, soluțiile tradiționale RDBMS nu au putut face față. 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 împărțite între diverse mașini (din motive de performanță și limitări de spațiu), operațiile de JOIN nu pot fi utilizate, și nici nu există garanții ACID (Atomicitate - fiecare tranzacție va fi "totul sau nimic", în sensul în care eșecul unei părți a unei tranzacții conduce la eșecul întregii tranzacții, deci la modificarea bazei de date,

Consistentă - orice tranzacție va determina trecerea bazei de date dintr-o stare consistentă, validă în raport cu regulile definite (triggere, constrangeri, etc) într-o altă stare tot consistentă. Izolare - executia concurentă a tranzacțiilor va determina trecerea într-o stare care poate fi obținută și prin executia secvențială a tranzacțiilor respective.

Durabilitatea - tranzacțiile finalizate (pentru care s-a dat commit) vor rămâne așa și în cazul apariției unor probleme hard sau soft, care în cazul RDBMS-urilor garantează procesarea în siguranță a tranzacțiilor.[9]

Principalele caracteristici ale sistemelor NoSQL sunt:

Nu folosesc SQL ca limbaj de interogare

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

Arhitectura distribuită, tolerantă la defecte[9]

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

Acest tip de bază de date de obicei, scalează pe orizontală și este utilizat pentru a gestiona cantități mari de date, atunci când performanțele în timp real sunt mult mai importante decât consistența (ca în cazul indexării unui număr mare de documente, a paginilor de pe site-urile web de mare trafic, sau a livrării stream-urilor media).

Sistemele NoSQL de baze de date sunt adesea extrem de optimizate pentru operații de regasire/adaugare și oferă puține funcționalități legate de memorarea înregistrărilor (memorarea înregistrărilor sub forma unor perechi, cheie-valoare). Flexibilitatea run-time redusă (față de sistemele SQL) este compensată prin creșteri semnificative de performanță și scalabilitate pentru modele de anumite date.

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

Datele ar putea fi structurate (relațiile dintre elemente sunt mai puțin importante), însă ceea ce contează în cazul sistemelor NoSQL este capacitatea de a stoca și de a prelua cantități 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 printre caracteristicile de mai sus memorează datele ca documente în format BSON (asemănător JSON (JavaScript Object Notation)).

În Figura 4 sunt prezentate caracteristicile de stocare a MongoDB în paralel cu bazele de date de tipul SQL.

Termen SQL	Termen MongoDB
Baza de date	Baza de date
Tabela	Colectie
Rand (înregistrare)	Document
Index	Index
Coloana	Camp
Join	Embedding & link

Fig.4 Paralela MongoDB cu SQL [10]

Comparativ cu bazele de date relaționale, în cele relaționale avem înregistrarea ca structură fixă specificată în crearea tabelului, în timp ce documentele dintr-o colecție pot avea câmpuri diferite, singura “schemă” necesară fiind aceea ca fiecare document are un câmp `_id` care este o cheie unică.

Un exemplu de schema în 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: În plus față de cele amintite anterior, în interiorul unui document pot fi imbricate (embedding) alte documente. Între documente se pot specifica și legături.

Posibilități de indexare completa: Pentru fiecare colecție se pot specifica indici secundari și compusi. Orice "camp" (atribut) poate fi indexat.

Interogări flexibile, de orice tip, asupra documentelor: se pot realiza căutări ale unor câmpuri, ale unor domenii de valori, căutări pe baza unor expresii regulate sau căutări care utilizează funcții JavaScript definite de către utilizator.

Replicare și disponibilitate: MongoDB asigură replicarea asincronă a datelor între servere în cazul unei probleme. Practic, fiecare document va avea o multitudine de replici.[10]

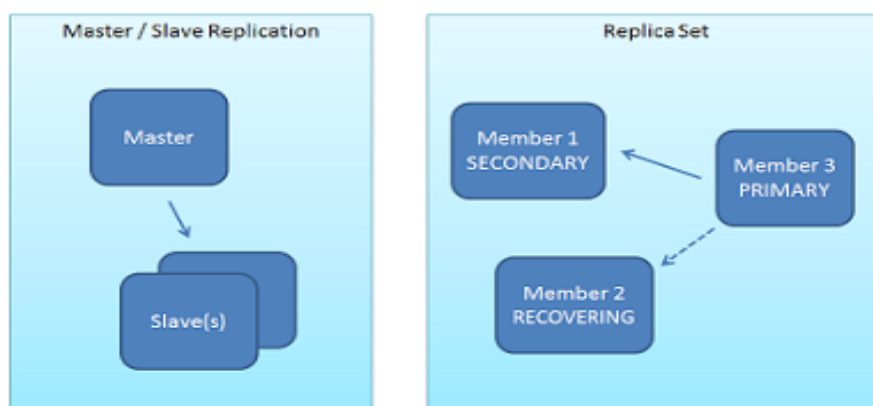


Fig.6 Replicare și disponibilitate în MongoDB [10]

Configurarea este automată, astfel încât noi servere pot fi adăugate la o bază de date care lucrează (deci fără afectarea funcționalității).

Scalabilitate orizontală: Pot fi adăugate noi servere, fără să fie afectată funcționalitatea (în timpul în care baza de date "lucrează") [10].

Așadar în cazul alegerii bazei de date este nevoie de a lua în considerare toți termenii însă 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 cât și comunitatea de pe internet contribuie și întrețin fiind în special folosit în dezvoltarea de aplicații într-o singură pagină (SPA). Componentele JavaScript completează Apache Cordova, care este un framework folosit aplicații destinate platformei mobile. Oferă un framework pentru client-side (MVC- model-view-controller) și (MVVM) model-view-viewmodel.

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

- Model-ul se ocupă de comportarea și datele aplicației; răspunde la cereri despre starea sistemului, la cereri de schimbare de stare și notifică utilizatorul atunci când aceste schimbări au avut loc pentru ca acesta să poată reacționa.
- View-ul transformă model-ul într-o formă care permite o interacționare ușoară, în mod tipic o interfață vizuală. Pot exista multiple view-uri pentru un singur model pentru scopuri diferite. Controller-ul primește input de la utilizator și inițiază un răspuns în urma cererilor către obiectele model.
- Controller-ul este cel care controlează celelalte două clase de obiecte, view și model, instructându-le să execute operații pe baza input-ului primit de la utilizator [11]

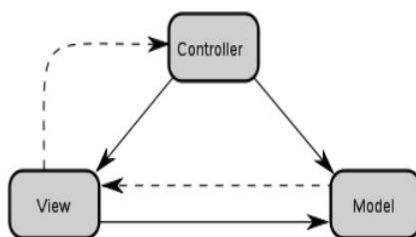


Fig.7 Diagrama MVC[11]

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 acestui model arhitectural. Unul din cele mai cunoscute astfel de toolkit-uri este MVVM Light Toolkit [12]

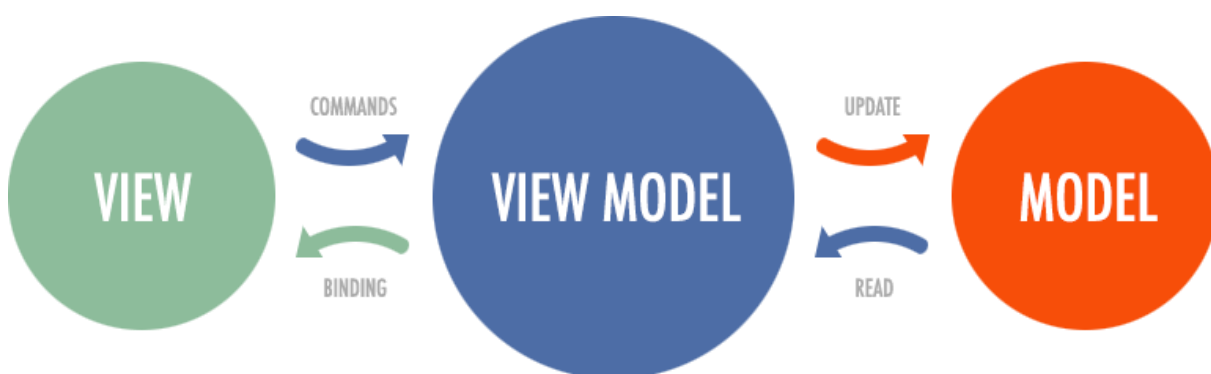


Fig.8 MVVM diagram[13]

Framework-ul Angular funcționează 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 părți de pagina cu modelul, adică iti permite să folosesti HTML ca și template lăsându-te să sa extinzi sintaxa html pentru a exprima componentele aplicației tale cât mai clar și suncint.[15]

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

În ceea ce privește exemplul 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>
```

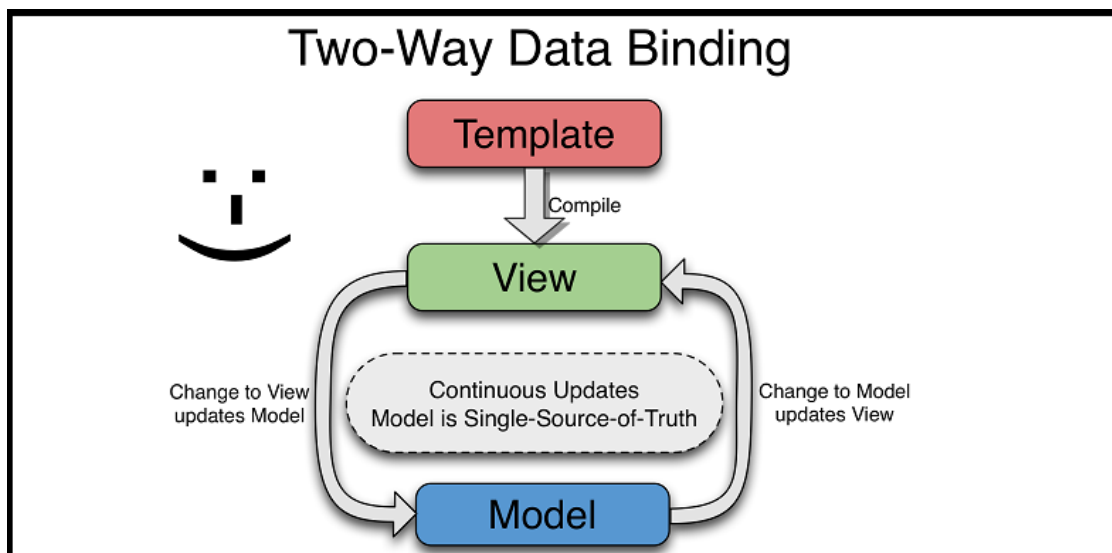


Fig.9 Cum functioneaza AngularJS[14]

De ce sa folosim Angular?

- Usor de înțeles
- Un proiect Google
- Creare de directive (taguri gandite de voi “<om></om>”)
- Documentație detaliată
- Folosit pentru REST
- Destul rapid
- Are la bază MVC
- Prezintă plusul de Dependency 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 puțin cele 2 părți de cod de mai sus, putem observa câteva asemănări ca și în JavaScript, acum codul este reprezentat de o clasă, asta prezintă și primul avantaj TypeScript. Typescript include suport pentru noi caracteristici cum ar fi modulele, clase, constant, interfețe, și funcțiile Lambda care sunt parte din ECMAScript6.[16]

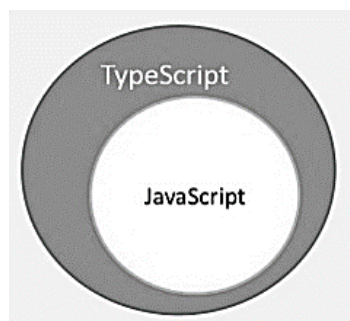


Fig.10 Ce este Typescript/Javascript[17]

În ceea ce privește legătura dintre TypeScript și ECMAScript (specificațiile 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 în ceea ce privește tipurile. 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 Express.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 framework 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, printre caracteristicile din Express se enumeră Middleware(ce reprezintă un pod de comunicare între sistemele software 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 în 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 în 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 și sunt sigură că se vor descurca din prima în a accesa și a interacționa cu aplicația, astfel sănătatea și tehnologia mână în 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 în 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 întru totul.

3.1 Accesul la informații legate de donare

În această parte 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 în 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 cea 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 aplicații pentru utilizatorii din mediul de sănătate, este foarte important fie să reproducem o copie destul de fidelă a ceea ce folosesc dâșii 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 în 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 în 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 de 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 gândit-o pentru ușurarea accesului la serviciul de donare de sânge din Timișoara, fiind destinată atât cetățeniilor însă și cadrelor medicale, deoarece oferă accesul la informație foarte utilă în 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.

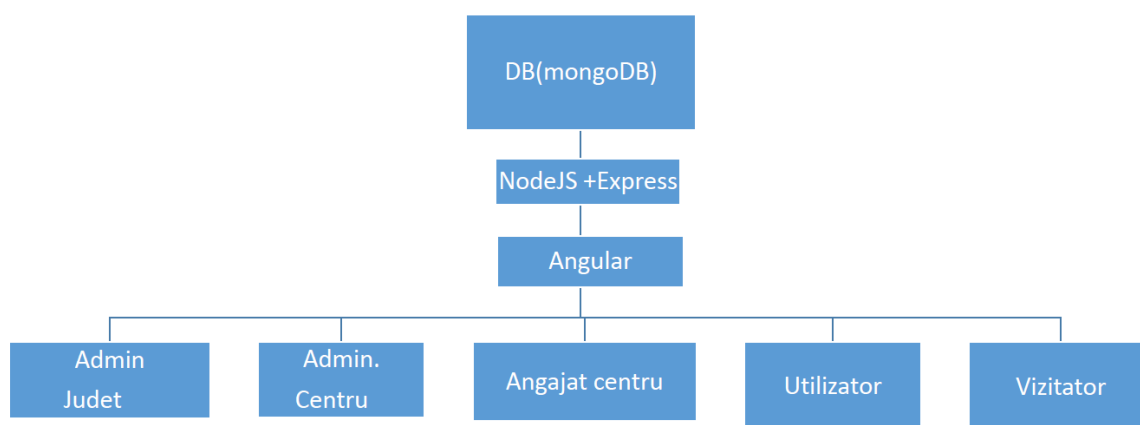


Fig 12. Arhitectura Aplicației

Utilizatori au acces la interfața aplicației prin intermediul căreia pot accesa atât informații generale legate de donare însă pe lângă asta au acces la o hartă a Timișoarei unde sunt reprezentate toate centrele de donare iar în plus de asta fiecare centru de donare reprezentat pe harta are afișate informații legate de nevoia de sânge în 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 în funcție de grupa vizitatorului și nevoile centrului respectiv.

Consider că este una dintre cele mai importante funcționalități datorită faptului că spre deosebire de alte posibilități de informare disponibile pe internet și nu numai oferă informații generale însă nimic concret care să îți ușureze tot acest drum și prețurile de donare, la urma urmei fiecare dintre noi știe cum este cu timpul, iar acest proces doar face din acest gând bun pe care îl 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 și de un meniu ce se afișează 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 în 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 idei personale combinate cu elemente din Angular, partea cea mai interesantă în a folosi Angular este că ai la dispoziție pe lângă multe elemente predefinite ce te ajută să nu mai te gândești la aranjarea în ecran și la tranziții ale elementelor, mai precis unul dintre elementele folosite în aplicație este Sidenav, care împreună cu side-nav-container îți oferă avantaje ca: ajustarea la redimensionarea ecranului, traziția elementului de navigare, fără a scrie eu toate detaliile în CSS, adică fără a avea grija stilizării elementelor.

Un alt avantaj în materie de lucruri vizuale este că poți crea elemente, numite în Angular componente, de exemplu creăm o componentă de tipul meniu în care punem elemente ce le întâlnim peste tot în aplicație, așadar pentru a nu face copiere de cod și să facem aplicație greu de înțeles și mai ales de citit 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 în aplicație utilizatorul este întâmpinat de un ecran cu o coloristică plăcută ochiului și primitoare, dacă discutăm despre acțiunile fiecărui buton, avem astfel:

Ecranul principal oferă un meniu către accesarea hărții, butonul de întoarcere în pagina principal indiferent de unde este utilizatorul în aplicație, mai departe diferă în funcție 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 creării 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 indiferent 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 într-un anumit mod, cu anumite opțiuni și cu o anumită interfață, păstrând tot odată nota generală a aplicației în materie de coloristica și afisare de informații, toate acestea pentru ca utilizatorul să nu defină debusolat, dacă de exemplu are cont de donator dar și de administrator.

Implementarea facilităților este prezentată în codul aplicație, elementele vizuale fiind reprezentate de elemnete din framework-ul Angular și continuând cu apeluri către partea de date, unde se întâmplă prelucrarea datelor și trimiterea de răspunsuri înapoi către interfața utilizatorului, unde vor fi prezentate fie rezultatele în 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.

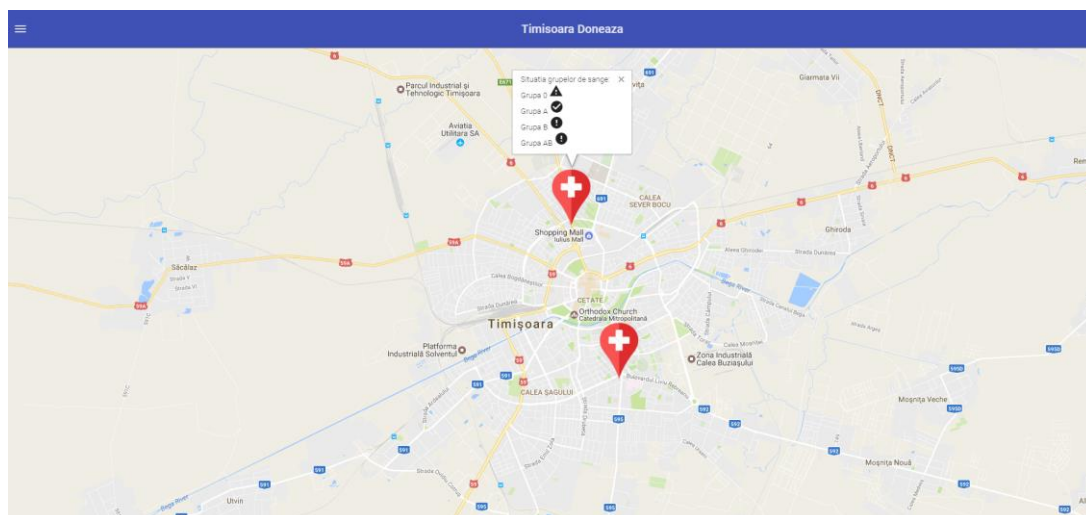


Fig. 13 Interfața opțiunii Harta

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

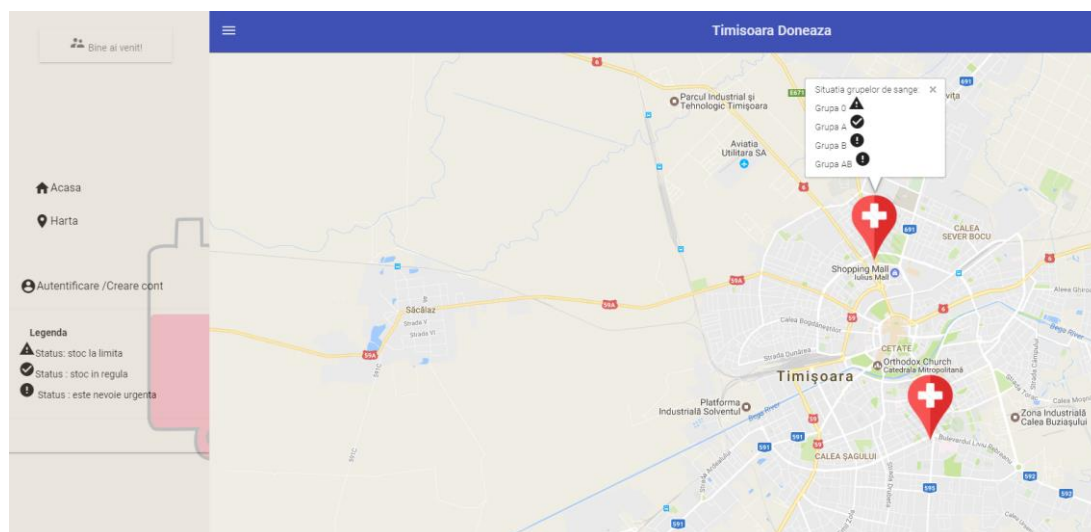


Figura 13.1 Interfata Legendei

Figura 13.1 reprezintă meniul în 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 cea 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 în materie de donare de sânge.

5.2 Accesul în 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 în funcție de cine ești, de exemplu administrator de județ, administrator de centru, angajat, utilizator, vizitator, ești întâmpinat de diferite opțiuni. În continuare voi prezenta și 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 opțiuni, fiind proiectat să folosească la managementul acestui proces de donare la nivel de județ. Mai departe voi prezenta cum arată funcționalitățile și împreună cu explicațiile aferente.



Figura 14 Prezentarea paginii de start



Figura 14.1 Prezentarea paginii se start cu opțiunile meniului

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

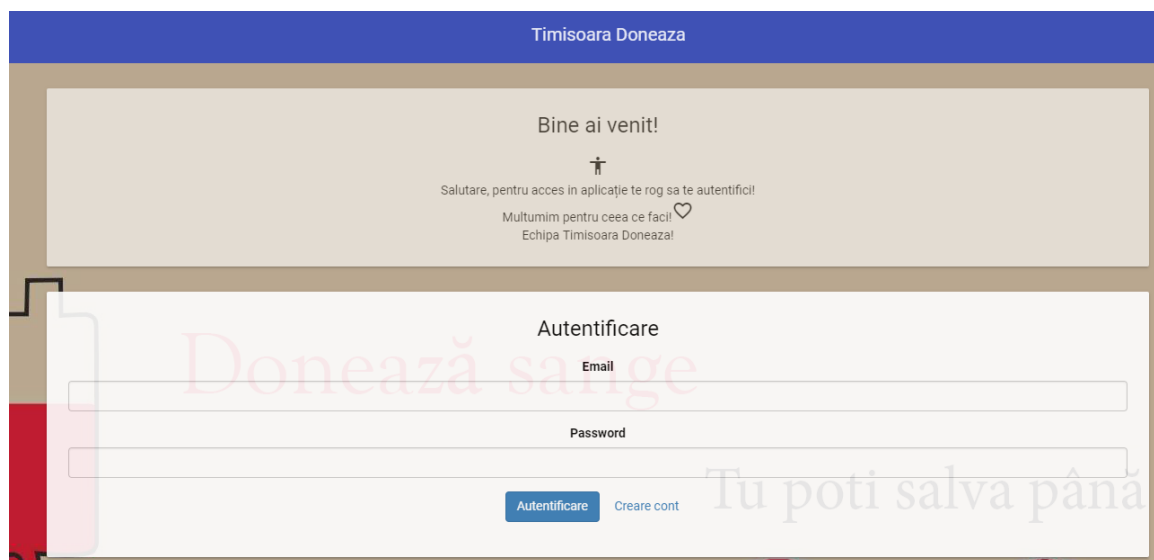


Figura 15 Interfața pentru autentificare

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



Figura 15.1 Interfața meniului după autentificare

➤ Adăugare centru

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

The screenshot shows the 'Adaugare centru' form within the 'Timisoara Doneaza' application. The interface includes a sidebar menu on the left with options like 'Acasa', 'Harta', 'Adauga centru', 'Sterge centru', 'Adauga utilizatori', 'Profil', and 'Dezautentificare'. The main content area features a welcome message, a list of menu items, and a form titled 'Adaugare centru'. The form contains three input fields: 'Numele centrului', 'Adresa centrului' (with a sub-label 'Adresa sa fie de tipul Strada, Oras'), and 'Numar de telefon'. A blue 'Adauga centru' button is positioned below the form. The background of the application features a large graphic of a blood bag and the text 'Donează sânge' and 'Tu poți salva până la 3 vieți'.

Figura 15.2 Interfata de adaugare a unui centru

➤ Ștergere centru

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

The screenshot shows the 'Sterge centru' form within the 'Timisoara Doneaza' application. The interface is similar to the previous one, with the same sidebar menu. The main content area features a welcome message and a form titled 'Sterge centru'. The form contains a single input field labeled 'Numele centrului' and a blue 'Sterge centru' button. The background of the application features a large graphic of a blood bag and the text 'Donează sânge' and 'Tu poți salva până la 3 vieți'.

Figura15.3 Interfata de stergere a unui centru

➤ Adăugare de utilizatori

În această secțiune utilizatorul nostru, Administrator de județ, are opțiunea de a adăuga noi utilizatori la nivel de administrator de județ (Figura 15.4). Sunt necesare introducerea numelui noului utilizatorului, email-ul (pe baza căruia va fi identificat la partea de autentificare) și o parolă inițială pe care utilizatorul o va schimba după prima autentificare în secțiunea profil.



Figura 15.4 Interfața de adăugare de utilizatori la nivel de județ

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 opțiunile gândite să folosească la managementul acestui proces de donare la nivel de centru, în Figura 16 sunt prezentate cum arată funcționalitățile împreună cu explicațiile aferente.



Figura 16 Meniul pentru utilizatorul de tip admin. centru

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

➤ Adăugarea de angajați

Această opțiune prezentată în 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 introducerea numelui noului utilizatorului, email-ul (pe baza căruia va fi identificat la partea de autentificare) și o parolă inițială pe care utilizatorul o va schimba după prima autentificare în secțiunea profil.

The image is a screenshot of a web application interface. At the top, there is a blue header bar with the text 'Timisoara Doneaza'. Below the header, on the left, is a sidebar menu with icons and labels: 'Acasa', 'Harta', 'Adauga angajati', 'Adauga utilizatori', and 'Dezautentificare'. The main content area has a light brown background. At the top of this area, there is a white box with the text 'Bine ai venit!' and a small icon of a person. Below this, there is a message in Romanian: 'In aceasta sectiune ai posibilitatea de a adauga noi noi angajati pentru acest centru. Multumim pentru ceea ce faci! Echipa Timisoara Doneaza!'. In the center, there is a white form titled 'Adaugare angajati centru de donare'. The form has three input fields: 'Numele', 'Email', and 'Parola initiala'. Below the 'Parola initiala' field is a blue button labeled 'Adauga angajat'. At the bottom of the form, there are three circular icons representing people. The background of the main content area also features a large, faint watermark that says 'Donează sânge' and 'Tu poți salva până la 3 vieți'.

Figura 16.1 Opțiunea de adaugare de angați

➤ Adăugarea de utilizatori

Această opțiune permite adăugarea de noi utilizatori pe nivelul de administrator de centru opțiune prezentată în Figura 16.2 și la fel ca în cazul adăugării angajaților în sistem sunt necesare introducerea numelui noului utilizatorului, email-ul (pe baza căruia va fi identificat la partea de autentificare) și o parolă inițială pe care utilizatorul o va schimba după prima autentificare în secțiunea profil.

Bine ai venit!
Administrator centru

Acasa
Harta
Adauga angajati
Adauga utilizatori
Dezautentificare

Bine ai venit!
Salutare,
In aceasta sectiune ai posibilitatea de a adauga noi utilizatori de centru
Multumim pentru ceea ce faci!
Echipa Timisoara Doneaza!

Adaugare utilizatori centru de donare

Numele

Email

Parola initiala

Adauga utilizator

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 informații cât și cu donarea efectivă, de aceea m-am rezumat la câteva opțiuni ce le-am considerat că o să îl ajute în 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 în Figura 17, unde sunt necesare introducerea datelor despre donator, cum ar fi Numele, Prenumele, Vârsta, Grupa de sânge, Adresa și nu în ultimul rând CNP-ul care este identificator unic al fiecărui om.

Bine ai venit!
Angajat centru

Acasa
Harta
Adauga donator
Cauta donator
Dezautentificare

Bine ai venit!
Salutare,
In aceasta sectiune ai posibilitatea de a adauga noi donatori la sange
Multumim pentru ceea ce faci!
Echipa Timisoara Doneaza!

Adaugare donatori

Numele

Prenume

Vârsta

Grupa de sange

Adresa

CNP-ul

Adauga donator

Figura 17 Interfata de adaugare a unui donator

➤ Căutare donator

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

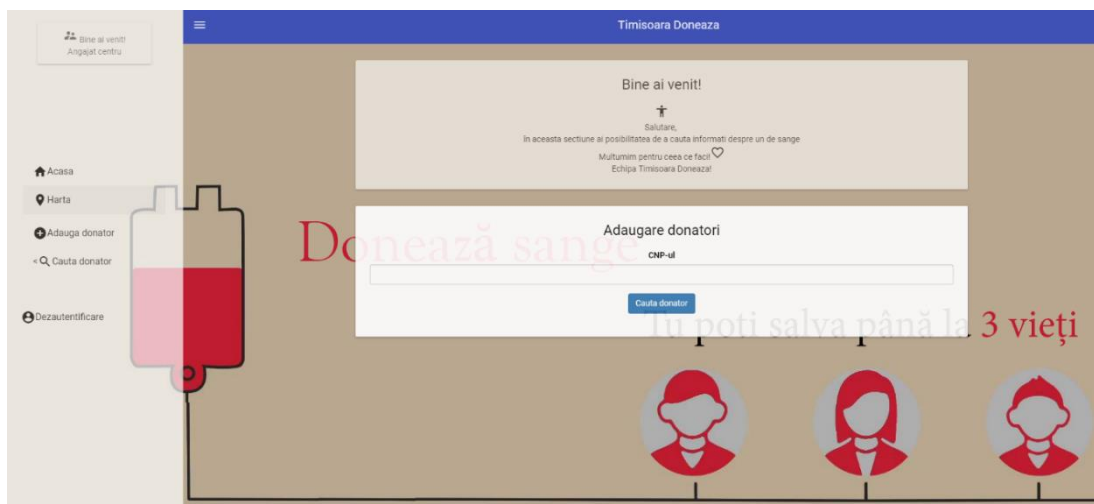


Figura 17.1 Interfata de căutare după donatori

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

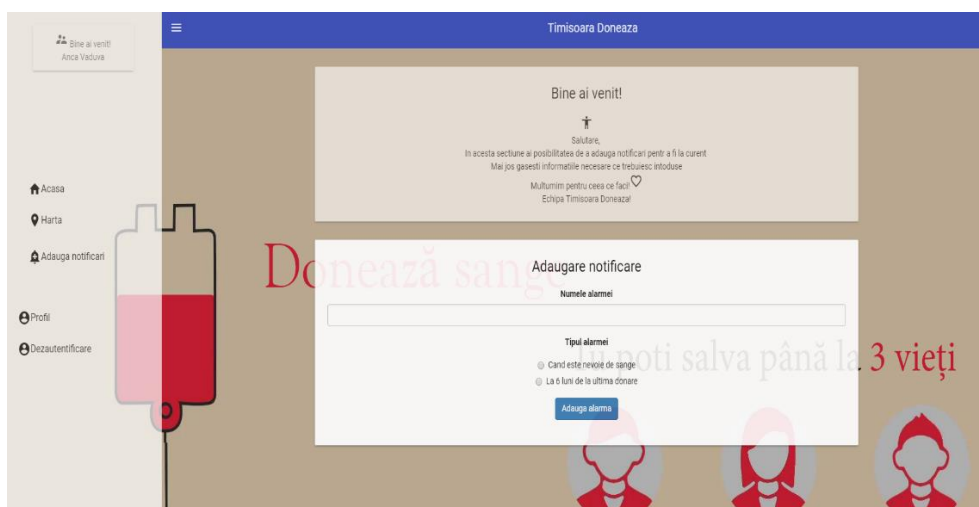


Figura 18.1 Interfața de adăugarea de notificări

În Figura 18.1 este prezentată interfața funcționalității de adăugare a notificărilor care prezintă 2 opțiuni, una dintre ele este ca utilizatorul să fie notificat în cazul în care este nevoie de grupă sa de sânge la un anumit centru, iar cea de-a doua este aceea de a fi notificat când au trecut 6 luni de la ultima donare.

O altă funcționalitate, este aceea de vizualizare a profilului, aici utilizatorul are prezentat tot felul de informații legate de profilul său.

➤ Vizualizare profil



Figura 18.2 Interfața de vizualizare a profilului

În Figura 18.2 este prezentată opțiunea ca utilizatorul să poată modifica anumite informații legate de profilul său de donator în cadrul aplicației, acesta poate introduce date care nu există încă în 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 în Figura 19.



Figura 19 Interfața de creare a unui cont

Toate profilurile, din moment ce sunt accesate mai prezintă și opțiunea de a părăsi aplicația, prin opțiunea de dezautentificare, 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 testată pe module după care de-a lungul creării aplicația a devenit mai mare iară cu modulele funcționale testate în prealabil a rezultat funcționalitatea finală 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 în cazul în care apar. Totodată, testând pe module diferite, nu exista riscul să 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 în 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 situații când au fost organizate, de exemplu, la locul de munca, campanii de donare pentru a ajuta cumva în 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 în 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 în timp real, lipsa de motivare pentru a veni la centru să doneze.

Consider că am atins destul de multe puncte specifice domeniul și mai ales rezolvării problemei în curs, însă întotdeauna este loc de mai bine, iar dezvoltarea aplicației nu va rămâne la nivelul la care este acum, urmând să îl transform în 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 în lucrul și să descopere astfel, dacă nu au descoperit încă, avantajele tehnologiei.

6.2 Direcții de dezvoltare

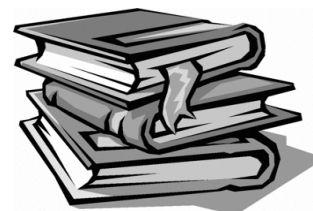
Când mă gândesc la direcțiile de dezvoltare, am în minte următoarele, doresc să integrez mai multe funcționalități atât pentru donatori cât și pentru utilizatori de cadrul medical, una dintre funcționalități pe care o am în plan pentru viitoare direcție de dezvoltare este ca utilizator să își poți face 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 imagine de ansamblu asupra numărului de donatori și de organizarea la nivel de județ.

O funcționalitate pe care o am în 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 în care unul dintre centre este prea aglomerat, conform statisticii.

Așadar sper ca în final să construiesc o aplicație la nivel național pentru a crea o întreaga comunitate legată de donarea de sânge, iar ce este și până acum, în ceea ce privește aplicație, a apărut în 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.

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-general.html>
- [11]. http://stst.elia.pub.ro/news/IS/IS_Teme1101/mvc.pdf
- [12]. <http://mvvm-light.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://mindbrowser.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, 2016
- [21]. http://andrei.clubcisco.ro/cursuri/3ip/curs/6.1_Testarea.doc