



ZEPHYR

APLICATIE DE CHAT

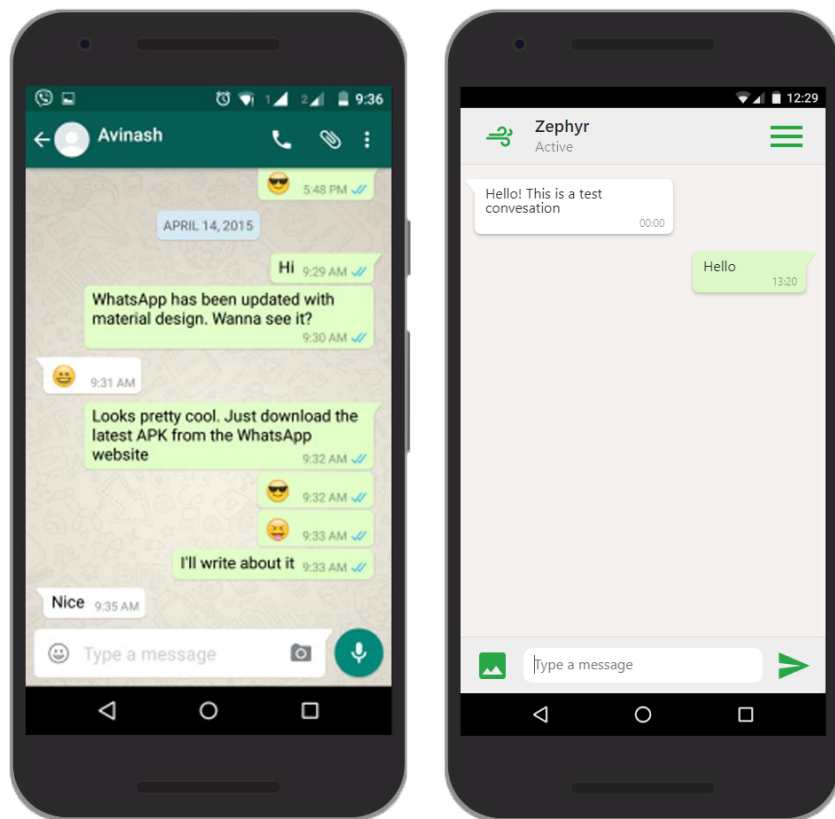
Cuprins

1. Despre Site
2. Limbajele Utilizate
3. Dezvoltarea Aplicatiei
4. Codul Sursa
5. Bibliografie

I.Despre site

Zephyr, este o aplicatie de comunicare online, in mod anonim sau cu ajutorul unui cont. Un avantaj fata de alte siteuri de socializare este faptul ca nu se salveaza nici un mesaj in server. Datele exista doar pe dispozitivele clientilor.

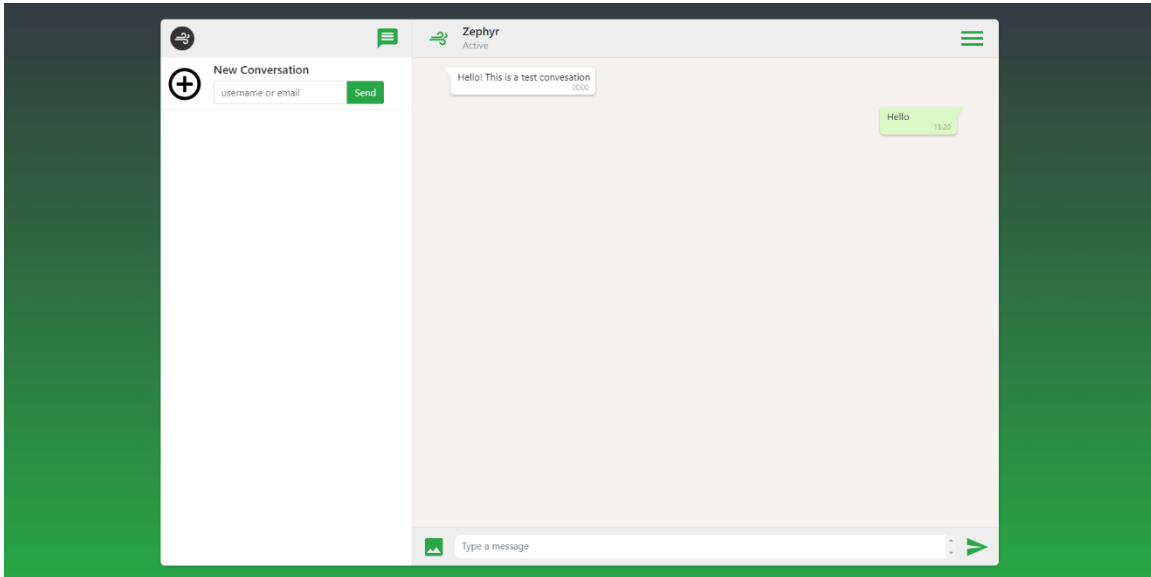
Aceasta aplicatie este inspirata de pe site-uri precum WhatsApp, in special pe partea de design, sau Connected2.me, putand sa comunicati anonim.



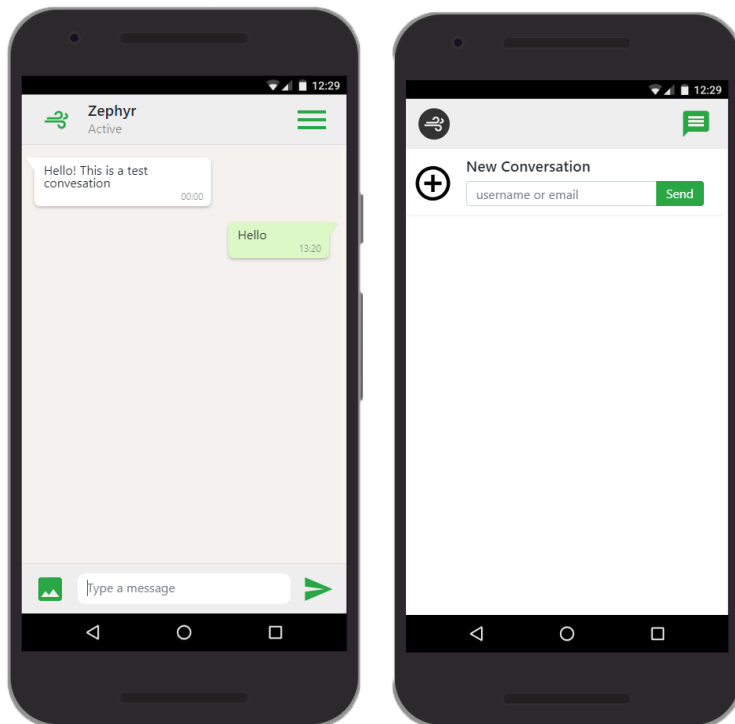
Asemanarea intre WhatApp si Zephyr

Se poate observa interfața simplificată a Zephyr (un buton pentru a încărca poze, caseta de text și buton pentru expedierea mesajului). În momentul de față, mesajele se limitează doar la format text sau imagini. Probabil, în versiuni viitoare, vor putea fi trimise și clipuri video și înregistrări audio.

In timpul dezvoltarii, am dedicat foarte mult timp pentru a face aplicatia usor de utilizat pe orice dispozitiv sau browser. Se va adapta in functie de rezolutie, iar in cazul anumitor browsere, se vor face modificari incat continutul paginii sa arate la fel ca in browserul folosit de dezvoltator, „Google Chrome”.



Cum arata interfata pe „Google Chrome”, rezolutia 1920 x 1080px



Si pe mobil, rezolutia 412 x 732px

Pentru a putea comunica, utilizatorii se pot loga anonim, alegand o „porecla”, sau cu ajutorul contului. Pentru a crea un cont, utilizatorul trebuie sa isi aleaga un nume si o parola, furnizand totodata si adresa sa de email.

In cazul in care utilizatorul isi uita parola, acesta poate cere sa o reseteze, apasand pe butonul „Forgot password?” din fereastra de logare. Va primi un email pe adresa furnizata cand a creat contul, ce contine un link de unde isi poate alege o parola noua.

The diagram illustrates the user flow for password recovery. On the left, the 'Sign in to Zephyr' screen features a logo, a title, and two input fields: 'Username or email address' and 'Password'. A blue 'Sign in' button is at the bottom, and a 'Forgot password?' link is next to the password field. A large black arrow points to the right, where the 'Find your account' screen is shown. This screen has the same logo and title, followed by an 'Email address' input field containing the placeholder 'youradress@somewhere.com'. A blue 'Recover password' button is at the bottom.

Meniul de logare si cel de resetare a parolei

II. Limbaje Utilizate



1. JAVASCRIPT

JavaScript (JS) 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 accesul 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.

Cea mai des întâlnită utilizare a JavaScript este în scriptarea paginilor web. Programatorii web pot îngloba în paginile HTML script-uri pentru diverse activități cum ar fi verificarea datelor introduse de utilizatori sau crearea de meniuri și alte efecte animate.

Browserele rețin în memorie o reprezentare a unei pagini web sub forma unui arbore de obiecte și pun la dispoziție aceste obiecte script-urilor JavaScript, care le pot citi și manipula. Arborele de obiecte poartă numele de Document Object Model sau DOM. Există un standard W3C pentru DOM-ul pe care trebuie să îl pună la dispoziție un browser, ceea ce oferă premiza scrierii de script-uri portabile, care să funcționeze pe toate browserele. În practică, însă, standardul W3C pentru DOM este incomplet implementat. Deși tendința browserelor este de a se alinia standardului W3C, unele din acestea încă prezintă incompatibilități majore, cum este cazul Internet Explorer.

O tehnică de construire a paginilor web tot mai întâlnită în ultimul timp este AJAX, abreviere de la „Asynchronous JavaScript and XML”. Această tehnică constă în executarea de cereri HTTP în fundal, fără a reîncărca toată pagina web, și actualizarea numai

anumitor porțiuni ale paginii prin manipularea DOM-ului paginii. Tehnica AJAX permite construirea unor interfețe web cu timp de răspuns mic, întrucât operația (costisitoare ca timp) de încărcare a unei pagini HTML complete este în mare parte eliminată.



2. JSON

JSON este un acronim în limba engleză pentru JavaScript Object Notation, și este un format de reprezentare și interschimb de date între aplicații informatice. Este un format text, inteligibil pentru oameni, utilizat pentru reprezentarea obiectelor și a altor structuri de date și este folosit în special pentru a transmite date structurate prin rețea, procesul purtând numele de serializare. JSON este alternativa mai simplă, mai facilă decât limbajul XML. Eleganța formatului JSON provine din faptul că este un subset al limbajului JavaScript (ECMA-262 3rd Edition), fiind utilizat alături de acest limbaj. Formatul JSON a fost creat de Douglas Crockford și standardizat prin RFC 4627. Tipul de media pe care trebuie să îl transmită un document JSON este application/json. Extensia fișierelor JSON este .json.

EJS

<% Embedded JavaScript %>

3. EJS

EJS este un limbaj de „template-uri”, care permite generarea de HTML cu ajutorul JavaScript, „server-side”. Sintaxa este foarte asemanatoare cu cea din html, dar permite incorporarea de JavaScript in cod, care este rulat mult mai repede, datorita folosirii Chrome V8.

HTML



4. HTML

HyperText Markup Language (HTML) este un limbaj de marcare utilizat pentru crearea paginilor web ce pot fi afișate într-un browser (sau navigator). Scopul HTML este mai degrabă prezentarea informațiilor – paragrafe, fonturi, tabele ș.a.m.d. – decât descrierea semanticii documentului. Specificațiile HTML sunt dictate de World Wide Web Consortium (W3C).

HTML este o formă de marcare orientată către prezentarea documentelor text pe o singura pagină, utilizând un software de redare specializat, numit agent utilizator HTML, cel mai bun exemplu de astfel de software fiind browserul web. HTML furnizează

mijloacele prin care conținutul unui document poate fi adnotat cu diverse tipuri de metadata și indicații de redare. Indicațiile de redare pot varia de la decorațiuni minore ale textului, cum ar fi specificarea faptului că un anumit cuvânt trebuie subliniat sau că o imagine trebuie introdusă, până la scripturi sofisticate, hărți de imagini și formulare. Metadatele pot include informații despre titlul și autorul documentului, informații structurale despre cum este împărțit documentul în diferite segmente, paragrafe, liste, titluri etc. și informații cruciale care permit ca documentul să poată fi legat de alte documente pentru a forma astfel hyperlink-uri (sau web-ul).

HTML este un format text proiectat pentru a putea fi citit și editat de oameni utilizând un editor de text simplu. Totuși scrierea și modificarea paginilor în acest fel solicită cunoștințe solide de HTML și este consumatoare de timp. Editoarele grafice (de tip WYSIWYG) cum ar fi Macromedia Dreamweaver, Adobe GoLive sau Microsoft FrontPage permit ca paginile web să fie tratate asemănător cu documentele Word, dar cu observația că aceste programe generează un cod HTML care este de multe ori de proastă calitate.

HTML se poate genera direct utilizând tehnologii de codare din partea serverului cum ar fi PHP, JSP sau ASP. Multe aplicații ca sistemele de gestionare a conținutului, wiki-uri și forumuri web generează pagini HTML.

HTML este de asemenea utilizat în e-mail. Majoritatea aplicațiilor de e-mail folosesc un editor HTML încorporat pentru compunerea e-mail-urilor și un motor de prezentare a e-mail-urilor de acest tip. Folosirea e-mail-urilor HTML este un subiect controversat și multe liste de mail le blochează intenționat.



5. CSS

CSS (Cascading Style Sheets) este un standard pentru formatarea elementelor unui document HTML. Stilurile se pot atașa elementelor HTML prin intermediul unor fișiere externe sau în cadrul documentului, prin elementul `<style>` și/sau atributul `style`. CSS se poate utiliza și pentru formatarea elementelor XHTML, XML și SVG.

CSS3 reprezintă un upgrade ce aduce câteva attribute noi și ajută la dezvoltarea noilor concepte în webdesign.

Unele dintre cele mai importante segmente (module) noi adăugate acestui standard pentru formatarea elementelor HTML aduc un plus considerabil în dezvoltarea activității webdesign.

Mai jos sunt prezente în listă cele mai importante modulele adăugate în CSS3:

- Selectors

- Box Model
- Backgrounds and Borders
- Image Values and Replaced Content
- Text Effects
- 2D/3D Transformations
- Animations
- Multiple Column Layout
- User Interface

Deși au apărut unele deficiente de compatibilitate între browsere, majoritatea proprietăților CSS3 au fost implementate cu succes în variantele browserelor noi.



6. JQUERY

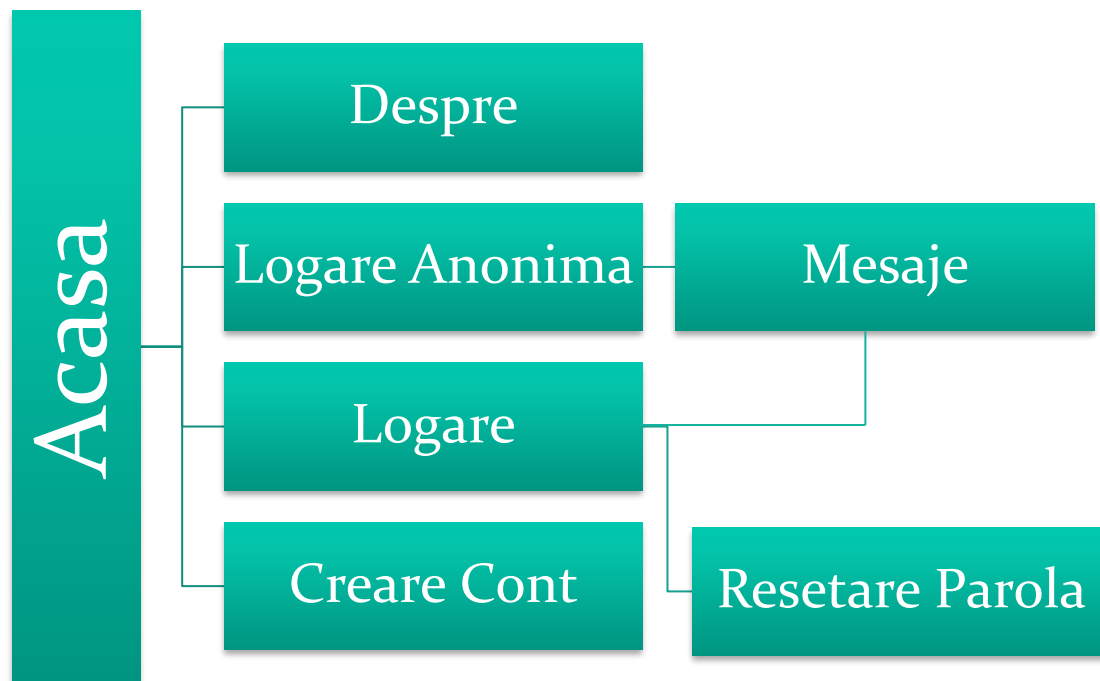
jQuery este o platformă de dezvoltare JavaScript, concepută pentru a ușura și îmbunătăți procese precum traversarea arborelui DOM în HTML, managementul inter-browser al evenimentelor, animații și cereri tip AJAX. jQuery a fost gândit să fie cât mai mic posibil, disponibil în toate versiunile de browsere importante existente, și să respecte filosofia "Unobtrusive JavaScript". Biblioteca a fost lansată în 2006 de către John Resig.

jQuery se poate folosi pentru a rezolva următoarele probleme specifice programării web:

- selecții de elemente în arborele DOM folosind propriul motor de selecții open source Sizzle, un proiect născut din jQuery
- parcurgere și modificarea arborelui DOM (incluzând suport pentru selectori CSS 3 și XPath simpli)
- înregistrarea și modificarea evenimentelor din browser
- manipularea elementelor CSS
- efecte și animații
- cereri tip AJAX
- extensii
- utilități - versiunea browser-ului, funcția „each”

III.Dezvoltarea Aplicatiei

STRUCTURA SITEULUI



Acasa (Home) – pagina principala

Despre (About) – pagina unde se regasesc informatii referitoare la site

Logare Anonima (Log in as Guest) – pagina unde ne alegem o porecla pentru a putea trimite mesaje in mod anonim

Logare (Log in) - pagina de logare

Resetare Parola („Forgot password?”) - pagina unde ne trecem email-ul pentru a putea primit linkul de resetare

Mesaje („/chat”) – pagina esentiala a site-ului. Locul de unde putem trimite si primi mesaje

PROCESUL DE DEZVOLTARE

Acest site a fost scris cu ajutorul editorului de cod „Visual Studio Code”, intrucat acesta suporta o multitudine de limbaje, are o interfata prietenoasa, un terminal integrat si posibilitatea de adauga pluginuri, cum ar fi „Beautify” (un plugin care indenteaza codul si adauga spatii pentru a-l face mai lizibil)

La baza site-ului se afla un server scris in Node.js, un runtime bazat pe „Chrome V8”. „Package managerul” NPM, incorporat in Node, faciliteaza importarea de librarii JavaScript, reducand astfel timpul necesar crearii unui server.

Serverul foloseste o baza de date „MongoDB” pentru a salva informatiile legate de conturi. Fata de alte tipuri de baze de date, MongoDB este mai rapid si mai simplu de folosit, cel mai mare avantaj fiind faptul ca este optimizat sa lucreze impreuna cu servere Node.

Pentru imbunatatirea securitatii, este folosita libraria „bcrypt”, un sistem de criptare care asigura faptul ca, in cazul in care baza de date ar fi „sparta”, parolele nu pot fi citite.

Alte librarii utilizate sunt „express”, ce simplifica sintaxa necesara crearii unui server, „socket.io”, folosita pentru a trimite informatii intre clienti, sau „nodemailer”, folosita pentru a trimite emailuri.

CERINTE DE SISTEM

Serverul a fost dezvoltat si testat pe un sistem cu urmatoarele specificatii:

- Sistem Operare : Windows 10, 64 bits
- Procesor: Intel Core i5-2320 3.00 Ghz
- Memorie: 16 Gb

Totusi, serverul ar trebui sa functioneze si pe versiuni mai vechi de windows, sau pe alte sisteme de operare. Atata timp cat exista cel putin 200Mb liberi in spatiul de stocare.

INSTALAREA SERVERULUI

Inainte de a putea folosi serverul, trebuie urmati cativa pasi:

1. Instalati [Node](#) – recomand ultima versiune stabila

2. Instalati [MongoDB](#). Cu ajutorul MongoDB Compass, creati o baza de date numita „Zephyr”, in care adaugati o colectie „users”. Aceasta tabela trebuie sa aiba urmatoarea structura :

users				
_id ObjectId	username String	email String	password String	resetPasswordToken Null

Iar in fereastra indexes :

<u>_id</u> _id	REGULAR ⓘ	36.0 KB	0 since Mon Apr 30 2018	UNIQUE ⓘ	
email email	REGULAR ⓘ	36.0 KB	1 since Mon Apr 30 2018	UNIQUE ⓘ	🗑
username username	REGULAR ⓘ	36.0 KB	0 since Mon Apr 30 2018	UNIQUE ⓘ	🗑

3. Deschideti folderul ce contine fisierul „Server.js”, si apoi, deschideti acolo o fereastra CMD sau PowerShell (File > Open Windows PowerShell > Open Windows PowerShell – daca utilizati Windows 10)
4. Tastati urmatoarele comenzi
 - "npm install -all" (pentru instalarea modulelor necesare functionarii serverului)
 - "npm install nodemon -g" (pentru a rula serverul mai usor)
5. Daca doriti sa accesati siteul din exteriorul retelei locale, trebuie sa va creati un cont [Ngrok](#), si sa primiti authtoken-ul. Apoi, tastati in fereastra CMD/PowerShell "npm install ngrok -g" (pentru a putea accesa siteul si din afara LAN) , iar apoi tastati `"/ngrok authtoken " + authtoken-ul contului de ngrok.`

SETAREA SERVERULUI

In interiorul folderului „Zephyr\private\settings”, veti gasi un fiser numit general.json. In acest fiser trebuie trecute anumite informatii pentru a asigura functionarea serverului:

serverSettings

- port : portul prin care se vor realiza conexiunile – valoare recomandata : 80

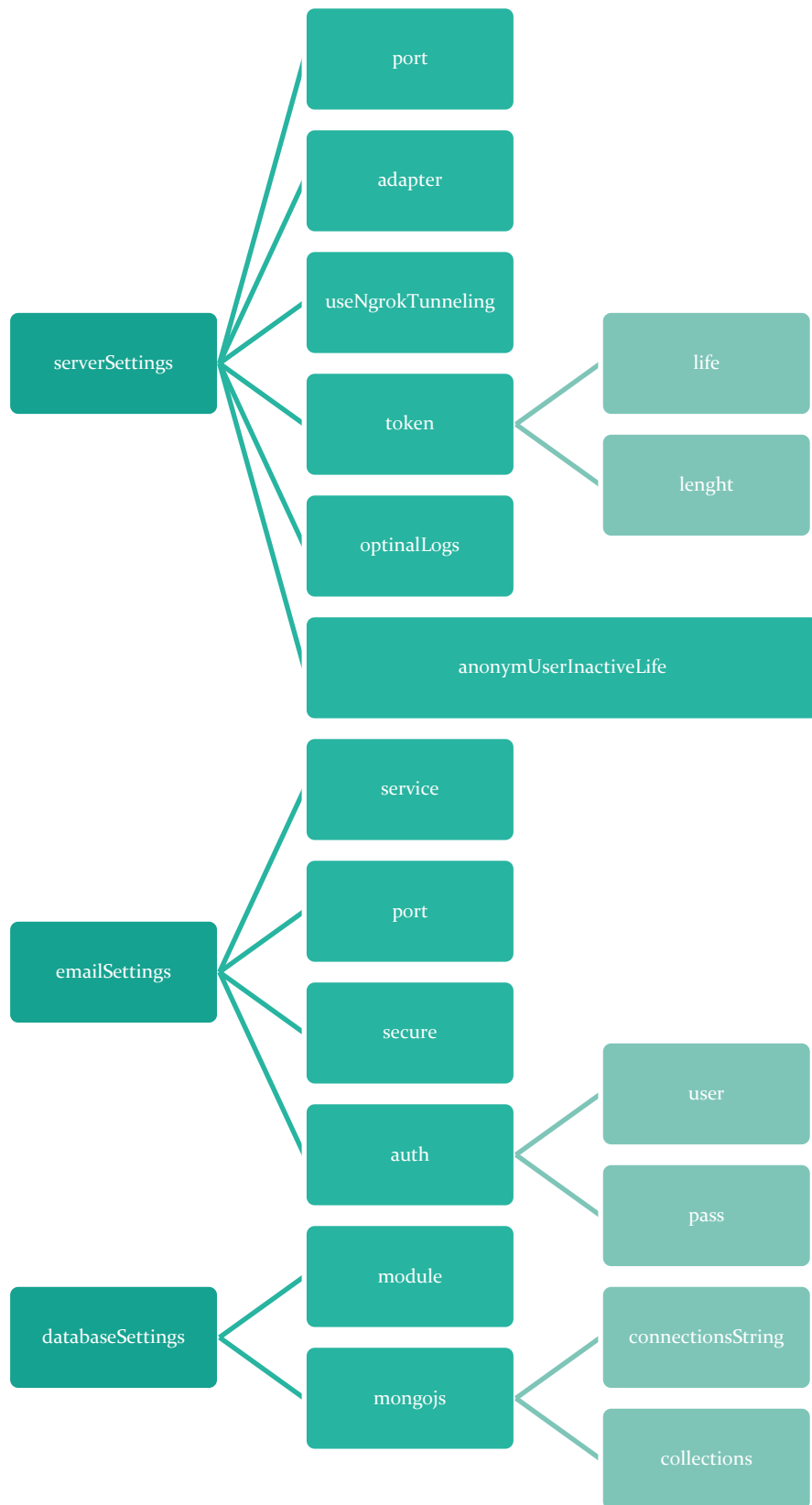
- adapter : adaptorul de retea principal , prin care se va face publica conexiunea – valori posibile: „Ethernet”, „Wi-Fi Adapter”, etc. (navigati la „Control Panel > Network and Internet > Network Connections” pentru a afla numele adaptorului)
- useNgrokTunneling: in cazul in care aveti Ngrok instalat(vedeti pasul nr. 4 al instalarii), aceasta setare activeaza crearea de tuneluri(generarea de URL global)
Valori posibile : „true” sau „false” (daca se/nu se activeaza)
- token
 - life : cat timp sa fie valabil linkul de resetare a parolei (in milisecunde)
 - lenght : cate caractere random sa aiba linkul de resetare a parolei (default – 32)
- optionalLogs : daca doriti sa fie afisate mai multe informatii legate de functionarea serverului setati la valoarea „true”. In caz contrar, valoarea „false”
- anonymUserInactiveLife: cat timp dureaza (in milisecunde) pana un nume de anonym este sters, pentru a putea fi refolosit (default – 60000)

emailSettings

- service : serviciul folosit pentru a trimite emailuri (default : gmail)
- port : portul prin care se vor trimite emailurile (valoare pentru gmail – 465)
- secure : daca conexiunea va fi securizata (true sau false)
- auth
 - user : adresa de email a administratorului (contul folosit pentru a trimite emailuri)
 - pass : parola pentru adresa de email mentionata mai sus

databaseSettings

- module : modulul folosit pentru baza de date (valoare default „mongojs”)
- mongojs
 - connectionString : numele bazei de date (valoare default „Zephyr”)
 - collections : colectia de informatii din baza de date (valoare default „users”)



Dupa ce a-ti realizat aceste setari, serverul poate fi pornit

PORNIREA SERVERULUI

Pentru a porni serverul, deschideti o fereastră, la fel cum ati facut la pasul 2 al instalarii, si tastati una din urmatoarele comenzi:

- „node Server.js”
- „nodemon” – recomandat

Dupa o scurta durata, in fereastră trebuie sa va apara mesajul „Server is now running. Go to [...]”, ce indica faptul ca serverul a pornit fara eroori si este gata sa accepte conexiuni.

OPRIREA SERVERULUI

Pentru a opri serverul, puteti sa inchideti pur si simplu fereastră in care lucreaza sau sa apasati de cateva ori „CTRL + C”.

IV.Codul Sursa

In acest capitol va voi prezenta codul sursa al serverului si alte fisiere mai importante.

SERVER.JS

```
//Headers  
  
var express = require('express');  
var expressValidator = require('express-validator');  
var compression = require('compression');  
var flash = require('express-flash');  
var session = require('express-session');  
var socket = require('socket.io');  
var ngrok = require('ngrok');  
var mongojs = require('mongojs');  
var nodemailer = require('nodemailer');  
var path = require('path');  
var fs = require('fs');  
var sleep = require('system-sleep');  
var bodyParser = require('body-parser');  
var favicon = require('serve-favicon');  
var cookieParser = require('cookie-parser');  
var randomstring = require('randomstring');  
var chalk = require('chalk');  
var bcrypt = require('bcrypt');  
var localip = require('local-ip');
```

Regiunea Headerilor – importarea librariilor


```
//----- Settings & Important variables -----//
//#region Settings and Var's

var config = require(path.join(__dirname,
'/private/settings/general.json'));
var iface = config.serverSettings.adapter;
var serverIp = localip(iface);
var io;
var tunnelUrl;
var app = express();

var port = config.serverSettings.port;
var transporter = nodemailer.createTransport(config.emailSettings);

var texterror = chalk.bold.red;
var textwarning = chalk.keyword('orange');
var textlogo = chalk.green;
var textlogobold = chalk.green.bold;
var textlogoline = chalk.green.underline;
var textplain = chalk.white;

//#endregion
```

Regiunea in care se incarca setarile si se creeaza variabilele globale

Aici se incarca setarile din „general.json”, se detecteaza ip-ul calculatorului pentru a putea rula serverul. De asemenea, se instantieaza „nodemailer”, pentru a putea trimite email-uri.

Middleware

Este un termen destul de vag ce se refera in general la toate nivelurile software intermediare care sprijina comunicatia dintre un client si un server.

(www.roportal.ro)

In cazul aplicatiei mele, folosesc middleware-uri pentru a transforma paginile ejs in html, pentru a trimite fisiere json intre client si server, pentru a trimite resursele statice , etc.

```

//#region Middleware

app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, '/private/views'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({
    extended: false
}));
app.use(compression());
app.use(expressValidator());
app.use('/static', express.static(path.join(__dirname, 'public')));
app.use(favicon(path.join(__dirname, 'public/images/logo/icon.png')));
app.use(cookieParser());
app.use(session({
    secret: 'aeolus',
    resave: true,
    saveUninitialized: true
}));
app.use(flash());
app.use(function (req, res, next) {
    res.locals.errors = null;
    res.locals.success = null;
    res.locals.url = null;
    next();
});
app.use(expressValidator({
    errorFormatter: function (param, msg, value) {
        var namespace = param.split('.'),
            root = namespace.shift(),
            formParam = root;

        while (namespace.length) {
            formParam += '[' + namespace.shift() + ']'
        }
        return {
            param: formParam,
            msg: msg,
            value: value
        };
    }
}));
//#endregion

```

Regiunea pentru middleware-uri

Rute

În momentul în care un client accesează o adresă (de ex. <http://127.0.0.1/about> - dacă serverul se află pe același pc ca și clientul), el va accesa ruta „about” din server, folosind funcția get. În server, se va declanșa un eveniment, apoi apelând o funcție cu ajutorul căreia răspunde cererii clientului.

În cazul rutei „get” pentru index (sau home), „/”, serverul îi va trimite clientului care accesează această ruta pagina „index”, împreună cu variabila „url” (ce va fi folosită de client pentru a crea QR Code-ul).

Pentru ruta „change/: token” (linkul trimis utilizatorului pentru a reseta parola), serverul verifică existența tokenului, și dacă acesta este găsit în baza de date și nu este expirat, îi va răspunde clientului cu pagina html corespunzătoare. În caz contrar, îl va trimite la pagina de unde se cere resetarea parolei („/recover”). Sintaxa cu care denumim ruta este schimbată întrucât linkul pe care user-ul l-a primit este de forma `url_site/change/string_random`. Serverul va transfera tot ce este scris după „change/” (adică stringul random) într-o variabilă denumită token, pentru a putea să o caute în baza de date.

În cazul celorlalte rute „get”, răspunsul va fi doar o pagina html.

Un alt tip de ruta este „POST”. În general, aceasta este folosită atunci când clientul trimite informații serverului. În cazul acestei aplicații, este folosită la logare, la crearea contului, la cererea de resetare și la resetarea propriu-zisă.

1. Signup – serverul verifică dacă datele sunt valide (parola mai lungă de 6 caractere, adresa de email validă, adresa de email și username unice). Dacă datele sunt valide, se criptează parola folosind metoda „Hash and Salt”, cu ajutorul bibliotecii „bcrypt”. Apoi, se introduce în baza de date adresa de email, username-ul și parola criptată. Dacă nu au existat erori, clientul este redirectionat spre pagina de home.
2. Signin – serverul verifică dacă datele sunt valide, și apoi, dacă ele există în baza de date. Dacă ambele condiții sunt îndeplinite, clientul este redirectionat spre pagina de chat.
3. Recover – serverul verifică dacă adresa de email trimisă de client este validă. Dacă da, serverul va genera linkul de recuperare și va trimite email-ul. Chiar și în cazul în care adresa de email nu există, va fi afișat mesajul „An email was sent to (adresa de email scrisă) with further instructions.”.

```

//----- ROUTING -----//

//#region GET routes
app.get('/', function (req, res) {
    optionalLog(req.connection.remoteAddress + " - accessed /home",
"warning");
    optionalLog('');
    optionalLog('User connected : IP address = ' +
req.connection.remoteAddress, "logo");
    res.render('index', {
        url: tunnelUrl
    });
});

[ ... ] - restul rutelor de get servesc in mare parte pagini html

app.get('/change/:token', function (req, res) {
    optionalLog(req.connection.remoteAddress + " - accessed /change with
token " + req.params.token, "warning");
    optionalLog('');
    db.users.findOne({ // Validates the token
        resetPasswordToken: req.params.token, // Checks if the token exist
for the user
        resetPasswordDate: {
            $gt: Date.now() // Checks if the token has expired
        }
    }, function (err, user) {
        if (!user) { // If no user was found
            req.flash('error', 'Password reset token is invalid or has
expired.');
```

return res.redirect('/recover');

```

        } else {
            res.render('change'); // Else shows the page to change the
password
        }
    });
});

//#endregion

```

Rutele „GET”

```

//#region SignUp POST
app.post('/signup', function (req, res) {
  var pass = req.body.password.length;
  var vErrors = [];
  if (pass < 6) {
    vErrors.push("password");
  }
  [ ... ]
  var salt = bcrypt.genSaltSync(10);
  var hash = bcrypt.hashSync(req.body.password, salt);

  [ ... ]
});
//#endregion
//#region SignIn POST
app.post('/signin', function (req, res) {
  db.users.findOne({
    $or: [{
      username: req.body.username
    }, {
      email: req.body.username
    }]
  }, function (err, result) {
    if (err) {
      optionalLog(err, "error");
      return res.status(500).render('500');
    }
    if (result) {
      if (!bcrypt.compareSync(req.body.password, result.password)) {
        return res.render('signin', {
          errors: "general"
        });
      }
      res.redirect('chat');
    } else {
      res.render('signin', {
        errors: "general"
      });
    }
  });
});
//#endregion

```

Rutele POST pentru „signup” si „signin”

```

//#region Recover POST (where you enter the email adress to get the reset
link)
app.post('/recover', function (req, res) {
  db.users.findOne({
    email: req.body.email
  }, function (err, result) {
    if (err) {
      optionalLog(err, "error");
      return res.status(500).render('500');
    }

    [ ... ] - o bucata de cod destul de mare
  });
});
//#endregion

```

Ruta POST pentru „recover”

4. Change/:token : Serverul va verifica daca parola corespunde cu cea trecuta in campul de confirmare a parolei si va verifica daca are cel putin 6 caractere. Daca ambele conditii sunt indeplinite, va cripta parola noua si o va stoca in baza de date. Apoi, va trimite un email de confirmare utilizatorului.

```

//#region Change POST (where you change the password)
app.post('/change/:token', function (req, res) {
  [ ... ] - o bucata de cod destul de mare
});
//#endregion

```

Ruta POST pentru „change/:token”

```

//#region ERROR Status routes
app.use(function (req, res) {
    res.status(404);
    res.render('404');
});

app.use(function (req, res) {
    res.status(500);
    res.render('500');
});
//#endregion

```

Rutele pentru eroorile 404 si 500

Acestea nu sunt chiar rute propriu-zise, dar se comporta ca unele. Cu ajutorul acestor doua functii specificam cum trebuie sa raspunda serverul in cazul unei eroori 500 („Internal server error”) sau 404 („Page not found”).

```

//----- HELPER FUNCTIONS -----//
function optionalLog(string, type) {
    if (config.serverSettings.optionalLogs) {
        if (type == null) {
            console.log(string);
        } else if (type == "error") {
            console.log(texterror(string));
        } else if (type == "warning") {
            console.log(textwarning(string));
        } else if (type == "logo") {
            console.log(textlogo(string));
        }
    }
}

```

Functii de „ajutor”

Aici intalnim functia cu ajutorul careia putem sa afisam informatii suplimentare in CMD sau PowerShell (terminalul in care lucreaza serverul). Ea poate determina tipul mesajului, de eroare, avertizare sau mesaj colorat (in culoarea logo-ului).

```

//#region Server
console.clear();
console.log(textlogo("-----
-----"));
console.log(textlogobold(" Zephyr Server"));
console.log("");
console.log(textplain(" Grama Nicolae - 2018"));
console.log(textlogo("-----
-----"));
console.log("");
sleep(1000);
console.time("Server started in "); //starting to count time it took to
start the server
console.log(textwarning("Starting Server ..."));

var server = app.listen(port, function () {

    console.log("");
    if(port!=80){
        console.log(textwarning('Server is now running. Go to localhost:'
+ port + ' on your browser'));
    }
    else console.log(textwarning('Server is now running. Go to localhost
on your browser'));
    console.timeEnd('Server started in ');
    console.log("");

    if (config.serverSettings.useNgrokTunneling) {
        ngrok.connect(port, function (err, url) {
            tunnelUrl = url;
            console.log(textwarning('Ngrok started on port ' + port));
            console.log('To access the webpage from outside, go to ' +
chalk.green.underline.bold(url));
            console.log("");
            console.log(textlogo("-----
-----"));
            console.log("");
        });
    } else {
        tunnelUrl = "http://" + serverIp + ":" +
config.serverSettings.port.toString();
    }
});
//#endregion

```

Codul de pornire al serverului

In aceasta sectiune din cod sunt scrise instructiunile legate de pornirea serverului : mesajele care trebuie afisate la pornirea lui, initializarea serverului si tunelarea conexiuni.

Tunele

Reprezinta o metoda prin care conexiunea locala, serverul deschis pe un port, cu un ip local poate fi legat de un URL. Altfel spus, putem sa vedem serverul local de oriunde in lume, fara sa avem nevoie de vre-un domeniu. Cum am spus in pasul al 5-lea al instalarii serverului, acest tip de conexiune este posibil folosind Ngrok.

```
//#region Databases

var db = mongojs('Zephyr', ['users']);

db.users.find(function (err, docs) {
  if (docs) {
    console.log(textlogo('Connected to the local MongoDB'));
  } else console.log(texterror("Couldn't connect to local MongoDB"));
});

//#endregion
```

Initializarea si conectarea la baza de date

In aceasta ultima parte a codului, se creeaza un obiect baza de date, cu ajutorul libreriei mongojs. Se specifica „connection string-ul” (sau numele bazei de date) si „colectia” (sau tabela in care se afla informatiile). Apoi, aplicatia se conecteaza la baza de date.

Codul a mai suferit modificari de cand a fost scrisa aceasta documentatie, fiindu-i adaugate cookie-uri pentru a face logarea, algoritmi pentru trimiterea de mesaje si managementul userilor activi.

La fel se intampla si in urmatorul fisier prezentat, caruia i-au fost adugate funtiile pentru trimiterea si primirea de mesaje.

CHAT.JS

Acesta este codul javascript, public, folosit in cadrul paginii de chat. Multe bucati din cod sunt repetitive, asa ca nu voi afisa aici tot codul.

```
$('#userTab').on('click', function () {
    //evenimentul pentru click pe tab cu user
    console.log("userTabClick");
});

function add_firstUserMessage(message, time) {
    if (!time) {
        time = getTime();
    }

    $("#chatWindow").append('<div id="messageSpace"
class="space"></div>');
    $("#chatWindow").append('<div class="animated bounceInRight"
id="userMessageWrapper"><div id="userMessage"><div id="messageText">' +
message + '</div><div id="messageTime">' + time + '</div></div><div
id="userCorner"></div></div>');
    $("#chatWindow").scrollTop($("#chatWindow")[0].scrollHeight);
}
[...]
```

Bucata din cod responsabila de afisarea de mesaje

In aceasta parte a codului au fost implementate peste 12 functii, care afiseaza diferite tipuri de mesaje: primul mesaj al cuiva (avand stilul un pic diferit fata de un mesaj normal), mesaje normale, poze.

- add_friendMessage("ceva"); - mesaj normal de la cealalta persoana
- add_firstFriendMessage("ceva"); - primul mesaj
- add_friendPictureUrl("http://localhost/static/images/logo/darklogo.png"); - poza (daca accesati pagina de pe alt calculator decat cel pe care ruleaza serverul, inlocuiti „localhost” cu URL-ul tunelului sau adresa IP a serverului)
- add_firstFriendPictureUrl("http://localhost/static/images/logo/darklogo.png"); - prima poza de la prieten. (aceleasi specificari ca mai sus)

Restul codului este si mai complicat. Exista functii folosite pentru a anima anumite elemente, a adauga useri (a incepe o conversatie), a incarca o anumita conversatie (neimplementata in totalitate – va trebui adaugat un manager al informatiilor, o metoda astfel incat browserul sa salveze mesajele), a tine cont cine a trimis ultimul un mesaj, pentru a stii cum trebuie afisat urmatorul.

```
function add_friendUser() {  
    var userName = $('#searchText').val();  
    if (userName) {  
        //check if user exist  
        addUserTab(userName);  
        $('#searchText').val('');  
    } else {  
        addFriendError("empty search text");  
    }  
}
```

Functia pentru a „deschide” o conversatie noua

In urmatoarea imagine putem observa o functie cu ajutorul careia putem afla ora exacta, si sa o transforma intr-un format ce poate fi utilizat pentru a specifica cand a fost trimis mesajul.

Urmatoarele doua functii, jQuery, sunt folosite pentru a schimba intre fereastra cu Useri (conversatii) si fereastra cu mesaje. Acest lucru este necesar intrucat pe ecranele telefoanelor mobile nu incap ambele ferestre simultan.

```

function getTime() {
    var d = new Date();
    if (d.getMinutes() < 10) {
        time = d.getHours() + ":0" + d.getMinutes();
    } else {
        time = d.getHours() + ":" + d.getMinutes();
    }
    return time;
}

$("#chatMenuButton").click(function () {
    $("#main").removeClass("leftMenu");
    $("#main").addClass("rightMenu");
});

$("#mainMenuButton").click(function () {
    $("#main").removeClass("rightMenu");
    $("#main").addClass("leftMenu");
});

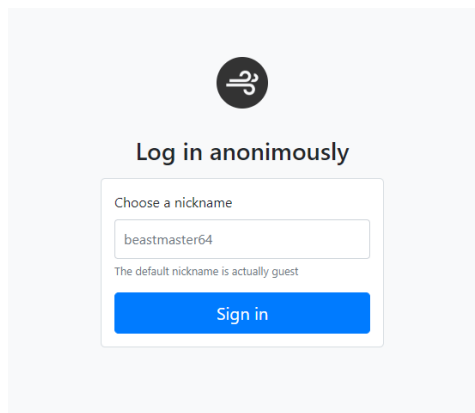
```

Functii ca sa aflam timpul si sa schimbam taburile pe mobil

GUESTLOG.EJS

Aceasta este pagina afisata atunci cand dam click in pagina principala pe butonul „Log in as Guest”.

Chiar daca extensia fisierului este .ejs, se poate observa faptul ca se aseamana foarte mult cu paginile html. In cazul acestei pagini, chiar este identic, intrucat nu am avut nevoie sa folosesc javascript. Se poate observa utilizarea intensiva a elementelor bootstrap, cum ar fi „container”, „mx-auto d-block”, etc., dar si a unor elemente ce au fost stilizate cu ajutorul unui fiser css scris de mine.



```

<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Zephyr</title>
  <link rel="icon" href="static/images/logo/icon.png" type="image/x-
icon" />
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

<body class="bg-light">
  <div class="container" id="containerFull">
    <div class="row" id="containerFull">
      <div class="col align-self-center" id="signinForm">
        
        <p class="h3 text-center" style="margin-bottom: 16px">Log
in anonymously</p>
        <form class="form-signin">
          <div class="form-group">
            <label for="nicknameInput">Choose a
nickname</label>
            <input type="text" class="form-control"
id="nicknameInput" aria-describedby="nicknameHelp"
placeholder="beastmaster64">
            <small id="nicknameHelp" class="form-text text-
muted">The default nickname is actually guest</small>
          </div>
          <a class="btn btn-primary btn-lg btn-block"
href="chat">Sign in</a>
        </form>
      </div>
    </div>
  </div>
</body>
[... Importarea de librarii ...]
</html>

```

Codul paginii „GUESTLOG.EJS”

INDEX.CSS

Acesta este css-ul paginii principale. Am preferat sa il folosesc pe acesta si nu pe cel al paginii de chat, intrucat celalalt este foarte complicat, fiind minimizat (i-au fost scoase toate spatiile inutile, pentru a salva din dimensiunea fisierului, si implicit, viteza paginii).

```
[alt="Scan me!"] {
  padding-top: 8px;
  padding-bottom: 8px;
  margin-left: auto;
  margin-right: auto;
  display: block;
}
.btn-primary {
  background-color: #0068E6 !important;
  border-color: #0068E6 !important;
}
.btn-success {
  background-color: #208836 !important;
  border-color: #208836 !important;
}
.cardHr {
  margin: 0px;
}
.navigation-item {
  margin-left: 4px;
  margin-right: 4px;
}
#containerFull {
  height: calc(100vh - 60px) !important;
}
#footer {
  position: absolute;
  right: 0;
  bottom: 0;
  left: 0;
  text-align: center;
}
@media screen and (max-width: 767px) {
  #footer {
    display: none;
  }
}
```

Rolul acestui fisier :

- stilizarea QRCode-ului
- schimbarea culorilor butonelor bootstrap
- eliminarea marginilor elementului cu id-ul „cardHr”
- adaugarea unor margini pentru elementele din navbar
- setarea inaltimii paginii, pentru a ocupa tot spatiul posibil
- adaugarea unui footer cu pozitie absoluta
- eliminarea footer-ului, pentru o rezolutie mai mica de 767px

V.Bibliografie

- <https://ro.wikipedia.org/wiki/JavaScript>
- [https://ro.wikipedia.org/wiki/Cascading Style Sheets](https://ro.wikipedia.org/wiki/Cascading_Style_Sheets)
- [https://ro.wikipedia.org/wiki/HyperText Markup Language](https://ro.wikipedia.org/wiki/HyperText_Markup_Language)
- <https://ro.wikipedia.org/wiki/JSON>
- <https://nodejs.org/en/>
- <http://ejs.co/>
- [https://www.roportal.ro/articole/despre/servere si protocoale 91/](https://www.roportal.ro/articole/despre/servere_si_protocoale_91/)
- <https://ro.wikipedia.org/wiki/JQuery>