VISOKA TEHNIČKA ŠKOLA
STRUKOVNIH STUDIJA

Node.JS i Express asinhrona chat aplikacija

Predmet: Klijent Server Sistemi

Profesor:

Dr Mirko Kosanović

Miloš Kosanović

Student:

Goran Đukić

REr 38/17

07.01.2020.

SADRŽAJ

1. Uvod

U ovom projektu je obrađena izrada web aplikacije za asinhrono chat-ovanje i deljenje fajlova preko interneta. Tehnologije koje su korišćene na klijentskoj strani su HTML5, CSS sa Bootstrap framework-om i JavaScript sa jQuery bibliotekom, dok su na serverskoj strani korišćeni NodeJS sa Express framework-om. Za komunikaciju u realnom vremenu (eng. Realtime Communication) korišćena je JavaScript biblioteka Socket.IO koja je slična WebSocket protokolu. Za autentikaciju korisnika korišćen je Passport, a za komunikaciju sa MongoDB bazom podataka biblioteka Mongoose. Alati koji su korišćeni prilikom izrade aplikacije su Google Chrome i Safari pretraživači uz Visual Studio Code kao text editor i Robo 3T kao navigator baze.

Aplikacija se sastoji iz dva dela. Prvi deo sadrži formu za login i registrovanje, kao i dugmeta za ulaz u aplikaciju pomoću Facebook ili Google naloga. Drugi deo aplikacije, odnosno glavni deo, sadrži dugme za logout, podešavanja gde biramo do kog datuma želimo da nam aplikacija učita poruke, uz opciono brisanje poruka starijih od tog datuma, i prostor za dopisivanje, koji prilikom login-a automatski učitava poslednje dve nedelje poruka iz baze. Kad god se novi korisnik konektuje na aplikaciju, ostalim korisnicima će se prikazati poruka da je novi korisnik konektovan. Dok neki korisnik kuca poruku, ostali korisnici su obavešteni. U bilo kom trenutku je moguće upload-ovati i podeliti neki fajl sa drugim korisnicima putem običnog drag-and-dropa.

2. Instalacija i podešavanje projekta

Da bismo pokrenuli Node.JS aplikaciju, potrebno je da instaliramo Node.JS, a za ovaj projekat koristimo i MongoDB, tako da je potrebno i njega instalirati.

2.1. Instaliranje modula

U ovom projektu nalazi se **package.json** fajl koji se inicijalizuje sa projektom, naravno na zahtev programera, i ukoliko popunimo sve informacije ispravno biće kreiran fajl. Zatim kad krenemo da instaliramo nove module i ukoliko upotrebimo neke ključne reči (-S , -save) prilikom instalacije u ovom fajlu biće nam upisani svi moduli koje koristimo za našu aplikaciju. Ukoliko je sve to ispravno kreirano, da bismo na nekom drugom računaru pokrenuli i instalirali module, potreno je ukucati sledeću naredbu: **npm install.** Ova naredba prvo

pretražuje **package.json** fajl i u njemu traži i instalira sve dependence (tj. module ili biblioteke) koje su potrebne za ovaj projekat.

3. Arhitektura aplikacije

Aplikacija u sebi sadrži korenski (engl. Root) direktorijum '/' koji u sebi sadrži **index.js** Node.JS izvršni fajl, **db.js** fajl za komunikaciju sa MongoDB bazom podataka, **package.json** fajl koji specificira neophodne module za pokretanje aplikacije, '**/node_modules**' direktorijum koji sadrži instalirane module, '**/models**' direktorijum u kome su smešteni modeli pri komunikaciji sa bazom, '**/user**' direktorijum u kome je smešten kontroler za autorizaciju korisnika, '**/views**' direktorijum u kome su smešteni templeti koji su prikazani korisniku, '**/public**' direktorijum u kome je smešten klijentski deo aplikacije + svi fajlovi koje korisnici upload-uju. Svakoj datoteci iz '/public' direktorijuma je moguće pristupiti javno, odnosno direktno iz URL-a veb čitača. '/public' direktorijum se sastoji iz '**/js**' direktorijuma gde se nalazi logika za prikazivanje chat-a na klijentskoj strani, kao i datepicker biblioteka koju koristimo za formatirani prikaz vremena, '**/css**' direktorijuma koji sadrži CSS fajlove za stilizovanje raznih delova aplikacije, '**/images**' direktorijuma koji sadrži statičke slike korišćene na frontend-u, i '**/files**' direktorijuma gde se skladište fajlovi upload-ovani od strane korisnika.

3.1. Serverski deo

Node.JS - višeplatformsko JavaScript radno okruženje otvorenog koda za izvršavanje JS skripta na serverskoj strani. Osnova za ovaj projekat.

Express.js - Node.JS framework za razvoj veb aplikacija otvorenog koda. De fakto standardni serverski framework za Node.JS.

Mongoose - Object Data Modeling (ODM) biblioteka koja spaja Node.JS sa MongoDB.

Passport - autentikacioni posredni softver (engl. middleware) za Node.JS.

Socket.IO - biblioteka koja nam omogućava asinhronu komunikaciju između servera i većeg broja klijenata; Node.JS serverski deo.

express-session - biblioteka koja nam omogućava da kreiramo sesije za svakog korisnika.

express-socket.io-session - biblioteka koja deli express-session middleware tj. cookie sa bibliotekom Socket.IO

Formidable - biblioteka za parsiranje podataka forme kod upload-ovanja fajlova.

3.2. Klijentski deo

HTML - HyperText Markup Language - standardni markap jezik kojim opisujemo elemente na stranici.

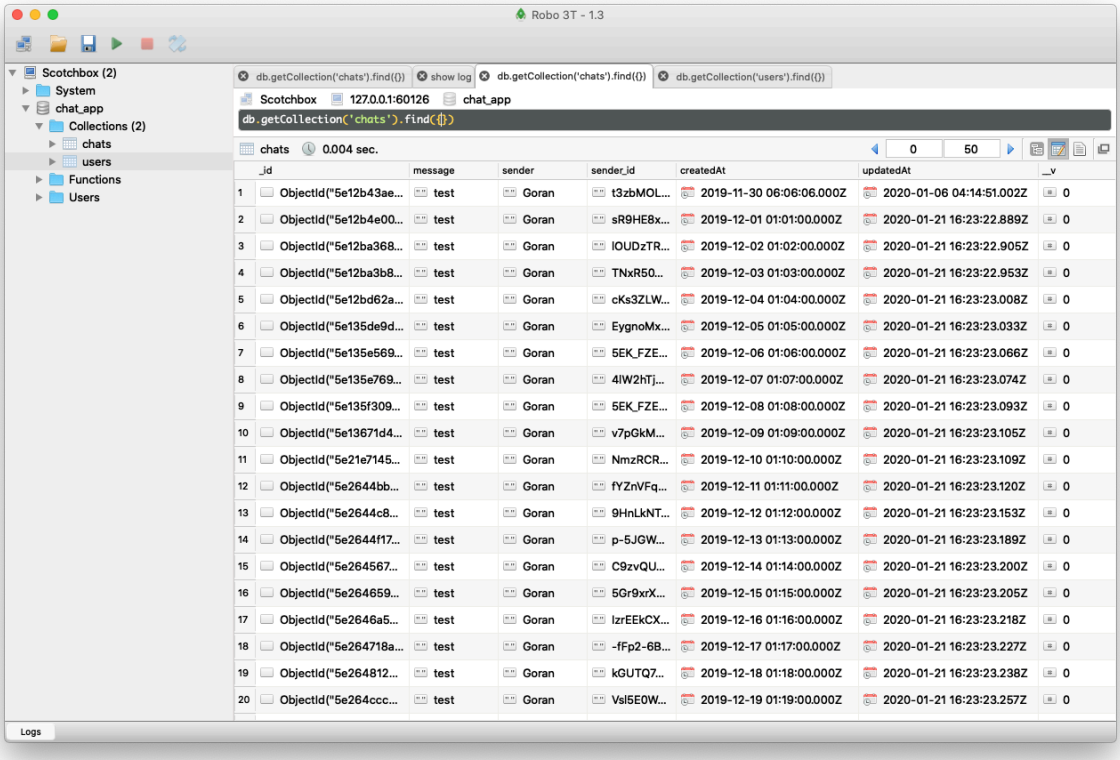
CSS - Cascading Style Sheets - standardni jezik za stilizovanje HTML elemenata.

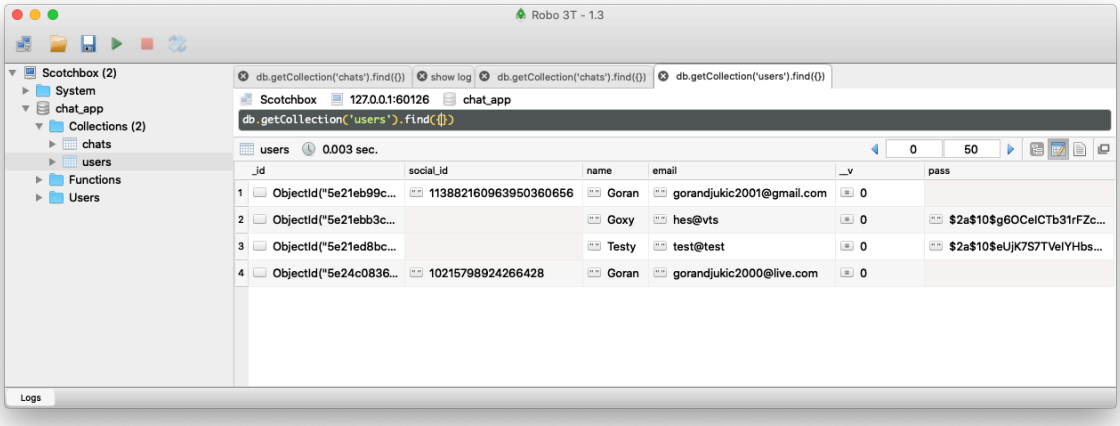
JavaScript - dinamičan, slabo tipiziran, interpretiran programski jezik višeg nivoa.

Socket.IO - JavaScript klijentski deo.

3.3. Baza podataka

MongoDB - višeplatformski dokumentno-orijentisani (NoSQL) jezik za bazu podataka.





3.4. Komunikacija

Rute:

GET '/' - prikazuje login stranicu.

POST '/' - prima formu sa korisničkim podacima, kreira korisnika i sesiju ako je registrovanje, kreira samo sesiju ako je login.

GET '/signup' - prikazuje stranicu za registrovanje.

GET '/logout' - uništava sesiju i vraća nazad na login stranicu.

GET '/chat' - ako postoji sesija, prikazuje chat stranicu.

POST '/date' - ako postoji sesija, prima zadati datum kroz formu i refresh-uje stranicu tako da se prikažu samo poruke nakon tog datuma.

POST '/del' - ako postoji sesija, prima zadati datum kroz formu i briše sve poruke u bazi kreirane pre tog datuma, zatim refresh-uje stranicu da se prikažu sve poruke nakon toga.

POST '/upload' - ako postoji sesija, parsira formu, primljeni fajl smešta u direktorijum '/public/files', zatim to kao poruku emituje ostalim korisnicima i čuva je u bazu.

GET '/auth/facebook/' - poziva Passport middleware koji nas prosleđuje na Facebook sajt, gde se ulogujemo na lični FB nalog i dajemo aplikaciji potrebne podatke.

GET '/auth/facebook/callback' - putanja na koju nas Facebook vraća nakon autentikacije; kreira sesiju sa podacima korisnika.

GET '/auth/google/' - poziva Passport middleware koji nas prosleđuje na Google sajt, gde se ulogujemo na lični Google nalog i dajemo aplikaciji potrebne podatke.

GET '/auth/google/callback' - putanja na koju nas Google vraća nakon autentikacije; kreira sesiju sa podacima korisnika.

4. Rad aplikacije

4.1. Opis implementacije

Rute koje vode ka Facebook-u i Google-u za autentikaciju, kao i njihove callback rute:

```
index.js > ...
132  app.get('/auth/facebook', passport.authenticate("facebook", { scope: ['email'] }));
133  app.get('/auth/facebook/callback', passport.authenticate('facebook', { failureRedirect: '/' }), (req, res) => {
134    CreateSession(req, res);
135  });
136
137  app.get('/auth/google', passport.authenticate('google', { scope: ['profile', 'email'] }));
138  app.get('/auth/google/callback', passport.authenticate('google', { failureRedirect: '/' }), (req, res) => {
139    CreateSession(req, res);
140  });
141
142  function CreateSession(req, res) {
143    req.session.name = req.user.name;
144    req.session.loaddate = moment().subtract(14, 'days').format('YYYY-MM-DD');
145    res.redirect('/chat');
146  }
```

Strategije pomoću kojih se čitaju i čuvaju podaci koje nam vraćaju FB i Goog:

```
user > user.controller.js > ...
17  passport.use(
18      new FacebookStrategy(
19          {
20              clientID: '2550070455269720',
21              clientSecret: '179afa9c4964481048dc34f718985a7c',
22              callbackURL: 'http://localhost:8080/auth/facebook/callback',
23              profileFields: ['id', 'email', 'name']
24          },
25          function(accessToken, refreshToken, profile, done) {
26              let acc = {
27                  social_id: profile.id,
28                  name: profile._json.first_name,
29                  email: profile._json.email
30              }
31              User.findOrCreate(acc, (err, user) => {
32                  if (err) return done(err);
33                  return done(err, user);
34              });
35          }
36      )
37  );
38
39  passport.use(
40      new GoogleStrategy(
41          {
42              clientID: '1002736574284-ofrvfc1i3d9paq9u47uueqth7ob0tanq.apps.googleusercontent.com',
43              clientSecret: 'nSLB0JuiGRuWwQ9k-IyV5-jl',
44              callbackURL: 'http://localhost:8080/auth/google/callback'
45          },
46          function(accessToken, refreshToken, profile, done) {
47              let acc = {
48                  social_id: profile.id,
49                  name: profile._json.given_name,
50                  email: profile._json.email
51              }
52              User.findOrCreate(acc, (err, user) => {
53                  if (err) return done(err);
54                  return done(err, user);
55              });
56          }
57      )
58  );
```

Sa klijentske strane, nakon što se korisnik uloguje, chat stranica obaveštava server da je ulogovana. Server pravi sesiju i vraća poslednjih 14 dana poruka:

```
public > js > chat.js > ...
1  let socket = io.connect('http://localhost:8080');
2
3  socket.emit('username');
```

```
index.js > ...
154  io.on('connection', (socket) => {
155    socket.on('username', async () => {
156      socket.username = socket.handshake.session.name;
157      socket.handshake.session.id = socket.id;
158      await Chat.find({"createdAt" : {$gte : new Date(socket.handshake.session.loaddate)}}).then(async chats => {
159        chats.forEach(async chat => {
160          socket.emit('chat_message', '<strong>' + chat.sender + '</strong>: '
161            + chat.message
162            + ' <i style="float: right">' + moment(chat.createdAt).format('DD/MM/YYYY, LT') + '</i>');
163        });
164      });
165      io.emit('is_online', '<i>'+ socket.username + ' je ušao u chat <3</i>');
166    });
```

Nakon čega taj klijent učitava poruke, a svi ostali klijenti primaju poruku da je novi korisnik ušao u chat:

public > js > JS chat.js > submit() callback

```
socket.on('all_messages', (msg) => {
    msg.forEach(row => {
        $('#messages').append($('<li>').html(row));
    }, (err) => {
        if (err) console.log(err);
    });
});

socket.on('chat_message', (msg) => {
    $('#messages').append($('<li>').html(msg));
});

socket.on('is_online', (username) => {
    $('#messages').append($('<li>').html(username));
    $('#messages').scrollTop(function() { return this.scrollHeight; });
});
```

Dok neki korisnik kuca, ostali su obavešteni. Kada korisnik pritisne submit, ta poruka se emituje svima. Kada se neki korisnik diskonektuje, ta poruka se emituje svima:

public > js > JS chat.js > checkDel

```
$('#chatForm').submit((e) => {
    e.preventDefault();
    socket.emit('chat_message', $('#txt').val());
    $('#txt').val('');
    return false;
});

document.getElementById('txt').oninput = (e) => {
    if (e.target.value == '') socket.emit('stop_typing');
    else socket.emit('is_typing');
}

let typingIndicator = document.getElementById('typing');
    socket.on('is_typing', (msg) => {
    typingIndicator.innerText = msg;
});

socket.on('stop_typing', () => {
    typingIndicator.innerText = '';
});
```

JS index.js > ...

```
socket.on('disconnect', (reason) => {
  console.log("Disconnected due to " + reason);
  if(socket.username != undefined)
    io.emit('is_online',
      '<i>' + socket.username + ' je izašao iz chata ;-;</i>');
});

socket.on('chat_message', (message) => {
  saveAndEmit(message, socket.username, socket.id);
});

socket.on('is_typing', () => {
  socket.broadcast.emit('is_typing', socket.username + ' kuca...');
});

socket.on('stop_typing', () => {
  socket.broadcast.emit('stop_typing');
});
});
```

JS index.js > io.on('connection') callback > socket.on('stop_typing') callback

```
function saveAndEmit(message, sender, sender_id) {
  let chatMessage = new Chat({ message: message, sender: sender, sender_id: sender_id});
  chatMessage.save();
  io.emit('chat_message', '<strong>' + sender + '</strong>: ' + message + ' <i style="float: right">' + moment(Date.now()).format('DD/MM/YYYY, LT') + '</i>');
}
```

Model pomoću kog chat poruke čuvamo kao objekte klase Chat, zatim beležimo u MongoDB bazu podataka:

models > JS ChatSchema.js > ...

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const chatSchema = new Schema({
    message: { type: String },
    sender: { type: String },
    sender_id: { type: String },
}, { timestamps: true } );

let Chat = mongoose.model("Chat", chatSchema);
module.exports = Chat;
```

Za prikazivanje poruka novijih od zadatog datuma, datum čitamo iz forme, beležimo u sesiju i ponovo učitavamo stranu (učitavamo poruke):

index.js > ...

```
60  app.post('/date', (req, res, next) => {
61    if (!req.session.name) res.send('nem\'re');
62    if (req.body.datum != "") req.session.loaddate = req.body.datum;
63    res.redirect('/chat');
64  });
65
66  app.post('/del', (req, res, next) => {
67    if (!req.session.name) res.send('nem\'re');
68    Chat.find({'createdAt' : { $lte: req.body.datum}}).remove().exec().then(() => {
69      console.log('Documents Removed Successfully');
70      req.session.loaddate = req.body.datum;
71      res.redirect('/chat');
72    }).catch((err) => {
73      console.error(err);
74    });
75  });
```

Za brisanje poruka starijih od zadatog datuma, šaljemo upit bazi podataka koji joj kaže da obriše sve poruke starije od vrednosti pročitane iz forme. Zatim se to beleži u sesiju i opet učitava strana.

Za drag-and-drop na klijentskoj strani dodajemo slušaoca (engl. listener) koji pokreće neku funkciju kada se okine. U ovom slučaju, funkcija beleži fajlove koji su drop-ovani na stranu, i za svaki fajl šalje POST zahtev serveru sa fajlom u telu forme.

public > js > chat.js > ...

```
75  msgs.addEventListener('drop', handleDrop, false);
76
77  function handleDrop(e) {
78      let dt = e.dataTransfer;
79      let files = dt.files;
80
81      handleFiles(files);
82  }
83
84  function handleFiles(files) {
85      ([...files]).forEach(uploadFile);
86  }
87
88  function uploadFile(file) {
89      let url = '/upload';
90      let formData = new FormData();
91
92      formData.append('file', file);
93
94      fetch(url, {
95          method: 'POST',
96          body: formData
97      })
98  }
```

Na serverskoj strani, fajl čuvamo u /public/files direktorijumu da bi bio dostupni svima, i emitujemo poruku:

index.js > ...

```
77  app.post('/upload', (req, res, next) => {
78    if (!req.session.name) res.send('nem\'re');
79    let form = new formidable.IncomingForm();
80    form.parse(req, (err, fields, files) => {
81      let oldpath = files.file.path;
82      let newpath = "/Users/goxy/Documents/KSS-Projekat/public/files/" + files.file.name;
83      fs.rename(oldpath, newpath, (err) => {
84        if (err) throw err;
85        saveAndEmit('<a href="files/' + files.file.name + '"  target="_blank">'
86          + files.file.name + '</a>', req.session.name, req.session.id);
87      });
88    });
89  });
```

4.2. Opis funkcionalnosti - korisničko uputstvo

Podesiti **db.js** fajl konfiguracijom vaše MongoDB instalacije (u ovom slučaju, koristimo SSH tunel ka lokalnoj Vagrant mašini koja služi kao database server):

```
JS db.js > ...
1   const mongoose = require('mongoose');
2   const tunnel = require('tunnel-ssh');
3
4   const config = {
5       username: 'vagrant',
6       password: 'vagrant',
7       host: '192.168.33.10',
8       agent: process.env.SSH_AUTH_SOCK,
9       port: '22',
10      dstPort: '27017'
11  };
12
13  const connect = tunnel(config, (err, server) => {
14      if (err) console.log('SSH connection error: ' + err);
15      mongoose.connect('mongodb://localhost:27017/chat_app', {
16          useNewUrlParser: true,
17          useUnifiedTopology: true,
18          useCreateIndex: true
19      });
20      mongoose.connection.on('connected', () => console.log('Connected'));
21  });
22
23  module.exports = connect;
```

Zatim komandom **npm install** instaliramo potrebne module, i komanda **node index.js** pokreće aplikaciju:

```
Asinhrona_Chat_Aplikacija — node index.js — 80×33
Last login: Thu Jan 23 18:23:39 on ttys005
goxy@Goxys-MacBook-Pro Documents % git clone https://github.com/SkyforgerCFW/Asi
nhrona_Chat_Aplikacija
Cloning into 'Asinhrona_Chat_Aplikacija'...
remote: Enumerating objects: 134, done.
remote: Counting objects: 100% (134/134), done.
remote: Compressing objects: 100% (89/89), done.
remote: Total 134 (delta 44), reused 126 (delta 39), pack-reused 0
Receiving objects: 100% (134/134), 11.27 MiB | 10.80 MiB/s, done.
Resolving deltas: 100% (44/44), done.
goxy@Goxys-MacBook-Pro Documents % cd Asinhrona_Chat_Aplikacija
goxy@Goxys-MacBook-Pro Asinhrona_Chat_Aplikacija % npm install

> ejs@3.0.1 postinstall /Users/goxy/Documents/Asinhrona_Chat_Aplikacija/node_mod
ules/ejs
> node ./postinstall.js

Thank you for installing EJS: built with the Jake JavaScript build tool (https:/
/jakejs.com/)

npm WARN projekat@1.0.0 No repository field.

added 163 packages from 96 contributors and audited 319 packages in 4.762s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities

goxy@Goxys-MacBook-Pro Asinhrona_Chat_Aplikacija % node index.js
listening on *:8080
Connected
```

5. Literatura

How to Build a Real-time Chat App With NodeJS, Socket.IO, and MongoDB – Eze Sunday

- https://dev.to/rexeze/how-to-build-a-real-time-chat-app-with-nodejs-socketio-and-mongodb-2kho

Starting with Authentication (A tutorial with Node.js and MongoDB) – Daniel Deutsch

- https://medium.com/createdd-notes/starting-with-authentication-a-tutorial-with-node-js-and-mongodb-25d524ca0359

Add Facebook Login to your Node.js App with Passport.js – Felistas Ngumi

- https://www.twilio.com/blog/facebook-oauth-login-node-js-app-passport-js

How to add Passport.js Google OAuth Strategy to your website – Jon Preece

- https://developerhandbook.com/passport.js/how-to-add-passportjs-google-oauth-strategy/

Passport dokumentacija

- http://www.passportjs.org/docs

Bootstrap Datepicker in Modal Popup Window - Fengyuan Chen

- https://www.codehim.com/date-time/bootstrap-datepicker-in-modal-popup-window/

How To Make A Drag-and-Drop File Uploader With Vanilla JavaScript

- Joseph Zimmerman

- https://www.smashingmagazine.com/2018/01/drag-drop-file-uploader-vanilla-js/

Node.js Upload Files - W3Schools

- https://www.w3schools.com/nodejs/nodejs_uploadfiles.asp