Osnovna načela rada Meteor.js izomorfnog radnog okvira

Barišić, Petar

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Preddiplomski sveučilišni studij računarstva

Osnovna načela rada Meteor.js izomorfnog radnog okvira

Završni rad

Petar Barišić

Osijek, 2021. godina.

SADRŽAJ

1.UVOD

Razvoj mrežnih aplikacija predstavlja proces potreban za izgradnju aplikacije kao i za njeno daljnje održavanje korištenjem klijentske i poslužiteljska strane. Mrežne aplikacije se održavaju na udaljenim poslužiteljima i te im se pristupa preko interneta koristeći mrežne preglednike. Većina ih je napisana koristeći hipertekstualni opisni jezik (engl. *HyperText Markup Language - HTML)* za prikaz sadržaja, JavaScript skriptni programski jezik za dodavanje raznih dinamičkih sadržaja i mogućnosti na aplikaciji, kao što je autentifikacija, te stilski jezik (engl. *Cascading Style Sheets - CSS*) za dizajn aplikacije [1].

Da bi se kreirala mrežna aplikacija, prvo je potrebno odrediti problem koji aplikacija treba riješiti. Zatim treba smisliti logiku kojom će se aplikacija voditi dok se na kraju piše programski kod aplikacije. U početku razvojni inženjeri nisu imali upute kako najučinkovitije napisati kod za zahtjevnije mrežne aplikacije, što je rezultiralo stvaranjem programskog koda s velikim brojem linija. Razvojni inženjeri su morali konstantno testirati svoj kod, smišljajući sami algoritme za nadolazeće greške u kodu. Da bi se to izbjeglo, stvoreni su radni okviri (engl. *frameworks*). Radni okvir je alat koji korisniku nudi unaprijed optimizirana i automatizirana rješenja za održavanje i razvoj programskog koda. Svaki radni okvir je specijaliziran za određene situacije dajući korisniku vremena da se usredotoči na zahtjevnije komponentne u daljnjem razvoju mrežne aplikacije. Zbog sve veće potražnje za mrežnim aplikacijama, radni okviri su se vremenom razvijali. Danas postoje tzv. izomorfni radni okviri. To su radni okviri koji pokreću mrežnu aplikaciju i na klijentskoj i na poslužiteljskoj strani. Dok se na ostalim radnim okvirima treba pisati poseban kod za klijentsku stranu i za poslužiteljsku stranu, što može rezultirati nepotrebnim dupliciranjem koda, u izomorfnim radnim okvirima se koristi isti programski kod za obje strane. Time se znatno skraćuje vrijeme pisanje koda, kao i mogućnost nastanka pogreške.

Potražnja za razvojnim inženjerima specijaliziranim za mrežne aplikacije nikada nije bila veća. Najpopularniji radni okviri koji takvi inženjeri koriste su oni temeljeni na *Node.js* [2] razvojnoj platformi. Trenutno jedan od najkorištenijih *Node.js* razvojnih okvira je *Meteor.js* [3]. Objasnit će se što je to uopće Node.js te koji su to aktualni radni okviri konkurenti *Meteor.js-u.* Radni okviri će se usporediti s *Meteor.js-om,* kako bi se utvrdile njihove razlike. Detaljno će se objasnit struktura *Meteor.js* radnog okvira, kakve usluge pruža te koji su prednosti i nedostatci. Na jednostavnoj aplikaciji za razgovor će se prikazati i objasniti mogućnosti koje pruža te će se usporediti sa aplikacijom za razgovor napravljenoj na *Express.js* [4] radnom okviru. Cilj je na

konkretnom primjeru utvrditi u kojim dijelovima je *Meteor.js* bolji, odnosno lošiji od konkurencije kao što je *Express.js*.

2. NODE.JS RADNI OKVIRI

Node.js je višeplatformski projekt otvorenog izvora (engl. *open-source*) koji pruža siguran i lak način za izgradnju mrežnih aplikacija visokih performansa u JavaScript skriptnom programskom jeziku. Omogućuje stvaranje različitih aplikacija koristeći V8 JavaScript programsko-logički motor (engl. *engine*) izvan internetskog preglednika. JavaScript se inače izvršava u internetskom pregledniku, no *Node.js* omogućuje njegovo pokretanje na poslužiteljskoj razini. Zahvaljujući tome razvojni inženjeri mogu pisati poslužiteljski dio koda uz klijentski dio koristeći JavaScript-u. *Node.js* se pokreće u jednom procesu i to bez potrebe stvaranja novih niti za svaki zahtjev. Node.js izvršava ulazno/izlazne operacije, npr. čitanje ili pisanje podataka iz ili u disk, bez da blokira nit. Zahvaljujući tome ne treba koristi puno resursa, a samim time je i brži [5].

Radni okvir je alat koji nudi programeru unaprijed dizajnirane funkcije i mogućnosti koje se najviše koriste za razvoj raznih aplikacija. Unaprijed je testiran i optimiziran, zbog toga se proces kreiranja aplikacija znatno ubrzava i olakšava. Zbog velike potražnja za mrežnim aplikacijama, razvojni inženjeri istražuju najbolje radne okvire koje pružaju razne i napredne mogućnosti za kreiranje mrežnih aplikacija. Prema *Node.js* korisničkoj anketi iz 2018., 85% ispitanika koristi *Node.js* za razvoj mrežnih aplikacija. Također 4 od 5 *backend* i *full stack* razvojnih inženjera koristi *Node.js* radne okvire kao glavni radni okvir [6]. Razlog tome je što *Node.js* radni okviri nude brzo i jednostavno razvijanje aplikacija koje se mogu koristiti u različite svrhe. U nastavku će se objasniti njih četiri te posebno *Meteor.js*.

2.1. Adonis.js

Adonis.js [7] je *Node.js* model-pogled-upravljač (engl. *model.view-controller- MVC)* radni okvir koji se fokusira na stabilnost i brzinu. Ima podršku za sve operacijske sustave i sadrži razne pakete koje se pokreću u aplikaciji na razini poslužitelja. Koristi napredni sistem usmjeravanja te ima snažnu mrežnu zaštitu [8].

Tablica 2.1. Usporedba *Adonis.js-a* i *Meteor.js-a*.

Adonis.js	**Meteor.js**
SQL baza podataka	NoSQL baza podataka
TypeScript	JavaScript
Mrežne aplikacije	Mobilne, mrežne i aplikacije za radnu površinu

Iz tablice 2.1. se vidi da *Adonis.js* koristi bazu podataka na temelju strukturnog upitnog jezika (engl. *Structured Query Language - SQL*), dok *Meteor.js* koristi bazu podataka koja nije temeljena samo na strukturnom upitnom jeziku (engl. *Not only Structured Query Language - NoSQL*), tj. *MongoDB* [9]. Time je *Adonis.js* ograničen na predefinirane sheme za određivanje strukture podataka, dok *Meteor.js* može koristiti dinamične sheme i spremati podatke na različite načine. Također, *Adonis.js* je ograničen na mrežne aplikacije, dok se *Meteor.js* može koristiti i na drugim platformama. No s druge strane, TypeScript je u suštini JavaScript s dodatnim mogućnostima te omogućuje korištenje objektno orijentirane paradigme, bolji je u stvaranju većih aplikacija i pronalaženju grešaka.

2.2. Express.js

Express.js je minimalistični i prilagodljivi *Node.js* mrežni radni okvir, dizajniran za mrežne aplikacije s jednom ili više stranica. Omogućuje definiranje ruta, tj. određuje kako će se aplikacija ponašati prilikom primanja zahtjeva od klijenta, za aplikacije koje su bazirane na metodama protokola prijenosa hipertekstualnih dokumenata (engl. *HyperText Transfer Protocol - HTTP)* te je idealan za stvaranje dodatnih zadataka na zahtjev. Također ima dobru ulazno/izlaznu performansu [10].

Tablica 2.2. Usporedba *Express.js-a* i *Meteor.js-a*.

Express.js	**Meteor.js**
Backend	*Frontend i backend*
Mobilne i mrežne aplikacije	Mobilne, mrežne i aplikacije na radnoj površini
JsonWebToken	Ugrađeni sustav za autentifikaciju
Nema integriranu bazu podataka	*MongoDB*

Iz tablice 2.2. se vidi da je *Express.js* fokusiran samo na *backend*, dok je *Meteor.js* fokusiran i na *frontend* i na *backend*. Dok *Meteor.js* ima integriranu bazu podataka s *MongoDB-om, Express.js* nema integriranu bazu podataka te ju treba posebno instalirati i postaviti. Time se troši vrijeme i resursi te je veća vjerojatnost nastanka greške. *Meteor.js* također ima ugrađeni jednostavni i efikasni sustav za autentifikaciju, dok je za *Express.js* potrebna instalacija tzv. *„JsonWebToken“* metode za autentifikaciju.

2.3. LoopBack

LoopBack [11] je visoko proširivi *Node.js* radni okvir otvorenog koda. Omogućava brzo stvaranje aplikacijskog programskog sučelja (engl. *application programming interface - API*), koji omogućuje aplikacijama da međusobno komuniciraju, kao mikroservisa, kreiranja elektroničkih pošta i notifikacija. Posjeduje unaprijed definirane strukture i funkcionalnosti [12].

Tablica 2.3. Usporedba *LoopBack-a* i *Meteor.js-a*.

LoopBack	**Meteor.js**
Snažna podrška za API	Podrške za različite alate
Nema integriranu bazu podataka	*MongoDB*
IBM Cloud	*Meteor Galaxy*
TypeScript	JavaScript

Iz tablice 2.3. se vidi da *LoopBack* ima snažnu podršku za API, dok *Meteor.js* ima podršku za različite alate, uključujući i za API. LoopBack ima mogućnost korištenja *IBM Cloud-a* [13] za usluge u oblaku, a *Meteor.js* ima svoj vlastiti *Meteor Galaxy cloud* [14] sustav koji je dizajniran posebno za njega.

2.4. Sails.js

Sails.js [15] je prilagodljiv *Node.js* radni okvir koji se vodi principom *„convention over configuration“.* To je jednostavan koncept koji govori da okruženje u kojem radimo sam pretpostavlja logičke situacije koje možemo koristiti umjesto da ih stvaramo sami svaki put. Time je programiranje lakše i produktivnije. Omogućava brzo kreiranje REST API-a, jednostraničnih aplikacija i aplikacija u pravom vremenu. Napravljen je na *Express.js-u* i *Socket.io-u*, JavaScript biblioteci za dodavanje aplikaciji elemenata pravog vremena[16].

Tablica 2.4. Usporedba *Sails.js-a* i *Meteor.js-a*.

Sails.js	**Meteor.js**
JavaScript	TypeScript
Waterline	*MongoDB*
Socket.io	U stvarnom vremenu
Frontend	*Frontend i backend*

Iz tablice 2.4. se vidi da *Sails.js* koristi *Waterline* [17], objektno orijentacijsko mapiranje koji stvara sloj za pristup podacima s bilo koje baze podataka, dok je *Meteor.js* integriran isključivo za korištenje *MongoDB* baze podataka. *Meteor.js* ima svoju podršku pravog vremena, dok *Sails.js* koristi *Socket.io* kako bi to postigao. Također, *Meteor.js* se može koristit za *frontend i backend*, a Sails.js samo za *frontend*.

3. METEOR.JS

Meteor ili *Meteor.js* je izomorfni razvojni ekosustav (engl. *Isomorphic Development Ecosystem - IDevE*) otvorenog koda službeno lansiran 2012. godine od strane *Meteor Software* tima. Napisan je u JavaScript skriptnom programskom jeziku na *Node.js* platformi i integriran je s *MongoDB* bazom podataka. Omogućuje jednostavnu i učinkovitu izradu svih vrsta aplikacija u stvarnom vremenu. *Meteor.js* automatski integrira sve komponente potrebne za stvaranje i postavljanje aplikacija. Njegova arhitektura ažurira aplikaciju tijekom promjena za sve klijente istovremeno, zbog čega ne treba osvježivati stranicu, što je idealno za stvaranje zahtjevnijih mrežnih aplikacija [18]. Koristi se isti kod za mrežne aplikacije, Android ili iOS. To je moguće zahvaljujući *Cordovi* [19], mobilnim razvojnim radnim okvirom tvrtke Apache.

3.1. Izomorfni razvojni ekosustav

Izomorfnost se očituje u korištenju istog programskog koda i na *frontend-u* i na *backend-u,* odnosno korištenjem istog API-a u svakom djelu aplikacije. Pomoću *Meteor.js* metoda se to može najbolje prikazati. Jedan od načina je da se metoda deklarira samo na poslužitelju te pristupi preko klijenta pomoću povratnog poziva *Meteor.call()*. Drugi način je da se metode dekaliraju izvan */server* i */client* mape, gdje će klijent pozvati metodu pomoću *Meteor.call()*, koja će se onda izvoditi i na klijentu i na poslužitelju (Slika 3.1.). Zapravo metodu poslužitelj izvršava, a klijent samo simulira metodu kako bi nadoknadio kašnjenje između klijenta i poslužitelja, stvarajući time dojam trenutnog odgovora.

```
1:   //Poslužitelj
2:   Meteor.methods({
3:      methodeOne: function(input){},
3:
4:      methodeTwo: function(input){}
5:   });
6:
7:   //Klijent
8:   Meteor.call('MethodeOne',input,function(error,output){
9:
10:  });
11:
```

Slika 3.1.Prikaz *Meteor.js* metode koja se istovremeno izvršava na klijentu i poslužitelju .

Razvojni dio *Meteor.js-a* predstavlja ponudu raznih alata, napravljenih isključivo za *Meteor.js,* za korištenje tijekom razvoja aplikacije. Jedan od korisnih alata je *Meteor DevTools* [20], proširenje na mrežnom pregledniku *Google Chrome* [21] koje omogućava jednostavan pregled infrastrukture mrežne aplikacije.

Ekosustav *Meteor.js-a* nudi optimiziranu povezanost *frontend-a, backend-a* i *MongoDB* baze podataka, kao i snažnu povezanost s dodatnim paketima i uslugama. Time omogućuje stvaranje, razvijanje i proširivanje aplikacije jednostavnijim i bržim. Ovakav ekosustav omogućuje *Meteor.js* aplikacijama pokretanje korištenjem ostalih davatelja usluga.

3.2. JavaScript

JavaScript je skriptni jezik korišten za stvaranje i upravljanje dinamičkog sadržaja mrežne stranice. Skriptni jezici su jezici koji se koriste za automatizaciju posla koji bi inače korisnik morao sam raditi, kao što je osvježavanje mrežne stranice ili navigacija kroz izbornike kako bi se došlo do željenog sadržaja. Zbog toga što je JavaScript trenutno glavni sastavni dio dinamičkih funkcionalnosti mrežnih stranica, svi ga aktualni mrežni preglednici podržavaju. To omogućuje direktno korištenje JavaScript naredbi u *HTML-u* korištenjem <script> oznaka i mrežni preglednik će ih razumjeti. Drugim riječima, JavaScript ne zahtijeva skidanje dodatnih programa i prevoditelja (engl. *compiler*), što omogućava njegovu fleksibilnost. Zbog toga *Meteor.js* koristi ovaj jezik i na klijentu i na poslužitelju. Time se uklanja mogućnost dupliciranja logike aplikacije između poslužitelja i klijenta. Korisnici mogu iskoristiti maksimalno JavaScript kako bi smanjili kompliciranost koda. Dolazi do značajne uštede vremena, jer nema potrebe za prebacivanje iz programskog jezika na poslužitelju u JavaScript i obrnuto [22].

3.3. MongoDB

MongoDB je NoSQL program baze podataka orijentiran na dokumentima. Slično kao što se relacijske baze podataka temelje na tablicama, baze podataka orijentirane na dokumentima poput *MongoDB* temelje se na zbirci dokumenata temeljenim na odnosu ključ-vrijednost. Vrijednost može biti jednostavna kao što je broj ili riječ, a može biti i složenija u obliku objekta. Ključ predstavlja jedinstvenu vrijednost pomoću koje brzo i efikasno pronalazimo objekt. *MongoDB* također može dinamički promijeniti shemu te tako lako napraviti bitne promjene bez potrebe za preoblikovanjem postojeće baze podataka. Uz to, hijerarhija dokumenata se lako preslikava na

hijerarhije objekata unutar aplikacijskog koda, pojednostavljujući operacije stvaranja, čitanja, ažuriranja i brisanja baze podataka [23].

Meteor.js je integriran s *MongoDB-om* i koristi njegovo svojstvo kolekcija. Kolekcije predstavljaju skup povezanih podataka, preko kojih *Meteor.js* pristupa *MongoDB-u*. To pruža *Meteor.js-u* brze interakcije s bazom podataka uz jednostavne implementacije.

3.4. Apache Cordova

Meteor.js se integrira s *Cordovom,* radnim okvirom tvrtke Apache za izradu mobilnih aplikacija. *Cordova* omogućuje korištenje istog koda, koji se koristi za izradu mrežnih aplikacija, za izradu mobilnih aplikacija. Zahvaljujući tome, *Meteor.js* aplikacija se može pokrenuti na iOS ili Android uređaju s nekoliko jednostavnih naredbi.

Cordova aplikacija se pokreće u mrežnom prikazu koji se ugradi u *Meteor.js* aplikaciju umjesto u mrežnom pregledniku na mobitelu. Prednost *Cordova* aplikacije je što osigurava da su svi bitni dokumenti u paketu s aplikacijom. Aplikacija će se time učitati brže za korisnike na sporim mobilnim vezama. Također ažuriranje aplikacije na uređajima je vrlo jednostavno. Korisnik samo treba skinuti nadogradnje sa poslužitelja i spremiti na uređaj. Stranica će se sama ponovno učitati i povući nadogradnje skinute na mobilni uređaj. Posebni dodaci omogućuju upotrebu značajki koje obično nisu dostupne mrežnim aplikacijama, poput pristupa uređaju ili lokalnom sustavu datoteka [24].

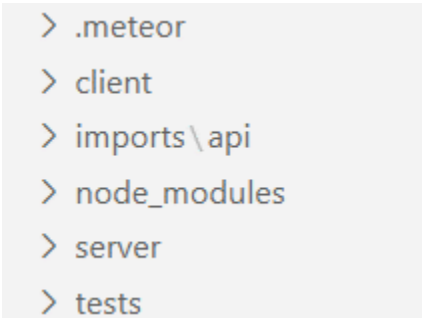
3.5. Meteor Cloud

Meteor.js posjeduje vlastite usluge u oblaku, *Meteor Cloud*, pomoću kojeg se izvršava usluge dijeljenja, nadzora i upravljanja aplikacijama. *Meteor Cloud* nudi centraliziran pristup posebnim mogućnostima na svoj mrežnoj stranici:

- *Meteor* promatranje performanse aplikacije (engl. *application performance monitoring - APM*) je alat za promatranje performanse *Meteor.js* aplikacija.
- *Atmosphere* je posebni direktorij u kojem *Meteor.js* razvojni inženjeri postavljaju svoja rješenja za česte probleme s kojima se korisnik susreće. Paketi su dostupni svima i besplatni za skidanje.

- *Galaxy Hosting* nudi usluge poslužitelja kao što su okidači i obavijesti te napredne sigurnosne mjere poput popisa pouzdanih IP adresa i zaštite aplikacija [25].

3.6. Posebni direktoriji



Slika 3.2.Prikaz *Meteor.js* posebnih direktorija koji se automatski generiraju .

Meteor.js JavaScript datoteke su povezane i učitane i na klijentu i na poslužitelju. Nazivi datoteka i direktorija unutar projekta bitno utječu na raspored njihovog učitavanja, stoga s njima treba oprezno rukovati [26]. U nastavku slijedi popis posebnih direktorija koje *Meteor.js* automatski generira (Slika 3.2.):

- **client –** programski kod u ovom direktoriju će se učitati samo na klijentu. Slično se može postići i s *if (Meteor.isClient) { ... }* naredbom. Sve datoteke učitane na klijentu se automatski povežu.
- **imports** – datoteke se u ovaj direktorij moraju uvesti korištenjem naredbe *import.* Služi kako bi se *Meteor.js* povezao i sadržavao samo datoteke koje su referencirane iz druge datoteke.
- **server –** programski kod u ovom direktoriju će se učitat samo na poslužitelju. Slično se može postići i s *if (Meteor.isServer) { ... }* kodom. Koristi se za programske kodove koje mogu sadržavati sigurnosne rizike, poput zaporki.
- **tests –** ovaj direktorij služi za pokretanje bilo kojeg testnog koda korištenjem posebnih alata za testiranje.

Meteor.js povezuje sve JavaScript datoteke, osim onoga što se nalazi u *client/, public/* i *private/* poddirektoriju i učitava ih na *Node.js* instancu poslužitelja. Programski kod napisan na poslužitelju se odvija u jednoj niti po zahtjevu.

- **public** - datoteke ovog poddirektorija su posluženi kao da su klijentski. Ikone, tekstualne datoteke i njima slične se nalaze u ovom poddirektoriju.

- **private –** datoteke ovog poddirektorija se mogu dohvatiti iz poslužiteljskog koda i učitati pomoću *Assets* API-a. Koristi se za privatne datoteke i za datoteke za koje ne želimo da budu dohvaćeni izvana.

3.7. Nedostatci Meteor.js-a

Meteor.js pruža integriranost i proširenost s različitim paketima. To može predstavljati i nedostatke:

1. Uska povezanost *frontend-a*, *backend-a* i baze podataka predstavlja njihovu nemogućnost da se rastave na posebne cjeline. Time je otežana zamjena i promjena većeg broja cjelina.

2. Povezanost s NoSQL *MongoDB* bazom podataka može biti zahtjevna za korisnike koji inače koriste SQL bazu podataka. Za manji broj podataka su SQL baze podataka idealnije jer je jednostavnije i brže s njima baratati.

3. Svojstvo stvarnog vremena (engl. *Real time*) da se stranica aplikacije konstantno ažurira automatski se uključi za svaku mrežnu aplikaciju, što znači da će raditi u pozadini i kada ne treba. To uzrokuje značajno vrijeme koje je potrebno da se stranica učita u odnosu na ostale radne okvire koji nemaju to svojstvo.

4. Iako integriranost s posebnim alatima olakšava programiranje, ona ograničava korisnika na samo te alate. U mnogim slučajevima se koriste tek neki od njih, dok ostali, koji rade samo u pozadini, usporavaju aplikaciju.

5. Svojstvo *Meteor.js-a* za prebacivanje aplikacija na mobilne uređaje pomoću *Cordova* radnog okvira je korisno, ali i dalje ima grešaka te nije potpuno optimizirano.

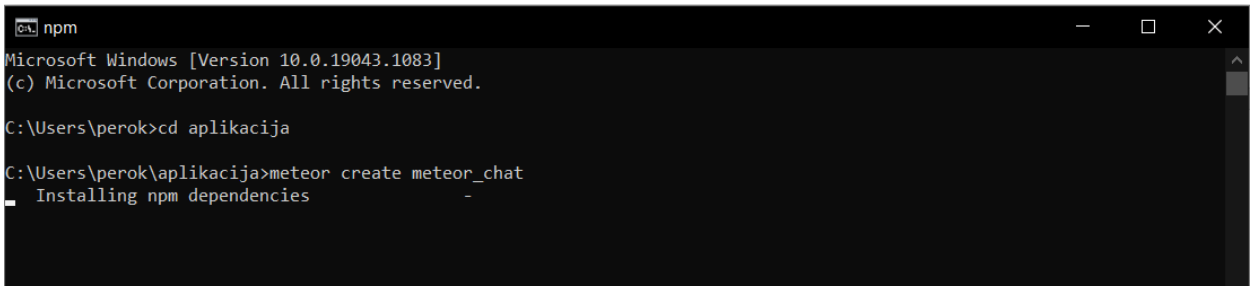
6. *Atmosphere* direktorij na *Meteor Cloud-u* sadrži datoteke koje se ne održavaju te ih je nemoguće instalirati na *Meteor.js* aplikacijama.

4. METEOR.JS APLIKACIJA ZA RAZGOVOR

Kako bi prikazali na konkretnom primjeru prednosti i nedostatke *Meteor.js-a*, kreirat će se jednostavna aplikacija za razgovor u stvarnom vremenu na *Windows 10* operacijskom sustavu .

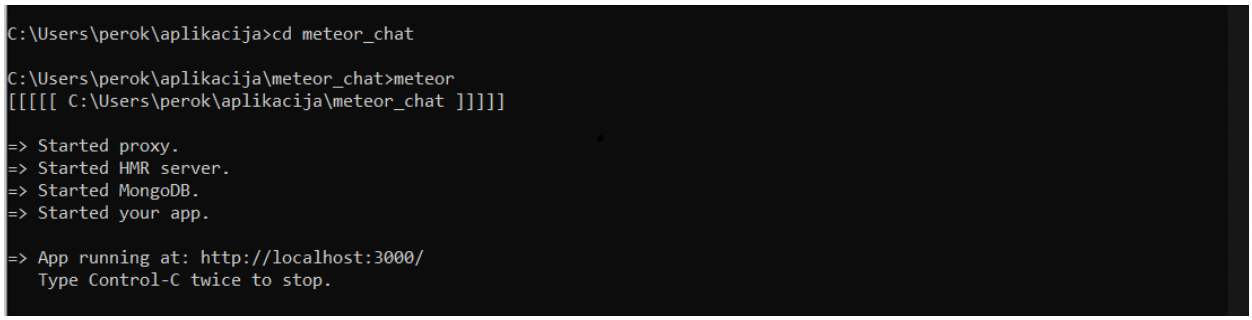
4.1. Početak izrade aplikacije

Za početak treba pokrenuti *command promt* te u terminalu upisati naredbu *cd aplikacije* za odabiranje datoteke u koju će se nalaziti *meteor* datoteka. Zatim jednostavno upisat naredbu *meteor create meteor_chat* i *Meteor.js* će sam instalirati sve datoteke i alate koje su mu potrebni (Slika 4.1.).



Slika 4.1. Inicijalizacija *Meteor.js* aplikacije u command promt-u.

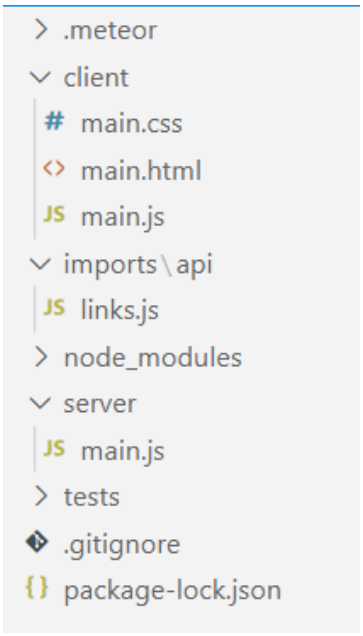
Nakon toga treba upisati naredbu *cd meteor_chat* za odabiranje datoteke aplikacije i pokrenuti aplikaciju naredbom *meteor* (Slika 4.2.).



Slika 4.2. Pokretanje *Meteor.js* aplikacije.

Iz slike 4.2. se vidi da je *Meteor.js* prvo pokrenuo usluge koje mu trebaju, uključujući i *MongoDB* bazu podataka te je onda tek pokrenuta naša aplikacija. Aplikacija je pokrenuta na standardnoj

lokalnoj mrežnoj stranici http://localhost:3000/ , gdje *Meteor.js* svaki put pokreće svoje aplikacije. Za mrežnog preglednika se koristi *Google Chrome*, a za okruženje u kojem će se pisati programski kod se koristi *Visual Studio Code.*



Slika 4.3. Meteor.js datoteke.

U *VS Code-u* se vidi već spomenuta struktura datoteka sa slike 4.3. koje *Meteor.js* sam stvori i time olakšava i ubrzava vrijeme potrebno za pisanje programskog koda. JavaScript datoteku *main.js* koja se nalazi u *server/* direktoriju je preimenovana *server.js* radi lakšeg snalaženja i smanjenja mogućnosti nastanka pogreške.

4.2. Glavni dijelovi programskog koda

U nastavku slijedi prikaz programskoga koda aplikacije koji se nalazi u P.4.1. Objasnit će se glavni i najbitniji dijelovi programskoga koda.

```
1:   /…/
2:   export const Messages = new Mongo.Collection('messages');
3:
4:   Meteor.methods({
5:       'setMessage'(message){
6:           check(message, String);
7:
8:           if(!Meteor.userId()){
9:               throw new Meteor.Error();
10:          }
```

Slika 4.4. Dio koda iz *messages.js* datoteke

```
11:
12:
13:         Messages.insert({
14:            id: Meteor.userId(),
15:            time: new Date(),
16:            message
17:           });
18:      }
19:  });
```

Slika 4.5. Nastavak koda iz *messages.js* datoteke

Na slici 4.4. se vidi kako se stvara nova *MongoDB* kolekcija baze podataka *messages.* Također se vidi kako se stvara *Meteor.js* metoda nazvana *setMessage* koja prvo provjerava je li poslana poruka objekta *String*, te postoji li pošiljatelj. Na slici 4.5. se vidi kako metoda sprema u bazu podataka sve potrebne informacije za razgovor: pošiljatelja, vrijeme poslane poruke i sama poruka.

```
1:   /…/
2:     getUser(id){
3:       if(id){
4:          const user = Meteor.users.findOne(id);
5:          if(user)
6:              return user.username;
7:          }
8:       },
9:   /…/
10:  /…/
11:      Meteor.call('setMessage', message,(err)=>{
12:        if(err){
13:           alert(err.message);
14:        } else {
15:          event.target.reset();
16:        }
17:     });
18:  /…/
```

Slika 4.6. Dio koda iz *main.js* datoteke na klijentu

Iz slike 4.6. se vidi integracija *MongoDB-a* i *Meteor.js-a* u metodi *getUser()* gdje se traži ime pošiljatelja prema njegovom *id-u.* Prikazana je i uporaba *Meteor.call()* povratnog poziva, gdje se preko klijenta poziva *setMessage()* metoda. U nju se sprema upisana poruka i provjerava te se preko *HTML-a* dohvaća pomoću *#each* iteracije koja prolazi kroz poruke i ispisuje ih na zaslon.

```
1:   /…/
2:   Meteor.startup(()=>{
3:       Messages.remove({});
4:   });
```

Slika 4.7. Dio koda iz *server.js* datoteke

Na poslužiteljskoj strani aplikacije se koristi *Meteor.js* metoda *startup(),* kako bi se svaki put, prilikom ponovnog pokretanja aplikacije, izbrisale poruke iz kolekcije ne bi li došlo do preopterećenja i usporenosti aplikacije (Slika 4.7.).

Također je kao posebno svojstvo dodana i potreba za autentifikacijom. Implementacije je bila jednostavna, jer *Meteor.js* posjeduje svoj sistem za registraciju, odnosno prijavu korisnika. U terminalu treba dodati autentifikacijski sistem pomoću naredbe *meteor add accounts-ui accounts-password,* te napisati jednostavan programski kod u JavaScript datoteci (Slika 4.8.).

```
1:   /…/
2:   Accounts.ui.config({
3:       passwordSignupFields: 'USERNAME_AND_EMAIL',
4:   });
5:   /…/
```

Slika 4.8. Dio koda iz *main.js* datoteke

Korisnik se treba registrirati elektroničkom poštom i imenom te lozinkom. Integracija i s ostalima vrstima računa, kao što je *Facebook,* je također dostupna.

4.3. Usporedba sa Express.js aplikacijom

Kako bi najbolje prikazali prednosti i mane *Meteor.js* aplikacije, napravljena je usporedba s aplikacijom programiranoj na *Express.js-u,* koja je također aplikacija za razgovor. *Express.js* je trenutno jedan od najkorištenijih radnih okvira za programiranje mrežnih aplikacija.

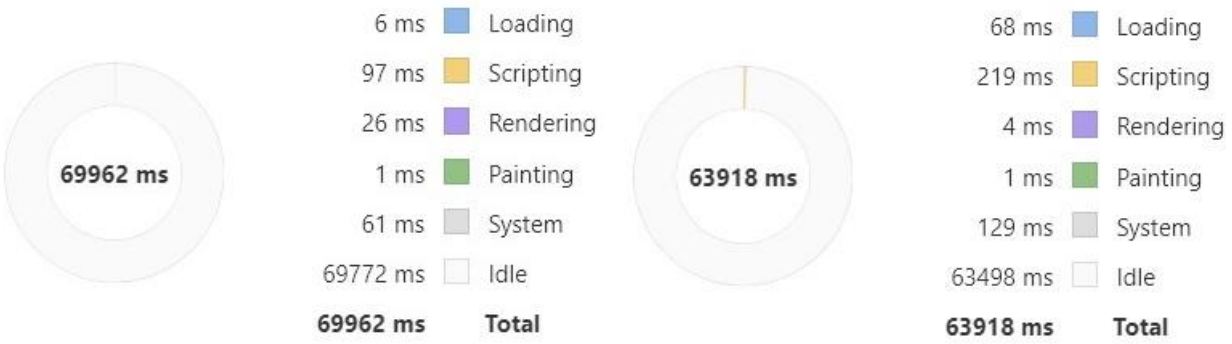
4.3.1. Strukture

Dok se *Meteor.js* sam brine za osvježavanje aplikacije, za *Express.js* je bilo potrebno instalirati *Node.js* alat pod nazivom nodemon. *Nodemon* omogućava automatsko ažuriranje aplikacije svaki put kad se dogodi promjena. Zatim, da bi se postigla *Meteor.js-ova* mogućnost rada u stvarnom vremenu, instaliran je *Socket.io-a. Socket.io* je JavaScript biblioteka zadužena za stvarno vrijeme i dvosmjernu komunikaciju između poslužitelja i mrežnog klijenta. Naposljetku, *Express.js* nije integriran s bazom podataka te je i instalacija i implementacija iste potrebna.

Na konkretnom primjeru se da zaključiti kako *Meteor.js-ova* integracija sa različitim alatima i bazom podataka od velike koristi. Time je pisanje programskog koda bilo znatno lakše i brže, nego programskog koda u *Express.js-u.* Programski kod *Express.js* aplikacije se nalazi u P.4.2.

4.3.2. Vrijeme izvođenja i memorijski otisak aplikacija

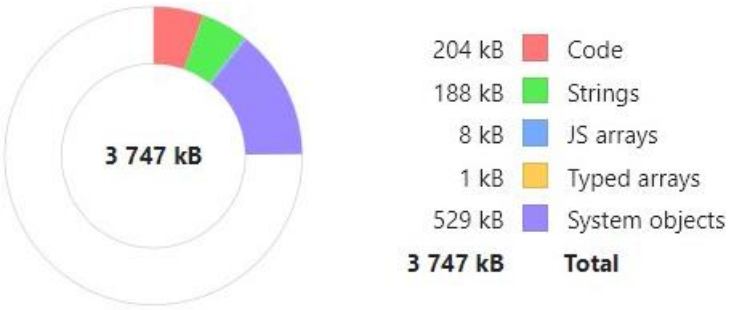
Prvo su testirana vremena koja su potrebna procesoru za izvođenje aplikacija (engl. *CPU time*) tijekom učitavanja te njihov memorijski otisak. Za ove testove su korišteni ispitivač memorije (engl. *Memory inspector)* i ploča za pregled izvedbe (engl. *Performance panel*), razvojni alati ugrađeni u mrežnom pregledniku *Google Chrome* pod nazivom *Chrome DevTools* [27].



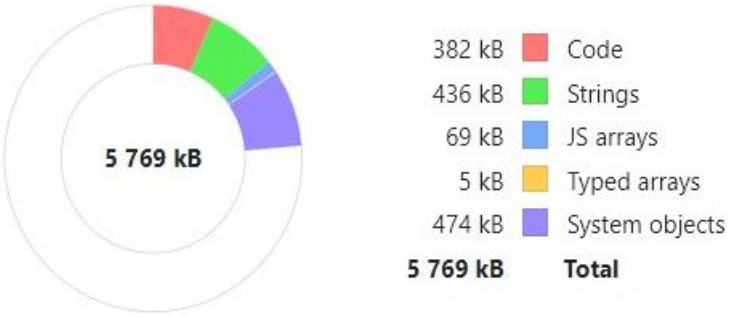
a) **b)**

Slika 4.9. Vrijeme izvođenja tijekom učitavanja **a)** Express.js aplikacije i **b)** Meteor.js aplikacije.

Iz grafova a) i b) sa slike 4.9, se vidi da procesoru kod *Express.js* aplikacije treba znatno manje vremena za odziv prilikom učitavanje aplikacije, manje vremena za izvršavanje skripti i manje vremena za izvršavanje funkcija sustava korištenih u aplikaciji nego kod *Meteor.js* aplikacije. Razlog tome je što *Meteor.js* automatski koristi protokol distribuiranih podataka (engl. *Distributed Data Protocol - DDP*) kako bi učitao podatke sa baze podataka na poslužitelju i poslao rezultate klijentu. DDP zahtjeva skidanje dodatnih informacija zbog kojih se i produljuje vrijeme učitavanja. No procesor kod *Meteor.js* aplikacije troši manje vremena na prikazivanje elemenata i na mirovanje.



a)



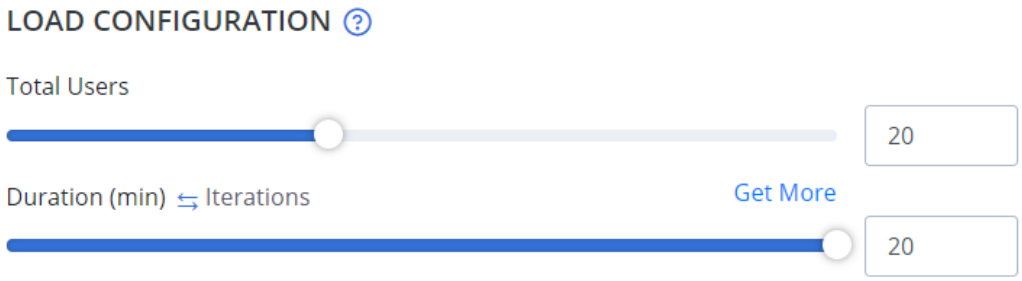
b)

Slika 4.10. Memorijski otisak **a)** Express.js aplikacije i **b)** Meteor.js aplikacije.

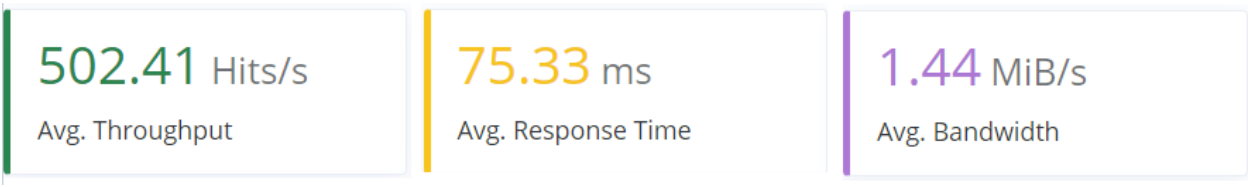
Iz grafova a) i b) sa slike 4.10. se vidi znatno manje zauzeće memorije kod *Express.js-a* nego kod *Meteor.js-a* zbog dodatnih datoteka koje *Meteor.js* na početku instalira, a ne koristi u svakoj situaciji.

4.3.3. Blazemeter testiranje aplikacija

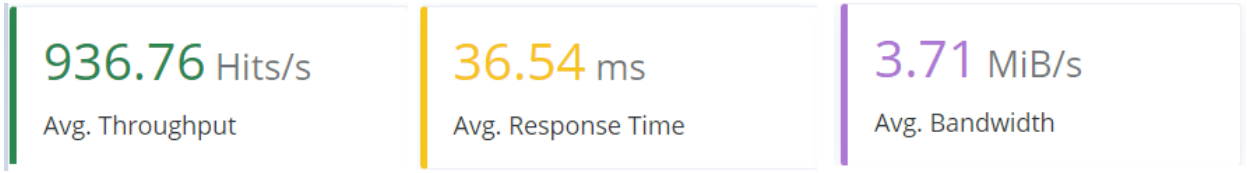
U ovome potpoglavlju su prikazane aplikacije kako se ponašaju prilikom jednostavnih radnji korištenjem *Blazemeter* [28] platforme za testiranje. Radnje se sastoje od prijave korisnika, unosa poruke pozdrava i odjavljivanje korisnika. Zahvaljujući *Blazemeter-u* ti događaji se mogu ponavljati konstantno određeno vrijeme kako bi se dobio što precizniji uvid u ponašanju aplikacija prilikom simuliranja uobičajenih radnji. Test je podešen da se izvršava 20 minuta i da ga izvršava 20 virtualnih korisnika odjednom (Slika 4.11.).



Slika 4.11. Podešavanje parametara testa



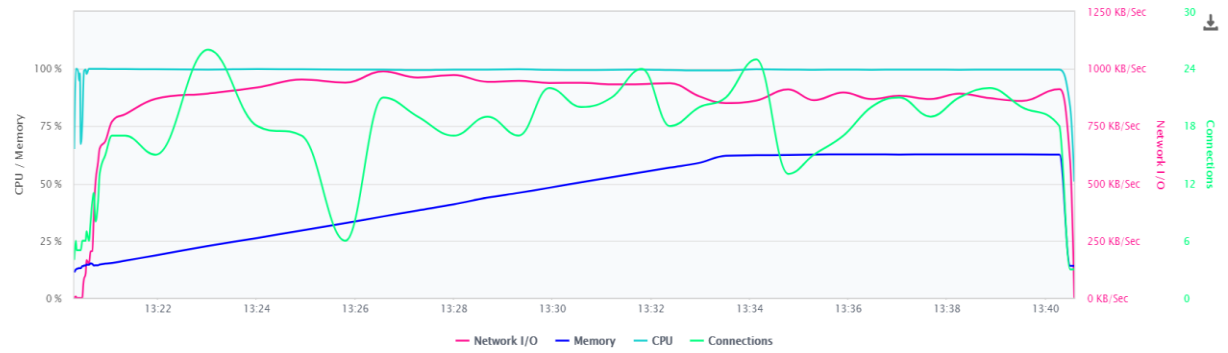
a)



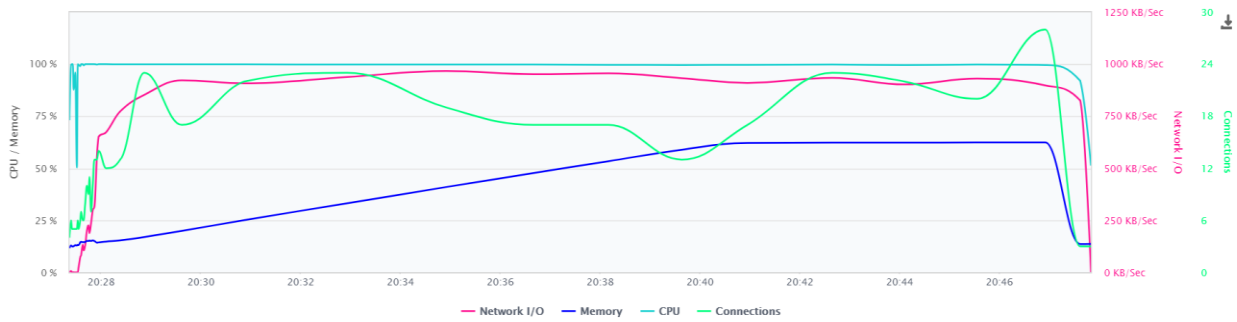
b)

Slika 4.12. Rezultati testa **a)** *Meteor.js* aplikacije i **b)** *Express.js* aplikacije.

Sa slike 4.12. se vidi da da *Meteor.js* aplikacija ima manju propusnost, odnosno šalje manji broj *HTTP* zahtjeva po sekundi mrežnom poslužitelju nego *Express.js* aplikacija. Također, *Express.js* aplikacija ima manje prosječno vrijeme odgovora poslužitelja na zahtjeve i veću širinu pojasa. To znači da *Express.js* aplikacija u prosjeku šalje veću količinu informacija po sekundi nego *Meteor.js* aplikacija.



a)



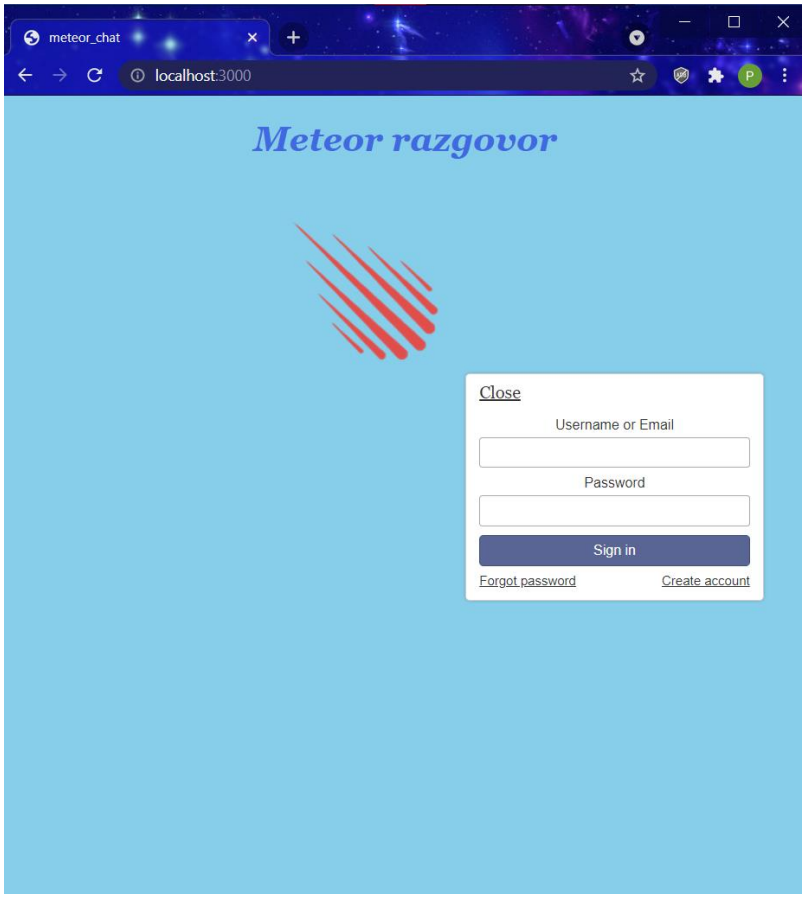
b)

Slika 4.13. Prikaz korištenja radne memorije i memorije procesora, količine podataka poslanih ulazno/izlaznim operacijama i broja uspostavljenih veza **a)** *Express.js* aplikacije i **b)** *Meteor.js* aplikacije.

Iz grafova **a)** i **b)** sa slike 4.13. se vidi da obje aplikacije u prosjeku jednako koriste memoriju procesora i radnu memoriju prilikom obavljanja zadanih radnji. Također jednaku količinu podataka šalju pomoću ulazno/izlaznih operacija i broj trajnih veza uspostavljenih za svaku transakciju tijekom testa je približno jednak, ali *Meteor.js* aplikacija ima manja odstupanja i anomalije od *Express.js* aplikacije što se vidi na grafu zelene boje.

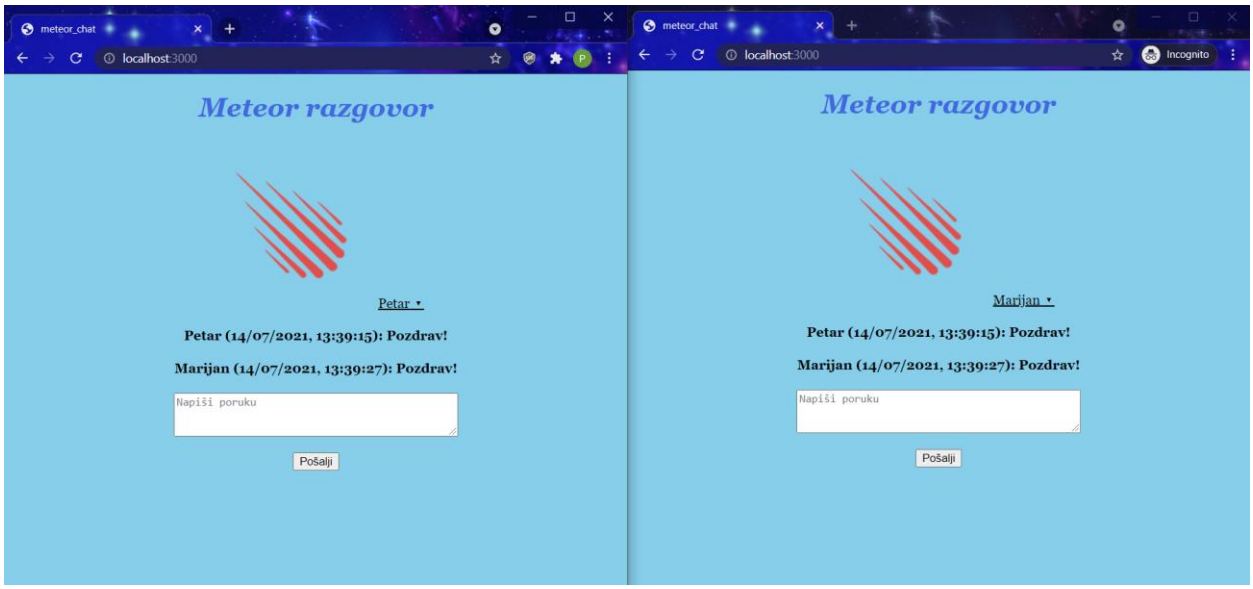
4.4. Izgled aplikacija

Prikazat će se izgled aplikacija. Također će se prikazati i objasniti njihove mogućnosti .



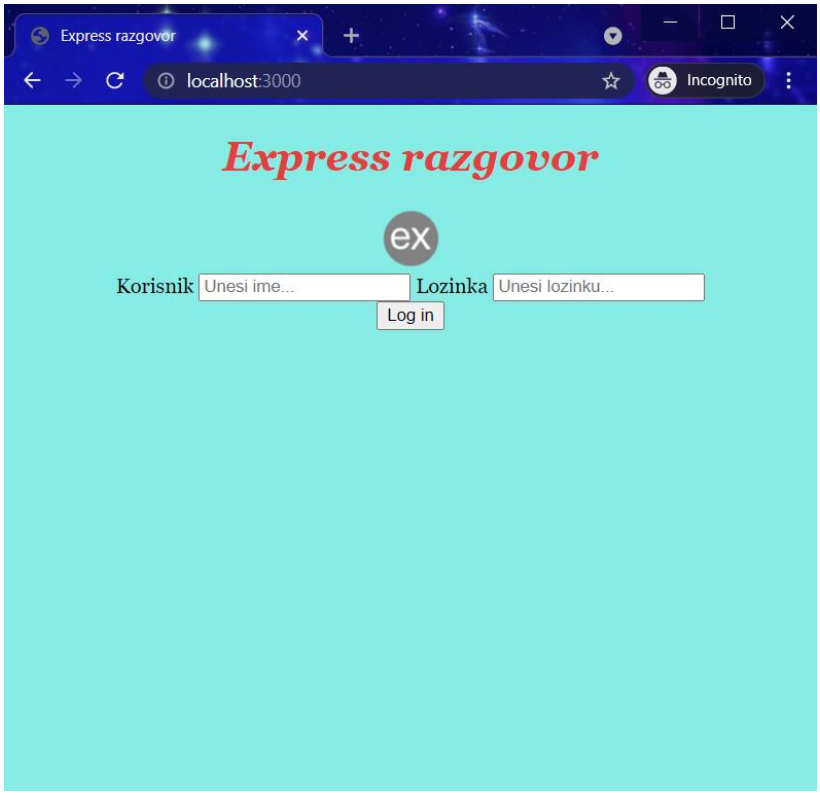
Slika 4.14. Prikaz *Meteor.js* aplikacije i procesa autentifikacije.

Iz slike 4.14. se vidi početna stranica koja korisnika traži da se registrira ili prijavi. Također se vidi da je aplikacija pokrenuta na standardnoj lokalnoj mrežnoj stranici *http://localhost:3000/.*

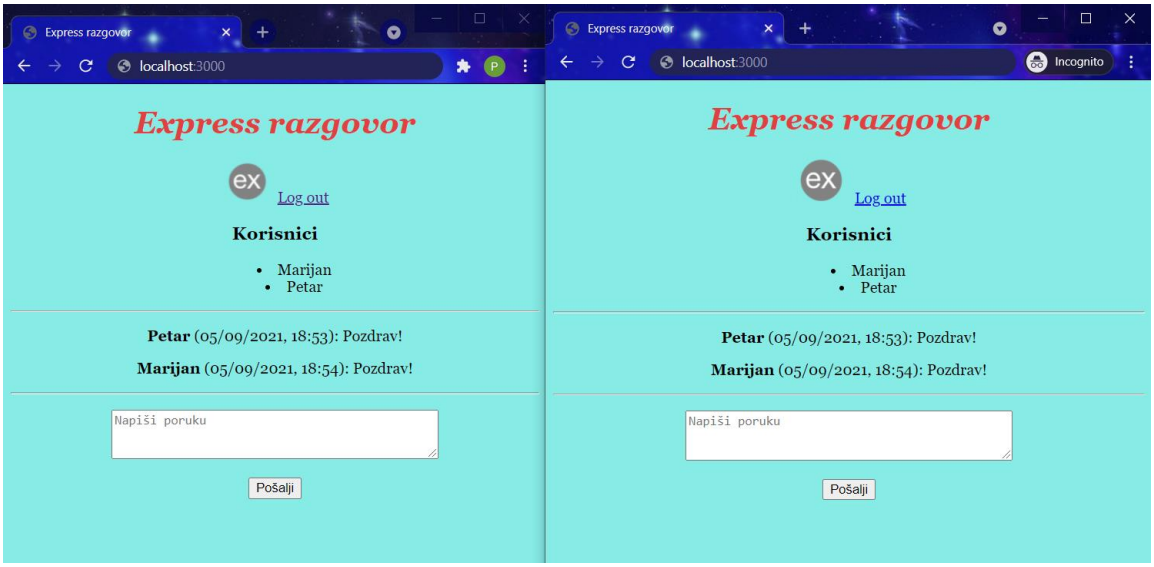


Slika 4.15. Prikaz razgovora između dva korisnika u stvarnom vremenu.

Iz slike 4.15. se vidi izgled aplikacije tijekom stvarnog razgovora. Korisnik jednostavno upiše poruku i pošalje primatelju pomoću tipke. Prikazano je kako i primatelj i pošiljatelj vide tko je i kada poslao poruku.



Slika 4.16. Prikaz početne stranice *Express.js* aplikacije.



Slika 4.17. Prikaz razgovora između dva korisnika u stvarnom vremenu.

Sa slika 4.16. i 4.17. se vidi da je *Express.js* aplikacija isto pokrenuta na standardnoj lokalnoj mrežnoj stranici *http://localhost:3000/.* Aplikacija posjeduje skoro pa identične karakteristike kao i *Meteor.js* aplikacija.

5. ZAKLJUČAK

Broj *Node.js* radnih okvira je s vremenom sve veći. Ovisno o potrebama, razvojni inženjeri izabiru one radne okvire koji su optimalni za rješavanje zadanog problema. *MVC* radni okviri kao što su *Express.js*, *Adonis.js* i *Sails.js* rastavljaju aplikaciju na modele, prikaze i upravljače. Time su idealni za lako održavanje aplikacije. REST API radni okviri, kao što je *LoopBack*, koriste unaprijed napravljene arhitekture za brže i lakše stvaranje mrežnih aplikacija. *Full-stack* radni okviri su prikaz punog potencijala *Node.js-a*. Koristeći razne biblioteke, pakete i razvojne dodatke, omogućuju kreiranje aplikacija u pravom vremenu (eng. *real-time*) te istodobno povezuju *frontend* i *backend*. Jedan od primjera takvog sveobuhvatnog radnog okvira je *Meteor.js*. Uspoređujući ga s ostalim radnim okvirima, može se vidjeti koliko je *Meteor.js* napredniji u smislu povezanosti s *MongoDB* bazom podataka, korištenjem JavaScript skriptnog jezika na *frontend-u* i *backend-u* te posjedovanje mogućnosti kreiranje mrežnih aplikacija u pravom vremenu. No *Meteor.js-ova* proširenja i dodaci koje nudi mogu imati i svoje nedostatke. Iako podržava korisne mogućnosti kao što je prebacivanje mrežne aplikacije na mobilni uređaj pomoću sustava *Apache Cordove* ili korištenje usluge *Atmosphere* koja se nalazi na *Meteor Cloud-u*, oni nisu još uvijek do kraja optimizirani, sadrže greške i nisu često ažurirani. Također se, pomoću usporedbe mrežnih aplikacija napravljenim na *Meteor.js-u* i *Express.js-u*, vidi se da je mrežna aplikacija bila brža i jednostavnija za napraviti na *Meteor.js-u*, no zauzima više memorije te joj je potrebno više vremena da se učita. Razlog tome su opravo dodaci i proširenja koje *Meteor.js* sadrži, a ne koristi. *Meteor.js* izomorfni razvojni ekosustav je odličan za kreiranje prototipa aplikacija i aplikacija u stvarnom vremenu, ali za aplikacije kojima je potrebna brzina i optimiziranost je bolje koristiti radne okvire koji su jednostavniji i specijalizirani za određeni problem.

LITERATURA

[1] J. Johnston, „A beginners guide to web application development“, dostupno na: https://www.budibase.com/blog/web-application-development/, pristupljeno: 1.9.2021.

[2] OpenJS Foundation, „Node.js“, dostupno na: https://nodejs.org/en/, pristupljeno: 1.9.2021.

[3] Meteor, „Meteor.js“, dostupno na: https://www.meteor.com/, pristupljeno: 1.9.2021.

[4] OpenJS Foundation, „Express.js“, dostupno na: https://expressjs.com/, pristupljeno: 1.9.2021.

[5] OpenJS Foundation, „Introduction to Node.js“, dostupno na: https://nodejs.dev/learn, pristupljeno: 8.6.2021.

[6] The Linux Foundation, „2018 Node.js User Survey Report“, dostupno na: https://nodejs.org/en/user-survey-report/, pristupljeno: 8.6.2021.

[7] A. Rana, H. Virk, „Adonis.js“, dostupno na: https://adonisjs.com/, pristupljeno: 9.6.2021.

[8] F. Mendes, „Why-adonisjs“, dostupno na: https://github.com/adonisjs/legacy.adonisjs.com/blob/develop/docs/pages/why-adonisjs.md, pristupljeno: 9.6.2021.

[9] MongoDB, Inc., „MongoDB“, dostupno na: https://www.mongodb.com/, pristupljeno: 9.6.2021.

[10] Tekreliance, „Get Feature-rich and scalable enterprise application developed with ExpressJS“, dostupno na: https://tekreliance.com/express-js-development/, pristupljeno: 9.6.2021.

[11] IBM / StrongLoop, „LoopBack 4“, dostupno na: https://loopback.io/, pristupljeno: 9.6.2021.

[12] IBM / StrongLoop, „LoopBack 4“, dostupno na: https://loopback.io/doc/en/lb4/index.html, pristupljeno: 9.6.2021.

[13] IBM, „IBM Cloud“, dostupno na: https://www.ibm.com/cloud, pristupljeno: 9.6.2021.

[14] Meteor, „Meteor Cloud“, dostupno na: https://www.meteor.com/cloud, pristupljeno: 9.6.2021.

[15] The Sails Company, „Sails.js“, dostupno na: https://sailsjs.com/, pristupljeno: 9.6.2021.

[16] The Sails Company, „Take as much or as little as you need“, dostupno na: https://sailsjs.com/features, pristupljeno: 9.6.2021.

[17] The Sails Company, „Waterline“, dostupno na: https://waterlinejs.org/, pristupljeno: 9.6.2021.

[18] R. Bovell, „Learn Meteor.js Properly“, dostupno na: http://javascriptissexy.com/learn-meteor-js-properly/, pristupljeno: 9.6.2021.

[19] The Apache Software Foundation, „Apache Cordova“, dostupno na: https://cordova.apache.org/, pristupljeno: 9.6.2021.

[20] The Bakery, „Meteor DevTools“, dostupno na: https://chrome.google.com/webstore/detail/meteor-devtools/ippapidnnboiophakmmhkdlchoccbgje, pristupljeno: 9.6.2021.

[21] Google, „Google Chrome“, dostupno na: https://www.google.com/intl/hr/chrome/, pristupljeno: 9.6.2021.

[22] S. Morris, „Tech 101: What Is JavaScript?“, dostupno na: https://skillcrush.com/blog/javascript/, pristupljeno: 9.6.2021.

[23] M. Wilson, „MongoDB Explained in 5 Minutes or Less“, dostupno na: https://www.credera.com/insights/mongodb-explained-5-minutes-less, pristupljeno: 10.6.2021.

[24] Meteor, „Cordova: How to build mobile apps using Meteor's Cordova integration.“, dostupno na: https://guide.meteor.com/cordova.html, pristupljeno: 10.6.2021.

[25] Meteor, „Introduction: The guide for deploying, scaling, and managing Meteor apps on Meteor Cloud“, dostupno na: https://cloud-guide.meteor.com/, pristupljeno: 10.6.2021.

[26] Meteor, „Application Structure: How to structure your Meteor app with ES2015 modules, ship code to the client and server, and split your code into multiple apps.“, dostupno na: https://guide.meteor.com/structure.html#javascript-structure, pristupljeno: 10.6.2021.

[27] Google Developers, „Chrome DevTools“, dostupno na: https://developer.chrome.com/docs/devtools/, pristupljeno: 10.6.2021.

[28] Broadcom, „Blazemeter“, dostupno na: https://www.blazemeter.com/, pristupljeno: 10.6.2021

SAŽETAK

U sklopu ovog završnog rada objašnjeno je što je to *Node.js* radni okvir te je napravljena usporedba odabranih radnih okvira s *Meteor.js-om*. *Meteor.js* izomorfni razvojni ekosustav je opširnije opisan, prikazujući njegove prednosti i nedostatke. Napravljana je mrežna aplikacija za razgovor kako bi se najbolje prezentirale mogućnosti koje *Meteor.js* nudi. Aplikacija je zatim uspoređena sa sličnom aplikacijom napravljenoj u *Express.js* radnom okviru kako bi se na konkretnom primjeru testirale performanse *Meteor.js-a* u odnosu na konkurenciju.

Ključne riječi: *Node.js,* radni okvir, *Meteor.js,* JavaScript, *MongoDB,* mrežna aplikacija.

Basic principles of Meteor.js isomorphic framework

ABSTRACT

In this final paper is explained what the Node.js framework is and the selected frameworks are compared with Meteor.js. The Meteor.js isomorphic development ecosystem is described in more details, showing its advantages and disadvantages. An online chat application was created to best present the capabilities that Meteor.js offers. The application is then compared to a similar application created in the Express.js framework in order to test the performance, on a specific example, of Meteor.js in relation to the competition.

Keywords: Node.js, framework, Meteor.js, JavaScript, MongoDB, web application.

ŽIVOTOPIS

Petar Barišić je rođen 2.6.1999. godine u Đakovu. Pohađao je Osnovnu školu Vladimira Becića u Osijeku. Nakon završene osnovne škole, 2014. godine se upisuje u Isusovačku klasičnu gimnaziju s pravom javnosti u Osijeku. Nakon završetka gimnazije, 2018. godine se upisuje na sveučilišni preddiplomski studij Računarstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. 2019. godine je sudjelovao kao demonstrator na 3. LABUS sajmu te je član studentskoga zbora.

Petar Barišić

_____

PRILOZI

Prilog P.4.1. – Programski kod *Meteor.js* aplikacije, nalazi se u digitalnom formatu na CD-u priloženom u mapi pod nazivom *meteor_chat.*

Prilog P.4.2. – Programski kod *Express.js* aplikacije, nalazi se u digitalnom formatu na CD-u priloženom u mapi pod nazivom *expressChat.*