SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
Ak. god. 2016./ 2017.

Robert Pavlinić

Izrada web aplikacije za pregledavanje i izradu rasporeda

Završni rad

Mentor: dr.sc. Kristina Kocijan, doc.

Zagreb, 2017.

Sadržaj

Sažetak

U ovom radu opisan je proces razvoja web aplikacije za izradu rasporeda za kolegije Filozofskog fakulteta. Cijela aplikacija, kako *back-end*, tako i *front-end* izrađeni su pomoću programskog jezika *JavaScript*. U izradi su dodatno korištene aktualne tehnologije za razvoj web aplikacija: Node.js i Express razvojni okvir te React programska knjižnica. Uz opis procesa izrade, dan je i kratak opis korištenih tehnologija. Rezultat je REST API i Single Page Aplikacija (SPA) koja pokušava riješiti neke probleme trenutačnog sustava rasporeda iz perspektive studenta.

Ključne riječi: *raspored, web aplikacija, JavaScript, Node.js, SPA*

Development of a web application for viewing and creating schedules

Abstract

This paper describes the process of developing a web application for schedule creation for courses at the Faculty of Humanities and Social Sciences. The complete application, the back-end, as well as the front-end was developed using the JavaScript programming language. Current web development technologies, such as: Node.js and Express framework and React library were used during the development. A short description of these technologies was given along with the description of the development process. The final result is a REST API and a Single Page Application (SPA) which attempts to solve some of the issues of the current schedule system.

Key words: *schedule, web application, JavaScript, Node.js, SPA*

1. Uvod

U ovom radu opisat će se proces razvoja web aplikacije za izradu rasporeda na Filozofskom fakultetu.

Naime, kako na Filozofskom fakultetu postoji mnogo različitih kombinacija kolegija koji se mogu pohađati, na početku svakog semestra, pri upisu, studenti često imaju problema s usklađivanjem kolegija kako ne bi došlo do kolizija.

Sadašnji način prikaza rasporeda je pomalo nepregledan i vizualno neprivlačan, no posebno bih istaknuo nedostatak funkcionalnog pretraživanja. Za brže pretraživanje korisnici moraju koristiti funkcionalnost pretrage svog web preglednika što nije optimalno za brzo dobivanje relevantnih podataka o rasporedu kolegija.

Također, trenutno rješenje nije najprikladnije za mobilne uređaje gdje se još više ističe problem nepreglednosti i nemogućnosti pretraživanja zbog manjih dimenzija ekrana mobilnih uređaja.

S druge strane, sadašnji sustav ima i neke prednosti kao što su brzina i izrazito mala količina prenesenih podataka prilikom zahtjeva za raspored određene studijske grupe, no smatram da su, zbog napretka tehnologije i jeftinijih paketa podatkovnog prometa i rasprostranjenosti bežičnih mreža, manja povećanja u veličini stranice i smanjenje brzine prihvatljivi kompromisi s ciljem poboljšanja iskustva krajnjih korisnika, tj. studenata.

Glavni cilj ovog rada je izrada aplikacije za pregledavanje i izradu rasporeda koja će pokušati riješiti navedene probleme pružajući studentima prikladniji način pregledavanja rasporeda, a u isto vrijeme i istražiti i upoznati se s aktualnim tehnologijama na području razvoja web aplikacija. Gotova aplikacija bi stoga trebala biti brza, pouzdana, jednostavna i intuitivna za korištenje te prilagodljiva svim veličinama ekrana.

Nakon kratkog uvoda gdje ću objasniti svrhu aplikacije i ideju, u drugom poglavlju pobliže ću opisati tehnologije korištene u razvoju. Korištene tehnologije baziraju se uglavnom na JavaScript programskom jeziku kako bi se skratilo vrijeme razvijanja te uklonila potreba za čestom promjenom konteksta do koje dolazi korištenjem raznih programskih jezika različitih sintaksi. Glavne korištene tehnologije su: Node.js i Express razvojni okvir, React programska knjižnica te MySQL sustav za upravljanje bazom

podataka. Uz njih korištene su još neke tehnologije poput Webpack-a i Babel-a, no kako one nisu nužne za razvijanje sličnih aplikacija, već se koriste samo kako bi dodatno olakšale razvoj, neće se opširnije opisivati u nastavku rada. U trećem poglavlju opisat ću proces razvoja same aplikacije, krajnji rezultat te dati kratki osvrt na cijeli proces s prijedlozima mogućih poboljšanja. Na samom kraju iznijet ću zaključak te popis korištene literature.

2. Korištene tehnologije

U ovom poglavlju ukratko ću opisati tehnologije korištene u razvoju web aplikacije za izradu rasporeda. Kroz potpoglavlja ću proći kroz JavaScript, a zatim kroz *back-end* i *front-end* tehnologije aplikacije.

2.1. JavaScript

JavaScript je visoko-razinski programski jezik koji se interpretira te podupire mnoge programske paradigme kao što su: objektno-orijentirano, imperativno i deklarativno programiranje („JavaScript“, 2017). JavaScript je najpoznatija implementacija ECMAScript specifikacije. Posljednje izdanje specifikacije je ES7 (ECMAscript 7) te od 2015. godine svake godine izlazi novo izdanje s novim dodacima u svrhu njezinog poboljšanja i modernizacije („ECMAScript“, 2017).

JavaScript se najviše upotrebljava kao skriptni jezik u web preglednicima te moderni preglednici uglavnom u potpunosti podržavaju barem ES6, tj. ECMAScript 2015 specifikaciju. Kako su web preglednici neki od najkorištenijih programa, bilo na računalu ili na pametnim telefonima, danas je i JavaScript jedan on najpopularnijih i najkorištenijih programskih jezika na svijetu. Za manipuliranje HTML dokumentom u web pregledniku koristi se Document Object Model (DOM) API web preglednika koji povezuje web stranice s programskim jezicima („Document Object Model (DOM)“, bez dat.).

JavaScript je od svoje prve pojave 1995. godine sve do nedavno imao lošu reputaciju. Dijelom jer je smatran "igračkom" za animacije ili slične stvari na web stranicama, a dijelom jer se neki dijelovi JavaScripta razlikuju od drugih programskih jezika zbog toga što je JavaScript implementiran u vrlo kratkom vremenskom roku, svega 10 dana („A Short History of JavaScript - Web Education Community Group“, bez dat.). Upravo ga se zbog toga smatra najneshvaćenijim programskim jezikom na svijetu („JavaScript: The World’s Most Misunderstood Programming Language“, bez dat.). S vremenom se povećavao broj korisnika web preglednika i broj web aplikacija te su proizvođači web preglednika sve više i više optimizirali svoje JavaScript interpretere.

Stoga je i JavaScript sve više i više sazrijevao kao programski jezik pogodan i za ozbiljnije aplikacije. Tako se od 2009. godine i pojave Node.js-a, JavaScript, osim na klijentskoj strani, tj. web pregledniku, može koristiti i na poslužiteljskoj strani za razvijanje aplikacija web servera, pa čak i za razvijanje cijelih aplikacija za razne *desktop* platforme uz pomoć razvojnih okvira poput Electrona[1].

U izradi ove aplikacije JavaScript se koristio u obje varijante, na klijentskoj strani uz pomoć programske knjižnice React i na poslužiteljskoj strani uz pomoć Node.js *runtimea* i Express razvojnog okvira. Također, koriste se već navedeni alati koji olakšavaju i ubrzavaju razvoj JavaScript aplikacija:

- Babel - JavaScript kompajler koji omogućava korištenje sintakse najnovijeg ECMAScript standarda u okruženjima gdje ona nije podržana[2];
- Webpack - služi za objedinjavanje koda podijeljenog u module kako bi ga se lakše prenijelo preko mreže[3].

Kako ovi alati nisu obavezni za izradu aplikacija u programskom jeziku JavaScript, u nastavku se neće opširnije opisivati, već su navedeni samo u svrhu reference.

2.2. Back-end

Na *back-endu,* tj. serverskoj strani aplikacije korištene su sljedeće tehnologije: Node.js, Express i MySQL.

2.2.1. Node.js

Node.js je *JavaScript runtime* izrađen na temelju Google Chromeovog V8 JavaScript enginea. Glavna odlika Node.js-a je što koristi ne-blokirajući *input/output* model. Ne-blokirajući znači da kada program naiđe na operaciju za koju se ne zna točno

1 "Electron," *Electron*, posjećeno 19.08.2017., https://electron.atom.io/.
2 "Babel · The Compiler for Writing next Generation JavaScript," posjećeno 19.08.2017., https://babeljs.io/.
3 "Webpack," posjećeno 19.08.2017., https://webpack.js.org/.

kada će se izvršiti, program se nastavlja izvršavati dalje, a ta operacija se šalje na izvršavanje izvan konteksta glavnog programa. Kada se operacija izvrši, pokreće se tzv. *callback* funkcija, tj. funkcija koja je dana kao argument pozadinskoj operaciji. Takav način razvijanja pogodan je za mrežne aplikacije i podržava mnogo istovremenih povezivanja (Foundation, bez dat.).

Uz instalaciju Node.js-a, automatski se instalira i npm - upravitelj paketima, tj. programskim knjižnicama i razvojnim okvirima za Node.js. Za rad s npm-om koristi se tekstualno sučelje i razne naredbe poput *npm install, npm init,* itd. Prednost korištenja npm upravitelja paketima je ta što su svi paketi koji su potrebni za rad neke aplikacije navedeni u posebnoj *package.json* datoteci koja se kreira uz pomoć *npm init* naredbe. Tako je prilikom dijeljenja aplikacije potrebno priložiti samo *package.json* datoteku umjesto svih modula i programskih knjižnica koje su potrebne za rad aplikacije što znatno smanjuje veličinu aplikacije. Također, potrebni paketi su definirani prema svojim izdanjima što uklanja poteškoće do kojih bi došlo prilikom usklađivanja izdanja („01 - What is npm? | npm Documentation“, bez dat.). npm je jedan od najpopularnijih upravitelja paketima za programske jezike na svijetu, te broji daleko više raspoloživih paketa od ostalih upravitelja paketima za druge programske jezike. npm repozitorij se svakim danom povećava za prosječno 460 novih paketa („Modulecounts“, bez dat.).

2.2.2. Express

Express je relativno malen razvojni okvir koji radi na temelju Node.js-ove funkcionalnosti web servera pojednostavljujući njegov API. Na taj joj način dodaje nova svojstva te tako bitno olakšava izradu i organizaciju web aplikacija („liveBook - Express in Action: Writing, building, and testing Node.js applications - Chapter 1. What is Express?“, bez dat.). Za izradu ove aplikacije korišten je primarno zbog svoje jednostavnosti i minimalizma te mogućnosti integracije s različitim tehnologijama pružajući brz i jednostavan način za izradu potrebnog API-ja. Sama osnova Express

razvojnog okvira je izrazito mala te se po potrebi dodaju moduli za željene funkcionalnosti poput rada s kolačićima, rada sa sesijama i sl.

2.2.3. MySQL

MySQL je open source sustav za upravljanje relacijskom bazom podataka (RDBMS) („MySQL“, 2017). MySQL je najčešće korišten kao dio LAMP (Linux, Apache, MySQL, PHP) razvojnog okruženja („What is LAMP (Linux, Apache, MySQL, PHP)?“, bez dat.), dok je s druge strane MongoDB baza podataka koja je najčešće korištena u kombinaciji s Node.js-om u tzv. MEAN (MongoDB, Express, Angular, Node.js) razvojnom okruženju („What is MEAN (MongoDB, Ember, Angular, Node)?“, bez dat.). Iako su navedene kombinacije najučestalije, većina tehnologija se može zamijeniti za neku drugu prikladniju, ovisno o projektu. Tako je za izradu ove aplikacije korišten MySQL sustav jer se u podatcima o kolegijima i terminima jasno mogu prepoznati relacije koje se tada lako mogu prikazati kroz bazu podataka. Za izradu samih tablica korišten je alat phpMyAdmin za administraciju MySQL baze preko weba[4].

2.3. Front-end

Za izradu front-enda, tj. korisničkog sučelja aplikacije, korištena je React programska knjižnica. React je JavaScript programska knjižnica za izradu korisničkih sučelja koju je razvio Facebook („React - A JavaScript library for building user interfaces“, bez dat.). React je uz Angular, Vue.js, Meteor.js i druge samo jedno od mogućih rješenja za razvoj Single Page aplikacija. No, React se ponešto razlikuje od ostalih navedenih rješenja po tome što ostala rješenja dolaze s dodacima koja ih čine kompletnim razvojnim okvirima (dodaci poput ugrađenog modula za usmjeravanje, modula za navigaciju i sl.) dok je React samo programska knjižnica za izradu korisničkih sučelja te se za dodatnu funkcionalnost mora kombinirati s dodatnim programskim knjižnicama.

4 phpMyAdmin contributors, “PhpMyAdmin,” *PhpMyAdmin*, posjećeno 19.08.2017., https://www.phpmyadmin.net/.

Najbitniji koncept na kojemu je React baziran su zasebni dijelovi koda koji se nazivaju komponente. Komponente su neovisni, ponovno iskoristivi dijelovi korisničkog sučelja, usporedivi s funkcijama koje mogu primati proizvoljne argumente, a vraćati proizvoljne HTML elemente.

Dodatna stvar koja je bitna, iako ne i nužna pri korištenju Reacta je JSX. JSX je Facebookov dodatak ECMAScriptu sa sintaksom sličnom XML-u. Nije namijenjen za standardizaciju već za korištenje s kompajlerima kao što su već navedeni Babel i slični („JSX | XML-like syntax extension to ECMAScript“, bez dat.). Svrha JSX-a je definiranje sintakse za definiranje struktura stabla s atributima - slično kao kod XML elemenata, kako bi se olakšala izrada komponenata s Reactom.

Kako imena metoda i nekih svojstava objekata i klasa (poput *render, constructor, props, state*, i dr.) koja će se pojavljivati u nastavku rada nisu korinički definirana, već su unaprijed zadana, na njih ću se referirati u njihovom izvornom engleskom obliku. Najjednostavniji način definiranja komponente koristeći JSX je putem jednostavne funkcije:

```
const Component = props => (
     <h1> Bok, {props.ime} </h1>
 )
```

Za definiranje komponente u ovom primjeru korištena je ES6 sintaksa tzv. *arrow* funkcija koja implicitno vraća vrijednost ukoliko je u definiciji funkcije jedan izraz. Imena komponenti u Reactu započinju velikim slovom. Poseban argument u React komponentama je objekt *props* (eng. *properties*), tj. svojstva putem kojeg komponenta prima neke podatke koje tada vraća kao povratnu vrijednost funkcije. JSX dio povratne vrijednosti omogućava lakši način definiranja strukture komponente. Svaki element u JSX-u mora imati kosu crtu za zatvaranje ili pripadajuću zatvarajuću oznaku. Također, ukoliko funkcija vraća više JSX elemenata, svaki mora biti ugniježđen u jednom vršnom elementu,

najčešće elementu <*div*>. Unutar JSX-a se može izvršavati i JavaScript kod koji mora biti unutar vitičastih zagrada - u ovom primjeru je to vrijednost *ime* objekta *props.*

Ukoliko je potrebno definirati složeniju komponentu koja sadrži pripadajuće podatke i ima neko stanje (eng. *state),* preporuča se korištenje ES6 klasne sintakse:

```
class Component extends React.Component {
        constructor(){
            super()
            this.state = {
                ime:'Ivan'
        }
    }
        render(){
            return(
            <h1>Bok, {this.state.ime}</h1>

        )
    }
}
```

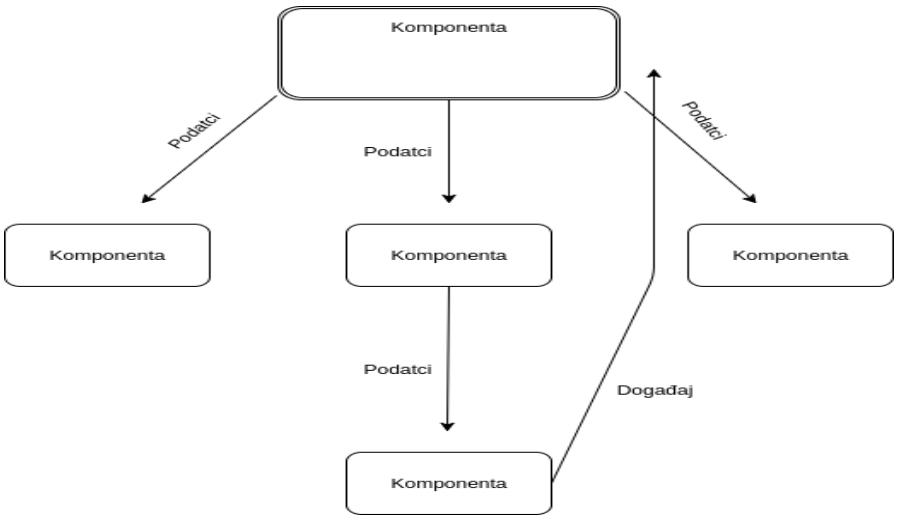
Za vraćanje JSX elemenata pri korištenju klasne sintakse, potrebno je koristiti *render*[5] metodu. Za definiranje posebnog *state* objekta komponenti, preporuča se koristiti *constructor*[6] metoda koja se poziva pri kreiranju novog objekta.

Tipična React aplikacija sastoji se od manjeg broja složenih, tj. *pametnih* komponenti koje pružaju potrebne podatke prezentacijskim, jednostavnim komponentama koje tada te podatke prikazuju. Takav tip protoka podataka, iz *state-a* složenijih komponenata prema *props-ima* jednostavnih komponenti naziva se *one-way-binding* ili *one-way data flow* („Thinking in React - React“, bez dat.) a njezin dijagram je prikazan na slici 1.

---

5 *render* metoda koristi se za definiranje JSX strukture koja će u korisničkom sučelju predstavljati klasu čija je ta metoda

6 *constructor* metoda nije dio React programske knjižnice već je standardna metoda ES6 klasne sintakse koje se koristi za definiranje početnih vrijednosti svojstava objekta te je stoga idealno mjesto za definiranje početnog *state* objekta React komponente



Slika 1: prikaz protoka podataka u React-u

Složene komponente definiraju i metode koje se također prenose do prezentacijskih komponenti putem *props-a.* Te metode reagiraju na događaje poput klikova mišem, pritiska tipki na tipkovnici i sl. te se kroz njih mijenja *state* u vršnoj složenoj komponenti. Nakon promjene *state-a,* prezentacijske komponente se automatski osvježavaju kako bi prikazale promjene u podatcima. Osim automatskog osvježavanja, *React* komponente imaju definirane i još neke metode, tzv. *lifecycle* metode koje se pozivaju pri određenom koraku u procesu prikazivanja komponente, kao npr. netom prije osvježavanja komponente - *ComponentWillUpdate(),* netom nakon uklanjanja komponente - *ComponentWillUnmount()* i druge.

3. Proces izrade

U ovom poglavlju opisat će se proces izrade aplikacije, počevši od strukture, preko same izrade sve do završnog rezultata. Kroz svako potpoglavlje opisat će se struktura i sama izrada jednog od tri glavna dijela aplikacije: baze podataka, API-ja i SPA. Osim kratkih isječaka programskog koda kroz sam rad, programski kod je u potpunosti dostupan na adresi: https://github.com/rpavlini/ffzg-raspored.

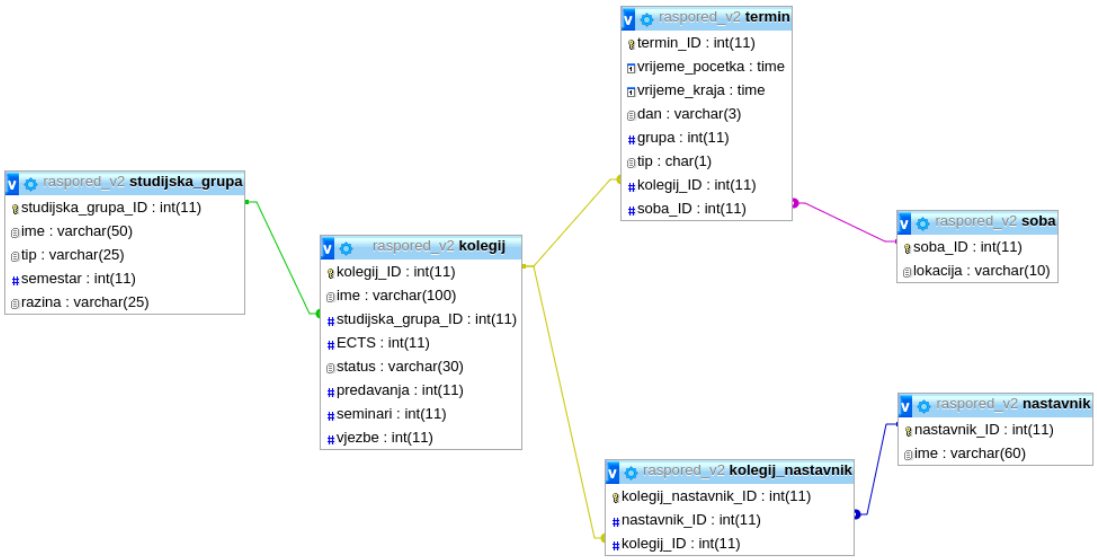
3.1. Baza podataka

3.1.1. Struktura baze podataka

Na samom početku potrebno je izraditi shemu podataka kojima aplikacija raspolaže u svom radu. Podatci su podijeljeni na sljedeće entitete:

- studijska_grupa
- kolegij
- termin
- soba
- nastavnik.

Uz pomoć već navedenog alata, phpMyAdmin-a i njegovog alata za dizajn, dobiven je vizualni prikaz entiteta i njihovih relacija prikazan na slici 2. Entiteti su povezani relacijama preko svojih identifikacijskih brojeva. Sve relacije su jedan-prema-više (eng. *one-to-many*) osim relacije više-prema-više (eng. *many-to-many)* među entitetima kolegij i nastavnik koja je ostvarena uz pomoć tablice spoja kolegij_nastavnik.



Slika 2: prikaz sheme baze podataka

3.1.2. Izrada baze podataka

Za potrebe razvoja aplikacije, podatci o kolegijima i terminima pribavljeni su uz pomoć *web-scraping* skripte napisane u Python 2.7 programskom jeziku. Skripta koristi modul *requests*[7] za slanje HTTP zahtjeva na službene web stranice rasporeda Filozofskog fakulteta i modul *BeautifulSoup*[8] za obradu primljenog HTML dokumenta. Uz modul *BeautifulSoup* koristi se i standardni *Python* modul html5lib za parsiranje HTML dokumenta. Preuzeti HTML dokument se parsira i obrađuje te se rezultati spremaju u Python listu - *raspored* čiji su elementi Python rječnici - *studijske grupe*. Rječnik *studijske grupe* sadrži listu *kolegiji* čiji su elementi Python rječnici - *kolegiji* te na kraju *kolegiji* sadrže rječnike *termina* pohranjene u listu *termini*.

Za izradu samih tablica baze podataka korišteno je grafičko sučelje već prije navedenog phpMyAdmin alata.

7 "Requests: HTTP for Humans — Requests 2.18.4 Documentation," posjećeno 19.08.2017., http://docs.python-requests.org/en/master/.

8 "Beautiful Soup Documentation — Beautiful Soup 4.4.0 Documentation," posjećeno 19.08.2017., https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

Za povezivanje s MySQL bazom korišten je službeni *mysql.connector*[9] kojeg je razvio sam MySQL. Nakon što je ostvarena veza s bazom podataka, iterira se kroz navedenu listu *raspored* te se svaka studijska grupa s pripadajućim ključevima i vrijednostima pohranjuje u bazu. Zatim se iterira kroz svaki *kolegij* u listi kolegija te na kraju kroz svaki *termin* u listi termina. Pri pohranjivanju *termina* u bazu podataka, provjerava se je li lokacija termina već pohranjena u bazi podataka kako ne bi došlo do duplikata.

Za ovaj zadatak odabran je programski jezik Python zbog lakoće obrade stringova i zbog njegove ekspresivnosti. Zbog mogućih promjena u HTML dokumentu rasporeda, ovaj način pribavljanja podataka nije najpouzdaniji. Također, zbog nedosljednosti u formatiranju podataka, točnije podataka o profesorima koji predaju određen kolegij, ti podatci su izostavljeni.

3.2. API

3.2.1. Struktura API-ja

Aplikacijsko programsko sučelje (eng. a*pplication programming interface*) definira način komunikacije između različitih softverskih komponenti („What is RESTful API?“, bez dat.) te će u ovoj aplikaciji REST API biti način kako će front-end tražiti podatke od back-enda. REST API podatke iz baze pruža u JSON formatu, pogodnom za obradu i daljnje korištenje na front-endu.

Definirane su sljedeće API krajnje točke :

- api/kolegiji/{predd | dipl}
  - dohvaćanje podataka o svim kolegijima preddiplomske ili diplomske razine
- api/studijske_grupe/{predd | dipl}

9 “MySQL :: Download Connector/Python,” posjećeno 19.08.2017., https://dev.mysql.com/downloads/connector/python/.

- dohvaćanje podataka o svim studijskim grupama s pripadajućim kolegijima preddiplomske ili diplomske razine

- api/kolegiji/{predd | dipl}/:id
  - dohvaćanje pojedinog kolegija preddiplomske ili diplomske razine na temelju identifikacijskog broja kolegija
- api/studijske_grupe/{predd | dipl}/:id
  - dohvaćanje pojedine studijske grupe preddiplomske ili diplomske razine na temelju identifikacijskog broja studijske grupe.

Osim ovdje navedenih krajnjih točaka za pristup podatcima za potrebe ove aplikacije, bilo bi moguće i dodavanje dodatnih krajnjih točaka za razne potrebe, npr. krajnje točke za pristup podatcima svim lokacijama gdje se održavaju termini kolegija ili krajnje točke za pristup svim terminima određenog dana u tjednu i sl.

3.2.2. Izrada API-ja

Za izradu API-ja korišten je razvojni okvir Express zbog lakoće kreiranja potrebnih krajnjih točaka. Za komunikaciju s bazom podataka putem JavaScripta korišten je MySQL2 modul[10]. Zbog male veličine same datoteke nije izvršena podjela prema popularnoj MVC paradigmi arhitekture aplikacija[11]. Krajnje točke definirane su kao metode instance *express* objekta. Primjer definiranja jedne krajnje točke:

```
app.get('/api/studijske_grupe/predd',     (request,
response)=> {

    /*SQL za dohvaćanje podataka iz baze

    ...
```

10 Andrey Sidorov, *Node-Mysql2: :Zap: Fast Node-Mysql Compatible Mysql Driver for Node.Js*, JavaScript, (2013; repr., 2017), posjećeno 19.08.2017., https://github.com/sidorares/node-mysql2.
11 "What Is Model-View-Controller (MVC)? - Definition from WhatIs.Com," *WhatIs.Com*, posjećeno 19.08.2017., http://whatis.techtarget.com/definition/model-view-controller-MVC.

```
    */
    conn.query(sql_studijske,   (error,   results,
fields)=>{
    if (error) {
        response.send('Došlo je do greške')
    }
    response.send(JSON.stringify(results))
    })
})
```

U ovom primjeru prepoznaje se klasičan način rada s Node.js-om putem tzv. *callback* funkcija. Prilikom zaprimanja HTTP GET zahtjeva na navedeni URL, poziva se zadana anonimna funkcija definirana u samoj *app.get* metodi. U anonimnoj funkciji se zatim poziva asinkrona metoda *query conn* objekta koja kada vrati neki rezultat, bila to greška ili rezultat SQL upita, šalje odgovor na početni HTTP zahtjev putem *send* metode *response* objekta.

Na sličan način definirane su i ostale krajnje točke za dohvaćanje podataka o rasporedu iz baze podataka. Također, definirana je i krajnja točka početnog URL-a - */*, tj. *root*-a koja uvijek vraća html dokument koji sadrži samu aplikaciju.

3.3.SPA

3.3.1. Struktura SPA

React single page aplikacija na *front-endu* dohvaća podatke putem API-ja na temelju korisnikovog unosa te se promjenom izgleda korisničkog sučelja i ažuriranjem odražava promjena u raspoloživim podatcima.

Na samom početku, pri učitavanju aplikacije, te na svakoj sljedećoj promjeni željene razine studija (preddiplomska ili diplomska) učitavaju se podatci o kolegijima i studijskim grupama te razine. Pomoću tih podataka postiže se funkcionalnost savjeta (eng. *search hint*) prilikom pretraživanja kolegija ili studijskih grupa.

Pri odabiru željene studijske grupe ili kolegija, putem API-ja se dohvaćaju podatci o tom odabiru kako bi se u korisničkom sučelju prikazalo više detalja.

Ukoliko je odabran kolegij - prikazuju se dodatne informacije samo o odabranom kolegiju. Dodatne informacije su: status kolegija (izborni ili obavezan), broj ECTS bodova koje kolegij donosi, raspodjela predavanja, seminara i vježbi te vrijeme i lokacija izvođenja termina. U slučaju odabira studijske grupe - prikazuju se dodatne informacije o svim kolegijima te studijske grupe.

Osim funkcionalnosti pretrage kolegija i studijskih grupa, aplikacija pruža i mogućnost izrade rasporeda na temelju odabranih kolegija.

Za izradu rasporeda koristi se *Canvas* API koji omogućuje crtanje u HTML5 *Canvas* elementu putem JavaScripta[12]. Odabran je *Canvas* kao način prikaza rasporeda zbog jednostavnosti korištenja. Također, mogućnošću spremanja *Canvas* elementa kao svake druge slike u web preglednicima, na najlakši mogući način je omogućeno spremanje rasporeda na korisničko računalo.

Nakon odabira opcije *Izradi raspored*, u korisničkom sučelju pojavljuju se dva nova elementa, jedan na kojemu su prikazani trenutno odabrani termini kolegija za prikaz na *Canvas* elementu i drugi - sam *Canvas* element. Pri odabiru termina za dodavanje u raspored ili za brisanje iz rasporeda, osvježava se cijeli *Canvas* element, odabrani predmeti se ponovno crtaju te se detektiraju eventualne kolizije koje su tada istaknute crvenom bojom.

3.3.2. Izrada SPA

Front-end aplikacije za izradu rasporeda izrađen je u potpunosti uz pomoć programske knjižnice za izradu korisničkih sučelja - React u programskom jeziku JavaScript.

12 "Canvas API," *Mozilla Developer Network*, n.d., https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API.

Prvi korak u radu s React programskom knjižnicom je podjela korisničkog sučelja na već navedene komponente. U ovom slučaju, korisničko sučelje je podijeljeno na dvije kontejnerske, složene komponente i sedam jednostavnijih, prezentacijskih komponenti. Tako je dobivena sljedeća podjela:

Kontejnerske komponente:

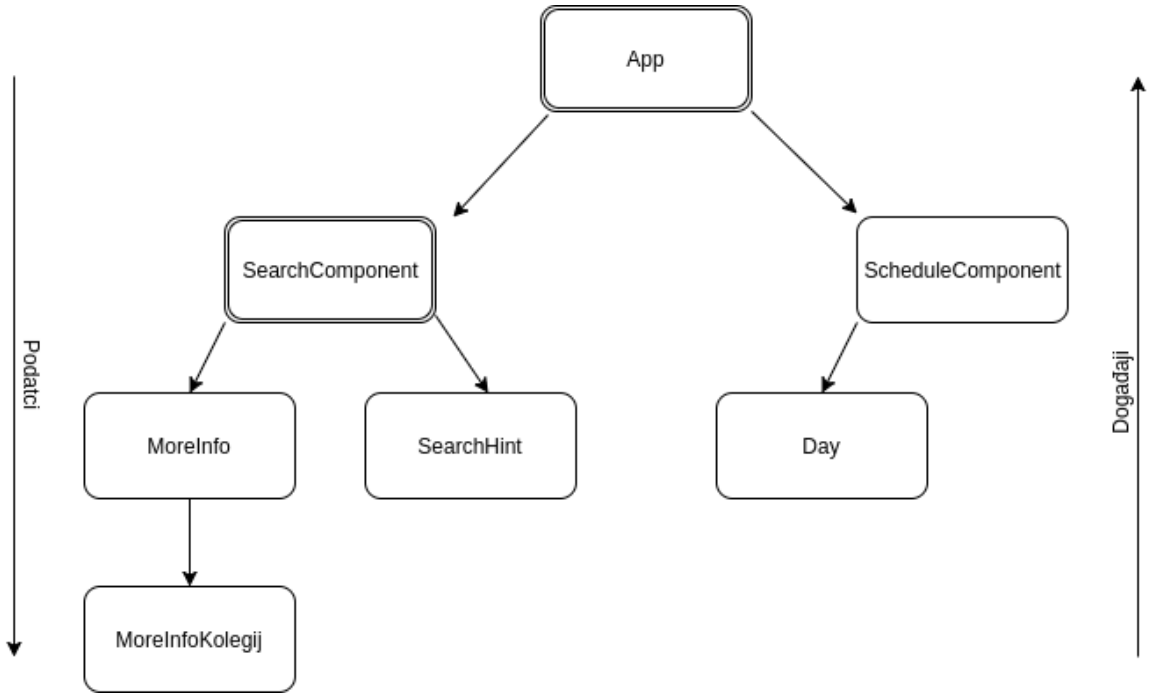
- *App* komponenta
  - App komponenta je temeljna komponenta u cijeloj aplikaciji. Ova komponenta u *stateu* sadrži termine koji su odabrani za prikaz u rasporedu. Komponenta App prosljeđuje svoj *state* na komponentu ScheduleComponent koja ažuriranjem automatski reagira na promjene u *stateu*.
  - Funkciju za promjenu *statea,* tj. dodavanje termina u *state,* komponenta App prosljeđuje na komponentu SearchComponent koja ih dalje prosljeđuje komponenti MoreInfo koja ju naposlijetku prosljeđuje komponenti MoreInfoKolegij.
- *SearchComponent* komponenta
  - SearchComponent komponenta druga je kontejnerska komponenta aplikacije. Ona u svom *stateu* sadrži ponuđene rezultate korisnikovog unosa pri pretraživanju kolegija ili studijskih grupa te zadnji odabir korisnika - bilo studijske grupe ili kolegija.
  - Zadnji odabir SearchComponent komponenta prosljeđuje na prezentacijsku komponentu MoreInfo koja na temelju toga dalje kreira potreban broj instanci MoreInfoKolegij komponente.
  - Pronađeni rezultati korisnikove pretrage prosljeđuju se na komponentu SearchHint koja ažuriranjem reagira na promjene u *stateu*.
  -

Prezentacijske komponente:

- SearchHint
- MoreInfo
- MoreInfoKolegij

- ScheduleComponent
- Day.

Grafički prikaz podjele komponenti dan je na slici 3.



Slika 3: prikaz strukture komponenata

App i SearchComponent komponente svoj *state* prosljeđuju prezentacijskim komponentama za prikaz kroz *propse*. Događaji na prezentacijskim komponentama mijenjaju *state* te se relevantne prezentacijske komponente automatski ažuriraju.

ScheduleComponent je komponenta koja je zbog svoje kompleksnosti lako mogla biti i zasebna kontejnerska komponenta, no zbog lakšeg protoka novih podataka je implementirana kao prezentacijska komponenta.

Ova komponenta odgovorna je za izradu rasporeda na HTML5 *Canvas* elementu koristeći *Canvas* API metode.

Na svaku promjenu *statea* App komponente, ScheduleComponent komponenta reagira iscrtavanjem svih termina koji su joj proslijeđeni na HTML5 *Canvas* element. Termini su tako raspoređeni u tablicu visine 600px i širine 800px.

Tablica se sastoji od sveukupno 21 retka i 6 stupaca. Redci odgovaraju vremenskom periodu od 8:00 do 21:30 podijeljenom na termine po 45 minuta. Stupci odgovaraju danima u tjednu od ponedjeljka do subote te početni stupac za prikaz vremena.

Svakom 45 minutnom terminu i svakom danu pridružene su koordinate kako bi se omogućilo pozicioniranje termina kolegija u tablicu ovisno o početku i završetku određenog termina.

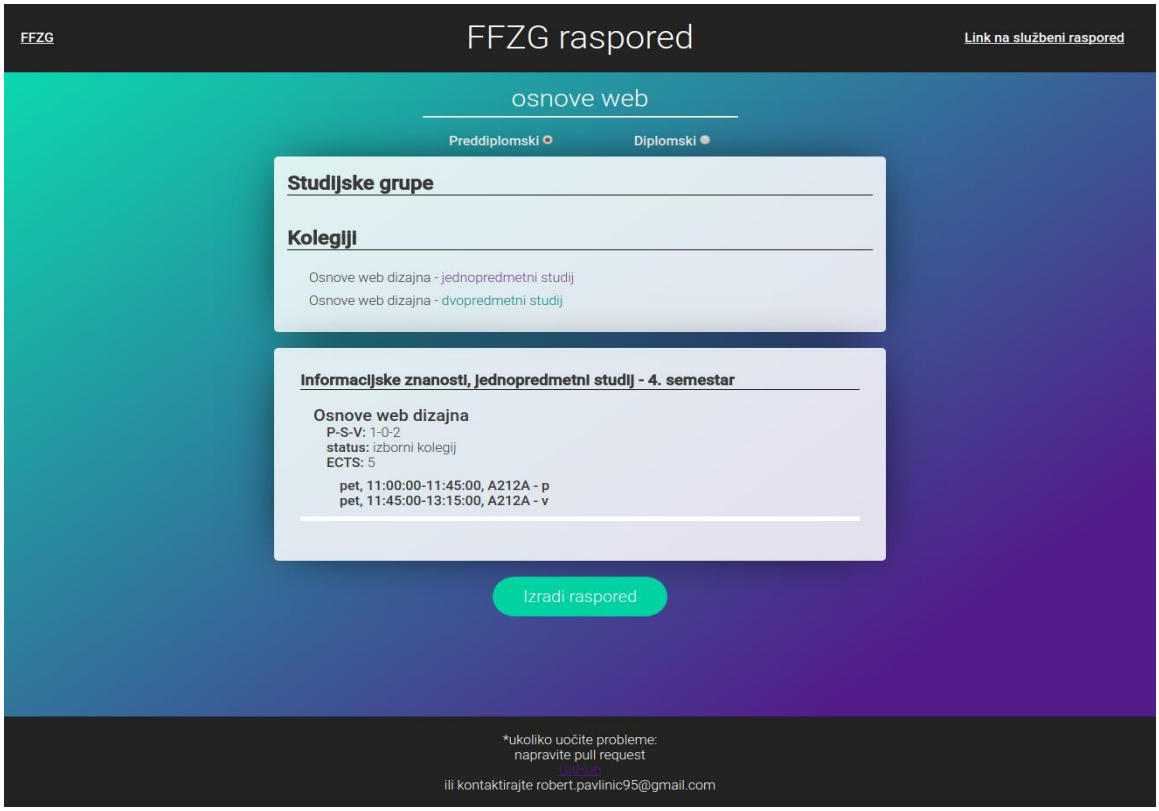
Jedna ćelija u tablici visoka je 30px i široka 150px te odgovara jednom terminu od 45 minuta.

Prikaz termina u tablici sastoji se od imena kolegija kojemu termin pripada, tipa termina - predavanje, seminar ili vježbe te lokacije gdje se termin održava.

3.4. Rezultat

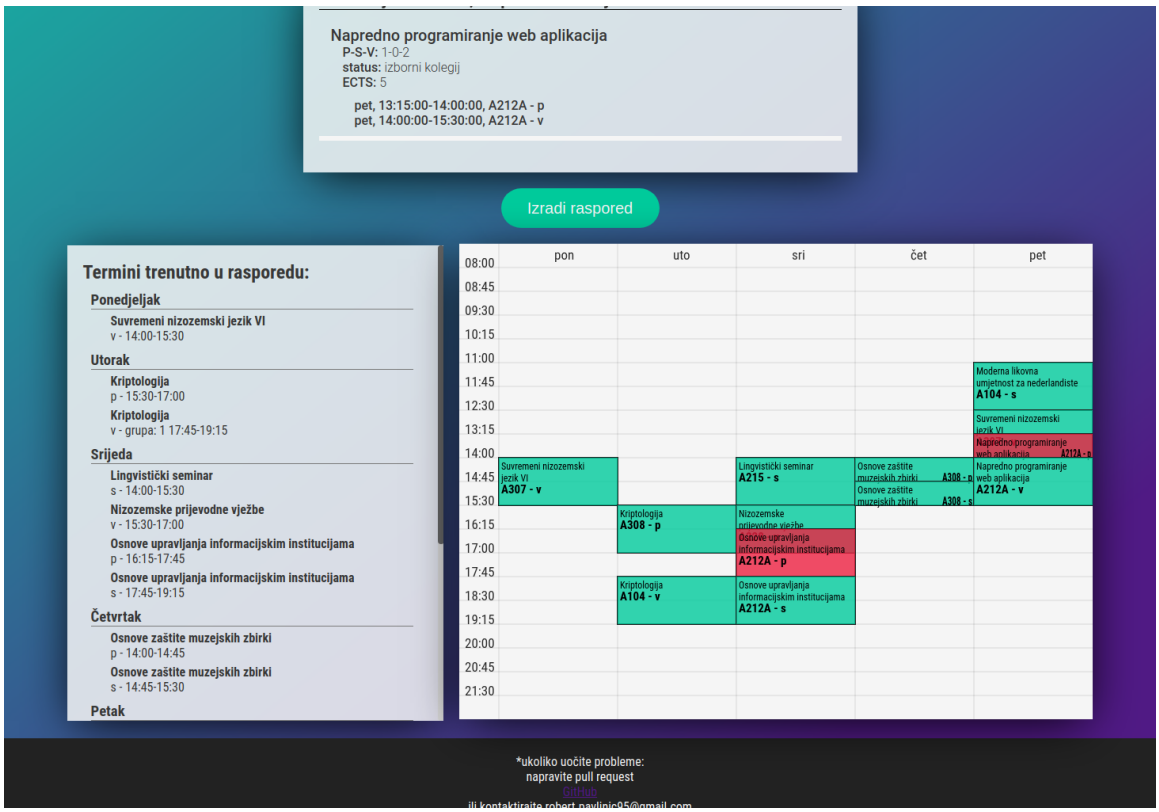
Kada se sva tri glavna dijela (baza podataka, API i SPA) povežu, završni rezultat je kompletna aplikacija za pregled i izradu rasporeda kolegija na Filozofskom fakultetu. Dodatnim dijelom može se smatrati i API zasebno, no ovdje će se promatrati samo u kontekstu aplikacije.

Postignut je zadani cilj aplikacije - pružanje rješenja nekih problema sadašnjeg sustava. Aplikacija pruža mogućnost brze pretrage termina kolegija ili studijske grupe, kako preddiplomske tako i diplomske razine (vidi sliku 4). Primjenom responzivnog dizajna aplikacija je prilagođena i za mobilne uređaje.



Slika 4: prikaz pretrage kolegija u aplikaciji

Također, aplikacija omogućava izradu rasporeda na temelju odabranih termina kolegija te na jednostavan način daje uvid u eventualne kolizije termina u rasporedu (vidi sliku 5).



Slika 5: prikaz izrade rasporeda (kolizije su označene crvenom bojom)

Iako je završni rezultat funkcionalna aplikacija, uvijek postoje načini za poboljšanje. Neka od mogućih poboljšanja su:

- Prilagodba kompletne aplikacije za mobilne uređaje
  - U trenutnoj verziji aplikacije, na mobilnom uređaju dostupna je samo funkcionalnost pretrage termina kolegija i studijskih grupa. Promjenom rasporeda komponenti i orijentacijom *Canvas* elementa dobila bi se mogućnost kreiranja rasporeda i na mobilnim uređajima
- Uređivanje i održavanje koda
  - Moguća su poboljšanja u vidu razdvajanja koda SPA dijela zbog lakšeg održavanja koda, popravljanja *bugova* ili dodavanja novih funkcionalnosti
- Službeni podatci

- Iako je aplikacija funkcionalna i s podatcima dobivenim na način *scrapeanja* službenih web stranica rasporeda Filozofskog fakulteta, uvijek postoji mogućnost malih promjena u HTML-u koje bi mogle dovesti do *bugova* u Python skripti što bi rezultiralo potrebom za konstantnim prilagođavanjem i popravljanjem skripte. Službeni podatci u već odgovarajućem formatu mogli bi riješiti taj problem. No, ni to ne bi bilo idealno riješenje jer su ti podatci tek privremeni te kroz školsku godinu može doći do raznih promjena.

- Izrada općenitog rasporeda
  - Iako su podatci o rasporedu preuzeti sa službenih stranica, uvijek postoji mogućnost promjene nekih termina što može uzrokovati nepodudaranje podataka u bazi i stvarnih podataka. Jedan način rješavanja ovog problema bio bi dodavanje mogućnosti ručne promjene vremena izvršavanja termina. Kao krajnje rješenje bilo bi dodavanje mogućnosti korisnički definiranih termina. Tada bi aplikacija prerasla u općenitu aplikaciju za izradu rasporeda s mogućnošću uvoza podataka o terminima kolegija.
- Ostala poboljšanja
  - Poboljšanja u kontekstu prilagodbe dizajna za sve veličine mobilnih uređaja, prilagođavanje samog dizajna za poboljšanje korisničkog iskustva, prilagodba izgleda generiranog rasporeda i sl.

4. Zaključak

Kroz ovaj rad opisao sam kompletan proces razvoja web aplikacije za pregled i izradu rasporeda na Filozofskom fakultetu počevši od kratkog uvoda u korištene tehnologije sve do strukture i samog procesa izrade aplikacije. Osim razvoja aplikacije svrha ovog rada bila je također i istražiti aktualne tehnologije u svijetu razvoja web aplikacija. Tako sam koristio popularne tehnologije bazirane na JavaScript programskom jeziku kao što su Node.js i React.

Sadašnji način prikaza rasporeda ima nekoliko problema poput: nedostatak funkcionalnog pretraživanja, neprilagođenost korisničkog sučelja mobilnim uređajima i otežan dolazak do relevantnih informacija o terminima kolegija. Probleme sam pokušao riješiti izradom web aplikacije koja bi prikazivala raspored na drugačiji način i dodavanjem funkcionalnosti izrade rasporeda. Cilj sam uglavnom postigao te je izrađena aplikacija intuitivna i prilagodljiva mobilnim uređajima, a dolazak do relevantnih informacija o terminima izvođenja kolegija brži i jednostavniji. Unatoč tome, ostaje mnogo mjesta za poboljšanja koja su uglavnom u kontekstu daljnje prilagodbe dizajna korisničkog sučelja, dodavanja novih funkcionalnosti i dobivanja točnijih informacija o terminima kolegija. Kako je do potpuno točnih informacija o kolegijima teško doći jer se kroz školsku godinu mogu dogoditi razne nepredviđene situacije, jedno od najboljih mogućih poboljšanja bilo bi omogućavanje definiranja proizvoljnih termina te na taj način omogućiti studentima samostalno korigiranje generiranog rasporeda.

5. Literatura

1. 01 - What is npm? | npm Documentation. (bez dat.). Preuzeto 10. rujan 2017., od https://docs.npmjs.com/getting-started/what-is-npm
2. A Short History of JavaScript - Web Education Community Group. (bez dat.). Preuzeto 11. rujan 2017., od https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript
3. Document Object Model (DOM). (bez dat.). Preuzeto 10. rujan 2017., od https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model
4. ECMAScript. (2017, kolovoz 6). U *Wikipedia*. Preuzeto 10. rujan 2017., od https://en.wikipedia.org/w/index.php?title=ECMAScript&oldid=794175102
5. Foundation, N. js. (bez dat.). About. Preuzeto 10. rujan 2017., od https://nodejs.org/en/about/
6. JavaScript. (2017, kolovoz 18). U *Wikipedia*. Preuzeto 10. rujan 2017., od https://en.wikipedia.org/w/index.php?title=JavaScript&oldid=796055675
7. JavaScript: The World's Most Misunderstood Programming Language. (bez dat.). Preuzeto 19. kolovoz 2017., od http://www.crockford.com/javascript/javascript.html
8. JSX | XML-like syntax extension to ECMAScript. (bez dat.). Preuzeto 10. rujan 2017., od http://facebook.github.io/jsx/index.html
9. liveBook - Express in Action: Writing, building, and testing Node.js applications - Chapter 1. What is Express? (bez dat.). Preuzeto 11. rujan 2017., od https://livebook.manning.com/#!/book/express-in-action/chapter-1/35
10. Modulecounts. (bez dat.). Preuzeto 19. kolovoz 2017., od http://www.modulecounts.com/
11. MySQL. (2017, kolovoz 18). U *Wikipedia*. Preuzeto 10. rujan 2017., od https://en.wikipedia.org/w/index.php?title=MySQL&oldid=796112735
12. React - A JavaScript library for building user interfaces. (bez dat.). Preuzeto 10. rujan 2017., od https://facebook.github.io/react/

13. Thinking in React - React. (bez dat.). Preuzeto 10. rujan 2017., od https://facebook.github.io/react/docs/thinking-in-react.html
14. What is LAMP (Linux, Apache, MySQL, PHP)? - Definition from WhatIs.com. (bez dat.). Preuzeto 10. rujan 2017., od http://whatis.techtarget.com/definition/LAMP-Linux-Apache-MySQL-PHP
15. What is MEAN (MongoDB, Ember, Angular, Node)? - Definition from WhatIs.com. (bez dat.). Preuzeto 10. rujan 2017., od http://searchsoftwarequality.techtarget.com/definition/MEAN-MongoDB-Ember-Angular-Node
16. What is RESTful API? - Definition from WhatIs.com. (bez dat.). Preuzeto 10. rujan 2017., od http://searchcloudstorage.techtarget.com/definition/RESTful-API