Naruči Has

Predmet: Klijent Server Sistemi

Januar 2021.

Student:
Uroš Simić - REr 2/18

Profesor:
Dr. Mirko Kosanović

SADRŽAJ

Uvod

Projekat “Naruči Has” je veb aplikacija za ugostiteljske objekte. Zamisao ove aplikacije je laka komunijacija izmedju konobara i kuhinje. Kada konobar primi porudžbinu klikom na neki od proizvoda šalje kuhinji informaciju šta je potrebno da pripreme. Ceo meni je vidljiv na stranici za konobare dok je poručeno vidljivo na stranici za kuhinju. Sve radi u realnom vremenu.

Tehnologije koje koristimo kako bi napravili ovu aplikaciju su:
- Node.js za serverski deo aplikacije (Express framework)
- RethinkDB za bazu
- Angular biblioteku za front-end deo aplikacije

Ceo projekat napisan je u programu PhpStorm

Šta je Node.js ?

To je višeplatformsko JavaSkript radno okruženje otvorenog koda za izvršavanje JavaSkript-a na serverskoj strani. Node.js omogućava da se JavaSkript koristi za skripte na serverskoj strani koje omogućavaju da se sadržaj dinamičnih veb stranica generiše na serveru pre nego što se pošalje do veb pregledača korisnika. Node.js omogućava uniformisanje razvoj veb aplikacija u jednom programskom jeziku, bez potrebe da se za skripte na serverskoj strani koristi različit programski jezik. Ovakav izbor arhitekture omogućava optimizaciju propusnosti i skalabilnosti u veb aplikacijama sa mnogo ulazno/izlaznih operacija kao i za veb aplikacije u realnom vremenu (npr. programi za komunikaciju u realnom vremenu i igrice u veb pregledaču).
Više o NodeJS-u možete pogledati na *nodejs.org*

Šta je Express.js ?

To je veb aplikacija u okviru Node.js servera, dizajnirana za izradu jednostraničnih, višestraničnih i hibridnih veb aplikacija. To je standardni serverski okvir za Node.js. Exspress.js je pozadinski deo MEAN steka.
Više o Express framwork-u možete pogledati na *expressjs.com*

Šta je RethinkDB ?

To je prva open-source, prilagodljiva JSON baza podataka napravljena za pravljenje aplikacija u realnom vremenu. RethinkDB baza podatka, umesto zahtevanja (polling) promena, developer može reći RethinkDB-u da neprekidno push-uje nove podatke u realnom vremenu. Zahvaljujući ovome, pravljenje aplikacija u realnom vremenu je skraćeno i olakšano.
Više o RethinkDB-u možete pogledati na *rethinkdb.com*

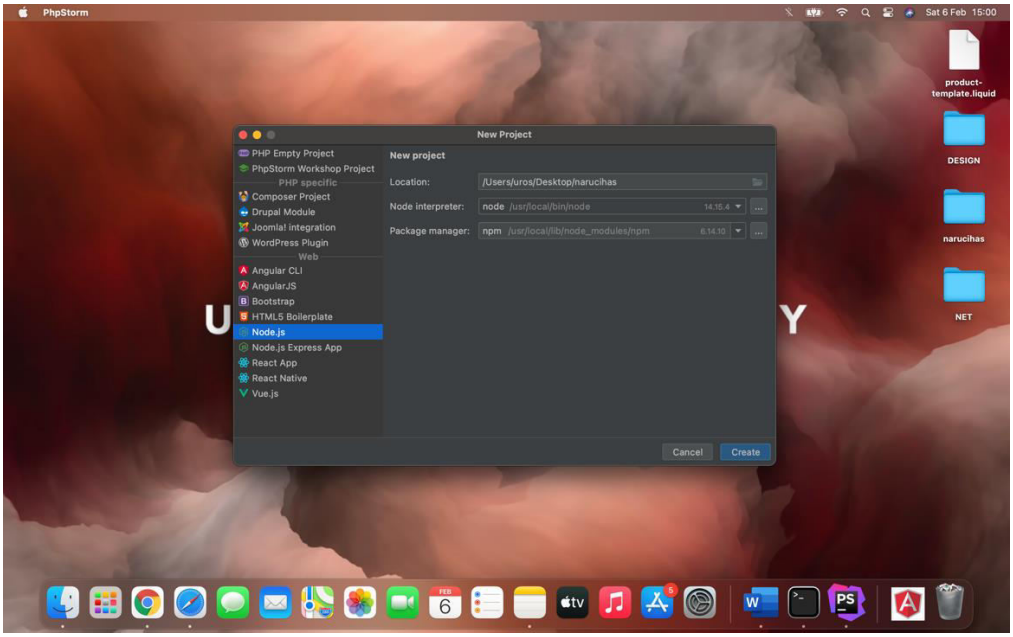
RethinkDB

Šta je AngularJS?

To je open source veb aplikacioni framework koji omogućava pravljenje dinamičke veb aplikacije i proširuje HTML vokabular. AngularJS prilagođava JavaScript kod u zavisnosti od pretraživača.
Više o AngularJS-u možete pogledati na *angularjs.org*

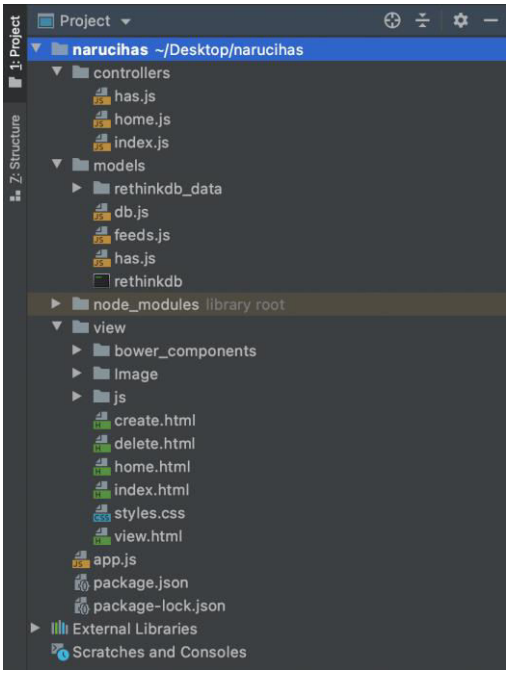
Kako instaliramo i podešavamo projekat

Pre nego da počnemo sa radom moramo da instaliramo tehnologije koje su nam potrebne za razvoj i rad projekta. Preuzimamo Node.js i RethinkDB bazu podataka, linkovi od sajtova sa kojih preuzimamo su navedeni u tekstu iznad.
Kao što smo već naveli aplikaciju razvijamo u PhpStorm-u. Kreiranje Node.js projekta izgleda ovako:



Kad otvorimo PhpStorm biramo lokaciju gde ćemo da smestimo naš projekat i biramo Node.js

Naredni korak jeste kreiranje foldera potrebnih za rad. Ovo je izgled potpuno kreiranog projekta, gde se koristi Model View Controller obrazac (MVC). Folder node_modules se kreira naknadno.



Instaliranjem RethinkDB baze podataka dobijamo jedan fajl rethinkdb, on nam služi da kasnije pokrenemo bazu. Kada pokrenemo taj rethinkdb fajl, on otvara Terminal i tu pokreće bazu. Node.js projekti zahtevaju package.json fajl, njega kreiramo iz Terminala pozicioniranjem na folder i komandom npm init. Express framework možemo instalirati komandom npm install -g express.

Rute koje koristimo

Našem projektu trebaju rute, njih ćemo smestiti u folder Controllers. Koristićemo 3 različite rute.

- GET /has – Ova ruta nam služi da vrati sve menije koje smo prethodno kreirali iz aplikacije i smestili ih u RethinkDB bazu
- POST /has – Ova ruta nam služi da kreiramo novi meni i smestimo ga u našu bazu
- DELETE /has – Ova ruta nam služi da obrišemo sve menije koje smo prethodno kreirali u našoj bazi

Index i Home rute možemo videte ovde

index.js home.js

```
var express = require('express');
var router = express.Router();

router.use('/',require('./home'));
router.use('/has',require('./has'));

module.exports = router;
```

index.js home.js

```
var express = require('express');
var router = express.Router();
var app = express();

module.exports = router;
```

/ služi da pozovemo statičke fajlove
/has vrši pozive server

Podešavanje Baze – RethinkDB

Ovde ćemo da prikažemo podešavanje baze podataka koju ćemo koristiti u našem projektu.
Fajl u kome kreiramo bazu zove se db.js a nalazi se u folderu Models. Baza koju kreiramo zove se Hrana. Kreiramo i tabelu u bazi, ona se zove Food.

```
"use strict";
var rethinkdb = require('rethinkdb');
var async = require('async');

class db {
    setupDb() {
        var self = this;
        async.waterfall([
            function(callback) {
                self.connectToRethinkDbServer( callback: function(err,connection) {
                    if(err) {
                        return callback(true,"Error in connecting RethinkDB");
                    }
                    callback(null,connection);
                });
            },
            function(connection,callback) {
                rethinkdb.dbCreate('hrana').run(connection,function(err, result) {
                    if(err) {
                        console.log("Database already created");
                    } else {
                        console.log("Created new database");
                    }
                    callback(null,connection);
                });
            },
            function(connection,callback) {
                rethinkdb.db('hrana').tableCreate('food').run(connection,function(err,result) {
                    connection.close();
                    if(err) {
                        console.log("Table already created");
                    } else {
                        console.log("Created new table");
                    }
                    callback(null,"Database is setup successfully");
                });
            }
        ],function(err,data) {
            console.log(data);
        });
    }
    connectToRethinkDbServer(callback) {
        rethinkdb.connect({
            host : 'localhost',
            port : 28015
        }, function(err,connection) {
            callback(err,connection);
        });
    }
    connectToDb(callback) {
        rethinkdb.connect({
            host : 'localhost',
            port : 28015,
            db : 'hrana'
        }, function(err,connection) {
            callback(err,connection);
        });
    }
}
module.exports = db;
```

U kodu iznad vidimo tri glavne funkcije
- setupDb() – Vrši konektovanje sa RethinkDB serverom, pravimo bazu i tabelu ukoliko ne postoji
- connectToRethinkDbServer() – Vrši konektovanje sa našim lokalnim serverom
- connectToDb() – Vrši konektovanje sa našom lokalnom bazom

var dbModel = new db();
dbModel.setupDb();
Dodajemo ovih komanda u fajl app.js, zbog njih prvi put kada pokrenemo konzolu pisaće Created new database, Created new table i Database is setup successfully.

RethinkDB Funkcije

Ovde vidimo fajl has.js iz foldera Models. U ovom fajlu kreiramo klasu Has.
Klasa sadrži nekoliko funkcije
- addNewHas() – Služi za dodavanje nove hrane
- getAllHas() – Služi za prikazivanje hrane
- deleteHas() – Služi za brisanje hrane

```
addNewHas(foodData,callback) {
    async.waterfall([
        function(callback) {
            var foodObject = new db();
            foodObject.connectToDb( callback: function(err,connection) {
                if(err) {
                    return callback(true,"Error connecting to database");
                }
                callback(null,connection);
            });
        },
        function(connection,callback) {
            rethinkdb.table('food').insert({
                "hrana" : foodData.hrana
            }).run(connection,function(err,result) {
                connection.close();
                if(err) {
                    return callback(true,"Error happens while adding new food");
                }
                callback(null,result);
            });
        }
    ],function(err,data) {
        callback(err === null ? false : true,data);
    });
}
```

```
getAllHas(callback) {
    async.waterfall([
        function(callback) {
            var foodObject = new db();
            foodObject.connectToDb( callback: function(err,connection) {
                if(err) {
                    return callback(true,"Error connecting to database");
                }
                callback(null,connection);
            });
        },
        function(connection,callback) {
            rethinkdb.table('food').run(connection,function(err,cursor) {
                connection.close();
                if(err) {
                    return callback(true,"Error fetching food to database");
                }
                cursor.toArray(function(err, result) {
                    if(err) {
                        return callback(true,"Error reading cursor");
                    }
                    callback(null,result)
                });
            });
        }
    ],function(err,data) {
        callback(err === null ? false : true,data);
    });
}
```

```
deleteHas(foodData,callback) {
    async.waterfall([
        function(callback) {
            var foodObject = new db();
            foodObject.connectToDb( callback: function(err,connection) {
                if(err) {
                    return callback(true,"Error connecting to database");
                }
                callback(null,connection);
            });
        },
        function(connection,callback) {
            rethinkdb.table('food').delete().run(connection,function(err,result) {
                connection.close();
                if(err) {
                    return callback(true,"Error happens while deleting food");
                }
                callback(null,result);
            });
        }
    ],function(err,data) {
        callback(err === null ? false : true,data);
    });
}
```

has.js iz Controlers foldera koji smo prethodno naveli ima get, post, put i delete. Tu pozivamo ove funkcije koje smo kreirali (slike iznad) u has.js iz Models foldera.

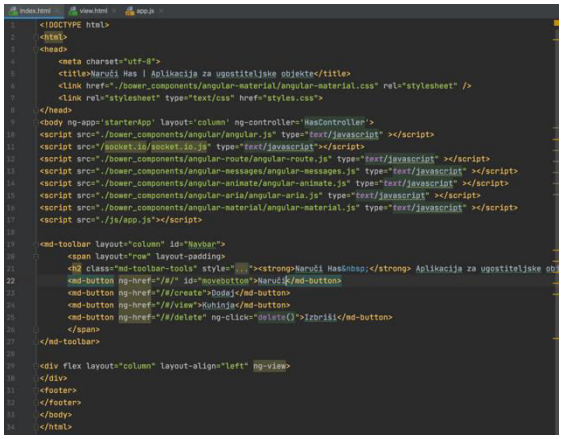
Komande koje koristimo za to

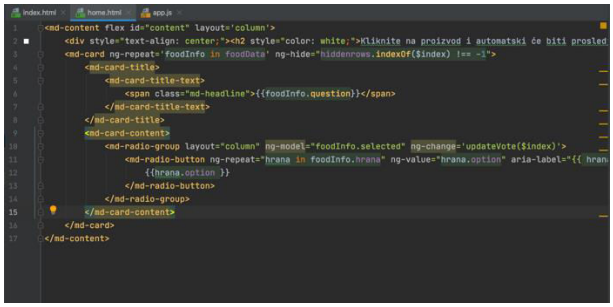
- rethinkdb.table(food).delete().run(connection,function(err,result){}
- rethinkdb.table(food).run(connection,function(err,cursor) {}
- rethinkdb.table(food).get(foodData.id).run(connection,function(err,result) {}
- rethinkdb.table(food).insert({})

Frontend

Aplikacija ima deo koji je vidljiv u Browseru, ovde ćemo videti neke kodove iz tog Frontend dela. Koristimo HTML i Angular.js (na početku imamo objašnjenje)

index.html i home.html su nam vidljivi na na početnoj stranici aplikacije. Prikazujemo navigaciju naseg projekta pomoću koje prelazimo na druge stranice i ispod prikazujemo listu proizvoda koje smo dodali u našu bazu. Klikom na neki proizvod iz te liste vršimo porudžbinu.





Ovaj deo govori koji fajl browser otvara. Ako je putanja napr. /create otvorice create.html fajl. Ispred /create u našem slucaju stoji localhost:3000

```
app.config(function($routeProvider){
    $routeProvider
        .when( path: '/', route: {
            templateUrl: 'home.html'
        })
        .when( path: '/create', route: {
            templateUrl: 'create.html'
        })
        .when( path: '/view', route: {
            templateUrl: 'view.html'
        })
        .when( path: '/delete', route: {
            templateUrl: 'delete.html'
        })
    ;
});
```

Ovde u data skupljamo podatke koji su prethodno upisani u formi na stranici Dodaj

```
var data = {
    "question" : $scope.formData.hasQuestion,
    "hrana" : [{
        "option" : $scope.formData.hasFood1, "vote" : 0
    },{
        "option" : $scope.formData.hasFood2, "vote" : 0
    },{
        "option" : $scope.formData.hasFood3, "vote" : 0
    },{
        "option" : $scope.formData.hasFood4, "vote" : 0
    },{
        "option" : $scope.formData.hasFood5, "vote" : 0
    },{
        "option" : $scope.formData.hasFood6, "vote" : 0
    },{
        "option" : $scope.formData.hasFood7, "vote" : 0
    },{
        "option" : $scope.formData.hasFood8, "vote" : 0
    },{
        "option" : $scope.formData.hasFood9, "vote" : 0
    },{
        "option" : $scope.formData.hasFood10, "vote" : 0
    }]
};
```

Ispis poruke nakon dodavanja novih proizvoda u bazu

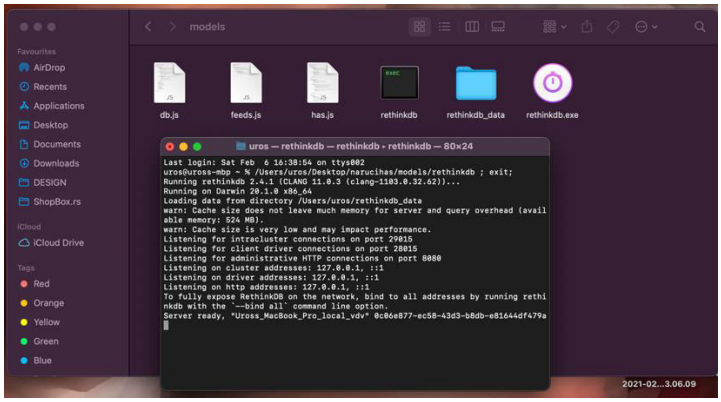
```
var message = {"title" : "", "message" : ""};
$http.post( url: '/has',data).success(function(response) {
    if(response.responseCode === 0) {
        message.title = "Uspeh !";
        message.message = "Ponuda je uspešno napravljena.";
        data["id"] = response.data.generated_keys[0];
        $scope.foodData.push(data);
    } else {
        message.title = "Greška !";
        message.message = "Greška u toku pravljenja ponude.";
    }
    $mdDialog.show(
        $mdDialog.alert()
            .parent(angular.element(document.querySelector( selectors: '#popupContainer')))
            .clickOutsideToClose(true)
            .title(message.title)
            .textContent(message.message)
            .ok('U redu.')
            .targetEvent(ev)
    );
});
```

Zaključak Projekta

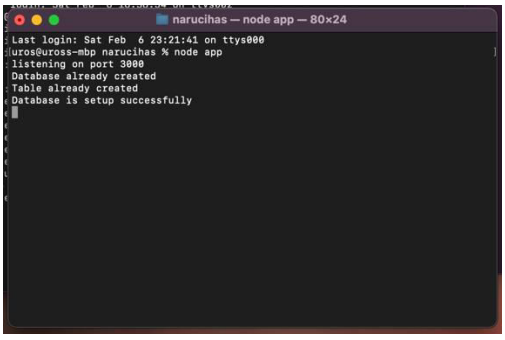
Moj zaključak je da sam uspeo da napravim aplikaciju koja radi u realnom vremenu. Kada se nešto poruči sa početne stranice, koja je za konobare, to se automatski ispisuje na stranici Kuhinja.
Ovo su nam omogućile ove sve tehnologije koje smo koristili Node.js, Express framework i RethinkDB.

U nastavku sledi objašnjenje kako pokrećemo projekat i kako on ustvari funkcioniše

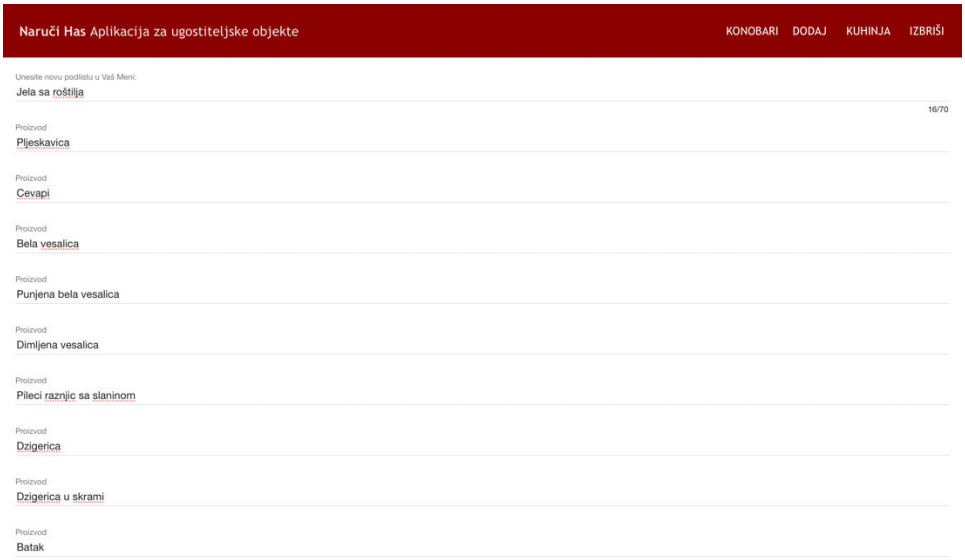
Za početak pokrenućemo RethinkDB. U folderu našeg projekta postoji folder Models, u njemu se nalazi fajl rethinkdb. Kada ga pokrenemo otvoriće se Terminal i pokrenuti server.



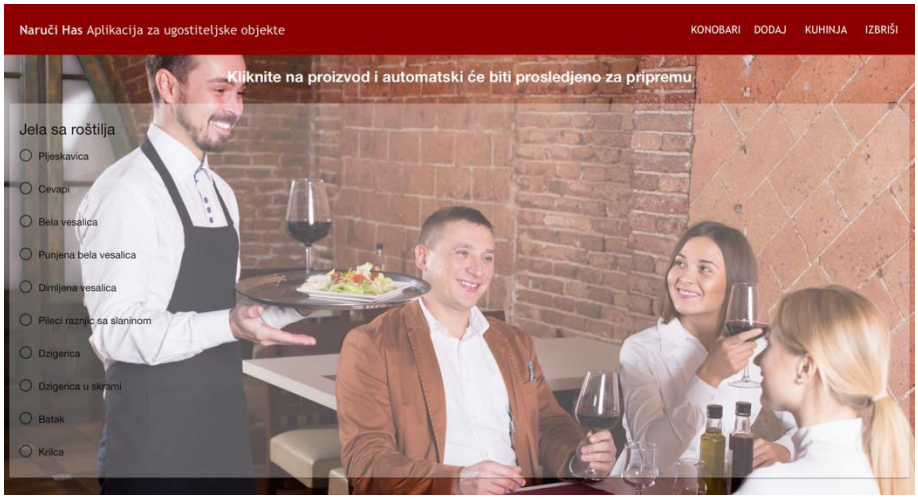
Nakon toga otvaramo novi Terminal prozor pozicioniran na folder našeg projekta i kucamo komandu node app.



Kada smo to sve uradili možemo da otvorimo naš broswe i kucamo localhost:3000
Prvo ćemo otići na stranicu Dodaj kako bi u našu bazu dodali hranu koju ćemo kasnije moći poručiti.



Kada smo uspesno kreirali ponude u naš meni sada ćemo videti kako izgleda poručivanje. Zamisao je da konobar prosledi kuhinji šta sve treba pripremiti za goste.



Sada ćemo da izvršimo jednu porudžbinu. Kliknemo napr. na pljeskavica i dobijamo poruku potvrde. Ovo sve radi u realnom vremenu i odmah se prikazuje.

Sada ćemo preći na stranicu kuhinja. Ovde se skupljaju porudžbine. Kuvari ovde vide šta im je sve konobar poslao da treba da spreme.

Button Izbriši nam služi kao funkcija za brisanje svega iz baze.