

Rezervacija apartmana

Predmet: Klijent server sistemi

Mentor:
dr. Mirko Kosanović

Student:
Veljko Jevtić REr 3/18

Januar 2021.

SADRŽAJ

Uvod.....	2
1. Instalacija i podešavanje projekta	3
2. Podešavanje ruta	4
3. Podešavanje baze - RethinkDB.....	5
4. RethinkDB funkcije	7
5. Front-end (AngularJS) Index.html.....	11
6. Zaključak.....	15

Uvod

Projekat Rezervacija apartmana je veb aplikacija gde korisnik može da izabere hotel i da ga rezerviše. Rezervacija radi u realnom vremenu (real time), pa se rezultati vide odmah.

Tehnologije koje koristimo kako bi napravili ovu aplikaciju su: Nodejs za serverski deo aplikacije (Express framework), RethinkDB kao našu bazu i Angular biblioteku za front-end deo aplikacije.

Okruženje korišćeno za pisanje aplikacije je PhpStorm.

Šta je NodeJS?

Node.js je programski jezik zasnovan na JavaScript jeziku. On je ne-blokirajući, event driven, lightweight, efikasan jezik čija je glavna namena da se koristi kod distribuiranih aplikacija koje rade na različitim platformama i koje imaju potrebu da rade sa velikim količinama zahteva ili podataka u realnom vremenu.

NodeJS je naročito pogodan za aplikacije koje moraju da održavaju perzistentnu konekciju sa serverom, najčešće korišćenjem veb soketa (primer takve aplikacije bi bio chat program). Mrežne aplikacije koje zahtevaju brzinu, skalabilnost i podržavaju veliki broj istovremenih konekcija se razvijaju u ovom programskom jeziku.

Više o NodeJS-u možete pogledati ovde: <https://nodejs.org/en/>

Express frejmwork predstavlja najpoznatiji framework za razvoj veb aplikacija zasnovanih na node.js programskom jeziku. U sebi sadrži podršku za rutiranje, konfiguraciju, templejt engine, POST parsiranje zahteva, i pristup različitim bazama podataka i druge funkcionalnosti.

Više o Express framwork-u možete pogledati ovde: <https://expressjs.com/>

Šta je RethinkDB?

RethinkDB je prva open-source, prilagodljiva JSON baza podataka napravljena za pravljenje aplikacija u realnom vremenu. RethinkDB baza podataka, 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 ovde: <https://www.rethinkdb.com/>

Šta je Angular?

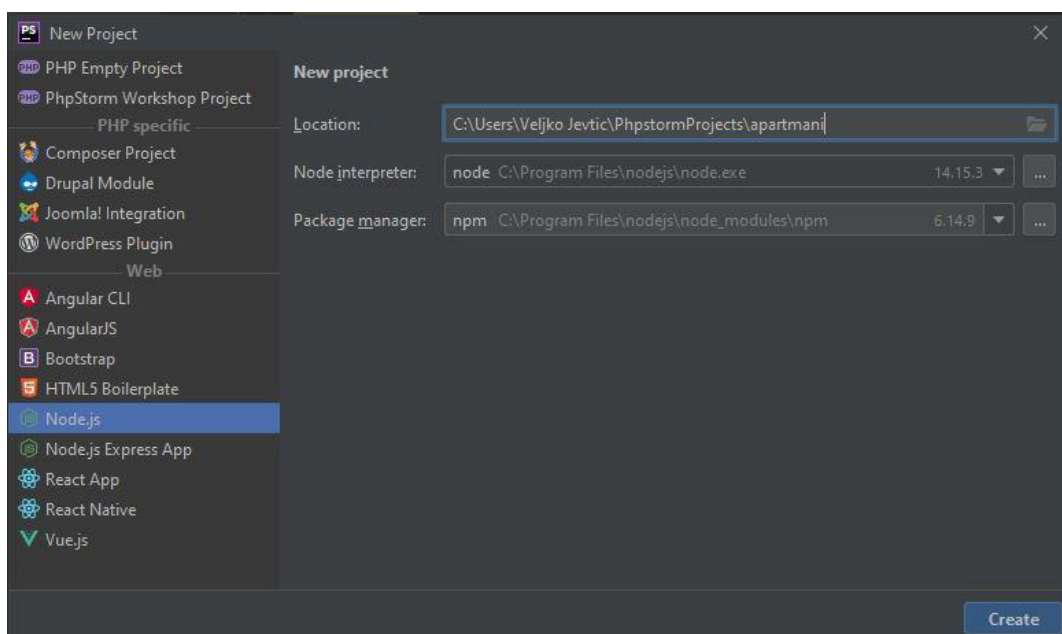
AngularJS je open source veb aplikacioni framework koji omogućava pravljenje dinamičke veb aplikacije i proširuje HTML vokabular. AngularJS prilagođava JavaScript kod u zavisnosti od pretraživača.

Više o AngularJS-u možete pogledati ovde: <https://angularjs.org/>

1. Instalacija i podešavanje projekta

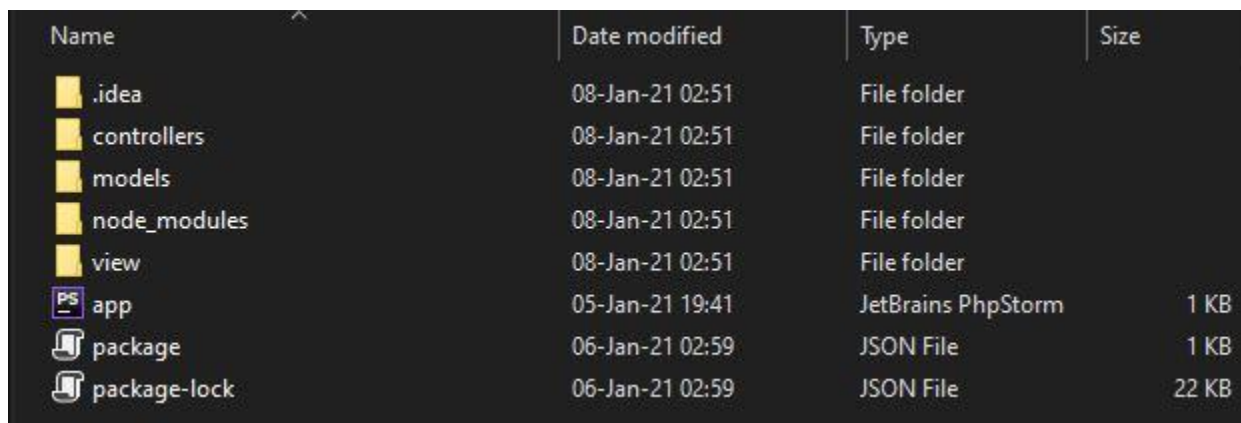
Prvo je potrebno instalirati sve tehnologije koje se koriste u projektu. Preuzimamo Node.js i RethinkDB bazu podataka sa linkova koji se nalaze u uvodu teksta.

Aplikacija je pisana u programu PhpStorm. Sledeći korak je kreiranje projekta i on izgleda ovako:



Sa leve strane prozora od tehnologija biramo Node.js, a sa desne strane u polju „Location“ upisujemo gde želimo projekat da se kreira.

Naredni korak jeste kreiranje foldera potrebnih za rad. Ovako izgleda kompletan projekat, gde se koristi Model View Controller obrazac (MVC). Folder idea i node_modules se kreiraju naknadno.



Name	Date modified	Type	Size
.idea	08-Jan-21 02:51	File folder	
controllers	08-Jan-21 02:51	File folder	
models	08-Jan-21 02:51	File folder	
node_modules	08-Jan-21 02:51	File folder	
view	08-Jan-21 02:51	File folder	
app	05-Jan-21 19:41	JetBrains PhpStorm	1 KB
package	06-Jan-21 02:59	JSON File	1 KB
package-lock	06-Jan-21 02:59	JSON File	22 KB

Kada kreiramo foldere sa slike, potrebno je u folder Models smestiti rethinkdb.exe fajl koji dobijemo instaliranjem rethink baze podataka. Nakon toga pokrećemo konzolu kucanjem cmd u polje za pretrživanje u Windowsu. Potrebno je iskoristiti npm init komande, za pravljenje package.json, dok Express framework možemo instalirati komandom: npm install -g express.

2. Podešavanje ruta

U ovom poglavlju će biti prikazan kompletan kod aplikacije.

U našem controllers folderu moramo napraviti različite rute za našu aplikaciju. Koristićemo 3 vrste http ruta u ovoj aplikaciji:

- GET /hotels – vrati sve hotele iz RethinkDB baze
- POST /hotel – napravi nov hotel
- DELETE /hotel – obriši sve hotele

Index i home rute:

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

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

module.exports = router;
```

Prva ruta koristi „ / “ kao rutu kako bi pozivala statičke fajlove. Dok u „/hotel“ radimo API pozive. (pozive serveru)

3. Podešavanje baze - RethinkDB

Sledeći deo teksta će biti posvećen podešavanju baze podataka RethinkDB. U folderu Models kreiramo db.js fajl koji izgleda ovako :

```
5  class db {
6    setupDb() {
7      var self = this;
8      async.waterfall([
9        function(callback) {
10         self.connectToRethinkDbServer( callback: function(err,connection) {
11           if(err) {
12             return callback(true,"Error in connecting RethinkDB");
13           }
14           callback(null,connection);
15         });
16       },
17       function(connection,callback) {
18         rethinkdb.dbCreate('hotels').run(connection,function(err, result) {
19           if(err) {
20             console.log("Database already created");
21           } else {
22             console.log("Created new database");
23           }
24           callback(null,connection);
25         });
26       },
27       function(connection,callback) {
28         rethinkdb.db('hotels').tableCreate('hotel').run(connection,function(err,result) {
29           connection.close();
30           if(err) {
31             console.log("Table already created");
32           } else {
```

```
32     } else {
33         console.log("Created new table");
34     }
35     callback(null,"Database is setup successfully");
36     });
37 }
38 ],function(err,data) {
39     console.log(data);
40 });
41 }
42
43 connectToRethinkDbServer(callback) {
44     rethinkdb.connect({
45         host : 'localhost',
46         port : 28015
47     }, function(err,connection) {
48         callback(err,connection);
49     });
50 }
51
52 connectToDb(callback) {
53     rethinkdb.connect({
54         host : 'localhost',
55         port : 28015,
56         db : 'hotels'
57     }, function(err,connection) {
58         callback(err,connection);
59     });
60 }
```

U kodu vidimo tri glavne funkcije, koje koriste Async modul kako bi radile callback bolje:

- `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.

Nakon dodavanja: `var dbModel = new db();` i `dbModel.setupDb();` u `app.js`, u konzoli će prvi put pisati Created new database, Created new table i Database is setup successfully.

4. RethinkDB funkcije

U našem models folderu, hotel.js, napravićemo klasu hotel, koja će sadržiti funkcije za dodavanje hotela - addNewHotels, prikazivanje hotela - getAllHotels i brisanje hotela – deleteHotels.

```
6  class hotel {
7    addNewHotels(hotelData,callback) {
8      async.waterfall([
9        function(callback) {
10         var hotelObject = new db();
11         hotelObject.connectToDb( callback: function(err,connection) {
12           if(err) {
13             return callback(true,"Error connecting to database");
14           }
15           callback(null,connection);
16         });
17       ],
18       function(connection,callback) {
19         rethinkdb.table( tabularData: 'hotel').insert({
20           "hotels" : hotelData.hotels
21         }).run(connection,function(err,result) {
22           connection.close();
23           if(err) {
24             return callback(true,"Error happens while adding new hotels");
25           }
26           callback(null,result);
27         });
28       }
29     ],function(err,data) {
30       callback(err === null ? false : true,data);
31     });
32   }
33 }
```



```

34 voteHotelOption(hotelData,callback) {
35   async.waterfall([
36     function(callback) {
37       var hotelObject = new db();
38       hotelObject.connectToDb( callback: function(err,connection) {
39         if(err) {
40           return callback(true,"Error connecting to database");
41         }
42         callback(null,connection);
43       });
44     },
45     function(connection,callback) {
46       rethinkdb.table( tabularData: 'hotel').get(hotelData.id).run(connection,function(err,result) {
47         if(err) {
48           return callback(true,"Error fetching hotels to database");
49         }
50         for(var hotelCounter = 0; hotelCounter < result.hotels.length; hotelCounter++) {
51           if(result.hotels[hotelCounter].option === hotelData.option) {
52             result.hotels[hotelCounter].vote += 1;
53             break;
54           }
55         }
56       rethinkdb.table( tabularData: 'hotel').get(hotelData.id).update(result).run(connection,function(err,result) {
57         connection.close();
58         if(err) {
59           return callback(true,"Error updating the vote");
60         }
61         callback(null,result);

```

```

70 getAllHotels(callback) {
71   async.waterfall([
72     function(callback) {
73       var hotelObject = new db();
74       hotelObject.connectToDb( callback: function(err,connection) {
75         if(err) {
76           return callback(true,"Error connecting to database");
77         }
78         callback(null,connection);
79       });
80     },
81     function(connection,callback) {
82       rethinkdb.table( tabularData: 'hotel').run(connection,function(err,cursor) {
83         connection.close();
84         if(err) {
85           return callback(true,"Error fetching hotels to database");
86         }
87         cursor.toArray(function(err, result) {
88           if(err) {
89             return callback(true,"Error reading cursor");
90           }
91           callback(null,result)
92         });
93       });
94     }
95   ],function(err,data) {
96     callback(err === null ? false : true,data);
97   });
98 }

```

```
101 deleteHotels(hotelData,callback) {
102   async.waterfall([
103     function(callback) {
104       var hotelObject = new db();
105       hotelObject.connectToDb( callback: function(err,connection) {
106         if(err) {
107           return callback(true,"Error connecting to database");
108         }
109         callback(null,connection);
110       });
111     },
112     function(connection,callback) {
113       rethinkdb.table( tabularData: 'hotel').delete().run(connection,function(err,result) {
114         connection.close();
115         if(err) {
116           return callback(true,"Error happens while deleting hotels");
117         }
118         callback(null,result);
119       });
120     }
121   ],function(err,data) {
122     callback(err === null ? false : true,data);
123   });
124 }
125 }
126 module.exports = hotel;
```

Najbitnije komande koje koristimo za rad sa rethinkdb tabelom su:

rethinkdb.table(hotel).delete().run(connection,function(err,result){}

rethinkdb.table(hotel).run(connection,function(err,cursor) {}

rethinkdb.table(hotel).get(HotelData.id).run(connection,function(err,result) {}

rethinkdb.table(hotel).insert({})

U našem controllers folderu, u hotel.js, imamo get, post, put i delete zahteve koji pozivaju funkcije koje smo napravili u models/hotel.js.

```
1  var express = require('express');
2  var router = express.Router();
3  // require model file.
4  var hotelModel = require('../models/hotel');
5
6  router.route('/')
7  .get(function(req,res) {
8    // Code to fetch the hotels.
9    var hotelObject = new hotelModel();
10   // Calling our model function.
11   hotelObject.getAllHotels( callback function(err,hotelResponse) {
12     if(err) {
13       return res.json({"responseCode" : 1, "responseDesc" : hotelResponse});
14     }
15     res.json({"responseCode" : 0, "responseDesc" : "Success", "data" : hotelResponse});
16   });
17 })
18 .post(function(req,res) {
19   // Code to add new hotels.
20   var hotelObject = new hotelModel();
21   // Calling our model function.
22   // We need to validate our payload here.
23   hotelObject.addNewHotels(req.body, callback function(err,hotelResponse) {
24     if(err) {
25       return res.json({"responseCode" : 1, "responseDesc" : hotelResponse});
26     }
27     res.json({"responseCode" : 0, "responseDesc" : "Success","data" : hotelResponse});
28   });
29 })
30 .delete(function(req,res) {
31   // Code to delete hotels.
32   var hotelObject = new hotelModel();
33   // Calling our model function.
34   // We need to validate our payload here.
35   hotelObject.deleteHotels(req.body, callback function(err,hotelResponse) {
36     if(err) {
37       return res.json({"responseCode" : 1, "responseDesc" : hotelResponse});
38     }
39     res.json({"responseCode" : 0, "responseDesc" : "Success","data" : hotelResponse});
40   });
41 })
42 .put(function(req,res) {
43   // Code to update votes hotels.
44   var hotelObject = new hotelModel();
45   // Calling our model function.
46   // We need to validate our payload here.
47   hotelObject.voteHotelOption(req.body, callback function(err,hotelResponse) {
48     if(err) {
49       return res.json({"responseCode" : 1, "responseDesc" : hotelResponse});
50     }
51     res.json({"responseCode" : 0, "responseDesc" : "Success", "data" : hotelResponse});
52   });
53 });
54
55 module.exports = router;
```

5. Front-end (AngularJS)

Index.html

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="utf-8">
5    <title>Rezervacija apartmana</title>
6    <link href="../bower_components/angular-material/angular-material.css" rel="stylesheet" />
7    <link rel="stylesheet" type="text/css" href="styles.css">
8  </head>
9  <body ng-app='starterApp' layout='column' ng-controller='HotelController'>
10   <script src="../bower_components/angular/angular.js" type="text/javascript" ></script>
11   <script src="/socket.io/socket.io.js" type="text/javascript"></script>
12   <script src="../bower_components/angular-route/angular-route.js" type="text/javascript" ></script>
13   <script src="../bower_components/angular-messages/angular-messages.js" type="text/javascript" ></script>
14   <script src="../bower_components/angular-animate/angular-animate.js" type="text/javascript" ></script>
15   <script src="../bower_components/angular-aria/angular-aria.js" type="text/javascript" ></script>
16   <script src="../bower_components/angular-material/angular-material.js" type="text/javascript" ></script>
17   <script src="/js/app.js"></script>
18
19   <md-toolbar layout="column">
20
21     <span layout="row" layout-padding>
22       <h2 class="md-toolbar-tools" style="text-align: center;"> Rezervacija apartmana</h2>
23       <md-button ng-href="#/" id="movebottom">Hoteli</md-button>
24       <md-button ng-href="#/create">Ponuda</md-button>
25       <md-button ng-href="#/view">Rezervacije</md-button>
26       <md-button ng-href="#/delete" ng-click="delete()">Izbriši</md-button>
27     </span>
28   </md-toolbar>

```

Create.html

```

1  <md-content layout-padding>
2    <form name="projectForm">
3      <md-input-container class="md-block">
4        <label>Unesite ime hotela :</label>
5        <input md-maxlength="70" required name="question" ng-model="formData.hotelQuestion">
6        <div ng-messages="projectForm.question.$error">
7          <div ng-message="required">Ovo polje je obavezno.</div>
8          <div ng-message="md-maxlength">Manje od 70 karaktera duzine.</div>
9        </div>
10     </md-input-container>
11     <md-input-container class="md-block">
12       <label>Ime apartmana</label>
13       <input required name="apartman1" ng-model="formData.hotelApartman1">
14       <div ng-messages="projectForm.hotelApartman1.$error">
15         <div ng-message="required">Ovo polje je obavezno.</div>
16       </div>
17     </md-input-container>
18     <md-input-container class="md-block">
19       <label>Ime apartmana</label>
20       <input required name="apartman2" ng-model="formData.hotelApartman2"/>
21       <div ng-messages="projectForm.hotelApartman2.$error" role="alert">
22         <div ng-message="required">Ovo polje je obavezno.</div>
23       </div>
24     </md-input-container>
25     <md-input-container class="md-block">
26       <label>Ime apartmana</label>
27       <input required name="apartman3" ng-model="formData.hotelApartman3"/>
28       <div ng-messages="projectForm.hotelApartman3.$error" role="alert">
29         <div ng-message="required">Ovo polje je obavezno.</div>

```


app.js (views\js):

```
1  var app = angular.module( name: 'starterApp', requires: ['ngMaterial','ngRoute','ngMessages']);
2
3  app.factory('socket',function(){
4      var socket = io.connect('http://localhost:3000');
5      return socket;
6  });
7
8  app.config(function($routeProvider){
9      $routeProvider
10         .when( path: '/', route: {
11             templateUrl: 'home.html'
12         })
13         .when( path: '/create', route: {
14             templateUrl: 'create.html'
15         })
16         .when( path: '/view', route: {
17             templateUrl: 'view.html'
18         })
19         .when( path: '/delete', route: { //Open delete.html
20             templateUrl: 'delete.html'
21         })
22     ;
23 });
24
25 app.controller('HotelController',function($scope,$mdDialog,$http,socket) {
26
27     $scope.hotelData = [];
28     $scope.formData = {};
29     $scope.voteData = {};
```

```

29     $scope.voteData = {};
30     $scope.hiddenrows = [];
31     getHotelData();
32     function getHotelData() {
33         $http.get( url: "/hotel").success(function(response){
34             $scope.hotelData = response.data;
35         });
36     }
37     $scope.submitHotel = function(ev) {
38         var data = {
39             "question" : $scope.formData.hotelQuestion,
40             "hotels" : [{
41                 "option" : $scope.formData.hotelApartman1, "vote" : 0
42             },{
43                 "option" : $scope.formData.hotelApartman2, "vote" : 0
44             },{
45                 "option" : $scope.formData.hotelApartman3, "vote" : 0
46             }]
47         };
48         var message = {"title" : "", "message" : ""};
49         $http.post( url: '/hotel',data).success(function(response) {
50             if(response.responseCode === 0) {
51                 message.title = "Uspeh !";
52                 message.message = "Ponuda je uspešno napravljena.";
53                 data["id"] = response.data.generated_keys[0];
54                 $scope.hotelData.push(data);
55             } else {
56                 message.title = "Greška !";
57                 message.message = "Greška u toku pravljenja ponude.";
58             }
59             $mdDialog.show(
60                 $mdDialog.alert()
61                     .parent(angular.element(document.querySelector( selectors: '#popupContainer'))))
62                     .clickOutsideToClose(true)
63                     .title(message.title)
64                     .textContent(message.message)
65                     .ok('U redu.')
66                     .targetEvent(ev)
67             );
68         });
69     }
70
71     $scope.updateVote = function(index) {
72         var data = {
73             "id" : $scope.hotelData[index].id,
74             "option" : $scope.hotelData[index].selected
75         };
76         $http.put( url: "/hotel",data).success(function(response) {
77             if(response.responseCode === 0) {
78                 console.log("Success");
79                 $scope.hiddenrows.push(index);
80             } else {
81                 console.log("error");
82             }
83         });
84     }

```

```

86  $scope.delete = function(index) {
87      $http({
88          method: 'DELETE',
89          url: '/hotel/',
90          data: {
91              "id" : $scope.hotelData[index],
92              "option" : $scope.hotelData[index]
93          },
94          headers: {
95              'Content-type': 'application/json;charset=utf-8'
96          }
97      })
98      .then(function(response) {
99          console.log(response.data);
100      }, function(rejection) {
101          console.log(rejection.data);
102      });
103
104      alert = $mdDialog.alert({
105          title: 'Uspeh',
106          textContent: 'Podaci uspešno izbrisani.',
107          ok: 'U redu.'
108      });
109
110      $mdDialog
111          .show( alert )
112          .finally(function() {
113              alert = undefined;

```

```

104      alert = $mdDialog.alert({
105          title: 'Uspeh',
106          textContent: 'Podaci uspešno izbrisani.',
107          ok: 'U redu.'
108      });
109
110      $mdDialog
111          .show( alert )
112          .finally(function() {
113              alert = undefined;
114              location.reload();
115          });
116
117  };
118
119
120  socket.on('changeFeed',function(data) {
121      for(var hotelCounter = 0 ;hotelCounter < $scope.hotelData.length; hotelCounter++) {
122          if($scope.hotelData[hotelCounter].id === data.id) {
123              $scope.hotelData[hotelCounter].hotels = data.hotels;
124              $scope.$apply();
125          }
126      }
127  });
128  });
129

```

6. Zaključak

Zahvaljujući NodeJS-u, Express framework-u i RehinkDB bazi podataka možemo lako napraviti aplikaciju u realnom vremenu.

Sve ovo u kombinaciji sa Angular framework-om daje moderan izgled našoj responsive veb aplikaciji.

Konačan izgled aplikacije:

Rezervacija apartmana

HOTELI PONUDA REZERVACI IZBRIŠI

Unesite ime hotela :
Hotel Panorama 14/70

Ime apartmana
Gold

Ime apartmana
Silver

Ime apartmana
Bronze

KREIRAJ PONUDU

U gornjem delu slike vidimo četiri dugmeta, trenutni prikaz jeste prikaz dugmeta Ponuda, gde možemo da dodamo hotele i apartmane. Klikom na Kreiraj ponudu dobijamo sledeću poruku :

The screenshot shows a web form titled 'Rezervacija apartmana'. At the top, there is a navigation bar with the title and four tabs: 'HOTELI', 'PONUDA', 'REZERVACI', and 'IZBRIŠI'. The form contains four input fields for hotel name and apartment type, each with a character count (14/70). The first hotel is 'Hotel Panorama' and the first apartment is 'Gold'. A modal dialog box is displayed in the center with the text 'Uspeh !' and 'Ponuda je uspešno napravljena.' Below the dialog is a button labeled 'U REDU.'. At the bottom left of the form is a button labeled 'KREIRAJ PONUDU'.

Rezervacija apartmana

HOTELI PONUDA REZERVACI IZBRIŠI

Unesite ime hotela :
Hotel Panorama 14/70

Ime apartmana
Gold

Ime apartmana
Silver

Ime apartmana
Bronze

Uspeh !
Ponuda je uspešno napravljena.

U REDU.

KREIRAJ PONUDU

Nakon kreiranja dva hotela, klikom na dugme Hoteli dobijamo sledeći prozor :

The screenshot shows the 'HOTELI' tab selected in the navigation bar. It displays two hotel entries. The first entry is 'Hotel Panorama' with three radio button options: 'Gold', 'Silver', and 'Bronze'. The second entry is 'Hotel Lux' with three radio button options: 'Platinum', 'Diamond', and 'Emerald'.

Rezervacija apartmana

HOTELI PONUDA REZERVACI IZBRIŠI

Hotel Panorama

☐ Gold

☐ Silver

☐ Bronze

Hotel Lux

☐ Platinum

☐ Diamond

☐ Emerald

Klikom na određeni apartman se beleži rezervacija koju možemo da vidimo klikom na dugme Rezervacije u realnom vremenu.

Rezervacija apartmana			HOTELI	PONUĐA	REZERVACI	IZBRIŠI
Gold : 1		Silver : 0		Bronze : 0		
Platinum : 0		Diamond : 0		Emerald : 1		

Ovaj deo aplikacije je isključivo radi testiranja i pokazivanja funkcionalnosti. Klikom na dugme Izbriši, brišu se svi podaci iz baze podataka uz adekvatnu poruku.

