Find your restaurant

O aplikaciji

Projekat Find your restaurant omogućava korisniku da pregleda listu restorana gde može videti u kom gradu se restoran nalazi, kao i ime restorana. Klikom na određeni restoran prikazuju se detaljnije informacije o samom restoranu kao što su broj telefona, meni koji sadrži hranu i cenu. Korisnik ukoliko želi može da doda svoje omiljene restorane i određene informacije kao što su ime restorana, lokacija i kratak opis, kao i da ih obriše i izmeni ukoliko želi da doda neku novu informaciju za određeni restoran. Za pokretanje aplikacije potrebno je u konzoli otkucati nodemon app čime ćemo pokrenuti aplikaciju, dok u browser-u otkucati localhost:3000, 3000 zato što aplikacija sluša na portu 3000.

Tehnologije

Za kreiranje ovog projekta korišćene su sledeće tehnologije:

- NodeJS(Express framework)
- MongoDB(Cloud)
- HTML, Bootstrap

Node.js

Node.js je radno okruženje JavaScript programskog jezika koje se koristi na strani web servera, dok se JavaScript tradicionalno koristi na strani klijenta odnosno na front-endu. Node.js je omogućio da se JavaScript koristi i za back-end deo, odnosno za serversko izvršavanje koda.

Node.js ima veliki broj modula – JavaScript biblioteka ili funkcija koje obavljaju određene specifnične zadatke i koje možete dodati svojoj aplikaciji. Postoje neki ugrađeni moduli, dok su drugi dostupni pre svega putem NPM repozitorijuma – Node Package Manager.

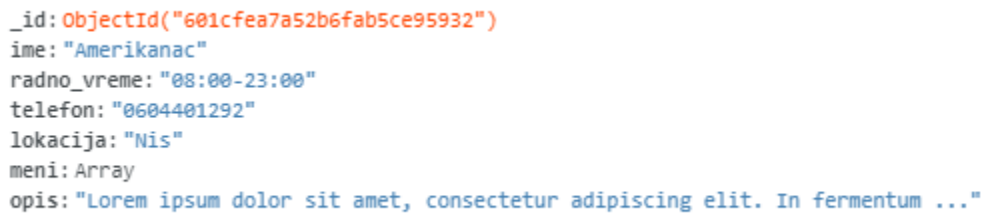
Express

Express je Node.js web framework. Koristi se za programiranje web aplikacija i API-ja(Application programming interface). Express olakšava i ubrzava mnoge radnje i pomaže nam pri pisanju koda prilikom kreiranja Node.js aplikacija, gde se pretežno misli da pisanje suvišnih linija koda. Express.js je pozadinski deo MEAN steka, zajedno sa Mongo DB bazom podataka i celokupnom Angular.js okvirom.

MongoDB

MongoDB pripada grupi nerelacionih baza podataka. Postoje više vrsta nerelacionih baza podataka, a baze podataka u MongoDB pripadaju vrsti koja se naziva baza dokumenata. To znači da se svi podaci čuvaju u obliku dokumenata, gde kolekcija predstavlja neku vrstu tabele, a dokument red. Format u kojem su zapisani ovi dokumenti je sličan formatu JSON objekata. Vrednosti svojstava dokumenata mogu biti brojevi, stringovi, nizovi...

Sledećom slikom



ću ilustrovati kako jedna kolekcija izgleda:

Bootstrap

Bootstrap je front-end biblioteka koja sadrži već kreirane komponente kao i utility klase kreiranje uz pomoć HTML-a, CSS-a i JavaScripta-a. Bootstrap su razvili developeri u Twitter-u kako bi dobili konzistentnost prilikom kreiranja novih web stranica. On sadrži stilove vezane za forme, modal-e, sladjere, naslove, forme..

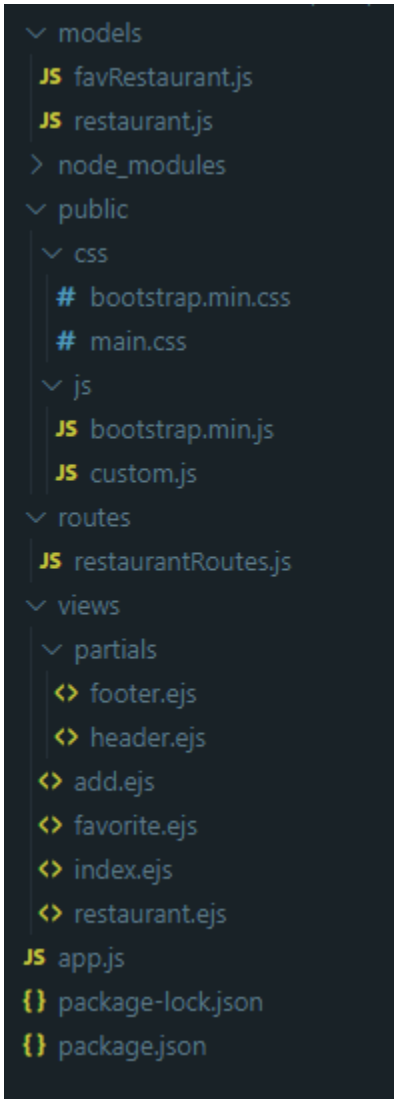
Podešavanje projekta

Da bismo koristili Node.js kao i Express, ali i neke druge module, potrebno je pre svega imati instaliran Node.js na računaru. U folderu gde želimo da započnemo projekat treba da pokrenemo sledeće komande.

Npm init – inicijalizujemo package.json fajl

Npm install express ejs mongoose – ovom komandom omogućavamo korišćenje express framework-a, ejs template jezika kao i rad sa mongo bazom podataka.

Struktura fajlova



- Models – Modeli koje koristimo pri radu sa bazom podataka

- Node_modules – Sadrži sve pakete koji se automatski dodaju kada pokrenemo npm install
- Public – Ubacujemo fajlove koji su potrebni za front-end deo
- Routes – Sadrži rute koje imamo u aplikaciji
- Views – Sadrži .ejs templejte koji sadrže HTML strane
- Partial – Sadrzi delove koji se stalno ponavljaju, kao što su header i footer
- App.js – Glavni fajl aplikacije koji pokreće projekat

Instalacija Boostrapa

Bootstrap možemo instalirati na par načina, preuzimanjem kompresovanih css i js fajlova sa njihovog sajta ili preko cnd-a tako što u header.ejs templejtu dodamo link.

Rute

Kako bi naša aplikacija radila, potrebno je definisati rute koje naša aplikacija ima.

```
router.get('/', (req, res) => {
    Restaurant.find()
        .then((result) => {
            res.render('index', { restaurants: result });
        })
        .catch((err) => {
            console.log(err);
        });
});
```

Default ruta, odnosno ruta koja se prikazuje korisniku kada pristupi sajtu. Koristimo model Restaurant i metodu find kako bismo pronašli sve restorane. Funkcija render ima dva parametara, prvi je template koji želimo da prikažemo korisniku, dok je drugi parametar objekat koji šaljemo u template i koji ćemo prikazati korisniku.

```
//get selected restaurants
router.get('/restaurant/single/:id', (req, res) => {
    const id = req.params.id
    Restaurant.findById(id)
        .then((result) => {
            res.render('restaurant', { restaurant: result });
        })
        .catch((err) => {
            console.log(err);
        });
});
```

Get selected restaurant ruta služi za za prikaz sleketovanog restorana iz liste sa početne strane. Kada se pogodi ova ruta, uzimamo id iz req.params i prosleđujemo ga kao parametar funkciji findById, a zatim dobijeni rezultat prosleđujemo templejtu restaurant kao objekat restaurant.

```
//Add favorite restaurant
router.post('/restaurant/favorite', (req, res) => {
    const favRestaurant = new FavoriteRestaurant(req.body);
    console.log(req.body);
    favRestaurant.save()
        .then((result) => {
            res.redirect('/restaurant/add');
        })
        .catch((err) => {
            console.log(err);
        })
});
```

Add favorite restaurant je ruta preko koje korisnik može da dodaje svoje omiljene restorane putem forme. Forma se nalazi u templejtu add.ejs, gde je metoda forme POST, a ruta koja se gađa /restaurant/favorite. Podatke koje smo poslali iz forme preuzimamo tako što kreiramo novi objekat modela FavoriteRestaurant i zatim mu prosledimo kao parametar req.body(podaci iz forme). Kada se uspešno unesu podaci, aplikacija nas redirectuje na rutu /restaurant/add.

```
// Delete one
router.get('/delete/:_id', (req, res) => {
    const {_id} = req.params;
    FavoriteRestaurant.deleteOne({_id})
        .then( ()=>{
            console.log('Uspesno obrisano');
            res.redirect("/restaurant/favorite");
        })
        .catch((err)=>{
            console.log(err);
        })
});
```

Delete one je ruta koja omogućava korisniku da obriše restoran iz svoje liste omiljenih rezultata. Klikom na dugme obriši, gađa se ruta /delete/:id gde se iz requesta.params uzima id kliknutog restorana. Ukoliko je restoran uspešno obrisan, aplikacija nas redirectuje na rutu /restaurant/favorite.

```
router.get('/restaurant/favorite', (req, res) => {
    FavoriteRestaurant.find()
        .then((result) => {
            res.render('favorite', { favRestaurant: result });
        })
        .catch((err) => {
            console.log(err);
        })
});
```

Ruta /restaurant/favorite koja je get ruta nam vraća listu restorana koje je korisnik označio kao omiljene.

Funkcijom .render otvaramo templejt favorite, dok rezultat prosleđujemo kao objekat favRestaurant koji kasnije prikazujemo na stranici.

```
// Update one
router.post('/restoran/single/update', function(req, res) {
    const {_id} = req.body;
    const {ime} = req.body;
    const {opis} = req.body;
    const {lokacija} = req.body;
    console.log(_id);
    FavoriteRestaurant.updateOne({_id:_id}, {ime: ime, opis: opis, lokacija: lokacija}
        if(err) throw err;
        console.log('Restoran update-ovan');
    })
    res.redirect("/restaurant/favorite");

});
```

Ruta Update one nam služi da korisnik može da izvrši promene na odabranom restoranu iz svoje liste sa omiljenim restoranima. Podaci se preuzimaju iz forme putem req.body, a zatim se metodom updateOne vrši query koji nam vrši update and određenim restoranom, a zatim redirektuje na rutu /restaurant/favorite.

Modeli

Model određuje kako treba izgledati podatak koji upisujemo ili čitamo iz baze, dakle kao neka vrsta skice samog objekta. Mi u aplikaciji imamo 2 modela, to su restaurant i favRestaurant. Da bismo upotrebili modele potrebno ih je kreirati putem mongoose.Schema metode.

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const restaurantSchema = new Schema({
    ime: {
        type: String,
        required: true
    },
    radno_vreme: {
        type: String,
        required: true
    },
    telefon: {
        type: String,
        required: true
    },
    lokacija: {
        type: String,
        required: true
    },
    meni: {
        type: Array,
        required: true
    },
    opis: {
        type: String,
        required: true
    }
});
```

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const favRestaurantSchema = new Schema({
    ime: {
        type: String,
        required: true
    },
    opis: {
        type: String,
        required: true
    },
    lokacija: {
        type: String,
        required: true
    }
});
```

Kao što se može videte sa slika, imamo objekat koji ima dva svojstva, to su type koji predstavlja tip koji možemo očekivati, I required koji nam određuje da li je polje obavezno ili ne.

App.js

App.js je glavni fajl koji pokreće ceo projekat. U njemu je neophodno registrovati rute koje koristimo, kao I povezati aplikaciju sa bazom podataka, kao I odrediti koji se templejt jezik koristi za prikaz strana korisniku.

```
const express = require('express');
const mongoose = require('mongoose');
const restaurantRoutes = require('./routes/restaurantRoutes');

//express app
const app = express();

//conect to mongodb
const dbURI = 'mongodb+srv://nikola_nodejs:nikola123@nodejsapp.we6ko.mongodb.net/restaura
mongoose.connect(dbURI, { useNewUrlParser: true, useUnifiedTopology: true })
    .then((result) => app.listen(3000))
    .catch((err) => console.log(err));

//register view engine
app.use(express.urlencoded({ extended: true }));
app.use(express.static('public'));
app.set('view engine', 'ejs');


//Restaurant Routes
app.use(restaurantRoutes);
```

Public

U public folderu se nalaze fajlovi koje je neophodno prikazati na front-endu, kao što su css I js fajl. Da bi ovaj fajl bio vidljiv browser-u, nepohodno je da bude static, a to postižemo sledećom linijom koda:

App.use(express.static(folder));

Views

Views folder sadrži sve naše .ejs strane koje prikazujemo korisniku, ali takođe sadrži I pod folder partials u kome se nalaže dva fajla, footer.ejs I header.ejs. Ovaj podfolder kreiramo za one fajlove koje koristimo više puta, odnosno u našem slučaju na svakoj .ejs strani I samim tim smanjujemo pisanje koda I činimo naš kod preglednijim I čitljivijim.

Zaključak

Uz pomoć Node.js-a možemo koristiti JavaScript jezik I na frontend-u I na backend-u. Ovaj framework nam omogućava izvršavanje JavaScript koda van web browser-a što programerima jako puno olakšava izradu ovakvih aplikacija jer mogu sve odraditi u jednom programskom jeziku, bez upotrebe nekog drugog backend jezika I iz te prespektive dosta olakšava posao. EJS templejt jezik nam omogućava da naše stranice ne budu statične, vec dinamične I da je moguće menjati sadržaj na osnovu podataka iz baze podaka koju korsnik koristi.