

Principi Programskih jezikov

05.06.2022

Skupina: Najaki
Stefan Srnjakov
Makedonka Binova
Kristijan Mitrov



Vsebina

Viri potrebnih informacij	3
Implementacija.....	4
Zagon.....	5
Vizualizacija	6
Namizna aplikacija.....	7
Začetna stran	7
Ustvarjanje novega računa iz nič	8
Registriranje že obstoječo restavracijo	9
V kodi	10
Razred DatabaseConnect	11

Viri potrebnih informacij

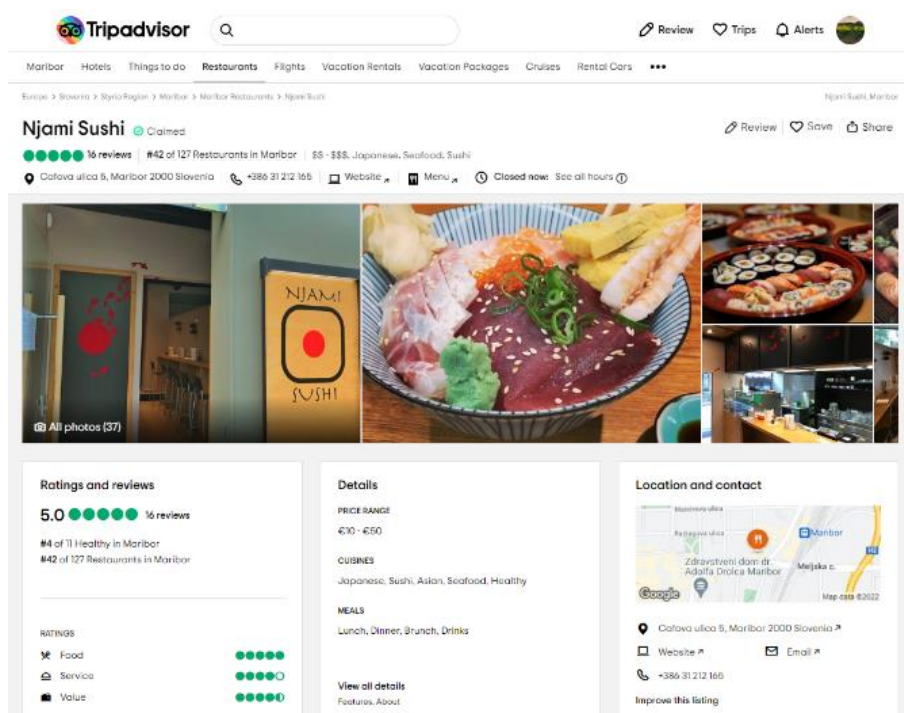
Za našo namizno in spletno aplikacijo 'eatical' smo najprej potrebovali podatke o restavracijah v Mariboru. Namen naše aplikacije je vzpostaviti povezavo med restavracijami in strankami, ki si skupaj želijo čistejše okolje in manj nepotrebnih odpadkov. Tudi na enem mestu imajo prebivalci in turiste informacije o najboljših restavracijah v mestu.

Informacije, ki smo jih potrebovali za shranjevanje restavracij v bazo podatkov, smo posredovali prek spletne strani:

[https://www.tripadvisor.com/Restaurants-g274874-Maribor Styria Region.html](https://www.tripadvisor.com/Restaurants-g274874-Maribor_Styria_Region.html)

S shranjevanjem podatkov, ki jih posreduje to spletno mesto, ne dobimo dovolj informacij. Kak bi dobili vse informacije ki jih rabimo je bli nujno da uporabnime podatki tudi od posamezne spletne strani za vsaka restavracija.

https://www.tripadvisor.com/Restaurant_Review-g274874-d14105839-Reviews-Njami_Sushi-Maribor_Styria_Region.html



Slika 1. Trip Advisor

Implementacija

Podatki za vsako restavracijo so shranjeni v bazi podatkov mongodb. MongoDB shranjuje podatke v ionski obliki.

```
4  const restaurantSchema = new Schema({
5    'username' : String,
6    'password' : String,
7    'name' : String,
8    'address' : String,
9    'opening_hours' : String,
10   'email' : String,
11   'telephone' : String,
12   'website' : String,
13   'directions_link' : String,
14   'orders' : [{type: Schema.Types.ObjectId, ref: 'order'}],
15   'image_id' : {
16     type: Schema.Types.ObjectId,
17     ref: 'image'
18   },
19   'meals' : [{type: Schema.Types.ObjectId, ref: 'meal'}],
20   'place_id' : String,
21   'google_rating' : Number,
22   'location' : {
23     type: {
24       type: String,
25       enum: ['Point'],
26       required: true
27     },
28     coordinates: {
29       type: [Number],
30       required: true
31     }
32   },
33   'ratings' : [{type: Number}]
34 });
```

Slika 2. JavaScript Shema

```
_id: ObjectId('629a71a4d2752654d71f4a78')
name: "Jack & Joe Steak and Burger Club"
address: "Ob bregu 20, Maribor 2000 Slovenia"
opening_hours: "11:00 AM - 12:00 AM"
email: "steak@jackandjoe.si"
telephone: "+386 51 370 621"
website: "http://jackandjoe.si/"
> orders: Array
> meals: Array
google_rating: 4.5
> location: Object
> ratings: Array
__v: 4
image_id: ObjectId('629cd50c2f2ed9fddd600c71')
password: Binary('JDDJhJDEwJG18Tm1wVEJNcndiU3VUS3FUSGgza2VJSURyMGhSeEdkekeSEUTJud01Ba3hadDJkdnRyYl1T', 0)
username: "jackAndJoe"
```

Slika 3. Primer Dokument v MongoDB

Program za pridobivanje in razčlenjevanje html strani je napisan v NodeJs. Knjižnice in moduli za pridobivanje spletne strani v html formatu.

<https://www.npmjs.com/package/puppeteer>

<https://www.npmjs.com/package/axios>

Knjižnice in moduli za pridobivanje spletne strani za razčlenjevanje podatkov.

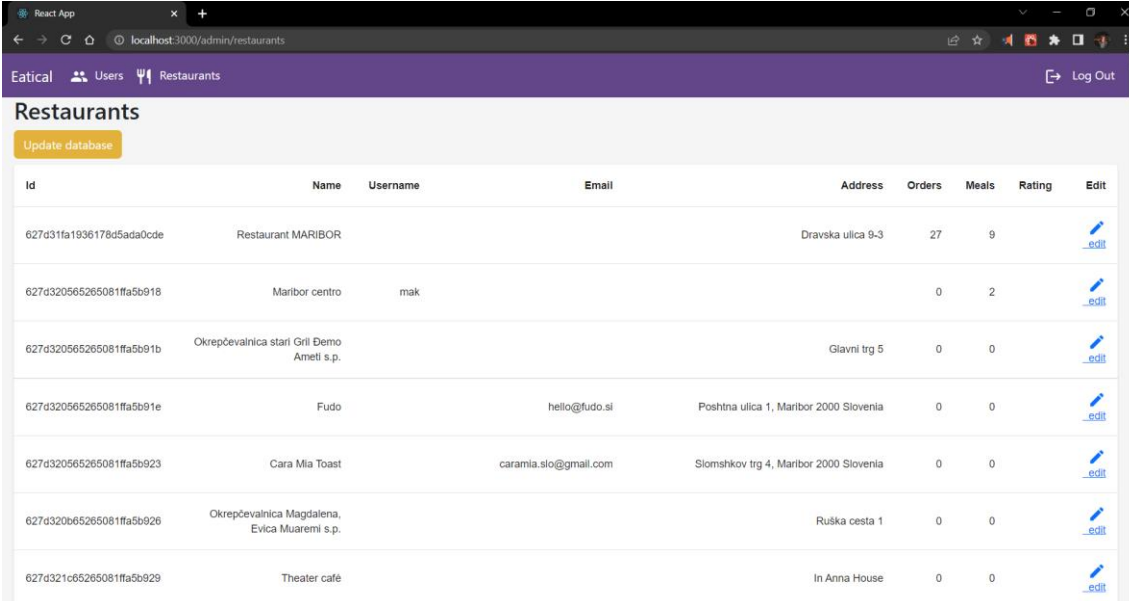
<https://cheerio.js.org/>

Zagon

Na našem zalednem strežniku je pot, ki kliče razčlenjevalnik. Naša zapletena aplikacija omogoča registracijo skrbnikov, ki imajo v vmesniku gumb, ki pošlje zahtevo strežniku.

```
21 */
22 router.post( path: '/', restaurantController.create);
23 router.post( path: '/login', loginValidation, restaurantController.login);
24 router.post( path: '/logout', authenticateRestaurant, restaurantController.logout);
25 router.post( path: '/api', restaurantController.refreshRestaurants);
26 */
27 * PUT
```

Slika 4. Funkcija refreshRestaurants

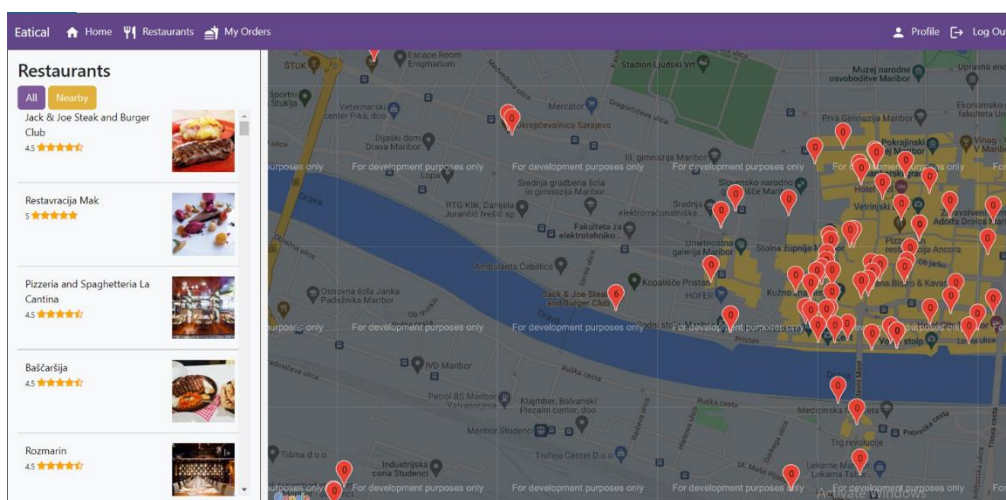


Id	Name	Username	Email	Address	Orders	Meals	Rating	Edit
627d31fa1936178d5ada0cde	Restaurant MARIBOR			Dravska ulica 9-3	27	9		edit
627d320565265081ffa5b918	Maribor centro	mak			0	2		edit
627d320565265081ffa5b91b	Okrepčevalnica stari Grli Demo Armeti s.p.			Glavni trg 5	0	0		edit
627d320565265081ffa5b91e	Fudo		hello@fudo.si	Poshtna ulica 1, Maribor 2000 Slovenia	0	0		edit
627d320565265081ffa5b923	Cara Mia Toast		caramia.slo@gmail.com	Slomshkov trg 4, Maribor 2000 Slovenia	0	0		edit
627d320b65265081ffa5b926	Okrepčevalnica Magdalena, Evica Muaremi s.p.			Ruška cesta 1	0	0		edit
627d321c65265081ffa5b929	Theater café			In Anna House	0	0		edit

Slika 5. Gumb ki zažene funkcijo

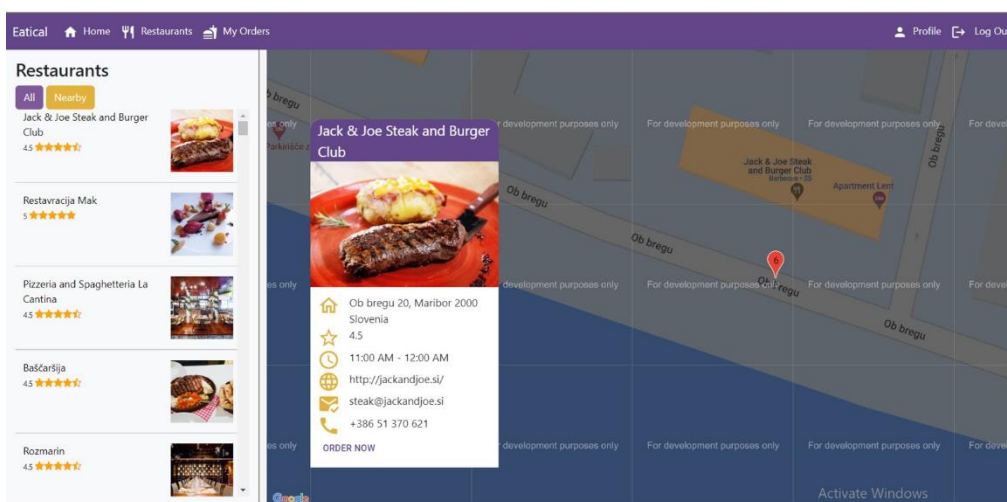
Vizualizacija

Uporabljamo React za prikaz naših informacij. Na našem vmesniku si lahko ogledate splošne informacije o posamezni restavraciji in kartografski pogled z možnostjo ogleda najbližje ali vseh restavracij.



Slika 6. Mapa vse restavracije

S klikom na ime katere koli restavracije si lahko ogledate podrobnejše informacije o njej.



Slika 7. Prikaz ene restavracije

Namizna aplikacija

V naši spletni aplikaciji sta dve strani: restavracije in stranke. Stranke in restavracije se lahko prijavijo v obstoječi račun prek naše spletne aplikacije, stranke pa lahko tudi ustvarijo novi račun. Da bi restavracija ustvarila svoj račun, smo izdelali majno namizno aplikacijo z pomočjo ogrodja Compose Multiplatform.

Začetna stran

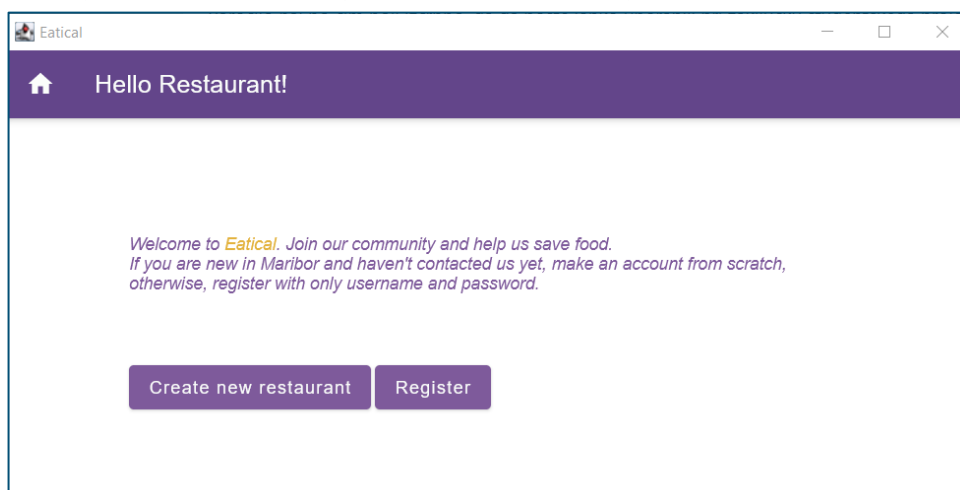
Če neka aplikacija želi sodelovati z nami obstajajo dva načina za to:

- Ustvarjanje novega računa iz nič
- Registriranje že obstoječo restavracijo

Mi že imamo vse podatke za večinoma restavracij v Mariboru, ki smo jih dobili s pomočjo HTML razčlenjevanja. Zanje ustvarjamo brezplačen oglas, tako da na zemljevidu prikažemo vse restavracije, tudi tiste, ki niso naročene na naše storitve (niso registrirani).

Restavracija se odloči za prvi način če še ni v naši bazi in mora ustvariti svoj račun iz nič z vsemi potrebnimi podatki (vključno s koordinatami).

Restavracija se odloči za drugi način če ima vse potrebne podatke za prikaz v mapi (je že v naši bazi ali nas je kontaktirala kako bi jo vstavili v bazo ročno). Za to vrsto registracije sta potrebna samo uporabniško ime in geslo.



Slika 8. Začetni intreface

Ustvarjanje novega računa iz nič

V vnosna polja mora restavracija vnesti:

- Ime
- Uporabniško ime (vsaj tri znake)
- Geslo (vsaj 6 znake)
- Naslov
- Zemljepisna širina (v pravilnem formatu in z vrednostmi med –90 in 90)
- Zemljepisna dolžina (v pravilnem formatu in z vrednostmi med –180 in 180)

Slika 9. Ustvarjanje novi račun

Majhni del kode za robustnosti prijave:

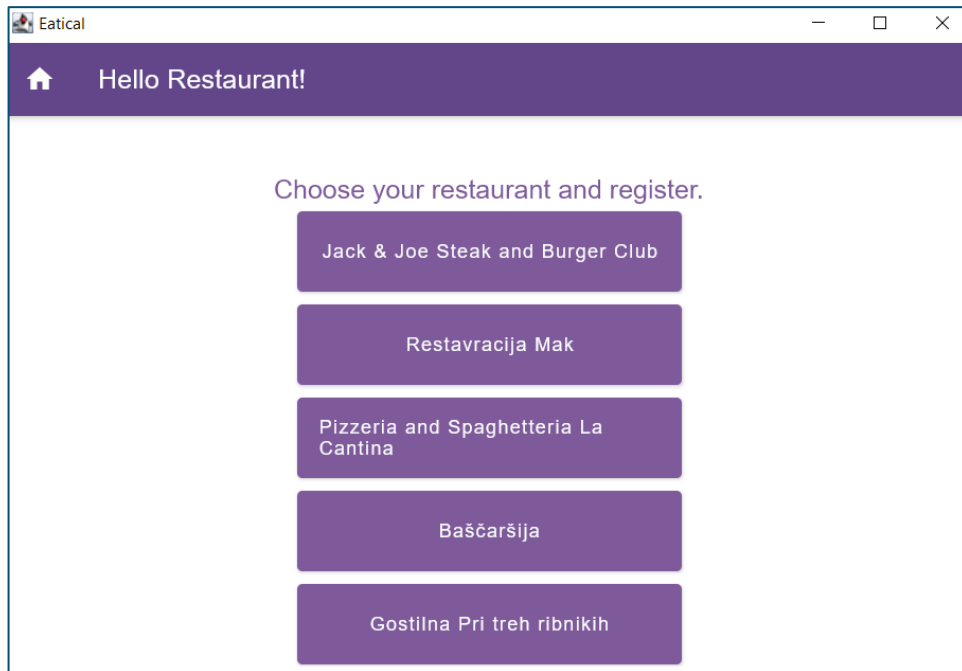
```
Button(onClick = {  
    latitudeFormatChecker.value = latitude.value.text.matches("-?\\d+(\\.\\d+)?".toRegex())  
    longitudeFormatChecker.value = longitude.value.text.matches("-?\\d+(\\.\\d+)?".toRegex())  
    if(latitude.value.text != "" && latitudeFormatChecker.value)  
        latitudeValueChecker.value = BigDecimal(latitude.value.text).compareTo(BigDecimal( val: -91)) == 1 &&  
        BigDecimal(latitude.value.text).compareTo(BigDecimal( val: 91)) == -1  
    if(longitude.value.text != "" && longitudeFormatChecker.value)  
        longitudeValueChecker.value = BigDecimal(longitude.value.text).compareTo(BigDecimal( val: -181)) == 1 &&  
        BigDecimal(longitude.value.text).compareTo(BigDecimal( val: 181)) == -1  
    if(name.value.text == "")  
        error.value = "Please enter name."  
    else if(username.value.text == "")  
        error.value = "Please enter username."  
    else if(username.value.text.length < 3)  
        error.value = "Please enter username with at least 3 characters."  
    else if(password.value.text == "")  
        error.value = "Please enter password."  
    else if(password.value.text.length < 6)  
        error.value = "Please enter password with at least 6 characters."  
    else if(repeatPassword.value.text == "")
```

Slika 10. Koda aplikacije

Ko si restavracija ustvari novi račun jo lahko vidimo v mapi.

Registriranje že obstoječo restavracijo

Od ponujenih restavracij je restavracijo mogoče najti in izbrati za registracijo.



Slika 11. Seznam restavracije

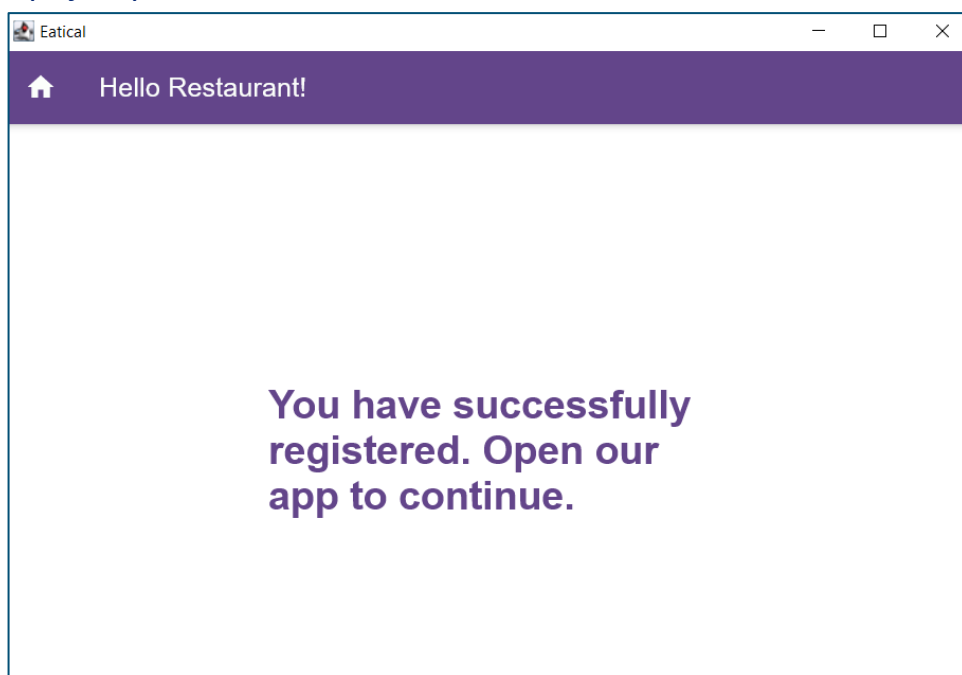
V vnosna polja mora restavracija vnesti:

- Uporabniško ime (vsaj 3 znake)
- Geslo (vsaj 6 znake)

A screenshot of the same "Eatical" application window, but now showing a registration form. The header remains the same. The main content area contains three input fields stacked vertically, each with a label and a value: "username" with the value "bascarsija", "password" with a masked value "*****", and "repeat password" with a masked value "*****". Below these fields is a purple "Register" button.

Slika 12. Registracija

Ob uspešni prijavi pri obeh načinov:



Slika 13. Sporočilo v aplikaciji

Na "HOME" ikono lahko gremo spet na začetno stran in ustvarimo novo restavracijo/registriramo obstoječo.

V kodi

- Razred Location za vnos koordinate kot GeoJSON objekt v bazo.
- Razred Restaurant za prikaz vseh restavracij da bi izbrali eno za registracijo.
- Barve, ki smo jih izbrali kot našo paleto in s katerimi najprej v kodo barvamo komponente.
- Composable tableScreen za prikaz vseh restavracij da bi izbrali eno za registracijo.
- Composable newRestaurantForm – obrazec za ustvarjanje novi restavracijski račun
- Composable oldRestaurantForm – obrazec za registriranje že obstoječo restavracijo
- Composable inputField – navadni TextField znotraj, kako bi zmajnsali kodo.

```

class Location(val type: String, val coordinates: Array<BigDecimal>)
class Restaurant(val _id: Id<String> = newId(), val name: String)

val CYBER_GRAPE = Color( color: 0xff63458A)
val ROYAL_GRAPE = Color( color: 0xff7E5A9B)
val MAXIMUM_YELLOW_RED = Color( color: 0xffE3B23C)
val BABY_POWDER = Color( color: 0xffFBFEFB)

@Composable
fun tableScreen(database: DatabaseConnection, something: FindIterable<Restaurant>) {...}

@Composable
@Preview
fun app(database: DatabaseConnection) {...}

@Composable
fun newRestaurantForm(database: DatabaseConnection){...}

@Composable
fun oldRestaurantForm(database: DatabaseConnection, restaurant: Restaurant){...}

@Composable
fun inputField(type: String, state: MutableState<TextFieldValue>){...}

fun main() = application {...}

```

Slika 14. Koda Aplikacije

Razred DatabaseConnect

Pomaže nam da upravljamo s potrebnimi podatki v naši bazi. V tem primeru so to restavracijah, zato imamo metode kot so insertRestaurant, updateRestaurant, getRestaurants.

```

class DatabaseConnection(url: String, name: String) {
    private val client: MongoClient
    private val database: MongoDBDatabase

    init{
        client = KMongo.createClient(url)
        database = client.getDatabase(name)
    }

    fun insertRestaurant(document: Document){
        val collection = this.database.getCollection( collectionName: "restaurants")
        collection.insertOne(document)
    }

    fun updateRestaurant(newDocumentValue: BasicDBObject, restaurantId: Id<String>){
        val collection = this.database.getCollection( collectionName: "restaurants")
        val query = BasicDBObject()
        query.put("_id", restaurantId) //filtering the restaurant that needs update

        val updateObject = BasicDBObject()
        updateObject.put("$set", newDocumentValue)

        collection.updateOne(query, updateObject)
    }

    fun getRestaurants(): MongoCollection<Restaurant> = database.getCollection<Restaurant>( collectionName: "restaurants")
}

```

Slika 15. Povezava z database

Viri

- <https://www.tripadvisor.com>
- <https://www.npmjs.com/package/puppeteer>
- <https://www.npmjs.com/package/axios>
- <https://www.npmjs.com/package/axios>
- <https://cheerio.js.org/>
- <https://www.mongodb.com/>
- <https://reactjs.org/>
- <https://developer.android.com/jetpack>
- <https://foso.github.io/Jetpack-Compose-Playground/>
- <https://stackoverflow.com/search?q=jetpack+compose>
- <https://github.com/ToxicBakery/bcrypt-mpp>