# Web aplikacija:

# Iznajmljivanje e-trotineta

Mentor:

Ilija Antović

Student:

Veljko Cukanić 2022/0005

# Opis Sistema

Ova aplikacija omogućava korisnicima da rezervišu e-trotinet na određenoj lokaciji u određenom vremenskom terminu ukoliko je on slobodan. Aplikacija je građena korišćenjem tehnologija React za frontend i Spring Boot za backend, i omogućava uvid u sve moguće termine na osnovu unetih parametara i upravljanje napravljenim rezervacijama takodje.

# Slučajevi korišćenja

SK1- Registracija: korisnik unosi svoje podatke i kreira profil

SK2- Prijavljivanje na sistem(Logovanje): korisnik unosi svoje kredencijale i pristupa stranici za rezervisanje

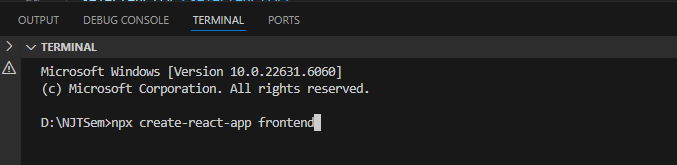
SK3-Odjavljivajne sa sistema: korisnik se odjavljuje sa sistema

SK4-Kreiranje iznajmljivanja: korisnik bira parametre i kreira iznajmljivanje

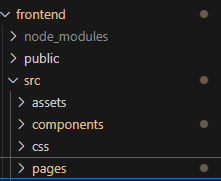
SK5-Izmena iznajmljivanja: korisnik bira jedno od kreiranih iznajmljivanja i vrši izmenu

SK6-Brisanje iznajmljivanja: korisnik briše iznajmljivanje

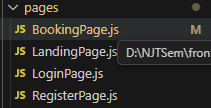
# Razvijanje Fronenda

Počinjemo kreiranjem React projekta, potrebno je instalirati Node.js sa sajta **https://nodejs.org**, a zatim u terminalu otvorimo folder naseg projekta i pokrenuti komandu **npx create-react-app frontend** 

Zatim kreiramo strukturu foldera:

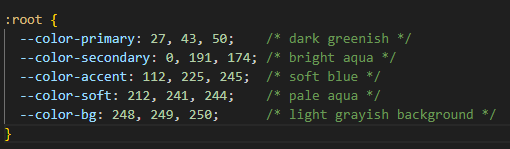


Sledi kreiranje fajlova za stranice i konfiguracija globalnog router taga u App.js, ovim kodom uspostavljamo strukturu patha (recimo localhost:3000/booking):





Zatim pravimo i globalni set boja u App.css:



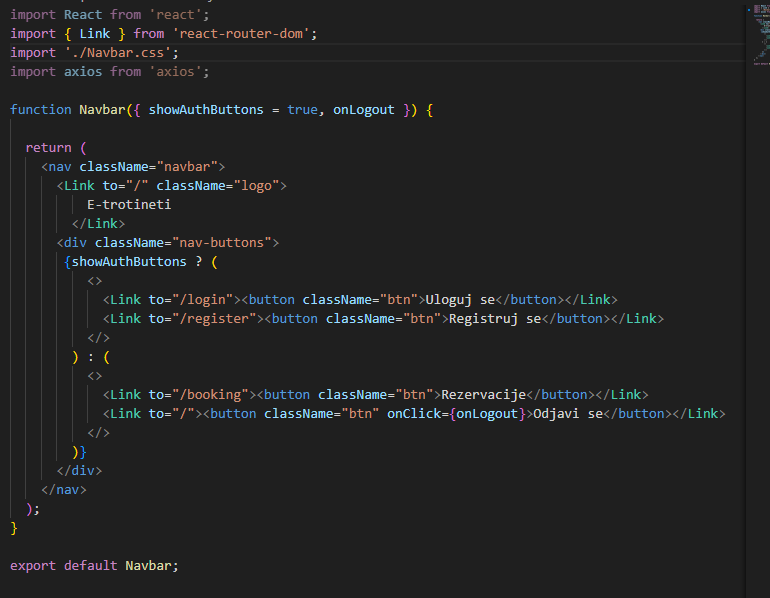
Prelazimo na lending page i postavljanje komponente navbar na nju, možemo videti da se koristi useState za user sto predstavlja ulogovanog korisnika i Location koji nam u useEffect omogućava da izvršavamo fetchUser kad god dodjemo na path ove stranice, navbar prima parametre showAuthButton koji gleda da li postoji ulogovan user i onLogout koji klikom na dugme za odavu šalje serveru post koji odjavljuje korisnika:



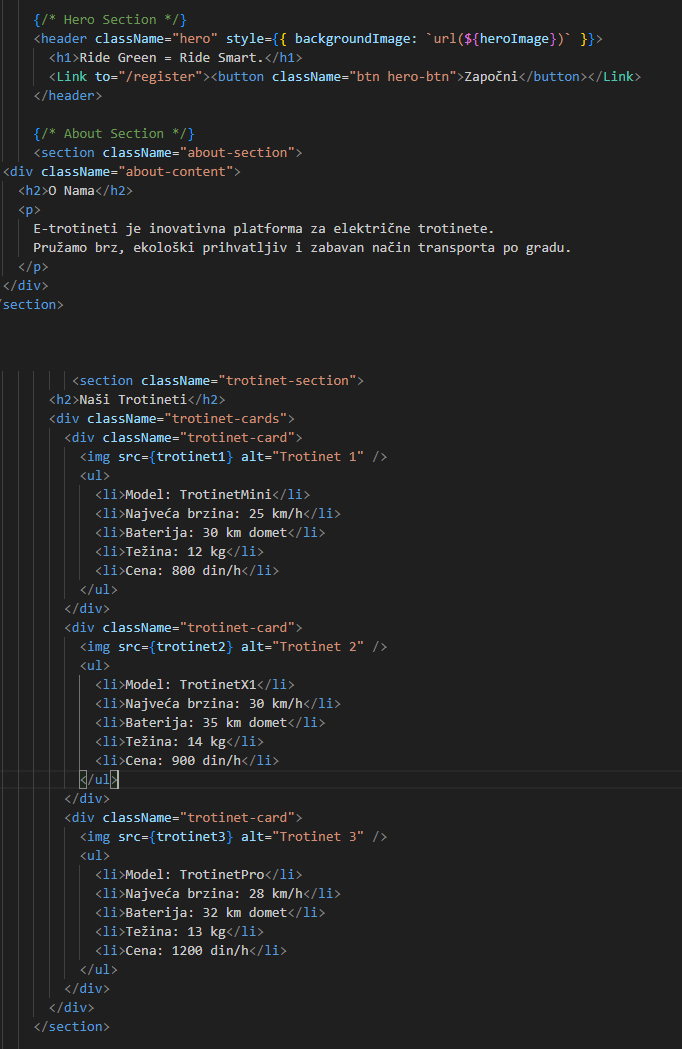
Css za bookingpage:



Prelazimo na navbar komponentu, kod nje je specifično što se menjaju prikazana dugmad od zavisnosti od toga da li je korisnik ulogovan:

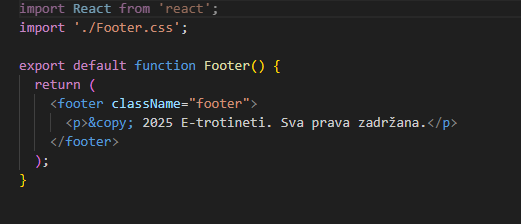


Zatim na landing page imamo hero, about i trotinet sekcije kod kojih nema specifičnosti u kodu osim što su kartice za prikaz trotineta tipa flex:



I na kraju sekcija za lokacije, ggde je korišćen iframe tag i google mape i futer:



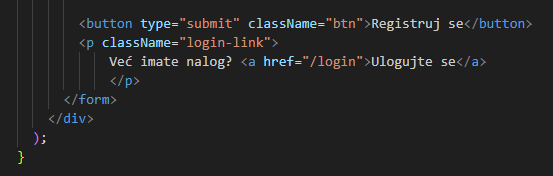


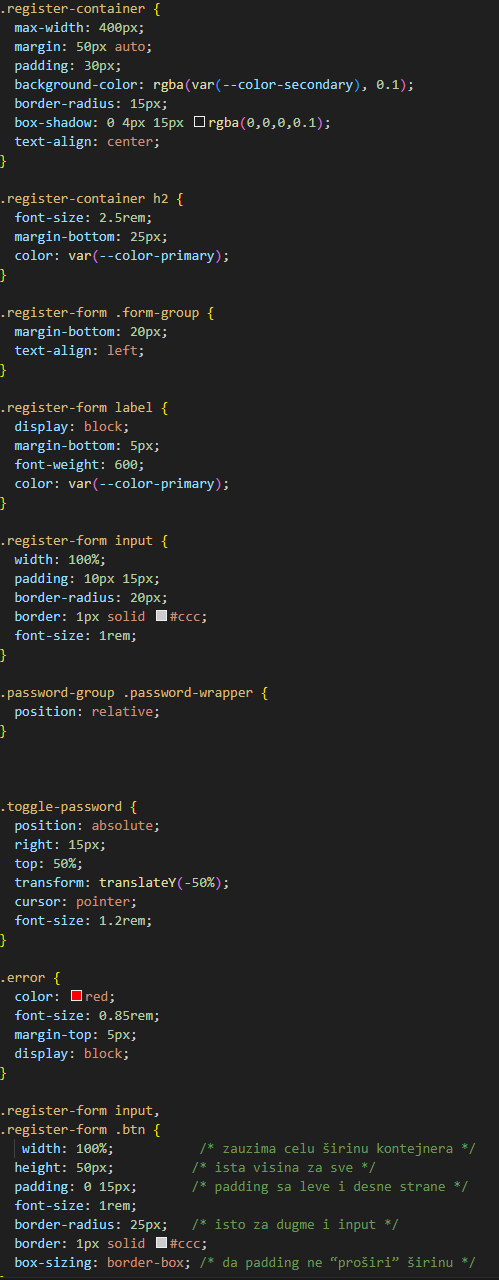
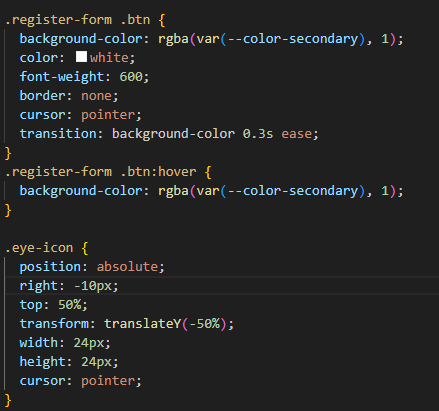
Ovime završavamo lending page i prelazimo na register page, ovde vidimo da imamo više useState elemanata koji nam služe za prikupljanje podataka sa forme(formData), prikazivanje i sakrivanje lozinke(showPassword) i za pamćenje gešaka (errors), imamo regex validaciju da bismo obezbedili ispravnost podataka i handleChange koji nam omogućava da svi podaci dinamički budu zapamćeni pri promeni, handleSubmit prosledjuje formData serveru koji kreira korisnički nalog koristeći sxios post:



Ovde vidimo html i css ove forme:





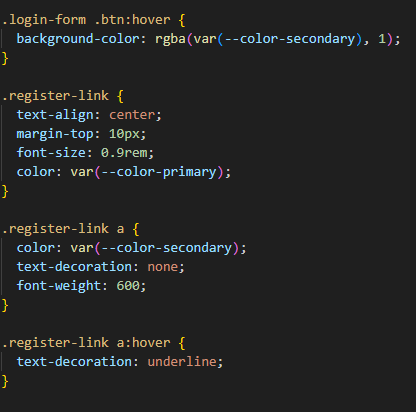
Prelazimo na login page koja se ponaša isto kao i register page samo što zahteva samo email i lozinku, ovde možemo videti tu logiku:



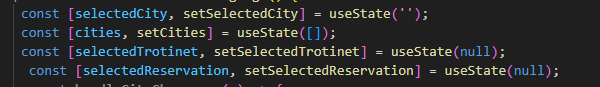
Zatim vidimo html i css ove strane koji nemaju specifičnosti sem eventualno transition u css koja služi za menjanje ikonice oka kao i na register page:

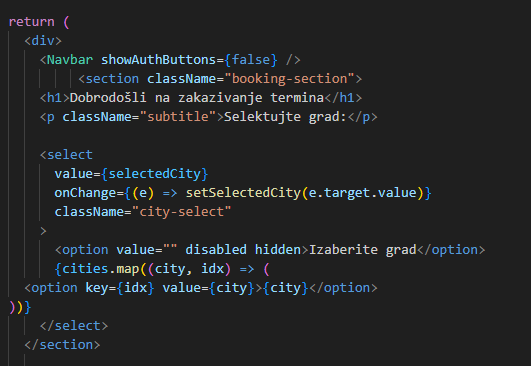
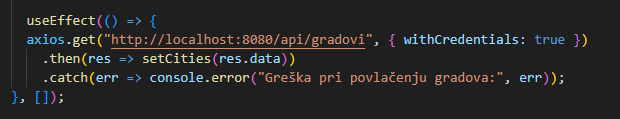




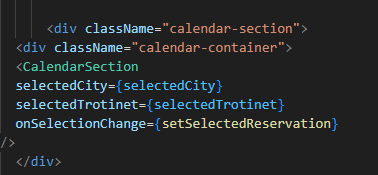


I prelazimo na booking page možemo na prvoj slici videti sve useState komponente, zatim počećemo od ubacivanja navbar komponente (po difoltu false jer neulogovan korisnik ne može pristupiti ovoj strani) i select (dropdown) sekcije koja se sa servera puni dostupnim lokacijama što se vidi na poslednjoj slici :

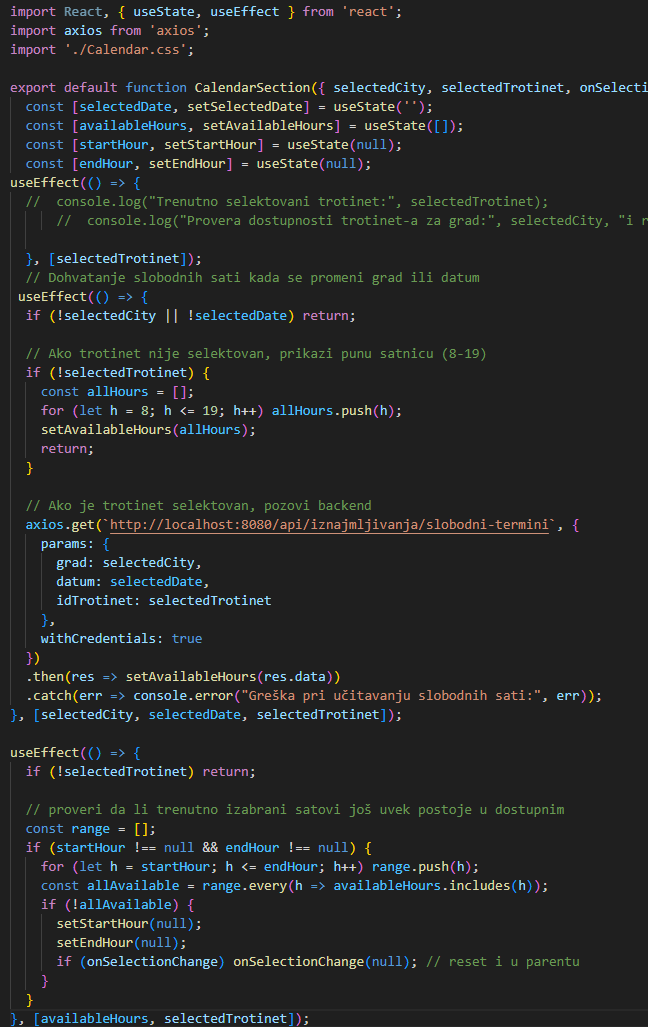


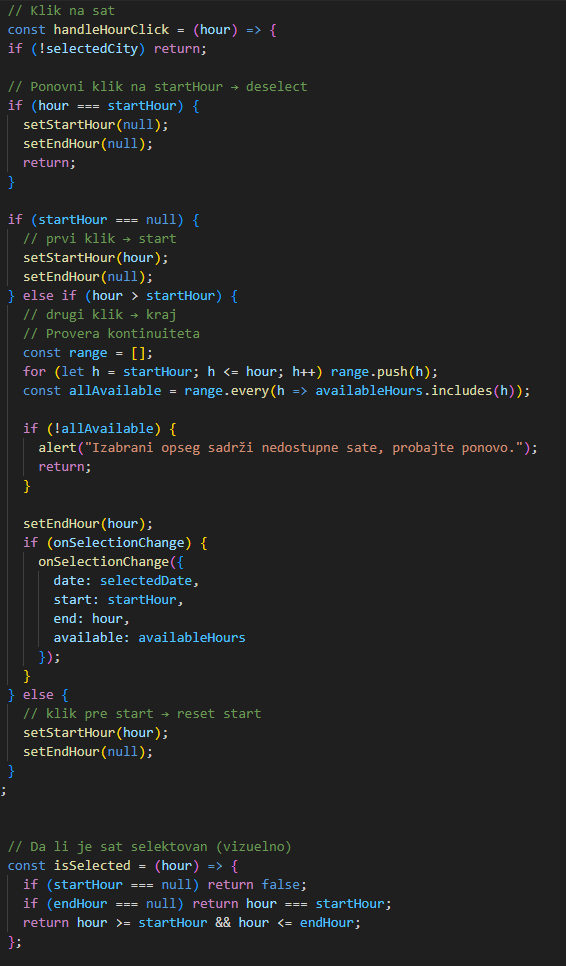
Prelazimo na komponentu kalendara, na prvoj slici ubacivanje na booking page, ona zavisi od selektovanog grada, selektovanog trotineta i vraca selectedResercvation sto predstavlja termin:



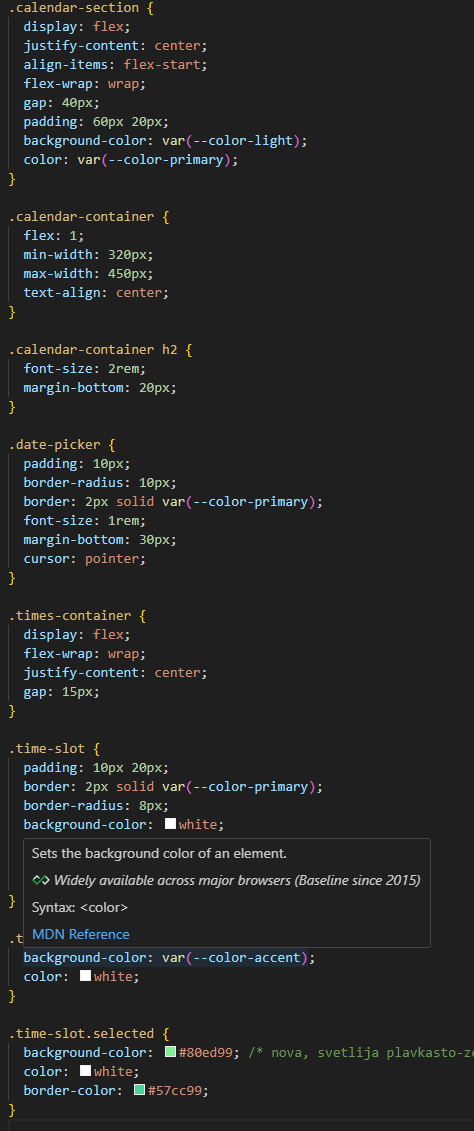
Zatim idemo na komponentu kalendara, na sledećoj slici vidimo useState i useEffect koji nam omogućava da vratimo ili punu slobodnu satnicu ukoliko trotinet nije još uvek selektovan ili ukoliko on to jeste da vrati samo slobodne termine za njeg (više o logici u backendu):



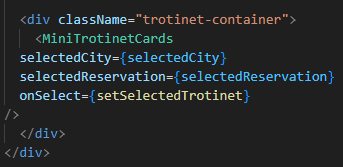
Naredni kod se bavi ponašanjem dugmića kada su kliknuti:



I ovde vidimo html i css:

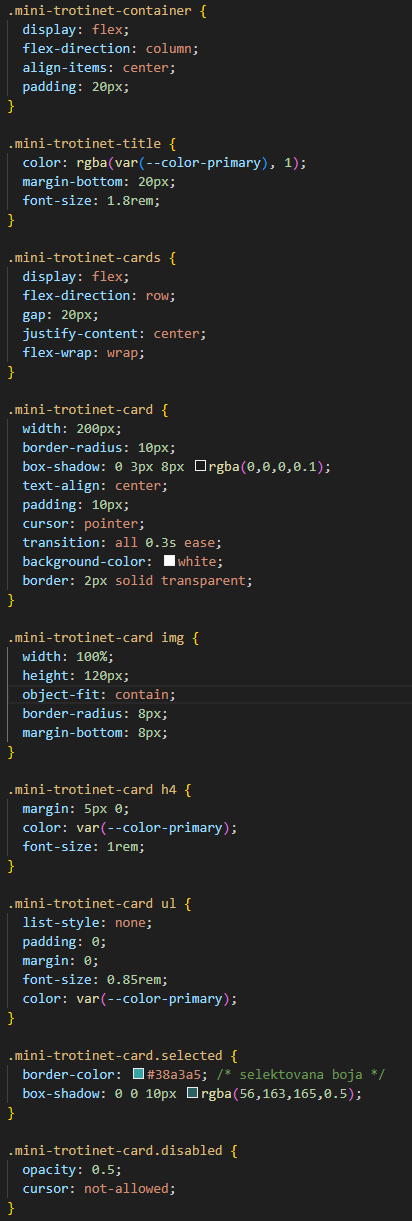
Onda prelazimo na sekciju za odabir trotineta ona zavisi od grada i selektovanog termina i ovo je njen poziv na booking page:



Dalje vidimo logiku ove komponetne koja omogućava da ukoliko nije selektovan ni jedan termin svi trotineti budu dostupni za odabir a ukoliko postoji izabran termin budu dostupni samo oni trotineti koji se mogu iznajmiti u tom terminu kao i handleSelect koji omogućava samu selekciju:



Dalje se mogu videti html i css:

I sada možemo poreći na samu rezervaciju koja se dešava klikom na dugme:



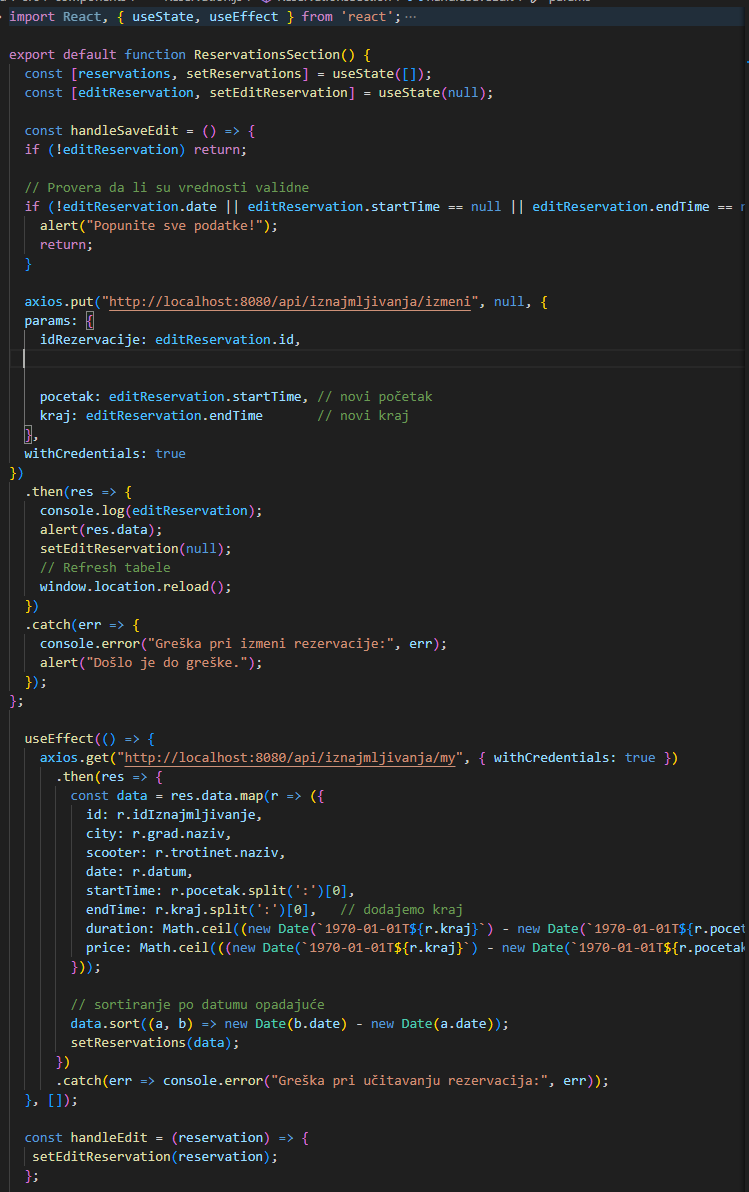
I tada handleRezerviši vrši neophodne provere i šalje post ka serveru sa svim neophodnim podacima:



I na kraju sekcija odnosno komponenta za upravljanje rezervacijama i njen tag na booking page kao:



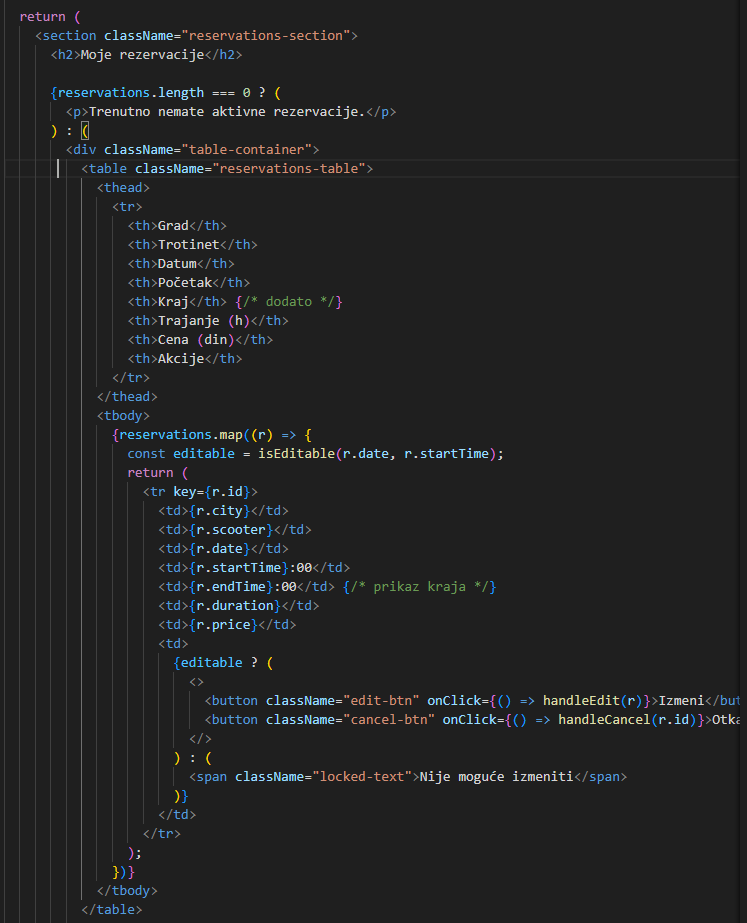
Sada gledamo logiku rezervacije, imamo handleSaveEdit koji serveru šalje post sa izmenjenim podacima nakon izvršenja izmena, i useEffect koji sa servera vraća iznajmljivanja korisnika:



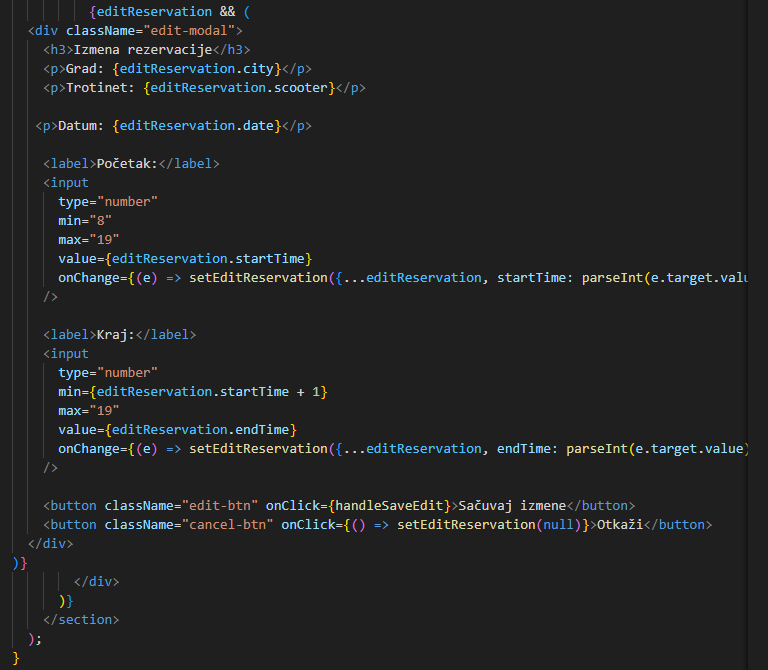
Zatim vidimo deo koda koji rešava izmenu i brisanje kada se kliknu dugmad i proveru da li do termina iznajmljivanja ima 24h da bi se dugmad uopšte prikazala:

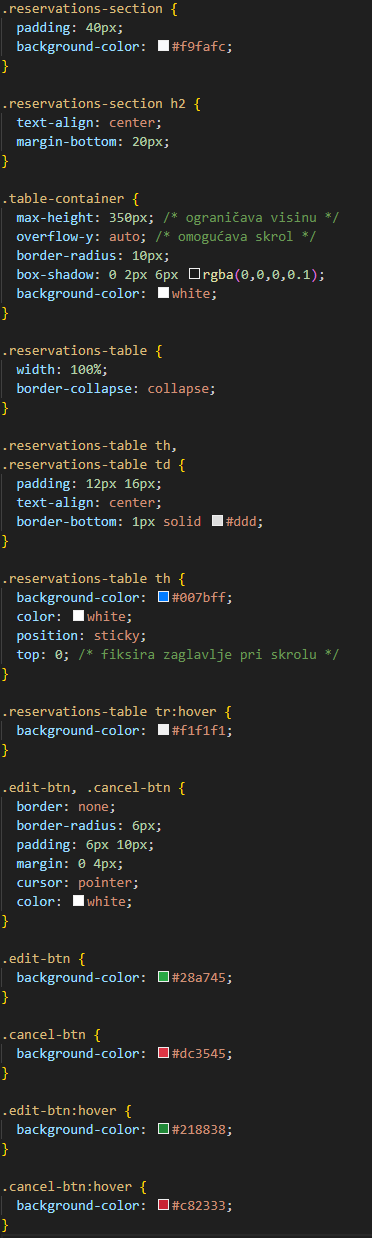


Dalje vidimo tabelu koja ima slucaj kad nema rezervacija slucaj kada je neka rezevacija dostupna za izmenu i kad nije:



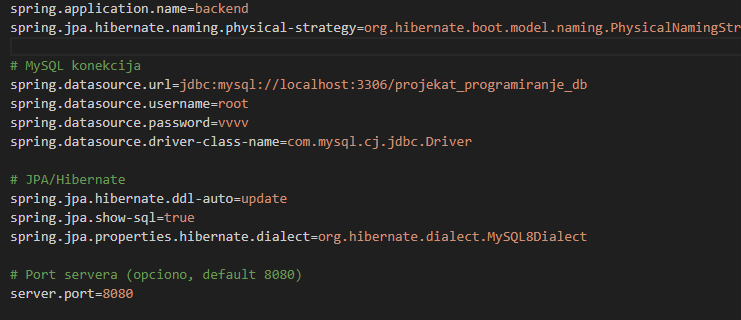
A ovde vidimo html koji se pojavljuje nakon klika na dugme izmeni praćeno css-om ove komponente(overflowz omogućava scrolovanje u tabeli):



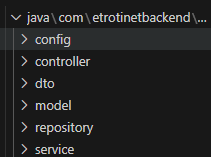


# Razvijanje Backenda

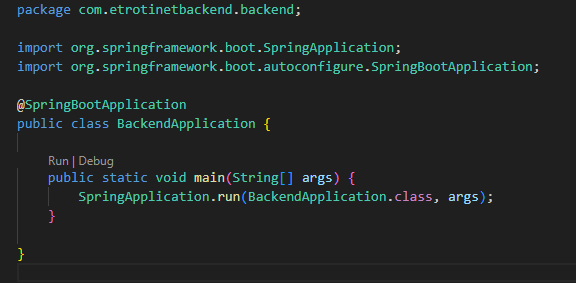
Kreiranje springboot projekta kreće na sajtu <https://start.spring.io>, nakon raspakivanja projekta u fajlu application.properties povezujemo projekat sa bazom :



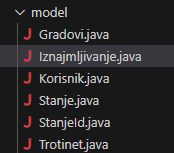
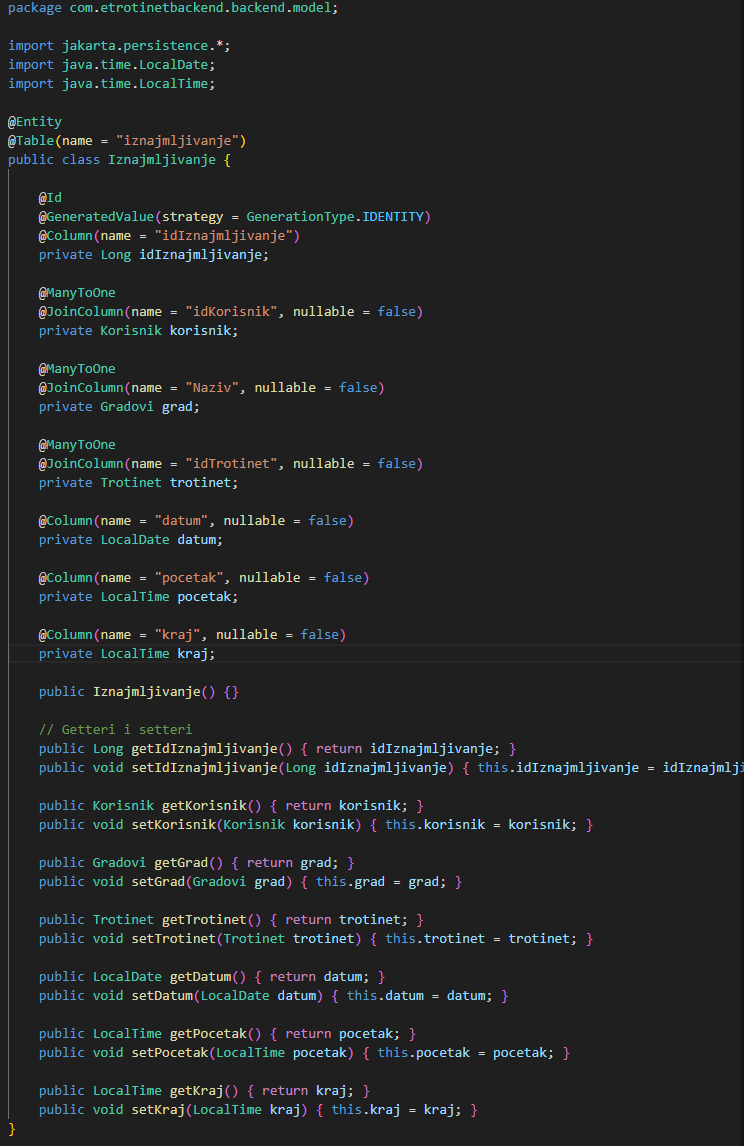
Kreiramo strukturu fajlova:



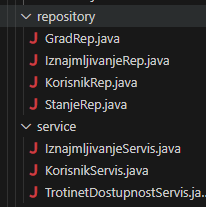
U config pravimo WebConfig koji nam omogućava axios primanje i slanje kroz CORS browser-a:



U model kreiramo klase koje odgovaraju tabelama u bazi, klasa iznajmljivanja kao primer:



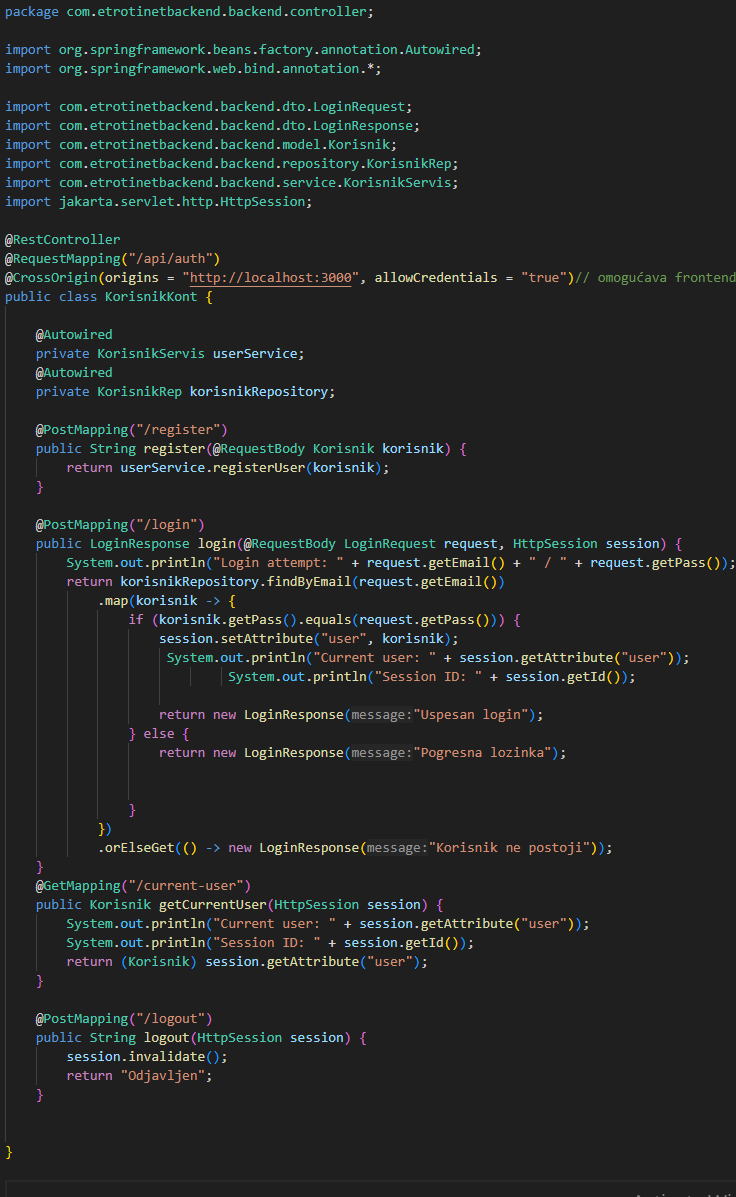
Repository i Service fajlovi definišu funkcije koje ćemo zvati u kontrolerima, repositorz je bitan jer ga JPA koristi za pravljenje sql upita:



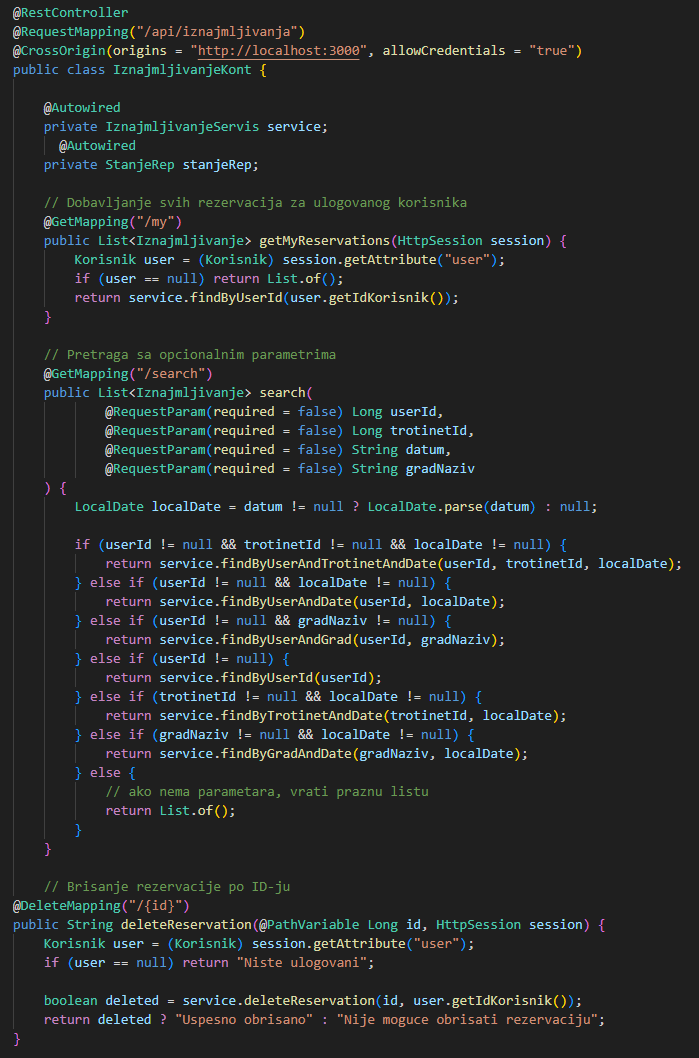
Kontrolor za grad nalazi nazive gradova:

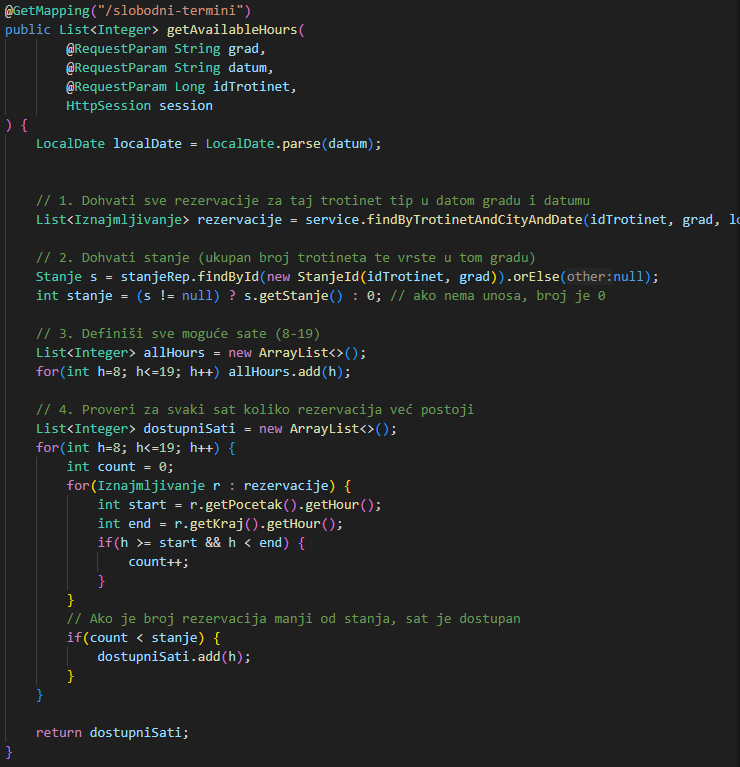


Korisnik kontrolor proverava kredencijale za registraciju i login kao i što vraća korisnika iz trenutne sesije i ubija sesiju kada se izloguje:

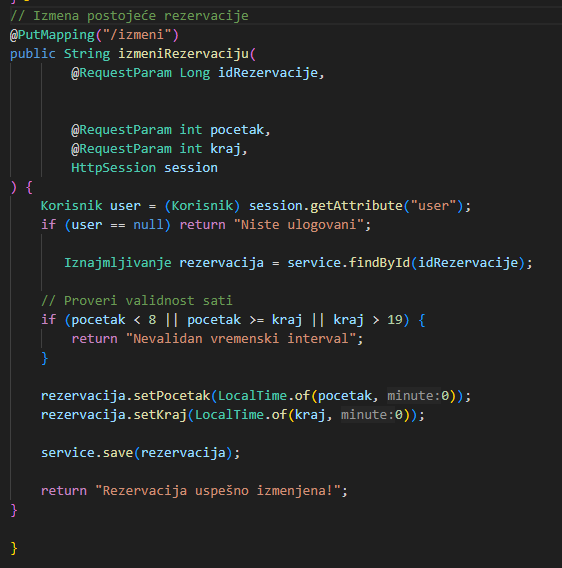


Dalje se prikazuje kontroler iznajmljivanja u lčijim komentarima piše namena koda, sva logika se zasniva na prikupljanju potrebnih podataka iz baze na osnocvu primljenih parametara:









# 5.Dijagram finalne softverske arhitekture

