Univerzitet u Beogradu

Fakultet organizacionih nauka

WEB APLIKACIJA ZA UPRAVLJANJE PARKIRALIŠTIMA

Seminarski rad iz predmeta Internet tehnologije

Beograd, 2020.

Sadržaj

[Korisnički zahtev 3](#_Toc57666896)

[Opis tehnologija korišćenih u izradi aplikacije 4](#_Toc57666897)

[React 4](#_Toc57666898)

[Express 4](#_Toc57666899)

[Axois 5](#_Toc57666900)

[Mysql 5](#_Toc57666901)

[TypeORM 5](#_Toc57666902)

# Korisnički zahtev

Potrebno je napraviti aplikaciju za automatizaciju procesa upravljanja parkirališta. Osnovna funkcionalnost aplikacije bi trebala biti praćenje ulaska i izlaska vozila sa parkirališta kao i kreiranje računa za boravak vozila. Pored toga, potrebno je omogućiti kreiranje i brisanje parkirališta i kreiranje vozila u sistemu kako bi mogli da se sakupe određeni statistički podaci o boravku na parkingu kao što su:

* Prosečno vreme zadržavanja vozila na parkingu
* Prosečan broj vozila na parkingu
* Prosečan broj dolazaka jednog vozila u određenom vremenskom periodu...

Ove statistike je moguće računati na celom skupu podataka, a mogu se i grupisati po raznim kriterijumima( npr. dan u nedelji).

Kao poslednja funkcionalnost javlja se upravljanje korisnicima. Podrazumevano je da postoje dve vrste korisnika: običan korisnik i administrator. Običnom korisniku je omogućeno izdavanje računa, beleženje vremena dolaska i odlaska vozila, odjava sa sistema i promena sopstvenih podataka. Ukoliko korisnik nije prijavljen na sistem ili želi da se registruje on to može učiniti preko odgovarajućih formi. Da bi se korisnik uspešno registrovao potrebno je da zahtev prethodno bude odobren od strane administratora. Administrator, takođe, može vršiti kreiranje, izmenu i brisanje parkirališta, brisanje korisnika, praćenje statistika kao i sve funkcionalnosti koje ima običan korisnik.

# Opis slučajeva korišćenja

Prepoznali smo sledeće slučajeve korišćenja:

1. Prijava korisnika na sistem
2. Registrovanje korisnika
3. Odjava korisnika sa sistema
4. Odobravanje novog korisnika
5. Pregled parkirališta
6. Kreiranje parkirališta
7. Izmena parkirališta
8. Brisanje parkirališta
9. Pregled računa
10. Kreiranje računa
11. Pretraga boravaka
12. Kreiranje boravka
13. Beleženje napuštanja vozila sa parkirališta
14. Pregled popularnosti parkirališta
15. Pregled korisnika

**SK1: Slučaj korišćenja – Pretraga boravka**

**Aktori SK**

Korisnik

**Učesnici SK**

Korisnik i sistem

**Preduslov: Sistem je uključen i korisnik je ulogovan pod svojom šifrom. Sistem prikazuje korsiniku formu za rad sa boravko.**

**Osnovni scenario SK**

1. Korisnik unosi vredonost po kojoj pretražuje boravak. (APUSO)
2. Korisnik poziva sistem da nađe boravak po zadatoj vrednosti. (APSO)
3. Sistem traži boravak po zadatoj vrednosti. (SO)
4. Sistem prikazuje korsiniku boravke. (IA)

Alternativna scenarija

* 1. Ukoliko sistem ne može da nađe boravak on prikazuje korsiniku objavu: „Sistem ne može da nađe boravak“.(IA)

**SK2: Slučaj korišćenja – Kreiranje računa**

**Aktori SK**

Korisnik

**Učesnici SK**

Korisnik i sistem

**Preduslov: Sistem je uključen i korisnik je ulogovan pod svojom šifrom. Sistem prikazuje korsiniku formu za rad sa računom.**

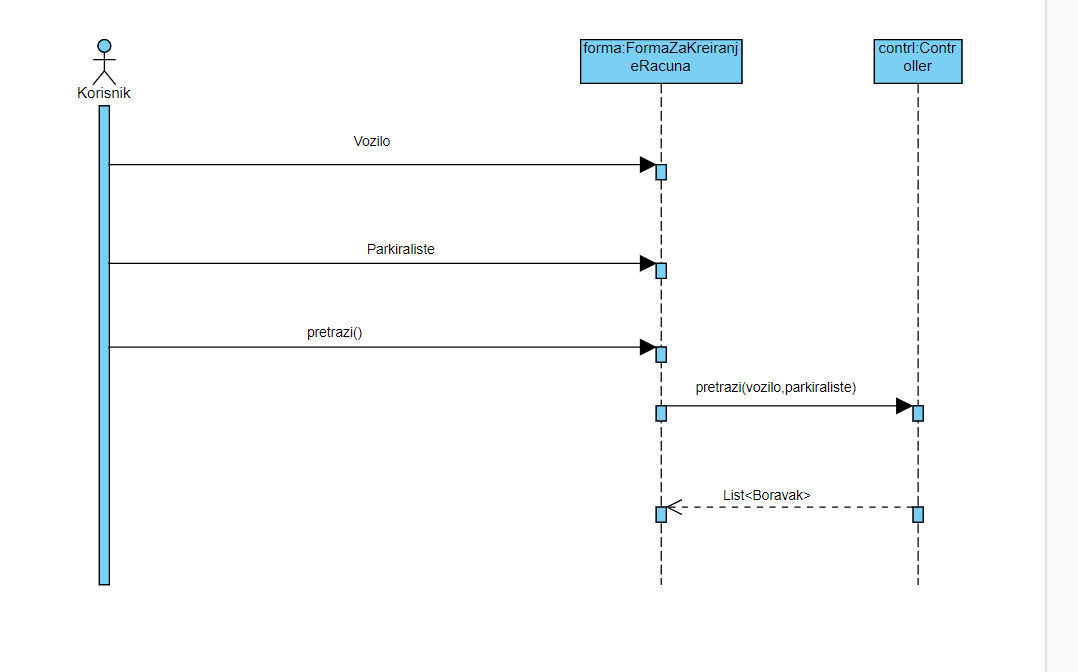
**Osnovni scenario SK**

1. Korisnik bira boravke koje želi da budu uključene u račun.(APUSO)
2. Korisnik poziva sistem da kreira račun.(APSO)
3. Sistem kreira račun.(SO)
4. Sistem prikazuje korisniku novi račun.(IA)

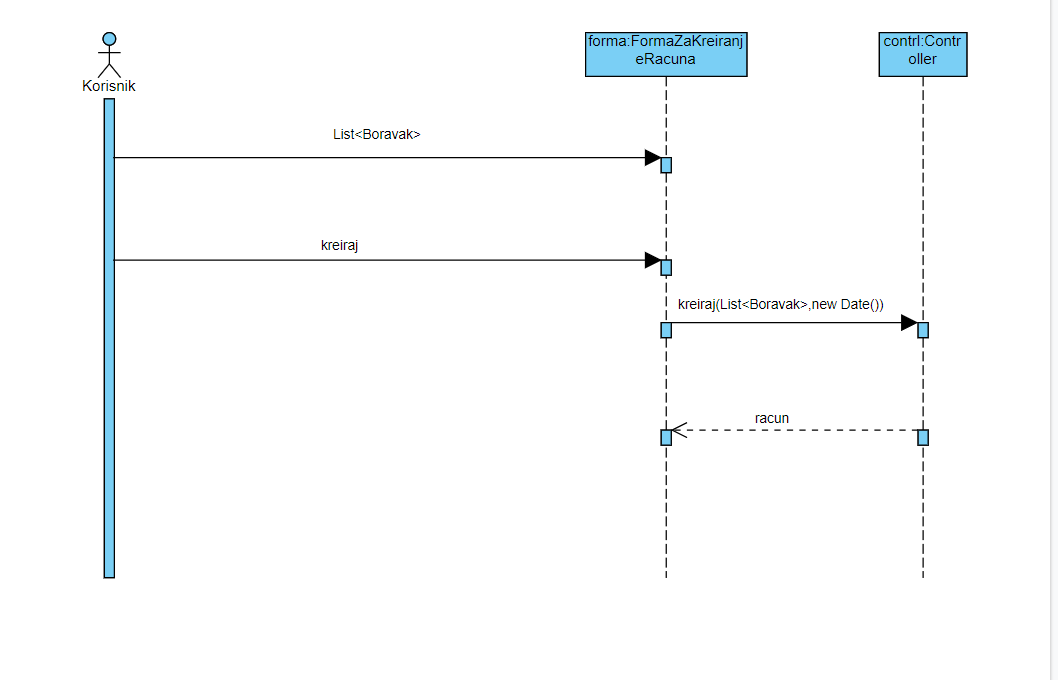
Alternativna scenarija

4.1 Ukoliko sistem ne može da kreira račun on prikazuje korisniku poruku: „Sistem ne može ka kreira račun“.

# Opis procesa – dijagrami sekvenci

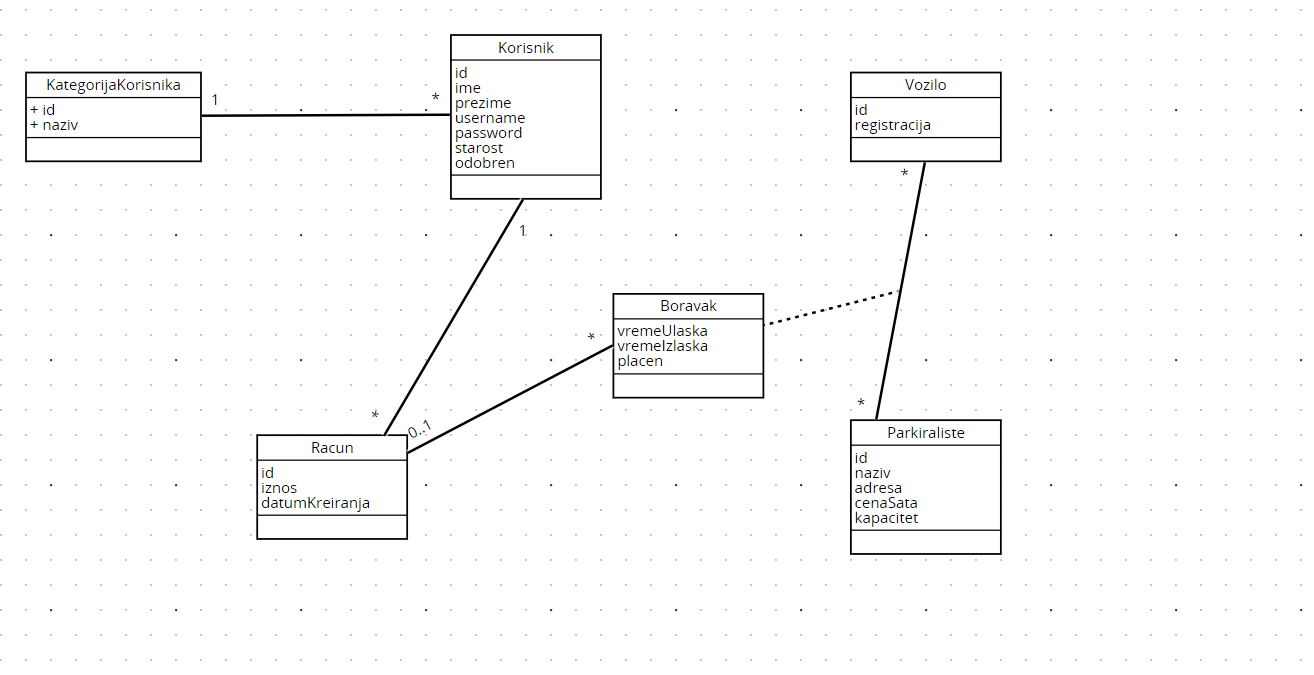


Slika 1 Pretraga boravaka



Slika 2 Kreiranje racuna

# Konceptualni model



Slika 3 Konceptualni model

# Specifikacija REST apija

|  |  |
| --- | --- |
| Opis funkcije | Vrati sve korisnik |
| HTTP metoda | GET |
| URL | /user |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | Niz JSON objekata. Svaki element niza ima atribute: category, id, firstName, lastName, age, username,odobren  Primer:  [  {  “category”:”admin”,  “id”:1,  firstName:”marko”,  “lastName”:”Milosevic”,  “age”:33,  “username”:”mareko87”,  “odobren”:true  }  ] |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prijavi se na sistem |
| HTTP metoda | POST |
| URL | /login |
| URL parametri | (nema) |
| HTTP body parametri | {  “username”:” mareko87”,  “password”:”mareko87”  } |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | JSON objekat kojiima atribute: category, id, firstName, lastName, age, username,odobren  Primer:  {  “category”:”admin”,  “id”:1,  firstName:”marko”,  “lastName”:”Milosevic”,  “age”:33,  “username”:”mareko87”,  “odobren”:true  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Vrati sva parkirališta |
| HTTP metoda | GET |
| URL | /parkiraliste |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | Niz JSON objekata.Svaki objekat ima atribute: adresa, id, kapacitet, cenaSata  Primer:  [  {  “id”:1,  “naziv”:”vracar”,  “adresa”:”Borislava Pekica 3”,  “cenaSata”:100,  “kapacitet”:500  }  ] |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Kreiraj parkiraliste |
| HTTP metoda | POST |
| URL | /parkiraliste |
| URL parametri | (nema) |
| HTTP body parametri | {  “naziv”:”vracar”,  “adresa”:”Borislava Pekica 3”,  “cenaSata”:100,  “kapacitet”:500  } |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | JSON objekat koji predstavlja novo parkiraliste.  Primer:  [  {  “id”:1,  “naziv”:”vracar”,  “adresa”:”Borislava Pekica 3”,  “cenaSata”:100,  “kapacitet”:500  }  ] |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Izmeni parkiraliste |
| HTTP metoda | PATCH |
| URL | /parkiraliste/:id |
| URL parametri | Id parkiralista |
| HTTP body parametri | {  “naziv”:”vracar”,  “adresa”:”Borislava Pekica 3”,  “cenaSata”:100,  } |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | JSON objekat koji predstavlja izmenjeno parkiraliste.  Primer:  [  {  “id”:1,  “naziv”:”vracar”,  “adresa”:”Borislava Pekica 3”,  “cenaSata”:100,  “kapacitet”:500  }  ] |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Obrisi parkiraliste |
| HTTP metoda | DELETE |
| URL | /parkiraliste/:id |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | Putanja ne vraca nikakve podatke |
| Format izlaznih parametara | application/json |

# Opis tehnologija korišćenih u izradi aplikacije

Za prilikom izrade aplikacije korišćene su sledeće tehnologije:

* React – kreiranje poslovne logike na strani klijenta
* Express – kreiranje servera
* Axios – biblioteka za pravljenje AJAX zahteva
* Typeorm – objektno – relacioni maper
* Mysql – baza podatakaReact

## React

React predstavlja biblioteka za kreiranje korisničkog interfejsa i poslovne logike na klijentskoj strani. React služi za kreiranje SPA web aplikacija odnosno aplikacija koje imaju samo jednu html stranu i kojima glavnu ulogu ima javascript.

Glavni koncept u React – je komponenta.

Komponenta predstavlja nešto što može da se iscrta kao jedna celina.

Celokupni prikaz aplikacije dobija se kompozicijom raznih elemenata u okviru jedne glavne komponente koja se, najčešće, naziva App.

Komponente mogu definisane kao funkcija i kao klasa.

Svaka komponenta ima svoje atribute koje je dobila komponente roditelja i svoje stanje koje je ona sama kreirala. Komponenta može da menja svoje stanje ali ne može da menja svoje atribute. Prilikom promene stanja poziva se posebna funkcija koja to radi nakon čega se prikaz na ekranu automatski menja na svim mestima koja su se iscrtavala na osnovu tog stanja.

## Express

Predstavlja „framework“ za izradu serverske strane u Node.js okruženju. Za razliku od većine ostalih servera, Express ne podržava višenitno programiranje. Problem konkurentnosti on rešava pomoću asinhronog programiranja i „event loop“ – a.

Express aplikacije su zamišljene da rade kao REST servisi, a obrada zahteva klijenta vrši se na sledeći način:

* Pristiže zahtev klijenta
* Izvršavaju se funkcije koje su definisane da se izvrše nad svakim zahtevom („middleware“)
* Zahtev se rutira na osnovu svog URL – a
* Izvršava se funkcija pridodata za dati URL
* Šalje se odgovor korisniku

## Axois

Axios je biblioteka koja pojednostavljuje rad sa AJAX zahtevima. Axios omogućava kreiranje svih vrsta http zahteva kao i kreiranje zaglavlja. Kao odgovor sa servera, axios vraća korisniku originalni odgovor sa pridruženim metapodacima koje je axios izgenerisao.

## Mysql

Relaciona baza podataka koja je besplatna za korišćenje.

## TypeORM

Objektno – relacion maper u javascript – u. Pravila prevođenja vrše se anotacijom klasa u programskom kodu. Sva konfiguracija nalazi se fajlu ormconfig.json. TypeORM omogućava rad sa bazom na 2 načina:

* Preko repozitorijuma (Repository pattern)
* Preko menadžera

U prvom slučaju olakšan je rad sa bazom ali su mogućnost kontrole toka i prilagođavanje rada ograničeni dok je u drugom omogućena mnogo veća kontrola i kastomizacija na račun komplikovanijeg koda.

# Link ka udaljenom repozitorijumu

<https://github.com/mareko87/parking>