

Programsko inženjerstvo

Ak. god. 2021./2022.

PassDirect

Dokumentacija, Rev. 2

Grupa: *CozySolutions*

Voditelj: *Tomislav Pavković*

Datum predaje: *14.01.2021.*

Nastavnik: *Daria Primorac*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	6
3 Specifikacija programske potpore	13
3.1 Funkcionalni zahtjevi	13
3.1.1 Obrasci uporabe	15
3.1.2 Sekvencijski dijagrami	20
3.2 Ostali zahtjevi	27
4 Arhitektura i dizajn sustava	29
4.1 Baza podataka	31
4.1.1 Opis tablica	31
4.1.2 Dijagram baze podataka	35
4.2 Dijagram razreda	36
4.3 Dijagram stanja	39
4.4 Dijagram aktivnosti	40
4.5 Dijagram komponenti	43
5 Implementacija i korisničko sučelje	44
5.1 Korištene tehnologije i alati	44
5.2 Ispitivanje programskog rješenja	47
5.2.1 Ispitivanje komponenti	47
5.2.2 Ispitivanje sustava	51
5.3 Dijagram razmještaja	56
5.4 Upute za puštanje u pogon	57
6 Zaključak i budući rad	60
Popis literature	61
Indeks slika i dijagrama	63

Dodatak: Prikaz aktivnosti grupe

64

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Mirta Vučinić	25.10.2021.
0.2	Funkcionalni zahtjevi,UC.	Ivan Hajpek	28.10.2021.
0.3	UC dijagram.	Martina Galić	28.10.2021.
0.4	DB dijagram i sekvencijski dijagrami.	Martina Galić	03.11.2021.
0.5	DB dijagram.	Ivan Hajpek	03.11.2021.
0.6	Dorađeni UC-ovi i sekvencijski dijagrami.	Ivan Hajpek	04.11.2021.
0.7	Dorađeni UC-ovi i dijagram baze podataka.	Martina Galić	04.11.2021.
0.8	Napisan opis za entitete.	Martina Galić	07.11.2021.
0.9	Napisan kratak opis baze i pojedinih entiteta.	Ivan Hajpek	08.11.2021.
0.10	Opis arhitekture sustava.	Martina Galić	10.11.2021.
0.11	Opis dizajna sustava i ostali zahtjevi.	Ivan Hajpek	10.11.2021.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
0.12	Dorađeni sekv. dijagrami i pregled dokumentacije.	Ivan Hajpek	15.11.2021.
0.13	Dorađeni sekv. dijagrami i opis projektnog zadatka.	Martina Galić	15.11.2021.
0.14	Izmjena BP i opis Controller Diagrama.	Ivan Hajpek	16.11.2021.
0.15	Class Diagrams.	Martina Galić	16.11.2021.
0.16	Dodatak, literatura, opis sastanaka.	Ivan Hajpek	19.11.2021.
0.17	Tablica aktivnosti i izmjena pojedinih dijagrama.	Martina Galić	19.11.2021.
1.1	Dodan dijagram atktivnosti.	Ivan Hajpek	19.12.2021.
1.2	Popravljene greške s prve predaje.	Krešimir Pavlović	19.12.2021.
1.3	Dodan dijagram stanja.	Krešimir Pavlović	20.12.2021.
1.4	Dodan drugi dijagram atktivnosti.	Ivan Hajpek	20.12.2021.
1.5	Dodan opis dijagrama aktivnosti, dopunjeni ostali zahtjevi, sitne promjene ostatka.	Ivan Hajpek	20.12.2021.
1.6	Dodan opis dijagrama stanja.	Krešimir Pavlović	4.1.2022.
1.7	Dodan dijagram komponenti.	Krešimir Pavlović	5.1.2022.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
1.8	Dodan dijagram razmještaja, dopunjen dnevnik sastajanja, dodan opis korištenih tehnologija, dodan zaključak.	Ivan Hajpek	6.1.2022.
1.9	Dodan opis puštanja aplikacije u pogon.	Krešimir Pavlović	10.1.2022.
1.10	Napravljen specifikacijski dijagram klasa.	Ivan Hajpek	10.1.2022.
1.11	Dodano ispitivanje sustava.	Krešimir Pavlović	12.1.2022.
1.12	Opis dijagrama klasa, dodan UC, promjena baze i dijagrama.	Ivan Hajpek	12.1.2022.
1.13	Unit testing dodan.	Krešimir Pavlović, Ivan Hajpek	12.1.2022.

2. Opis projektnog zadatka

Velik problem sadašnjice, a i prošlosti u području željezničke strukture napokon je dobio rješenje. U prošlosti je zasigurno bilo teže riješiti problem nejednolike raspodjele mase unutar vagona vlaka zbog nedostatka naprednije tehnologije, što je uzrokovalo povećanje istrošenosti kotača, slabljenje strukture osovine te postojanje veće vjerojatnosti iskliznuća vlaka iz tračnica. Mehaničari su stoga periodično morali održavati sustav kotača radi njegove rekonfiguracije i optimizacije što nekada možda i nije bilo nužno, no mehaničari nisu bili sigurni te su svakako napravili potrebno održavanje, što u nekim zemljama moraju i danas, no ne i u Nizozemskoj, jednoj od najnaprednijih država po pitanju željezničkog prometa. Utvrđivanje dijela vagona na kojem je potrebno provesti održavanje je dugotrajan i kompleksan posao. Kako bi smanjili broj nesreća uzrokovanih nastalim problemima zbog nejednolike raspodjele mase te sačuvali infrastrukturu, odnosno produžili joj životni vijek, Nizozemci su osmislili sustav senzora Gotcha.

Gotcha se sastoji od senzora postavljenih duž tračnica kolosijeka na odabranim točkama. Pri prolasku vlaka preko senzora, senzori mjere opterećenje kotača u stvarnom vremenu. Iz vremenskog niza podataka se tada može procijeniti akumulativna šteta na svakom kotaču i odrediti optimalan trenutak održavanja svakog vagona.

Uz to što će senzor zabilježiti podatke koji pomažu pri određivanju optimalnog trenutka održavanja vagona, cilj je i doći do manjeg broja popravaka odnosno održavanja. To se postiže na način da se ravnomjerno rasporedi masa u vagonu, odnosno da putnici sjede na odgovarajućim mjestima. Dakle, postiže se pomoć u prevenciji oštećenja nastalim nejednolikom raspodjelom mase, smanjuje se trošak održavanja i povećava vrijeme dostupnosti vagona. Ideja aplikacije PassDirect je upućivanje putnika u odgovarajuće vagone (i perone na kojima ih mogu čekati) u svrhu optimalne distribucije njihove težine, uporabom podataka sustava senzora Gotcha. Vlakovi nad kojima se provode mjerenja namijenjeni su brzom povezivanju prigradskih naselja te nemaju omogućenu rezervaciju sjedala.

Sustav senzora:

Svako mjerenje senzora daje podatak koji se sastoji od: položaja senzora na pruži, identifikacijske oznake vlaka, brzine vlaka, vrijeme mjerenja te niza prirodnih brojeva, po dva za svaki vagon. Niz prirodnih brojeva označava višak detektirane mase u kilogramima u svakom vagonu i to jedan podatak za prednji dio vagona, a drugi za stražnji dio. Na primjer: 80, 60, 110, 220, 756 i 923. Prvi vagon (80,60) je uravnotežen, drugi vagon (110, 220) također, dok treći vagon (756, 923) ima najviše putnika no i najveću razliku između prednjeg i stražnjeg dijela. Sva mjerenja na pojedinačnom vlaku obavljaju su istovremeno. Zanimljivo je da razliku između lijeve i desne strane vagona. PassDirect je implementiran kao web aplikacija te podržava rad više korisnika u stvarnom vremenu.

Simulacija senzora ostvarena je korištenjem sustava Posthook koji šalje http zahtjeve na backend naše aplikacije s očitanjima senzora koja sadrže broj osoba na svakoj poziciji (naprijed i nazad) za svaki vagon tog vlaka.

Mjerenja su napisana za 3 vlaka na 3 stanice i različita su za 3 uzastopna dana te se nakon toga ponavljaju. Vlak polazi sa stanice u Zagrebu pa prvo očitavanje primamo nakon napuštanja te stanice što znači da korisnici koji kupuju kartu iz Zagreba ne dobivaju upute gdje trebaju sjesti nego imaju slobodan izbor jer je vlak prazan. Korisnici koji polaze sa sljedeće stanice (Split), nakon što vlak napusti stanicu u Zagrebu, dobivaju preporuku o broju vagona i perona te poziciji u vagonu (naprijed ili nazad) gdje trebaju sjesti.

Broj vagona i pozicija računaju se na temelju očitavanja s prethodne stanice na sljedeći način:

- Ako je razlika u broju putnika naprijed i nazad u nekom vagonu veća od 30, putnik se šalje u taj vagon (s najvećom razlikom ako ih je više takvih) na poziciju koja je manje popunjena.
- Ako ni u jednom vagonu razlika broja ljudi nije veća od 30, putnik se šalje u vagon s najmanjim brojem ljudi, na manje popunjenu poziciju.
- Ako je putnik usmjeren, podaci o broju putnika u tom vagonu se ažuriraju kako ne bi svi putnici bili usmjereni na istu poziciju. Nakon kupnje karte, putnik prima email s kartom, a nakon što vlak napusti prethodnu stanicu, putnik prima email s ažuriranom kartom s podacima o broju vagona, perona

i poziciji na koju treba sjesti.

- Ako korisnik kupi kartu za vlak koji je već napustio prethodnu stanicu, odmah prima email s kartom koja sadrži podatke o broju vagona, perona i poziciji na koju treba sjesti kako bi izbjegli nepotrebno slanje više emailova.

Ako vlak kasni na određenu stanicu (senzor) više od dvije minute, na početnoj stranici aplikacije, kod pregleda voznog reda na toj i svim sljedećim stanicama, vrijeme dolaska postaje crveno kako bi se naglasilo da vlak kasni i vjerojatno neće stići u to vrijeme. Čak i kada vlak dođe na tu stanicu (senzor), ako je vrijeme kada je trebao doći prošlo za više od 2 minute, na sljedećim stanicama ostaje oznaka kašnjenja. Ako vlak to kašnjenje nadoknadi te na sljedeću stanicu dođe na vrijeme, oznaka kašnjenja se miče za sljedeće stanice.

Korisnik može imati ulogu:

- Administratora
- Konkretnog korisnika

Administrator može pregledati i ukloniti korisnike te pregledati njihove transakcije, no nije zadužen za mijenjanje voznog reda. Također ima pristup svim provedenim transakcijama, bazi podataka korisnika i redu vožnje. PassDirect upravlja rasporedom putnika dok je za vozni red zadužen drugi ogranak željeznice.

Konkretnan korisnik može pristupiti funkcionalnostima web stranice PassDirect registracijom, odnosno kreiranjem svog korisničkog računa te prijavom na isti. Prilikom registracije, korisnik unosi svoje podatke, i to:

- ime
- prezime
- e-mail adresu
- lozinku

Nakon prijave i registracije:

Nakon uspješne registracije i prijave, korisniku se na pregledniku prikazuju ponuđena stajališta, od kojih on može odabrati jedno. Odabir ga vodi na listu dolazećih vlakova na to stajalište (u stvarnom vremenu). Informacije koje korisnik treba dobiti odabirom stajališta su:

- identifikacijska oznaka vlaka
- krajnje odredište
- vrijeme dolaska

Zadnje tri informacije bi trebale biti istaknute i dane samo za one vlakove koji su napustili prethodno stajalište. Do zaključka je li vlak napustio prethodno stajalište dolazimo uporabom senzora koji se nalazi na prošlom stajalištu. Ako je mjerenje detektirano, vlak je napustio prethodno stajalište. Vlak se zadržava na svakom stajalištu 10 minuta. Vrijeme u kojem senzor detektira vlak, koristimo za određivanje kašnjenja vlaka na sljedeće stajalište, točnije određujemo hoće li kasniti i koliko će kasniti.

Klasično pretraživanje voznog reda:

Klasično pretraživanje uključuje:

- odabir mjesta polaska
- odabir mjesta dolaska
- datum

Na zaslonu se potom izlistavaju odgovarajuće linije vlakova zajedno s ostalim informacijama (npr. identifikacijska oznaka vlaka, mjesto i vrijeme polaska, mjesto i vrijeme dolaska, kolosijek dolaska, kašnjenje vlaka i cijena karte).

Kupovina karte:

Skup korisnika zainteresiranih za konzumaciju prijevozne usluge, prilikom prikazivanja linija, mogu kupiti kartu za svaku od njih. PassDirect sadrži upitnik koji oponaša plaćanje. Pri prvoj transakciji, korisnik popunjava formular, no pri svakoj narednoj, formular se popunjava sam. Nakon kupovine, korisniku stižu dva

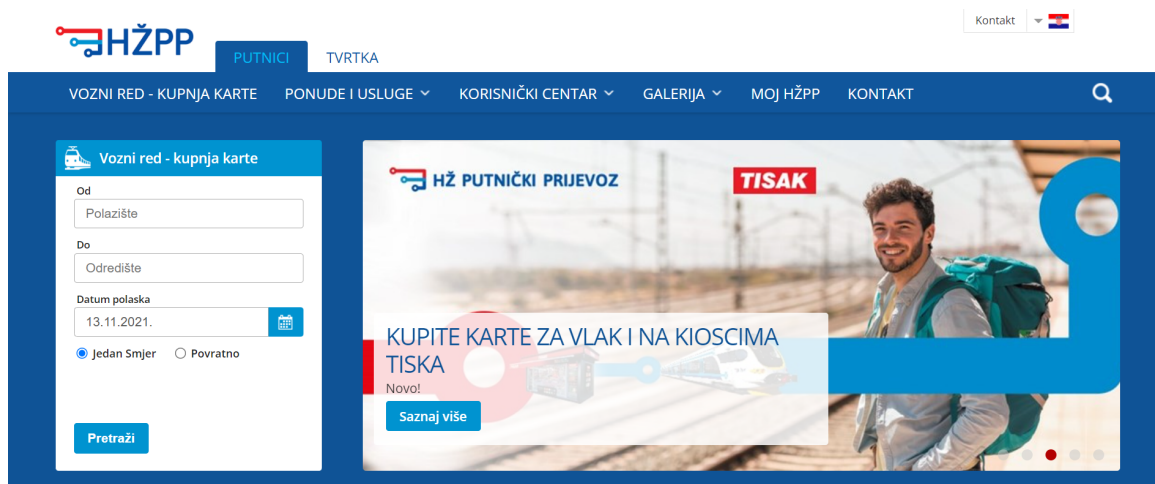
e-maila. U jednom e-mailu se nalazi kupljena karta, a u drugom obavijest koja ga upućuje na određeni kolosijek i peron koji je procijenjen kao najpogodniji. Drugi e-mail treba stići tek nakon što vlak napusti prethodno stajalište. Drugi e-mail treba sadržavati:

- informativni tekst
- šifru vlaka
- vrijeme njegovog dolaska
- kolosijek
- uputu za korisnika o prikladnom vagonu
- dijelu vagona
- broju perona

Naravno, cilj aplikacije jest uputiti putnika u prazniji dio najmanje okupiranog vagona.

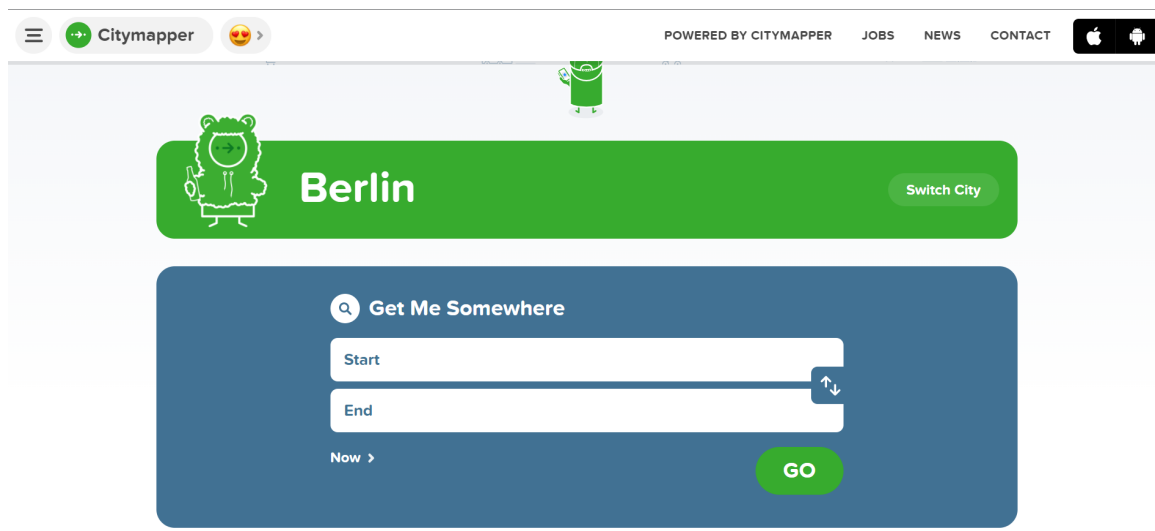
Slična programska rješenja:

Postoje mnoga slična programska rješenja koja omogućavaju korisniku kupovinu karte, odabir početnog i krajnjeg stajališta koji rezultira izlistavanjem svih linija. Jedna od njih je naravno HŽPP-ova stranica. Navedena stranica ima implementirane sve funkcionalnosti kao i PassDirect poput kupovine karte, pregleda svih linija, pregleda svih linija s obzirom na početno i krajnje stajalište, datum, vrijeme i slično, no s obzirom da u Hrvatskoj još nije u primjenu stupio Gotcha sustav senzora, na ovako napredno rješenje kao u Nizozemskoj ćemo morati sačekati.



Slika 2.1: Web stranica HŽPP-a

Također je tu i web aplikacija CityMapper koja sugerira korisniku mjesto sjedenja, no ne s obzirom na Gotcha sustav, nego s obzirom na korisnikove potrebe. Konkretnije, predlaže najbolje mjesto s obzirom na korisnikovo krajnje stajalište ili presjedanje. CityMapper posjeduje sve navedene funkcionalnosti kao i HŽPP web stranica. No, jako je malo web aplikacija koje sugeriraju korisniku koje mjesto bi bilo poželjno koristiti.



Slika 2.2: Web stranica cityMapper

Moguća nadogradnja:

- Web aplikaciju PassDirect bi se moglo unaprijediti dodavanjem novih funkcionalnosti poput zaključivanja o potrebi obnove infrastrukture na osnovu mjerenja senzora. Predviđanjem obnove, mogli bismo pravovremeno ukloniti određene vozne redove kako korisnik ne bi dobivao krivu informaciju pri ranoj kupovini karte ili ako dođe do otkazivanja linije, aplikacija bi trebala poslati e-mail korisniku. Uz to bi bila poželjna i funkcionalnost koja bi pri određivanju sjedala za korisnika, uz podatke mjerenja, uzimala u obzir i krajnje stajalište korisnika ili presjedanje i na taj način poticala korisnika još više da koristi upravo to određeno sjedalo, jer u tom slučaju, korist bi imao i korisnik i željeznički prijevoz.
- Aplikaciju je moguće proširiti i na mobilnu platformu čime bi njena funkcionalnost bila fleksibilnija. Korisnik bi umjesto korištenja e-maila imao opciju čuvanja karte te dohvaćanja informacija o narednoj vožnji nakon kupovine na samoj mobilnoj aplikaciji. Također, bila bi olakšana komunikacija s korisnikom u slučaju promjena u redu vožnje (npr. promjena perona ili otkazivanja vožnje) putem notifikacija.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Klijent
2. Administrator
3. Razvojni tim
4. Željezničko poduzeće

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani korisnik (inicijator) može:
 - (a) se registrirati kako bi se mogao prijaviti u aplikaciju
2. Klijent (inicijator) može:
 - (a) se prijaviti u stranicu kako bi mogao koristiti aplikaciju
 - (b) odabrati jedno od ponuđenih stajališta vlaka i dobiti prikaz informacija za svaku ponuđenu vožnju (identifikacijsku oznaku vlaka, krajnje odredište i vrijeme dolaska)
 - (c) klasično pretražiti red vožnje u bilo kojem trenutku(odabir mjesta polaska, odabir mjesta dolaska i datum)
 - (d) kupiti kartu za svaku prikazanu liniju na aplikaciji
 - (e) obrisati profil/račun
3. Administrator (inicijator) može:
 - (a) pristupati podacima o klijentima te pregledati njihove transakcije
 - (b) pristupati redu vožnje
 - (c) brisati klijente
4. Aplikacija (sudionik) mora:
 - (a) poslati mail s kupljenom kartom klijentu nakon što je klijent plati

(b) poslati drugi mail kojim ga upućuje na određeni kolosijek i peron

5. Baza podataka (sudionik):

(a) pohranjuje podatke o korisnicima i svim njihovim provedenim transakcijama(kupljenim kartama)

(b) pohranjuje podatke o voznom redu, poput polaznog i krajnjeg stajališta, cijene karte, datuma linije i slično

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Registracija

- **Glavni sudionik:** Korisnik
- **Cilj:** Stvaranje korisničkog računa za prijavu u sustav
- **Sudionici:** Baza podataka
- **Preduvjet:** –
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju registracije
 2. Korisnik upisuje potrebne vlastite podatke
 3. Korisnik potvrđuje registraciju
 4. Sustav validira podatke i sprema ih u bazu podataka u slučaju ispravnosti
- **Opis mogućih odstupanja:**
 - 2.a Odabir već postojećeg korisničkog imena, odnosno e-maila, upis korisničkog podatka u nedozvoljenom formatu ili pružanje neispravnog formata e-maila
 1. Sustav obavijesti korisnika o neuspješnom upisu podatka i vraća ga na stranicu za registraciju
 2. Korisnik mijenja pogrešne podatke ili odustaje od registracije

UC2 - Prijava

- **Glavni sudionik:** Klijent, administrator
- **Cilj:** Prijava za pristup aplikaciji za kupnju karata
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
 1. Korisnik unosi korisničko ime i lozinku
 2. Korisnik potvrđuje navedene podatke
 3. Korisnik dobiva pristup aplikaciji
- **Opis mogućih odstupanja:**
 - 2.a Neispravno korisničko ime ili lozinka
 1. Sustav obavijesti korisnika o neuspješnoj prijavi i vrati ga na stranicu za prijavu

UC3 - Pregled vlakova za stajalište

- **Glavni sudionik:** Klijent
- **Cilj:** Pregledati sve dolazeće vlakove na stajalište koje je klijent odabrao
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Korisnik odabire stajalište koje mu treba
 2. Korisnik potvrđuje odabir
 3. Aplikacija prikazuje popis dolazećih vlakova na odabrano stajalište
- **Opis mogućih odstupanja:**
 - 2.a Korisnik nije odabrao stajalište
 1. Sustav obavijesti korisnika o neuspješnom zahtjevu i zatraži odabir stajališta

UC4 - Pretraživanje voznog reda

- **Glavni sudionik:** Klijent
- **Cilj:** Pretražiti vozni red
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava
- **Opis osnovnog tijeka:**
 1. Korisnik odabire mjesto polaska i dolaska
 2. Korisnik odabire željeni datum
 3. Korisnik potvrđuje odabir
 4. Aplikacija prikazuje vozni red za odabrane podatke
- **Opis mogućih odstupanja:**
 - 3.a Korisnik nije odabrao mjesto polaska, dolaska ili datum
 1. Sustav obavijesti korisnika o neuspješnom zahtjevu i zatraži ponovni odabir i unos

UC5 - Kupnja karte

- **Glavni sudionik:** Klijent
- **Cilj:** Kupiti kartu za vlak
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijava i pretraga
- **Opis osnovnog tijeka:**

1. Korisnik odabire vlak za koji želi kupiti kartu
 2. Korisnik upisuje potrebne osobne podatke za kupnju(ime, prezime, podaci o kartici itd.) ili se automatski popunjavaju ako su već upamćeni
 3. Korisnik potvrđuje kupnju
 4. Stizanje 2 e-mailova
- **Opis mogućih odstupanja:**
 - 2.a Korisnik nije dobro upisao neki podatak
 1. Sustav obavijesti korisnika o neuspješnoj kupnji zbog neispravno unesenih podataka
 2. Korisnik ima mogućnost ponovno unijeti podatke ili odustati od kupnje

UC6 - Pregled klijenata

- **Glavni sudionik:** Administrator
- **Cilj:** Pristupiti i pregledati podatke klijenata
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik mora biti prijavljen i registriran s dodijeljenim pravom administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabere opciju pregledavanja klijenata
 2. Na aplikaciji se izlistaju podatci svakog korisnika aplikacije

UC7 - Pregled transakcija

- **Glavni sudionik:** Administrator
- **Cilj:** Pristupiti i pregledati podatke o provedenim transakcijama
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik mora biti prijavljen i registriran s dodijeljenim pravom administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabere opciju pregledavanja transakcija
 2. Na aplikaciji se izlistaju podatci svake provedene transakcije kroz aplikaciju

UC8 - Pregled reda vožnje

- **Glavni sudionik:** Administrator
- **Cilj:** Pristupiti i pregledati vozni red

- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik mora biti prijavljen i registriran s dodijeljenim pravom administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabere opciju pregledavanja voznog reda
 2. Na aplikaciji se izlista vozni red

UC9 - Brisanje klijenta

- **Glavni sudionik:** Administrator
- **Cilj:** Obrisati odabrane korisnike
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik mora biti prijavljen i registriran s dodijeljenim pravom administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabere klijenta na listi klijenata
 2. Administrator obriše odabranog klijenta opcijom brisanja te potvrdom odluke
 3. Aplikacija ga vraća na osvježenu listu klijenata

UC10 - Odjava

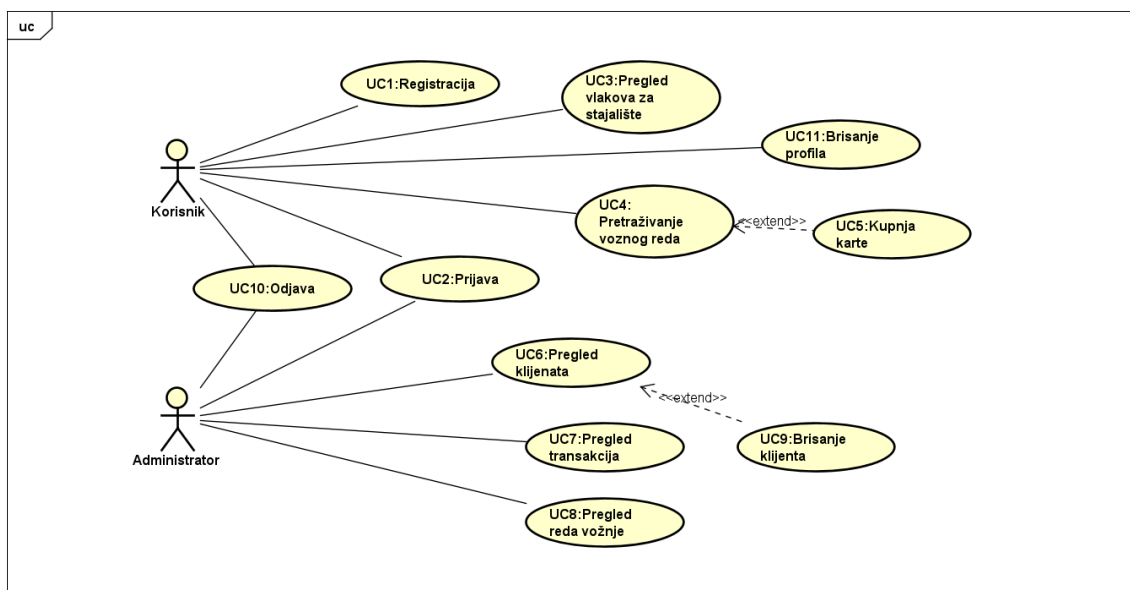
- **Glavni sudionik:** Klijent
- **Cilj:** Odjaviti se
- **Sudionici:**
- **Preduvjet:** Korisnik mora biti registriran i prijavljen
- **Opis osnovnog tijeka:**
 1. Klijent zatraži odjavu
 2. Klijentu je obrisana sesija
 3. Klijent je poslan na stranicu za prijavu

UC11 - Brisanje računa

- **Glavni sudionik:** Klijent
- **Cilj:** Obrisati račun na PassDirect stranici
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik mora biti prijavljen i registiran
- **Opis osnovnog tijeka:**
 1. Klijent odabire dugme s opcijom brisanja računa na PassDirect stranici

2. Klijentu se izbací obavijest s potvrdom brisanja računa, gdje bira između potvrde i odustajanja
3. Klijent bira opciju
4. Aplikacija ostane na istoj stranici ako odustane, a ukoliko potvrdi, vraća ga na Login stranicu

Dijagrami obrazaca uporabe

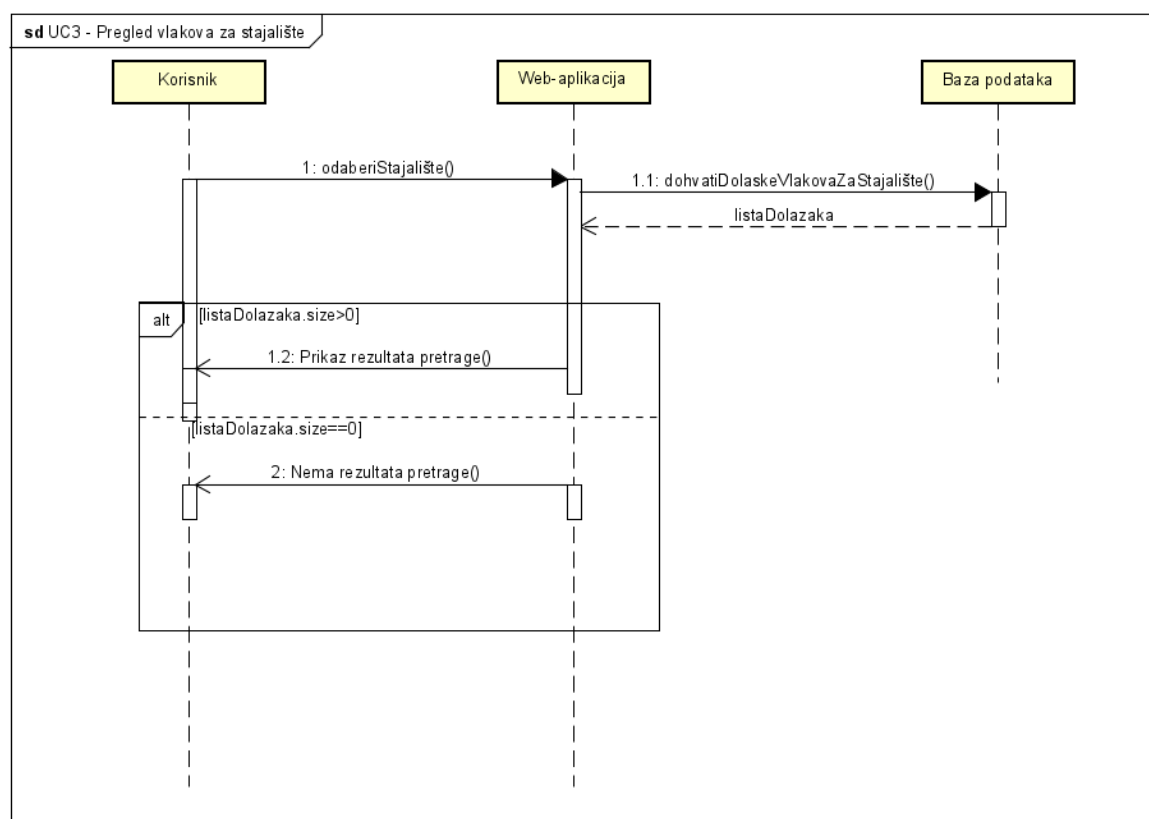


Slika 3.1: UC dijagram

3.1.2 Sekvencijski dijagrami

UC3 - Pregled vlakova za stajalište

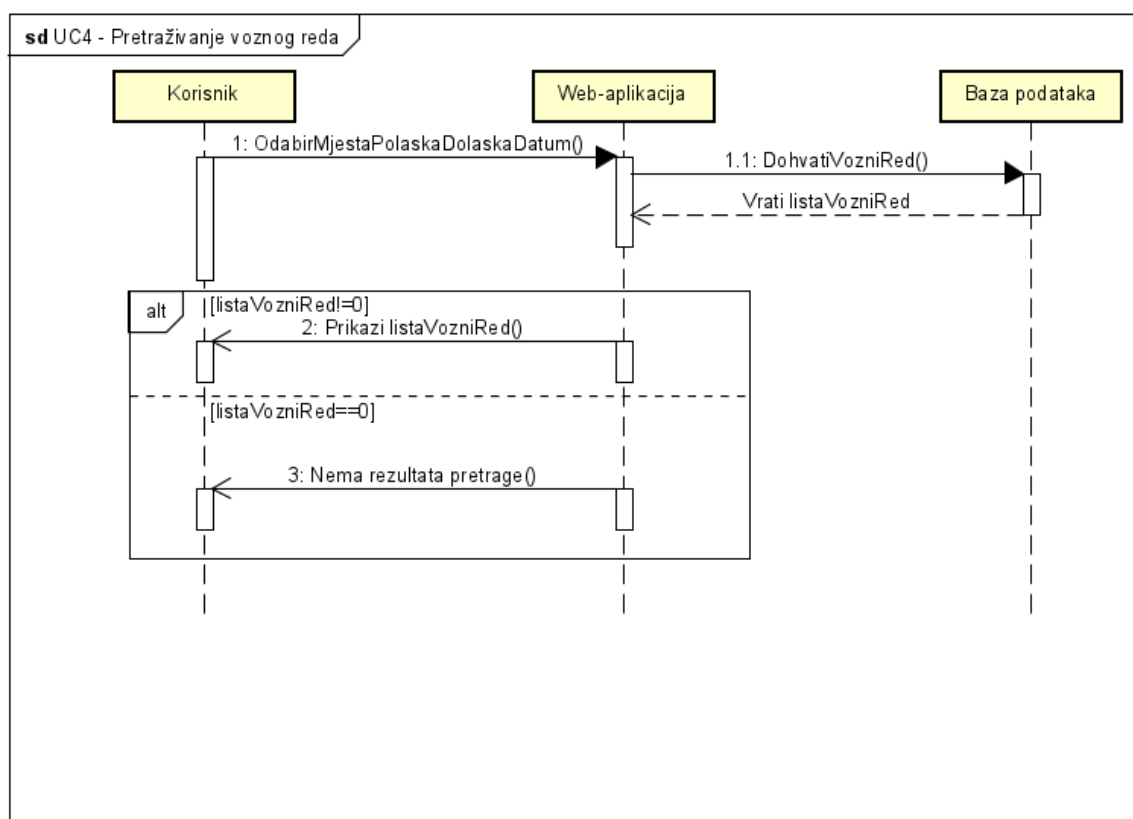
Klijent odabire i potvrđuje stajalište za koje želi dobiti nadolazeće vlakove. Poslužitelj dohvaća i prikazuje popis i informacije o vlakovima koji stižu na odabrano stajalište.



Slika 3.2: UC3 - Pregled vlakova za stajalište

UC4 - Pretraživanje voznog reda

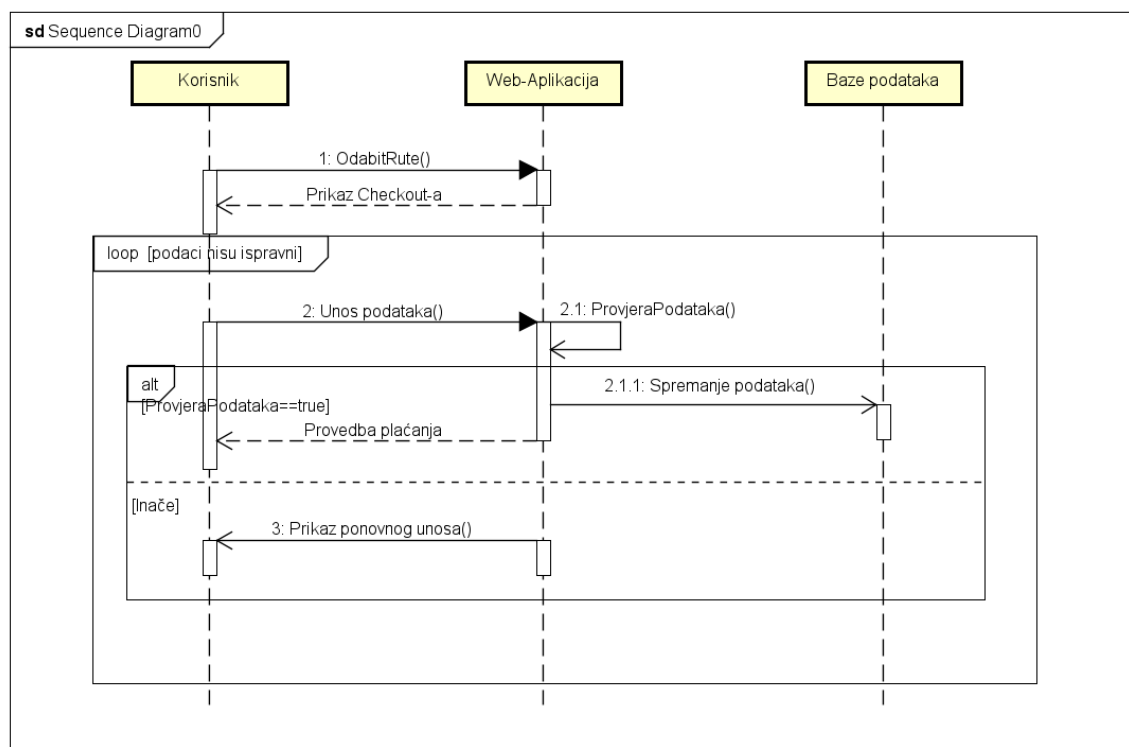
Klijent odabire mjesto polaska i dolaska te datum polaska, a poslužitelj mu vraća listu mogućih linija za unesene podatke.



Slika 3.3: UC4 - Pretraživanje voznog reda

UC5 - Kupnja karte

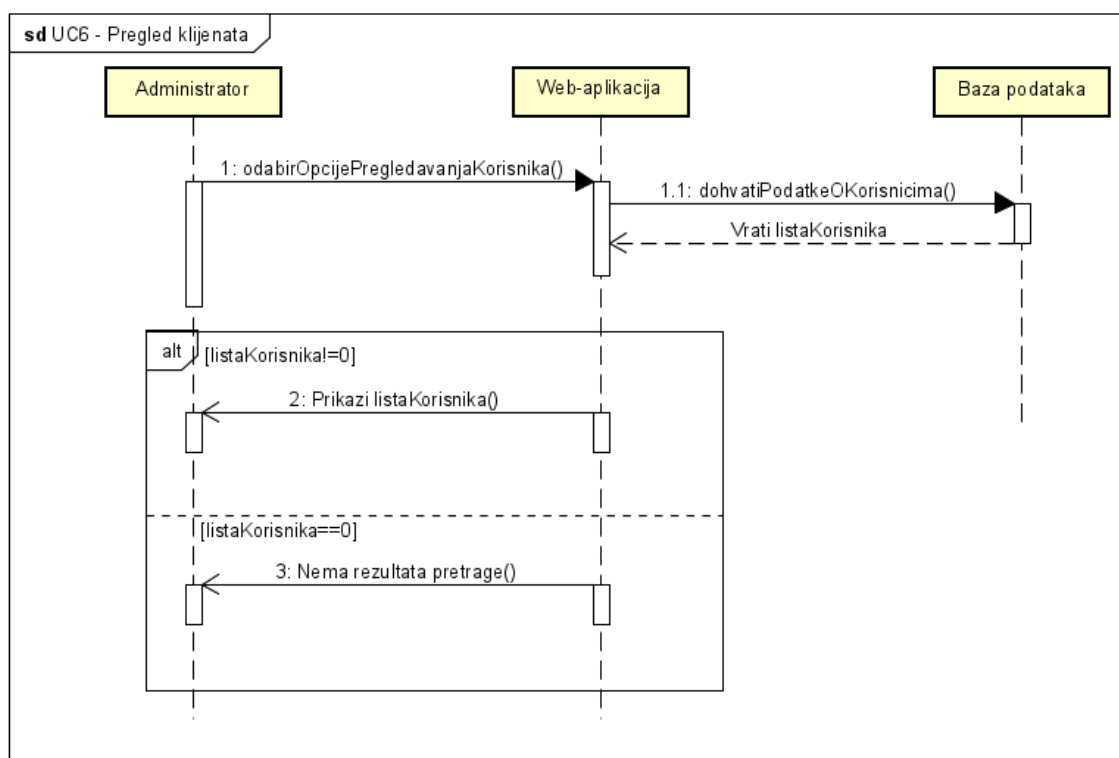
Klijent odabire vlak za koji želi kupiti kartu pa ga poslužitelj odvede na stranicu za checkout. Ovdje unosi sve potrebne podatke za provesti transakciju, nakon čega potvrdi kupnju. Ukoliko su neki podaci netočni ili neispunjeni, poslužitelj daje ponovno mogućnost checkout-a ili odustajanje od kupnje. Kada su podaci točni i ispunjeni, klijentu se javlja uspješna kupnja.



Slika 3.4: UC5 - Kupnja karte

UC6 - Pregled klijenata

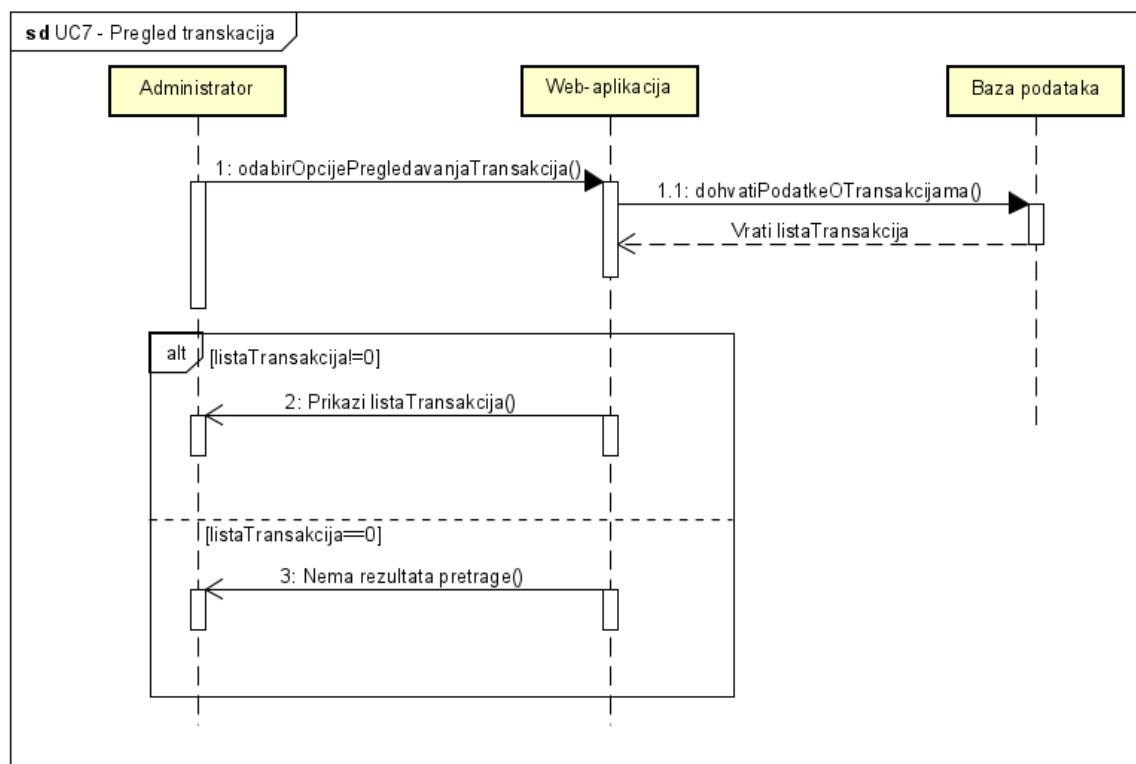
Administrator odabire opciju pregleda podataka o svim klijentima. Poslužitelj dohvaća i prikazuje popis i informacije o klijentima.



Slika 3.5: UC6 - Pregled klijenata

UC7 - Pregled transakcija

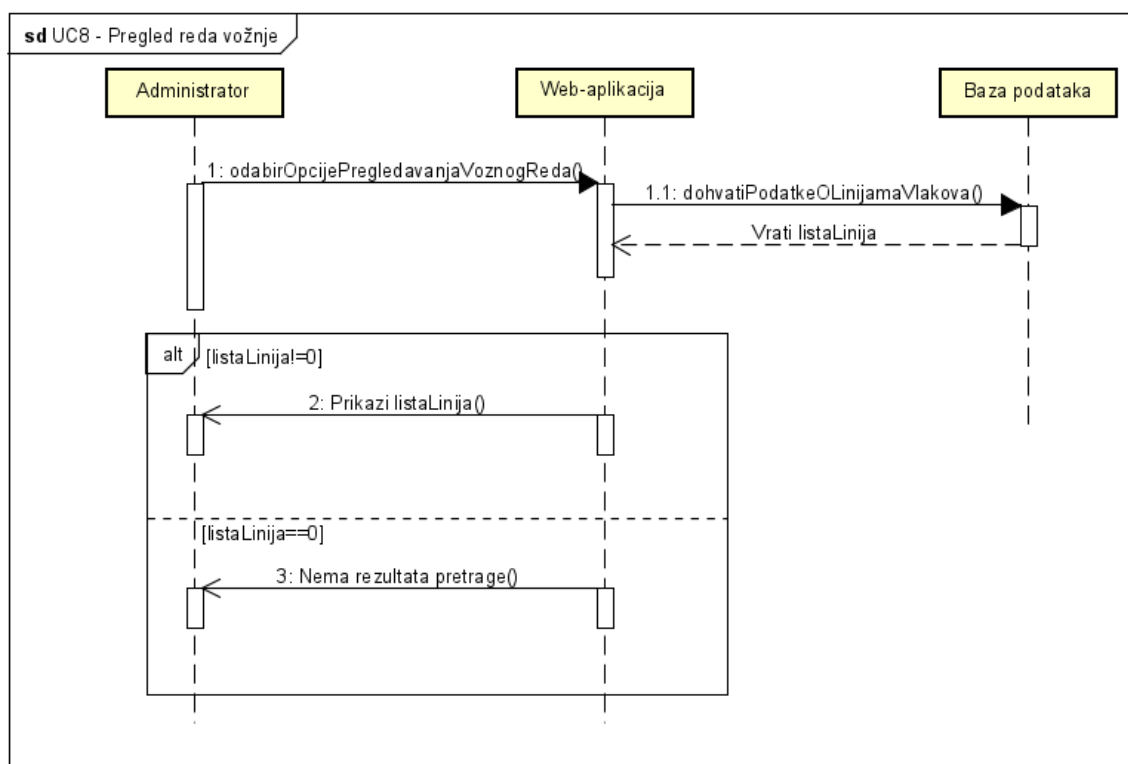
Administrator odabire opciju pregleda podataka o svim provedenim transakcijama. Poslužitelj dohvaća i prikazuje popis i informacije o svim provedenim kupnjama karata.



Slika 3.6: UC7 - Pregled transakcija

UC8 - Pregled reda vožnje

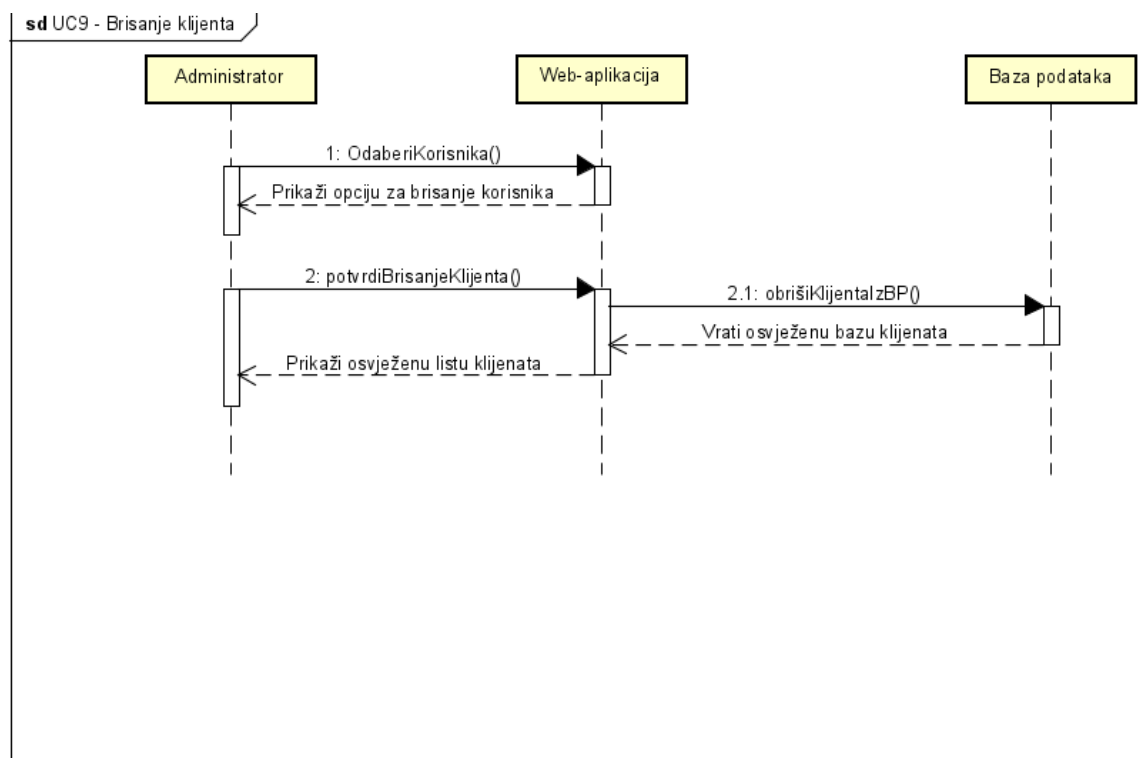
Administrator odabire opciju pregleda podataka o voznom redu, odnosno linijama vlakova. Poslužitelj dohvaća i prikazuje popis i informacije o svim voznim linijama vlakova.



Slika 3.7: UC8 - Pregled voznog reda

UC9 - Brisanje klijenta

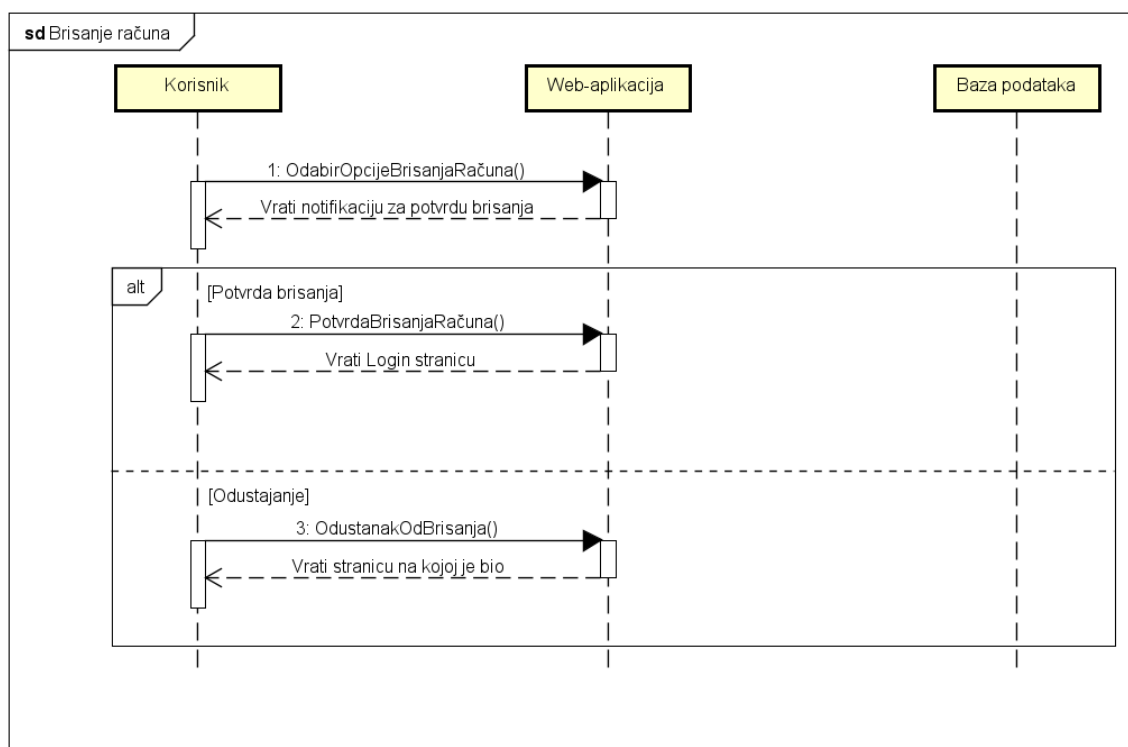
Administrator odabire klijenta te mu se izbacuje opcija za brisanje odabranog klijenta. Nakon odabira opcije, poslužitelj briše klijenta i vraća administratora na osvježenu listu klijenata.



Slika 3.8: UC9 - Brisanje klijenta

UC11 - Brisanje računa

Klijent odabire opciju brisanja računa u zaglavlju stranice. Web-aplikacija mu izbacuje notifikaciju s upitom o potvrdi brisanja računa. Ukoliko klijent potvrdi, aplikacija ga vrati na Login stranicu, a u slučaju da odustane od brisanja računa, ostane na stranici na kojoj je bio.



Slika 3.9: UC11 - Brisanje računa

3.2 Ostali zahtjevi

- Sustav treba omogućiti rad više korisnika u isto vrijeme.
- Sustav mora biti izveden kao web aplikacija s objektno orijentiranim jezikom.
- Komunikacija između korisnika i sustava mora biti brza i jednostavna.
- Aplikacija mora prikazivati aktualne, odnosno ažurirane podatke u svakom trenutku.
- Aplikacija mora raditi u stvarnom vremenu.
- Sustav kao valutu koristi HRK.
- Baza podataka treba biti brza, učinkovita i dobro povezana sa sustavom, otporna na bilo kakve greške korisnika i administratora.
- U slučaju pogrešaka ili zlonamjernih unosa od strane korisnika, sustav mora izbaciti odgovarajuće upozorenje
 - U slučaju lozinke kraće od 6 znakova, izbaciti se obavijest da je nužno barem 6 znakova.
 - Ukoliko neki od podataka koji se moraju ispuniti nisu ispunjeni, izbaciti

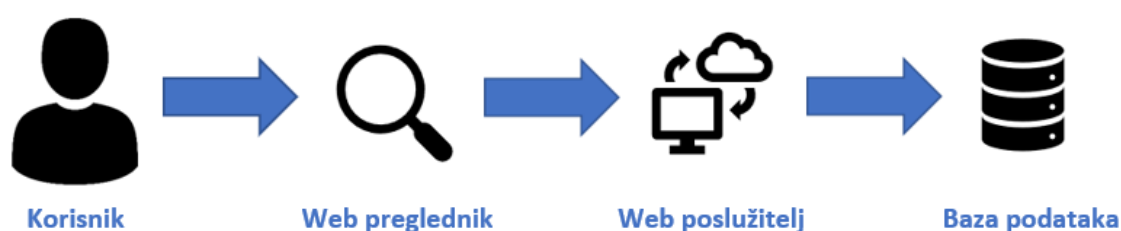
se upozorenje o tome.

- Prilikom prijave ili registracije netočnim podacima, također se ispisuje upozorenje o tome.
- Dodatna poboljšanja(senzor za pomoć očuvanju vagona) ne smiju ugroziti već postojeće osnovne funkcionalnosti aplikacije.
- Korisnik jednim mailom može otvoriti samo jedan račun na stranici.
- Aplikacija mora automatski popunjavati informacije u korisniku pri kupnji karata.

4. Arhitektura i dizajn sustava

Arhitektura programske potpore predstavlja strukturu sustava ili više njih koji sadrži elemente, njihova obilježja i odnose među njima. Koristimo objektno orijentiranu arhitekturu koja najbolje odgovara razvoju složene web aplikacije namijenjene za više korisnika u stvarnom vremenu. Možemo ju klasificirati na četiri ključna dijela koji osiguravaju izvršavanje naredbi korisnika:

- 1. Web preglednik
- 2. Web poslužitelj
- 3. Web aplikacija
- 4. Baza podataka

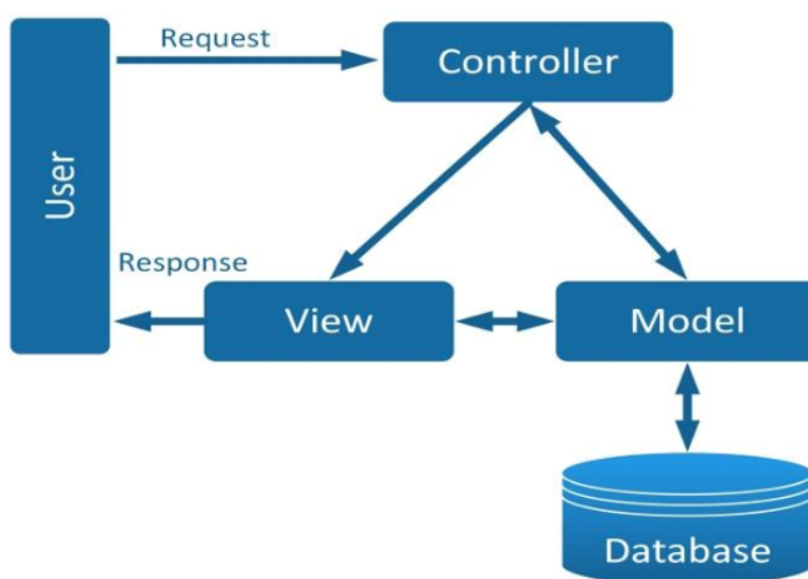


Slika 4.1: Arhitektura sustava

Prilikom pristupa web aplikaciji korisniku na jednostavan način postaje moguće pretražiti putovanja s odabranog polazišnog stajališta i kupiti kartu. Da bi korisnik mogao pristupiti navedenim funkcionalnostima treba pristupiti web pregledniku koji za njega vrši daljnju interakciju s ostalim podsustavima koji su sastavni dijelovi arhitekture sustava. Web preglednik dalje mora komunicirati s web poslužiteljem na kojem je sama implementacija web aplikacije. Preko web poslužitelja se obavljaju svi mogući funkcionalni zahtjevi aplikacije koji su prethodno definirani, a oni su izravno povezani s poslužiteljem baze podataka. U bazi podataka pohranjeni su svi podaci nužni za potpuno ispravan rad aplikacije, ali i podaci koje ručno unose korisnici poput osobnih podataka. Web poslužitelj šalje objekte na zahtjev klijenata, ti se zahtjevi realiziraju u obliku HTTP poruka. Web poslužitelju se pristupa preko nekog web preglednika.

Za izradu frontend dijela web aplikacije korišten je programski jezik Javascript s razvojnim okvirom React.js. Za izradu backend dijela korišten je Typescript s razvojnim okvirom Express.js. Baza podataka je implementirana pomoću PostgreSQL.

Za realizaciju arhitekture sustava korišten je koncept Model-View-Controller. Model-View-Controller je obrazac koji razdvaja aplikaciju u tri glavne logičke komponente: Model, View i Controller. Svaka od nabrojenih komponenti ima zadatak rukovati s određenim razvojnim aspektima aplikacije. Također, one su nezavisne jedna od druge i kao rezultat toga je jednostavno dodavanje i preoblikovanje svojstava.



Slika 4.2: Model-View-Controller

- **Model** Poznat je kao najniža razina što znači da je odgovoran za održavanje podataka s kojima korisnik radi. Glavni zadatak je dohvat, manipulacija podacima odnosno suradnja s bazom podataka. Sprema tražene podatke u objekte i na taj način ih šalje bazi podataka. Reagira na zahtjeve Controllera jer on nikada sam ne razgovara s bazom podataka i nakon komunikacije prosljeđuje potrebne podatke Controller-u. Jedna od bitnijih stvari za napomenuti je da Model nikada izravno ne komunicira s View.

- **View** Služi za prikazivanje podataka tako što generira korisničko sučelje za korisnika. Ti podaci su rezultat rada Model-a, ali se oni ne preuzimaju izravno nego putem Controller-a tako da View surađuje samo s Controller-om.

- **Controller** Upravlja komponentama Model i View. Ne mora brinuti o rukova-

nju logikom podataka, već samo govori Model-u što treba učiniti. Nakon primanja podataka od Model-a, on ih obrađuje i prosljeđuje rezultat do View-a.

4.1 Baza podataka

Za potrebe našeg sustava odabrali smo relacijsku bazu podataka koja nam olakšava modeliranje stvarnog svijeta. Baza je definirana skupom tablica, odnosno relacija koje su definirane svojim imenima i skupovima atributa (i ključevima). Zbog pouzdanosti, jednostavnosti i zadovoljavajućih preformansi, odabrali smo rad s PostgreSQL bazom. Baza podataka omogućava nam brz i lak dohvat i spremanje podataka, odnosno obradu i izmjenu podataka korištenih u aplikaciji. Naša baza sastoji se od sljedećih entiteta:

- User
- Ticket
- Journey
- TrainRoute
- Station
- SensorData

4.1.1 Opis tablica

User - Entitet User sadrži sve potrebne informacije o korisniku aplikacije koji može biti kupac ili administrator što određuje atribut UserType. User ima također i attribute firstName, lastName, email, password te createdAt što označava datum registracije korisnika. Email je jedinstven atribut, tako da se svaki User razlikuje po svom e-mailu. Ključ entiteta je email.

Korisnik		
email	VARCHAR	jedinstveni e-mail korisnika
firstName	VARCHAR	ime korisnika
lastName	VARCHAR	prezime korisnika
password	VARCHAR	lozinka registriranog korisnika

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Korisnik		
role	VARCHAR	vrsta korisnika,može biti kupac ili administrator
createdDate	DATE	datum registracije korisnika

Ticket - Entitet Ticket je slabi entitet koji ovisi o entitetu User iz razloga što karta ne može biti izdana bez da korisnik zatraži njezino izdavanje. Ključ entiteta Ticket je Id ticket-a. Atribut passengerEmail je strani ključ koji referencira entitet User te se radi o *Many-to-Many* vezi. Atribut idJourney rute je strani ključ koji referencira entitet Journey te se radi o *Many-To-One* vezi.

Ticket		
Id	INT	jedinstveni identifikacijski broj karte koja je kupljena
userId	INT	strani ključ entiteta User
idJourney	INT	strani ključ entiteta Journey
passenger-Name	VARCHAR	ime kupca karte
passenger-Surname	VARCHAR	prezime kupca karte
wagon	INT	vagon za koji je kupljena karta
wagonPosition	INT	pozicija vagona na kolodvoru
travelDate	DATE	datum putovanja

Journey - Ovaj entitet sadržava sve važne podatke za jedno putovanje vlakom. Sadrži attribute: IdJourney, departureTime i arrivalTime, price, departureStationName, arrivalStationName, routeId i sensorDataId. Atributi departureStationName i arrivalStationName su strani ključevi koji referenciraju entitet Station te se radi o *Many-To-One* vezi. Atributi IdTrainRoute i sensorDataId su strani ključevi koji referenciraju entitet TrainRoute i SensorData putem *Many-To-One* veze.

Journey		
idJourney	INT	jedinstveni identifikacijski broj putovanja
departureTime	DATETIME	datum i vrijeme polaska s polazišnog stajališta
arrivalTime	DATETIME	datum i vrijeme dolaska na odredišno stajalište
price	FLOAT	cijena karte za putovanje
departure-StationName	VARCHAR	početno stajalište putovanja i strani ključ entiteta Station
arrival-StationName	VARCHAR	završno stajalište putovanja i strani ključ entiteta Station
idTrainRoute	INT	jedinstveni identifikacijski broj TrainRoute-e
sensorDataId	INT	jedinstveni identifikacijski broj sensorData

TrainRoute - Entitet TrainRoute ima attribute: idTrainRoute, departureTime, arrivalTime, trainId i dates. U ruti Zagreb-Karlovac-Knin-Split, početak cjelokupne rute je Zagreb, a završetak Split. Entitet TrainRoute će imati sve moguće kombinacije podruta za primjer navedene rute. Journeys navedene rute će biti Zagreb-Karlovac, Zagreb-Knin, Zagreb-Split, Karlovac-Knin, Karlovac-Split, Knin-Split. Zbog potrebe zadatka da za svako putovanje imamo početno i završno odredište cjelokupne rute tog putovanja, nastao je ovaj entitet TrainRoute koji je povezan s Journeys *One-To-Many* vezom.

TrainRoute		
IdTrainRoute	INT	jedinstveni identifikacijski broj cjelokupne rute
departureTime	DATETIME	datum i vrijeme polaska s polazišnog stajališta
arrivalTime	DATETIME	datum i vrijeme dolaska na odredišno stajalište
trainId	INT	jedinstveni identifikacijski broj vlaka koji vozi tu rutu

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

TrainRoute		
dates	DATETIME	datum vožnji vlaka

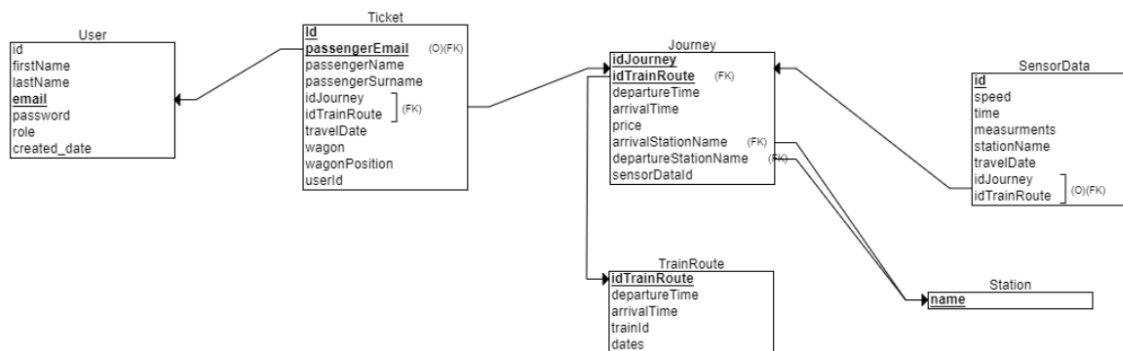
Station - Ovaj entitet sadržava imena svih stajališta. Entitet Stajalište povezan je putem dvije *One-To-Many* veze s entitetom Putovanje, gdje jedna veza označava početno stajalište, a druga završno.

Station		
name	VARCHAR	ime stajališta

SensorData - Ovaj entitet sadržava sve važne podatke za jedno mjerenje senzora. Sadrži attribute: id, time, speed, measurments, stationName, idJourney, idTrainRoute i travelDate. Entitet SensorData povezan je putem *Many-To-One* veze s entitetom Journey.

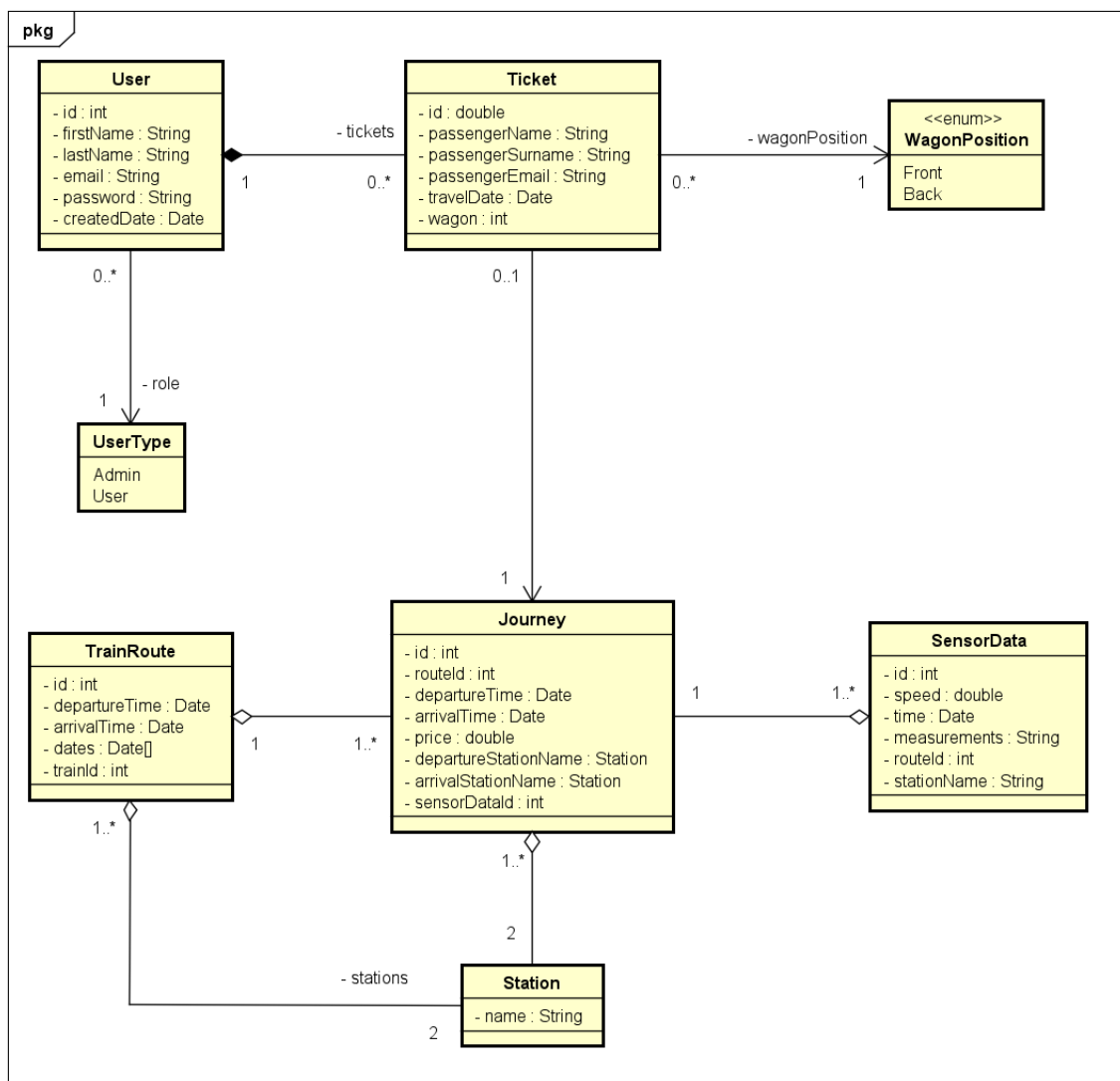
SensorData		
id	INT	jedinstveni identifikacijski broj mjerenja
time	DATETIME	datum i vrijeme mjerenja senzora
measurments	VARCHAR	vrijednost mjerenja senzora
speed	INT	brzina vlaka u trenutku mjerenja senzora
stationName	VARCHAR	stanica koju je vlak prošao uz senzor
travelDate	DATE	datum putovanja vlaka koji prolazi kraj senzora
idJourney	INT	strani ključ entiteta Journey
idTrainRoute	INT	strani ključ entiteta TrainRoute

4.1.2 Dijagram baze podataka



Slika 4.3: Dijagram baze podataka

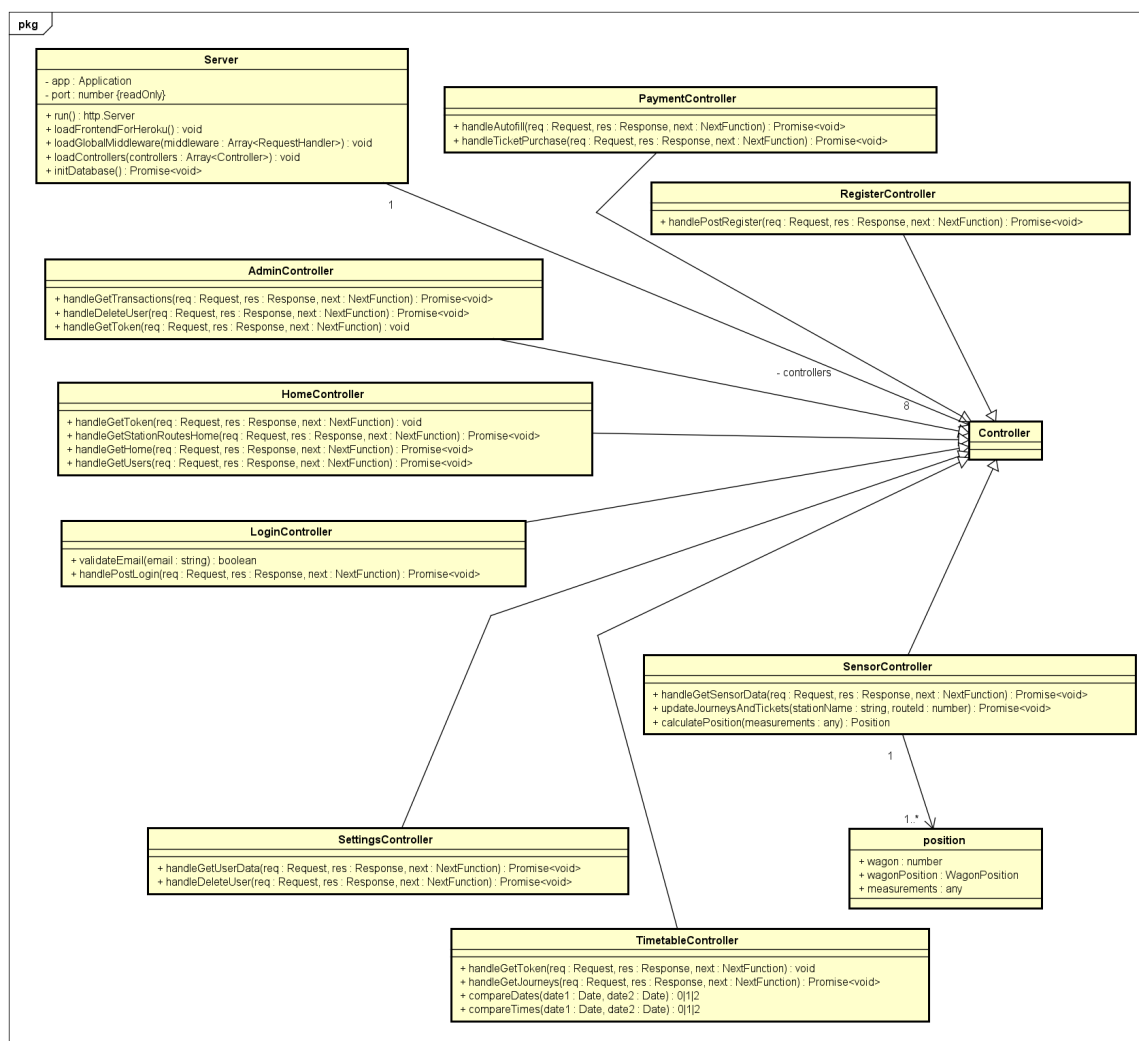
4.2 Dijagram razreda



Slika 4.4: Dijagram razreda Modela

Model razredi preslikavaju strukturu baze podataka u aplikaciji. Implementirane metode direktno komuniciraju s bazom podataka te vraćaju tražene podatke. Razred *User* predstavlja registriranog i prijavljenog korisnika koji može koristiti njegove osnovne funkcionalnosti. Razred *Ticket* predstavlja jednu kartu koju je kupio *User* za jedno putovanje - *Journey*. Razred *Journey* predstavlja relaciju putovanja za koju korisnik(*User*) kupuje željezničku kartu. Razred *TrainRoute* predstavlja kompletnu rutu na kojoj putuje vlak na kojem je korisnik kupio kartu za određeno

putovanje. Razred *Station* predstavlja listu svih stajališta kojima vozi naš prijevoznik, a razred *SensorData* predstavlja mjerenja sustava senzora Gotcha za svako putovanje.



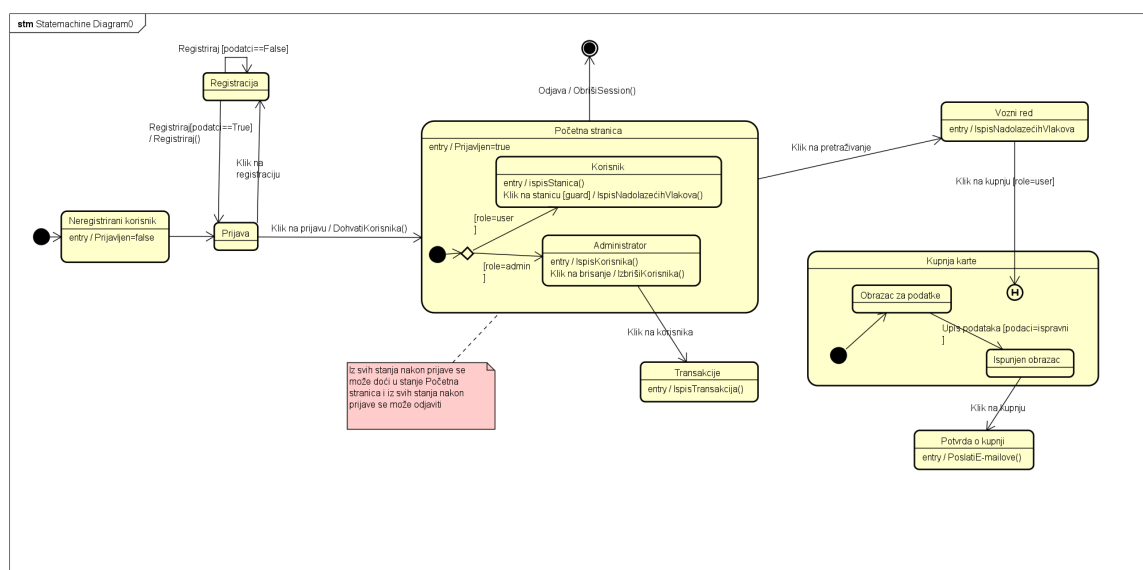
Slika 4.5: Dijagram razreda Controllera

Dijagram se sastoji od 8 nadglednika: HomeController, LoginController, RegisterController, AdminController, PaymentController, SensorController, SettingsController, TimetableController te svi naslijeđuju apstraktni razred Controller. U serveru se inicijaliziraju svi kontroleri (stavljaju u listu). HomeController se koristi prilikom ulaska korisnika na Home ekran i prikaz voznog reda na stranici. RegisterController se koristi prilikom registracije korisnika na stranicu, a LoginController prilikom prijave korisnika/admina u stranicu. AdminController se koristi prilikom funkcionalnosti koje su omogućene za Admina (pregleda, brisanje

korisnika). SensorController se koristi prilikom dohvata i obrade podataka sa senzora. PaymentController služi za obradu plaćanja prilikom kupnje karta te sprema podatke o transakcijama. SettingsController koristimo pri dohvat podataka korisnika te brisanju njegova računa. TimetableController koristimo pri dohvat putovanja za odabranu odlaznu i dolaznu stanicu i datum.

4.3 Dijagram stanja

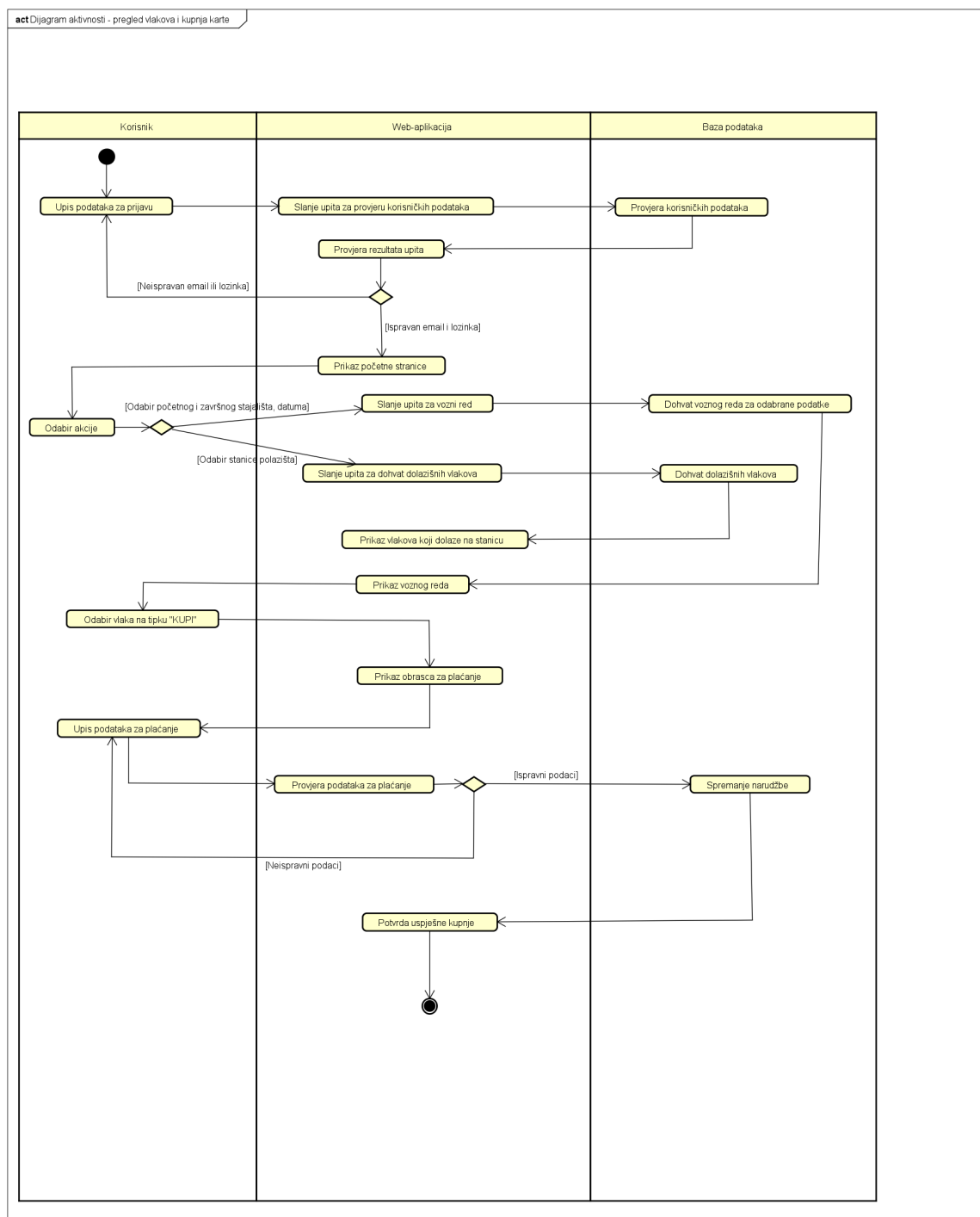
Dijagram stanja opisuje dinamičko ponašanje dijela sustava u vremenu. Ono sadrži konačan broj stanja i prijelaza među stanjima temeljenima na događajima. Na slici je prikazan dijagram stanja za sve uloge korisnika: konkretnog korisnika i administratora. Početno stanje neregistriranog korisnika (odnosno neprijavljenog) vodi u stanje prijave. Neregistriranim korisnicima je ostavljena mogućnost registracije, nakon čega je prijavom moguće doći do početne stranice aplikacije. Uloga korisnika razdjeljuje stanje početne stranice na dva unutarnja. Administratorima je omogućena akcija prikaza svih korisnika, njihovog brisanja i ispisa njihovih transakcija. Konkretnim korisnicima je omogućena akcija ispisa nadolazećih vlakova na odabranu stanicu. Svi korisnici imaju i opciju prikaza voznog reda. Međutim, samo konkretni korisnici imaju opciju kupnje karte za odabrano putovanje. Da bi se izvršila kupnja potrebno je ispuniti obrazac (ukoliko nije upamćen jer se radi o prvoj transakciji) te ju potvrditi. Potvrdom se šalju dva e-maila, prvi za potvrdu o kupnji te drugi sa sadržajem karte i informacijama o odabranom putovanju. Iz svih stanja je moguće odjaviti se ili se vratiti u stanje početne stranice.



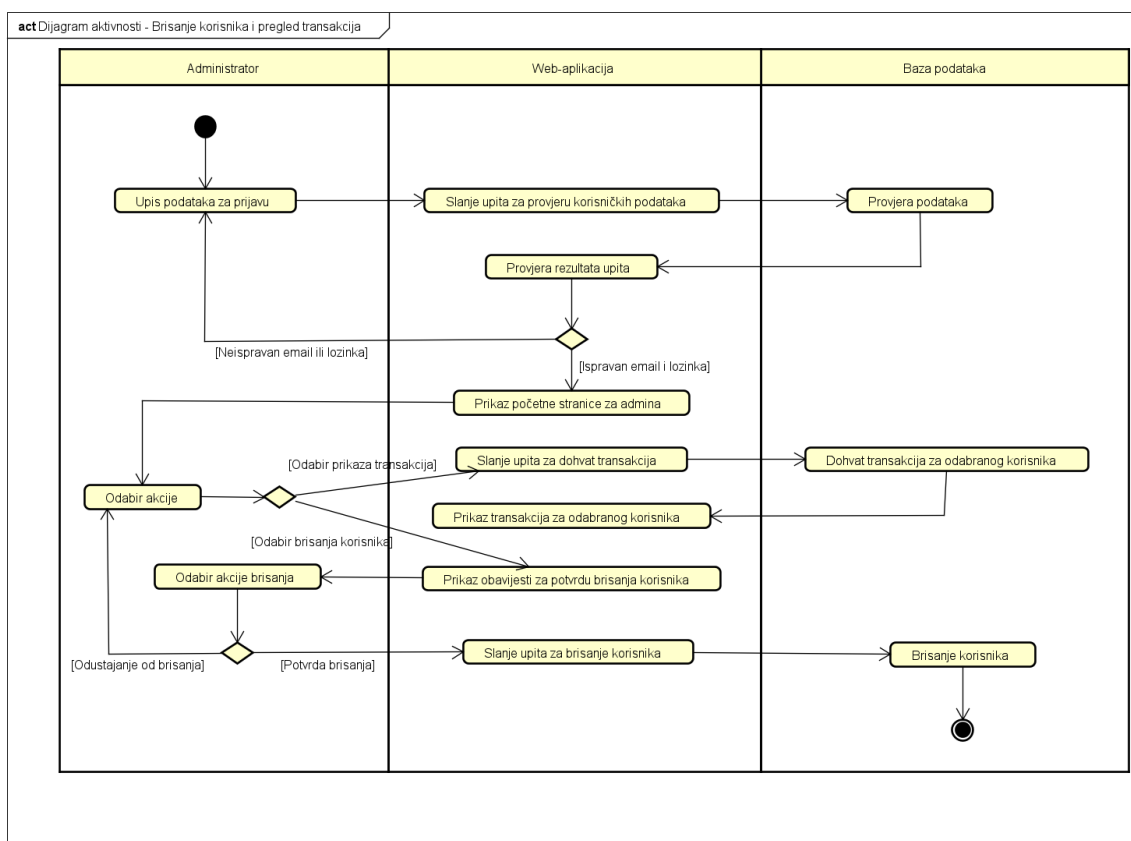
Slika 4.6: Dijagram stanja aplikacije

4.4 Dijagram aktivnosti

Dijagrami aktivnosti služe za modeliranje ponašanja nizom akcija, odnosno modeliranje toka i podataka. Pomaže nam pri zamišljanju redoslijeda aktivnosti i tijeka rada aplikacije. Na slici 4.7 prikazan je dijagram aktivnosti koji prikazuje procese pregledavanje i pretraživanje vlakova i voznih linija, odnosno kupnju karata za te linije. Korisnik se prijavljuje u sustav, zatim može pregledati vozni red pomoću odabira stanica i datuma polaska, te može pregledati koji vlakovi dolaze na označeno stajalište. Nakon odabira stanica i datuma dolazi do izlistanog voznog reda za te podatke pa odabire opciju kupnje. Zatim ispunjava obrazac za kartično plaćanje te potvrđuje svoju kupnju. Na slici 4.8 prikazan je dijagram aktivnosti administratora za brisanje korisnika i pregled njihovih transakcija. Nakon prijave, administrator dobiva izlistane korisnike za koje može odabrati brisanje ili pregled transakcija. Ukoliko odabere mogućnost brisanja, sustav će ga upozoriti i pitati je li siguran u brisanje toga klijenta. Administrator može prihvatiti brisanje ili odustati. Ako odabere pregled transakcija, aplikacija ga odvede na novu stranicu gdje su izlistane sve kupnje tog korisnika.



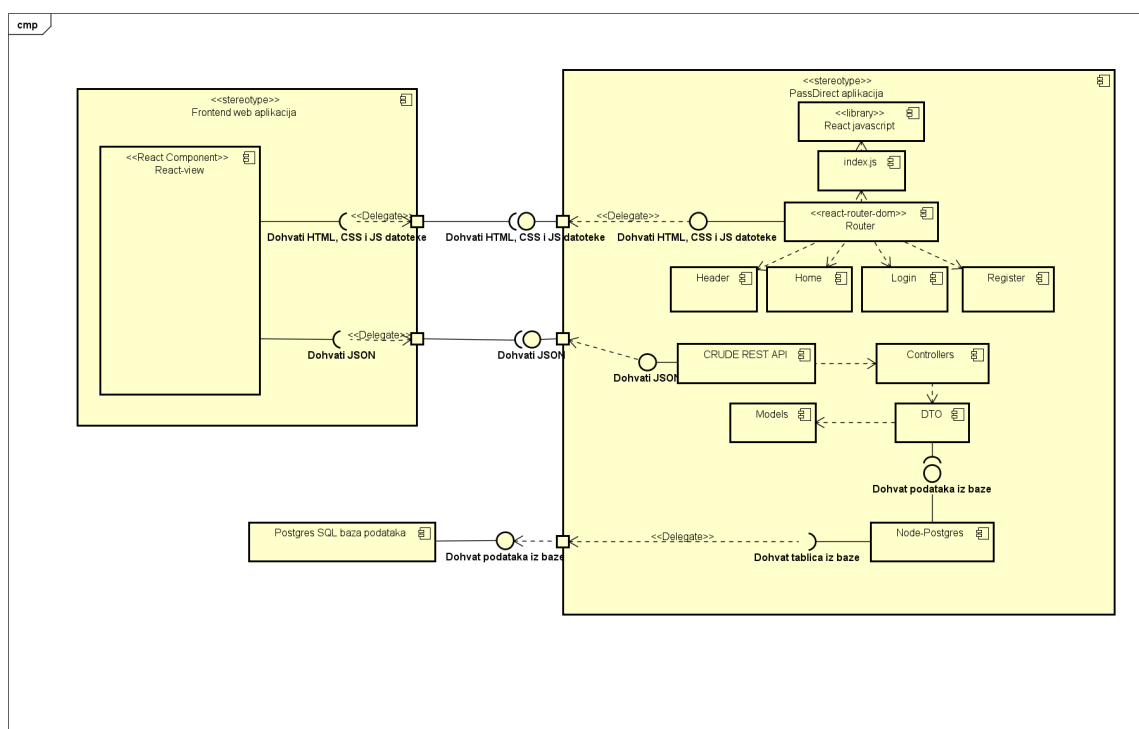
Slika 4.7: Dijagram aktivnosti - Pregled vlakova i kupnja karte



Slika 4.8: Dijagram aktivnosti - Brisanje korisnika i pregled transakcija

4.5 Dijagram komponenti

Dijagramom komponenti se vizualizira organizacija i međuovisnost između implementacijskih komponenata te odnos programske potpore prema okolini. Pogodan je za uslužno-orijentiranu arhitekturu i sačinjen je od komponenti, sučelja i poveznica. Sučeljem za dohvat HTML, CSS i JS datoteka dohvaćaju se datoteke sa frontenda aplikacije. Komponentom Router poslužuju se komponente stranice i React biblioteke na upit s URL-a. Dohvatom JSON podataka pristupa se CRUDE REST API komponenti koja komunicira sa backendom aplikacije. Node-Postgres je kolekcija funkcija za komunikaciju Node.js-a i PostgreSQL-a. Pristigli podaci iz baze se šalju MVC arhitekturi u obliku Data transfer object-a.



Slika 4.9: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

JavaScript (<https://www.javascript.com/>)

JavaScript je namijenjen omogućavanju dinamičnog načina stvaranja HTML elemenata i stvaranju interaktivnog sadržaja u HTML-u. Zajedno s CSS-om, JavaScript je osnova dinamičke web stranice.

TypeScript (<https://www.typescriptlang.org/>)

TypeScript je objektno orijentirani jezik, nadskup JavaScript-a koji se pri izvođenju prevodi u JavaScript, a u aplikaciji je korišten pri izradi *backenda*.

CSS (<https://www.w3schools.com/css/>)

CSS koristimo za oblikovanje stila i izgleda stranice na način prilagođen zaslonu korisnika.

HTML (<https://html.com/>)

HTML je označni jezik pomoću kojega definiramo strukturu, sadržaj i funkcije web-stranica.

React (<https://reactjs.org/>)

React je JavaScript knjižnica koja nam nudi mogućnost opisivanja korisničkog sučelja komponentama koje je lako oblikovati i upravljati njima. Korišten je za izradu *frontend* dijela aplikacije.

PostgreSQL (<https://www.postgresql.org/>)

PostgreSQL je sustav za upravljanje bazom podataka otvorenog koda. Sprema podatke u zasebne tablice uz mogućnost naknadne obrade i pristupa podacima.

SQL (<https://www.w3schools.com/sql/>)

SQL je alat za organiziranje, upravljanje i preuzimanje podataka pohranjenih u bazi podataka korištenoj u PostgreSQL-u.

Git (<https://git-scm.com/>)

Git je softver otvorenog koda korišten za kontroliranje verzija i praćenje promjena koda u aplikaciji koju izrađuje više korisnika.

GitLab (<https://about.gitlab.com/>)

GitLab je web platforma na kojoj se nalazi udaljeni repozitorij projekta i koja omogućuje lak uvid u promjene koda i dokumentacije, grafove aktivnosti i jednostavnu suradnju više sudionika projekta.

LaTeX (<https://www.latex-project.org/>)

LaTeX je softver koji omogućuje strukturirano slaganje i pripremu tekstova s opcijom formatiranja u oblik pogodan za pregled ili ispis.

Astah (<https://astah.net/>)

Astah je alat koji se koristio za izradu UML dijagrama.

Heroku (<https://www.heroku.com/>)

Heroku je web platforma koja omogućuje postavljanje i upravljanje web-aplikacijama.

WhatsApp (<https://www.whatsapp.com/>)

WhatsApp je aplikacija koja je korištena svakodnevno za lakšu komunikaciju i razmjenu ideja, problema.

Microsoft Teams (<https://www.microsoft.com/hr-hr/microsoft-teams/group-chat-software>)

Teams je aplikacija koja je korištena za e-sastanke i jedan dio pomoćne dokumentacije.

Postman (<https://www.postman.com/>)

Postman je API platforma namijenjena za korištenje API-ja. Njime se pojednostavljaju API lifecycle-i radi brže i jednostavnije izgradnje.

PostHook (<https://posthook.io/>)

Servis PostHook koristimo kako bismo ostvarili slanje POST zahtjeva s vremenskom odgodom. Pomoću ovog servisa simuliramo slanje sa senzora tijekom vožnje vlaka. Koristeći servis Postman, na PostHook šaljemo podatke koje smo definirali kao iščitavanja senzora, uz dodatne metapodatke s informacijom o zakazanom vremenu slanja danog podatka, ruti za slanje, itd. Servis Posthook dalje čeka definirano vrijeme slanja te prosljeđuje unesene podatke na rutu koju smo zadali.

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Ispitni slučaj 1: Testiranje funkcije validateEmail

Ulaz: "IspravanEmail123@email.com".

Očekivani rezultat: Funkcija vraća *true* jer je format e-maila dozvoljen.

Rezultat: Očekivani rezultat je zadovoljen. *Metoda je prošla test.*

```
it("Validacija e-maila sa ulazom u okviru dozvoljenih znakova",function(){
    email = "IspravanEmail123@email.com"
    assert.equal(true,validateEmail(email));
})
```

Slika 5.1: Ispravna validacija

Ispitni slučaj 2: Testiranje funkcije validateEmail

Ulaz 1: "NeispravanEmail.com".

Ulaz 2: "NeispravanEmail@domena".

Ulaz 3: "NeispravanEmail@domena.a".

Ulaz 4: "NeispravanEmail@domena.123".

Očekivani rezultati: Funkcija vraća *false* za svaki od ulaza jer su formati e-maila nedozvoljeni.

Rezultat: Očekivani rezultat je zadovoljen. *Metoda je prošla test.*

```
it("Validacija e-maila bez znaka @ , točke u domeni, samo jednim znakom nakon točke ili brojevima nakon točke",function(){
    let email = "NeispravanEmail.com";
    assert.equal(false,validateEmail(email));
    email = "NeispravanEmail@domena";
    assert.equal(false,validateEmail(email));
    email = "NeispravanEmail@domena.a";
    assert.equal(false,validateEmail(email));
    email = "NeispravanEmail@domena.123";
    assert.equal(false,validateEmail(email));
})
```

Slika 5.2: Validacija neispravno formatiranog e-maila

Ispitni slučaj 3: Testiranje funkcije validateEmail

Ulazi: znak+"@email.com", znak = "i", "z", "(", ")", "[", "]", ".", ",", ";", ":", " ", "@", "\"".

Očekivani rezultat: Funkcija vraća *false* za svaki od ulaza jer je format e-mailova nedozvoljen.

Rezultat: Očekivani rezultat je zadovoljen. *Metoda je prošla test.*

```
it("Validacija e-maila sa zabranjenim znakovima",function(){
  email = "@email.com"
  zabranjeniSimboli=["<",">","(",")","[","]","\\",".",",",";",":", " ","@", "\""];
  for (let i = 0; i < zabranjeniSimboli.length; i++) {

    let simbol = zabranjeniSimboli[i];
    assert.equal(false,validateEmail(simbol+email));
  }
})
```

Slika 5.3: Validacija e-maila koji sadrži zabranjene znakove

Ispitni slučaj 4: Kalkulacija pozicije s ispravnim podacima

Ulazi: Podaci="1":[10,9],"2":[12,23],"3":[6,0],"4":[2,9].

Očekivani rezultat: Funkcija vraća vagon i poziciju u vagonu s najmanjim brojem ljudi ukoliko razlika putnika u ostalim vagonima nije 30 ili više.

Rezultat: Očekivani rezultat je zadovoljen. *Metoda je prošla test.*

```
it("Kalkulacija pozicije s ispravnim podacima",function(){
  let measurements = {"1":[10,9],"2":[12,23],"3":[6,0],"4":[2,9]}
  let position = calculatePosition(measurements);
  assert.equal(3,position.wagon);
  assert.equal(1,position.wagonPosition);
})
```

Slika 5.4: Kalkulacija s ispravnim podacima

Ispitni slučaj 5: Kalkulacija pozicije s vlakom bez putnika

Ulazi: Podaci="1":[0,0],"2":[0,0],"3":[0,0],"4":[0,0].

Očekivani rezultat: Funkcija vraća stražnji dio prvog vagona.

Rezultat: Očekivani rezultat je zadovoljen. **Metoda je prošla test.**

```
it("Kalkulacija pozicije s vlakom bez putnika",function(){
  let measurements = {"1":[0,0],"2":[0,0],"3":[0,0],"4":[0,0]}
  let position = calculatePosition(measurements);
  assert.equal(1,position.wagon);
  assert.equal(1,position.wagonPosition);
})
```

Slika 5.5: Kalkulacija bez putnika

Ispitni slučaj 6: Kalkulacija pozicije s razlikom putnika većom od 30 u jednom vagonu

Ulazi: Podaci="1":[0,30],"2":[20,24],"3":[15,20],"4":[18,26].

Očekivani rezultat: Funkcija vraća vagon s razlikom od 30 putnika ili više.

Rezultat: Očekivani rezultat je zadovoljen. **Metoda je prošla test.**

```
it("Kalkulacija pozicije s razlikom putnika većom od 30 u jednom vagonu",function(){
  let measurements = {"1":[0,30],"2":[20,24],"3":[15,20],"4":[18,26]}
  let position = calculatePosition(measurements);
  assert.equal(1,position.wagon);
  assert.equal(0,position.wagonPosition);
})
```

Slika 5.6: Kalkulacija s razlikom od 30 putnika u jednom vagonu

Ispitni slučaj 7: Kalkulacija pozicije s razlikom putnika većom od 30 u više vagona

Ulazi: Podaci="1":[0,30],"2":[60,94],"3":[31,58],"4":[50,73].

Očekivani rezultat: Funkcija vraća vagon s najvećom razlikom.

Rezultat: Očekivani rezultat je zadovoljen. **Metoda je prošla test.**

```
it("Kalkulacija pozicije s razlikom putnika većom od 30 u više vagona",function(){
  let measurements = {"1":[30,0],"2":[60,94],"3":[31,58],"4":[50,73]}
  let position = calculatePosition(measurements);
  assert.equal(2,position.wagon);
  assert.equal(0,position.wagonPosition);
})
```

Slika 5.7: Kalkulacija s razlikom od 30 putnika u više vagona

Ispitni slučaj 8: Kalkulacija pozicije s neispravnim podacima

Ulazi: Podaci="1":"Krivi podatak","2":"Krivi podatak","3":"Krivi podatak","4":"Krivi podatak".

Očekivani rezultat: Funkcija izbacuje error o krivom podatku.

Rezultat: Očekivani rezultat je zadovoljen. **Metoda je prošla test.**

```
it("Kalkulacija pozicije s neispravnim podacima",function(){
  let measurements = {"1":"Krivi podatak","2":"Krivi podatak","3":"Krivi podatak","4":"Krivi podatak"}
  try {
    let position = calculatePosition(measurements);
  } catch (error) {
    assert.equal(error.constructor===TypeError,true);
  }
})
```

Slika 5.8: Kalkulacija s neispravnim podacima

Ispitivanje komponenti:

Ispitivanje funkcije za validaciju e-maila:

- ✓ Validacija e-maila sa ulazom u okviru dozvoljenih znakova
- ✓ Validacija e-maila sa zabranjenim znakovima
- ✓ Validacija e-maila bez znaka @ , točke u domeni, samo jednim znakom nakon točke ili brojevima nakon točke

Ispitivanje funkcije za kalkuciju pozicije:

- ✓ Kalkulacija pozicije s ispravnim podacima
- ✓ Kalkulacija pozicije s vlakom bez putnika
- ✓ Kalkulacija pozicije s razlikom putnika većom od 30 u jednom vagonu
- ✓ Kalkulacija pozicije s razlikom putnika većom od 30 u više vagona
- ✓ Kalkulacija pozicije s neispravnim podacima

8 passing (29ms)

Slika 5.9: Prikaz rezultata izvođenja JUnit testova u razvojnom okruženju

5.2.2 Ispitivanje sustava

Ispitivanje je provedeno pomoću Selenium IDE. Cilj je bila provjera funkcionalnosti, rubnih uvjeta i pogrešaka u sustavu. Selenium IDE je potrebno instalirati kao ekstenziju u tražilici. Zatim, nakon postavljanja URL-a testirane stranice, izvrši se test čije akcije ostaju pamćene u okviru Seleniuma i kako bi se test mogao izvršavati automatizirano. Testovi su izvršeni na aplikaciji otvorenoj na lokalnom računalu.

Ispitni slučaj 1: Registracija

Uvjeti:

1. Prilikom pokretanja testa korisnik ne smije već biti prijavljen.
2. Korisnik mora otvoriti stranicu za registraciju na PassDirect-u.

Očekivani rezultat: Registracija neće uspijeti za netočno potvrđenu lozinku i netočan oblik e-maila, a uspjeh će ako se točno ispravi i pokuša ponovno.

Tijek:

1. U polja na registracijskom obrascu uneseni su korisnički podatci s krivom potvrdom lozinke i krivim formatom e-maila.
2. Odabir gumba za registraciju.
3. Ispis poruke i odbijanje registracije zbog neispravnih podataka.
4. Točno ispravljena potvrda lozinke.
5. Ispis poruke i odbijanje registracije zbog neispravnih podataka.
6. Točno ispravljen e-mail.
7. Odabir gumba za registraciju.
8. Prihvata registracije i prikaz početnog ekrana.

Rezultat: Svi očekivani rezultati su zadovoljeni. Aplikacija je prošla test.

Running 'Registracija'
open on /login OK
click on css=.register OK

```
click on name=firstname OK
type on name=firstname with value Ivan OK
type on name=lastname with value Ivanic OK
type on name=email with value kriviemail@email@email.com OK
type on name=password with value Lozinka123 OK
type on name=confirmpassword with value KrivaLozinka OK
sendKeys on name=confirmpassword with value ${KEY_ENTER} OK
click on css=.app-wrapper OK
type on name=confirmpassword with value Lozinka123 OK
sendKeys on name=confirmpassword with value ${KEY_ENTER} OK
mouseDownAt on css=.app-wrapper with value 24,368 OK
mouseMoveAt on css=.app-wrapper with value 24,368 OK
mouseUpAt on css=.app-wrapper with value 24,368 OK
click on css=.app-wrapper OK
type on name=email with value dobaremail@email.com OK
sendKeys on name=email with value ${KEY_ENTER} OK
'Registracija' completed successfully
```

Ispitni slučaj 2: Neispravna prijava

Uvjeti:

1. Prilikom pokretanja testa korisnik ne smije već biti prijavljen.
2. Korisnik mora otvoriti početnu stranicu PassDirecta(Login).

Očekivani rezultat: Prijava u stranicu će biti odbijena zbog unosa e-maila neregistriranog korisnika.

Tijek:

1. U polja na obrascu za prijavu uneseni su e-mail za koji ne postoji prijavljeni korisnik u sustavu i lozinka.
2. Odabir gumba za registraciju.
3. Ispis poruke i odbijanje prijave zbog neispravnih podataka.

Rezultat: Očekivani rezultat je zadovoljen, pošto korisnik nije ulogiran. Aplikacija je prošla test.

Running 'Neispravna prijava '
open on /login OK
click on name=email OK
type on name=email with value kriviEmail@email.com OK
type on name=password with value Lozinka123 OK
click on css=.submit OK
mouseOver on css=.submit OK
mouseOut on css=.submit OK
'Neispravna prijava ' completed successfully

Ispitni slučaj 3: Prijava, pretraga voznog reda i kupnja karte

Uvjeti:

1. Prilikom pokretanja testa korisnik ne smije već biti prijavljen.
2. Korisnik mora otvoriti početnu stranicu PassDirecta(Login).

Očekivani rezultat: Uspješna prijava u stranicu PassDirecta, pretraga voznog reda, te kupnja karte za odabranu liniju, odnosno vlak.

Tijek:

1. U polja na obrascu za prijavu uneseni su e-mail i lozinka za postojećeg korisnika.
2. Odabir gumba za registraciju.
3. Prikaz početnog ekrana aplikacije.
4. Odabir mjesta polaska i mjesta dolaska te datuma putovanja u zaglavlju stranice.
5. Odabir gumba za pretraživanje voznog reda.
6. Prikaz dostupnih linija za odabrane podatke.
7. Odabir željene linije za koju se kupuje karta.
8. Prikaz ekrana za checkout s već popunjenim e-mailom, imenom i prezime-nom.

9. Unos podataka o kartici.
10. Odabir gumba za plaćanje.
11. Prikaz obavijesti o uspješnoj transakciji.

Rezultat: Očekivani rezultati se zadovoljeni: nakon prijave u sustav, korisnik pretraži vozni red i kupi kartu. Aplikacija je prošla test.

```
Running 'Prijava , pretraga voznog reda i kupnja karte '  
open on /login OK  
click on name=email OK  
type on name=email with value pero@email.com OK  
type on name=password with value Lozinka123 OK  
sendKeys on name=password with value ${KEY_ENTER} OK  
click on name=to OK  
select on name=to with value label=Split OK  
click on css=.right > .button OK  
mouseOver on css=.right > .button OK  
mouseOut on css=.right > .button OK  
click on xpath=//div[@id='100']/h5/span[8]/button OK  
click on id=cardNo OK  
type on id=cardNo with value 1111 1111 1111 1111 OK  
click on id=CVV OK  
type on id=CVV with value 111 OK  
click on id=expDate OK  
click on id=expDate OK  
click on id=expDate OK  
type on id=expDate with value 0001-01 OK  
type on id=expDate with value 0012-01 OK  
click on css=.submit OK  
click on css=.button-succesful OK  
'Kupnja karte' completed successfully
```

Ispitni slučaj 4: Prijava administratora, ispis korisnika i brisanje korisnika
Uvjeti:

1. Prilikom pokretanja testa korisnik ne smije već biti prijavljen.
2. Korisnik mora otvoriti početnu stranicu PassDirecta(Login).
3. Korisnik prije prijave mora imati dodijeljenu rolu administratora.
4. U bazi/sustavu mora postojati barem jedan korisnik koji.

Očekivani rezultat: Uspješna prijava administratora u stranicu, pregled ispisa korisnika i brisanje korisničkih profila.

Tijek:

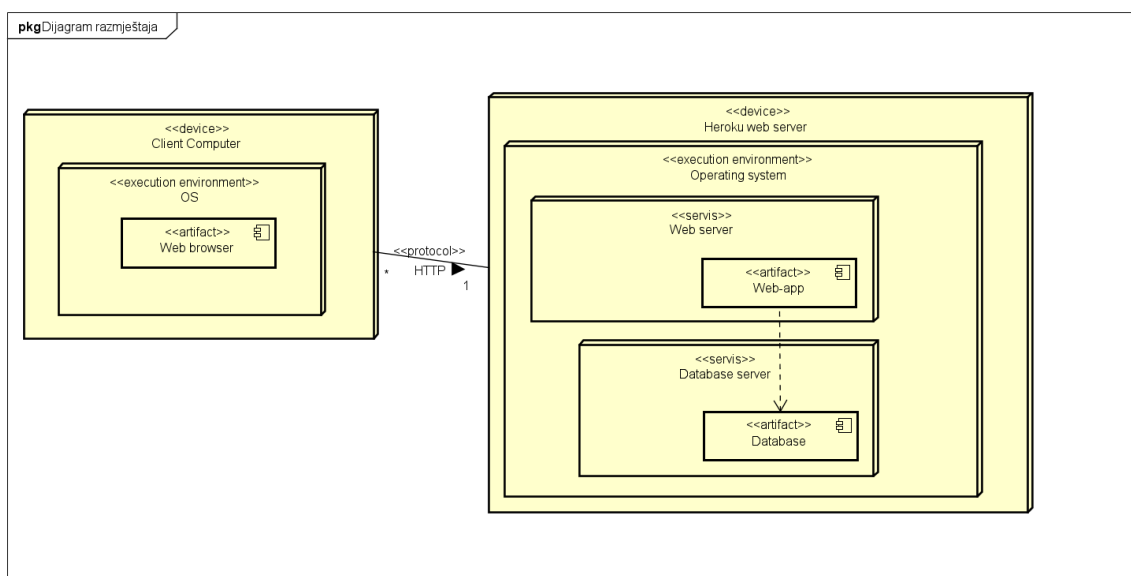
1. U polja na obrascu za prijavu uneseni su e-mail i lozinka za postojećeg administratora.
2. Odabir gumba za registraciju.
3. Prikaz početnog ekrana aplikacije s listom korisnika.
4. Odabir gumba za brisanje kraj jednog od korisnika.
5. Korisnik je obrisani iz baze podataka.
6. Prikaz osvježene liste korisnika.

Rezultat: Očekivani rezultati se zadovoljeni: nakon prijave u sustav, administrator pregleda i izbriše korisnika. Aplikacija je prošla test.

```
Running 'Prijava administratora ,ispis korisnika i brisanje profila '  
open on /login OK  
click on name=email OK  
type on name=email with value admin@email.com OK  
type on name=password with value Lozinka123 OK  
sendKeys on name=password with value ${KEY_ENTER} OK  
click on css=#korisnik1\@email\.com span:nth-child(5) > .buttonAH OK  
'Prijava administratora ,ispis korisnika i brisanje profila ' completed  
successfully
```


5.3 Dijagram razmještaja

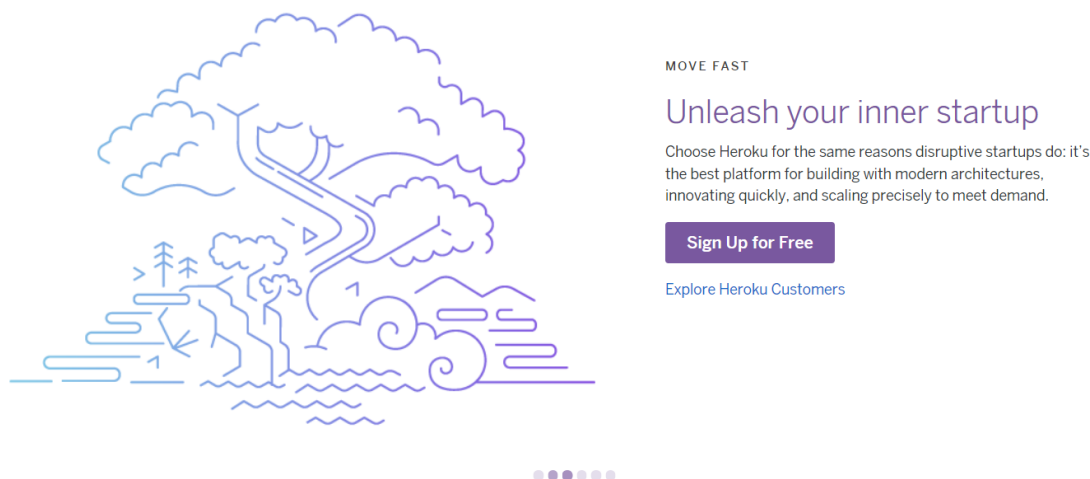
Dijagram razmještaja statički opisuje topologiju sustava s fokusom na odnos skopovlja i programskog dijela projekta. Organizacija sustava se zasniva na arhitekturi klijent - poslužitelj. Klijent na svome računalu koristi web preglednik kako bi pristupio aplikaciji koja se nalazi na Heroku servisu. Više korisnika može pristupiti aplikaciji, putem HTTP veze. Heroku web server sadrži web poslužitelj i poslužitelj baze podataka na kojima se nalaze sama aplikacija, odnosno baza podataka aplikacije.



Slika 5.10: Dijagram razmještaja

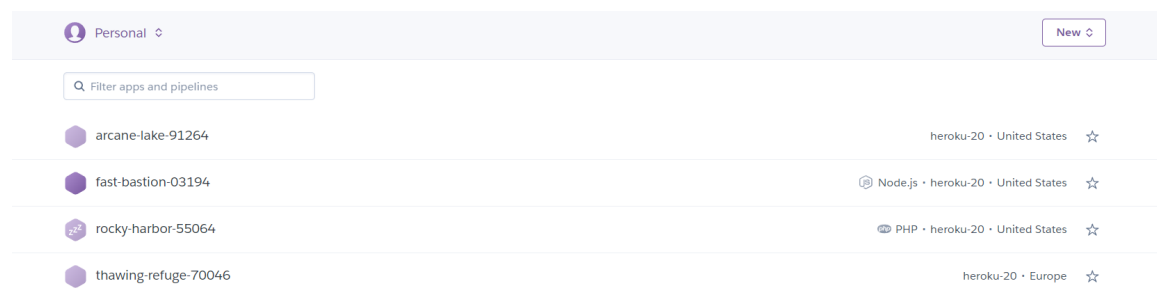
5.4 Upute za puštanje u pogon

U ovom poglavlju opisano je puštanje aplikacije u pogon. Puštanje je izvedeno uz pomoć platforme Heroku.



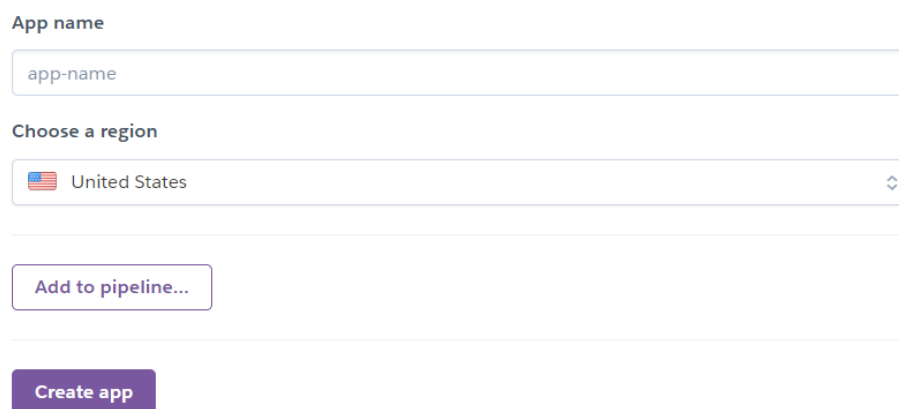
Slika 5.11: Platforma Heroku

Najprije je potrebno registrirati račun te se prijaviti. Nakon prijave korisniku je otvorena stranica sa postojećim aplikacijama te opcijom za stvaranje nove.



Slika 5.12: 1. korak: "New"

Odabirom gumba za stvaranje nove aplikacije otvara se obrazac za popunjavanje polja za osnovne informacije o aplikaciji (naziv i regija).



The image shows a web form for creating a new Heroku application. It consists of the following elements:

- A label "App name" above a text input field containing the placeholder "app-name".
- A label "Choose a region" above a dropdown menu showing "United States" with a small American flag icon and a reset icon.
- A button labeled "Add to pipeline..." below the region dropdown.
- A button labeled "Create app" at the bottom of the form.

Slika 5.13: 2. korak: Obrazac

Nakon što su je Heroku platformi napravljena i postavljena aplikacija, još je preostao upload izvornog koda uz pomoć HerokuCLI. U terminalu na računalu se pozicionira u direktorij IzvorniKod i unese se naredba za login u heroku:

- heroku login

Sada moramo izvesti sljedeće naredbe kako bismo postavili aplikaciju na Heroku:

- cd ../izvorniKod

- heroku git: remote -a naziv-aplikacije-na-heroku

- git add .

- git commit -m "Commit"

- git push heroku master

Kreiranje baze podataka

U ormconfig.json datoteci su nam zapisani podaci za povezivanje i kreiranje baze podataka kako je prikazano na slici ispod.

```
{  
  "username": "postgres",  
  "password": "bazepodataka",  
  "host": "localhost",  
  "database": "passdirect-test",  
  "port": 5432,  
  "type": "postgres",  
  "entities": ["src/models/*.json"],  
  "synchronize": true,  
  "logging": false  
}
```

Slika 5.14: Kreiranje baze podataka

Kako bismo imali kontrolu nad našom bazom podataka koja se sada nalazi na herokuovom serveru, za naš heroku projekt dodajemo Heroku Postgres plugin. Ovaj plugin nam omogućuje da se premjestimo u psql CLI te otud koristeći SQL upite kontroliramo i izmjenjujemo stanje BP. Kako bismo pokrenuli CLI potrebno je u terminalu računala na kojem radimo upisati naredbu: `heroku pg:psql --app pass-direct`

6. Zaključak i budući rad

Zadatak naše projektne grupe bio je razvoj aplikacije za pregled i kupnju karata za vlakove s ciljem očuvanja željezničke infrastrukture prijevoznika. Nakon nešto više od 3 mjeseca rada, uglavnom smo uspjeli ostvariti taj cilj. Naš projekt se odvijao u dvama ciklusima.

U prvome ciklusu, na početku samoga projekta najveći problemi bili su neiskustvo većine članova s alatima potrebnim za izradu aplikacije te grupni rad na ovakvom tipu zadatka (projekta), no vremenom se stekao osjećaj za rad u timu i radom su se stekla nova znanja i vještine s dotad nepoznatom programskom podrškom. Nakon odabira vođe projekta i okvirne podijele posla, izradili su se obrasci uporabe i njima pripadajući dijagrami koji su uvelike pomogli pri početku razvoja aplikacije kao glavne vodilje pri nastalim nejasnoćama i nedoumicama. Cilj prvoga ciklusa je bio da se napravi kostur aplikacije i osnovne funkcionalnosti koje će se u drugom ciklusu nadograditi i specificirati.

U drugome ciklusu tim je podijeljen na *backend* i *frontend* timove zbog lakšeg i efikasnijeg rada. Dokumentacija je dorađena nakon prvog kolokviranja te se dalje izrađivala postupno s gradivom. Izrada i promjena aplikacije morala je biti u skladu s njom, što je u trenutcima bilo dosta korisno. U ovome ciklusu se pokazao odličan timski rad, unatoč tome što se većina članova tima morala upoznavati s novim alatima i metodama rada. Velik se dio funkcionalnosti napravio u ovom ciklusu uz intenzivan, ali efikasan rad.

Komunikacija je tekla putem WhatsApp-a što je pospiješilo informiranost i brzo rješavanje problema. Komunikacija je vremenom postajala sve bolja i opuštenija, što je dobro utjecalo na uspjehe tima.

Rad na ovome projektu bio je cijelome timu odlično iskustvo za daljnji rad u timovima, a iskustvo s novim tehnologijama i alatima može samo pomoći individualnim inženjerskim razvojem. Svijesni smo da su neke stvari mogle ići lakše, bolje, drugačije i efikasnije, ali smo na koncu zadovoljni rezultatom i sigurni smo da ćemo iz ovog projekta izvući puno dobrih stvari.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. The Unified Modeling Language, <https://www.uml-diagrams.org/>
3. Astah Community, <http://astah.net/editions/uml-new>
4. Software design-Vlado Sruk,
https://moodle.fer.hr/pluginfile.php/78699/mod_resource/content/1/erasmus_Software%20Design..pdf
5. Git community, <http://git-scm.com/>
6. Gitlab Platform, <https://about.gitlab.com/>
7. ReactJs community, <https://reactjs.org/docs/getting-started.html>
8. Express, <https://expressjs.com/>
9. Typescript community, <https://www.typescriptlang.org/>
10. pgAdmin organization, <https://www.pgadmin.org/>
11. Use Case Modeling, https://books.google.hr/books?id=zvxfXvEcQjUC&printsec=frontcover&dq=Use+Case+Modeling:+Software+Systems&hl=hr&sa=X&ved=0ahUKEwi jxbTF0p31AhWGbFAKHeuFC_gQ6AEIMTAB#v=onepage&q=Use%20Case%20Modeling%3A%20Software%20Systems&f=false
12. Git visualization, <http://git-school.github.io/visualizing-git/#free>
13. W3Schools, <https://www.w3schools.com/>
14. HŽ putnički prijevoz, <http://www.hzpp.hr/>
15. Citymapper, <https://citymapper.com/>

Indeks slika i dijagrama

2.1	Web stranica HŽPP-a	11
2.2	Web stranica cityMapper	11
3.1	UC dijagram	19
3.2	UC3 - Pregled vlakova za stajalište	20
3.3	UC4 - Pretraživanje voznog reda	21
3.4	UC5 - Kupnja karte	22
3.5	UC6 - Pregled klijenata	23
3.6	UC7 - Pregled transakcija	24
3.7	UC8 - Pregled voznog reda	25
3.8	UC9 - Brisanje klijenta	26
3.9	UC11 - Brisanje računa	27
4.1	Arhitektura sustava	29
4.2	Model-View-Controller	30
4.3	Dijagram baze podataka	35
4.4	Dijagram razreda Modela	36
4.5	Dijagram razreda Controllera	37
4.6	Dijagram stanja aplikacije	39
4.7	Dijagram aktivnosti - Pregled vlakova i kupnja karte	41
4.8	Dijagram aktivnosti - Brisanje korisnika i pregled transakcija	42
4.9	Dijagram komponenti	43
5.1	Ispravna validacija	47
5.2	Validacija neispravno formatiranog e-maila	47
5.3	Validacija e-maila koji sadrži zabranjene znakove	48
5.4	Kalkulacija s ispravnim podacima	48
5.5	Kalkulacija bez putnika	49
5.6	Kalkulacija s razlikom od 30 putnika u jednom vagonu	49
5.7	Kalkulacija s razlikom od 30 putnika u više vagona	50
5.8	Kalkulacija s neispravnim podacima	50

5.9	Prikaz rezultata izvođenja JUnit testova u razvojnem okruženju . . .	50
5.10	Dijagram razmještaja	56
5.11	Platforma Heroku	57
5.12	1. korak: "New"	57
5.13	2. korak: Obrazac	58
5.14	Kreiranje baze podataka	59
6.1	Dijagram aktivnosti na grani <i>develop</i>	69
6.2	Dijagram aktivnosti na grani <i>devdoc</i>	70

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 07.10.2021.
- Prisustvovali: T. Pavković, M. Vučinić, A. Vinković, K. Pavlović, F. Cindrić, M. Galić, I. Hajpek
- Teme sastanka:
 - Razgovor o poznavanju tehnologija
 - Dogovor o tutorialima i upute za okviran način rad

2. sastanak

- Datum: 16.10.2021.
- Prisustvovali: T. Pavković, M. Vučinić, A. Vinković, K. Pavlović, F. Cindrić, M. Galić, I. Hajpek
- Teme sastanka:
 - Komentiranje dobivene teme i razjašnjavanje nejasnoća
 - Upoznavanje s GitLab-om i LaTeX-om

3. sastanak

- Datum: 23.10.2021.
- Prisustvovali: T. Pavković, M. Vučinić, A. Vinković, K. Pavlović, F. Cindrić, M. Galić, I. Hajpek
- Teme sastanka:
 - Dogovorene razvojne okoline i tehnologije
 - Rasprava o izradi generičkih sposobnosti
 - Dogovor za početak razvijanje aplikacija
 - Dogovoren početak rada na predlošku dokumentacije i izgledu stranice

4. sastanak

- Datum: 30.10.2021.
- Prisustvovali: T. Pavković, M. Vučinić, A. Vinković, K. Pavlović, F. Cindrić, M. Galić, I. Hajpek

drić, M. Galić, I. Hajpek

- Teme sastanka:
 - Pregled prethodno zadanih zadataka
 - Određen kostur aplikacije
 - Završene početne verzije UC-ova i dijagrama
 - Početak dizajna baze podataka

5. sastanak

- Datum: 06.11.2021.
- Prisustvovali: T. Pavković, M. Vučinić, A. Vinković, K. Pavlović, F. Cindrić, M. Galić, I. Hajpek
- Teme sastanka:
 - Doradivanje dokumentacije
 - Modeliranje entiteta i dizajn kontrolera
 - Doradivanje dosadašnjeg front-end-a

6. sastanak

- Datum: 14.11.2021.
- Prisustvovali: T. Pavković, M. Vučinić, A. Vinković, K. Pavlović, F. Cindrić, M. Galić, I. Hajpek
- Teme sastanka:
 - Provjere na front-endu
 - Pravljenje arhitekture i dizajna sustava u dokumentaciji
 - Daljnji rad na back-endu, kontrolerima i modelima

7. sastanak

- Datum: 16.11.2021.
- Prisustvovali: T. Pavković, M. Vučinić, A. Vinković, K. Pavlović, F. Cindrić, M. Galić, I. Hajpek
- Teme sastanka:
 - Završne promjene na dokumentaciji i bazi podataka
 - Povezivanje back-end-a i front-end-a
 - Dizanje baze na Heroku i deployanje aplikacije

8. sastanak

- Datum: 6.12.2021.
- Prisustvovali: T. Pavković, M. Vučinić, A. Vinković, K. Pavlović, F. Cindrić, M. Galić, I. Hajpek
- Teme sastanka:

- Napravljen pregled programske podrške projekta
- Podijela projektnog tima na backend i frontend
- Frontend: fokus na radu funkcionalnog prikaza voznog reda i kupnje karata
- Backend: uređivanje ruta i rad na kontrolerima

9. sastanak

- Datum: 14.12.2021.
- Prisustvovali: T. Pavković, M. Vučinić, A. Vinković, K. Pavlović, F. Cindrić, M. Galić, I. Hajpek
- Teme sastanka:
 - Napravljen pregled odrađenih poslova s prošlog sastanka
 - Povezivanje frontenda i backenda
 - Doradivanje Admin stranica na frontendu
 - Rad na implementaciji senzora
 - Nastavak rada na UML dijagramima

10. sastanak

- Datum: 18.12.2021.
- Prisustvovali: T. Pavković, M. Vučinić, A. Vinković, K. Pavlović, F. Cindrić, M. Galić, I. Hajpek
- Teme sastanka:
 - Privođenje kraju posla na frontendu, doradivanje izgleda
 - Rad na implementaciji kontrolera za senzore i načinu dohvata podataka senzora
 - Rad na dokumentaciji postupno s gradivom
 - Provjera konzistentnosti i uređivanje modela na backendu

11. sastanak

- Datum: 12.1.2022.
- Prisustvovali: T. Pavković, M. Vučinić, A. Vinković, K. Pavlović, F. Cindrić, M. Galić, I. Hajpek
- Teme sastanka:
 - Doradivanje pojedinosti vezanih uz senzore
 - Unit testing
 - Izrada podataka za punjenje baze podataka
 - Provjera i testiranje rada aplikacije

Tablica aktivnosti

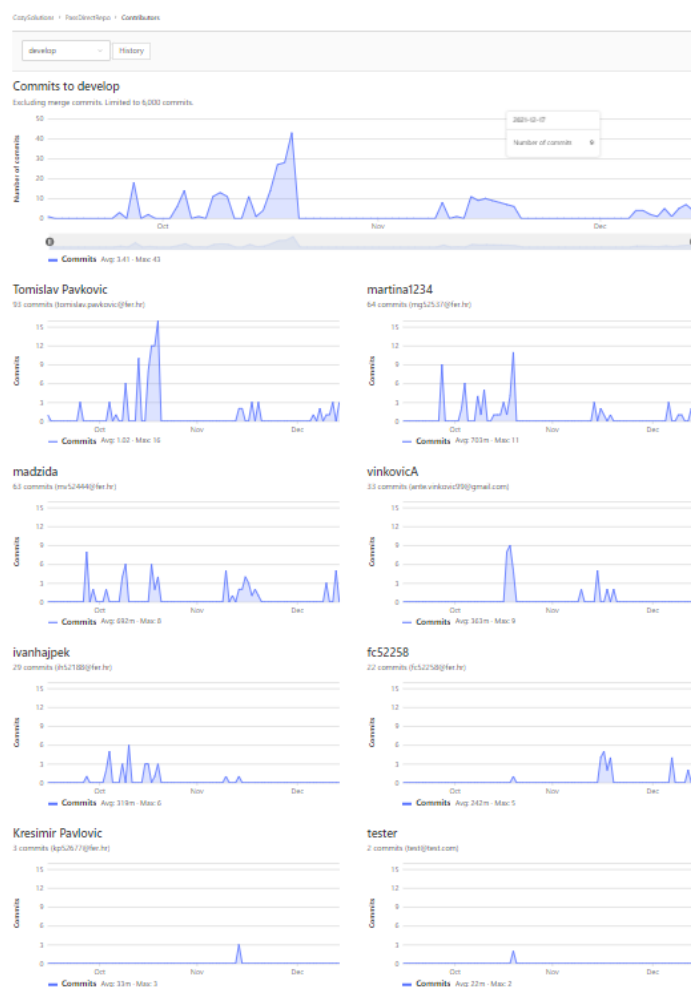
	Tomislav Pavković	Ante Vinković	Martina Galić	Filip Cindrić	Krešimir Pavlović	Mirta Vučinić	Ivan Hajpek
Upravljanje projektom	15	3					
Opis projektnog zadatka			7		3	1	
Funkcionalni zahtjevi							7
Opis pojedinih obrazaca					2		9
Dijagram obrazaca			5		1		1
Sekvencijski dijagrami			7				4
Opis ostalih zahtjeva							3
Arhitektura i dizajn sustava	1	2	5		2		4
Baza podataka		3	14				11
Dijagram razreda			6				6
Dijagram stanja					3		
Dijagram aktivnosti							3
Dijagram komponenti					3		
Korištene tehnologije i alati		1					2
Ispitivanje programskog rješenja					15		10
Dijagram razmještaja							1
Upute za puštanje u pogon		1			3		
Dnevnik sastajanja		10					3
Zaključak i budući rad							1

Nastavljeno na idućoj stranici

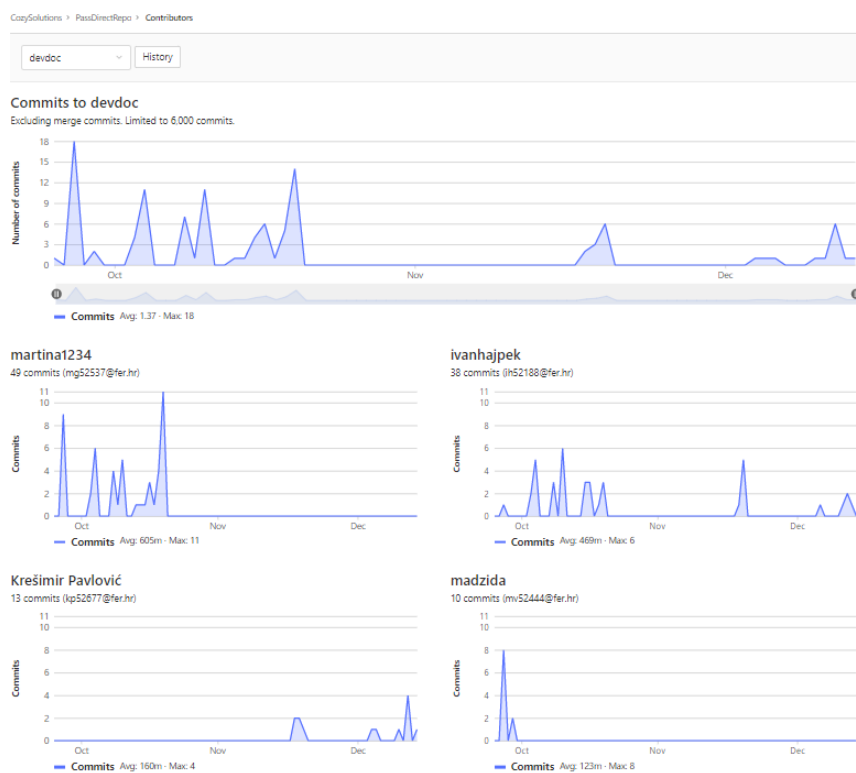
Nastavljeno od prethodne stranice

	Tomislav Pavković	Ante Vinković	Martina Galić	Filip Cindrić	Krešimir Pavlović	Mirta Vučinić	Ivan Hajpek
Popis literature							1
<i>Izrada početne stranice</i>	8				15	6	1
<i>Izrada baze podataka</i>	9	5	5			3	
<i>Spajanje s bazom podataka</i>	10	8				25	
<i>Back end</i>	55	20	17				
<i>Login</i>	5				5	12	
<i>Registracija</i>	7					15	
<i>Admin home</i>	6			16		8	2
<i>Pregled vlakova</i>	15		8			7	3
<i>Kupnja karata</i>	12			30		4	
<i>Deploy</i>	4	5	5				5
<i>Izrada dizajna</i>	6	5		5	15	4	
<i>Izrada testnih podataka</i>		15		2			
<i>Rad na senzorima</i>	20	5					
<i>Slanje mailova</i>	7		10			1	
<i>Pregled transakcija</i>				5			
<i>Brisanje računa</i>		1		2			
<i>Kalkulacija vagona i pozicija</i>	10						

Dijagrami pregleda promjena



Slika 6.1: Dijagram aktivnosti na grani *develop*

Slika 6.2: Dijagram aktivnosti na grani *devdoc*