Univerzitet Singidunum
Fakultet za informatiku i računarstvo

Aplikacija za upravljanje auto servisom

Razvojni tim

Petar Kresoja 2019200948
Marko Savic 2020202977

Razvoj aplikativnog softvera

Prof. dr Violeta Tomašević
Asistent mr Petar Jakić

Sadrzaj

1.Uvod

Zadatak projekta je razvoj aplikacije za upravljanje auto servisom.Aplikacija podrzava rad vise korisnika.Aplikacija je isprojektovana tako da prijavljenom korisniku prikazuje podatke svojstvene ulozi koju ima u sistemu.

Vlasnik vozila predaje zahtev za registraciju radi evidencije jednog ili više svojih vozila koja su u kvaru. Zaposleni ga informiše o proceduri i upućuje da popuni upitnik za registraciju.Nakon popunjenog upitnika(koji sadrzi podatke o licnim podacima vlasnika vozila i podatke o vozilu), zaposleni unosi podatke u sistem, otvara radni nalog i registruje vozilo, dodeljuje slobodnog servisera...

Aplikacija treba da omoguci vlasniku vozila da unosom šasije u odgovarajuće polje u aplikaciji, izvrši pregled statusa svog vozila, pregled radnih naloga, plaćanje...

Potrebno je omoguciti da serviser, vrši prijem vozila, pregled, otvara dodatne radne naloge, trebuje delove iz magacina, menja status radnih naloga njemu dodeljenih vozila. Fakturisanje radnih naloga koji se nalaze u statusu završeno, trebaju biti prioritet.

Aplikacija treba biti prilagodjena tako da zahteva minimalanu kolicinu resursa i ostvaruje vremenske uštede u radu autoservisa, pruzajući uvid u sve potrebne informacije vezane za ciklus poslovanja.U nastavku dokumenta bice predstavljen detaljan opis koraka u procesu razvoja aplikacije, kao i uputstvo za upotrebu datog softvera.

2.Proces razvoja aplikacije

Zadatak ovog koraka jeste utvrdjivanje skupa zadataka (faza u razvoju), koje treba realizovati u cilju dobijanja jedne funkcionalne aplikacije za upravljanje auto servisom

2.1.Metoda razvoja

U cilju koriscenja softvera u ranim fazama i mogucnosti obavljanja rane obuke korisnika, planirana metoda razvoja jeste fazni razvoj - Iterativni pristup. Isporuka softvera ce se vrsiti po fazama uz obavljanje aktivnosti nakon svake faze.

2.2 Kritične tačke

2.2.1 Inicijalni sastanak tima:

- **Cilj:** Diskusija o zahtevima klijenta
- **Ucesnici:** Marko Savic i Petar Kresoja.
- **Vremenski rok inicijalnog sastanka tima:** 15.05.2022.
- **Napomena**: Sa obzirom da se radi o faznom razvoju, sastanak se sprovodi (23.05.2022. godine, 31.05.2022. godine, 07.06.2022. godine), nakon isporuke verzije softvera.

2.2.2 Analiza zahteva :

- **Cilj:** Prikupljanje, specifikacija, verifikacija i validacija zahteva korisnika aplikacije i kupca.
- **Ucesnici:** Marko Savic i Petar Kresoja.
- **Vremenski rok inicijalne analize:** 16.05.2022.
- **Napomena**: Sa obzirom da se radi o faznom razvoju, analiza se sprovodi (24.05.2022. godine, 01.06.2022. godine, 08.06.2022. godine), a nakon isporuke verzije softvera.

2.2.3 Arhitektura sistema:

- **Cilj:** Odabir odgovarajuće arhitekture u skladu sa zahtevima korisnika, koja kao rezultat zadovoljava sve uslove za rad aplikacije
- **Ucesnici:** Marko Savic i Petar Kresoja.
- **Vremenski rok:** 17.05.2022.

2.2.4 Baza podataka:

- **Cilj:** Izrada funkcionalne baze podataka koja ispunjava zahteve korisnika u cilju upravljanja podacima - opsluživanja korisnika sa funkcionalnostima kreiranja, čitanja, ažuriranja, brisanja podataka (*CRUD* funkcionalnostima*);*
- **Ucesnici:** Marko Savic
- **Vremenski rok :** 18.05.2022.
- **Napomena**: Sa obzirom da se radi o faznom razvoju, izrada se sprovodi (27.05.2022. godine, 03.06.2022. godine, 10.06.2022. godine), nakon revizije analize zahteva i isporuke verzije softvera.

2.2.5 Aplikacija (*backend*, *frontend*):

- **Cilj**: Izrada funkcionalne aplikacije koja zadovoljava sve zahteve iz faze analiza zahteva
- **Ucesnici:** Petar Kresoja.
- **Vremenski rok :** 19.05.2022.
- **Napomena**: Sa obzirom da se radi o faznom razvoju, izrada se sprovodi (28.05.2022. godine, 04.06.2022. godine, 11.06.2022. godine), nakon revizije analize zahteva i isporuke verzije softvera.

2.2.6 Testiranje:

- **Cilj:** Utvrdjivanje što većeg broja grešaka u sistemu
- **Ucesnici:** Marko Savic i Petar Kresoja
- **Vremenski rok :** 20.05.2022.
- **Napomena**: Sa obzirom da se radi o faznom razvoju, testiranje se sprovodi (29.05.2022. godine, 05.06.2022. godine, 12.06.2022. godine), nakon implementacija novih funkcionalnosti .

2.2.7 Isporuka:

- **Cilj:** Instalacija i obuka korisnika
- **Ucesnici:** Marko Savic i Petar Kresoja
- **Vremenski rok :** 21.05.2022.
- **Napomena**: Sa obzirom da se radi o faznom razvoju, isporuka se sprovodi (30.05.2022. godine, 06.06.2022. godine, 13.06.2022. godine), nakon uspesnog testiranja novih funkcionalnosti .

3.Analiza zahteva

Zadatak ovog koraka jeste intenzivna saradnja sa naruciocima, a posebno sa korisnicima softvera, u cilju formiranja skupa zahteva – definicije i specifikacije.

3.1 Prikupljanje zahteva

3.1.1 Definisanje korisnika sistema, njihovih uloga I aktivnosti

3.1.1.1 Upotreba aplikacije namenjena je sledećim korisnicima:

1. Autentifikovani korisnik / vlasnik vozila / *customer*
2. Zaposleni - administrator / *admin*
3. Zaposleni – računovodstvo i finansije / *accounting and finance*
4. Zaposleni – serviser / *servicer*
5. Zaposleni – prijem i obrada / *reception and processing*

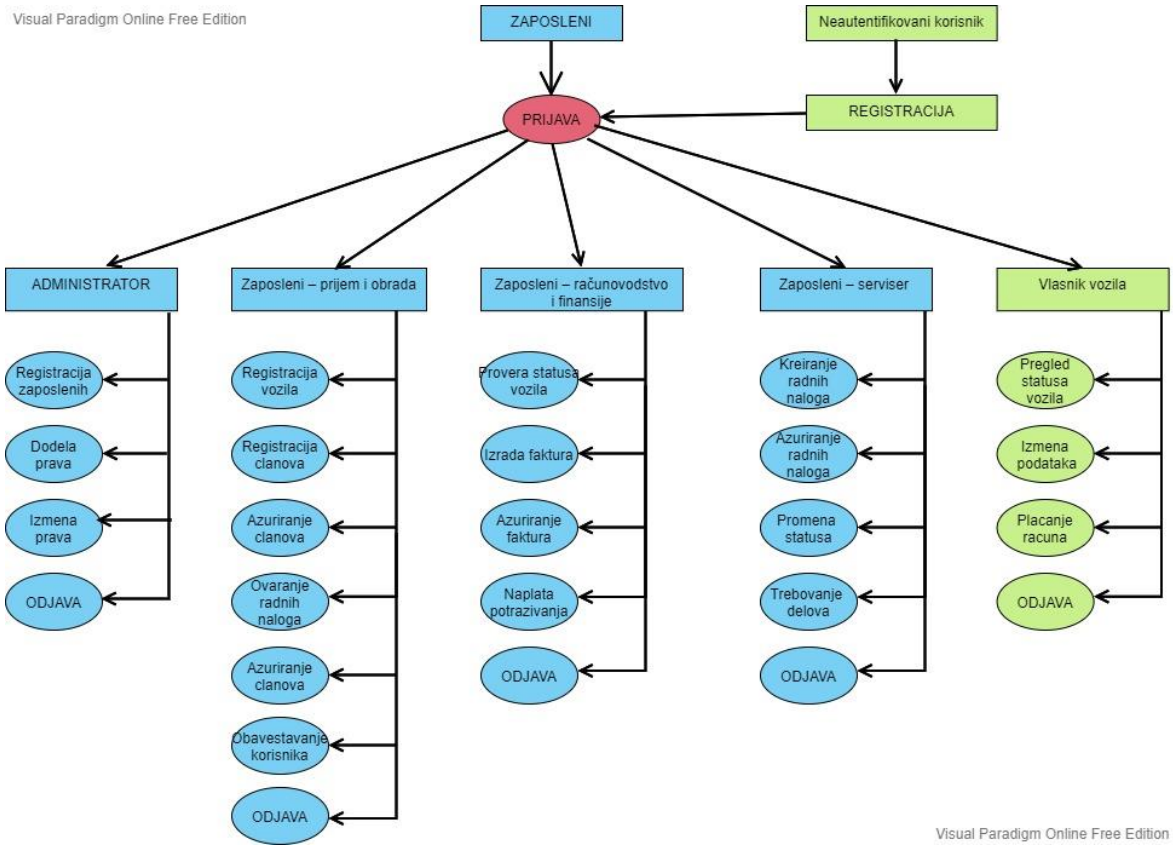
3.1.1.2 Aktivnosti:

1. Registracija vlasnika vozila (vrši zaposleni na prijemu i obradi zahteva);
2. Registracija vozila u sistemu (vrši zaposleni na prijemu i obradi zahteva);
3. Dostava vozila u servisnu garazu (vozilo prima dodeljeni serviser);
4. Serviser vrsi procenu, utvrđuje potrebne popravke i delove.Menja status vozila i dodaje neophodne popravke na vozilu, trebuje delove, vrši popravku;
5. Zaposleni na prijemu proverava magacin, vrsi nabavku i isporuku delova serviseru, azurira magacin, azurira podatke o kupcu, vrši aktivnosti usmerene ka kupcu(obaveštenja, rokovi, plaćanja, promene, dodavanja, uklanjanja, pruzanje informacija);
6. Zaposleni u finansijama proveravaju statuse popravki, završene radne naloge fakturišu, vrše naplatu potraživanja;
7. Vlasnik proverava status svog vozila putem broja sasije svog vozila na prijavnoj stranici aplikacije, proverava svoja zaduženja, vrši plaćanje;
8. Administrator pruža podršku u radu, vrši promenu uloga korisnika aplikacije, vrši promene i postavke u domenu autentifikacije i autorizacije korisnika;

3.1.2 Funkcionalni zahtevi, opis prema ulogama

1. **Autentifikovani korisnik / vlasnik vozila / *customer* -** predaje zahtev za registraciju radi evidencije jednog ili više svojih vozila koja su u kvaru.
2. **Zaposleni – prijem i obrada**, ga informiše o proceduri i upućuje da popuni upitnik za registraciju.
3. **Vlasnik vozila / *customer*** -popunjava zahtev svojim licnim podacima i podacima o vozilu ili vozilima.
4. **Vlasnik vozila / *customer*** -predaje zahtev za registraciju radi evidencije jednog ili više svojih vozila koja su u kvaru.
5. **Zaposleni – prijem i obrada -** registruje vozilo.Registracija vozila se sastoji iz unosa podataka u sistem, otvaranja radnog naloga i dodeljivanja slobodnog **servisera** radnom nalogu.
6. **Autentifikovani korisnik / vlasnik vozila / *customer*** – (nakon predhodnih koraka), moze unosom šasije u odgovarajuće polje u aplikaciji da izvrši pregled statusa svog vozila, pregled radnih naloga, plaćanje...
7. **Zaposleni – serviser / *servicer*** , vrši prijem vozila, pregled vozila, otvara dodatne radne naloge, trebuje delove iz magacina, menja status radnih naloga njemu dodeljenih vozila.
8. **Zaposleni – računovodstvo i finansije / *accounting and finance*** vrši fakturisanje radnih naloga koji se nalaze u statusu završeno.
9. **Zaposleni – prijem i obrada,** obaveštava korisnika, kada je vozilo završeno i vrši proveru naplate po odgovarajućim fakturama.
10. ...

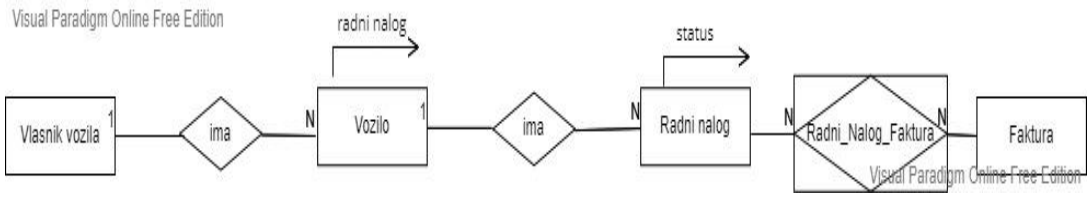
3.1.3 Funkcionalni zahtevi - *Dijagram funkcionalnih zahteva*



3.2 Modelovanje ponašanja

3.2.1 ER diagram

Vlasnik vozila može da ima jedno ili više vozila na servisu.Jedno vozilo može imati jedan ili više radnih naloga u slučaju od težine kvara.Jedan ili više radnih naloga može se nalaziti u okviru jedne ili više fakturi.



3.3 Zahtevi u pogledu kvaliteta

3.3.1 Performanse sistema (Brzina pristupa i dostupnost):

- Optimizacija aplikacije u cilju ostvarivanja podrške na većini internet pregledača, koji se najčešće koriste, uključujući i manje korišćene pretraživače;
- Optimizacija HTTP zahteva i upotreba keširanja podataka u cilju smanjenja opterećenja hardverskih resursa;
- Korišćenje modernih serverskih tehnologija, takodje utiče na brzinu pristupa što u paketu sa gore navedenim se odražava na zadovoljstvo korisnika, a isto tako ostvaruje značajne vremenske uštede na strani zaposlenih;

3.4 Projektna ogranicenja

3.4.1 Interakcija sa okruženjem:

- Administrator ima pravo kreiranja zaposlenih, dodele uloga, a time i ograničenja prava pristupa.
- Aplikacija treba da omogući logovanje zaposlenih na sistem, iz spoljašnjeg okruženja, putem username-a i password-a, što se odnosi i na administratora.
- Sistem bi bilo potrebno podesiti tako da svaki zaposleni, prema svojoj ulozi ima svoj domen pristupa.To znači da odredjene opcije koje nisu dozvoljene odredjenim korisnicima, nece ni biti vidljive.
- Vlasnik vozila može pristupiti samo unosom broja šasije svog vozila u aplikaciji za pregled vozila.Ukoliko vlasnik nije registrovan i nije registrovao vozilo u servisu, to vozilo nece biti prikazano.Nakon registracije u auto servisu vozilo se evidentira i za njega otvara radni nalog.

4.Projektovanje sitema

Zadatak ovog koraka jeste generisanje projekta sistema na osnovu definicije i specifikacije zahteva,

4.1 Arhitektura sistema

Aplikaciju je potrebno razviti po modelu klijent-server, troslojne arhitekture.U ovom modelu arhitekture princip rada se zasniva na serverskoj komponenti koja pruza usluge vecem broju klijentskih komponenti.Komponente mogu da imaju ulogu klijenta, servera i servera - klijenta.

Izbor programskih jezika se sveo na odabir sledecih tehnologija:

- Baza podataka: MySQL
- Serverski deo aplikacije: Java Spring
- Klijentski deo aplikacije : HTML5, CSS3, JS, Bootstrap

Baza podataka je smestena na serveru i time, u ovoj arhitekturi predstavlja serversku komponentu (upitima, sa klijentske strane, koji u ovaom slucaju predstavljaju backend aplikacije, dobijamo podatke koji su zahtevani).

Backend- JAVA Spring se ponasa kao klijent kada vrši upite nad bazom i zahteva odgovarajuće podatke, a isto tako i kao server kada vrši obradu podataka i šalje odgovor na korisnički deo aplikacije(frontend).Sa obzirom na prisutnost JAVA programskog jezika, ovaj deo aplikacije razvijan je prema pravilima i objektno-orijentisanog pristupa.

Korisnički deo aplikacije(frontend) se ponaša kao klijent.

Sa dijagrama u nastvku se vidi se vidi ilustrativni primer logike arhitekture.

Ilustracija 4.1a- Model klijent server arhitekture

4.2 Baza

Baza će sadržati 14 (četrnaest) tabela i dve asocijativne.

4.2.1 Pristup i ograničenja prema ulogama:

- Autentifikovani korisnik / vlasnik vozila / *customer*
  - `vozila`, `radni_nalog`, `klijenti`, `slike`, `faktura`;
- Zaposleni - administrator / *admin*
  - Pristup svim tabelama
- Zaposleni – računovodstvo i finansije / *accounting and finance*
  - `vozila`, `radni_nalog`, `klijenti`, `faktura`;
- Zaposleni – serviser / *servicer*
  - `vozila`, `radni_nalog`, `delovi_magacin`, `tip_prenosa`, `vrsta_goriva`;
- Zaposleni – prijem i obrada / *reception and processing*
  - `vozila`, `radni_nalog`, `delovi_magacin`, ` klijenti `, `fakture`;

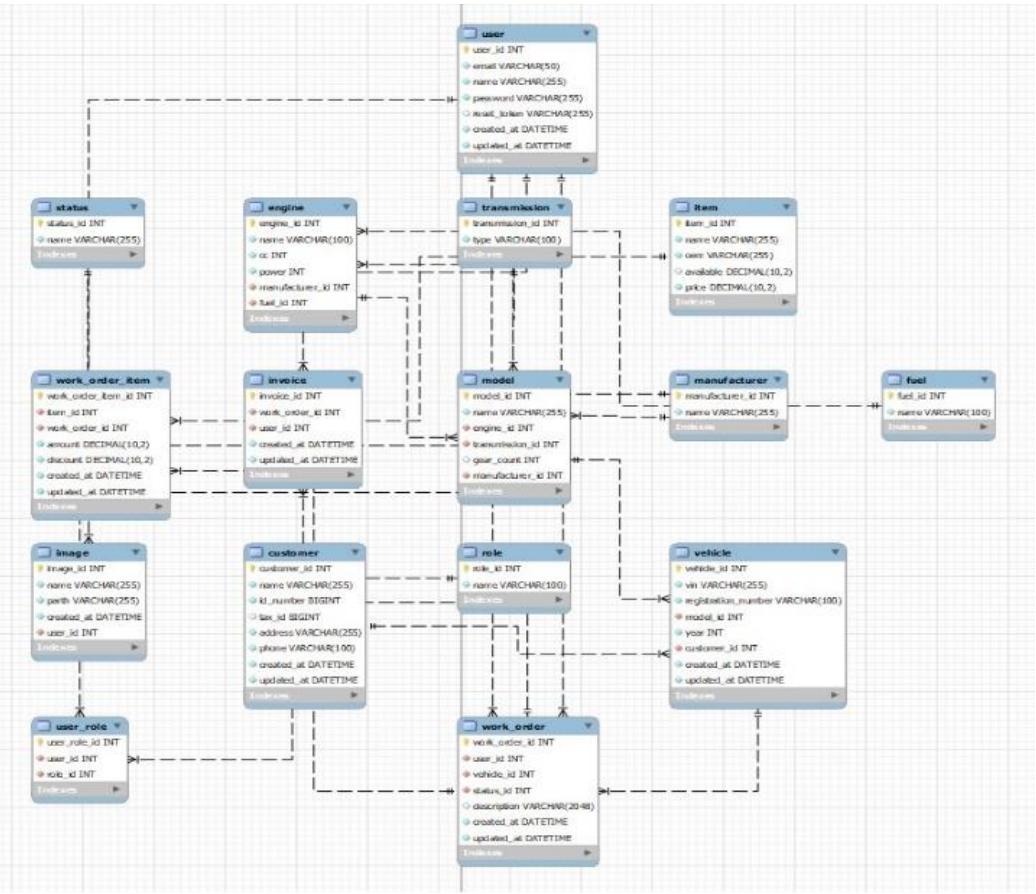
4.2.2 Entiteti baze i njihove relacije

1 . Customer
   - customer_id - *Jedinstveni id korisnika usluga autoservisa INT*
   - name - *Ime korisnika VARCHAR*
   - id_number - *Identifikacioni broj korisnika(primer: licna karta korisnika) BIG INT*
   - tax_id - *Broj tekuceg racuna korisnika BIG INT*
   - address - *Adresa korisnika VARCHAR*
   - phone - *Broj telefona korisnika VARCHAR*
   - created_at - *Datum i vreme kada je korisnik prvi put koristio usluge servisa DATETIME*
   - updated_at - *Datum I vreme promene odredjenih podataka korisnika DATETIME*

2. Engine
   - engine_id - *Jedinstveni id motora, pod kojim se void u tabeli(ne broj motora*) *INT*
   - name - *Naziv motora VARCHAR*
   - cc - *Kubikaza motora INT*
   - power - *Snaga motota INT*
   - manufacturer_id - *Id proizvodjaca motora, u u vezi (1:n) sa tabelom manufacturer preko manufacturer_id INT*
   - fuel_id - *Id goriva koji koristi automobil, u vezi (1:n) sa tabelom fuel preko polja/kolone fuel_id INT*

3 Fuel
4 Image
5 Invoice
6 Item
7 Manufacturer
8 Model
9 Role
10 Status
11 Transmission
12 User
13 User_role (asocijativna tabela – veza n:m , izmedju kolona user I role)
14 Vehicle
15 Work_order
16 Work_order_item (asocijativna tabela)

4.2.3 Model baze



Ilustracija 4.2.4 – model baze

4.3 Korisnicki deo aplikacije(backend)

4.3.1 Zahtevana struktura aplikacije

- public direktorijum poseduje *frontend*;
- src/main poseduje kod *backend*-a;
- src/test poseduje testove *backend*-a;

4.3.2 Organizacija koda aplikacije

4.3.2.1 Zahtevana struktura

- **rs.ac.singidunum.autoservis** je osnovni paket i u njemu se nalazi klasa sa main metodom
- **config paket** u sebi sadrzi bezbednosnu konfiguraciju. Klasa SecurityConfig.java proverava svaki upit ka webserver-u kako bi proverila dozvole korinsika
- **controller paket** sadrzi sve klase koje prihvataju komunikaciju putem http porokola.
- **domain paket** sadrzi entiete baze podataka sa tacno definisanim ogranicenjima.
- **error paket** sadrzi logiku za upravljanje sa izuzetcima.
- **model paket** sadrzi klase koje se koriste za komunikaciju izmedju front enda i backend-a
- **repository paket** poseduje interfejse koji upravljaju upitima nad bazom podataka
- **service paket** poseduje klase koje upravljaju logikom

4.3.2.2 Tehnologije / paketi

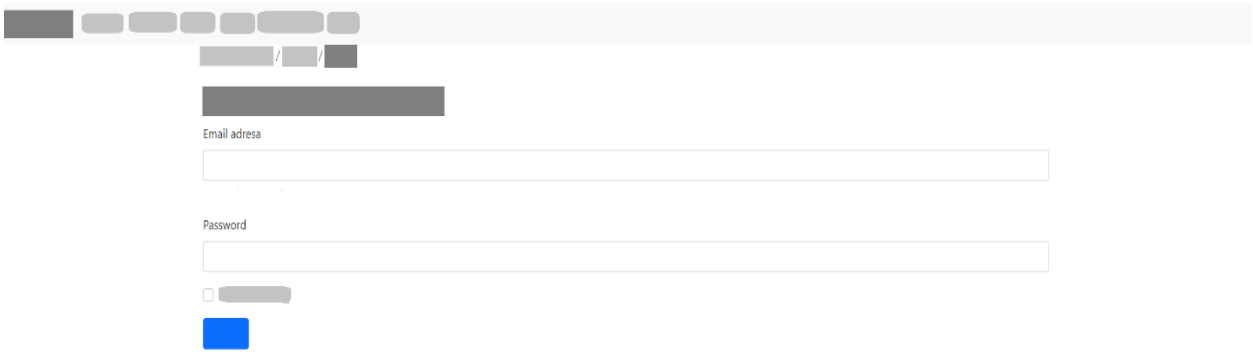
Backend: Spring Boot uz pomoc biblioteka:

- Java 11 kako bi se omogucila maksimalna bezbednost i efikasnost
- Spring Boot Web koja omogucava http komunikaciju
- Spring Security koji omogucava autentifkaciju i autorizaciju
- JWT Tokeni koriste za autentifkaciju i autorizaciju korinsika
- Java Mail Sender omogucava slanje elektronskih poruka
- Spring Data JPA olaksava rad za bazom podataka
- Lombook u velikoj kolicni smanjuje pisanje boilerplate koda

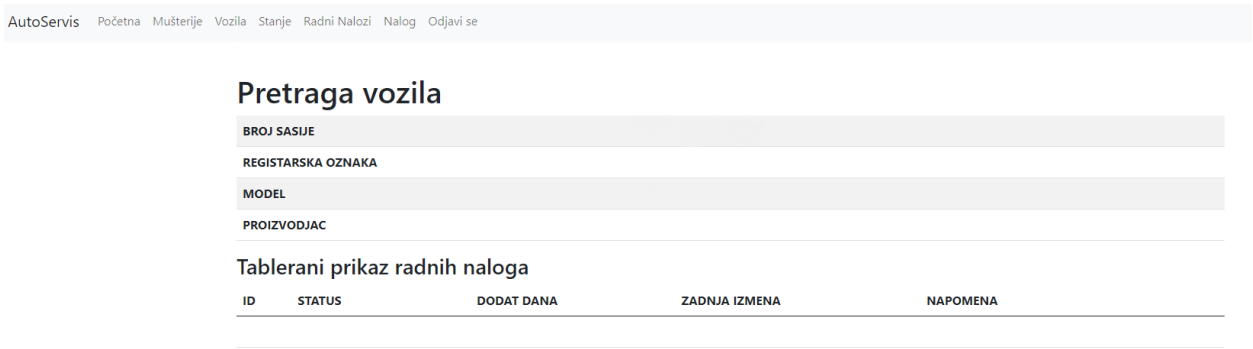
4.4 Klijentski deo aplikacije(frontend)

4.4.1 Strukturni zahtevi

- index.html -> pocetna stranica, predstavlja web prezentaciju auto servsa
- vehicle.html -> stranica predstavlja uvid u popravke izvrsene na motornom vozilu prema boju sasije, nije potrebna autentifikacija
- js/main.js -> glavni javascript fajl, poseduje sve zajednicke i servisne metode potrebene za funkcionisanje aplikacije
- css/styles.css -> poseduje stilove aplikacije
- img -> slike koriscene u aplikaciji
- assets/boostrap -> biblioteka za stilzovanje, boostrap; sve ostale biblioteke moraj se nalaziti u assets direktorijumu
- app -> glavni direktorijum aplikacije, interakcije sa stranicama u njemu potrebno je da je korinsik autentifkovan i autorizovan



Ilustracija 4.4.2a – Skica korisnickog interfejsa



Ilustracija 4.4.2b – Skica korisnickog interfejsa

5.UML modelovanje

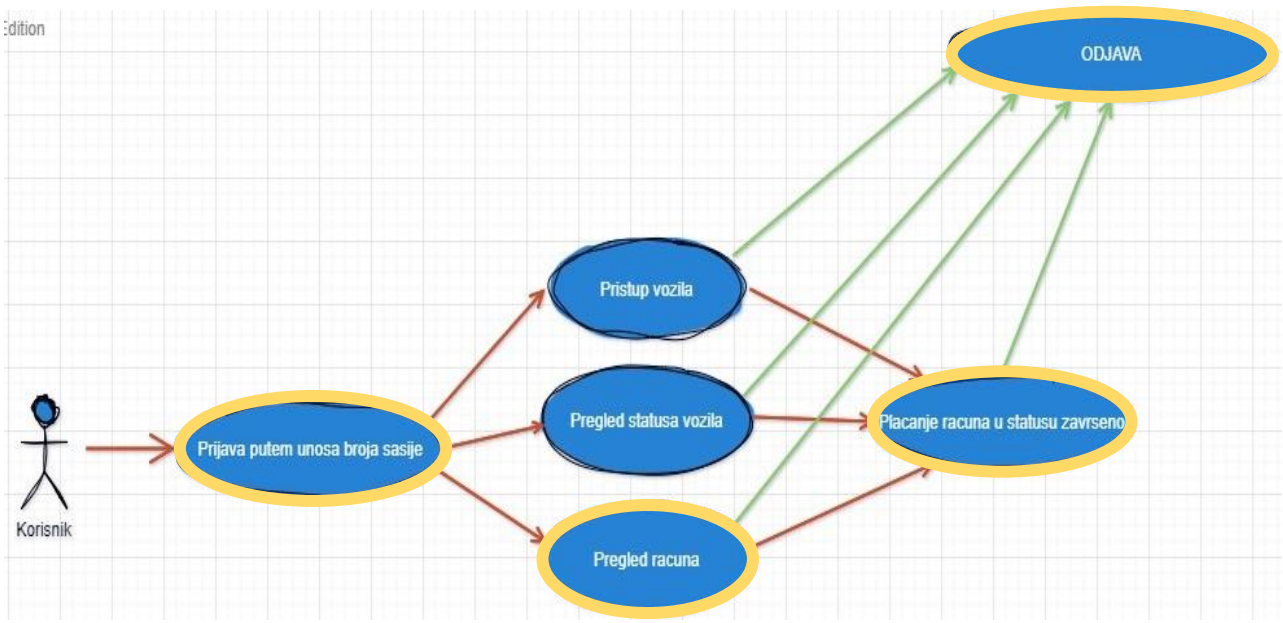
Zadatak ovog koraka jeste generisanja dijagrama slucajeva koriscenja sistema, prema ulogama, koji oslikava osnovne elemente i nacine njihovog generisanja.

5.1 Pregled slučajeva korišćenja prema ulogama

5.1.1 Autentifikovani korisnik / vlasnik vozila / *customer – Dijagram slucajeva koriscenja*

Primer slucaja korišćenja:

Vlasnik vozila se loguje na sistem, tj unosi broj šasije svog vozila.Pristupa informacijama o svom vozilu, vrši pregled radnih naloga i na osnovu statusa završeno, pristupa opciji plaćanja računa.Nakon plaćanja, vrši odjavu iz sistema, ukoliko je završio svoj rad u sistemu.

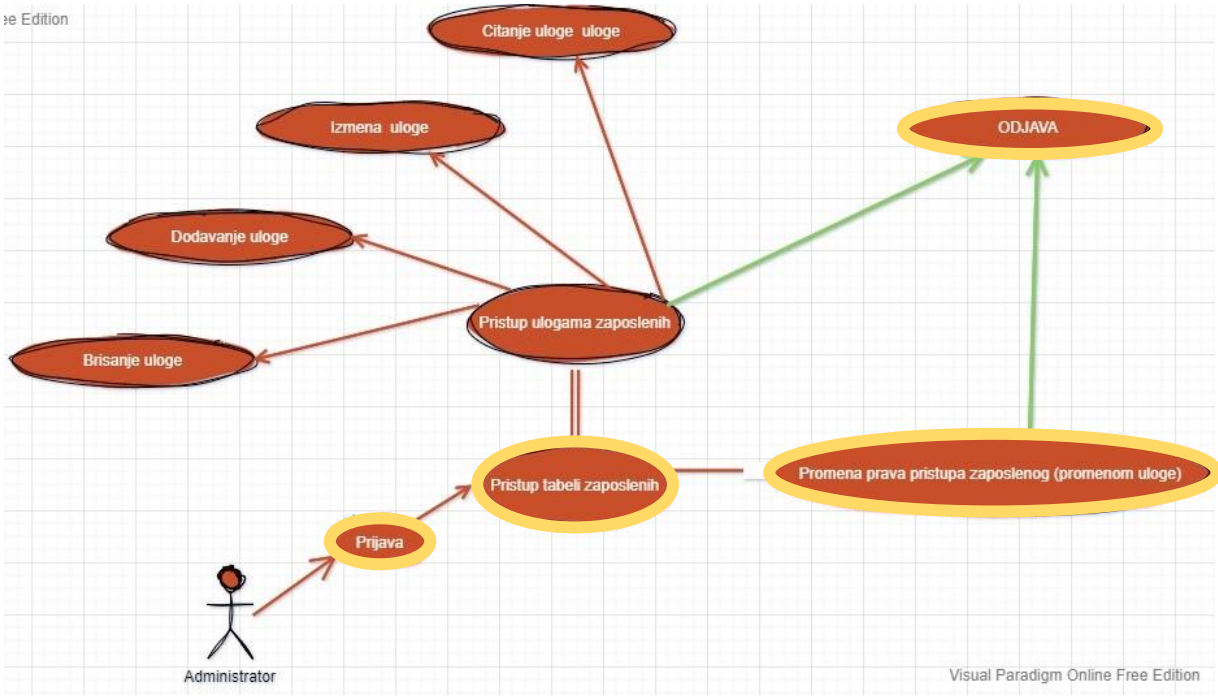


Ilustaracija 5.1.1a - slucaj koriscenja na dijagramu slucajeva koriscenja– Vlasnik vozila

5.1.2 Zaposleni - administrator / admin – Dijagram slucajeva koriscenja

Primer slucaja korišćenja:

Administrator se se loguje na sistem, tj. unosi svoj *username* i *password*.Pristupa administratorskom panelu i tabeli zaposlenih.Dodaje odgovarajuća prava pristupa odabranom zaposlenom entitetu, ukoliko se radi o novozaposlenom.Ukoliko se radi o promeni radnog mesta lica na drugu poziciju, vrši ažuriranje uloge na osnovu prava pristupa za odgovarajuću poziciju u odredjenoj direkciji/sektoru.Nakon plaćanja, vrši odjavu iz sistema, ukoliko je završio svoj rad u sistemu.



Ilustaracija 5.1.2a - slucaj koriscenja na dijagramu slucajeva koriscenja – Zaposleni-administrator

6.Implementacija softvera

Zadatak ovog koraka jeste realizacija programskog koda prema predhodno generisanim koracima, a u cilju dobijanja rezultata u vidu skupa programa koji ispunjavaju zadate funkcionalne zahteve.

6.1 Korak 1

Planirano je da po sablonu bude dokumentovan kod unutar aplikacije u vidu komentara

Zaglavlje unutar svakog fajla, treba na početku da sadrži opis koda koji sledi, ime autora, datum poslednje_izmene.

Primer: ***/* Komponenta: Custromer controller Autor : Petar Kresoja Datum poslednje izmene: 10.06.2022. */***

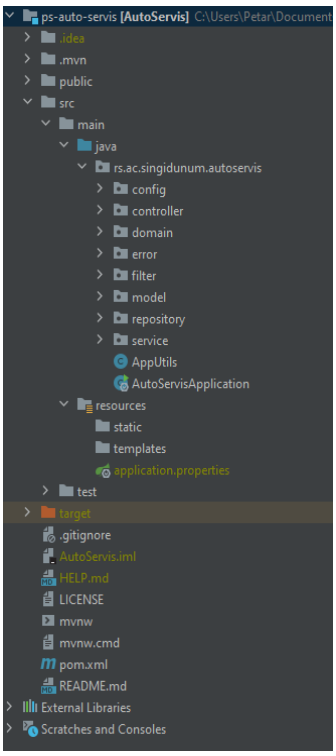
Dalji komentari u kodu sadrze nedvosmislene i jednostavne opise metoda ili promenljivih iznad kojih se nalaze, kako bi omogucili lako razumevanje onome ko dodje u situaciju da ima potrebu da radi nesto na ovom kodu ili da ga upotrebi u neke druge svrhe.(Odrzavanja, izmene, razvoja...)

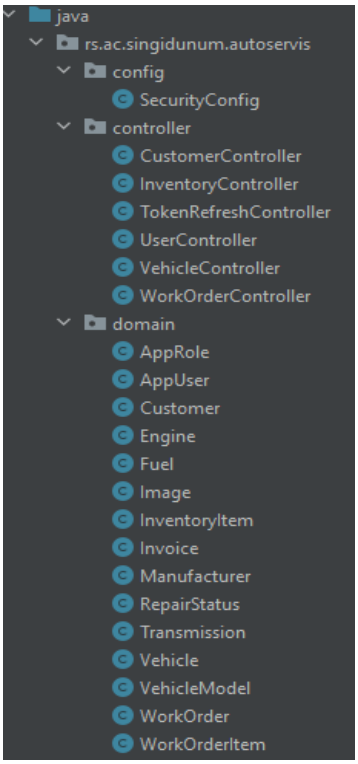
```
/*
Komponenta: Pomoćna servisna klasa
Autor : Petar Kresoja
Datum poslednje izmene: 10.06.2022.
*/
const baseUrl = "http://localhost:8080/api"

// Dynamic menu
const session = window.localStorage.getItem( key: "name");
if (session && !window.location.href.includes("logout.html")) {
    document.querySelectorAll( selectors: ".action-logout").forEach( callbackfn: e => e.hidden = false);
    document.querySelectorAll( selectors: ".action-login").forEach( callbackfn: e => e.hidden = true);
}

function alertError(e) {
    alert(e);
    window.location.href = `/index.html?from=${window.location.href}&reason=${e.message}`;
}
```

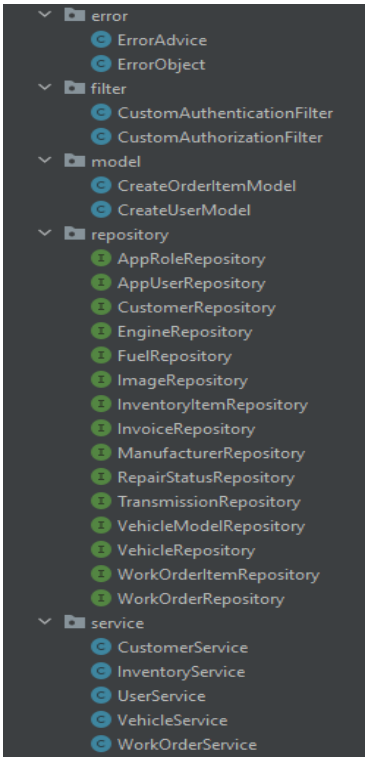
Ilustracija 6.1 – Sablon po kojem ce biti pisana unutrasnja dokumentacija

6.2 Korak 2

Implementirana struktura podataka i sadrzaj struktura, prema zahtevima iz analize i specifikacije zahteva, odgovara projektnom planu iz faze projektovanja sistema.







Ilustracija 6.2 – struktura i sadrzaj componenata

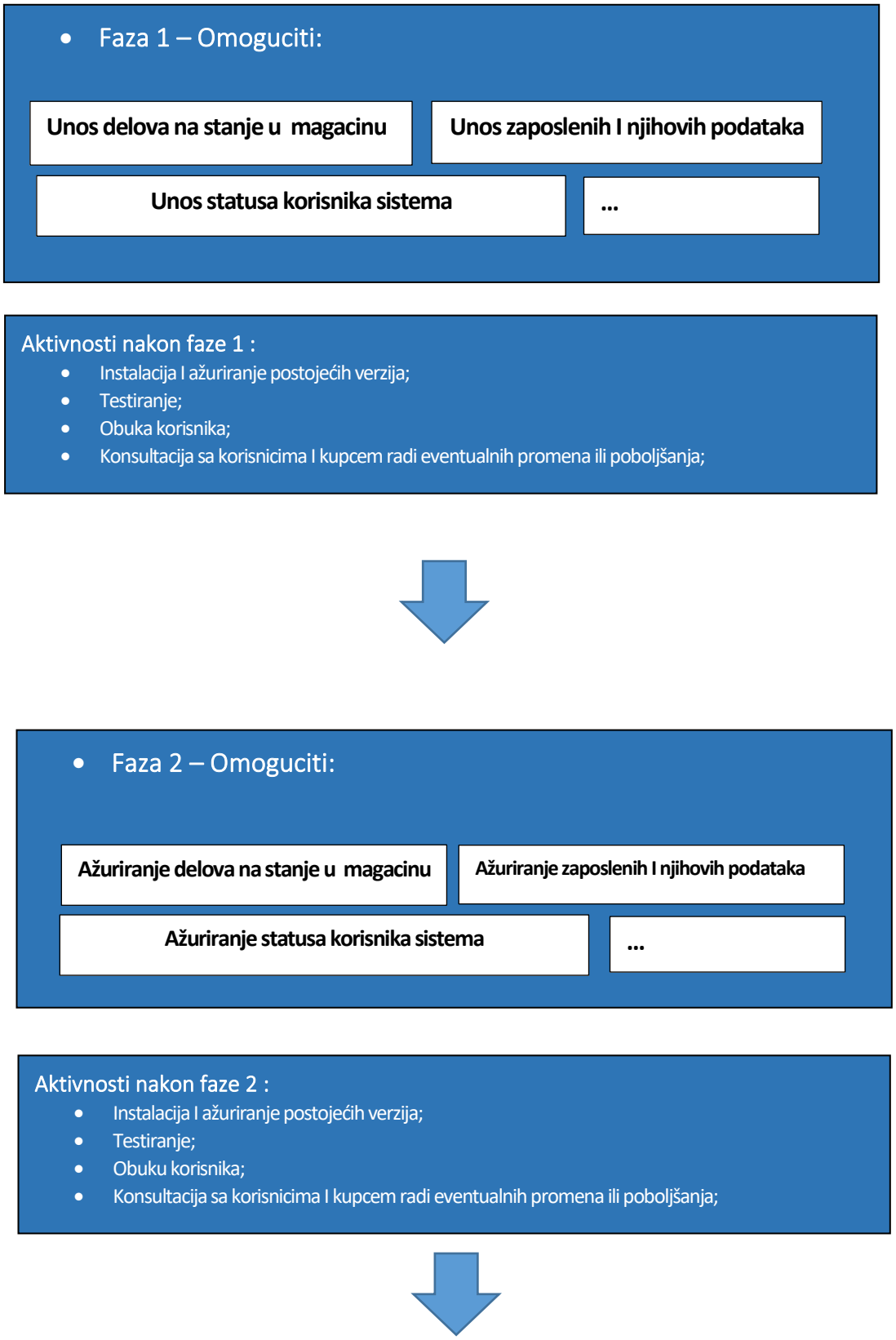
6.3 Korak 3

Isporuka operativnih funkcija po fazama

- **Faza 1 – Omoguciti:**
  - unos delova na stanje u magacinu;
  - unos zaposlenih I njihovih podataka;
  - unos statusa korisnika sistema;
  - **...**
- **Faza 2 – Omoguciti:**
  - ažuriranje delova u stanju u magacinu;
  - ažuriranje zaposlenih I njihovih podataka;
  - ažuriranje statusa korisnika sistema;
  - ...

Aktivnosti nakon svake faze:

- Instalacija I ažuriranje postojećih verzija;
- Testiranje;
- Obuka korisnika;
- Konsultacija sa korisnicima I kupcem radi eventualnih promena ili poboljšanja;



Ilustracija 6.3 – Ciklus implementacije faznog modela – iterativni pristup

6.Testiranje

Zadatak ovog koraka jeste veoma znacajan, jer nakon ovog koraka sledi korak isporuke softvera naruciocima.Cilj ovog koraka, jeste pronaci i ukloniti sto veci broj gresaka, a samim tim i otkaza sistema. Klasa UserControllerTest sadrži testove logike dodavanja novog korisnika u bazu podataka.

6.1 Koraci u testiranju

6.1.1 Jedinično testiranje

- Jediničan test provere dužine lozinke samo proverava logiku u okviru metode UserService#createUser.Od aplikacija je ocekivano da odgovori sa izuzetkom tipa RuntimeException i porukom da lozinka mora imati minimalno 8 karaktera
- Jediničan test provere ispravnosti adrese elektronske pošte samo proverava logiku u okviru metode UserService#createUser.Od aplikacija je ocekivano da odgovori sa izuzetkom tipa RuntimeException i porukom da adresa el. pošte nema pravilan format

```
/*
Jedinicno testiranje
Ocekivani odgovor: OK
 */
@Test
void sendHeartbeat() {
    HeartBeatController controller = new HeartBeatController();
     String response = controller.sendHeartbeat();
     Assertions.assertEquals( expected: "OK", response);
}
```

6.1.2 Integraciono testiranje

Prilikom pokretanja integracionog testa u metodu UserServiceTest#findUserByIdFromDataBase koristili smo simuliranu bazu podataka.Ona se nalazi u klasi MockUserRepository.Metod UserService#getUserFromPrincipal uzima email adresu (odnosno korisnicko ime) iz objekta klase Principal koja dolazi iz modula Spring Security.Nakon preuzimanja imena iz Principal#getName logika povlaci korisnika iz base podataka, tada proverama da li je ime korisnika isto kao ocekivano. Test se uspesno iyvrsava

```java
/*
Integracioni test pronalazenja korisnika iz baze podataka
Napomena: Baza podataka je simulirana u MockUserRepository#findById
Ulazni paramteri: ppetrovic@gmail.com
Ocekivan izlaz: Petar Petrovic
 */
@Test
void findUserByIdFromDataBase() {
    String input = "ppetrovic@gmail.com";
    String expected = "Petar Petrovic";

    // Konstrukcija objekta interfejsa Principal
    // Principal je integrisan korisnik u spring okruzenju
    // Koji nastaje autentifkacijom modula Spring Security
    AppUser output = service.getUserFromPrincipal(() -> input);
    Assertions.assertEquals(expected, output.getName());
}
```

6.1.3 Sistemsko testiranje

Neophodne stavke prilikom testiranja su:

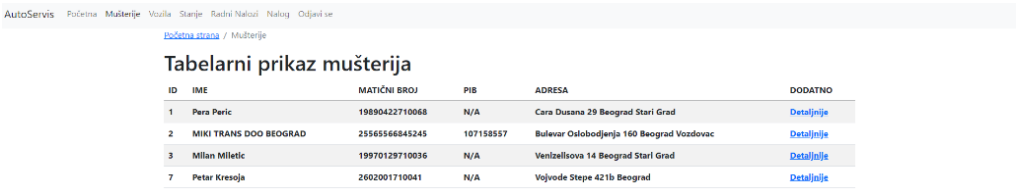
- Pokretanje aplikacije iz internet pregledača
- Funkcije prijave (zaposlenih, vlasnika vozila putem broja šasije);
- Funkcije kreiranja, čitanja, ažuriranja i brisanja sadržaja prema ulogama i pravima pristupa
- Funkcije podrške administratora
- Odjava

7.Isporuka

Aplikacija je prilagodjena tako da zahteva minimalanu kolicinu resursa. Velika paznja je posvecena kompatibilnosti sa svim vrstava hosting provajdera. Preporucen nacin isporuke je upotrebom VPS server-a kako bi se omogucila najveca stabilnost i jednostavnost u radu. Hardervski zahtevi su 2GB radne memorije i jedno procesorsko jezgro. Softwer je optimizovan za Ubuntu Server operativni sistem i kao takav predvidjen da radi sa Apache2 web serverom

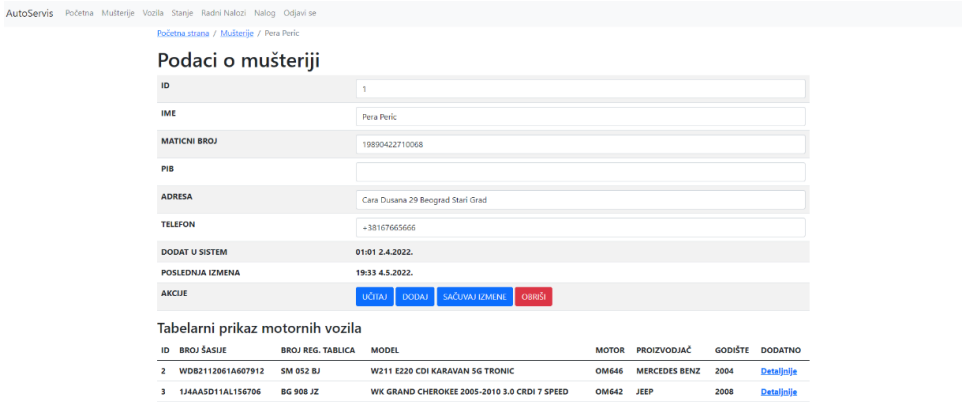
7.1 Uputstvo za prijavljivanje na aplikaciju po koracima:

- Pristupiti linku putem web-pretrazivaca ink za pristup: https://ps.pequla.com/
- Ulogovati se na stranici https://ps.pequla.com/app/user/login.html?return=/index.html
- Zapoceti sesiiju rada



Ilustracija 7.1a – prikaz korisnickog interfejsa

Ilustracija 7.1b – prikaz korisnickog interfejsa



Ilustracija 7.1c – prikaz korisnickog interfejsa

Hvala na pažnji