Programsko inženjerstvo ak. god 2025./2026

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Škoda Autoservis - Pouzdana aplikacija za autoservis

Tim: TG05.2

Ime tima: Havana

Nastavnik: Miljenko Krhen

Asistent: Karlo Bašić

Opis projektnog zadatka

1.1 Uvod i cilj projekta

Današnje poslovanje autoservisa suočeno je s potrebom digitalne transformacije i povećanja učinkovitosti u upravljanju svakodnevnim procesima. U praksi, veliki broj servisa još uvijek koristi zastarjele metode evidencije — ručne upisnike, tablice ili parcijalne aplikacije koje nisu međusobno povezane. Takvi sustavi otežavaju praćenje statusa popravaka, rezervaciju zamjenskih vozila, komunikaciju s korisnicima i analizu opterećenja servisa.

Cilj ovog projekta je razviti web aplikaciju za upravljanje radom autoservisa koja objedinjuje sve osnovne procese u jedan intuitivan i automatiziran sustav.

Aplikacija omogućava:

- jednostavnu prijavu i evidenciju vozila
- upravljanje terminima i resursima
- praćenje statusa popravaka u stvarnom vremenu
- automatsko generiranje i pohranu dokumenata (PDF obrazaca)
- transparentnu komunikaciju između korisnika i servisnog osoblja

Time se želi postići povećanje produktivnosti, preciznost evidencija i poboljšano korisničko iskustvo kroz digitalnu podršku cjelokupnom servisnom procesu — od prijave vozila do preuzimanja nakon popravka.

1.2. Problematika i postojeće stanje

Većina manjih i srednjih autoservisa nema razvijen informatički sustav koji bi podržao digitalnu prijavu vozila, praćenje statusa popravka i online komunikaciju s korisnicima.

U praksi se i dalje koriste:

- papirnate prijemnice i servisne knjižice
- ručni unos termina u kalendare
- nedostatak centralizirane baze podataka o vozilima i servisima

Takav način rada dovodi do čestih pogrešaka u unosu i gubitka informacija, neusklađenosti između djelatnika servisa te dugog čekanja korisnika.

Aplikacija koju stvaramo rješava upravo te izazove stvaranjem centralnog informacijskog sustava koji:

1. Omogućuje digitalnu prijavu vozila i automatsko generiranje dokumentacije
2. Povezuje korisnike, servisere i administratore u jedinstven sustav
3. Pruža jasan pregled statusa, resursa i termina
4. Omogućuje jednostavno upravljanje zamjenskim vozilima

1.3. Potencijalna korist projekta

Implementacija ovakve aplikacije korisna je, kako za servis, tako i za krajnjeg korisnika:

1.3.1. Za servis

- Učinkovitije poslovanje: digitalizacijom procesa smanjuje se potreba za ručnim unosima i administracijom.
- Točnija evidencija: svi podaci o prijavi, statusu i preuzimanju vozila dostupni su u stvarnom vremenu.
- Bolje planiranje resursa: jednostavno praćenje zauzeća servisnih linija i raspoloživosti zamjenskih vozila.
- Statistički uvidi: generiranje izvješća o broju vozila i trajanju popravaka.
- Standardizacija procesa: svaki korak - od prijave do završetka naloga - provodi se na isti način.

1.3.2. Za korisnike

- Transparentnost: korisnik u svakom trenutku zna status svog vozila.
- Brža komunikacija: obavijesi i potvrde automatski stižu putem e-pošte.
- Praktičnost: online prijava 24/7 bez potvrde za fizičkim dolaskom.
- Digitalna dokumentacija: svi obrasci dostupni u elektroničkom obliku.

1.3.3. Za poslovanje u cjelini

- Veća konkurentnost servisa na tržištu kroz modernizaciju poslovanja.
- Povećano zadovoljstvo korisnika, što može voditi većoj lojalnosti i boljoj reputaciji.
- Mogućnost integracije s budućim sustavima (npr. centralnim bazama, aplikacijama proizvođača vozila, sustavima rezervnih dijelova itd.).

1.4. Postojeća slična rješenja

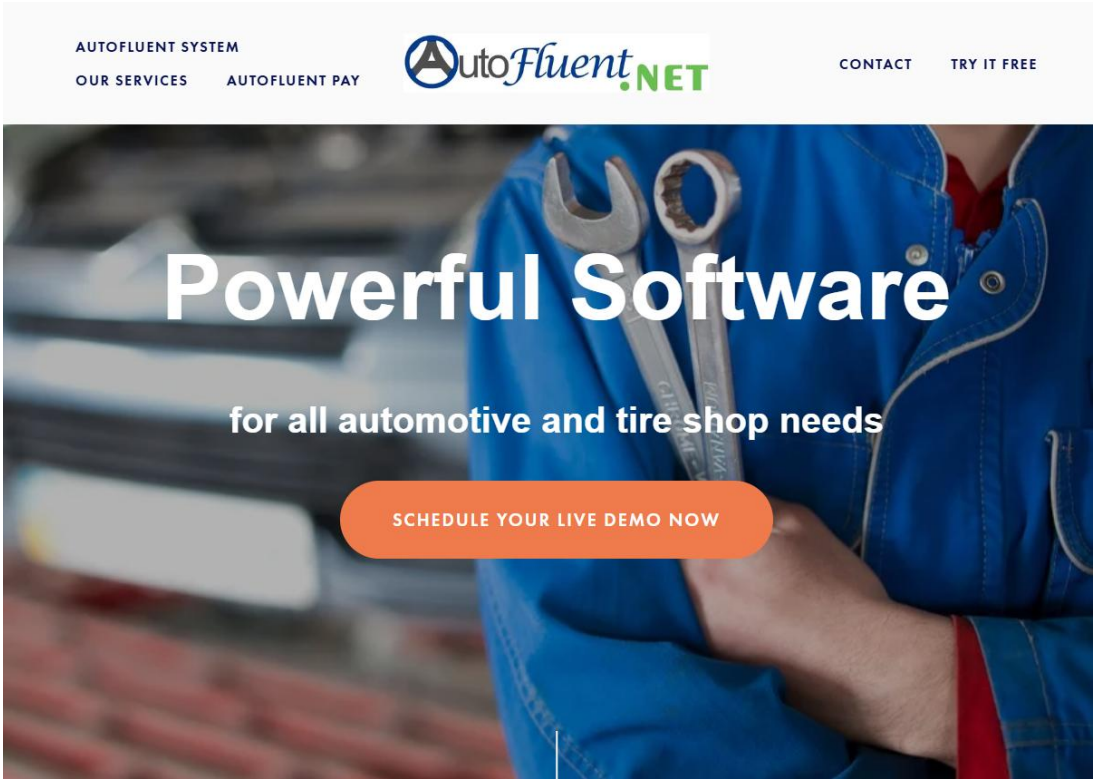
Na tržištu postoje različiti softveri za upravljanje autoservisima, no većina njih je komercijalna i zatvorenog tipa, te često ne odgovara specifičnim potrebama malih servisa.

U nastavku su navedeni primjeri nekih od najpoznatijih takvih softvera:

1.4.1. AutoFluent

Komercijalno rješenje koje nudi funkcionalnosti poput upravljanja zalihama, računa i servisnih naloga.

Razlike: orijentirano velikim servisnim mrežama i radionicama s integriranim financijskim modulima; složenije i zahtjevnije za implementaciju od predložene aplikacije.

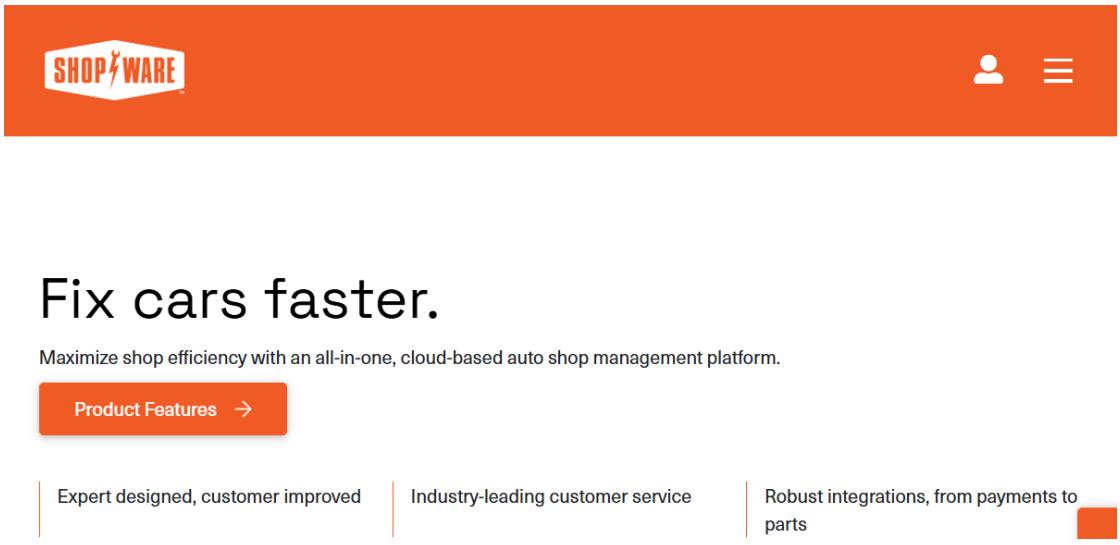


Snimka zaslona početne stranice AutoFuent

1.4.2. Shop-Ware

Cloud rješenje za servisne radionice s fokusom na digitalne procese i korisničku komunikaciju.

Razlike: vrlo napredno, ali pretplatnički model i visoka cijena; nema lokalizaciju ni prilagodbu za hrvatsko tržište.

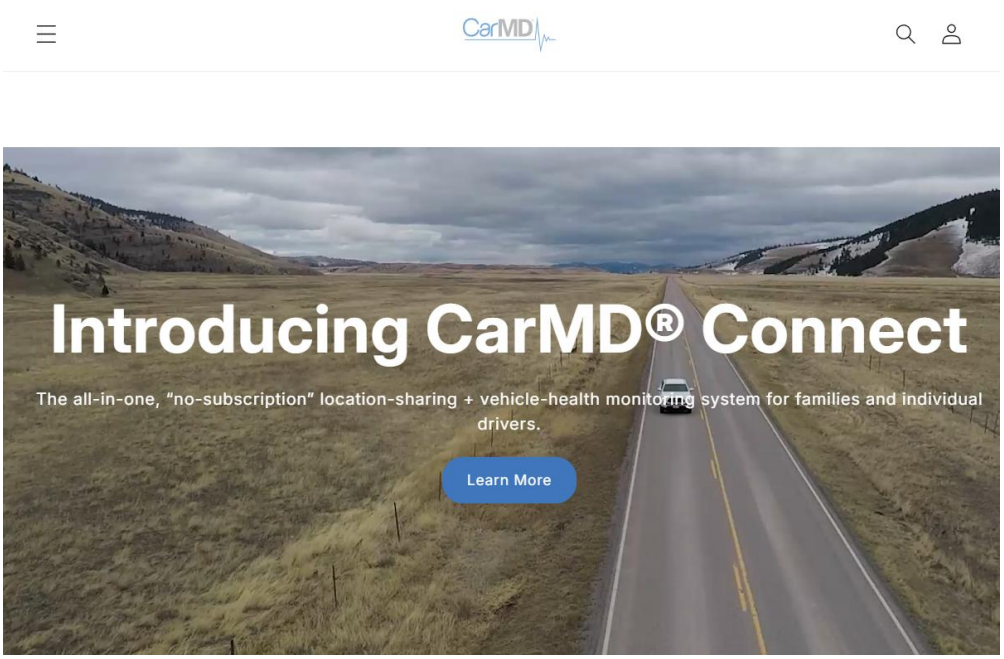


Snimka zaslona početne stranice Shop-ware

1.4.3. CarMD / AutoRepair Cloud

Pruža osnovne funkcionalnosti prijave vozila i statusa popravka.

Razlike: primarno orijentirano američkom tržištu, s integracijama prema tamošnjim sustavima dijagnostike; ne podržava dvojezično sučelje ni lokalne formate dokumenata (npr. potvrde u PDF-u prema hrvatskim propisima).



> Snimka zaslona početne stranice CarMD

Prednost predloženog rješenja: otvorenost, jednostavnost i prilagodljivost potrebama lokalnih servisa, te mogućnost daljnjeg razvoja i nadogradnje bez dodatnih troškova licenci.

1.5. Skup korisnika

Predviđeni korisnici aplikacije podijeljeni su u četiri glavne su:

Korisnička skupina	Opis	Primjeri
Korisnici servisa (vlasnici vozila)	Osobe koje putem aplikacije prijavljuju vozila, prate popravke i preuzimaju potvrde	Fizičke osobe, pravne osobe s voznim parkom
Ovlašteni serviseri	Zaposlenici servisa koji obrađuju naloge i ažuriraju statuse	Automehaničari, tehničari
Voditelj servisa	Osoba odgovorna za praćenje učinkovitosti i poslovnih rezultata	Uprava servisa ili voditelj poslovnice
Administrator	Održava aplikaciju, dodaje nove korisnike i resurse	IT podrška servisa

1.6. Mogućnost prilagodbe rješenja

Sustav je koncipiran kao modularna web aplikacija s mogućnošću kasnijeg proširenja.

Moguće prilagodbe uključuju:

- Integraciju s vanjskim sustavima (npr. ERP, CRM, sustavi proizvođača vozila)
- Dodavanje *mobilne* verzije (native aplikacija za Android/iOS)
- Višejezično sučelje i podršku za međunarodno tržište
- Uvođenje online plaćanja (kartice, mobilno bankarstvo) za usluge servisa
- Automatsku dijagnozu putem IoT uređaja i senzora u vozilima

Takva arhitektura omogućuje jednostavnu nadogradnju sustava bez potrebe za potpunom rekonstrukcijom.

1.7. Opseg projektnog zadatka

U prvoj fazi projekt obuhvaća:

- razvoj osnovnih modula: registracija, prijava, upravljanje vozilima, statusi, generiranje PDF-a;
- definiranje uloga korisnika (registrirani korisnik, serviser, administrator, voditelj);
- izradu responzivnog korisničkog sučelja (frontend);
- izradu backend servisa s integracijom na OAuth2 i Google Maps API;
- pohranu i preuzimanje dokumenata (PDF, XML, XLSX);
- postavljanje baze podataka i API sloja za sve glavne funkcije.

Fokus je na implementaciji ***glavnog poslovnog tijeka:***

> registracija -> prijava vozila -> servisni nalog -> status -> preuzimanje -> izvještaji

Izvan opseg prve faze (ali planirano za nadogradnju) su:

- financijski moduli (fakturiranje)
- komunikacija putem chata
- prediktivno planiranje kapaciteta (ML)

2. Analiza zahtjeva

> Ovdje ćemo definirati funkcionalne zahtjeve, nefunkcionalne zahtjeve i dionike/aktere.

2.1. Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-01	Pregled osnovnih podataka o servisu (naziv, radno vrijeme, lokacija, kontakt).	Visok	Dokument zahtjeva	Korisnik može vidjeti osnovne informacije o servisu i kontakt podatke bez prijave.
F-02	Unos podataka o vozilu (marka, model, registracija, opis).	Visok	Dokument zahtjeva	Podaci se validiraju i pohranjuju uz korisnika; vidljivi su u detaljima naloga.
F-02.1	Odabir servisera i slobodnog termina.	Visok	Dokument zahtjeva	Prikazuje se lista slobodnih termina; rezervacija uspijeva bez konflikta.
F-02.2	Provjera dostupnosti termina u realnom vremenu.	Visok	Dokument zahtjeva	Sustav provjerava zauzetost termina i sprječava dvostruke rezervacije.
F-02.3	Prikaz lokacije servisera putem Google Maps.	Visok	Dokument zahtjeva	Karta se prikazuje tek nakon odabira servisera i prikazuje točnu lokaciju.

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-02.4	Opcionalna rezervacija zamjenskog vozila.	Srednji	Dokument zahtjeva	Ako je dostupno, sustav pridružuje zamjensko vozilo.
F-03	Generiranje potvrde prijave u PDF formatu.	Srednji	Zahtjev dionika	Nakon kreiranja/završetka naloga generira se PDF i sprema uz prijavu.
F-03.1	Slanje PDF potvrde korisniku putem e-maila.	Srednji	Dokument zahtjeva	Nakon generiranja, potvrda se automatski šalje korisniku putem e-mail servisa.
F-04	Praćenje statusa popravka u stvarnom vremenu.	Visok	Zahtjev korisnika	Korisnik vidi statuse (zaprimljeno, u obradi, čekanje dijela, završeno).
F-05	Ažuriranje statusa/termina i napomena od strane servisera.	Visok	Interni zahtjev	Serviser može mijenjati status i dodavati napomene vidljive korisniku.
F-05.1	Automatsko obavještavanje korisnika o promjeni statusa/termina.	Visok	Dokument zahtjeva	Nakon svake promjene statusa, korisnik dobiva e-mail obavijest.

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-06	Administracija korisnika i resursa.	Visok	Interni zahtjev	Administrator može upravljati korisnicima i resursima koji se nalaze u sustavu ili i sam može nadodavati podatke u sustav.
F-06.1	Upravljanje korisnicima.	Srednji	Interni zahtjev	Administrator može upravljati korisnicima i njihovim podatcima.
F-06.2	Upravljanje terminima.	Srednji	Interni zahtjev	Administrator može mijenjati i upravljati terminima servisa u sustavu.
F-06.3	Upravljanje zamjenskim vozilima.	Srednji	Interni zahtjev	Administrator može nadodati novo zamjensko vozilo te može uređivati postojeća vozila.
F-07	Pregled statistika i dashboard za voditelja servisa.	Srednji	Zahtjev dionika	Voditelj vidi KPI-je (broj vozila, trajanja, zauzeće zamjenskih vozila).
F-08	Izvoz podataka u PDF, XML i XLSX.	Srednji	Dokument zahtjeva	Korisnik ili voditelj mogu preuzeti traženi izvoz bez pogreške.

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-09	Predaja vozila u autoservis.	Srednji	Dokument zahtjeva	Korisnik i serviser odrađuju predaju/preuzimanje vozila, korisnik i serviser dobivaju mail potvrdu i PDF.
F-10	Preuzimanje vozila iz autoservisa.	Srednji	Dokument zahtjeva	Serviser i korisnik odrađuju predaju/preuzimanje vozila, serviser i korisnik dobivaju mail potvrdu i PDF.

2.2. Nefunkcionalni zahtjevi

2.2.1. Zahtjevi performansi

ID zahtjeva	Opis zahtjeva	Prioritet
NF-P1	Sustav mora omogućiti učitavanje početne stranice u manje od 3 sekunde.	Visok
NF-P2	Generiranje PDF potvrda mora se izvršiti u roku kraćem od 5 sekundi.	Visok
NF-P3	Prikaz liste vozila i statusa popravka mora biti ažuran u realnom vremenu.	Visok

ID zahtjeva	Opis zahtjeva	Prioritet
NF-P4	Sustav mora podržati najmanje 100 istovremenih aktivnih korisnika bez značajnog pada performansi.	Srednji
NF-P5	Sustav mora osigurati vremenski odziv manji od 2 sekunde pri promjeni statusa popravka.	Srednji

2.2.2. Zahtjevi sigurnosti

ID zahtjeva	Opis zahtjeva	Prioritet
NF-S1	Autentifikacija korisnika mora se provoditi putem sigurnog OAuth2 protokola.	Visok
NF-S2	Komunikacija između klijenta i poslužitelja mora biti šifrirana (HTTPS).	Visok
NF-S3	Samo ovlašteni korisnici smiju pristupati zaštićenim dijelovima aplikacije.	Visok
NF-S4	Administrator jedini ima pravo dodavati, mijenjati i brisati korisničke uloge.	Visok
NF-S5	Sustav mora automatski odjaviti korisnika nakon 30 minuta neaktivnosti.	Srednji
NF-S6	Lozinke i tokeni ne smiju se pohranjivati u otvorenom obliku.	Visok
NF-S7	Sustav mora bilježiti sve administrativne akcije (audit log).	Srednji

2.2.3. Zahtjevi pouzdanosti

ID zahtjeva	Opis zahtjeva	Prioritet
NF-R1	Sustav mora biti dostupan korisnicima 24/7, osim u vrijeme planiranog održavanja.	Visok
NF-R2	Sustav mora automatski pohraniti sigurnosnu kopiju baze jednom dnevno.	Visok
NF-R3	U slučaju pogreške ili pada poslužitelja, sustav mora omogućiti povrat podataka iz posljednje sigurnosne kopije.	Visok
NF-R4	Sustav mora zadržati integritet podataka i u slučaju prekida veze s bazom.	Srednji
NF-R5	Korisnici moraju biti obaviješteni o eventualnim prekidima rada putem e-maila ili obavijesti u aplikaciji.	Nizak

2.2.4. Zahtjevi upotrebljivosti

ID zahtjeva	Opis zahtjeva	Prioritet
NF-U1	Korisničko sučelje mora biti jednostavno i intuitivno.	Visok
NF-U2	Aplikacija mora imati responzivan dizajn prilagođen za računala, tablete i mobilne uređaje.	Visok
NF-U3	Sve glavne funkcionalnosti moraju biti dostupne najviše u tri klika s početne stranice.	Srednji

ID zahtjeva	Opis zahtjeva	Prioritet
NF-U4	Korisniku se moraju prikazivati jasne poruke o uspješnim i neuspješnim radnjama.	Srednji
NF-U5	Tekst, tipke i ikone moraju biti jasno vidljivi i razumljivi za korisnike bez tehničkog predznanja.	Srednji

2.2.5. Zahtjevi prijenosa i razmjene podataka

ID zahtjeva	Opis zahtjeva	Prioritet
NF-D1	Sustav mora omogućiti izvoz podataka u formate PDF, XML i XLSX.	Visok
NF-D2	Sustav mora omogućiti uvoz i izvoz podataka o servisima u standardiziranom formatu.	Srednji
NF-D3	Sustav mora omogućiti prikaz točne lokacije servisa putem Google Maps servisa.	Visok
NF-D4	Sustav mora osigurati kompatibilnost s vanjskim sustavima putem REST API sučelja.	Srednji

2.2.6. Zahtjevi održavanja i održivosti

ID zahtjeva	Opis zahtjeva	Prioritet
NF-M1	Sustav mora biti izrađen modularno kako bi se omogućilo jednostavno dodavanje novih funkcionalnosti.	Visok

ID zahtjeva	Opis zahtjeva	Prioritet
NF-M2	Sav izvorni kôd mora biti dokumentiran u skladu s konvencijama jezika (JavaScript, Node.js).	Visok
NF-M3	Sustav mora biti popraćen korisničkom i tehničkom dokumentacijom.	Visok
NF-M4	Administratori moraju imati mogućnost nadzora i testiranja funkcionalnosti bez utjecaja na produkciju.	Srednji

2.3. Dionici i akteri

Dionici

Skupina	Opis
Korisnici	Krajnji korisnici aplikacije: registrirani korisnici koji prijavljuju vozila, prate status popravka i preuzimaju dokumente.
Administratori sustava	Ovlašteni djelatnici servisa i administratori koji upravljaju korisnicima, terminima i resursima.
Naručitelji	Organizacija koja koristi aplikaciju za digitalizaciju poslovanja autoservisa.
Razvojni tim	Programeri i dizajneri koji razvijaju, testiraju i održavaju sustav.
Vanjski sustavi	Povezani servisi kao što su Google Maps (lokacija), E-mail servis (notifikacije) i OAuth2 pružatelj (autentifikacija).

2.3.1. Aktori i njihovi funkcionalni zahtjevi

A-1 Registrirani korisnik (inicijator) može:

- Pregledati osnovne informacije o servisu (naziv, radno vrijeme, kontakt) — F-01
- Unijeti podatke o vozilu — F-02
- Odabrati servisera i slobodan termin — F-02.1
- Provjeriti dostupnost termina — F-02.2
- Pregledati lokaciju servisera — F-02.3
- Rezervirati zamjensko vozilo — F-02.4
- Generirati PDF potvrdu prijave — F-03
- Primiti PDF potvrdu e-mailom — F-03.1
- Pratiti status popravka u stvarnom vremenu — F-04
- Izvesti podatke u PDF/XML/XLSX — F-08
- Predaja/preuzimanje vozila - F-09

A-2 Ovlašteni serviser (inicijator) može:

- Ažurirati status/termin i dodavati napomene — F-05
- Predaja vozila - F-09
- Preuzimanje vozila - F-10

A-3 Voditelj servisa (inicijator) može:

- Pregled statistika i dashboarda poslovanja — F-07
- Izvoz podataka u PDF, XML i XLSX — F-08
-

A-4 Administrator (inicijator) može:

- Upravljati korisnicima, serviserima, terminima i zamjenskim vozilima — F-06

A-5 E-mail servis (sudionik) može:

- Slati PDF potvrde nakon prijave — F-03.1
- Slati obavijesti o promjeni statusa/termina — F-05.1

A-6 Google Maps servis (sudionik) može:

- Prikazati lokaciju odabranog servisera tijekom odabira termina — F-02.3

A-7 Baza podataka (sudionik)

- Sudjeluje u svim obrascima uporabe spremanjem i dohvatom podataka o korisnicima, vozilima, terminima i statusima popravka.

A-8 Naručitelj (sudionik)

- Prati provedbu projekta i definira zahtjeve aplikacije u suradnji s razvojnim timom.

A-9 Razvojni tim (sudionik)

- Implementira funkcionalnosti, održava sustav i provodi testiranja novih verzija.

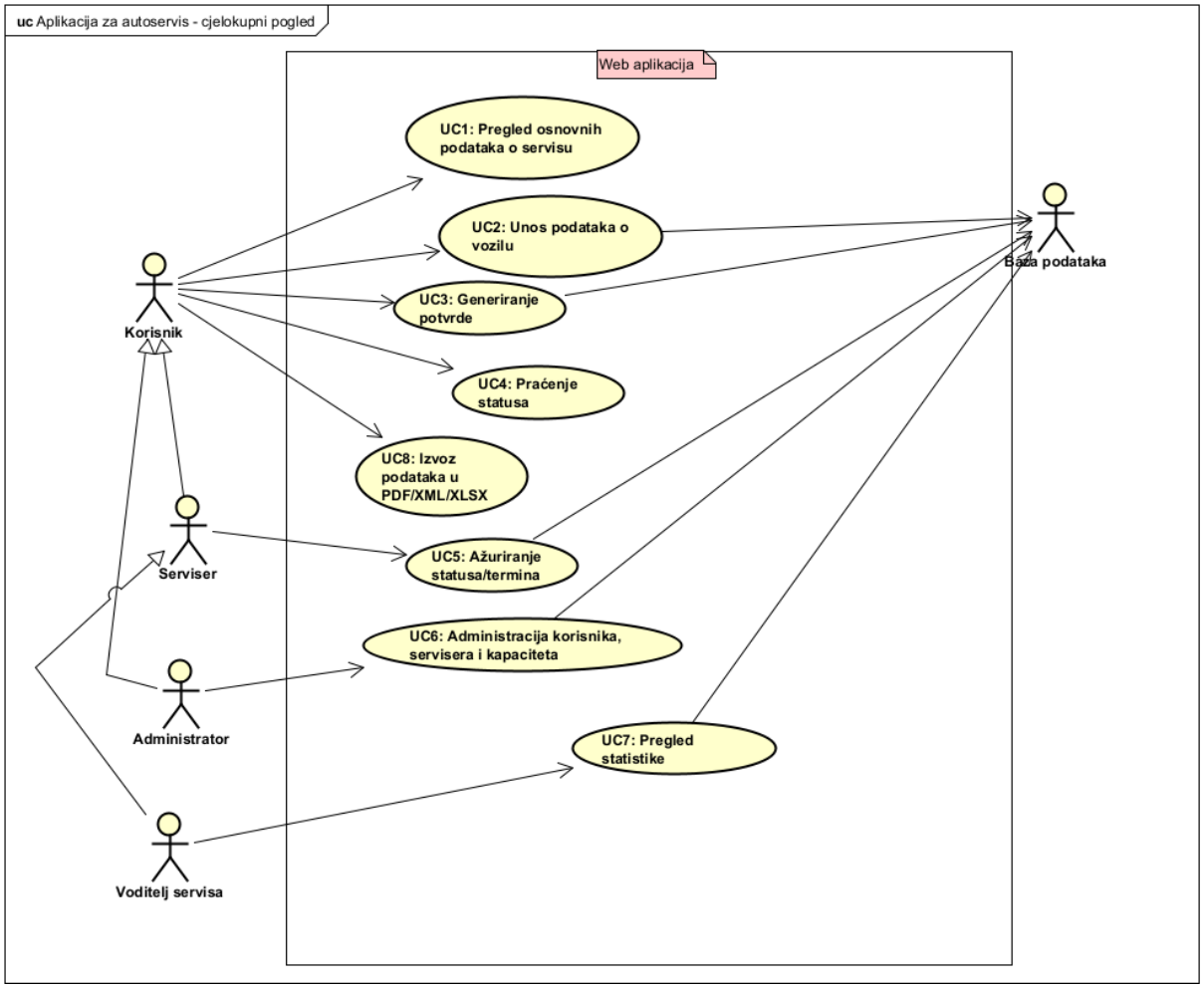
Specifikacija zahtjeva sustava

3.1. Dijagram obrasca uporabe

U ovom poglavlju prikazani su osnovni obrasci uporabe našeg sustava "Aplikacija za autoservis".

Dijagrami su grupirani prema razinama apstrakcije – od općeg pregleda sustava do prikaza pojedinih značajki, poslovnih procesa i integracija.

Za svaki identificirani obrazac uporabe izrađen je tekstualni opis koji sadrži aktere, ciljeve, preduvjete, osnovni tijek radnji i moguća odstupanja.



3.2. Obrasci uporabe

UC1 – Pregled osnovnih podataka o servisu

Glavni sudionik: Korisnik

Cilj: Pregledati osnovne informacije o servisu bez potrebe za prijavom.

Sudionici: Baza podataka

Preduvjet: Nema

Opis osnovnog tijeka:

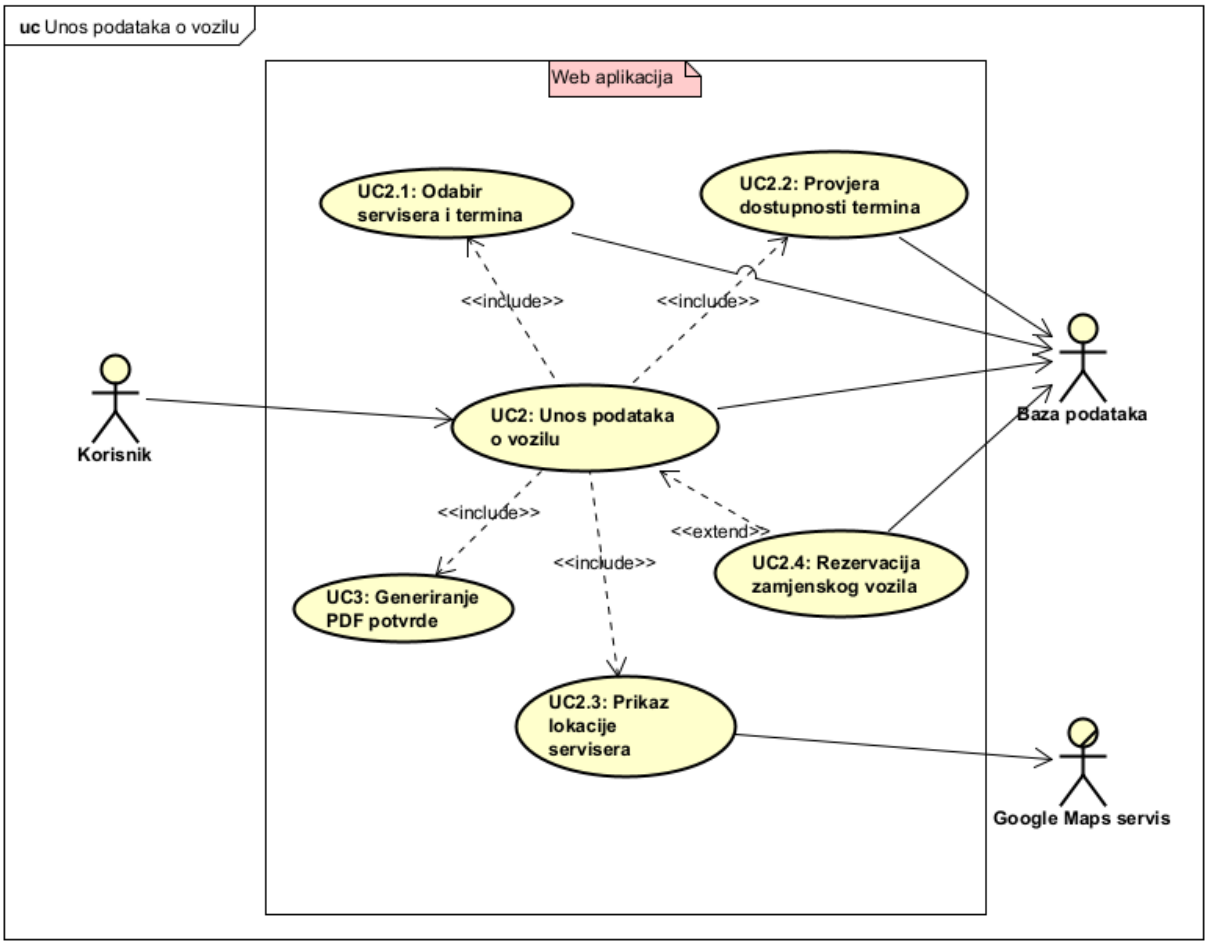
1. Korisnik otvara početnu stranicu aplikacije.
2. Sustav dohvaća osnovne informacije o servisu iz baze podataka.
3. Sustav prikazuje naziv, adresu, radno vrijeme i kontakt podatke servisa.

Opis mogućih odstupanja:

a) Baza podataka nije dostupna – sustav prikazuje poruku da trenutačno nije moguće dohvatiti podatke.

b) Neki podaci nedostaju – prikazuje se samo dostupne informacije uz upozorenje o nepotpunosti.

UC2 – Unos podataka o vozilu



Glavni sudionik: Korisnik

Cilj: Evidentirati vozilo koje će biti predano na servis.

Sudionici: Baza podataka

Preduvjet: Korisnik je prijavljen u sustav.

Opis osnovnog tijeka:

Korisnik otvara obrazac za unos podataka o vozilu.

1. Unosi marku, model, registraciju i opis problema.
2. Sustav provjerava ispravnost i potpunost unosa.
3. Sustav sprema podatke u bazu i prikazuje poruku o uspješnom unosu.

Opis mogućih odstupanja:

a) Nedostaju obavezni podaci – sustav prikazuje upozorenje i vraća korisnika na obrazac.

b) Došlo je do pogreške u komunikaciji s bazom – korisnik dobiva poruku o neuspjelom spremanju i može pokušati ponovno.

UC2.1 – Odabir servisera i termina

Glavni sudionik: Korisnik

Cilj: Odabrati ovlaštenog servisera i rezervirati termin.

Sudionici: Baza podataka, Google Maps servis

Preduvjet: Korisnik je unio podatke o vozilu (UC2).

Opis osnovnog tijeka:

1. Korisnik otvara izbornik za odabir servisera i termina.
2. Sustav prikazuje popis slobodnih termina i dostupnih servisera.
3. Korisnik odabire željeni termin i servisera.

Opis mogućih odstupanja:

a) Termin je zauzet – sustav prikazuje poruku o nedostupnosti i nudi druge termine.

b) Google Maps servis nije dostupan – prikazuje se obavijest o grešci i korisnik nastavlja bez karte.

UC2.2 – Provjera dostupnosti termina

Glavni sudionik: Sustav

Cilj: Provjeriti dostupnost termina u stvarnom vremenu.

Sudionici: Baza podataka

Opis osnovnog tijeka:

1. Sustav uspoređuje odabrani termin s postojećim rezervacijama.
2. Ako je termin slobodan, označava ga kao rezerviran.
3. Ako termin nije slobodan, sustav obavještava korisnika.

Opis mogućih odstupanja:

a) Greška u bazi – sustav obavještava korisnika da provjera nije moguća i predlaže ponovni pokušaj.

UC2.3 – Prikaz lokacije servisera putem Google Maps

Glavni sudionik: Korisnik

Cilj: Pregledati lokaciju odabranog servisera.

Sudionici: Google Maps servis

Preduvjet: Korisnik je u procesu odabira servisera.

Opis osnovnog tijeka:

1. Sustav šalje zahtjev Google Maps servisu s adresom servisera.
2. Servis vraća kartu s označenom lokacijom.
3. Sustav prikazuje kartu korisniku u okviru aplikacije.

Opis mogućih odstupanja:

a) Google Maps servis nije dostupan – sustav prikazuje poruku da prikaz karte trenutno nije moguć.

UC2.4 – Rezervacija zamjenskog vozila

Glavni sudionik: Korisnik

Cilj: Rezervirati zamjensko vozilo tijekom trajanja popravka.

Sudionici: Baza podataka

Preduvjet: Korisnik je u procesu odabira servisera.

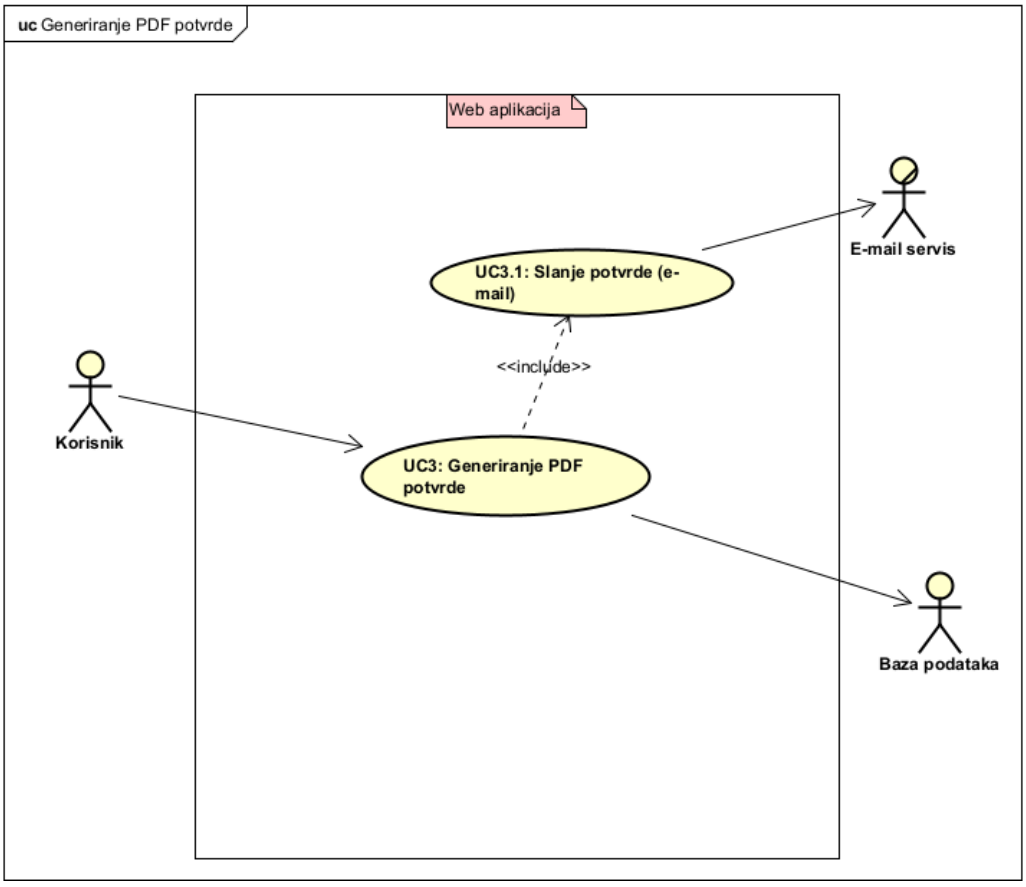
Opis osnovnog tijeka:

1. Korisnik odabire opciju rezervacije zamjenskog vozila.
2. Sustav bilježi rezervaciju i povezuje je s nalogom korisnika.

Opis mogućih odstupanja:

a) Nema dostupnih vozila – sustav obavještava korisnika i nudi mogućnost nastavka bez zamjenskog vozila.

UC3 – Generiranje PDF potvrde



Glavni sudionik: Korisnik

Cilj: Dobiti potvrdu o uspješno prijavljenom/preuzetom vozilu.

Sudionici: Baza podataka, E-mail servis

Preduvjet: Korisnik je prijavio/preuzeo vozilo.

Opis osnovnog tijeka:

1. Sustav generira PDF dokument s detaljima prijave.
2. PDF se sprema u bazu i povezuje s korisničkim nalogom.
3. Sustav šalje PDF potvrdu korisniku putem e-maila.

Opis mogućih odstupanja:

a) Greška pri generiranju PDF-a – korisnik može ponovno pokrenuti generiranje.

UC3.1 – Slanje PDF potvrde korisniku putem e-maila

Glavni sudionik: E-mail servis

Cilj: Poslati korisniku potvrdu o prijavi vozila.

Sudionici: Sustav aplikacije

Preduvjet: PDF potvrda je uspješno generirana.

Opis osnovnog tijeka:

1. Sustav šalje zahtjev e-mail servisu s privitkom PDF potvrde.
2. E-mail servis šalje potvrdu korisniku.

Opis mogućih odstupanja:

a) E-mail servis nije dostupan – poruka se sprema u red čekanja i šalje kasnije.

UC4 – Praćenje statusa popravka

Glavni sudionik: Korisnik

Cilj: Provjeriti trenutačni status popravka vozila.

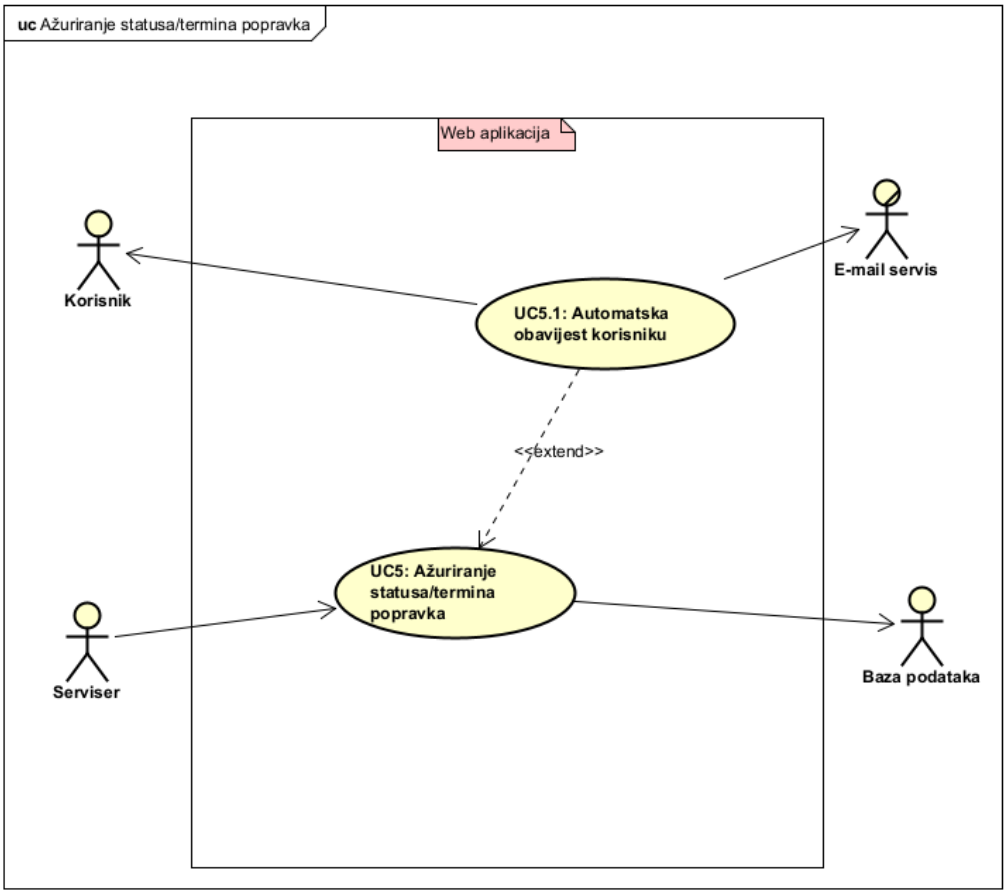
Sudionici: Baza podataka

Preduvjet: Korisnik ima registriran nalog i vozilo na servisu.

Opis osnovnog tijeka:

1. Korisnik otvara svoj profil i pregled aktivnih naloga.
2. Sustav prikazuje popis prijava s trenutačnim statusom.
3. Statusi se ažuriraju u realnom vremenu.

UC5 – Ažuriranje statusa ili termina popravka



Glavni sudionik: Serviser

Cilj: Promijeniti status/termin vozila i dodati napomenu.

Sudionici: Korisnik, Baza podataka, E-mail servis

Preduvjet: Vozilo postoji u sustavu.

Opis osnovnog tijeka:

1. Serviser otvara listu dodijeljenih vozila.
2. Odabire vozilo i mijenja njegov status/termin te time ažurira bazu podataka.
3. Obavještava korisnika o napravljenoj promjeni ako je termin promijenjen na više od 3 dana od početnog datuma servisa.

UC5.1 – Automatsko obavještavanje korisnika o promjeni statusa ili termina

Glavni sudionik: Sustav

Cilj: Obavijestiti korisnika o promjeni statusa ili termina vozila.

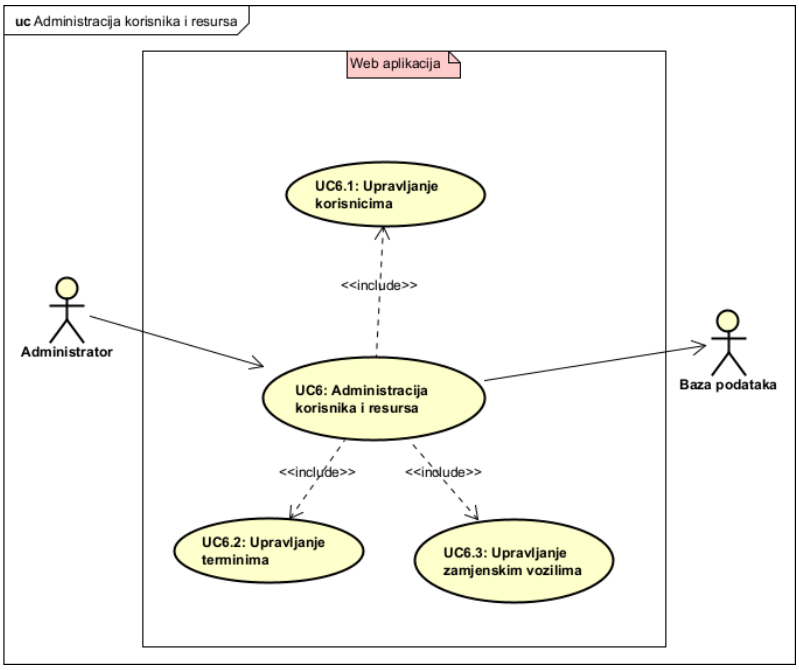
Sudionici: E-mail servis

Preduvjet: Serviser je promijenio status/termin vozila. Termin je promijenjen na više od 3 dana od početnog datuma servisa.

Opis osnovnog tijeka:

1. Sustav generira e-mail poruku s novim statusom/terminom.
2. E-mail servis šalje poruku korisniku.

UC6 – Administracija korisnika i resursa



Glavni sudionik: Administrator

Cilj: Upravljati korisnicima, serviserima, terminima i zamjenskim vozilima.

Sudionici: Administrator, Baza podataka

Preduvjet: Administrator ima aktivan račun s odgovarajućim ovlastima.

Opis osnovnog tijeka:

1. Administrator otvara upravljačko sučelje.
2. Dodaje, uređuje ili deaktivira korisnike i termine.
3. Upravlja raspoloživim vozilima.
4. Sprema promjene i osvježava podatke u bazi.

UC6.1 – Upravljanje korisnicima

Glavni sudionik: Administrator

Cilj: Kreirati, urediti, deaktivirati korisničke račune i dodijeliti uloge (RBAC).

Sudionici: Administrator, Baza podataka, E-mail servis (opcionalno)

Preduvjet: Administrator ima aktivan račun s ulogom ADMIN.

Opis osnovnog tijeka:

1. Administrator otvara modul “Korisnici”.
2. Odabire akciju: dodaj novog, uredi postojećeg ili deaktiviraj korisnika.
3. Unosi ili mijenja podatke (ime, e-mail, uloga, status).
4. Sustav validira podatke i sprema promjene u bazu.

Opis mogućih odstupanja:

a) Duplikat e-mail adrese – sustav vraća poruku i ne sprema promjene.

UC6.2 – Upravljanje terminima

Glavni sudionik: Administrator

Cilj: Upravljati kalendarom termina, blokadama i kapacitetima radionice ili zamjenskih vozila.

Sudionici: Administrator, Baza podataka, E-mail servis

Preduvjet: Administrator ima aktivan račun; postoje definirani serviseri i vozila.

Opis osnovnog tijeka:

1. Administrator otvara kalendar termina.
2. Dodaje, uređuje ili otkazuje termine i blokade (godišnji odmor, kvar, praznik).
3. Sustav provjerava kolizije (serviser, vozilo, kapacitet).
4. Ako nema konflikta, promjene se spremaju u bazu.

UC6.3 – Upravljanje zamjenskim vozilima

Glavni sudionik: Administrator

Cilj: Upravljati evidencijom i dostupnošću zamjenskih vozila.

Sudionici: Administrator, Baza podataka

Preduvjet: Administrator ima aktivan račun; postoje osnovni podaci o voznom parku.

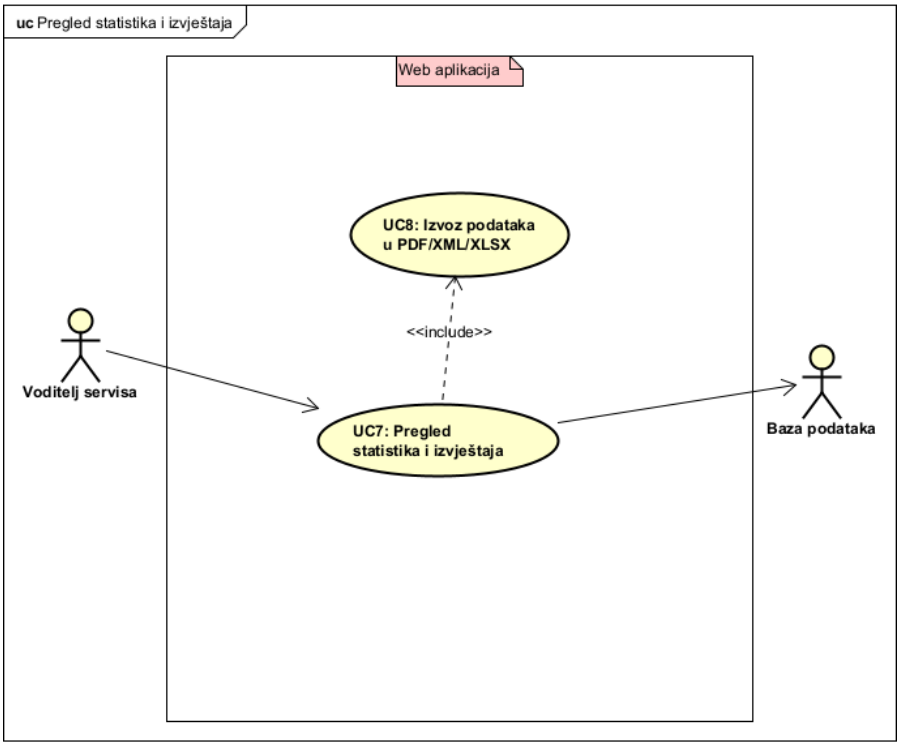
Opis osnovnog tijeka:

1. Administrator otvara modul “Zamjenska vozila”.
2. Dodaje novo vozilo ili uređuje postojeće (registracija, tip, status: dostupno, servis, blokirano).
3. Definira dostupnost po terminima i blokade (servis, tehnički pregled).
4. Sustav validira promjene i zapisuje ih u bazu.
5. Sustav automatski ažurira dostupnost u sustavu rezervacija korisnika.

Opis mogućih odstupanja:

a) Vozilo već rezervirano – sustav traži alternativu ili odgodu.

UC7 – Pregled statistika i izvještaja



Glavni sudionik: Voditelj servisa

Cilj: Pregledati ključne pokazatelje i izvesti izvještaje.

Sudionici: Baza podataka

Preduvjet: Postoje zapisi o servisiranim vozilima i korisnicima.

Opis osnovnog tijeka:

1. Voditelj otvara dashboard.
2. Sustav prikazuje broj vozila, trajanja popravaka i iskorištenost resursa.
3. Voditelj može izvesti podatke u PDF, XML ili XLSX formatu.

UC8 - Izvoz podataka u PDF, XML i XLSX

Glavni sudionik: Sustav

Cilj: Izvesti glavne podatke o servisu određenog vozila.

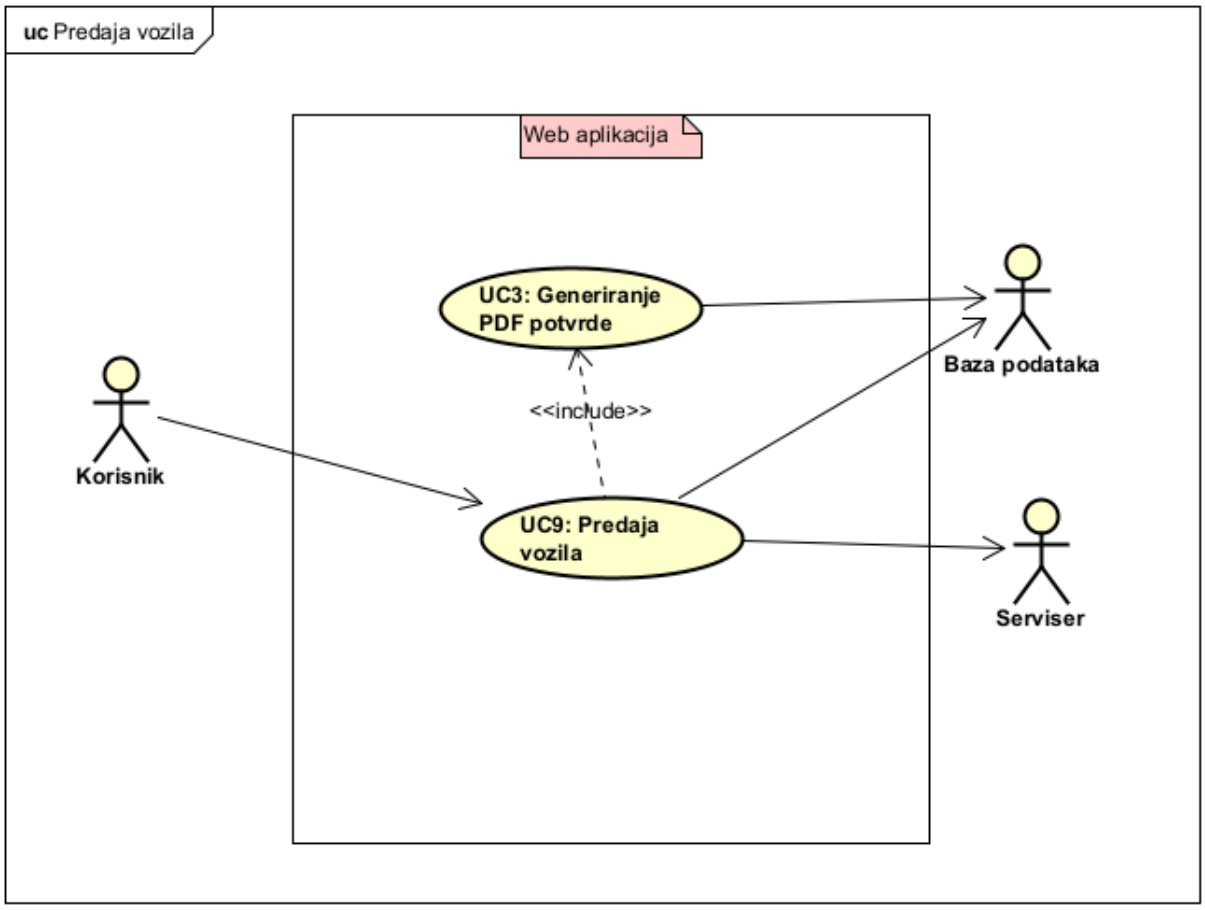
Sudionici: Korisnik

Preduvjet: Korisnik je postavio barem jedno vozilo na servis.

Opis osnovnog tijeka:

1. Korisnik pritisne gumb za izvoz određenog vozila koje je postavljeno na servis.
2. Sustav izvodi podatke vezano uz to vozilo u obliku PDF/XML/XLSX

UC9 - Predaja vozila



Glavni sudionik: Korisnik

Cilj: Predati vozilo na servis u dogovorenom terminu.

Sudionici: Korisnik, Serviser, Baza podataka

Preduvjet: Korisnik ima potvrđen termin servisa.

Opis osnovnog tijeka:

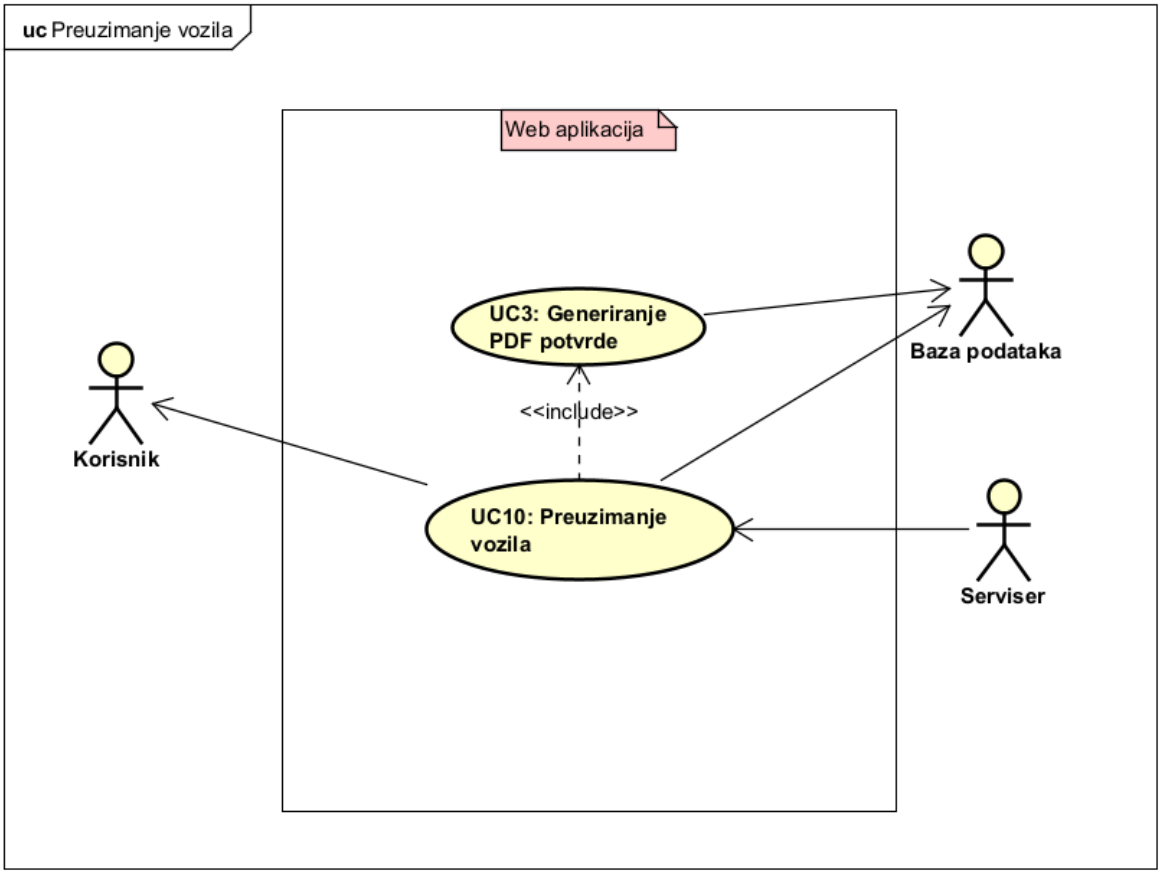
1. Korisnik dolazi u servis u dogovorenom terminu.
2. Serviser preuzima podatke o vozilu i nalogu iz sustava.

3. Sustav ažurira status vozila u bazi podataka.
4. Sustav generira potvrdu o predaji vozila i povezuje je s korisničkim nalogom.

Opis mogućih odstupanja:

a) Vozilo ne odgovara prijavljenom vozilu - serviser obvještava korisnika.

UC10 - Preuzimanje vozila



Glavni sudionik: Korisnik

Cilj: Preuzeti vozilo nakon završenog popravka.

Sudionici: Korisnik, Serviser, Baza podataka, E-mail servis

Preduvjet: Popravak vozila je dovršen.

Opis osnovnog tijeka:

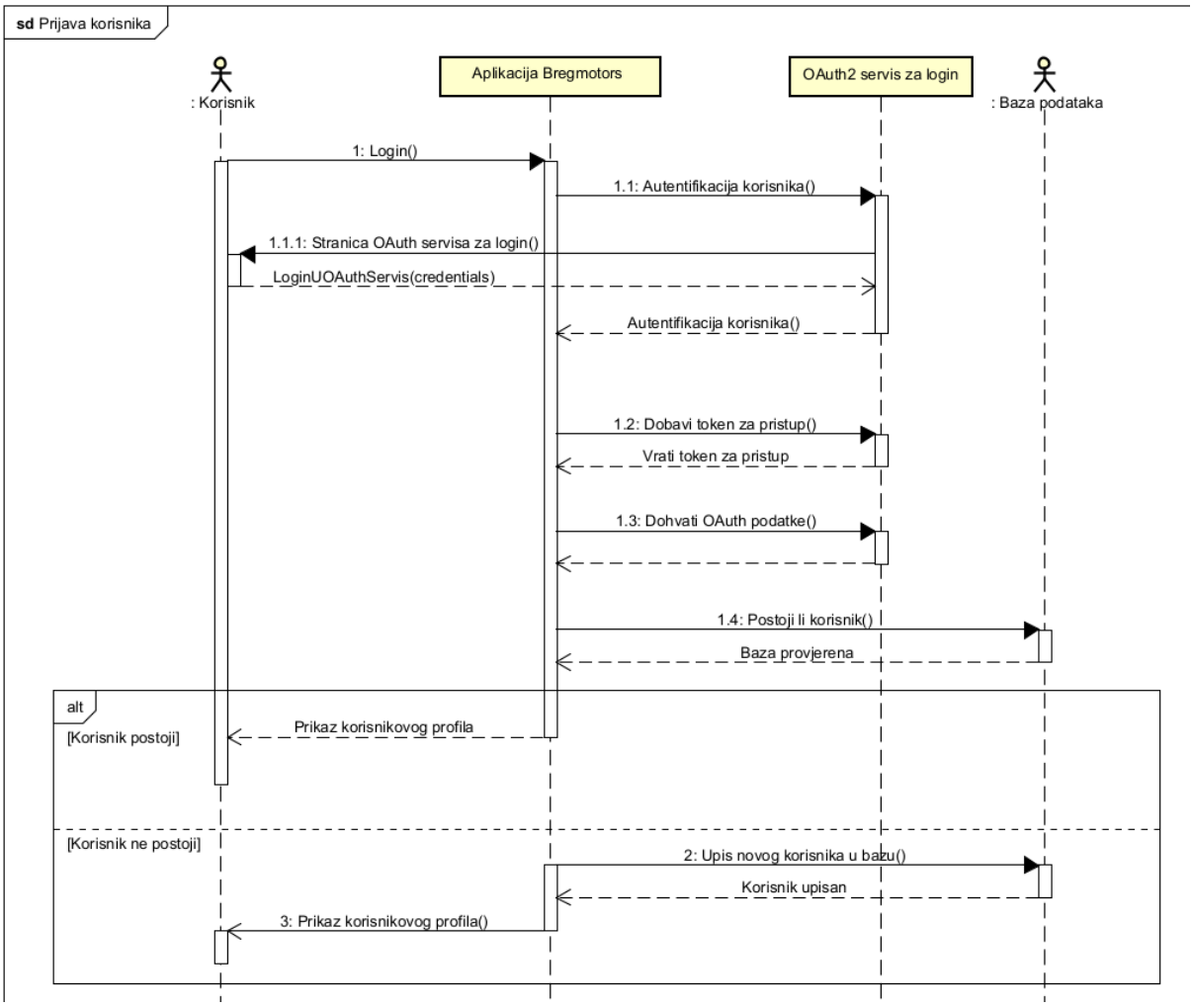
1. Serviser predaje vozilo i korisnik potvrđuje preuzimanje te potpisuje zapisnik o preuzimanju vozila.
2. Sustav ažurira status vozila u bazi podataka.
3. Sustav automatski arhivira sve podatke o servisu i zatvara nalog.

Opis mogućih odstupanja:

a) Korisnik ne može doći na vrijeme - termin preuzimanja se mijenja.

3.3. Sekvencijski dijagrami

Prijava korisnika



1. Pritišće gumb za prijavu te se prijavljuje putem Googlea.

    1.1 Aplikacija šalje zahtjev za autentifikaciju korisnika.

1.1.1 Korisnik upisuje svoje podatke u login obrazac.

1.2 Aplikacija dohvaća token od login servisa.

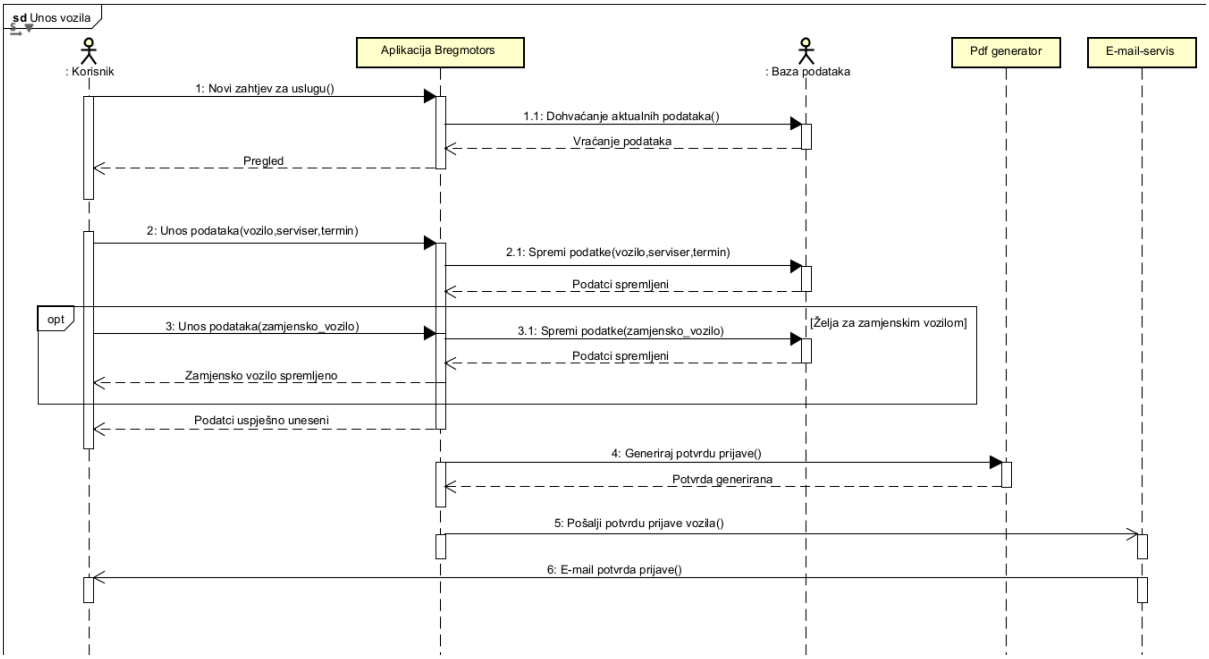
1.3 Dohvaćaju se OAuth podatci koji su dostupni.

1.4 Tražimo podatke iz baze o postoji li korisnik, ako da, onda se prikazuje korisnikov profil.

Inače (alt):

2. Unosi se novi korisnik u bazu.
3. Prikazuje se korisnikov profil.

Unos vozila



1. Šalje se novi zahtjev za unos vozila (pritiskom na gumb kojim se unosi novo vozilo).

1.1 Dohvaćaju se aktualni podaci o serviserima, terminima i vozilima koja se servisiraju.

2. Korisnik unosi podatke o svojem vozilu, serviseru kojeg želi, te željenom terminu.

2.1 Spremaju se podatci o usluzi u bazu podataka.

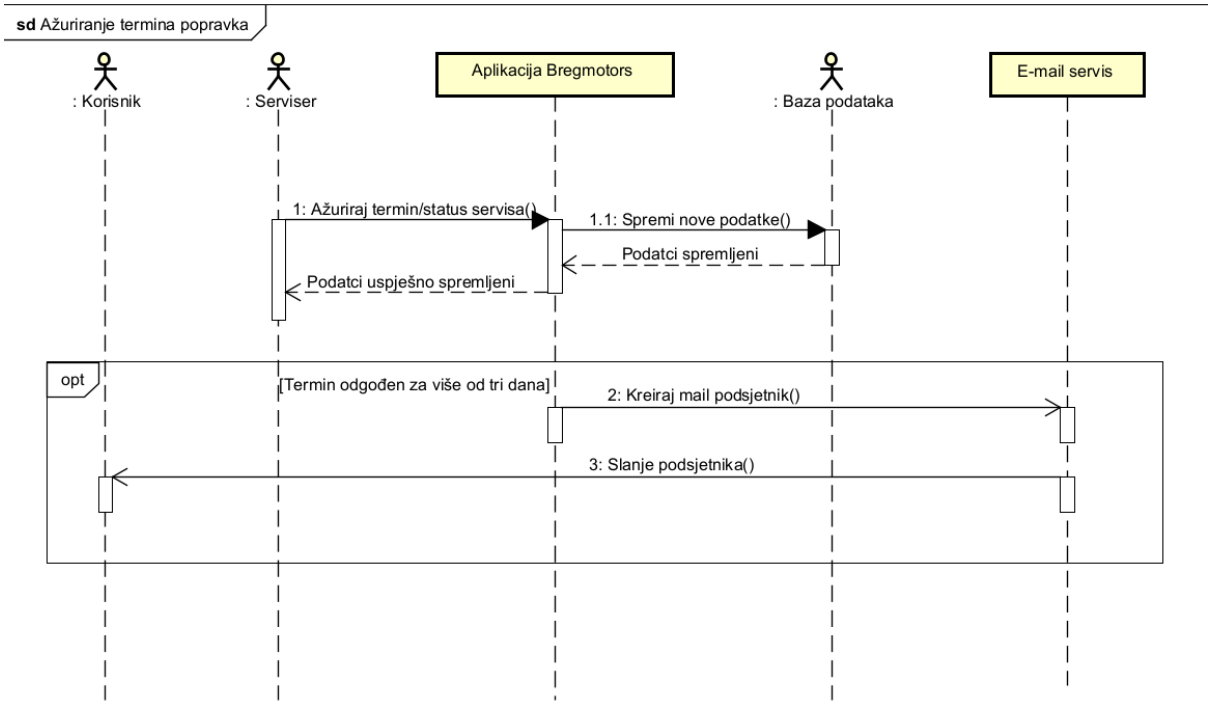
U slučaju da korisniku treba zamjensko vozilo (opt):

3. Korisnik unosi zamjensko vozilo koje želi.

    3.1 Spremaju se podatci o zamjenskom vozilu.

4. Aplikacija daje do znanja pdf generatoru da kreira potvrdu o prijavi vozila.
5. Aplikacija daje do znanja E-mail servisu da pošalje potvrdu korisniku.
6. Mail potvrda se šalje korisniku.

Ažuriranje termina popravka



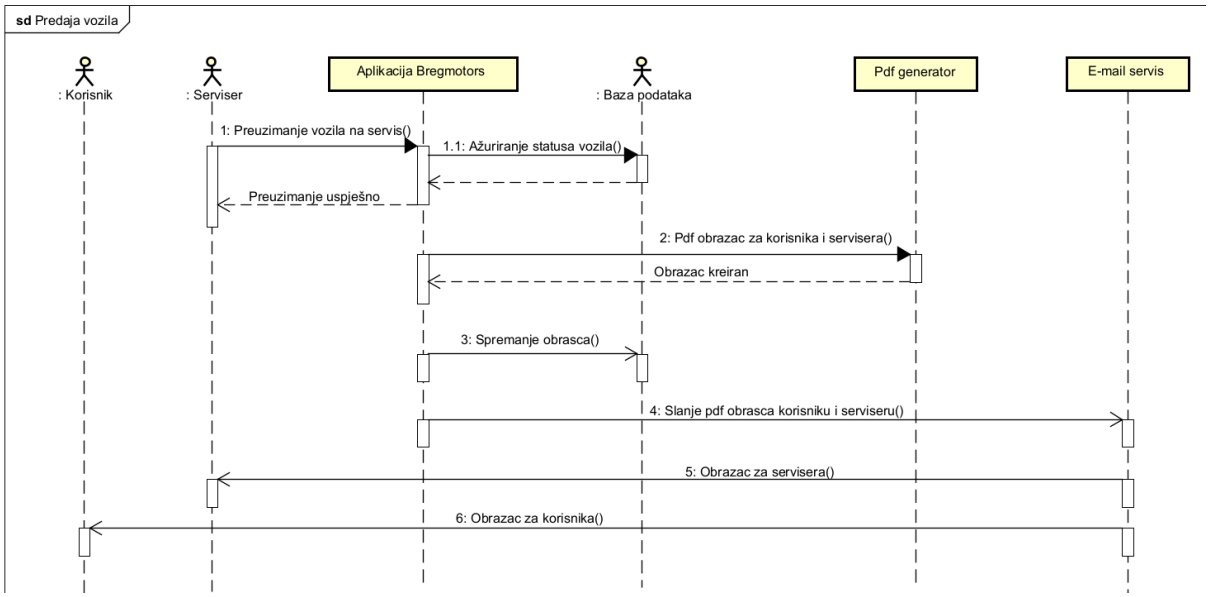
1. Serviser ažurira termin ili status servisa po svojoj volji.

    1.1 Novi se podatci spremaju u sustav.

U slučaju da je termin odgođen za više od tri dana (opt):

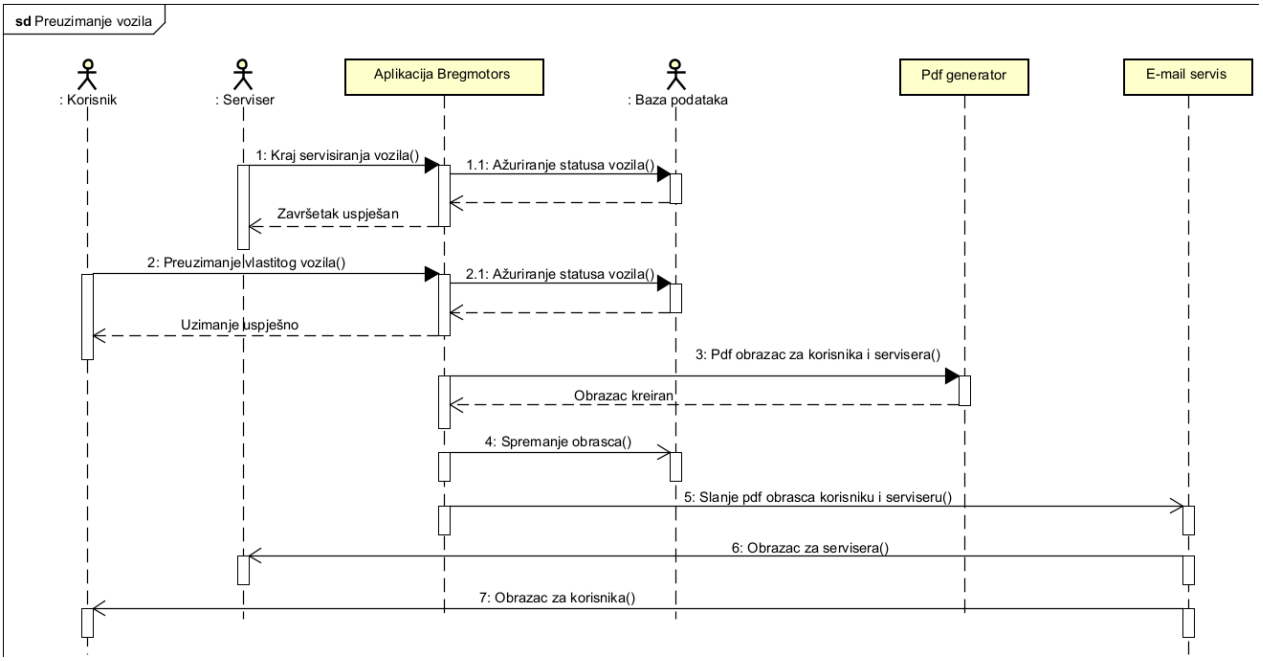
2. Aplikacija daje do znanja da se kreira mail podsjetnik za korisnika.
3. Podsjetnik se šalje korisniku.

Predaja vozila



1. Serviser preuzima vozilo na servis.
    1.1 Preuzimanje vozila se zapisuje u bazu.
2. Generira se pdf obrazac za korisnika i servisera.
3. Obrazac se sprema u bazu podataka.
4. Aplikacija daje do znanja da se generirani pdf obrazac pošalje korisniku i serviseru.
5. E-mail servis šalje serviseru obrazac.
6. E-mail servis šalje korisniku obrazac.

Preuzimanje vozila



1. Serviser predaje vozilo nakon završenog servisa.

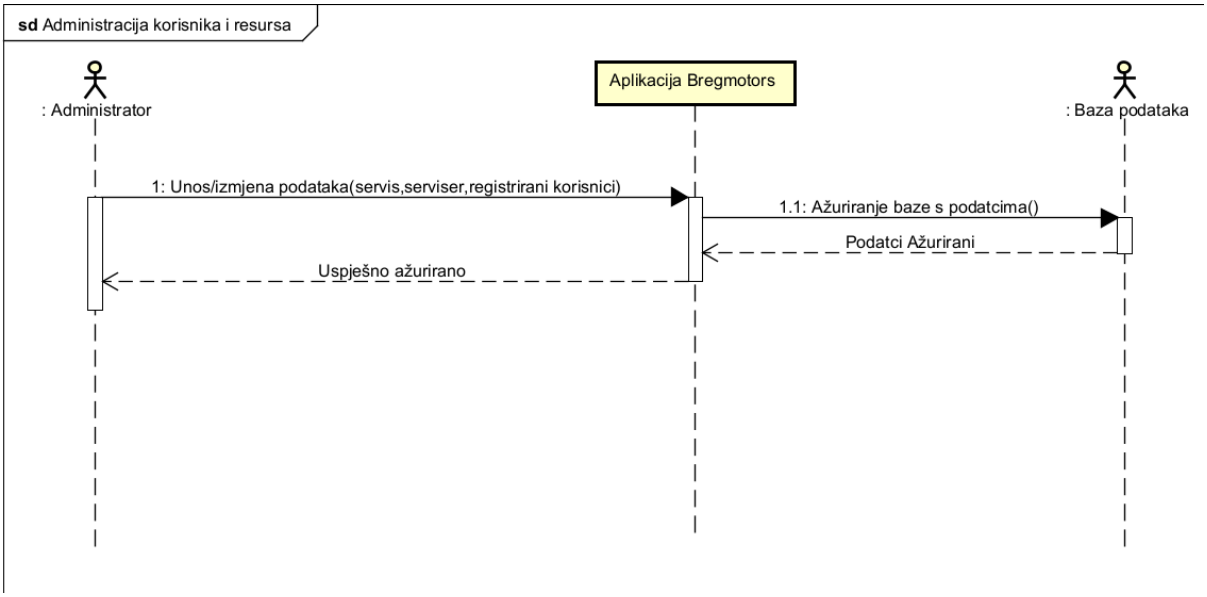
    1.1 Predaja se zapisuje u bazu podataka.

2. Korisnik preuzima svoje servisirano vozilo.

    2.1 Preuzimanje vozila se zapisuje u bazu.

3. Generira se pdf obrazac za korisnika i servisera.
4. Obrazac se sprema u bazu podataka
5. Aplikacija daje do znanja da se generirani pdf obrazac pošalje korisniku i serviseru.
6. E-mail servis šalje serviseru obrazac.
7. E-mail servis šalje korisniku obrazac.

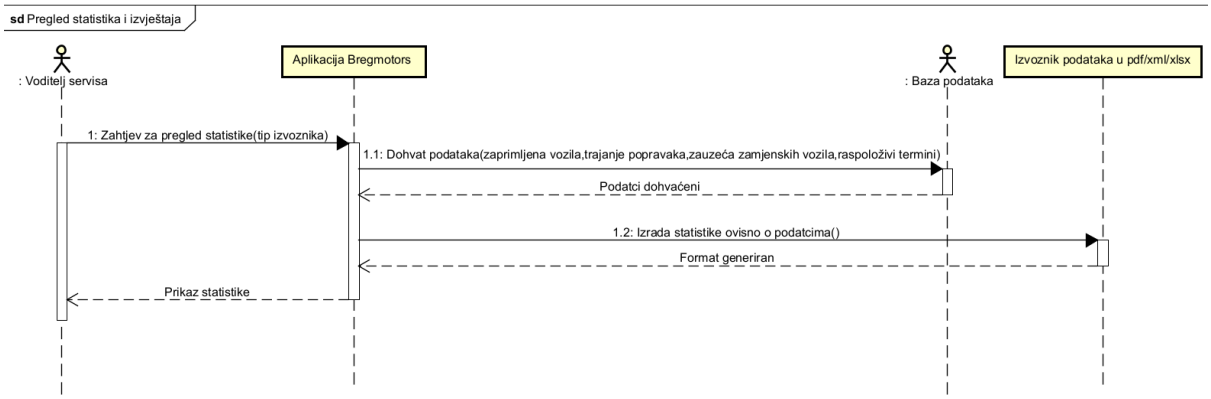
Administracija korisnika i resursa



1. Administrator može unositi ili izmjenjivati sljedeće podatke: sve pojedinosti o servisu, podatke o serviseru te podatke o registriranom korisniku.

    1.1 Baza podataka se ažurira novim podatcima.

Pregled statistika i izvještaja

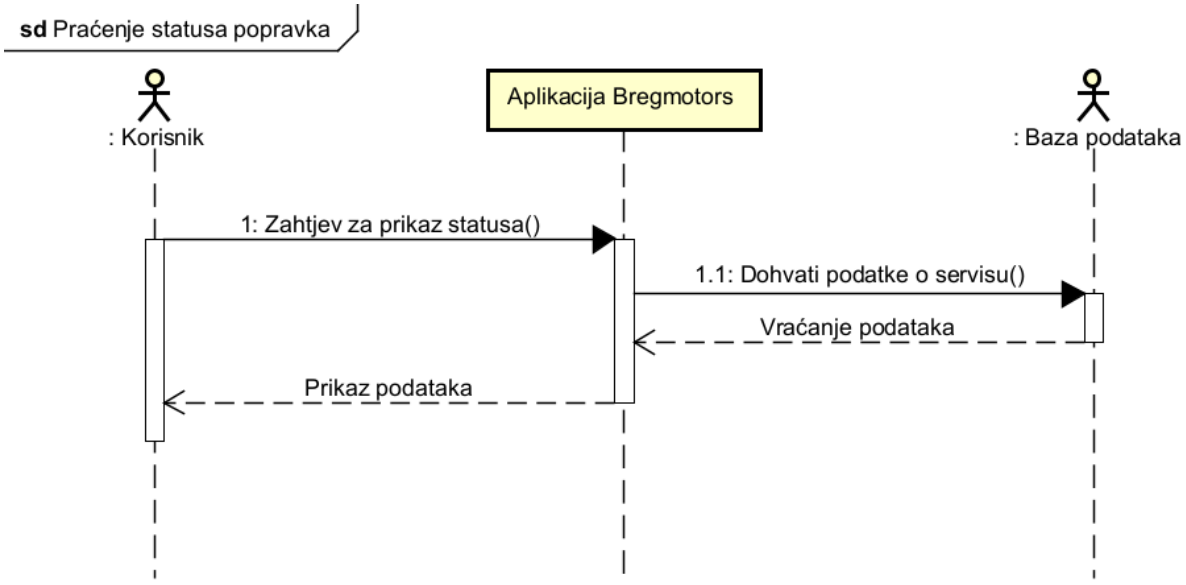


1. Administrator šalje zahtjev aplikaciji da mu kreira pregled statistike ovisno o željenom formatu.

    1.1 Podatci koji su potrebni u statistici se dohvaćaju iz baze (svi su navedeni u sekvencijskom dijagramu).

    1.2 Izrađuje se statistika ovisno o izabranom formatu.

Praćenje statusa popravka



1. Korisnik šalje zahtjev za prikaz trenutnog statusa servisa.

    1.1 Podatci o servisu se dohvaćaju iz baze podataka.

3.4 Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

UC dijagram	Funkcionalnosti
UC1	F-01
UC2	F-02
UC2.1	F-02.1
UC2.2	F-02.2
UC2.3	F-02.3
UC2.4	F-02.4

UC dijagram	Funkcionalnosti
UC3	F-03
UC3.1	F-03.1
UC4	F-04
UC5	F-05
UC5.1	F-05.1
UC6	F-06
UC6.1	F-06.1
UC6.2	F-06.2
UC6.3	F-06.3
UC7	F-07
UC8	F-08
UC9	F-09
UC10	F-10

4. Arhitektura i dizajn sustava

4.1. Arhitektura sustava

4.1.1. Opis arhitekture

Stil arhitekture

Sustav koristi kombinirani arhitektonski stil koji spaja principe višeslojne arhitekture (three-layered architecture) i MVC (Model–View–Controller) obrasca. Arhitektura je podijeljena u 3 glavna sloja: prezentacijski sloj (presentation layer), poslovni sloj (business layer) i sloj pristupa podacima (data access layer). Ovakva struktura je odabrana jer osigurava odvajanje prikaza, logike i podataka (separation of concerns), što olakšava održavanje, testiranje i daljnji razvoj sustava.

Podsustavi

- Usluga autentifikacije i autorizacije: Upravlja prijavom korisnika, autentifikacijom i autorizacijom pomoću usluge OAuth2 koja koristi JSON Web Token (JWT) za provjeru korisničkih zahtjeva.
- Usluga upravljanja korisnicima: Upravlja kreiranjem, ažuriranjem i brisanjem korisničkih profila.
- Usluga za upravljanje vozilima i prijavama servisa: Omogućuje unos i pregled zaprimljenih vozila, statusa popravaka i pridruženih servisera. Povezan je s modulom za slanje automatskih obavijesti i podsjetnika korisnicima.
- Usluga za upravljanje terminima: Raspoređuje termine i resurse (dijelove, servisere, zamjenska vozila). Prati zauzetost i dostupnost u realnom vremenu.
- Usluga za upravljanje zamjenskim vozilima: Evidentira i upravlja rezervacijama zamjenskih vozila, provjerava njihovu dostupnost i status.
- Usluga za statistiku i izvještaje: Generira pregledne izvještaje o broju zaprimljenih vozila, trajanju popravaka, zauzeću resursa i učinkovitosti servisa. Omogućuje izvoz podataka u PDF, XML i XLSX format.
- Usluga za obavijesti i komunikaciju: Automatski šalje obavijesti e-poštom korisnicima (npr. potvrda termina, završetak popravka).
- Usluga za geolokaciju servisa: Integrira Google Maps API za prikaz točne lokacije servisa i omogućuje korisnicima jednostavno planiranje dolaska.

Preslikavanje na radnu platformu

Za implementaciju arhitekture odabran je cloud pristup putem Render platforme. Ovaj pristup omogućuje jednostavnu implementaciju, skalabilnost i podršku za moderne tehnologije. Render je izabran zbog svojih besplatnih planova za web servise i baze podataka, što omogućuje besplatno testiranje i razvoj aplikacija bez dodatnih troškova.

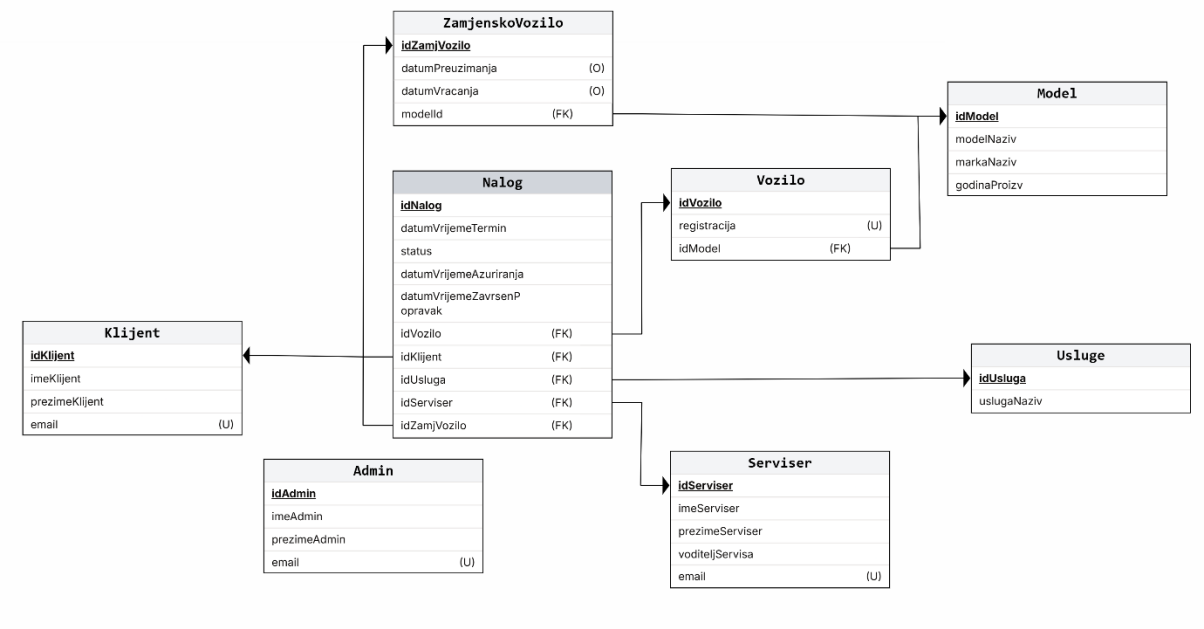
Proces gradnje i postavljanja aplikacije na Render platformu potpuno je automatiziran korištenjem GitHub Actions. Svaka promjena u kodu automatski se testira, gradi i postavlja u produkcijsko okruženje, čime se smanjuje potreba za ručnim intervencijama. Pipeline skripte, smještene u .github/workflows direktoriju, definiraju korake za build i deploy backend i frontend aplikacije.

Za obavještavanje Render platforme o dostupnosti nove verzije aplikacije koriste se webhookovi. Kada pipeline završi uspješno, GitHub Actions automatski aktivira webhook, pokrećući redeploy nove verzije na Renderu.

Ovaj proces omogućuje pouzdano, brzo i efikasno ažuriranje aplikacije u produkciji, uz minimalan ručni rad. Korištenjem kombinacije GitHub Actions i Render webhookova, svaka nova verzija aplikacije može biti postavljena jednostavno i bez zastoja.

Spremište podataka

Korištena je PostgreSQL baza podataka jer je open source, besplatna, standardizirana i dovoljna za sve potrebe projekta. Razvili smo i relacijski dijagram koji prikazuje veze između entiteta u bazi:



Mrežni protokoli

Za razgovor između frontenda i backenda se koristi protokol HTTP/HTTPS. Standardiziran je i služi kao osnovna za komunikaciju na webu. HTTP je stateless protokol što znači da nije potrebno pamtiti stanje za koristiti ga, a time se štedi na memoriji i brzini obrade zahtjeva. Svaki zahtjev između klijenta i poslužitelja je neovisan. HTTP podupire i širok spektar tipova podataka koji se specificiraju s content-type zaglavljem. Podupire metode: GET, POST, PUT, DELETE koje se koriste u RESTful API-jima.

Globalni upravljački tok

Korisnik započinje interakciju putem React frontenda, gdje unosi podatke u obrazac — primjerice za prijavu, registraciju, rezervaciju termina ili unos podataka o vozilu. Frontend komunicira s backendom putem REST API poziva koristeći HTTP/HTTPS protokol, a u zaglavlje zahtjeva uključuje OAuth2 token za autentifikaciju i provjeru korisničkih ovlasti.

Kada zahtjev stigne u Spring Boot backend, Controller sloj prima zahtjev i prosljeđuje ga Service sloju, koji obrađuje poslovnu logiku (npr. provjeru dostupnosti termina, ažuriranje statusa vozila ili generiranje izvještaja). Ako je potrebno pristupiti podacima, Service sloj koristi Repository sloj koji preko JPA (Java Persistence API) komunicira s PostgreSQL bazom podataka. Repository sloj koristi

JDBC konekciju za dohvaćanje, ažuriranje i pohranu podataka o korisnicima, vozilima, servisima, zamjenskim vozilima i izvještajima.

Nakon obrade, backend vraća odgovor — primjerice potvrdu rezervacije, ažurirani status vozila, generirani izvještaj ili poruku o grešci. Frontend zatim prikazuje korisniku povratne informacije i omogućuje daljnju interakciju sa sustavom, poput pregledavanja statistika ili unosa novih zahtjeva.

Cjelokupni proces odvija se unutar slojevite arhitekture, koja osigurava jasno razdvajanje odgovornosti između prikaznog, logičkog i podatkovnog sloja te omogućuje jednostavno održavanje, sigurnu komunikaciju i pravovremene odgovore na korisničke zahtjeve.

Sklopovskoprogramski zahtjevi

Za backend i bazu podataka treba Linux ili Windows Server. Potrebni su Java 21, Spring Boot za backend, Maven za upravljanje ovisnostima, PostgreSQL za bazu podataka te OAuth2 za autentifikaciju.

4.1.2 Obrazloženje odabira arhitekture

Razlozi za odabir arhitekture

Odabrana je slojevita arhitektura koja se sastoji od React frontenda, Spring Boot backend aplikacije i PostgreSQL baze podataka. Takav pristup omogućuje jasno razdvajanje prezentacijskog sloja (korisničko sučelje), poslovne logike (obrada zahtjeva i pravila rada sustava) i podatkovnog sloja (upravljanje bazom podataka).

Ova arhitektura odabrana je jer zadovoljava sve zahtjeve projekta – od responzivnog prikaza i brze komunikacije s korisnikom do sigurnog pristupa i pouzdanog čuvanja podataka o korisnicima, vozilima, terminima i izvještajima.

Ključni čimbenici koji su utjecali na odabir:

- Skalabilnost: React frontend i Spring Boot backend mogu se razvijati i nadograđivati neovisno jedan o drugome. Time se omogućuje jednostavno dodavanje novih funkcionalnosti (npr. proširenje modula za statistike ili upravljanje resursima) bez potrebe za većim promjenama u ostatku sustava.

- Održavanje: Zbog jasnog razdvajanja slojeva (Controller – Service – Repository) sustav je jednostavniji za održavanje, testiranje i ispravljanje pogrešaka.
- Modularnost: Svaka komponenta ima jasno definiranu odgovornost, što olakšava razvoj u timovima i ponovno korištenje koda.
- Sigurnost: Autentifikacija i autorizacija provode se putem OAuth2 protokola, što omogućuje sigurnu prijavu i kontrolu pristupa.
- Fleksibilnost: Arhitektura omogućuje buduće nadogradnje, primjerice prijelaz na mikrousluge (microservices) ili integraciju s vanjskim sustavima putem REST API-ja.

Izbor arhitekture temeljen na principima oblikovanja

Prilikom oblikovanja arhitekture poštovani su sljedeći principi dobrog programskog dizajna:

- Visoka kohezija: Svaki sloj obavlja jasno definiranu ulogu – prezentacijski sloj prikazuje podatke, servisni sloj upravlja poslovnom logikom, a sloj pristupa podacima komunicira isključivo s bazom.
- Niska povezanost (low coupling): Slojevi međusobno komuniciraju preko jasnih API sučelja, što omogućuje da se jedan sloj promijeni bez utjecaja na ostale.
- Sigurnost i pouzdanost: Korištenjem standardiziranih tehnologija (OAuth2, HTTPS, JPA) postiže se visoka razina sigurnosti i otpornosti na pogreške.
- Fleksibilnost i proširivost: Novi moduli (npr. analiza podataka, napredno planiranje resursa) mogu se jednostavno dodati bez promjene postojećih funkcionalnosti.
- Održivost: Jasna arhitektura omogućuje lakšu nadogradnju i održavanje koda tijekom dugoročnog korištenja sustava.

Razmatrane alternative

Nismo razmatrali alternative.

4.1.3. Organizacija sustava na visokoj razini

Klijent - poslužitelj

Sustav je organiziran u klijent–poslužitelj arhitekturi:

- Klijent (frontend): React web-aplikacija koja radi u pregledniku. Komunicira s backendom isključivo putem REST API poziva (HTTPS, JSON). U zahtjevima šalje OAuth2 bearer token radi autentifikacije i autorizacije.
- Poslužitelj (backend): Spring Boot aplikacija slojevite strukture (Controller → Service → Repository). Obrada poslovne logike (npr. rezervacija termina, promjena statusa popravka, izračun statistika) te validacija i sigurnosne provjere odvijaju se na poslužitelju.
- Protokol i format: HTTPS, JSON (request/response), s jasno definiranom verzioniranom API površinom (npr. /api/v1/...).

Baza podataka

- Tip i uloga: Relacijska baza podataka PostgreSQL je centralni izvor istine. Pohranjuje korisnike, uloge/ovlasti, vozila, termine, radne naloge, zapise o statusima, evidenciju zamjenskih vozila te metapodatke izvještaja.
- Pristup i integritet: Pristup preko JPA/Hibernate (Repository sloj). Osigurani su ACID svojstva, referencijalni integritet (FK), ograničenja i indeksi za ključna polja.

Datotečni sustav

Sustav ne koristi poseban datotečni sustav za pohranu podataka, već se svi podaci pohranjuju u relacijskoj bazi podataka PostgreSQL.

Baza podataka služi kao jedinstveno mjesto za pohranu svih informacija o korisnicima, vozilima, servisnim terminima, zamjenskim vozilima i generiranim izvještajima. Time se osigurava centralizirano upravljanje podacima, jednostavnije održavanje i veća pouzdanost sustava.

Ako se privremeno generiraju izvještaji (npr. u PDF, XML ili XLSX formatu), oni se ne trajno pohranjuju, već se korisniku odmah nude za preuzimanje putem web sučelja.

Grafičko sučelje

Vrsta: Web sučelje temeljeno na Reactu (SPA) s responzivnim dizajnom za desktop i mobilne uređaje.

Povezanost s komponentama:

- Dohvaća i šalje podatke backendu preko REST API-ja.
- Nakon prijave, u lokalnom stanju drži minimalne podatke o korisniku i ulogama (primljene kroz /auth/me), a oslanja se na HTTP-only kolačiće ili bearer tokene za sigurnu autentifikaciju.

Funkcionalni moduli UI-ja:

- Raspored/termini i upravljanje resursima
- Vozila i radni nalozi (statusi, povijest)
- Izvještaji i export (PDF/XML/XLSX)
- Administracija korisnika i uloga (RBAC)

4.1.4. Organizacija aplikacije

Frontend i Backend slojevi

Frontend (React) upravlja korisničkim sučeljem i predstavlja prezentacijski sloj sustava. Njegova je uloga omogućiti korisniku intuitivnu interakciju sa sustavom – prikaz stranica, unos podataka u obrasce (npr. prijava, registracija, unos vozila, rezervacija termina) te prikaz informacija koje dolaze s poslužitelja.

Korisničke akcije, poput prijave ili spremanja novog vozila, šalju se backendu (Spring Boot) putem REST API poziva. Komunikacija između frontenda i backenda odvija se preko HTTP/HTTPS protokola, uz korištenje CORS (Cross-Origin Resource Sharing)

mehanizma koji omogućuje sigurnu razmjenu podataka između dvaju odvojenih domena ili portova.

Backend je implementiran u Spring Bootu i organiziran po slojevitoj arhitekturi, čime se postiže čitljivost, održivost i modularnost koda. Sastoji se od tri glavna sloja:

- Controller sloj prima HTTP zahtjeve s frontenda. Njegova je uloga prepoznati vrstu zahtjeva (npr. POST /login, GET /vozila) i proslijediti ga odgovarajućem servisu. Controller također vraća odgovor (npr. JSON objekt, poruku o pogrešci) koji frontend može obraditi i prikazati korisniku.
- Service sloj sadrži poslovnu logiku aplikacije. Tu se odvijaju sve ključne operacije poput provjere korisničkih podataka, validacije unosa, izračuna statistika, dodjeljivanja termina servisima i generiranja izvještaja. Ako servis mora dohvatiti ili spremiti podatke, poziva repository sloj.
- Repository sloj zadužen je za komunikaciju s bazom podataka. Putem JPA (Java Persistence API) koristi relacijski model baze PostgreSQL, što omogućuje dohvat, spremanje, brisanje i ažuriranje podataka pomoću SQL naredbi. Sloj je apstrahiran od aplikacijske logike, čime se omogućuje lako održavanje i zamjena baze ako je potrebno.

Cjelokupan proces odvija se u obliku zatvorene petlje: korisnik pošalje zahtjev putem React sučelja, backend ga obradi i vrati rezultat, a frontend automatski ažurira prikaz korisničkog sučelja bez potrebe za ponovnim učitavanjem stranice.

MVC Arhitektura

Aplikacija koristi MVC arhitekturu (Model–View–Controller), što znači da je sustav organiziran u tri jasno odvojena sloja koji zajedno omogućuju čistu strukturu, lakše održavanje i mogućnost paralelnog razvoja.

- Model predstavlja podatke i poslovnu logiku sustava. U ovoj aplikaciji to uključuje entitete kao što su Korisnik, Vozilo, Serviser, TerminiliZamjenskoVozilo, koji odražavaju strukturu tablica u PostgreSQL bazi podataka. Model definira kako se podaci pohranjuju, povezuju i obrađuju te sadrži validacijsku i domensku logiku (npr. provjeru dostupnosti termina ili izračun trajanja popravka). Putem JPA (Java Persistence API) model

komunicira s bazom podataka i omogućuje automatsko mapiranje između objekata i relacijskih tablica.

- View predstavlja korisničko sučelje, koje je izrađeno u Reactu. View sloj prikazuje podatke koje dohvaća iz backend API-ja te omogućuje korisniku interakciju putem obrazaca, tablica i grafičkih elemenata (npr. prikaz liste servisa, unos novog vozila, pregled statusa popravka). Zahvaljujući Reactovoj modularnosti, sučelje je dinamično, responzivno i lako proširivo, što korisnicima osigurava brzo i intuitivno iskustvo rada sa sustavom.
- Controller upravlja zahtjevima koji dolaze s korisničkog sučelja i povezuje ih s poslovnom logikom u Spring Bootu. Kada korisnik putem React frontenda pošalje zahtjev (npr. prijavu, rezervaciju termina, izmjenu statusa vozila), Controller sloj ga prima, validira ulazne podatke te prosljeđuje odgovarajućem servisu (Service sloju). Nakon obrade, Controller vraća odgovor u obliku JSON objekta, koji React koristi za ažuriranje prikaza.

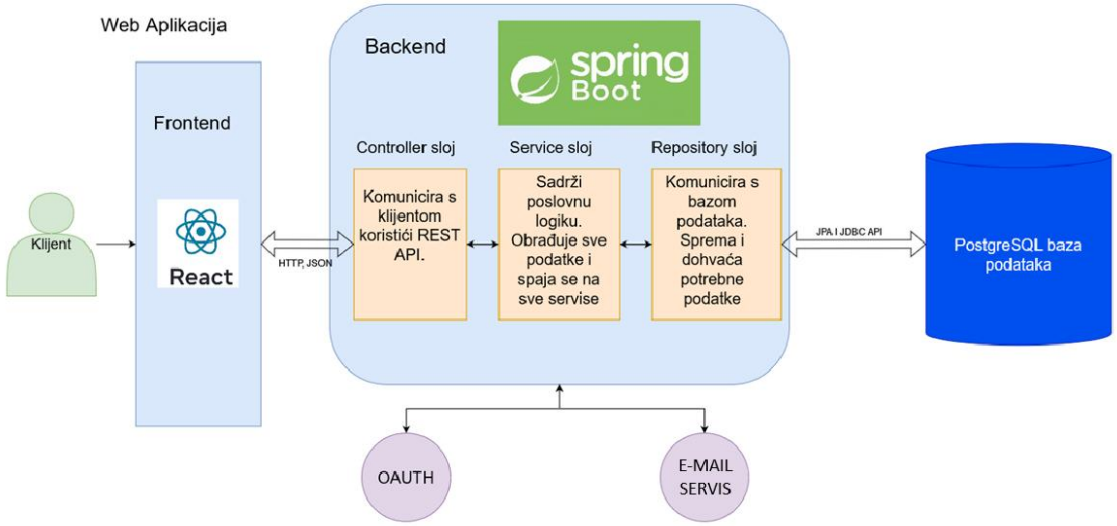
Primjena MVC arhitekture omogućila je jasno razdvajanje odgovornosti:

- poslovna logika i podaci nalaze se u Modelu i Service sloju,
- prikaz i interakcija s korisnikom u View sloju,
- dok Controller služi kao posrednik između njih.

Ovakav pristup donosi brojne prednosti:

- Paralelan rad nad komponentama – frontend i backend tim mogu raditi istodobno, jer su sučelja i logika odvojeni i međusobno komuniciraju putem REST API-ja.
- Jednostavno održavanje – promjene u sučelju ne utječu na poslovnu logiku, i obrnuto.
- Ponovna upotrebljivost (reusability) – komponente, modeli i kontroleri mogu se koristiti u različitim dijelovima aplikacije bez dupliciranja koda.
- Jednostavno testiranje – svaki dio sustava može se testirati zasebno, što omogućuje brže otkrivanje pogrešaka i veću stabilnost aplikacije.

Dijagram visoke razine



4.2. Baza podataka

Za potrebe projekta odabrana je relacijska baza podataka PostgreSQL, koja se koristi za pohranu, upravljanje i dohvat svih podataka sustava. PostgreSQL je open-source RDBMS poznat po svojoj stabilnosti, sigurnosti i podršci za kompleksne relacije među tablicama.

Baza se sastoji od više međusobno povezanih entiteta, među kojima su ključni: korisnici, vozila, servisni termini, zamjenska vozila, radni nalozi i izvještaji. Povezanost između tablica ostvarena je putem primarnih i stranih ključeva, čime se osigurava referencijalni integritet i konzistentnost podataka.

4.2.1. Opis tablica

Klijent

Atribut	Tip podatka	Ključ	Opis varijable
`idKlijent`	INTEGER	PK	Jedinstveni identifikator klijenta
`imeKlijent`	VARCHAR(100)		Ime klijenta
`prezimeKlijent`	VARCHAR(100)		Prezime klijenta
`email`	VARCHAR(75)	UNIQUE	Email adresa klijenta (jedinstvena)

Admin

Atribut	Tip podatka	Ključ	Opis varijable
`idAdmin`	INTEGER	PK	Jedinstveni identifikator administratora
`imeAdmin`	VARCHAR(100)		Ime administratora
`prezimeAdmin`	VARCHAR(100)		Prezime administratora
`email`	VARCHAR(100)	UNIQUE	Email adresa administratora (jedinstvena)

Serviser

Atribut	Tip podatka	Ključ	Opis varijable
`idServiser`	INTEGER	PK	Jedinstveni identifikator servisera

Atribut	Tip podatka	Ključ	Opis varijable
`imeServiser`	VARCHAR(100)		Ime servisera
`prezimeServiser`	VARCHAR(100)		Prezime servisera
`voditeljServisa`	BOOLEAN		Označava je li serviser voditelj servisa
`email`	VARCHAR(100)	UNIQUE	Email adresa servisera (jedinstvena)

Vozilo

Atribut	Tip podatka	Ključ	Opis varijable
`idVozilo`	INTEGER	PK	Jedinstveni identifikator vozila
`registracija`	VARCHAR(50)	UNIQUE	Registracijska oznaka vozila (jedinstvena)
`idModel`	INTEGER	FK	Poveznica na tablicu **Model** (`Model.idModel`)

ZamjenskoVozilo

Atribut	Tip podatka	Ključ	Opis varijable
`idZamjVozilo`	INTEGER	PK	Jedinstveni identifikator zamjenskog vozila

Atribut	Tip podatka	Ključ	Opis varijable
`datumPreuzimanja`	DATE		Datum kada je klijent preuzeo vozilo
`datumVracanja`	DATE		Datum kada je klijent vratio vozilo
`modelId`	INTEGER	FK	Poveznica na tablicu **Model** (`Model.idModel`)

Model

Atribut	Tip podatka	Ključ	Opis varijable
`idModel`	INTEGER	PK	Jedinstveni identifikator modela
`modelNaziv`	VARCHAR(100)		Naziv modela vozila
`markaNaziv`	VARCHAR(100)		Naziv marke vozila
`godinaProizv`	INTEGER		Godina proizvodnje modela

Usluge

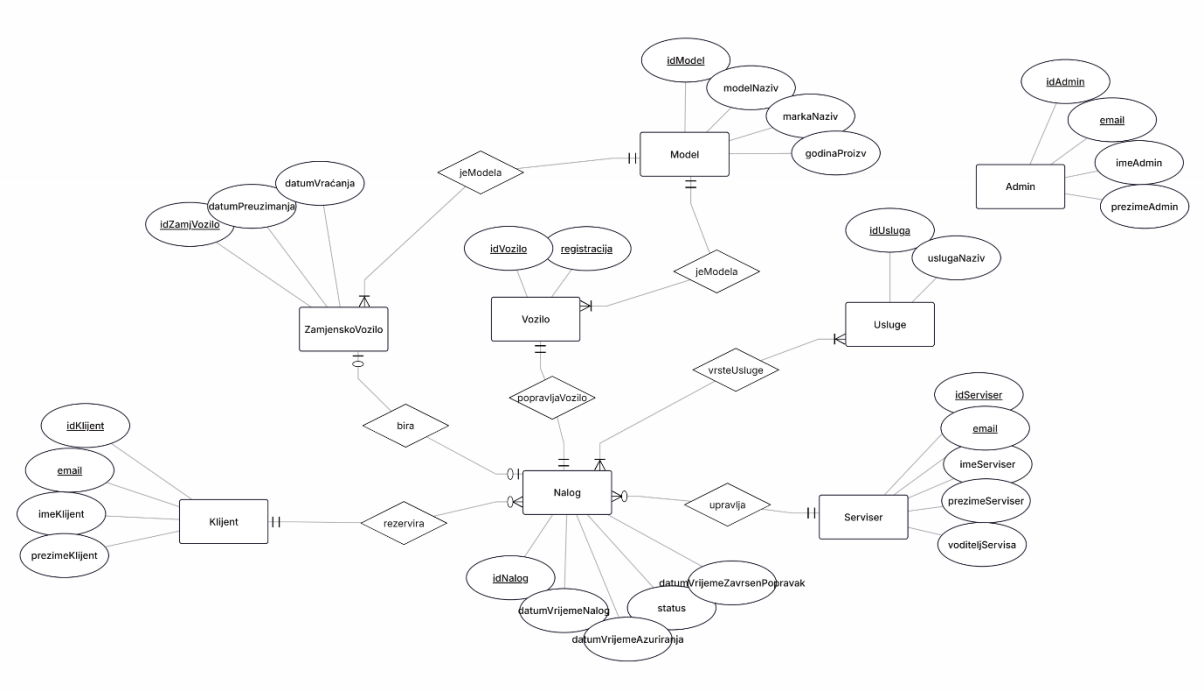
Atribut	Tip podatka	Ključ	Opis varijable
`idUsluga`	INTEGER	PK	Jedinstveni identifikator usluge
`uslugaNaziv`	VARCHAR(500)		Naziv usluge (npr. *Zamjena ulja*)

Nalog

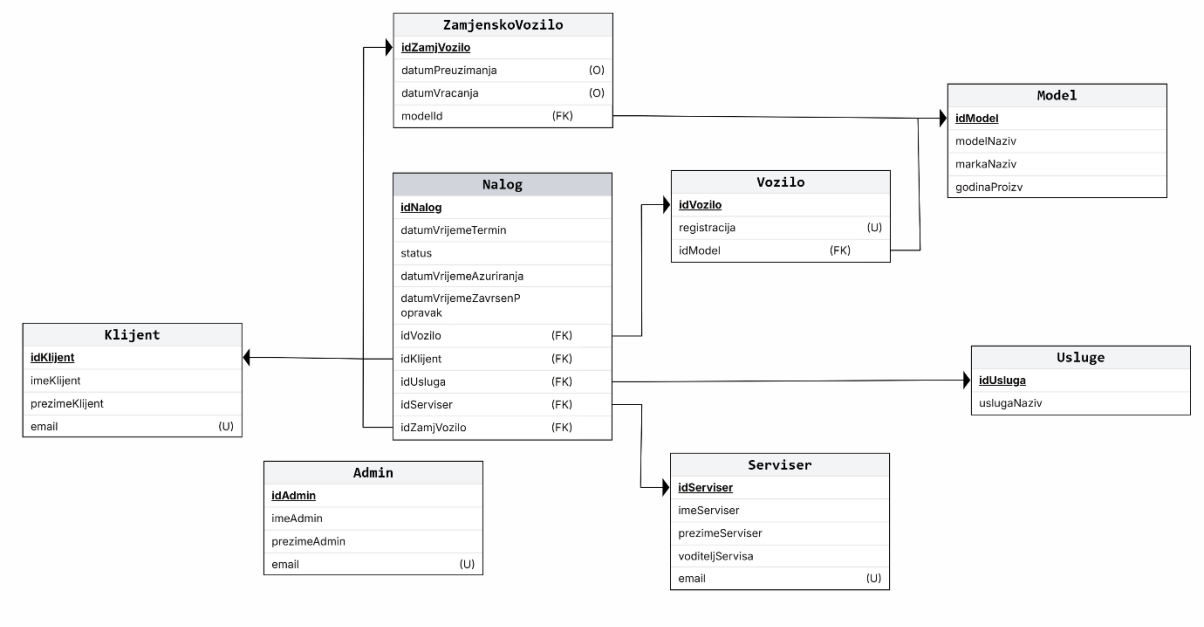
Atribut	Tip podatka	Ključ	Opis varijable
`idNalog`	INTEGER	PK	Jedinstveni identifikator naloga
`datumVrijemeTermin`	DATE		Datum kada je termin rezerviran
`datumVrijemeZavrsenPopravak`	DATE		Datum završetka popravka
`status`	INT		Status naloga (npr. *U tijeku*, *Završen*)
`datumVrijemeAzuriranja`	DATE		Datum kada je nalog ažuriran
`idVozilo`	INTEGER	FK	Poveznica na tablicu **Vozilo** (`Vozilo.idVozilo`)
`idKlijent`	INTEGER	FK	Poveznica na tablicu **Klijent** (`Klijent.idKlijent`)
`idUsluga`	INTEGER	FK	Poveznica na tablicu **Usluge** (`Usluge.idUsluga`)
`idServiser`	INTEGER	FK	Poveznica na tablicu **Serviser** (`Serviser.idServiser`)
`idZamjVozilo`	INTEGER	FK	Poveznica na tablicu **ZamjenskoVozilo** (`ZamjenskoVozilo.idZamjVozilo`)

4.2.2. Dijagrami baze podataka

ER dijagram



Relacijski dijagram



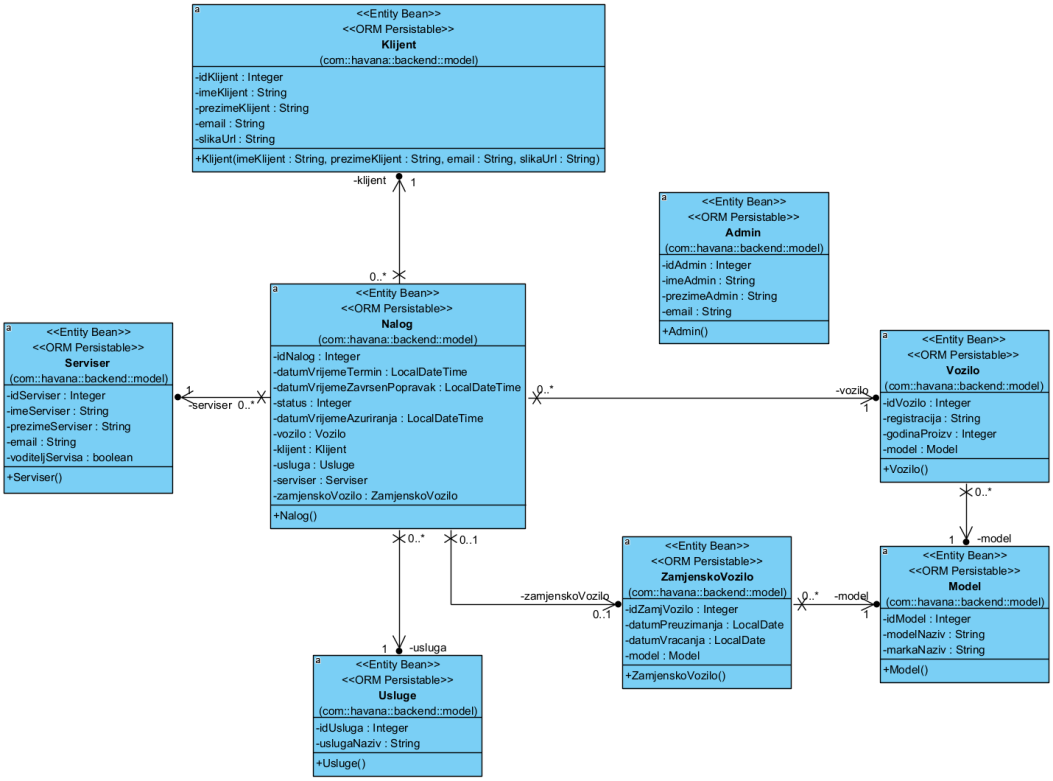
4.3 Dijagram razreda

Prva slika prikazuje sve entitete korištene u aplikaciji, koji su mapirani na odgovarajuće tablice u bazi podataka. Entiteti su označeni pomoću Entity Bean oznake, što ukazuje na to da su u pitanju Java klase koje predstavljaju podatke pohranjene u bazi podataka. ORM Persistable označava da klase komuniciraju s bazom podataka.

Odnosi među entitetima

- Jedan klijent može imati više naloga.
- Jedan serviser može izvršavati više naloga.
- Svaki nalog uključuje točno jednu uslugu, ali ista usluga može biti dio više naloga.
- Jedno vozilo može se pojaviti u više naloga.
- Jedno zamjensko vozilo može biti dodijeljeno najviše jednom nalogu u određenom razdoblju.
- Model se koristi i u klasama Vozilo i ZamjenskoVozilo, čime se ostvaruje veza između marke/modela i konkretnih vozila.

Svaki od entiteta ima generirane gettere i settere, prazne konsturktore i konstruktore sa svim atributima. Radi preglednosti nisu navedeni getteri i setteri.



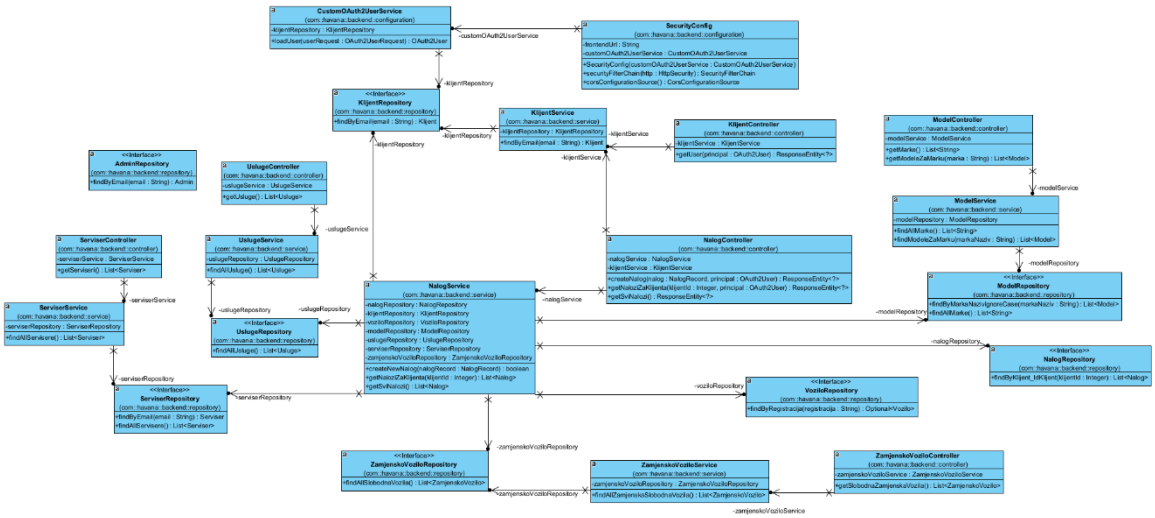
Ovaj dijagram razreda prikazuje glavne komponente aplikacije podijeljene na kontrolere, servise i repozitorije, koji zajedno čine tro-slojnu arhitekturu sustava.

Kontroleri (npr. NalogController, KlijentController) služe za obradu HTTP zahtjeva te prosljeđuju podatke servisima i vraćaju odgovore klijentu.

Servisi (npr. NalogService, KlijentService, StatistikaService) sadrže poslovnu logiku aplikacije i povezuju kontrolere s repozitorijima, obrađujući i kombinirajući podatke iz različitih izvora.

Repozitoriji (npr. NalogRepository, KlijentRepository) omogućuju pristup bazi podataka putem JPA sučelja te definiraju osnovne i prilagođene metode za dohvat podataka.

Sigurnosne klase (SecurityConfig, CustomOAuth2UserService) upravljaju autentifikacijom i autorizacijom korisnika putem Google OAuth2 prijave.


CustomOAuth2UserService
(com.havana.backend.configuration)
-klijentRepository : KlijentRepository
+loadUser(userRequest : OAuth2UserRequest) : OAuth2User
-customOAuth2UserService
SecurityConfig
(com.havana.backend.configuration)
-frontendUrl : String
-customOAuth2UserService : CustomOAuth2UserService
+SecurityConfig(customOAuth2UserService : CustomOAuth2UserService)
+securityFilterChain(http : HttpSecurity) : SecurityFilterChain
+corsConfigurationSource() : CorsConfigurationSource
-klijentRepository
<<Interface>>
KlijentRepository
(com.havana.backend.repository)
+findByEmail(email : String) : Klijent
KlijentService
(com.havana.backend.service)
-klijentRepository : KlijentRepository
+findByEmail(email : String) : Klijent
-klijentService
KlijentController
(com.havana.backend.controller)
-klijentService : KlijentService
+getUser(principal : OAuth2User) : ResponseEntity<?>
ModelController
(com.havana.backend.controller)
-modelService : ModelService
+getMarke() : List<String>
+getModeleZaMarku(marka : String) : List<Model>
-modelService
<<Interface>>
AdminRepository
(com.havana.backend.repository)
+findByEmail(email : String) : Admin
UslugeController
(com.havana.backend.controller)
-uslugeService : UslugeService
+getUsluge() : List<Usluge>
-uslugeService
ModelService
(com.havana.backend.service)
-modelRepository : ModelRepository
+findAllMarke() : List<String>
+findModeleZaMarku(marka : String) : List<Model>
-modelRepository
ServiserController
(com.havana.backend.controller)
-serviserService : ServiserService
+getServiseri() : List<Serviser>
UslugeService
(com.havana.backend.service)
-uslugeRepository : UslugeRepository
+findAllUsluge() : List<Usluge>
NalogController
(com.havana.backend.controller)
-nalogService : NalogService
-klijentService : KlijentService
+createNalog(nalog : NalogRecord, principal : OAuth2User) : ResponseEntity<?>
+getNaloziZaKlijenta(klijentId : Integer, principal : OAuth2User) : ResponseEntity<?>
+getSviNalozi() : ResponseEntity<?>
<<Interface>>
ModelRepository
(com.havana.backend.repository)
+findByMarkaNazivIgnoreCase(markaNaziv : String) : List<Model>
+findAllMarke() : List<String>
-serviserService
-uslugeRepository
-uslugeRepository
-nalogService
-modelRepository
-nalogRepository
ServiserService
(com.havana.backend.service)
-serviserRepository : ServiserRepository
+findAllServiseri() : List<Serviser>
<<Interface>>
UslugeRepository
(com.havana.backend.repository)
+findAllUsluge() : List<Usluge>
NalogService
(com.havana.backend.service)
-nalogRepository : NalogRepository
-klijentRepository : KlijentRepository
-vozilloRepository : VozilloRepository
-modelRepository : ModelRepository
-uslugeRepository : UslugeRepository
-serviserRepository : ServiserRepository
-zamjenskoVozilloRepository : ZamjenskoVozilloRepository
+createNewNalog(nalogRecord : NalogRecord) : boolean
+getNaloziZaKlijenta(klijentId : Integer) : List<Nalog>
+getSviNalozi() : List<Nalog>
<<Interface>>
NalogRepository
(com.havana.backend.repository)
+findByKlijent_IdKlijent(klijentId : Integer) : List<Nalog>
-serviserRepository
-serviserRepository
-vozilloRepository
<<Interface>>
ServiserRepository
(com.havana.backend.repository)
+findByEmail(email : String) : Serviser
+findAllServisers() : List<Serviser>
<<Interface>>
VozilloRepository
(com.havana.backend.repository)
+findByRegistracija(registracija : String) : Optional<Vozilo>
-zamjenskoVozilloRepository
<<Interface>>
ZamjenskoVozilloRepository
(com.havana.backend.repository)
+findAllSlobodnaVozila() : List<ZamjenskoVozilo>
ZamjenskoVozilloService
(com.havana.backend.service)
-zamjenskoVozilloRepository : ZamjenskoVozilloRepository
+findAllZamjenskaSlobodnaVozila() : List<ZamjenskoVozilo>
ZamjenskoVozilloController
(com.havana.backend.controller)
-zamjenskoVozilloService : ZamjenskoVozilloService
+getSlobodnaZamjenskaVozila() : List<ZamjenskoVozilo>
-zamjenskoVozilloRepository
-zamjenskoVozilloService