Oblikovanje programske potpore

Ak. god. 2013./2014.

*Prodajni informacijski sustav*

Dokumentacija, Rev. *1*

Grupa: *LeteciCirkus*

Voditelj: *Vlaho Poluta*

Datum predaje: *25.11.2013.*

Nastavnik: *Alan Jović*

Asistent: *Marko Horvat*

# Popis članova grupe i zaduženja

*Vlaho Poluta* - voditelj projekta, koordinacija članova projekta, obrasci uporabe, sekvencijski dijagrami,

*Stanko Krtalić Rusendić* - rječnik pojmova, dijagram razreda i objekata,

*Zrinka Lekić -* obrasci uporabe, sekvencijski dijagrami i opis,

*Marin Pogančić –* opis razreda, svrha i opći prioriteti sustava,

*Martin Sertić -* ostali zahtjevi, baza podataka,

*Mirjam Škarica -* opis projektnog zadatka,

*Mateja Škriljak -* dnevnik sastajanja, opis objekata, opis grafičkog sučelja,

# Sadržaj

TOC \o 2-3 \t "Heading, 4"

Popis članova grupe i zaduženja PAGEREF \_Toc \h 2

Sadržaj PAGEREF \_Toc1 \h 3

Dnevnik promjena dokumentacije PAGEREF \_Toc2 \h 5

Opis projektnog zadatka PAGEREF \_Toc3 \h 5

Rječnik pojmova PAGEREF \_Toc4 \h 8

Funkcionalni zahtjevi PAGEREF \_Toc5 \h 11

Ostali zahtjevi PAGEREF \_Toc6 \h 28

Arhitektura i dizajn sustava PAGEREF \_Toc7 \h 29

**Svrha, opći prioriteti i skica sustava PAGEREF \_Toc8 \h 29**

Svrha i opći prioriteti sustava PAGEREF \_Toc9 \h 29

Grafičko korisničko sučelje PAGEREF \_Toc10 \h 32

Baza podataka PAGEREF \_Toc11 \h 32

**Dijagram razreda s opisom PAGEREF \_Toc12 \h 39**

**Dijagram objekata PAGEREF \_Toc13 \h 48**

**Ostali UML dijagrami PAGEREF \_Toc14 \h 51**

Implementacija i korisničko sučelje PAGEREF \_Toc15 \h 51

**Dijagram razmještaja PAGEREF \_Toc16 \h 51**

**Korištene tehnologije i alati PAGEREF \_Toc17 \h 52**

**Isječak programskog koda vezan za temeljnu funkcionalnost sustava PAGEREF \_Toc18 \h 53**

**Ispitivanje programskog rješenja PAGEREF \_Toc19 \h 54**

**Upute za instalaciju PAGEREF \_Toc20 \h 55**

**Korisničke upute PAGEREF \_Toc21 \h 56**

Zaključak PAGEREF \_Toc22 \h 56

Reference PAGEREF \_Toc23 \h 56

Dodatak A: Dnevnik sastajanja PAGEREF \_Toc24 \h 56

# Dnevnik promjena dokumentacije

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev.** | **Opis promjene/dodatka** | **Autor(i)** | **Datum** |
| 0.1 | Napravljen predložak. | Poluta | 11.10.2013. |
| 0.2 | Uneseni *use case* i sekvencijski dijagrami. | Poluta | 28.10.2013. |
| 0.3 | Dodan rječnik pojmova. | Krtalić | 29.10.2013. |
| 0.4 | Unesen opis projektnog zadatka. | Škarica | 31.10.2013. |
| 0.5 | Dodan dijagram razreda, dijagram objekata. | Krtalić | 01.11.2013. |
| 0.6 | Opisi obrazaca uporabe i sekvencijskih dijagrama. | Lekić | 01.11.2013. |
| 0.7 | Dodani ostali zahtjevi. | Sertić | 01.11.2013. |
| 0.8 | Prikupljena i unesena arhitektura sustava. | Poluta | 03.11.2013. |
| 0.9 | Dodan dnevnik sastajanja. | Škriljak | 23.11.2013. |
| 1.0 | Verzija s dijelovima bitnim za 1. ciklus. | Poluta | 25.11.2013. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Opis projektnog zadatka

Cilj projekta je izrada prodajnog informacijskog sustava koji će podržavati poslovanje auto-servisa. Sustav bi trebao omogućiti lakšu pohranu podataka o strankama servisa i njihovim vozilima, praćenje, otvaranje novih te arhiviranje izvršenih radnih naloga, automatsko ažuriranje i uvid u financije auto-servisa te stanje skladišta. Također, sustav bi trebao biti dizajniran tako da bude jednostavan za korištenje svim zaposlenicima ovisno o njihovim ulogama.

Auto-servis mora imati barem jednog mehaničara, barem jednog električara, jednog voditelja servisa i jednog direktora. Iako su direktor i voditelj različite uloge, obje mogu biti dodijeljene istoj fizičkoj osobi. Osim djelatnika, auto-servis ima i skladište.

Svi djelatnici su ujedno i korisnici prodajnog informacijskog sustava, ali nemaju svi jednaka prava pristupa podacima i dozvoljenim funkcijama. S obzirom na prava pristupa dijelimo ih na:

servisnog savjetnika (u ovu skupinu spadaju svi mehaničari i električari),

direktora,

voditelja.

Sustav treba biti oblikovan tako da se svi korisnici računalnog sustava prijave u isti sa svojim korisničkim imenom i lozinkom te da se korisničko sučelje prilagođava njihovim pravima pristupa.

Voditelj je jedini koji ima pravo dodavanja i izmjene određenih podataka, dok servisni savjetnici i direktor mogu samo čitati podatke u skladu s njihovim ulogama.

Prilikom zaprimanja stranke, voditelj servisa otvara novi radni nalog. U slučaju da podaci o stranci i njenom vozilu već postoje u bazi, te podatke treba dodati radnom nalogu. U protivnom, voditelj preko sučelja unosi u bazu podataka ime i prezime stranke, kontakt telefon i adresu, a za vozilo stranke unosi model, godište i registarsku oznaku. Nužan je unos svih podataka osim adrese stranke.

Radni nalozi imaju svoju jedinstvenu šifru koju sustav automatski generira, datum i vrijeme nastanka i završetka, te status: u tijeku („otvoren“), storniran, uspješno završen („zatvoren“). Također, radni nalog mora sadržavati opis kvara ili postupaka na vozilu kojeg stranka traži, dijelove koji se mijenjaju te mora biti naveden jedan odgovorni servisni savjetnik.

Sustav trajno pohranjuje podatke o strankama koje su bile u servisu i njihovim vozilima te radne naloge koji su na vozilima već izvršeni.

Jedino voditelj može otvarati, zatvarati i stornirati radni nalog. Otvoreni radni nalog može mijenjati ili brisati postojeće te dodavati nove servisne postupke i rezervne dijelove.

Voditelj i direktor na upit mogu dobiti uvid u sve radne naloge, dok servisni savjetnik na upit dobiva uvid u samo one radne naloge na kojima je naveden kao odgovorni servisni savjetnik.

Sustav mora automatski voditi računa o stanju skladišta, odnosno koliko rezervnih dijelova servis u nekom trenutku ima i kojeg su tipa. Kada se neki rezervni dio pridruži radnom nalogu onda se njegovo stanje na skladištu automatski smanji za jedan. Za svaki pojedini rezervni dio koji pripada servisu sustav pamti jedinstvenu šifru, opis i cijenu.

Podatke o rezervnim dijelovima može unositi samo voditelj servisa, a mehaničari samo čitati. Svi djelatnici servisa mogu vidjeti izvješće o stanju skladišta.

Nad autima stranaka izvršavaju se različiti postupci servisa. Neki od njih su: pregledi, računalna dijagnostika, mehanički popravci, električni popravci, lakiranje i zamjena dijelova. Svi postupci servisa moraju biti unaprijed definirani u sustavu te svaki mora imati svoj opis, broj potrebnih rezervnih dijelova nekog tipa, minimalni broj djelatnika potrebnih za postupak i trajanje postupka. Sustav pohranjuje još i jediničnu cijenu *čovjek-sata* kao cijenu jednog sata rada pojedinog radnika na postupku. Iz tog podataka lako se računa cijena postupka kao: jedinična cijena *čovjek-sata* \* minimalni broj djelatnika \* trajanje postupka u satima.

Nakon završetka svih radova na vozilu, odnosno završetka svih servisnih postupaka navedenih u otvorenom radnom nalogu, sustav automatski računa ukupnu cijenu radnog naloga temeljem jedinične cijene *čovjek-sata* koja je pohranjena u bazi, ukupnog broja potrošenih radnih sati i cijene svih potrošenih dijelova.

Nakon što stranka plati izvršene radove i voditelj zatvori radni nalog, sustav odmah računa i ažurira dnevnu bilancu servisa.

Sustav mora automatski na kraju radnog vremena pokrenuti dnevni obračun, na kraju mjeseca mjesečni obračun koji je zbroj svih dnevnih obračuna tog mjeseca, a na kraju godine godišnji obračun kao zbroj svih mjesečnih obračuna te godine.

Samo direktor ima uvid u dnevni, mjesečni i godišnji promet, odnosnu bilancu servisa preko sučelja.

Sustav dizajniran na ovakav način, tako da se dio posla obavlja automatski i da se djelatnicima dodjeljuju različite uloge s ograničenim funkcionalnostima, bi trebao smanjiti vrijeme učenja sustava, olakšati vođenje cijelog auto-servisa te ga učiniti učinkovitijim.

# Rječnik pojmova

**Acts-as-taggable-on** je Ruby *gem* koji automatski implementira mogućnost označavanja sadržaja i pretraživanja po oznakama. Svrha ovog *gema* je olakšati pretraživanje korisniku.

**Ancestry**  je Ruby *gem* koji nudi mogućnost organizacije RoR objekata u stabla.

**Bcrypt** je Ruby *gem* koji nudi mogućnosti enkripcije niza znakova. Bcrypt je Ruby implementacija *crypt* algoritma enkripcije koji se koristi u svim *unix*-baziranim operativnim sustavima za enkripciju lozinki. Najveća prednost mu je što su rezultat enkripcije dva niza znakova (*eng. String*), *hash* i *salt*. Ako spremamo njih ne moramo nikada direktno spremati korisnikovu lozinku na server, niti ju možemo dekriptirati ako ne znamo i *hash* i *salt*. Ali možemo ponoviti proces enkripcije i provjeriti da se tako enkriptirana lozinka poklapa s onom u bazi.

**Carierwave:** Carierwave je Ruby *gem* koji omogućuje automatsku interpretaciju *multipart* podataka u RoR kontrolerima.

**CKEditor** je JS biblioteka i Ruby *gem* za grafički editor HTML-a. Nudi punu funkcionalnost programa poput Microsoft Worda. U projektu koristimo *fork* od jednog od članova ekipe jer nudi veću funkcionalnost.

**CoffeeScript** je skriptni jezik koji se transkompajlira u JS. Nudi funkcionalnost identičnu JavaScriptu. Coffee datoteke završavaju ekstenzijom .*coffee*.

***Cascading Style Sheet*** *(hrv. Kaskadna Lista Stilova)*skraćenoCSS, je prezentacijski jezik, odnosno služi za određivanje stila prikaza elemenata u *markup* jezicima poput HTML-a, XHTML-a, SVG-a ili XML-a. Jezik razlikuje elemente, razrede i identifikatore. Kod stiliziranja elemenata ne koristi se nikakav prefiks, za razrede se koristi prefiks točka (.*class*), a za identifikatore prefiks ljestve (#*identifikator*). Također je dozvoljeno korištenje regularnih izraza. Datoteke zapisane u CSS formatu završavaju ekstenzijom .*css*.

**FriendlyID**, je Ruby *gem* koji mijenja uobičajenu paradigmu pristupa sadržaju u RoR aplikacijama. Umjesto da se sadržaj poziva tako da se nakon URL adrese upiše ime kontrolera i zatim identifikacijski broj unosa (npr. *www.fer.hr/stranice/4*), FriendlyID nudi mogućnost da se umjesto identifikacijskog broja navede niz znakova koji se naziva “*slug*” (npr. *www.fer.hr/stranica/moja-stranica*). Primarna svrha ovog *gema* je ponuditi korisniku pamtljive URL adrese.

***HyperText Markup Language*** *(hrv. Prezentacijski Jezik za Izradu Webstranica), skraćeno*HTML je *markup* jezik, odnosno služi za stilizaciju tekstualnih elemenata. Primarna uporaba mu je za prikazivanje teksta na web stranicama. Tekstualni sadržaj se upisuje u HTML oznake (eng. *tagove*), koje se upisuju u streličaste zagrade (*< >*) koje uglavnom dolaze u parovima, pri čemu drugi služi kao zatvarajuća oznaka slična otvarajućoj, ali odmah nakon prve streličaste zagrade sadrži kosu crtu (</ >). Datoteke zapisane u HTML formatu završavaju ekstenzijom *.html* ili .*htm*.

**Image Magic**, je jako opsežna C biblioteka za manipulaciju slika. Jedina mana joj je što ulazni i izlazni podaci moraju biti datoteke. Međutim, za svrhe ovog projekta to će biti zadovoljavajuće.

**JavaScript**, skraćenoJS je skriptni jezik koji se izvodi na klijentovom računalu. Omogućuje asinkrono učitavanje podataka te manipulaciju DOM (*eng. Document Object Model, hrv. Model Objekata Dokumenta*) elementima učitanih iz HTML datoteke. JavaScript datoteke završavaju ekstenzijom .*js*.

**jQuery**, skraćeno jQ, je JavaScript biblioteka koja nudi gotove metode za manipulaciju DOM (*eng. Document Object Model, hrv. Model Objekata Dokumenta*) elemenata učitanih iz HTML datoteke. Njena najveća prednost je što osigurava kompatibilnost koda na svim internet preglednicima.

**MiniMagick** je Ruby *gem* koji mapira funkcionalnost Image Magicka u Ruby. Za razliku od RMagicka MiniMagick ne implementira funkcionalnost Image Magicka nego poziva Image Magick kroz *shell (hrv. ljusku operacijskog sustava)*. Najveća prednost mu je što nudi objektno orijentirani pristup manipulaciji slika. Ovisan je o Image Magick biblioteci.

**MySQL** je jedna od najpopularnijih SQL (*eng. Structured Query Language, hrv. Strukturirani Jezik za Upite*) baza podataka. Nudi gotovo potpunu SQL implementaciju. U projektu se koristi Ruby *gem* po imenu MySQL koji omogućuje RoR aplikacijama da pišu SQL bazu podataka u datoteku ili na MySQL server. U projektu MySQL koristimo kao sekundarnu (*eng. fallback*) bazu podataka.

**Node.js** je program dizajniran za stvaranje skalabilnih mrežnih aplikacija. Napisan je u JS-u koji se izvodi na serveru, ali može rezultat svoje izvedbe *pushati (hrv. gurnuti)* na korisnikov *JS stack (hrv. stog)*. Unutar ovog projekta nema direktnu uporabu, ali ga RoR koristi interno za serverske funkcije te za interpretaciju JS-a, CSS-a i slika unutar svog *pipelinea (hrv. cijevovoda)*.

**NProgress** je jQuery biblioteka koja nudi metode za grafički prikaz napretka učitavanja podataka pozvanih preko AJAX (*eng. Asynchronus JavaScript and XML, hrv. Asinkrona JavaScripta i XML*) poziva, slično traci za punjenje na YouTubeu. Konkretno, u projektu koristimo *fork* od jednog od članova tima jer nudi veću funkcionalnost od originalne biblioteke.

**Postgress SQL** je druga najpopularnija SQL (*eng. Structured Query Language, hrv. Strukturirani Jezik za Upite*) baza podataka. Nudi gotovo potpunu SQL implementaciju i dodatna proširenja na SQL standard poput spremanja polja i matrica. Najveća prednost nad MySQL-om je manja potrošnja memorije te dodatna funkcionalnost. Za postizanje komunikacije između RoR aplikacije i Postgress baze koristimo Ruby *gem* po imenu “PG” koji omogućuje razgovor RoR aplikacije i Postgress servera (*hrv. poslužitelja*), ali ne nudi zapis u datoteku.

**RMagick** je Ruby *gem (hrv. dragulj)* koji implementira većinu funkcionalnosti Image Magic biblioteke u Ruby. CKEditor ovisi o ovome *gemu*.

**Ruby** je objektno orijentiran, dinamički jezik razvijen 1990. godine. Podržava objektno orijentiranu, imperativnu i funkcijsku paradigmu. Ima implementirani automatski *Garbage collector* (*hrv. automatsko oslobađanje memorije*). Ruby ima sintaksu sličnu Perlu, ali nudi funkcionalnosti jezika Smalltalk (Objective-C). Do verzije 1.8.7 se koristio *single-pass (hrv. jednoprolazni) interpreter* napisan u C-u, a od verzije 1.9.3 na dalje se koristi *interpreter* baziran na YARV-u (*eng. Yet Another Ruby VM, hrv. Još Jedan Ruby Virtualni Stroj*) koji ga je performansama približio Pythonu. Važno je za napomenuti da se Ruby biblioteke nazivaju *gemovima (hrv. draguljima)*. Datoteke zapisane u Ruby formatu završavaju ekstenzijom .*rb*.

**Ruby on Rails**, skraćeno *RoR*, je Ruby (*hrv. rubin*) *gem (hrv. dragulj)* koji dodaje funkcionalnosti web servera (*hrv. poslužitelja*) te mogućnost dinamičkog dodavanja sadržaja u datoteke statičnog sadržaja. Razredi i kontroleri za RoR se pišu u Rubyju dok se datoteke kojima će se dinamički dodavati sadržaj pišu u jeziku te datoteke, npr. u HTML (*eng. Hyper Text Markup Language, hrv. Prezentacijski Jezik za Izradu Webstranica*) datoteke će se upisivati čisti HTML kod. Ako želimo da se u nekoj datoteci dinamički mijenja sadržaj moramo joj dodati ekstenziju .*erb* nakon ekstenzije asocirane uz jezik u kojemu je datoteka zapisana. Tako će npr. .*html* datoteka postati .*html.erb* datoteka. RoR omogućuje izvođenje dinamičkog Ruby koda u datotekama tako da ga se upiše u streličaste zagrade popraćene znakom postotka (*<% %>*). Ako želimo da se ispiše rezultat interpretacije Ruby koda tada moramo još nakon početne zagrade napisati znak jednakosti (*<%= %>*).

**Turbolinks** (*hrv. Turbo Poveznica*) je jQuery biblioteka i Ruby *gem (hrv. dragulj)* kojemu je primarna svrha ubrzavanje učitavanja sadržaja web stranice. Igra ulogu posrednika razgovora između klijentovog preglednika i servera (*hrv. poslužitelja*). Pošto je napisana u JavaScriptu, izvodi se na korisnikovom računalu unutar preglednika te tako rasterećuje server. Samo učitavanje ubrzava tako da pri svakoj promjeni URL (*eng. Uniform Resource Locator, hrv. Uniformni Pronalazitelj Dobara*) adrese putem klika na neku poveznicu, umjesto da učita cijelu stranicu ponovno, učita samo onaj dio koji se promijenio.

**Will\_paginate** je Ruby *gem (hrv. dragulj)* za automatsko dijeljenje sadržaja na stranice. Svrha ovog *gema* je smanjenje vremena učitavanja stranice tako da se sadržaj razlomi na više stranica.

# Funkcionalni zahtjevi

Funkcionalni zahtjevi opisuju kako se sustav treba ponašati na određenu pobudu, koje usluge mora pružati i kako se treba ponašati u određenim situacijama. Definira se interakcija sa dionicima (osobe s interesom ili nositelji odgovornosti) i aktorima (osobe koje izravno koriste ili komuniciraju sa sustavom). Pojedine situacije su opisane obrascima uporabe.

**Dionici:**

* Korisnici:

- Servisni savjetnik

- Voditelj

- Direktor

**Aktori i njihovi funkcionalni zahtjevi:**

* Korisnik, inicijator:
  + Može se prijaviti na sustav
  + Pregledava trenutno stanje rezervnih dijelova na skladištu.
  + Pregledava radne naloge.
* Baza podataka, sudionik:
  + Čuva podatke o nalozima, rezervnim dijelovima, popravcima, strankama i vozilima.
* Voditelj, inicijator:
  + Unosi u bazu nove radne naloge ili mijenja postojeće.
  + Unosi u bazu nove rezervne dijelove ili mijenja postojeće.
  + Zatvara i stornira radne naloge.
* Direktor, inicijator:
  + Pregledava dnevni, mjesečni i godišnji obračun servisa.
* Brojilo vremena, inicijator:
  + Periodički aktivira obavljanje obračuna (na kraju radnog vremena), te aktivira mjesečni i godišnji obračun
* UC1 – *PrijaviSe*:
  + **Glavni sudionik:** Korisnik.
  + **Cilj:** Prijaviti se u sustav.
  + **Sudionici:** Baza podataka.
  + **Preduvjeti:** Nije već prijavljen.
  + **Rezultat:** Korisnik je prijavljen u sustav i može koristiti funkcije sustava u skladu sa svojim ovlastima.
  + **Željeni scenarij:**
    1. Korisnik upisuje korisničko ime, lozinku i šalje ih sustavu.
    2. Sustav provjerava podatke u bazi i daje prijavljenom korisniku odgovarajuće ovlasti.
  + **Mogući drugi scenariji:**

1. Korisničko ime i lozinka nisu pronađeni u bazi.
   * + 1. Obavijesti korisnika da su podaci za prijavu neispravni.

* UC2 – *DohvatiStanjeSkladišta*:
  + **Glavni sudionik:** Korisnik
  + **Cilj:** Provjeriti trenutno stanje rezervnih dijelova u skladištu servisa.
  + **Sudionici:** Baza podataka.
  + **Preduvjeti:** Nema.
  + **Rezultat:** Korisnik dobiva trenutni ispis imena rezervnih dijelova i količine koliko tih dijelova ima u skladištu.
  + **Željeni scenarij:**

1. Korisnik odabire da želi pregled sadržaja skladišta.
2. Sustav iz baze podataka vraća korisniku ispis trenutnog stanja rezervnih dijelova u skladištu.

* UC3 – *PregledPodaciRezervniDijelovi*:
  + **Glavni sudionik:** Voditelj ili mehaničar.
  + **Cilj:** Dobiti uvid u sve informacije vezane uz rezervne dijelove.
  + **Sudionici:** Baza podataka.
  + **Preduvjeti:** Nema.
  + **Rezultat:** Voditelj ili mehaničar dobiva informacije o rezervnim dijelovima.
  + **Željeni scenarij:**

1. Voditelj ili mehaničar šalje upit sustavu za željene informacije.
2. Sustav pretražuje bazu podataka i daje korisniku informacije o rezervnim dijelovima.

* UC4 – *IzmjeniRezervneDijelove*:
  + **Glavni sudionik:** Voditelj.
  + **Cilj:** Unijeti nove rezervne dijelove ili editirati stare.
  + **Sudionici:** Baza podataka.
  + **Preduvjeti:** Došli su novi rezervni dijelovi ili treba promijeniti stare.
  + **Rezultat:** Novi rezervni dijelovi ili promjene na starima su unesene u bazu podataka.
  + **Željeni scenarij:**

1. Voditelj zatraži od sustava obrazac da unese nove rezervne dijelove ili editira stare.
2. Sustav mu vraća potrebni obrazac.
3. Voditelj ispunjava obrazac i šalje ga na pohranu u bazu podataka.

* UC5 – *OtvoriRadniNalog*:
  + **Glavni Sudionik:** Voditelj.
  + **Cilj:** Otvoriti novi radni nakog.
  + **Sudionici:** Baza podataka.
  + **Preduvjeti:** Voditelj je zaprimio stranku.
  + **Rezultat:** Voditelj je otvorio novi radni nalog sa svim potrebnim stavkama.
  + **Željeni scenarij:**

1. Voditelj provjerava ima li podatke o stranci i vozilu u bazi podataka.
2. Ako ima voditelj pridružuje stranku i vozilo bazi podataka.
3. Voditelj pridružuje servisnog savjetnika kao odgovornog servisnog savjetnika za taj radni nalog
4. Voditelj pridružuje postupke i rezervne dijelove radnom nalogu.
5. Umanji se stanje rezervnih dijelova na skladištu.
6. Radni nalog se sprema u bazu sa statusom „otvoren“.
   * **Mogući drugi scenariji:**
7. Ako u bazi podataka nema podataka o stranci ili vozilu.
   * + 1. Voditelj zatraži od sustava odgovarajući obrazac ili obrasce da unese novu stranku ili novo vozilo.
       2. Sustav mu vraća potrebni obrazac.
       3. Voditelj ispunjava obrazac i šalje ga na pohranu u bazu podataka.
       4. Nastavlja se od 3 stavke željenog scenarija.
8. Ako na skladištu nema odgovarajućih dijelova.
   * + 1. Radni nalog je storniran i pohranjen u bazu podatka.

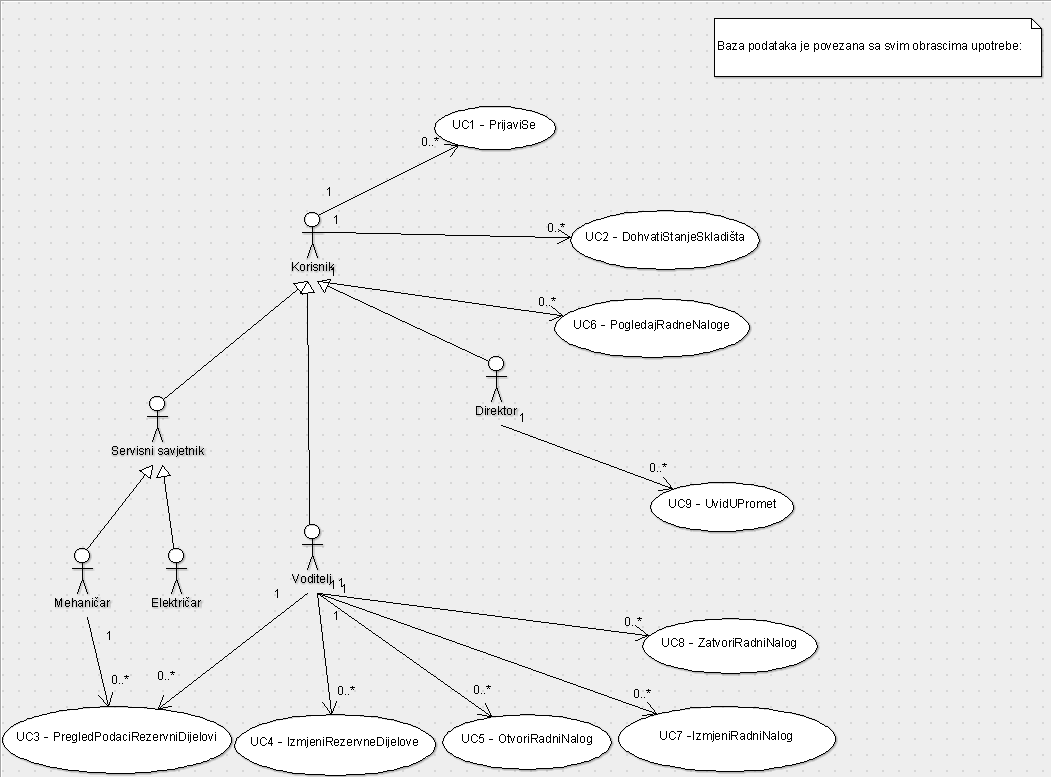
* UC6 – *PogledajRadneNaloge*:
  + **Glavni Sudionik:** Korisnik.
  + **Cilj:** Pregled radnih naloga.
  + **Sudionici:** Baza podataka.
  + **Preduvjeti:** Nema.
  + **Rezultat:** Korisnik je dobio informacije o radnim nalozima pohranjenim u arhivu.
  + **Željeni scenarij:**
    1. Voditelj ili direktor ili servisni savjetnik šalje upit sustavu za željene informacije.
    2. Sustav pretrazuje bazu i vraća direktoru i voditelju sve radne naloge, a servisnom savjetniku samo one na kojima je naveden kao odgovorni servisni savjetnik.
* UC7 – *IzmjeniRadniNalog*:
  + **Glavni Sudionik:** Voditelj
  + **Cilj:** Dodavanje mjenjanje ili brisanje postupaka i dijelova u radnom nalogu.
  + **Sudionici:** Baza podataka
  + **Preduvjeti:** Potreba za promjenom u radnom nalogu.
  + **Rezultat:** Promjenjen je status prethodno storniranom radnom nalogu u „otvoren“.
  + **Željeni scenarij:**

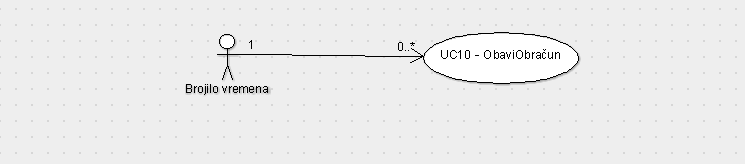
1. Voditelj otvara prethodno otvoreni ili stornirani radni nalog.
2. Voditelj editira postupke i rezervne dijelove pridružene radnom nalogu.
3. Umanji se stanje rezervnih dijelova na skladištu.
4. Radni nalog se sprema u bazu sa statusom „otvoren“.
   * **Mogući drugi scenariji:**
5. Ako na skladištu nema odgovarajućih dijelova.
6. Radni nalog je storniran i pohranjen u bazu podatka.

* UC8 – *ZatvoriRadniNalog*:
  + **Glavni Sudionik:** Voditelj
  + **Cilj:** Zatvoriti radni nalog i naplatiti.
  + **Sudionici:** Baza podataka
  + **Preduvjeti:** Završetak radova na vozilu.
  + **Rezultat:** Radni nalog je zatvren i pohranjen u bazi podatka a dnevna bilanca je uvećana.
  + **Željeni scenarij:**

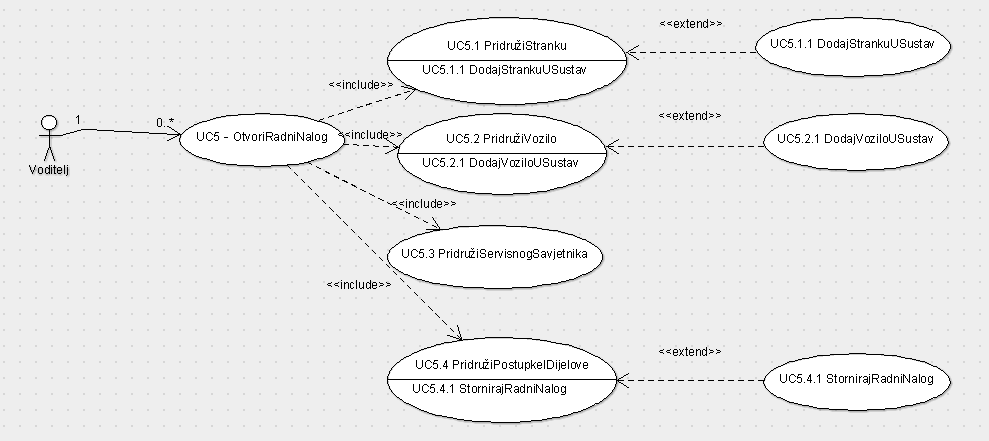
1. Voditelj šalje upit za izračun cijene.
2. Sustav računa cijenu i ispisuje voditelju.
3. Voditelj naplati radove.
4. Voditelj zatvara radni nalog tako da mjenja status radnog naloga u „zatvoren“.
5. Sustav automatski izračunava i ažurira dnevnu bilancu servisa.
   * **Mogući drugi scenariji:**
6. Voditelj ne može trenutno naplatiti radove.
7. Prekida se zatvaranje radnog radnog naloga.

* UC9 – *UvidUPromet*:
  + **Glavni Sudionik:** Direktor
  + **Cilj:** Uvid u dnevni, mjesečni i godišnji promet.
  + **Sudionici:** Baza podataka.
  + **Preduvjeti:** Nema.
  + **Rezultat:** Direktor dobiva sve informacije o bilanci servisa.
  + **Željeni scenarij:**
    1. Direktor šalje upit sustavu za željene informacije.
    2. Sustav pretražuje bazu podataka i daje direktoru informacije o bilanci servisa.
* UC10 – *ObaviObračun*:
  + **Glavni Sudionik:** Brojilo vremena.
  + **Cilj:** Obaviti obračun.
  + **Sudionici:** Baza podataka.
  + **Preduvjeti:** Kraj radnog vremena.
  + **Rezultat:** Obavljen je obračun.
  + **Željeni scenarij:**
    1. Broilo vremena aktivira automatski izračun dnevnog obračuna.
    2. Po potrebi se aktivira mjesečni i godišnji obračun.





**Slika 4.1** Obrazac upotrebe za UC1 do UC10

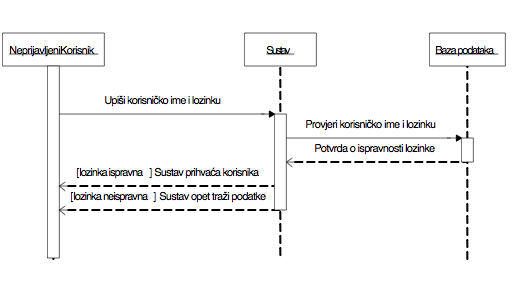


**Slika 4.2**. Obrazac upotrebe za UC5

**Sekvencijski dijagrami:**

Obrazac uporabe UC1: (*PrijaviSe*)

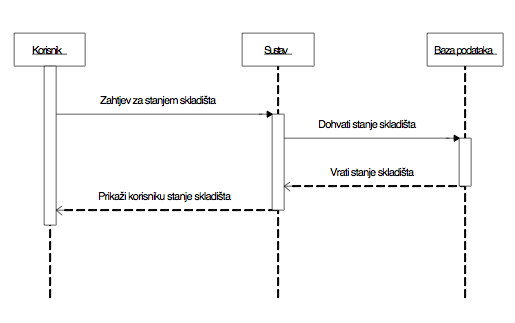
Neprijavljeni korisnik se prijavljuje u sustav svojim korisničkim imenom i lozinkom. Sustav provjerava podatke u bazi, prijavljuje korisnika te on dobiva određena prava ovisno o razini ovlasti koju ima. Ukoliko je neki od unesenih podataka neispravan, sustav će obavijestiti korisnika da su podaci za prijavu netočni te će ponovno tražiti podatke.



**Slika 4.3**. Sekvencijski dijagram za UC1 (*PrijaviSe*)

Obrazac uporabe UC2: (*DohvatiStanjeSkladišta*)

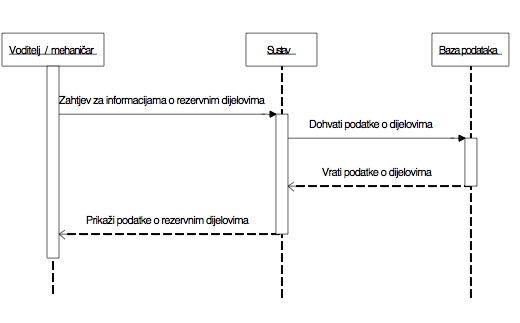
Prijavljeni korisnik odabire opciju provjere stanja skladišta. Sustav iz baze dohvaća podatke o stanju te ih prikazuje korisniku.



**Slika 4.4.** Sekvencijski dijagram za UC2 (*DohvatiStanjeSkladišta*)

Obrazac uporabe UC3: (*PregledPodaciRezervniDijelovi*)

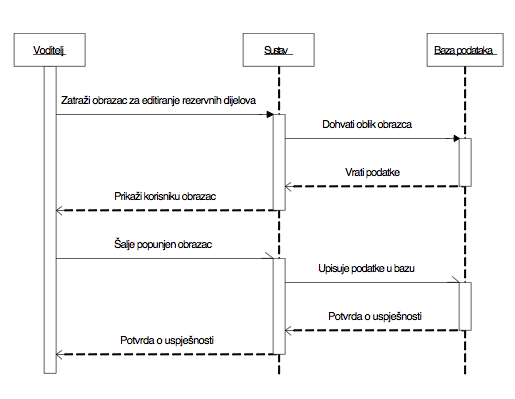
Prijavljeni korisnik odabire opciju provjere stanja rezervnih dijelova. Sustav iz baze dohvaća tražene podatke o dijelovima te ih prikazuje korisniku.



**Slika 4.5.** Sekvencijski dijagram za UC3 (*PodaciRezervniDijelovi*)

Obrazac uporabe UC4: (*IzmjeniRezervneDijelove*)

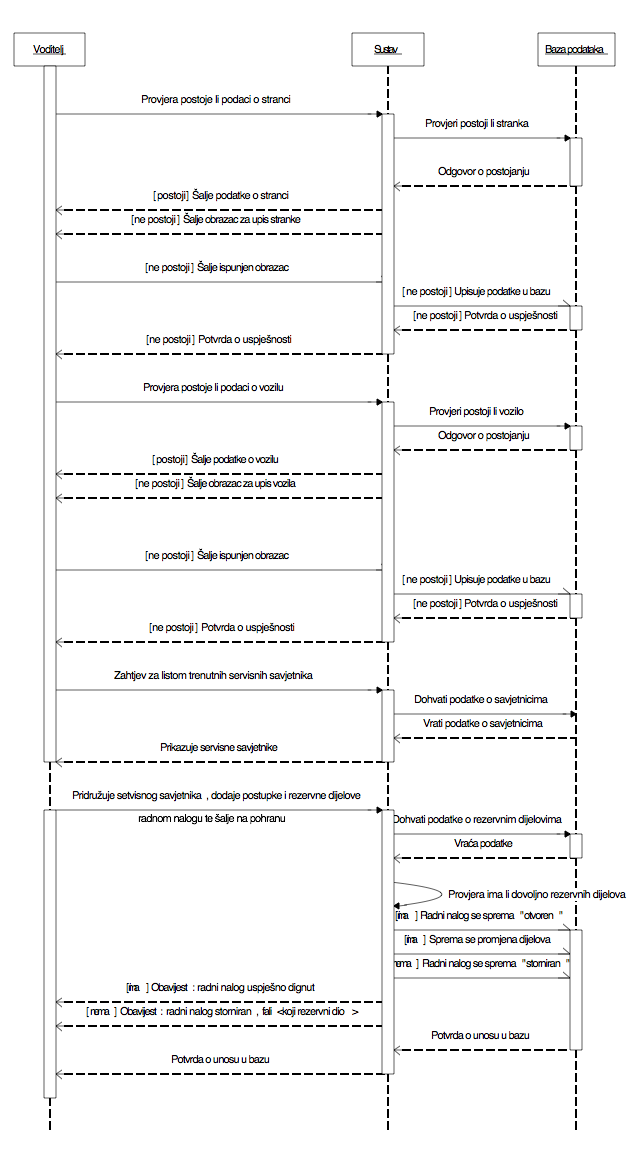
Voditelj odabere opciju mijenjanja postojećih rezervnih dijelova te zatraži obrazac. Sustav iz baze dohvaća traženi oblik obrasca te ga prikazuje voditelju. Voditelj zatim ispunjava obrazac, te odabere spremanje podataka. Sustav upisuje unesene podatke o rezervnim dijelovima u bazu podataka te po spremanju dobiva potvrdu o uspješnosti koja se prikaže voditelju.



**Slika 4.6.** Sekvencijski dijagram za UC4 (*IzmjeniRezervneDijelove)*

Obrazac uporabe UC5: (*OtvoriRadniNalog*)

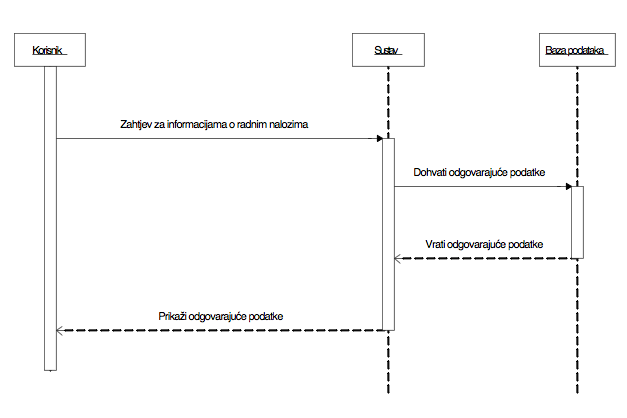
Voditelj servisa prilikom otvaranja radnog naloga prvo provjerava da li je stranka već upisana tj. da li je već bila u servisu. Sustav dohvaća traženi podatak iz baze te, ukoliko stranka nije pronađena u bazi, dostavlja obrazac za upis stranke, a ukoliko postoji, prikazuje podatke o stranci. Ukoliko stranka nije pronađena u bazi, voditelj ispunjava obrazac o stranci te sustav upisuje podatke u bazu. Po spremanju podataka, voditelj dobiva potvrdu o uspješnosti. Zatim voditelj provjerava postoje li već u bazi upisani podaci o vozilu stranke. Ukoliko ih nema, sustav šalje obrazac za upis vozila u bazu, a inače prikazuje već upisane podatke o vozilu iz baze. Nakon što voditelj ispuni obrazac za upis vozila koje je po prvi put u servisu, šalje ga sustavu koji upisuje navedene podatke u bazu i po završetku prikazuje voditelju potvrdu o uspješnosti. Nadalje, voditelj treba odabrati odgovornog servisnog savjetnika za taj radni nalog, te šalje zahtjev sustavu koji mu iz baze vraća podatke o trenutnim servisnim savjetnicima. Voditelj iz popisa odabere servisnog savjetnika te prema dogovoru sa strankom dodaje postupke koji će se činiti na vozilu te rezervne dijelove koji će se iskoristiti u radnom nalogu te šalje na pohranu sustavu. Sustav dohvaća podatke o rezervnim dijelovima i provjerava ima li ih dovoljno. Ako dijelova ima dovoljno, sprema se njihova promjena (smanjuje im se broj na skladištu) te se radni nalog sprema u bazu kao „otvoren“ i prikazuje se obavijest o uspješno otvorenom radnom nalogu. Ukoliko dijelova nema dovoljno, nalog se u bazu upisuje kao „storniran“ te se prikazuje obavijest o tome, te kojih dijelova nedostaje na skladištu. Baza podataka također potvrđuje voditelju da je sve uspješno spremljeno.



**Slika 4.7**. Sekvencijski dijagram za UC5 (*OtvoriRadniNalog)*

Obrazac uporabe UC6: (*PogledajRadneNaloge*)

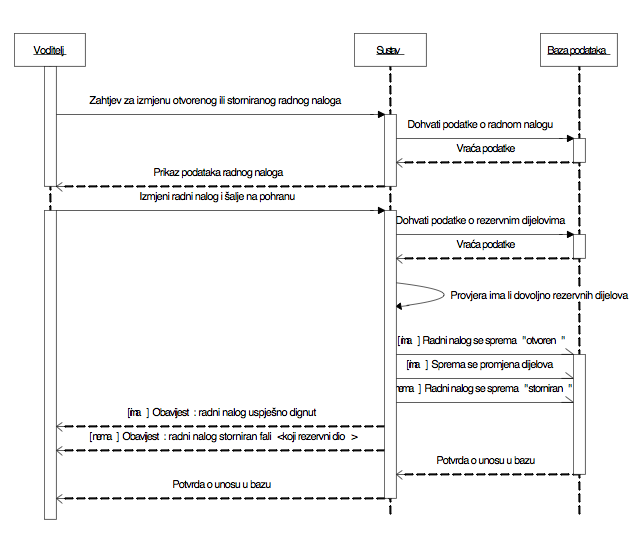
Korisnik šalje sustavu zahtjev za informacijama o radnim nalozima. Sustav, u ovisnosti o razini korisnika (servisni savjetnik ili voditelj), dohvaća odgovarajuće podatke – za voditelja su to svi radni nalozi, a za servisne savjetnike samo oni kojima su pridruženi.



**Slika 4.8.** Sekvencijski dijagram za UC6 (*PogledajRadneNaloge)*

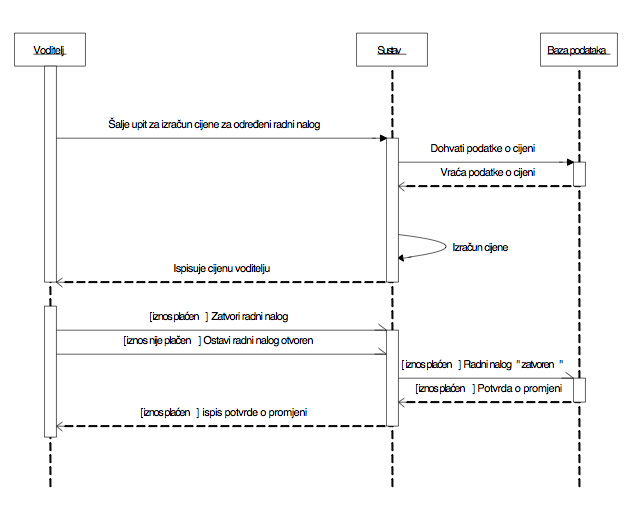
Obrazac uporabe UC7: (*IzmjeniRadniNalog*)

Voditelj šalje sustavu zahtjev za izmjenu otvorenog ili storniranog radnog naloga, sustav iz baze dohvaća podatke o traženom nalogu te ih prikazuje voditelju. Voditelj zatim mijenja nalog i šalje sustavu na pohranu. Sustav iz baze dohvaća podatke o rezervnim dijelovima unijetim u mijenjani radni nalog te provjerava ima li dovoljno traženih rezervnih dijelova. Ako ima, nalog se sprema kao „otvoren“ , sprema se u bazu i promjenjen broj dijelova te se voditelja obavijesti o učinjenom. Ukoliko nema dovoljno rezervnih dijelova na skladištu, radni nalog se sprema kao „storniran“ te sustav obavijesti voditelja o tome, i koji rezervni dijelovi nedostaju. Na kraju, baza podataka sustavu šalje potvrdu da je sve spremljeno, što sustav dostavlja i voditelju.

**Slika 4.9.** Sekvencijski dijagram za UC7 (*IzmjeniRadniNalog)*

Obrazac uporabe UC8: (*ZatvoriRadniNalog*)

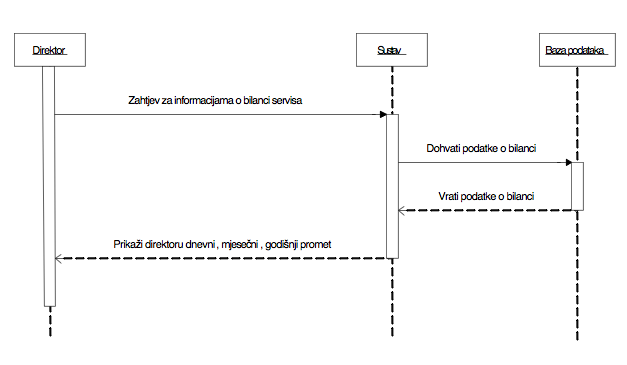
Voditelj prilikom zatvaranja radnog naloga šalje upit za izračun cijene. Sustav iz baze dohvaća podatke o postupcima, dijelovima, čovjek-satima na navedenom radnom nalogu, izračunava krajnju cijenu te ju ispisuje voditelju. Ukoliko je iznos plaćen, zatvara radni nalog čime se on sprema u bazu kao takav te baza šalje potvrdu o navedenoj promjeni. Ako iznos nije plaćen, radni nalog ostaje otvoren.



**Slika 4.10.** Sekvencijski dijagram za UC8 (*ZatvoriRadniNalog)*

Obrazac uporabe UC9: (*UvidUPromet*)

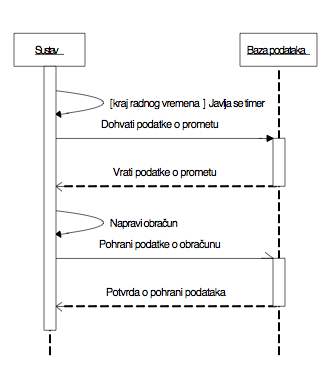
Direktor šalje sustavu zahtjev o uvidu u bilancu servisa, sustav dohvaća tražene podatke iz baze te prikazuje direktoru dnevni, mjesečni i godišnji promet.



**Slika 4.11.** Sekvencijski dijagram za UC9 (*UvidUPromet)*

Obrazac uporabe UC10: (*ObaviObračun*)

Sustav na kraju radnog vremena automatski (pomoću vremenskog brojila koje pokrene postupak) iz baze dohvati podatke o zatvorenim radnim nalozima te izračuna ukupnu dnevnu dobit. Taj podatak spremi u bazu i po završetku spremanja dobiva potvrdu o uspješnosti pohrane podatka.



**Slika 4.12.** Sekvencijski dijagram za UC10 (*ObaviObračun)*

# Ostali zahtjevi

* Sustav treba omogućiti rad više korisnika u stvarnom vremenu, ograničenje je na 20 korisnika.
* Sustav bi trebao jamčiti pouzdanost i točnost informacija.
* Sustav mora omogućiti prikaz hrvatskih dijakritičkih znakova.
* Aplikacija mora biti prilagođena izvedbi na Webu, stoga smo odabrali objektno-orijentiranu paradigmu i Ruby on Rails *framework*.
* Korištenje aplikacije mora biti jednostavno, tako da ju svi korisnici mogu koristiti bez posebnih uputa.
* Postupci servisa i jedinična cijena čovjek-sata mogu se ručno unijeti u bazu podataka te nije potrebno raditi sučelje za njihovo ažuriranje.
* Nije potrebno realizirati funkcionalnost kreiranja korisničkih profila.
* Za potrebe ispitivanja sustava, u sustavu se mora nalaziti barem 10 radnih naloga unutar 7 radnih dana s barem 5 stavki.

# Arhitektura i dizajn sustava

## Svrha, opći prioriteti i skica sustava

### Svrha i opći prioriteti sustava

Riječ je o prodajnom informacijskom sustavu koji omogućuje bolju organizaciju poslovanja unutar tvrtke. Najlakše ostvarenje navedenog je kroz vlastitu web stranicu i bazu podataka tvrtke, u skladu sa tim je implementirana **arhitektura web aplikacije**.

Pri slaganju arhitekture sustava posebna pažnja je posvećena hijerarhijskoj strukturi tvrtke koja zahtijeva da određene informacije budu dostupne samo određenim zaposlenicima. Kako bi se to postiglo, pojedinom zaposleniku je dodijeljena razina prava s obzirom na njegovu poziciju u tvrtki. Kako se radi o auto-servisu, sustav je napravljen sa ciljem praćenja dvije bitne stvari - dostupnost dijelova u skladištu i napredak pojedinih radnih naloga. Naglasak je na detaljnoj evidenciji radnih naloga i informacija povezanih s pojedinim nalogom, kao što su klijenti koji su ga naručili, zaposlenici koji su ga obavljali te dijelovi koji su potrebni za njegovo provođenje. Sve te informacije se trajno pohranjuju u bazu podataka da budu dostupne u svakom trenutku.

Kako se radi striktno o sustavu za organizaciju poslovanja, on nije predviđen za interakciju sa klijentima već su krajnji korisnici zaposlenici tvrtke. Naravno, grafičko sučelje je napravljeno da bude intuitivno te da omogućava što lakšu uporabu od strane zaposlenika. Detaljniji opis arhitekture slijedi u daljnjem tekstu.

Ovaj sustav možemo podijeliti na podsustave:

* Baza podataka
* Web aplikacija
* Poslužitelj

### 

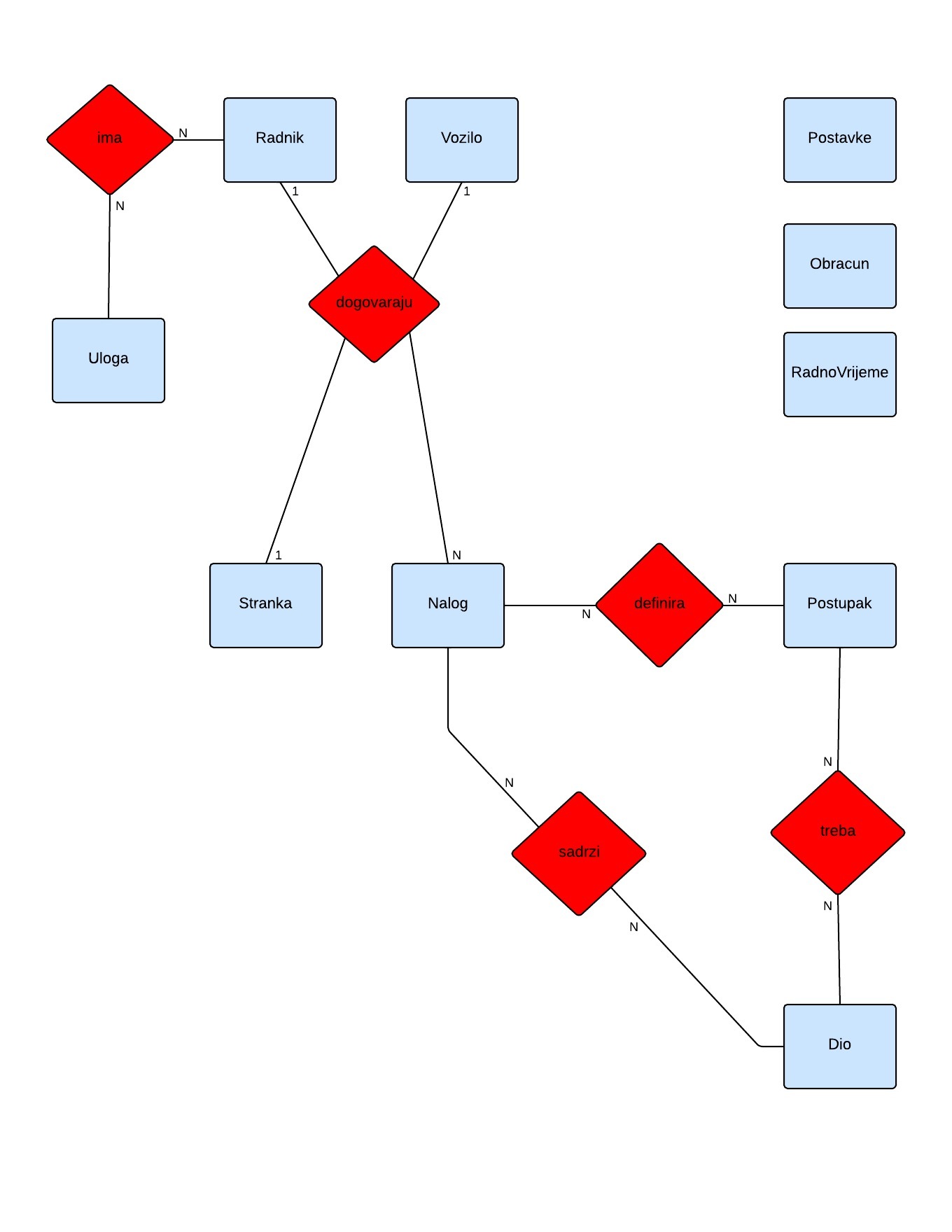
Slika 6.1 Arhitektura sustava

### Grafičko korisničko sučelje

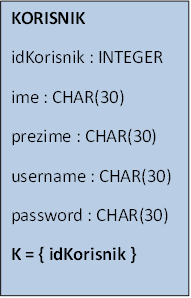
Kako bi se krajnjem korisniku omogućilo jednostavno korištenje svih funkcija sustava koje su mu potrebne za rad, implementirano je i grafičko korisničko sučelje (eng. Graphical User Interface – GUI). Elementi grafičkog sučelja su prilagođeni interakciji korisnika i sustava putem miša i tipkovnice na način koji im je razumljiv bez znanja o načinu implementacije bilo kojeg dijela sustava. Prikaz sučelja ovisi o vrsti korisnika koji mu pristupa. Neregistrirani korisnici imaju ograničenu mogućnost pregleda sadržaja. Registriranim korisnicima je nakon prijave u sustav omogućen pregled i izmjena sadržaja ovisno o dodijeljenoj razini prava. Glavni cilj pri dizajnu grafičkog sučelja je bio omogućiti izravan pristup i efikasno izvršavanje svih funkcija koje su određenom korisniku potrebne, bez zamaranja o dijelovima sustava koji mu nisu važni za rad.

### Baza podataka

Bazu podataka koristimo za pohranu, izmjenu i dohvat podataka te njihovo povezivanje u logičke cjeline. Za potrebe našeg sustava koristimo relacijski model baze podataka, koja se sastoji od skupa relacija. Svaka relacija svedena je na treću normalnu formu kako bi se izbjegla redundancija (zalihost) u sustavu. Kako će izgledati baza podataka može se vidjeti na priloženom ER-modelu (*entity-relationship* model). ER-model nudi nam detaljan ali lako razumljiv prikaz baze podataka. Entiteti su označeni pravokutnicima dok romb simbolizira relaciju. Sve relacije iz ER- modela prikazane su nakon ER-modela s popisima svih atributa i primarnim ključevima.



**Slika 6.1.** ER model baze podataka



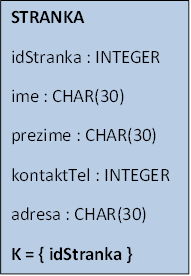
- šifra korisnika

- ime korisnika

- prezime korisnika

- korisničko ime za spajanje na sustav

- šifra koju korisnik koristi za spajanje na sustav



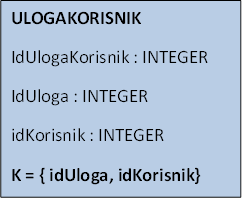
- šifra stranke

- ime stranke

- prezime stranke

- broj za kontakt

- adresa stranke (opcionalno)



**STRANKA**

idStranka : INTEGER

ime : CHAR(30)

prezime : CHAR(30)

kontaktTel : INTEGER

adresa : CHAR(30)

**K = { idStranka }**

**STRANKA**

idStranka : INTEGER

ime : CHAR(30)

prezime : CHAR(30)

kontaktTel : INTEGER

adresa : CHAR(30)

**K = { idStranka }**

**ULOGAKORISNIK**

IdUlogaKorisnik : INTEGER

IdUloga : INTEGER

idKorisnik : INTEGER

**K = { idUloga, idKorisnik}**

**ULOGAKORISNIK**

IdUlogaKorisnik : INTEGER

IdUloga : INTEGER

idKorisnik : INTEGER

**K = { idUloga, idKorisnik}**

- šifra korisnika s ulogom

- šifra uloge

- šifra korisnika



- šifra radnog naloga

- šifra stranke

- šifra vozila

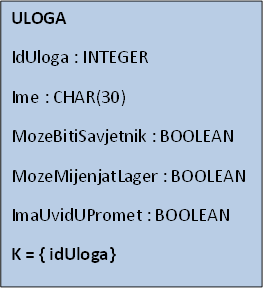
- opis kvara vozila

- vrijeme nastanka radnog naloga

- vrijeme zatvaranja radnog naloga

- status radnog naloga

- šifra korisnika



- šifra uloge

- ime uloge

- true ako može biti savjetnik, false ako ne

- true ako može mijenjati stanje lagera

- true ako ima uvid u promet

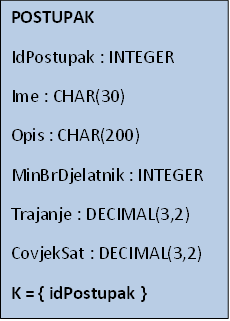


- šifra vozila

- model vozila

- godište vozila

- registracijska oznaka vozila



- šifra postupka

- ime postupka

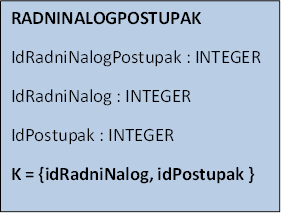
- opis postupka

- minimalni broj djelatnika za određeni postupak

- trajanje određenog postupka

- trajanje izraženo u jedinici čovjekSat

( minimalan broj ljudi\*trajanje u satima)



**RADNINALOGPOSTUPAK**

IdRadniNalogPostupak : INTEGER

IdRadniNalog : INTEGER

IdPostupak : INTEGER

**K = {idRadniNalog, idPostupak }**

**RADNINALOGPOSTUPAK**

IdRadniNalogPostupak : INTEGER

IdRadniNalog : INTEGER

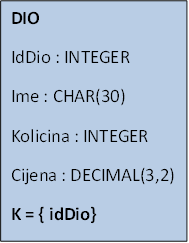
IdPostupak : INTEGER

**K = {idRadniNalog, idPostupak }**

- šifra radni nalog s postupkom

- šifra radnog naloga

- šifra postupka

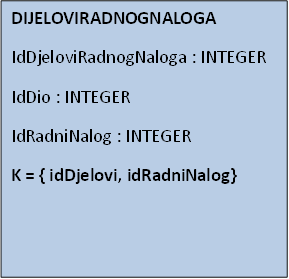


- šifra dijela

- ime dijela

- količina dijelova

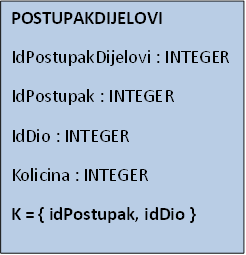
- cijena dijela



- šifra radni nalog s dijelovima

- šifra dijela

- šifra radnog naloga

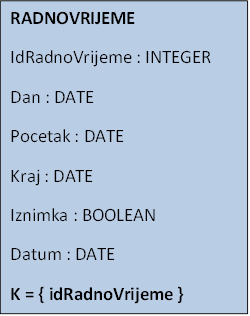


- šifra postupka s dijelovima

- šifra postupka

- šifra dijela

- količina dijela



- šifra radnog vremena

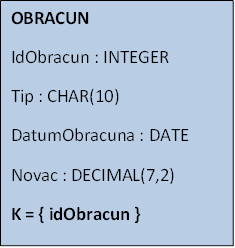
- dan u tjednu

- vrijeme početka rada

- vrijeme kraja rada

- true ako je iznimka tip blagdan, false ako nije

- datum iznimke



- šifra obračuna

- tip obračuna ( dnevni/ mjesečni/ godišnji )

- datum obračuna

- stanje novca nakon obračuna

## Dijagram razreda s opisom

**Kontoleri**

Kontroleri služe za dohvat odgovarajućih podataka iz modela te prikazivanje pogleda

povezanih uz danu akciju. U projektu imamo sljedeće kontrolere:

* **Application controller**

Glavni kontroler cijele aplikacije koji prikazuje *layout* u kojem se prikazuju ostali kontroleri. Korisnici i posjetitielji ne vide niti mogu utjecati na proces izvođenja ovog kontrolera te on sadrži metode koje moraju biti globalno dostupne. Osnovne funkcionalnosti ovog modela su: provjera je li korisnik prijavljen u sustav, vraćanje trenutnog korisnika, ispisivanje HTML tagova za Faviconu te ikonu za socijalne mreže, dohvaćanje URL-a Gravatar slike trenutnog korisnika i vraćanje punog naslova stranice.

* **Users controller**

Ovaj kontroler sadrži metode za stvaranje, brisanje, izmjenu i pregled korisnika u sustavu. On omogućuje postojećem korisniku, ako ima privilegije za to, da stvori, izmijeni ili pregleda profil drugog korisnika.

* **Categories controller**

Ovaj kontroler sadrži metode za stvaranje, brisanje, izmjenu i pregled kategorija. On kao kontroler kategorija ima prije svega kozmetičku svrhu te omogućuje grupiranje stranica u kategorije i registriranom korisniku nudi mogućnosti promjene, brisanja i stvaranja novih kategorija te prikaz stranica unutar neke kategorije. Pošto se kategorije obično organiziraju u strukture stabla model kategorija sadrži polje “ancestry” u koje *ancestry gem* upisuje organizaciju stabla.

* **Networks controller**

Ovaj kontroler sadrži metode za stvaranje, brisanje te izmjenu socijalnih mreža. Nema funkcionalnu, već kozmetičku svrhu te omogućuje dodavanje poveznica na socijalne mreže samo ovlaštenim korisnicima.

* **Statements controller**

Ovaj kontroler sadrži metode za stvaranje, brisanje, izmjenu i pregled prometa. Nudi opciju prikaza dnevnog, mjesečnog i godišnjeg prometa ovlaštenom korisniku.

* **Parts controller**

Sadrži metode za stvaranje, brisanje, izmjenu i pregled pojedinih dijelova.

* **Working Hours controller**

Sadrži metode za stvaranje, brisanje te izmjenu radnog vremena zaposlenika.

* **Roles controller**

Sadrži metode za stvaranje, brisanje te izmjenu uloga pojedinih zaposlenika tvrtke.

* **Sessions controller**

Sadrži metode za stvaranje i brisanje *session* kolačića u koje je upisan

autentifikacijski ključ.

* **Clients controller**

Sadrži metode za stvaranje, brisanje, izmjenu i pregled klijenata.

* **Procedures controller**

Sadrži metode za stvaranje, brisanje, izmjenu i pregled postupaka.

* **Pages controller**

Sadrži metode za stvaranje, brisanje, izmjenu i pregled stranica te nudi opciju pretraživanja stranica po njihovim tagovima. Dodatno još nudi opciju prikaza sakrivenih stranica i neobjavljenih stranica.

* **Settings controller**

Sadrži metode za izmjenu postavki.

* **Orders controller**

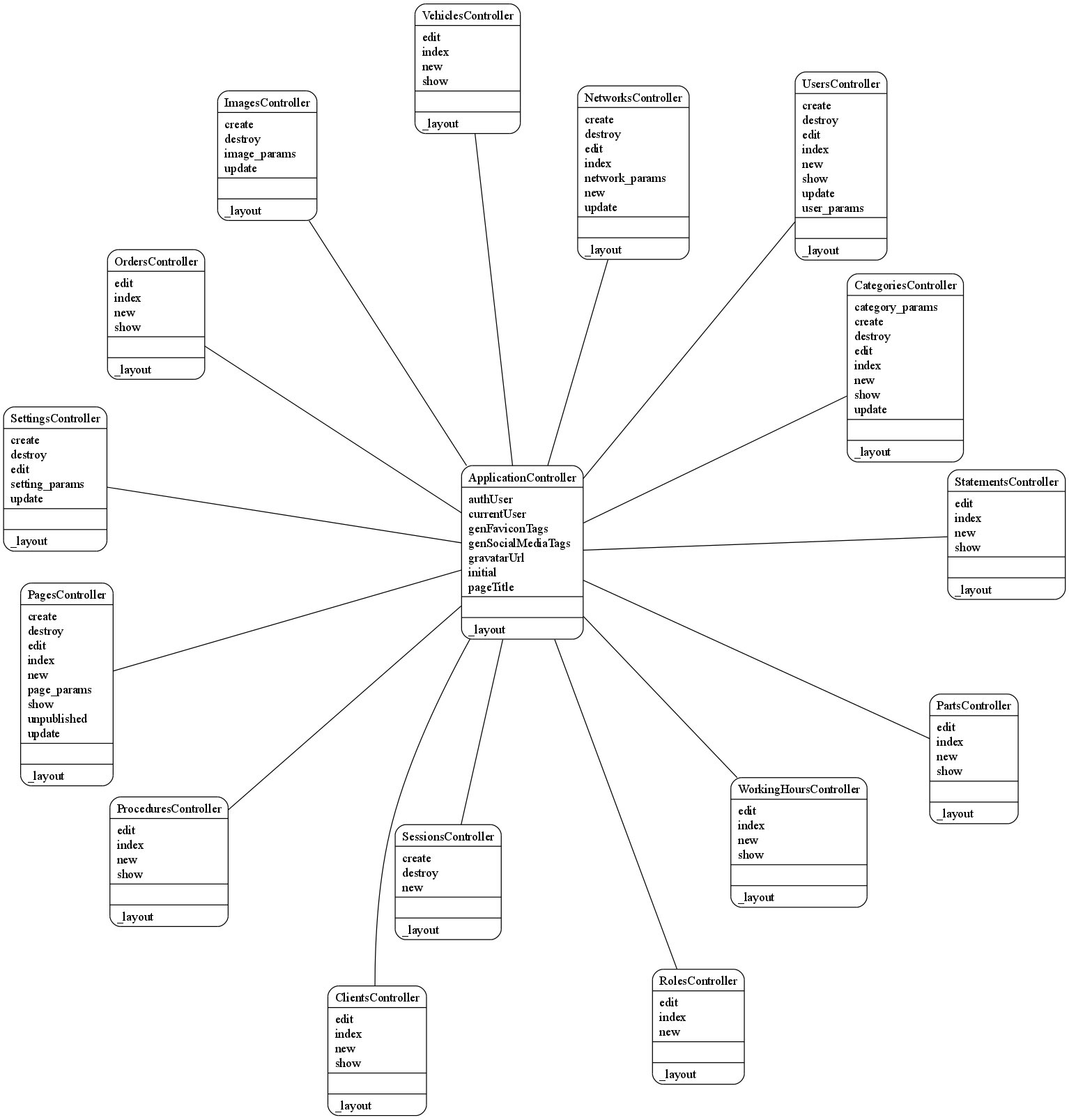
Sadrži metode za stvaranje, brisanje, izmjenu i pregled narudžbi.

* **Images controller**

Sadrži metode za stvaranje, brisanje te izmjenu slika.

* **Vehicles controller**

Sadrži metode za stvaranje, brisanje, izmjenu i pregled vozila.

Slika 6.2. Kontroleri

**Modeli**

Model je sloj u kojem se nalazi cjelokupna podatkovna logika. S obzirom da je model podataka detaljno opisan u opisu baze podataka, ovdje neće biti detaljno razmatran.

Važno je napomenuti da u RoR aplikacijama svaki model sadrži polje *id* tipa integer, polje *created\_at* tipa datetime, te polje *updated\_at* tipa datetime. Ako nije suprotno navedeno, pretpostavlja se da svaki model sadrži navedena polja.

* **Category**

Category je model koji pohranjuje kategorije stranica. Sadrži polja *title*, *slug*, *icon*, *ancestry*. Polje *title* je polje tipa string i u njega se pohranjuje naslov kategorije, *slug* je polje tipa string i u njega se pohranjuje željeni URL stranice, *icon* je polje tipa string i u njega se pohranjuje HTML kod za prikaz ikone koja će se postaviti prije naslova kategorije, ako je to moguće. Izuzetno, *ancestry* je polje tipa string i služi za organizaciju podataka u stablo, te ga zahtijeva *gem* Ancestry.

* **Ckeditor::Asset**

Ovaj model se koristi za pohranjivanje priložaka tekstu koji su dodani kroz CKEditor. Sadrži ime, tip, veličinu (u oktetima), širinu i visinu (ako je slika) te uz koji unos je asociran priložak. Ovo je apstraktni model koji je automatski generiran te za detaljni uvid u njegovu funkcionalnost se treba osvrnuti na dokumentaciju CKEditor *gema*.

* **Ckeditor::Attachmentfile**

Koristi se kao i Ckeditor::Asset, ali isključivo za datoteke koje nisu slike.

* **Ckeditor::Picture**

Koristi se kao i Ckeditor::Asset, ali isključivo za datoteke koje su slike.

* **Client**

Client je model koji sadrži podatke o klientima s kojima je radionica poslovala. Sadrži ime i prezime klijenta u poljima *nameFirst* i *nameLast* spremljene kao string. Kontakt adresu i kontakt broj telefona se pohranjuje u polja *contactAdress* i *contactTelephone,* također kao string.

* **Image**

Image je generični model za pohranjivanje slika, svrha mu je spremanje slika korisničkih profila i dijelova sa lagera. Ima mogućnost pohranjivanja imena slike u polje *imageName* kao string te također može pohraniti tip slike u polje *imageType* kao integer. *Path* je najbitnije polje u ovom model jer pamti gdje je na serveru slika pohranjena. *isHidden* je polje koje omogućuje sakrivanje slika, ako to korisnik želi.

* **Network**

Network je model za pohranjivanje socijalnih mreža koji pamti samo najosnovnije informacije poput imena, ikone i poveznice na društvenu mrežu. Ime se sprema u polje *name* kao string, ikona se sprema u polje *icon* kao string koji sadrži HTML kod za generiranje ikone. Poveznica na samu društvenu mrežu se sprema u polje *link* kao string. Polje *hidden* je tipa boolean i omogućuje skrivanje društvene mreže od posjetitelja, u slučaju da ju ne želimo prikazivati na stranici, ali ju ne želimo niti obrisati. *showName* je polje također tipa boolean i omogućuje prikaz imena društvene mreže uz njezinu ikonu.

* **Order**

Model Order u sebi pohranjuje informacije o pojedinačnim radnim nalozima.

*client*\_*id* pohranjuje identifikacijski broj (integer) klijenta koji je tražio zahvat.  
*vehicle*\_*id* pohranjuje identifikacijski broj (integer) vozila koje je klijent prijavio za zahvat. *user\_id* pohranjuje identifikacijski broj (integer) referenta zaduženog za radni nalog. Polje *description* pohranjuje opis željenog zahvata kao formatirani string (niz znakova). Polje *status* bilježi u kojem je stanju radni nalog (otvoren, storniran…) kao integer. *timeStart* i *timeFinish* u sebi pohranjuju vrijeme i datum početka te završetka radnog naloga.

* **OrderPart**

OrderParts je pomoćni model koji služi kao posebna vrsta *join table* u *to-many* odnosu. U RoR aplikacijama je moguće napraviti *join table* bez modela, ali ako se žele pohranjivati dodatni podaci, u *join table* se mora stvoriti model, ali ne i kontroler. Pošto moramo pamtiti koliko je dijelova bilo potrošeno u nekom radnom nalogu, treba nam ovaj pomoćni model kako bismo to mogli pohranjivati u sustav. *order\_id* sadrži identifikacijski broj radnog naloga, a *part\_id* identifikacijski broj dijela. Polje *quantity* sadrži količinu potrošenih dijelova. Tip ovog polja je float radi mogućnosti lakšeg proširenja, odnosno ako se korisnik odluči unijeti količinu tekućine, sustav mu to omogućava bez velikih preinaka.

* **Page**

Page model pohranjuje pojedinačne web stranice. Naslov stranice pohranjuje se u polje *title* kao string, a URL stranice u polje *slug* kao string. Ikonu pohranjuje u polje *icon* kao HTML string, sadržaj u polje *content* kao HTML formatirani string. *contentPreview* je polje za potencijalna proširenja sustava (predviđeno je za dodavanje kratkog opisa stranice kako se korisniku ne bi trebala učitavati cijela stranice već samo njezin opis ili primjer). *isPublic* je boolean koji određuje je li dopušteno neregistriranom korisniku pristupiti toj stranici, dok *isHidden* određuje hoće li se stranica prikazivati u popisu stranica. *hasPreview* je boolean koji određuje hoće li se u popisu stranica koristiti kratki opis ili puna stranica, a *category\_id* pohranjuje kategoriju kojoj stranica pripada. *Ancestry* je polje koje zahtijeva *gem* Ancestry i njegova je svrha organizacija stranica u stabla, iako postoji mogućnost da će ova funkcionalnost biti izbačena u novijim verzijama.

* **Part**

Part je model koji pohranjuje pojedinačne dijelove. Polje *name* sadrži ime dijela kao string, *Quantity* sadrži količinu dijelova istog tipa kao cijeli broj, međutim u novijim verzijama će to biti promijenjeno u tip float kako bi se mogle u sustavu prikazivati tekućine i slični materijali. Polje *price* sadrži jediničnu cijenu pojedinog dijela.

* **PartProcedure**

PartProcedure je pomoćni model koji služi kao *join table*. Njegova svrha je pohranjivanje količine korištenih dijelova nekog tipa u određenom postupku. *part\_id* i *procedure\_id* pohranjuju identifikatore dijela odnosno postupka, a *quantity* pohranjuje količinu utrošenog dijela u postupku, kao float.

* **Procedure**

Procedure je model koji pohranjuje postupke. Ime postupka pohranjuje se u polje *name*, a opis postupka se pohranjuje u polje *description* kao string, odnosno HTML string. *minNumOfWorkers* je polje koje pohranjuje minimalan broj zaposlenika potrebnih da se postupak obavi. Polje *duration* pohranjuje trajanje samog postupka u satima, a *manHour* je polje koje pohranjuje cijenu čovjek-sata za ovaj postupak.

* **Role**

Role je model koji pohranjuje uloge korisnika u sustavu te je jedan od najbitnijih modela jer određuje prava pristupa određenog korisnika u sustavu. Sadrži polje *name* u koje se pohranjuje ime uloge kao string i dva booleana za svaki kontroler u aplikaciji koji određuju može li korisnik dodavati, izmijenjivati i brisati sadržaj u tom kontroleru, odnosno smije li pregledavati sadržaj kojim raspolaže taj kontroler.

* **Setting**

Setting je model koji pohranjuje postavke stranice, sadrži ime stranice u polju *pageTitle* kao string, opis stranice u polju *pageDescription* kao formatirani string te identifikatore slika koje će se koristiti za favikonu stranice, odnosno za sliku na društvenim mrežama, pohranjene u poljima *pageFavicon* i *pagePreview*.

* **Statement**

Statement je model koji sadrži dnevne, mjesečne i godišnje izračune. U polje *type* spremamo vrstu izračuna. Vrijednosti su cijeli brojevi, pri čemu je 0 za dnevni, 1 za tjedni, 2 za mjesečni i 3 za godišnji izračun, što je bitno zbog optimizacije pri izračunu i optimizacije pretraživanja izračuna. *statementDate* pohranjuje datum provedenog izračuna, a polje *money* sadrži izračunatu bilansu.

* **Vehicle**

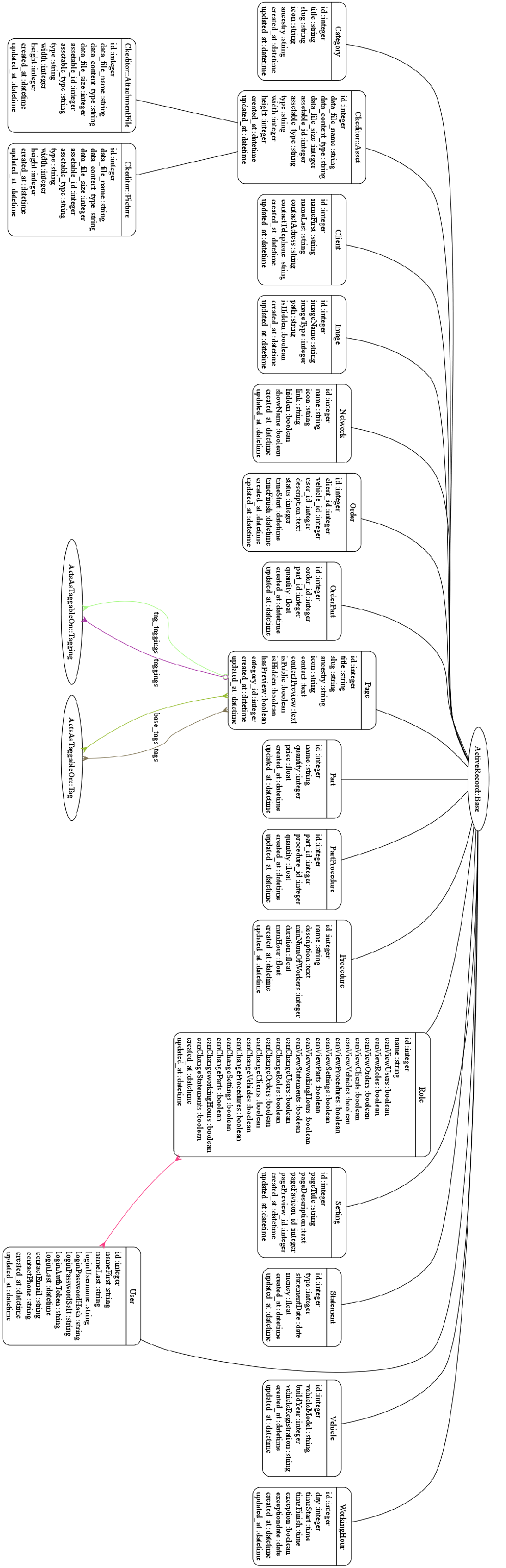
Model Vehicle sadrži vozila klijenata, pri čemu polje *vehicleModel* pohranjuje model automobila kao string, a godinu izgradnje vozila u polju *buildYear* kao integer. Registracija vozila je zapisana u polju *vehicleRegistration* kao string.

* **WorkingHour**

WorkingHours je model koji sadrži radno vrijeme radionice. Model je izrazito bitan za provedbu izračuna, jer se oni rade na kraju radnog vremena koje je određeno ovim modelom. Polje *day* sadrži dan u tjednu na koji se radno vrijeme odnosi, pri čemu je vrijednost 0 za ponedjeljak, 1 za utorak itd. *timeStart* i *timeFinish* su polja u koja se pohranjuje vrijeme otvaranja i zatvaranja radionice. Polje *exception* određuje je li unos izvanredan te se neće rotirati kao ostala radna vremena u sustavu već će se pojaviti samo na datum iznimke i nikada više (ovo je potrebno za praznike i posebne okolnosti). Polje *exceptiondate* pohranjuje datum iznimke.

* **User**

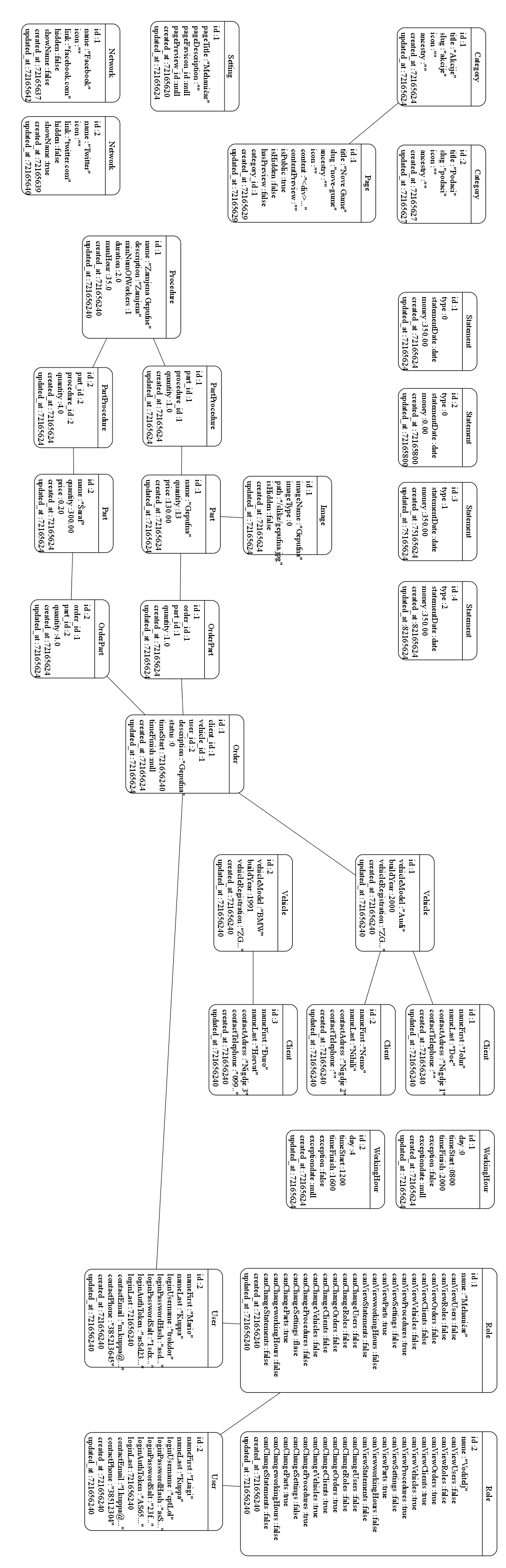
User je definitivno najvažniji model u sustavu. Sadrži sve podatke potrebne za funkcioniranje korisnika u sustavu i omogućuje postojanje korisnika. Ime i prezime korisnika su pohranjeni u poljima *nameFirst* i *nameLast* kao string dok se korisničko ime korisnika pohranjuje u polje *loginUsername* kao string. Korisnikova lozinka se pomoću *bycript* gema razlaže u njezine *hash* i *salt* komponente, koje se tada pohranjuju u *loginHash* i *loginSalt* polja ovog modela kao stringovi. *loginAuthToken* je polje koje sadrži nasumično generirani jedinstveni autentifikacijski *token*. Token ima dvije svrhe - za jednoznačnu identifikaciju korisnika iz *session* kolačića sa korisnikovog pretraživača te za mogućnost potencijalnog proširenja sustava i stvaranje eksternog API-ja za komunikaciju s aplikacijom. Polje *loginLast* pohranjuje vrijeme korisnikove zadnje prijave u sustav, a polja *contactEmail* i *contactPhone* sadrže email adresu i telefonski broj korisnika.

**Slika 6.3.** Modeli

## Dijagram objekata

Dijagram objekata prikazuje stanje sustava u određenom trenutku. Pokretanjem zahtjeva nad aplikacijom nastaje *ActiveRecord*, instanca razreda *Base* koja sadrži sve relacije baze podataka važne za nadzor i provedbu zahtjeva. Objekt klase *User* pritom sadrži podatke o korisniku koji pristupa sustavu, a objekt klase *Role* podatke o dopuštenjima tog korisnika. Ovisno o postavljenim vrijednostima, određenom korisniku je omogućen pregled ili izmjena podataka o korisnicima, klijentima, narudžbama i vozilima, kao i ostalih postavki poput promjene dopuštenja korisnika.

Objekt razreda *Client* sadrži podatke o klijentu, a instanciranjem klase *Order* stvaraju se podaci o narudžbi. Bilježi se opis, status, naručitelj te vrijeme trajanja narudžbe, dok objekt klase *OrderPart* sadrži zapise o posebno označenim dijelovima određene narudžbe. Objekt klase *Part* opisuje dijelove koji se prodaju, a *Procedure* podatke o trajanju provedbe narudžbe i minimalnom broju radnika potrebnom za provedbu. Objekt klase *Vehicle* sadrži podatke o vozilima, a *WorkingHour* podatke o ukupnom vremenu rada u jednom danu.



Slika 6.3. Dijagram objekata

## Ostali UML dijagrami

*Ovdje počinju sadržaji Revizije 2.*

*U ovom potpoglavlju potrebno je uvrstiti dodatna četiri dijagrama koji prikazuju arhitekturu sustava i to: komunikacijski dijagram, dijagram stanja, dijagam aktivnosti i dijagram komponenti. Potrebno je dati barem po jedan primjerak svakog od dijagrama. Potrebno je da svaki od dijagrama prikazuje (jedan) bitan dio funkcionalnosti sustava. Dijagram komponenti treba prikazivati sve komponente sustava. Prema potrebi, ovdje se mogu dodati i neki dodatni sekvencijski dijagrami za objekte.*

# Implementacija i korisničko sučelje

*U ovom poglavlju potrebno je:*

*dati dijagram razmještaja (deployment dijagram)*

*navesti koje su tehnologije i alati korišteni u razvoju sustava*

*dati isječak programskog koda koji implementira neku od temeljnih funkcionalnosti u sustavu*

*objasniti kako je ispitano implementirano rješenje i pokazati bar 4 ispitna scenarija*

*dati upute za instalaciju*

*dati upute za korištenje*

## Dijagram razmještaja

*Potrebno je umetnuti dijagram razmještaja i po potrebi ga opisati.*

## Korištene tehnologije i alati

*Navesti sve tehnologije i alate koji su primijenjeni pri izradi projekta te ih ukratko opisati; njihovo značenje i mjesto i način primjene.*

## Isječak programskog koda vezan za temeljnu funkcionalnost sustava

*U ovom poglavlju potrebno je prikazati isječak programa koji prema mišljenju studenta ostvaruje temeljnu funkcionalnost u sustavu (ili nekom modulu).*

## Ispitivanje programskog rješenja

*U ovom poglavlju je potrebno opisati provedbu ispitivanja implementiranih funkcionalnosti s prikazom odabira ispitnih slučajeva.*

*Prilikom prezentacije svojih Ispitnih scenarija (minimalno 4) studenti trebaju ispitati temeljnu funkcionalnost i rubne uvjete. Poželjno je da se napravi i ispitni slučaj koji koristi funkcionalnosti koje nisu implementirane te da se vidi na koji način sustav reagira kada nešto nije u potpunosti ostvareno.*

*Različiti ulazi za ispitne scenarije trebaju pokrivati temeljnu funkcionalnost nekog modula i nekoliko rubnih uvjeta.*

## Upute za instalaciju

*U ovom poglavlju potrebno je dati upute za instalaciju ostvarenog prototipa.*

## Korisničke upute

*Korisničke upute ovisit će o količini implementirane funkcionalnosti. Očekuje se da upute budu na oko 5 A4 stranica koje će dati potpuni opis funkcionalnosti sustava sa stajališta krajnjeg korisnika..*

# Zaključak

*U ovom poglavlju potrebno je navesti koji je status implementacije u odnosu na postavljene ciljeve, koji su tehnički izazovi prepoznati, jesu li riješeni ili kako bi mogli biti riješeni, koja su znanja stečena pri izradi projekta,koja bi znanja bila posebno potrebna za brže i kvalitetnije ostvarenje projekta i koje bi bile smjernice za daljnji rad kad bi se nastavilo s projektom.*

# Reference

*Popisati sve reference i literaturu koja je pomogla pri ostvarivanju projekta.*

1. Oblikovanje programske potpore, FER ZEMRIS, <http://www.fer.hr/predmet/opp>
2. Oblikovanje programske potpore, FER ZEMRIS, <http://www.zemris.fer.hr/predmeti/opp>
3. I. Sommerville, „Software engineering“, 8th ed, Addison Wesley, 2007.
4. T.C.Lethbridge, R.Langaniere, „Object-Oriented Software Engineering“, 2nd ed. McGraw-Hill, 2005.
5. Tigris.org Open Source Software Engineering Tools, <http://readyset.tigris.org/>
6. Software engineering ,Rutgers University, [http://www.ece.rutgers.edu/~marsic/Teaching/SE](http://www.ece.rutgers.edu/~marsic/teaching/se)
7. I. Marsic, „Software engineering book“, Department of Electrical and Computer Engineering, Rutgers University, [http://www.ece.rutgers.edu/~marsic/books/SE](http://www.ece.rutgers.edu/~marsic/books/se)
8. Concepts: Requirements, <http://www.upedu.org/upedu/process/gcncpt/co_req.htm>
9. UML 2 Class Diagram Guidelines, [http://www.agilemodeling.com/style/classDiagram.htm](http://www.agilemodeling.com/style/classdiagram.htm)
10. Domain Class Diagram Modeling Standards and Guidelines, <http://www.bced.gov.bc.ca/imb/downloads/classdiagramstandards.pdf>
11. ArgoUML, <http://argouml.tigris.org/>

# Dodatak A: Dnevnik sastajanja

1. sastanak: Petak, 11. 10. 2013., 17:00 - 18:30 h

Prisutni: Stanko Krtalić Rusendić, Zrinka Lekić, Marin Pogančić, Vlaho Poluta, Martin Sertić, Mirjam Škarica, Mateja Škriljak

Sadržaj sastanka: Zajedno smo proučili tekst zadatka projekta i pokrenuli raspravu o načinu implementacije. Raspravljali smo o mogućim problemima i nejasnoćama. Sastavili smo listu pitanja za asistenta i dogovorili se za konzultacije.

Rezultat: Odlučili smo implementirati rješenje koristeći Ruby on Rails framework i PostgreSQL za upravljanje bazom podataka. Dogovorili smo prve konzultacije i planirali daljnje aktivnosti s posebnim naglaskom na raspodjelu poslova.

2. sastanak: Četvrtak, 17.10, 18:00 - 19:30

Prisutni: Stanko Krtalić Rusendić, Zrinka Lekić, Marin Pogančić, Vlaho Poluta, Martin Sertić, Mirjam Škarica, Mateja Škriljak

Sadržaj sastanka: Sastanak je bio neposredno nakon susreta s asistentom na konzultacijama te smo raspravili njegove upute. Dogovorili smo se oko podjele poslova.

Rezultat: Napravili smo ER model baze podataka, obrazac upotrebe i sekvencijski dijagram. Napravili smo podjelu poslova vezanih za pisanje dokumentacije:

Vlaho Poluta - dnevnik promjena

Mirjam Škarica - opis projektnog zadatka

Stanko Krtalić Rusendić - rječnik pojmova, dijagram razreda i dijagram objekata

Zrinka Lekić, Vlaho Poluta - funkcijski zahtjevi

Martin Sertić - ostali zahtjevi, opis baza podataka

Marin Pogančić - svrha, opći prioriteti i skica sustava, opis dijagrama razreda

Mateja Škriljak - o grafičkom sučelju, opis dijagrama objekata, dnevnik sastajanja

3. sastanak: Petak, 25. 10. 2013., 17:00 - 18:00

Prisutni: Stanko Krtalić Rusendić, Zrinka Lekić, Marin Pogančić, Vlaho Poluta, Martin Sertić, Mirjam Škarica, Mateja Škriljak

Sadržaj sastanka: Ispravak ER modela baze podataka i dijagrama, otklanjanje nejasnoća vezanih uz dijagrame.

Rezultat: Izrada dijagrama obrazaca uporabe i sekvencijskih dijagrama