Dizajn sistema

eVehicleInspection

Mihailo Vasić 1179/19
Filip Delević 1150/19
Aleksandar Drljača 1119/19
Nemanja Vujinović 1121/19
Stefan Milaković 1144/20

Elektrotehnički fakultet
Banja Luka, april 2024

Sadržaj

1. Uvod

1.1. Namjena sistema

Softverski sistem eVehicleInspection ima višestruku namjenu. Sa jedne strane, omogućava građanima da brzo i jednostavnu zakažu tehnički pregled i/ili registraciju svojih vozila. To je moguće uraditi sa bilo koje lokacije putem eVehicleInspection sistema uz samo par klikova, čime se izbjegava potreba za odlaskom u servis ili zakazivanje termina putem telefona. Pored jednostavnog zakazivanja, korisnici mogu da na jednom mjestu dobiju čitavu ponudu svih kompanija koji pružaju usluge tehničkog pregleda i registracije vozila. Korisnici putem svojih naloga, u najkraćem roku, bivaju obaviješteni po pitanju ishoda pregleda vozila, kao i eventualnog pomijeranja zakazanog termina. Sistem omogućava da se za proizvoljan tip i karakteristike vozila dobije precizna procjena troškova pregleda i registracije vozila. Registrovani korisnici imaju mogućnost unosa podataka o svojim vozilima, koji će biti korišteni prilikom svakog narednog zakazivanja. Sistem čuva istoriju rezervacija i obavljenih pregleda za sva vozila registrovanih korisnika, čime se omogućava jednostavan uvid u podatke o vozilu. Omogućeno je i preuzimanje zapisnika, u pdf formatu, kreiranog za neki od prethodnih tehničkih pregleda/registracija odabranog vozila.

Sa druge strane, sistem omogućava svim kompanijama koje se bave pružanjem usluga tehničkog pregleda i registracije vozila da oglase svoje usluge. Kompanije mogu da na jednostavan način zaprimaju i upravljaju rezervacijama, da pošalju povratne informacije potencijalnim mušterijama, da evidentiraju obavljene preglede, te da upravljaju ljudskim resursima u kompaniji. Sistem omogućava jednostavno generisanje izvještaja sa obavljenih pregleda na osnovu podataka o vozilu koje su klijenti prethodno unijeli, čime se smanjuje vrijeme čekanja. Kompanije putem ovog sistema imaju uvid u kalendar rezervacija, dok se slobodni termini automatski generišu.

Pored navedenog postoji i administratorki dio sistema koji omogućava administratorima kompanija da dodaju nove radnike, nove uslužne stanice, da proširuju kapacitete postojećih stanica, kao i da manipulišu podacima vezanim za radno vrijeme, neradne dane i godišnje odmore radnika.

1.2. Projektni ciljevi

Projektni ciljevi kojima će biti vođeno projektovanja eVehicleInspection sistema, kao i odabir konkretne arhitekture sistema su: jednostavna upotreba, prenosivost, jednostavna nadogradnja funkcionalnosti, brz razvoj i niski troškovi razvoja sistema.

Jednostavna upotreba

Potreba za ostvarivanjem projektnog cilja jednostavnost upotrebe je posljedica namjene sistema. Sistem je namijenjen širokom spektru korisnika, za koje se ne pretpostavlja da imaju iskustva u korištenju sličnih aplikacija. Kako je cilj upotreba sistema od strane što većeg broja korisnika, neophodno je da sistem bude jednostavan i intuitivan za korištenje. Prilikom razvoja sistema potrebno je postići jednostavan i intuitivan korisnički interfejs.

Nadogradnja sistema i lako održavanje sistema

Prilikom projektovanja sistema potrebno da je se odabere arhitektura sistema koja će omogućiti da se, nakon što sistem zaživi među korisnicima, dodaju nove i proširuju postojeće funkcionalnosti. Ideja je, da se postigne modularnost sistema, kako bi se omogućilo jednostavno održavanje, izmjena i dodavanje funkcionalnosti bez ostvarivanja prevelikog uticaja na ostatak sistema. Sve navedeno zahtijeva da komponente sistema budu slabo spregnute i sa dobro definisanim interfejsima.

Prenosivost

Za očekivati je da će različiti korisnici pristupati sistemu putem različitih uređaja. Stoga je potrebno odabrati odgovarajuću arhitekturu sistema i tehnologije za njegovu implementaciju koje će omogućiti pristup sistemu bez obzira na uređaj i operativni sistema sa kojeg se pokušava pristupiti. Na taj način, svim zainteresovanim korisnicima će biti omogućen pristup čime se proširuje i potencijalno tržište za plasiranje sistema. Prenosivost će biti obezbijeđena implementacijom sistema u vidu web aplikacija umjesto native aplikacije koja je vezane za određenu platformu.

Brz razvoj sistema

Ukoliko se izuzmu web stranice pojedinih kompanija koje pružaju funkcionalnosti rezervacije termina pregleda vozila i online kalkulatora troškova pregleda, koje su isključivo vezane za datu kompaniju, eVehicleInspection sistem nema alternativu. Iz tog razloga je jako bitno što prije razviti sistem i izaći na tržište dok još ne postoje konkurentni sistemi.

Niski troškovi razvoja sistema

Uzimajuću u obzir da se sistem ne razvija po narudžbi za naručioca, nego se pokušava samostalno izaći na tržište, jako je bitno ograničiti ulaganja u sam razvoj sistema i tako smanjiti potencijalne rizike.

1.3. Definicije, skraćenice

Arhitektura sistema – konceptualni model sistema koji definiše njegovu strukturu i ponašanje

Dekompozicija – razlaganje složenijeg sistema na kolekciju jednostavnijih podsistema

Podsistem - zamjenljivi dio sistema koji može da pruža servise (funkcionalnosti) drugim podsistemima putem dobro definisanih interfejsa, ali i da koristi usluge drugih podsistema

Interfejs – eksterno vidljivi skup funkcionalnosti nekog (pod)sistema

Server – softver koji pruža servise drugim programima koji se nazivaju klijenti

Klijent – program koji zahtijeva neku funcionalnost/servis od drugog programa (servera)

Komunikacioni protokol – skup formalnih pravila kojima se definiše način prenosa i razmjene podataka putem računarske mreže

Native aplikacija – aplikacija koja je namjenski razvijena za ciljnu platformu i koja se direktno izvršava na toj ciljnoj platformi

Mobilna aplikacija – aplikacija koja je prilagođena za izvršavanje i prikazivanje na prenosnim uređajima poput telefona, tableta, pametnih satova i dr.

SPA (Single Page Application) – tip web aplikacije koja sastoji od samo jedne dinamičke HTML stranice koja se na početku dobavi sa web servera i čiji se sadržaj mijenja u skladu sa korisničkom interakcijom i podacima pribavljenim od aplikativnog servera (*backend*)

UML – Unified Modeling Language

UI – User Interface

DBMS – Database Management System

1.4. Referentni dokumenti

Kao osnova za projektovanje eVehicleInspection sistema koristila se specifikacija sistemskih (funkcionalnih i nefunkcionalnih) zahtjeva sadržana u dokumentu pod nazivom "Specifikacija sistemskih zahtjeva eVehicleInspection softverskog sistema".

1.5. Kratak pregled dokumenta

Za eVehicleInspection sistem se može reći da je "greenfield" projekat, pa samim tim nema analize postojećeg sistema. Iako web stranice pojedinih kompanija imaju sličnu namjenu, one pružaju samo podskup funkcionalnosti eVehicleInspection sistema, i obično omogućavaju samo procjenu troškova tehničkog pregleda, tj. registracije vozila.

U nastavku dokumenta će prvo biti predstavljene kandidatske arhitektura sistema. Za svaku od njih će biti navedene osnovne karakteristike, prednosti i mane, kao i obrazloženje zašto (ni)je odabrana.

U trećem poglavlju će detaljno biti predstavljena odabrana arhitektura za koju se smatra da ispunjava potrebe eVehicleInspection sistema i omogućava ostvarivanje postavljenih projektnih ciljeva. Zatim će biti predstavljena dekompozicija sistema na podsisteme i komponente, raspored softverskih komponenata sistema na hardverske čvorove, način perzistencije podataka, mehanizmi kojima se obezbjeđuje kontrola pristupa resursima sistema i sigurnost podataka. Na kraju će biti predstavljena granična stanja sistema i ponašanje sistema u tim situacijama. Za potrebe reprezentacije arhitekture sistema biće korišteni različiti tipovi UML dijagrama.

2. Potencijalne arhitekture sistema

Kako je u pitanju „green field" projekat, u nastavku poglavlja biće predstavljene potencijalne arhitekture eVehicleInspection sistema.

2.1 Android aplikacija u skladu sa Model–View–ViewModel arhitekturnim stilom

EVehicleInspection sistem bi se mogao implementirati u vidu mobilne Android aplikacije koja za potrebe perzistencije, obrade i razmjene podataka koristi namjenske web servise.

U slučaju odabira ovog pristupa, eVehicleInspection sistem bi imao slojevitu arhitekturu u skladu sa MVVM modelom.

Prvi i korisniku najbliži sloj arhitekture aplikacije je UI (prezentacioni) sloj. UI sloj ima ulogu View komponente u MVVM modelu. Zadatak ovog sloja je prikaz aplikativnih podataka u odgovarajućem formatu i omogućavanje korisničke interakcije. UI sloj u svakom trenutku treba da prikazuje aktuelno stanje aplikacije, dobijeno od Data sloja, i ažurira prikaz u skladu sa korisničkim interakcijama i izmjenom stanja aplikacije. Sastoji se od podslojeva: UI elementi i State Holder-i (uloga ViewModel-a).

„Najniže" pozicioniran je Data sloj koji ima ulogu Model-a i sadrži poslovnu logiku i podatke aplikacije, te vrši lokalnu pohranu podataka. Data sloj se sastoji od Repository-ja (definišu interfejs putem kojeg viši slojevi mogu koristiti servise poslovne logike) i Data Source-a.

Posrednik u komunikaciji između UI i Data slojeva je opcioni domenski sloj. Domenski sloj ima zadatak da enkapsulira kompleksnu aplikativnu logiku Data sloja i na taj način omogući njenu ponovnu upotrebu od strane više ViewModel-a.

Kako je aplikativna logika eVehicleInspection sistema djelomično kompleksna, i kako je cilj razmjena i dostupnost podataka, za potrebe obrade i perzistencije podataka koristio bi se i dodatni sloj poslovne logike i perzistencije, realizovan u vidu namjenske backend web aplikacije. Data sloj bi putem Web tehnologija koristio servise pomenute aplikacije.

Navedena arhitektura nije odabrana za eVehicleInspection sistem iz razloga što bi razvoj takve native aplikacije trajao dugo, zahtijevao veća ulaganja i ograničio upotrebu aplikacije na Android uređaje, čime se ne bi ispunili projektni ciljevi prenosivost, brz razvoj sistema i niska ulaganja.

2.2 Web aplikacija sa klasičnom troslojnom arhitekturom

Arhitektura klasičnih web aplikacija se uobičajeno sastoji od tri sloja.

Prvi sloj jeste klijentski sloj aplikacije, putem kojeg se korisniku prikazuju aplikativni podaci i omogućava njegova interakcija sa aplikacijom. Ovaj sloj se sastoji od statičkih ili dinamičkih (interaktivnih) HTML stranica prikazinih u klijentskom web čitaču. HTML stranice se putem zahtjeva poslatih od strane web čitača dobavljaju sa web servera aplikacije i potom iscrtavaju. Osnovna uloga ovih stranice jeste da korisniku prikažu relevantne podatke u odgovarajućem formatu, da obrade njegovu interakciju, te na osnovu nje upute nove zahtjeve ka web serveru. Obradu korisničke interakcije pored ugrađenih ulaznih HTML elemenata (dugme, checkbox, itd,) vrši i Javascript kod sadržan u dinamičkim web stranicama, koji se izvršava u okviru Javascript engine-a web čitača.

Središnji sloj klasičnih web aplikacija jeste web server. Web server zaprima zahtjeve pristigle od klijentskog web čitača i na osnovu tih zahtjeva, kao odgovor vraća odgovarajuću web stranicu za prikaz korisniku. Podsloj zadužen za manipulaciju web stranicama se obično naziva prezentacijski sloj, iz razloga što određuje prikaz podataka na klijentskoj strani. Prije slanja odgovora na klijentski zahtjev, prezentacijski sloj popunjava stranicu odgovarajućim podacima dobijenim u vidu rezultata obrade pristiglog zahtjeva. Za obradu podataka pristiglih u okviru zahtjeva, zadužen je podsloj aplikativne/domenske logike. Ukratko, aplikativna logika na osnovu zahtjeva odlučuje koji podaci će biti prikazani korisniku, a prezentacijski sloja određuje način, tj. format njihovog prikaza.

Sloj podataka, tj. server baze podataka, predstavlja posljednji sloj koji omogućava trajnu perzistenciju aplikativnih podataka kojima manipuliše domenska logika aplikacije.

EVehicleInspection sistem neće biti implementiran u vidu web aplikacije sa troslojnom arhitekturom. Iako ova arhitektura omogućava da se postigne brz razvoj sistema i njegova prenosivost, spajanje prezentacijske i aplikativne komponente u okviru jednog sloja, te realizacija čitave poslovne logike aplikacije kao monolitne komponente nije fleksibilan, niti modularan pristup, pa tako i ne omogućava jednostavno održavanje i nadogradnju sistema.

3. Predložena arhitektura sistema

3.1. Kratak pregled arhitekture i podsistema

EVehicleInspection sistem će biti implementiran u vidu višeslojne, tj. četveroslojne web aplikacije.

Podsistem za prezentaciju, koji ujedno predstavlja klijentski dio sistema će biti realizovan u vidu Single Page React.js Web aplikacije. Ovaj sloj omogućava korisničku interakciju sa sistemom. Upotreba single page pristupa će omogućiti brži odziv sistema i bolje korisničko iskustvo.

Podsistem RESTful Web servisa omogućava pristup domenskoj logici sistema putem HTTP protokola. Podaci će se razmjenjivati u JSON formatu. Realizacija ovog sloja, koji u suštini predstavlja web API poslovne logike sistema, omogućava jednostavnu integraciju sa drugim sistemima, kao i potencijalnu integraciju dodatnih klijentskih aplikacija, npr. native Android aplikacije.

Servisni sloj sadrži aplikativnu logiku eVehicleInspection sistema i predstavlja njegovu centralnu komponentu. Ovaj podsistem će biti realizovan kao grupa slabo spregnutih servisnih komponenti koje će po potrebi ostvarivati direktnu komunikaciju. Ovakav pristup omogućava jednostavan prelazak na servisno orijentisanu arhitekturu. U tom slučaju sva komunikacija između servisnih komponenti bi se preusmjerila na odgovarajuće RESTful web servise.

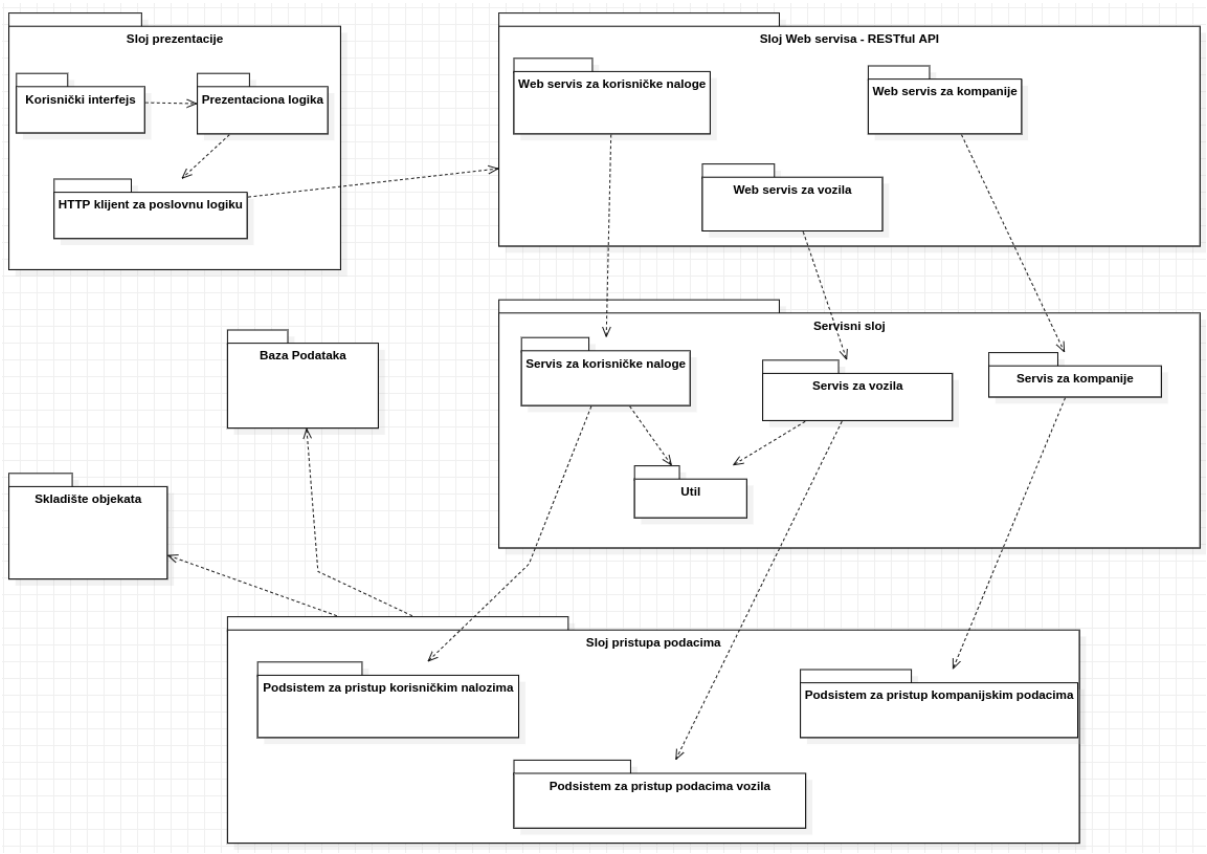
Razdvajanje poslovne logike od prezentacije podataka, i upotreba više slojeva omogućava jednostavnu nadogradnju i održavanje sistema.

Podsistem za pristup perzistentnim podacima enkapsulira logiku za pristup uskladištenim podacima. Ovaj podsistem izlaže API visokog nivoa apsktrakcije, kojeg koristi servisni sloj. Servisni sloj zahvaljujući ovom sloju ne mora da ima znanje o tome gdje i kako su podaci stvarno uskladišteni. Sloj web servisa, servisni sloj i sloj pristupa podacima će se realizovati u vidu Spring Boot aplikacije.

Za potrebe skladištenja podataka biće upotrijebljena MySQL relaciona baza podataka i MinIO S3 skladište objekata.

3.2. Dekompozicija sistema

U skladu sa predloženom arhitekturom eVehicleInspection će se sastojati od četiri sloja, koji su na dijagramu predstavljani u vidu paketa. Komponente u okviru sloja prezentacije omogućavaju prikaz aplikativnih podataka, navigaciju kroz aplikaciju i korisničku interakciju sa aplikacijom. Komponente u okviru aplikativnog sloja sadrže poslovnu logiku sistema. Poslovnoj logici sistema se pristupa putem namjenskog sloja RESTful web servisa. Različiti servisi u okviru servisnog sloja po potrebi pristupaju perzistentnim podacima putem sloja za pristup podacima. Komponenta za pristup podacima apstrahuje stvarni pristup perzistentnim podacima i servisnom sloju izlaže API visokog nivoa. Ova komponenta omogućava da se koriste različiti načini perzistencije podataka bez potrebe da se prilagođava servisni sloj. Baza podataka je jedna “off the shelf” komponenta koja se koristi za čuvanje aplikativnih podataka. Za čuvanje fotografija koristi se takođe “off the shelf” komponenta za skladištenje objekata. Ova komponenta je transparentna za servisni sloj, koji joj pristupa putem API-ja sloja za pristup podacima.



Slika 1. "Dijagram paketa"

3.3. HW/SW mapiranje

Dijagram komponenti

EVehicleInspection sistem se sastoji od 6 osnovnih komponenti, tj. podsistema.

Komponenta za prezentaciju se sastoji od 3 podkomponente i koristi servise usluge sloja RESTful Web servisa. Komponenta za grafički korisnički interfejs tj. prezentaciju podataka koristi servise komponente koja implementira logiku prezentacijskog dijela web aplikacije. Ova logika omogućava obradu korisničke interakcije, procesiranje i prilagođavanje rezultata poslovne logike. Router komponenta omogućava jednostavnu navigaciju korisnika kroz web aplikaciju. Komponenta za prezentaciju putem HTTP klijent podkomponente komunicira sa web servisima.

Sloj web servisa predstavlja sloj RESTful HTTP API-ja putem kojeg se pristupa poslovnoj logica sistema. U okviru ove komponente nalazi se podkomponenta koja implementira funkcionalnosti autorizacije i autentikacije prilikom pristupa API-ju sistema. Za svaku logičku cjelinu poslovne logike sistema, postoji namjenski kontroler, tj. web servis koji obrađuje korisničke zahjteve i prosljeđuje ih ka odgovarajućoj podkomponenti poslovne logike. API koji izlaže ova komponenta je servis koji koristi prezentacijski sloj sistema.

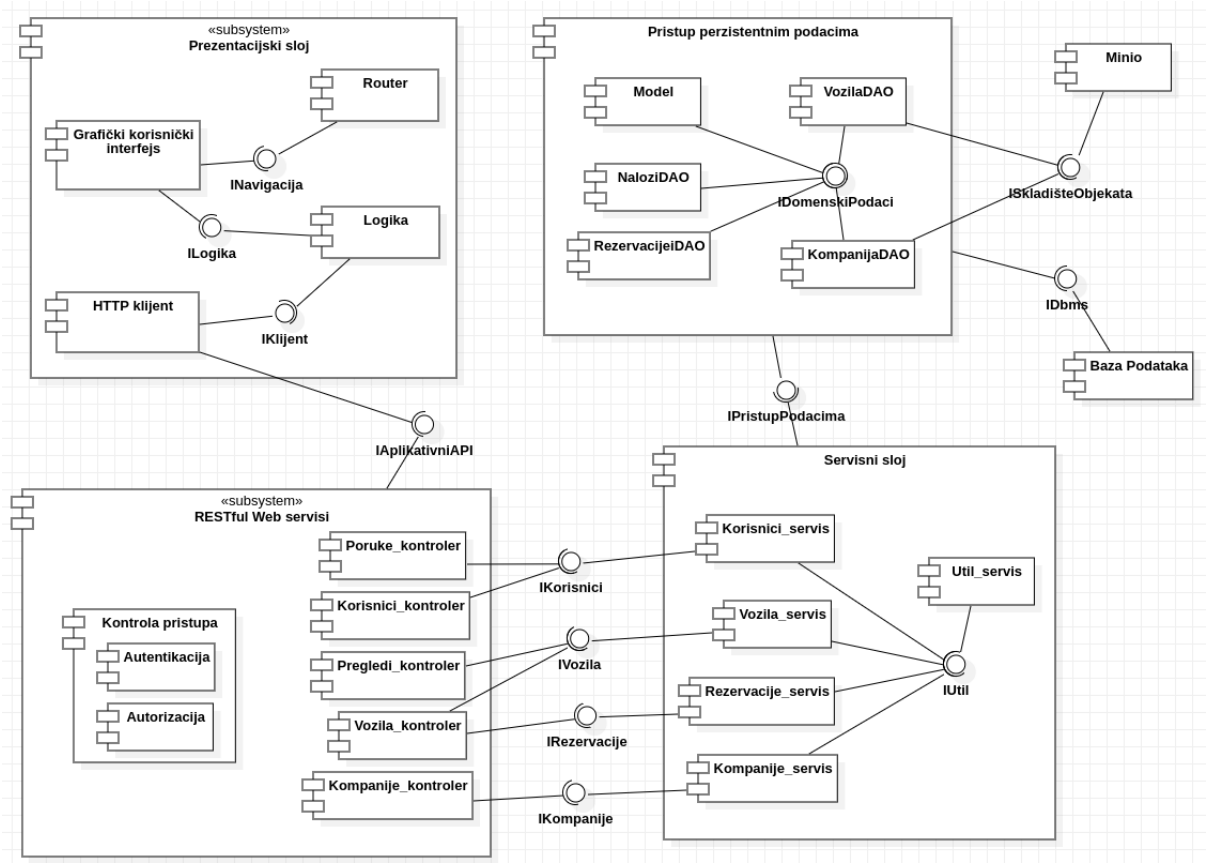
Sloj web servisa koristi servisni sloj za potrebe izvršavanja poslovne logike. Servisni sloj ima servisnu podkomponentu za svaku logičku cjelinu poslovne logike sistema (rad sa vozilima, korisnički nalozi, kompanije koje pružaju usluge, rezervisanje termina i pomoćni servis), koja

izvršava odgovarajuću logiku na osnovu podataka proslijeđenih u korisničkom zahtjevu. Servisne podkomponente su u velikoj mjeri autonomne, te je njihova komunikacija i međuzavisnost svedena na minimum.

Servisni sloj povremeno zahtijeva pristup perzistentnim podacima. Pristup podacima je apstrahovan i odvija se posredstvom komponente za pristup podacima. Komponenta za pristup podacima se sastoji od Data Access Object komponenti i koristi klase domenskog modela.

Stvarni pristup podacima se odvija putem baze podataka čije servisi koriste DAO komponente.

Pored baze podataka, dio podataka se skladišti putem komponente za skladištenje objekata (MinIO S3 Object Store).



Slika 2. "Dijagram komponenti"

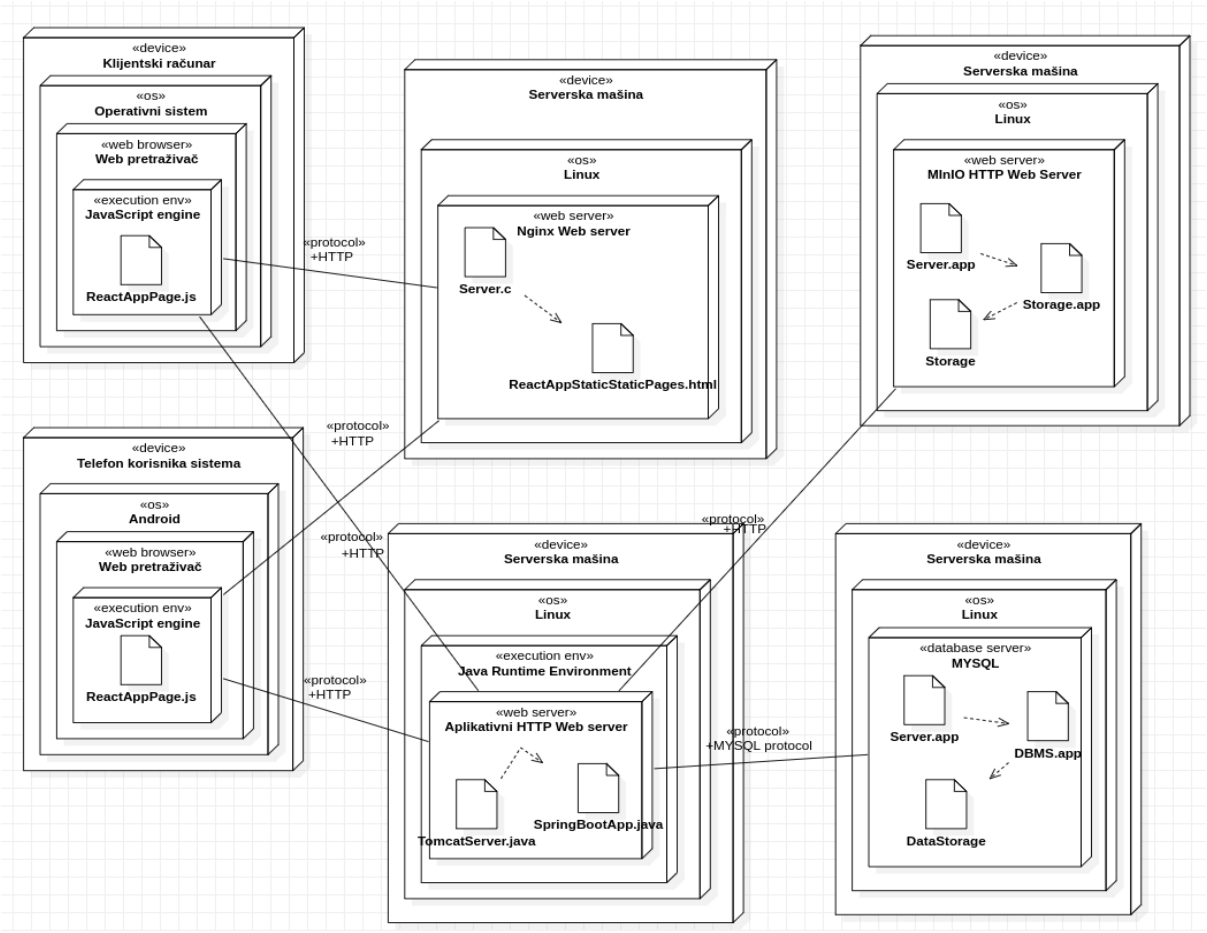
Dijagram razmještaja

Korisnici eVehicleInspection sistemu mogu pristupiti kroz web čitač putem odgovarajućeg URL-a. Pristup je moguć sa bilo kojeg uređaja, sa bilo kojim operativnim sistemom koji ima instaliran web čitač. Svaki web čitač ima ugrađen Javascript engine koji omogućava izvršavanje aplikacije.

Klijentski dio aplikacije, koji se izvršava u okviru web čitača, je potrebno prethodno dobaviti sa web servera putem HTTP protokola. Web server se izvršava na namjenskoj mašini koja ima instaliran Linux operativni sistem i Nginx serversku aplikaciju. Putem web servera klijentima se servira statički sadržaj prezentacijskog dijela web aplikacije u vidu HTML stranica koje sadrže CCS stil i Javascript logiku.

Klijentska aplikacija komunicira sa aplikativnim serverom putem HTTP protokola. Planirana je upotreba Tomcat servera koji se izvršava u okviru Java Runtime Environment-a i Linux operativnog sistema na namjenskoj serverskoj mašini. Tomcat server opslužuje klijente sa servisima koje pruža poslovna logika sistema, implementirana u vidu Spring Boot aplikacije.

Aplikativni server pristupa perzistentnim podacima putem servera baze podaka koji se izvršava na odvojenoj mašini. Na serveru baze podataka se izvršava Database Management System koji omogućava pristup bazi podataka.



Slika 3. "Dijagram razmještaja"

Pored pristupa perzistentnim podacima, aplikativni server pristupa i MinIO skladištu objekata putem HTTP protokola.

3.4. Perzistentni sloj

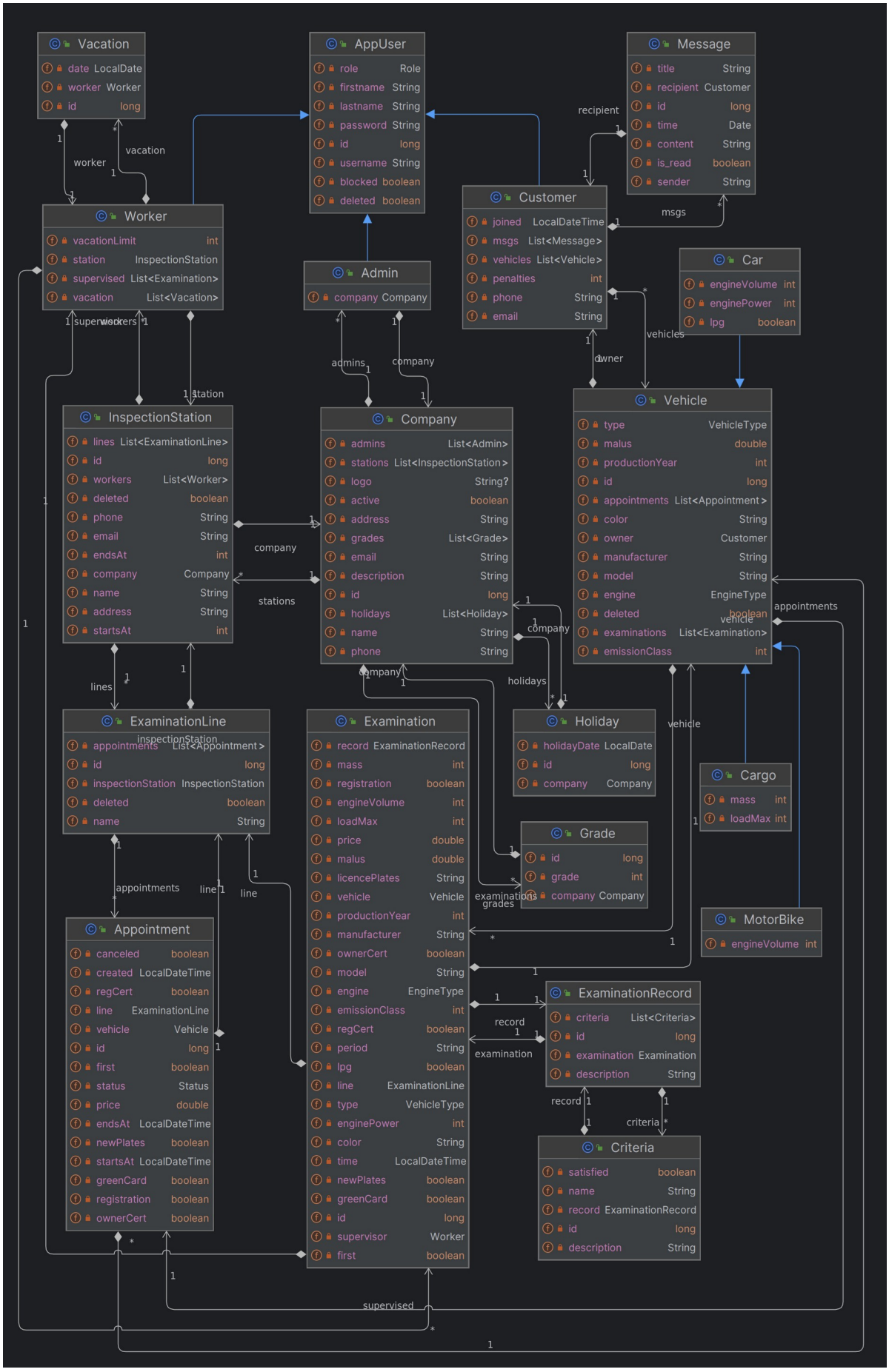
Za potrebe perzistencije podataka biće korištena relaciona baza podataka i MySQL DBMS. Kako su podaci kojima aplikacija manipuliše jako strukturisani, kao prirodan izbor nameće se upotreba relacione baze podataka. Odabrana relaciona baza podataka će omogućiti očuvanje integriteta podataka, kontrolu pristupa, centralizovano skladištenje, te jednostavan pristup i manipulaciju uskladištenima podacima.

Kako bi se ubrzao razvoj aplikacije i olakšao posao projektovanja perzistentnog sloja koristiće se pomoćni alat za generisanje relacija na osnovu izvornog koda aplikacije. Za te potrebe biće korišten objektno-relacioni alat pod nazivom Hibernate, koji za nas vrši mapiranje objekata u odgovarajuće relacije.

Svi podaci kojima aplikacija pristupa će se čuvati centralizovano i za to će biti zadužen jedan zajednički server baze podataka.

U nastavku je dat domenski model eVehicleInspection sistema, tj. dijagram klasa njegovog domenskog sloja.

Pored relacione baze podataka za potrebe perzistencije koriste se još JSON fajlovi i Object Store skladišta. Trenutno aktuelan cjenovnik usluga se čuva u JSON fajlu, a fotografije koje se koriste u web aplikaciji za potrebe reklamiranja kompanija se čuvaju u MinIO Object Store-u.



Slika 4. "Domenski model – dijagram klasa"

3.5. Kontrola prava pristupa i sigurnost

U okviru eVehicleInspection sistema koristiće se tri korisničke uloge putem kojih će se definisati prava pristupa resursima.

Prvu ulogu predstavlja običan korisnik, tj. "klijent". Korisnici sistema koji imaju ovu ulogu, sistem mogu koristiti za zakazivanje tehničkih pregleda i registracija, procjenu troškova, inforimisanje o dostupnim servisnim mjestima, čuvanje podataka o svojim vozilima i pregled istorije. Svako ko je zainteresovan za ovaj vid upotrebe eVehicleInspection sistema, može popunjavanjem jednostavne forma kreirati nalog sa ulogom običnog korisnika.

Radnici u kompanijama, koje se bave pružanjem usluga tehničkog pregleda i registracije vozila, koriste nalog sa drugom korisničkom ulogom. Ova uloga im omogućava da pristupaju rasporedu zakazanih termina, da generišu izvještaje sa pregleda, da manipulišu zakazanim terminima i da šalju povratne informacije klijentima. Ovaj tip korisničkog naloga nije moguće samoinicijativno kreirati, nego se on dodjeljuje radniku od strane sistemskog administratora na nivou kompanije.

Uloga sa najvišim stepenom prava pristupa jeste administrator sistema na nivou kompanije. Svaka kompanija koja se odluči za upotrebu eVehicleInspection sistema dobija administratorski nalog putem kojeg može pristupati i manipulisati svim podacima vezanim za kompaniju i njene radnike.

Sistemu je moguće pristupi i bez upotrebe bilo kakvog naloga i u tom slučaju korisniku je dostupan samo podskup funkcionalnosti običnog korisnika. "Gost" na sistemu može da pristupa samo kalkulatoru za procjenu troškova registracije i podacima vezanim za kompanije koje pružaju date usluge.

Za autentikaciju korisnika sa bilo kojim tipom naloga, koristiće se korisničko ime i lozinka. Za potrebe pristupa web servisima eVehicleInspection sistema potrebno je imati validan pristupni token koji se korisniku izdaje nakon uspješne autentikacije. Pristupni token će sadržavati korisničku ulogu u sistemu na osnovu koje će se vršiti kontrola pristupa web resursima sistema. Kontrola pristupa putem pristupnog tokena, koji ujedno definiše i tip korisničkog naloga, predstavlja statičku kontrolu pristupa, jer su prava pristupa pojedinih grupa unaprijed poznata i definisana.

Statička kontrola pristupa u eVehicleInspection sistemu jeste potreban ali ne i dovoljan sigurnosni mehanizam. Kako bi se omogućila prava pristupa unutar iste korisničke grupe i obezbijedilo da klijent može da zakazuje termina samo za svoja vozila, da radnici mogu pristupiti rasporedu zakazanih termina samo za stanicu u kojoj rade, i kako bi admin mogao manipulisati isključivo podacime svoje kompanije potrebna je i dodatna dinamička kontrola pristupa. Dinamička kontrola pristupa resursima će biti implementirana putem Protection Proxy projektnog obrasca.

Kao još jedan dodatni sigurnosni sloj, pristup serveru baze podataka će se vrši putem naloga sa ograničenim privilegijama.

3.6. Kontrola toka

S obzirom da se interakcija korisnika sa eVehicleInspection sistemom odvija putem grafičkog korisničkog interfejsa klijentske web aplikacije, kotrola toka je upravljana događajima. Na osnovu korisničke interakcije, kreiraju se događaji koje prepoznaje i obrađuje klijentska aplikacija. Na osnovu podataka sadržanih u događajima izvršava se odgovarajuća logika koja ažurira prikaz korisničkog interfejsa. U zavisnosti od složenosti operacija zahtijevanih od strane korisnika, klijentska aplikacija često mora da pošalje zahtjev ka odgovarajućim web servisima na strani aplikativnog servera, što ukazuje na distribuiranu prirodu kontrole toka. Na strani web servisa eVehicleInspection sistema, izvršava se poslovna logika u vidu CRUD operacija. Aplikativni server putem servera baze podataka pristupa perzistentnim podacima. Rezultat obrade zahtjeva se vraća klijentskoj strani. Podaci iz odgovora se obrađuju na klijentskoj strani i prilagođavaju za prikaz, putem odgovarajućih grafičkih

elemenata korisničkog interfejsa. Ažurirani korisnički interfejs sadrži aktuelne podatke i po potrebi ima izmijenjen vizuelni izgled. Komunikacija između klijentske web aplikacije i web servisa je bazirana na odgovor-zahtjev modelu koji je sinhroni distribuirani model interakcije, tj. kontrole toka.

Web servisi na strani aplikativnog servera podržavaju konkurentni rad kako bi opslužilo svi zainteresovani korisnici uz minimalno čekanje.

3.7. Granična stanja sistema

Pokretanje

Prilikom pokretanja aplikativnog servera dolazi do pokretanja web servisa koji osluškuju na odgovarajućim portovima i spremni su za obradu klijentskih zahtjeva.

Pokretanjem servera baze podataka omogućava se pristupanje i manipulacija perzistentnim podacima, slanjem odgovarajućih upita. Šema baze podataka se generiše automatski prilikom prvog pokretanja aplikativnog servera i uspostavljanja konekcije sa serverom baze podataka.

Nakon pokretanja web servera, klijentska aplikacije postaje dostupna na odgovarajućem URLu.

Kako bi se omogućio nesmetan rad aplikacije potrebno je pokrenuti i MinIO objektno skladište.

Terminacija

Nakon pokretanja aplikativnog, web i servera baze podataka nije predviđeno gašenje tih servera, osim u slučaju njihovog održavanja.

Web servisi na aplikativnom serveru su implementirani tako da se poslovna logika izvršava u vidu transakcija, tako da sistem na aplikativnom nivou ne može ostati u nekom prelaznom stanju prilikom terminacije rada sistema.

Server baze podataka i MinIO objektno skladište, takođe, imaju svoje ugrađene mehanizme koji omogućavaju sigurnu terminaciju rada.

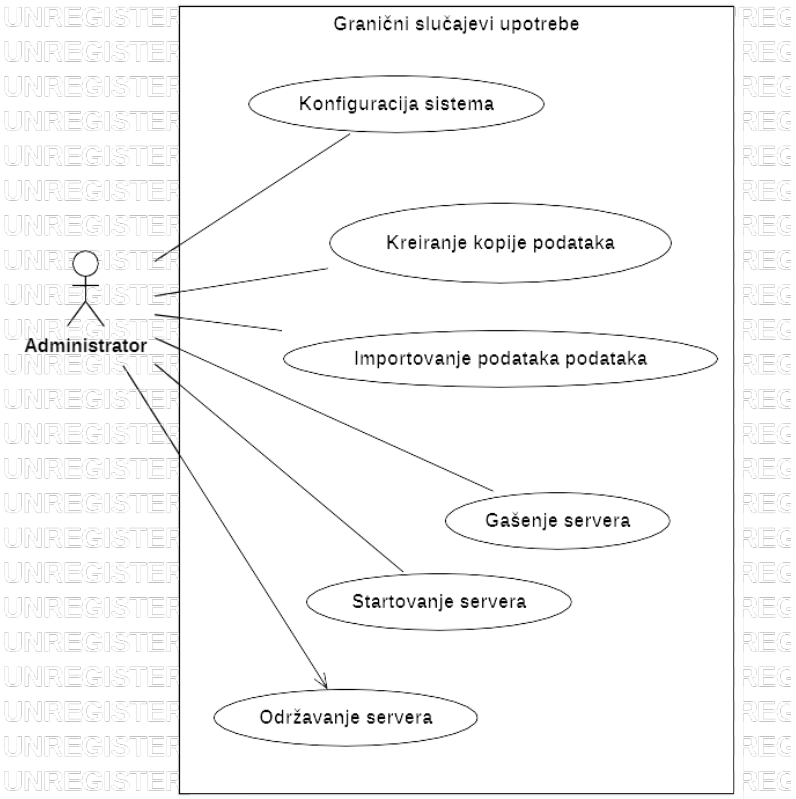
Usljed terminacije rada sistemu jedini mogući gubitak podataka jeste gubitak podataka koji su unijeti na forme grafičkog korisničkog interfejsa, koje nisu prethodno “spašene”.

Otkazi

U slučajevima kada postoje problemi sa aplikativnim ili serverom baze podataka, klijentima će putem klijentske aplikacije biti prikazano odgovarajuće obavještenje. Sve greške i otkazi se na strani aplikativnog servera bilježe u odgovarajuće log fajlove. Grafički interfejs će u tim slučajevima i dalje biti dostupan korisnicima, kao i svi podaci koji su učitani prije pojave otkaza, samo što će svi naredni zahtjevi ka aplikativnom serveru biti neuspješni.

Granični slučajevi upotrebe

Administrator sistema ima mogućnost kreiranje kopije podataka sa servera baze podatka i konfigurisanja sistema. Takođe, administrator može “uvesti” podatke u bazu podataka sistema. Dijagram graničnih slučajeva upotrebe je dat u nastavku.



Slika 5. "Dijagram graničnih slučajeva upotrebe"