Elektrotehnički fakultet Sarajevo

Razvoj programskih rješenja

Izvještaj o projektu

Aplikacija za upravljanje voznim parkom

Student: Haskić Asim

Profesor: Doc.dr Vedran Ljubović

Sarajevo, 13.09.2020.

Sadržaj:

1. Opis aplikacije

Aplikacija koja se opisuje u nastavku predstavlja aplikaciju koja se može u realnom životu i radu koristiti uz određena poboljšanja. Ideju o kreiranju ove aplikacije potiče iz porodice, i iz događaja koju su prepričani, a desili su se na poslu. Na početku aplikacija je trebala da bude veoma jednostavna i sa minimalnim akcijama, međutim kako su se implementirale akcije one su za sobom povlačile i dodatne akcije koje su također morale biti implementirane. Sada ćemo se osvrnuti na sam rad aplikacije.

Kada pokrenemo aplikaciju pojavi se prozor koji predstavlja spisak voznih parkova koje određena kompanija može da posjeduje. Na istom prozoru je omogućeno traženje, dodavanje, brisanje, izmjene i sam pristup voznim parkovima. Opcija za traženje je standardna opcija gdje možemo da unesemo podatke za pretragu određenog voznog parka, međutim ako odaberemo neku drugu opciju aplikacija će da zatraži unos korisničkog imena i lozinke. U slučaju pogrešno unesenih podataka koji se mogu odnositi na korisničko ime, lozinku ili podatke koje će korisnik naknadno da unosi tokom korištenja aplikacije, aplikacija će da obavjesti korisnika o pogrešno unesenom podacima. Kada korisnik odabere željeni vozni park i unese ispravne podatke pri prijavi korisnika, aplikacija omogućava korisniku da odabere jednu od mnogih opcija. Opcija pristup vozilima, prikazuje spisak vozila voznog parka i omogućava njihovu pretragu, dodavanje, brisanje, izmjenu ili prikaz opreme koju vozilo sadrži. Opcija pristup zaposlenim, prikazuje spisak zaposlenih osoba u odabranom voznom parku i omogućava pretragu, dodavanje, brisanje i izmjenu zaposlenih osoba. Opcija pristup skladištima, prikazuje spisak skladišta u odabranom voznom parku i omogućava pretragu, dodavanje, brisanje, izmjenu skladišta i prikaz dijelova koji se nalaze u tom skladištu. Opcija pristup servisima, prikazuje sva vozila voznog parka i omogućava pregled servisa za odabrano vozilo i naravno dodavanje, brisanje i izmjenu servisa za to vozilo. Opcija pristup opremi, prikazuje opremu koja se nalazi u vozilima zaduženim u odabranom voznom parku i omoguća dodavanje, brisanje i izmjenu opreme. Opcija pristup dijelovima, prikazuje dijelove koji se nalaze u skladištima zaduženim u odabranom voznom parku. Imamo više vrsta korisnika kao što su: glavni administrator, administrator za vozila, skladišta, servise, zaposlene i običnog radnika. U zavisnosti koji se korisnik prijavi za određeni vozni park, neke akcije mu budu dozvoljene dok neke akcije mu budu zabranjene. Također imamo i tri vrste vozila i to motorno vozilo, teretno vozilo i priključno vozilo. Naravno korisniku je omogućen prikaz određenih podataka u obliku izvještaja kojima može pristupiti poslije prijave na vozni park. Svaka od dijelova koji se nalaze u skladištu može da bude ili u ispravnom ili u neispravnom stanju. Osoba odnosno zaposleni koji može da kreira, briše ili mijenja vozni park je direktor koji je ujedno i admin cijele aplikacije, a osoba koja je odgovorna za određeni vozni park predstavlja glavnog administratora.

2. Način implementacije

Za implementaciju projekta korišten je Java programski jezik koji se uči na predmetu „Razvoj programskih rješenja“,korišten je „IntelliJ IDEA“ program, korištena je „SQLite“ baza podataka i „TIBCO Jaspersoft Studio“ za kreiranje izvještaja. Za realizaciju prozora korišteni su FXML fajlovi, koji nam omogućavaju interfejs. Naravno za modificiranje komponeniti na FXML fajlovima koristen je CSS. Pri implementaciji korištene su „obične“ Java klase, „enum klase“ i „exception“ klase. Za testiranje aplikacije kreirani su testovi koji testiraju određene mogućnosti klase.

3. Implementacija

U nastavnu teksta se opisuju pojedinačne implementacije određenih dijelova i razlog upotrebe tih komponenti i načina implementacije. Za stvari koje se budu ponavljaje u kodu, samo će biti na jednom mjestu objašnjene.

3.1 Osnovne komponente

Komponenta TableView je najviše zastupljena u prikaz podataka koje dobijemo iz baze podata za određene parametre, to možemo da vidimo na skoro svakom FXML fajlu. Za kolekciju podataka i manipulaciju sa njima najviše je korištena ArrayList, koju možemo da nađemo u skoro pa svakoj klasi. Sve klase prate JavaBeans specifikaciju.

3.2 Baza podataka

Za aplikaciju je kreirano 16 tabela, svaka od tabela sadrži primarni ključ i atribute koje ih sljeduje. Tabele imaju jasno definisanu funkciju. U aplikaciji imamo 8 tabele koje sadrže podatke dok su ostale tabele zadužene za povezivanje podataka između tabele. Za komunikaciju izmedju baze podataka i kontrolera koji su zaduženi za pojedinačne prozore koristi se klasa „MotorFleetDAO“.

3.3 Izvještavanje

Aplikacija koristi 6 izvještaja i to izvještaj za vozila, zaposlene, skladišta, servise, dijelove i opremu. Kontorleri komuniciraju sa klasom “Report“ koja u zavisnosti od akcije kontorlera poziva odnosno prikazuje određeni izvještaj. Svaki od izvještaja prikazuje samo određene podatke (neki čak i sve), iz razloga što osobe koje koriste izvještaje ne smiju da vide sve podatke. Primjer je kod izvještaja za zaposlene osobe gdje se ne prikazuje korisničko ime i lozinka. Također na izvještajima određena polja prikazuju različite podatke nego u bazi podataka, primjer za to je kod tipa vozila gdje se prikazuje motorno vozili, teretno vozilo ili priključno vozilo dok u bazi je to označeno sa indeksima od 1 do 3.

3.4 GUI

Na svakom od prozora korišten je GridPane, dok su najčešće korištene komponente TableView, Button, TextField itd. Svaki od prozora ima svoj kontroler koji komunicira sa FXML fajlom, komunicira sa ostalim klasama i bazom podataka preko „DAO“ klase. Mnogi prozori su iskorišteni za dvije funkcije, naprimjer kod dodavanja i izmjene određenih podataka. Također kada korisnik unese pogrešne podatke pri prijavi, unosu novih podatak ili izmjeni već postojećih podataka aplikacija prikazuje „Alert“ prozor koji upozorava korisnika o problemu. U aplikaciji imamo i definisane testove koji testiraju GUI same aplikacije.

3.5 Enumi

Aplikacija koristi jednu enum klasu koja se zove „LevelOfRensponsibility“. Klasa definiše koje vrst korisnika odnosno koje vrst zaposlenih možemo da imamo u aplikaciji. Svaki tip korisnika ima određene privilegije i određene zabrane koje su definisane u pojedinačnim kontrolerima.

3.6 Ostalo

Što se tiče nasljeđivanja to je prikazano kroz klasu „Vozila“ iz koje su nasljeđene tri dodatne klase. Svaka od klasa „krije“ svoje podatke i nije im moguće direktrno pristupiti. Što se tiče izuzetaka prilikom unosa pogrešnih podataka vlastiti izuzetak će biti bačen od strane aplikacije sa odgovarajućim tekstom.

3.7 Komunikacija

Ovdje ćemo da objasnimo komunikacije između klasa. Naime kada nam se prikaže prozor sa određenim mogućnostima i kada kliknemo na neko dugme odnosno kada odaberemo željenu akciju, ta ista akcija se u kontroleru poziva te nakon toga kontroler komunicira sa „DAO“ klasom ili sa nekim drugim klasama. Kao primjer ćemo uzeti dodavanje novog vozila u odabrani vozni park. Pretpostavnimo da je korisnik odabrao akciju za unos novog vozila i da je popunio potrebne podatke. Kada korisnik klikne na dugme „Save and Back“, sva polja koja je korisnik popunio prvo se provjeravaju da li su validni. Uzimamo i pretpostavku da je korisnik popunio valide odnosno ispravne podatke i da neće doći do bacanja izuzetka ili do otvaranja novog „Alert“ prozora. Kada se pozove akcija za dugme „Save and Back“ koja se nalazi u odgovarajućem kontroleru, tada aplikacija poziva metode u klasi „DAO“ putem kojih te podatke stavlja na bazu podataka koristeći prosljeđene parametre. Na kraju kada se podatak unese u bazu podataka i kada se metoda vrati u kontroler na mjesto gdje je bila pozvala, aplikacija završava svoj rad i vraća se ne prethodni

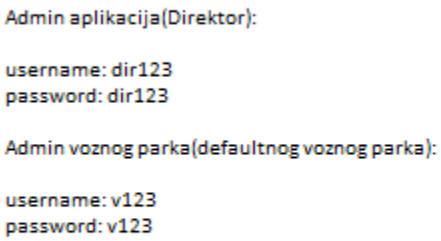
prozor.Također i za ostale akcije kao što su brisanje ili izmjena podataka, odgovarajući kontorler poziva „DAO“ klasu koja vrši izmjene u bazi podataka.

Klasa „DAO“ ima najdužu implementaciju, a razlog tome je što klasa „DAO“ mora da vrši dosta promjena, dosta insertovanja, brisanja i izmjena u bazi podataka koja ima također dosta klasa. Tako da za dodavanje ili brisanje nekog podatka, „DAO“ klasa mora vršiti izmjene u dvije ili tri tabele.

4. Unaprjeđivanje

U aplikaciji nije bilo potreba za kreiranjem interfejsa ili datoteka, što ne isključuje mogućnost da se u budućnosti javi potreba za tim. Naravno svaka od klasa može da sadrži jos parametara, samim tim i u tabelama baze podataka. Moguća su unaprjeđivanja i modifikacije prozora kao i njihova bolja sihronizacija. U aplikaciji nije korišteno generičko programiranje, mrežno programiranje, tredovi, marven, lokalizacija i jos neke dodatne stvari, ali naravno sve to je moguće implementirati u aplikaciju ako za tim bude bilo potrebe.

5. Dodatak



Slika 1: Nalog za log in