Izvještaj za projekat iz predmeta:

Razvoj programskih rješenja

Tema:

Skladište

Univerzitet u Sarajevu
Elektrotehnički fakultet Sarajevo
Odsjek za računarstvo i informatiku
Razvoj programskih rješenja

Profesor: Doc. dr Vedran Ljubović, dipl.ing.el
Student: Mehmedović Faris
Ak. god. 2019/2020.

Sadržaj:

1. Opis aplikacije

Aplikacija koja se opisuje u nastavku ovog rada predstavlja jednu moguću formu simulacije aplikacije za upravljanje skladištem sa određenim ograničenjima. Dizajn aplikacije je kreiran sa ciljem da bude jednostavan, jasan i praktičan za različite generacije. Naime, procjenjuje se da je u 2018. godini 2.65 milijardi ljudi koristilo društvene mreže, te se očekuje da će taj broj porasti i do 3.1 milijardu do 2021. Shodno tome, koncept rada aplikacije sličan je konceptu na kojem rade društvene mreže, što ne bi trebao predstavljati problem za korisnika/cu.

Početni interfejs koji korisnik susreće je prijava na njegov profil (account), a u slučaju da korisnik nema profil ponuđena je opcija kreiranja novog. Ukoliko korisnik želi kreirati novi profil neophodno je da popuni osnovne podatke o korisniku/ci, te unese adresu i naziv skladišta. Neophodno je da korisnik unese korisničko ime takvo da nema podudaranja sa već postojećim korisnicima, kao i da unese validan e-mail i šifru sa minimalno 9 karaktera (barem jedan broj). Nakon obavljenih koraka, korisnik ulazi u Home page (isti ulaz će biti nakon login-a postojećeg korisnika). Unutar Home page-a nalaze se osnovne informacije o skladištu, kao i generalne informacije o proizvodima, tj. brojno stanje i ukupna vrijednost proizvoda na skladištu, te lista kada, koliko i kojih proizvoda je uneseno. Naime, izborom *My Products* opcije sa lijeve strane ekrana, otvara se novi prozor sa prikazom imena proizvoda, količine i pojedinačne cjene. Korištenjem posebnih skala moguće je uvećavati i smanjivati brojno stanje po količini i cjeni, a što se jasno prikazuje u tabeli pojedinačnih stavki. Ovo je vrijedna opcija, koja vizuelno olakšava rad korisnika. Korisniku je omogućeno da mijenja brojno stanje proizvoda, da ih uređuje (*Alter*), dodaje (*Add*) i uklanja sa skladišta (*Remove*) u formi posebnih dugmića.

Izborom opcije *Find about other users* unutar Home page-a, moguće je vidjeti ostale korisnike drugih skladišta (izvještaj) s ciljem ostvarivanja poslovne komunikacije. Omogućene su opcije da korisnik može spasiti trenutnu listu ažuriranja proizvoda sa klikom na *Save updates* dugme (binarna ili XML datoteka), kao i pregledati spašena ažuriranja klikom na tipku *Load updates* (binarna ili XML datoteka).

Pri loginu je moguće birati jezik aplikacije (npr. bosanski i engleski). Shodno tome da se radi o demo verziji aplikacije, korisniku se preporučuje ukoliko ne želi da kreira novi profl, da koristi gotovi profil sa korisničkim imenom (username) “korisnik1” i lozinkom (password) “sifra123”.

2. Osnovni koncepti

2.1 Objektno zasnovano programiranje u Javi

Unutar projekta, paket *sample* se koristi kako bi se veći broj klasa organizirao u neku tematsku cjelinu. Izuzetci se koriste u aplikaciji, pored toga kreirana je vlastita dodatna klasa izuzetaka *WrongProductDataException* koji se baca ukoliko korisnik pokuša unijeti predmet s količinom 0.

(Predmeti sa cijenom 0 su dozvoljeni jer doista prostor mogu zauzimati predmeti koji su bezvrijedni). U projektu se koriste model klase koje prate JavaBeans specifikaciju.

Pobrojani tip (enum) zastupljen je u svrhu deklarisanja da li je rok isteka trajanja proizvoda (broj mjeseci) prošao ili ne. Tokom realizacije projekta uočena je potreba za korištenje interfejsa (npr. Serializable) ali i kreiranja vlasitog (*ProductOperations*) u svrhu implementiranja metoda za analizu podataka. Zastupljena je upotreba klase izvedene iz interfejsa List - ArrayList, kao i HashMape koja je korištena radi svoje efikasnosti pristupanju registrovanih korisnika.

2.2 Funkcionalno programiranje

Prilikom poziva inicijalizacije unutar controllera *LogController*, *ProductController* korištene su lambda funkcije u slučaju listenera. Upotreba streamova je zatupljena unutar projekta, naročito je podržan funkcionalni pristup programiranju preko streamova u *HomePageController* prilikom implementacije metoda interfejsa *ProductOperations*.

2.3 Grafički interfejs

Prilikom implementiranja grafičkog interfejsa korišteni su razni GUI elementi kao što su: *AnchorPane*, *Label*, *TextField*, *PasswordField*, *ChoiceBox*, *TableView* itd. Kontrola i specifikacija navedenih je obrađena unutar Controllera. Uzeti su u obzir koncepti dobrog dizajna korisničkog interfejsa (Gestalt principi) gdje su svi logički vezani elementi grupisani i na ekranu. Za prikaz validnih/nevalidnih polja pozivamo se na design.css i u kontroleru se ovisno o validnosti postavlja odgovarajuća boja polja.

2.4 Baza podataka

Kao što je već navedeno, korištena je SQLite baza podataka. Za potrebe projekta kreirano je 5 tabela. Uz ovaj rad nalazi se ER dijagram (u vidu slike) kreiran uz pomoć LucidCharts online alata. Tabele warehouses, products i users su intuitivno jasne, tako da neće zahtijevati objašnjenje. Tabela warehouse_products služi kao tabela veza s ciljem kako bi se implemantiralo povezivanje m:n između tabela *warehouses* i *products*. Dalje tabela *product_updates* ima ulogu da zapisuje promjene unutar skladišta s ciljem kako bi korisnik mogao imati uvid na home page-u aplikacije koliko, kada i po kojoj cijeni je unio podataka.

Rad sa bazom počinje sa controllerom *LogController* koji ima ulogu prijave na aplikaciju. Nivo iznad je klasa *WarehouseDAO* koja je zasnovana na singleton šablonu s ciljem da se izbjegava da imamo više različitih pogleda na iste podatke, ova klasa kreira konekciju sa bazom, te omogućava dodavanje, ažuriranje i uzimanje podataka iz SQLite baze podataka.

2.5 Singleton klase

Klasa *WarehouseDAO* predstavlja singleton klasu. Ona može imati samo jednu instancu kako ne bi došlo do komplikacija prilikom pristupa bazi. Realizovana je kao singleton na način da kao privatni statički atribut ima referencu na WarehouseDAO, konstruktor joj je privatan i sadrži javne metode za dohvaćanje i uklanjanje date reference. Dohvaćanje reference se vrši pozivom metode getInstance koja provjerava da li je atribut instance jednak null, u slučaju da jeste inicijalizuje se, te se data instanca vraća. Uklanjanje instance se vrši pozivom metode removeInstance koja postavlja atribut instance na null. Prilikom inicijalizacije ove klase pripramaju se upiti koji će biti potrebni za izvršavanje metoda pomoću kojih se dobavljaju podaci iz baze. Ova klasa je implementirala klasu *WarehouseDAO* koja predstavlja interfejs sa sljedećim metodama.

2.6 Unit testovi i TestFX u Javi

Izvršeno je testiranje niskog nivoa (unit testing) nad klasama *Product*, *User* i *Warehouse* koristeći JUnit 5. Testiranje grafičkih korisničkih aplikacija koristeći JavaFX framework implementirano je unutar projekta pomoću biblioteke TestFX. Svrha navedenog je da simuliraju ponašanje korisnika prilikom rada sa aplikacijom, te da pokriju što veću količinu koda unutar aplikcije.

2.7 MVC sa JavaFX

Implementiran je MVC šablon koji odgovara na pitanje organizacije koda u aplikacijama koje posjeduju UI. Konkretan primjer vidimo na proizvodima, gdje imamo *ProductModel* –model klasa, *ProductController* – kontroler klase te *Product* – view klasa. Također unutar klase *ProductController* implementirano je dvosmjerno uvezivanje TextField-a sa atributom TableView.

2.8 Datoteke

Korisiti se FileChooser klasa kojom je omogućeno pozivanje dialoga za spašavanje podataka u Text File datoteku. Podaci se spašavaju klikom na Save (MenuItem) unutar menija u prozoru za obrađivanje proizvoda na skladištu (products.fxml). Zastupljena je serijalizacija u radu sa binarnim datotekama. Klase korištene za serijalizaciju su ObjectInputStream i ObjectOutputStream, u svrhu pamćenja prethodnih ažuriranja vezanih za proizvode unutar skladišta. Također implementirana je XML serijalizacija pomoću omotačkih klasa XMLEncoder i XMLDecoder. Svrha XML serijalizacije je ista kao kod binarne, uzimajući u vidu da korisniku je omogućena druga opcija u svrhu rada sa aplikacijom.

2.9 Mrežno programiranje

Provjera podataka korisnika poput e-maila realizirana je uz pomoć klase URL, te su omogućene

metode za pristup pojedinačnim komponentama. U slučaju da URL nije validan baca se izuzetak, te se od korisnika traži ponovni unos podataka.

2.10 Višenitno programiranje

Koristimo niti (eng. threads) unutar klase *addWarehouseController*. JavaFX interfejs se izvršava (dodavanje skladišta u bazu podataka) u drugoj niti kako se ne bi desilo da se ekran prestane osvježavati dok se izvršava neki zadatak u pozadini.

2.11 Izvještaji

Prilikom pravljenja izvještaja Jasper Reports, koje predstavlja open source rješenje pravljeno je u Javi. Cilj izvještaja je prikaz drugih skladišnih jedinica s ciljem prezentovanja firme, kao i svojevrsne reklame aplikaciji. Klasa *WarehousesReport* koja služi za izvještaj ostalim skladištima. Sadrži jednu metodu *showReport* koja prima konekciju na bazu iz koje učitava podatke. Unutar izvještaja se prikazuju podaci o skladištima, te njihovim proizvodima respektivno, pri čemu se ne prikazuju informacije o korisnicima radi zaštite anonimnosti pojedinca.

2.12 Internacionalizacija

Prilikom internacionalizacije kreiran je ResourceBundle koji sadrži potrebne datoteke za internacionalizaciju projekta, gdje su engleski i bosanski jezik uzeti kao jezici koji su podržani u ovom projektu. Internacionalizacija je implementirana pri početku rada korisnika sa aplikacijom.

3. Prilike za unapređivanje

Prilikom rada na ovom projektu uočio sam brojne mogućnosti za poboljšanje. Za ovakav sistem aplikacije da bi bio upotrebljiv na business nivou, neophodno je da podaci budu implementirani na bazama podataka većih razmjera, kao i uvezivanje korisnika koji rade za iste kompanije, ili za sestrinske kompanije u okviru savremene digitalizacije u industriji. Također neophodno bi bilo poboljšati aplikaciju na nivou privatnosti po osnovi Internet of Things (IoT) sistema, kao i način rada koji bi predvidio situacije koje bi potencijalno narušile funkcionisanje aplikacije.

Istražujući na internetu, moguće je nadograditi RFID tagovanje proizvoda, tj. identifikaciju proizvoda sa kodovima. Korištenjem RFID čitača ili kamera, ubrzao bi se unos, odnosno izmjena na stanju proizvoda na skladištima, što bi uz mnogobrojne druge komponente (primjene industrijskih, mobilnih robota za transfer robe u skladištu industrijskih velikih zona) značajno olakšalo rad korisnika. Radi uskog radnog prostora, ovakve stvari nisu implementirane, međutim u eri digitalizacije, treba ih imati na umu.