Elektrotehnički fakultet Sarajevo

Razvoj programsih rješenja

Izvještaj o projektu

Sistem upravljanja skladištem

Student : Amina Babić

Predmetni nastavnik: Doc.dr.Vedran Ljubović

Sadržaj

1 Opis aplikacije

Aplikacija koja će se obrađivati u sklopu ovog rada je jednostavna i poprilično ograničena aplikacija koja omogućava jednostavno organizovanje, praćenje proizvoda i izvještavanje.

Korisnik aplikacije može imati jedan od tri pristupna nivoa kojim se odvajaju poslovi korisnika.

Firme koje koriste aplikaciju mogu samo pratiti stanje svojih proizvoda, te modifikovati svoj profil.

Zaposlenici su oni koji su odgovorni za stanje proizvoda.

Ukoliko se proizvod naruči ili smjesti u skladište, njihova uloga je da promijene stanje tog proizvoda u sistemu, ili da proizvod dodaju ukoliko on ne postoji.

Admin je onaj koji je zadužen za upravljanje računima i organizovanjem skladišta.

Stoga, mogućnost da se doda novi zaposlenik, firma, kategorija ili odjel ima samo admin.

Nakon što unese informacije o korisniku aplikacije, izdaje izvještaj sa pristupnim podacima i potpisom. Admin može dobiti uvid u spisak korisnika aplikacije i trenutnih proizvoda u sistemu.

2 Osnovne ideje i navike pri implementaciji

Ukoliko koristimo neku aplikaciju, želimo da ona izgleda lijepo I da je ugodna I jednostavna za koristiti.

Pošto je izgled aplikacije jako bitan za korisnike, jer im on na neki način pruža “ugodnost”, dosta truda je uloženo u izgled.

Aplikacija je poprilično jednostavna i korisnik se može jako brzo upoznati s njom, tj. očigledno je šta za šta služi.

Praćeno je da funkcionalnosti aplikacije budu zadovoljive I sa što manje problema.

Međutim, vremenom se shvati da se uvijek moglo nešta bolje realizovati.

U aplikaciji imamo sliku objekata iz baze kao DTO klase koje prate JavaBeans specifikaciju.

Također, kako bi se odvojila baza podataka od business logike, kreirane su model klase I to obično tako da za svaki kontroler postoji odgovarajuća model klasa.

Kvaliteta koda

Prilikom rada, metode koje bi bile previše dugačke su extract-ovane u manje, te je time omogućeno lakše shvaćanje logike i uloge svake metode.

Prilikom imenovanja klasa, metoda, datoteka(fxml, css, bundle) , promjenljivih korištena su imena koja su logički vezana za njih, tj. ime svakog od njih prati njihovu ulogu.

Potrudila sam se da projekat prati MVC arhitekturu.

Kod I projekat su organizovani tako da su resursi u resources folderu I po potrebi su razdvojeni u podfoldere, u src su klase koje su također , po potrebi, razdvojene u foldere/podfoldere .

3 Implementacija

3.1 Baza podataka

Korištena je SQLite baza podataka. Za potrebe projekta kreirane su 4 tabele.

Tabele su poprilično jednostavne, te njihovo objašnjenje nije neophodno.

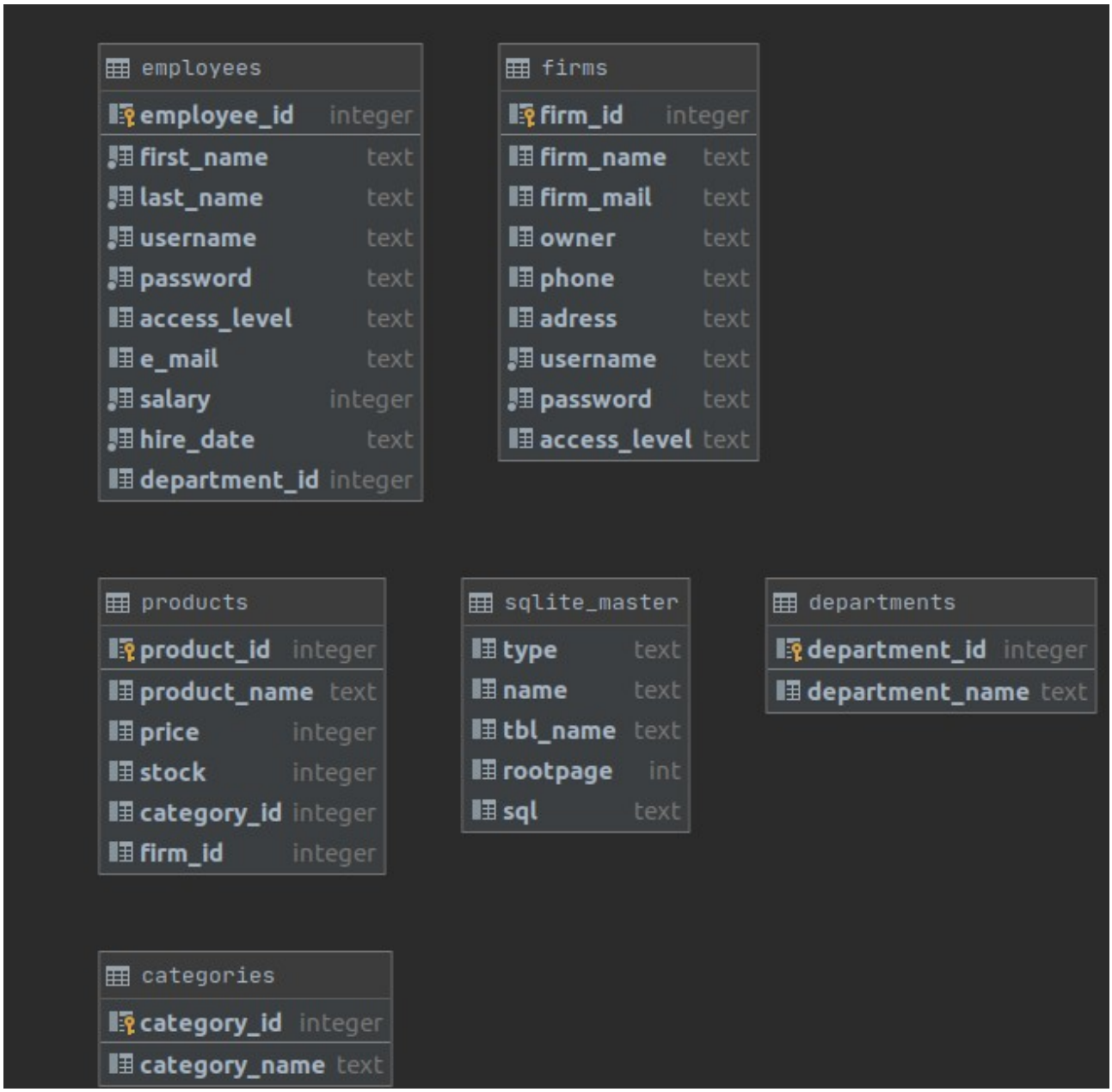
U prilogu se nalazi ER dijagram geenrisan od strane Intellij okruženja.

Rad s bazom počinje sa samim Log In / logovanjem u sistem.

Kako bi korisnik pristupio sistemu, potrebno je provjeriti da li on već postoji u bazi, inače bi log in bio besmislen i svako bi mogao pristupati.

Rad s bazom se nalazi u DAO klasama. Ovim su odvojeni svi upiti, te ukoliko bude potreba za nadogradnom sistema nije potrebno lutati kroz kod i tražiti na koji način pristupiti bazi.

Kao primjer se može uzeti dodavanje firme. Za ovu akciju nije bilo vremena da se realizuje u sistemu. Međutim, ukoliko se javi potreba, to ne bi trebao biti problem s obzirom da već postoji FirmDAO klasa u kojoj postavljamo upit, model klasa kroz koju dobavljamo ili ubacujemo potrebni podatak i preko kontrolera šaljemo/ primamo podatke na/ sa view-a.



3.2 Koncepti OOP-a

Kako su i zaposlenik, i firma korisnici aplikacije, u model klasama se vidi nasljeđivanje.

Employee i Firm nasljeđuju User. Enkapsulacija polja u svim klasama omogućava da se odvoji javni od privatnog dijela klasa.

3.3 Grafički interfejs

Iako je aplikacija poprilično jednostavna, po nekom pravilu aplikacija treba da sadrži Help. Te je iskoripten **menu bar** sa **menu items**. Klikom na help korisnik može da sazna nešto više o korištenju aplikacije(npr. Shortcuts), ali i općenito o aplikaciji(About).

Kako bi se dao vizuelan ugođaj, ali i prirodni osjećaj za određenie akcije, korištene su dosta slike, te samim tim i **ImageView.**

Prilikom log in-a za password je korišten posebno **Password fiel**d radi sigurnosti prilikom unošenja šifre, ali I u slučaju pogrešnog unosa, korisniku je omogućeno da vidi svoju šifru jednim klikom na **CheckBox**(iskorišten još prilikom search-a).

Kako bi korisniku bilo objašnjeno značenje pojedinih elemenata, korišteni su i **ToolTip.**

Pored toga, korišteni su:

Button, RadioButton, TextArea, Label, GridPane,Vbox, Hbox, TextField, ChoiceBox, DatePicker. TableView, AnchorPane, ToggleGroup.

U obzir su uzeti Gestalt principi tako da su I oni logički povezani elementi grupisani na ekranu.

VALIDACIJA

U sklopu validacije korišten je DatePicker kako bi se osigurao odabir validnog formata datuma(npr. edit employee).

Za prikaz validnih/nevalidnih pola korišteni su :

- .css file kojim se postavlja odgovarajuća boja polja (npr. prilikom Log In-a)
- Alert dialog (npr. prilikom unosa slova u search nakon što je čekirano pretraživanje po ID-u)
- poruke ostvarene preko dynamic fxml (nor. Unosom cifre u ime zaposlenika prilikom edit-a, pojavi se poruka da ime ne može sadržati brojeve I sl.)

Pažnja se obratila na to da prozori budu resizable i da se prilikom toga i ponašaju pravilno.

3.4 Datoteke

Prilikom brisanja produkta ili smanjivanja zalihe, produkt se zapisuje u fajl Ship.txt iz kojeg se kasnije čita kako bi se ti produkti mogli vidjeti klikom na dugme Shipping.

Ovim je omogućeno praćenje proizvoda. Na listi se može vidjeti datum brisanja proizvoda, kao I ostali podaci o njemu.

Rad s ovom datotekom se nalazi u klasi ShipmentController (čitanje),

kao i u ItemDetailsController(pisanje u datoteku)

```
private void writeToFile(Product product, int num) ;

private void readFile();
```

3.5 Enumi, interfejsi i izuzetci

U projektu se koristi jedan enum - AccesLevel kojim razlikujemo Admina i "običnog" zaposlenika.

U projektu se koriste dva interfejsa u kojim se nalaze često korištene metode između nekih klasa, te je ovim interfejsima smanjeno ponavljanje koda (neke su I default, tipa openNewStage), ali i omogućen lakši pregled šta pojedine klase koriste/ rade.

U DAOInterface postoji metoda baseRegeneration() koja može da baci vlastiti tip izuzetka FailedBaseRegeneration. Pošto baza predstavlja velik faktor I igra veliku ulogu u ovoj aplikaciji, svaki problem s njom je bitno dobro definisati i znati gdje je tačno problem. Bitno je imati dobro definisan izuzetak.
Ovim izuzetkom je to omogućeno i bilo je od pomoći na dosadašnjem radu na aplikaciji.

3.6 Mrežno programiranje i tredovi

Mrežno programiranje je iskorišteno za dohvatanje datuma sa određenog servera.

Konkretno u HomeController klasi imamo metodu setDate() kojom se postavlja datum na formu.

Nova nit je iskorištena u cilju prikazivanja trenutnog vremena. Kako se vrijeme prikazuje svake sekunde, da ne bi došlo do remećenja drugih aktivnosti, "otvoren" je novi tred. Nakon neuspješnog kreiranja socket-a koji bi koristio LocalDateTime, iskorišten je samo LocalDateTime za prikaz trenutnog vremena.

Ova nit se nalazi u HomeController → timeThread().

3.6 Funkcionalno programiranje i kolekcije

Lambda funkcije su dosta korištene u slučaju listenera.

S druge strane korišteni su i stream-ovi, a ujedno i kolekcije (ArrayList).

Jedan od primjera korištenja strema je prilikom search-a određenoj proizvoda / zaposlenika.

Od kolekcija je korištena i mapa.

```
private HashMap<String, String> views = new HashMap<>();
```

Ova kolekcija je korištena u HomeController-u i ima jako dobru ulogu koja se može protumačiti iz koda.

Ukratko, ona “mapira” određeni button sa view-om koji se treba prikazati kada se klikne na button.

3.7 Izvještavanje

Izvješaji su implementirani korištenjem Jaspersoft Studio alata za kreiranje izvještaja.

Prilikom brisanja proizvoda ili smanjenjem zalihe, kreira se izvještaj kao potvrda.

Ovo je u pozadini, izvedeno proslijeđivanjem parametra (id proizvoda koji je odabran).

Također, prilikom unosa novog zaposlenika I klika na duge Save, admin ima opciju da kreira izvještaj kao potvrdu i ugovor. U ovom izvještaju se nalaze pristupni podaci za novog zaposlenika, te prostori za potpise.

Ovo je u pozadini, izvedeno proslijeđivanjem parametra (id novododanog zaposlenika).

Admin može na svom panel-u odabrati još neku od opcija:

-Izvještaj o trenutnim zaposlenicima, firmama i proizvodima u sistemu/skladištu

3.8 Lokalizacija

U resources se nalaze budle-s kojim je omogućen prevod aplikacije na dva jezika : bosanski i engleski.

Kao default jezik je odabran engleski.

Klikom na duge bs/en vrši se prevod aplikacije na odabrani jezik.

Ovo je omogućeno u klasi HomeController gdje se nalazi jedna varijabla

boolean bs koja nosi informaciju da li je odabran bs, te na osnovu toga se postavlja odgovarajući bundle.

3.8 Generičko programiranje

Iako su “gotove” generičke klase korištene dosta kroz kod. Kreirana je I jedna vlastita nazvana Reference. Ova klasa je nastala kada se pokušavala Connection conneection poslati po referenci u odgovarajuću klasu I ishod nije bio uspješan.

Nakon guglanja saznala sam da je Java uvijek “pass-by-value”, te sam kreirala generičku klasu koja je riješila ovaj problem.

```
connectToBase(connReference);
```

Korištene su generičke metode (uglavnom kroz lambde u listener-ima).

Primjer :

```
departmentChoice.getSelectionModel().selectedItemProperty().addListener(new
        ChangeListener<Department>() {
        @Override
        public void changed(ObservableValue<? extends Department>
        observableValue, Department department, Department t1) {
                model.getEmployees().get(page)

                .setDepartment(departmentChoice.getSelectionModel()

                .getSelectedItem());
    }
```

PRISTUPNI PODACI

username : river

password : hello12345