Kielce, 13.06.2021r.

Politechnika Świętokrzyska w Kielcach

Studia Stacjonarne Pierwszego Stopnia

Kierunek: Informatyka II rok (semestr IV)

Grupa: 2ID12B

Programowanie w języku Java

Temat projektu:

Aplikacja (standalone) do prowadzenia oraz zarządzania hurtownią sprzętu RTV/AGD

Zespół:
Kwiatkowska Aleksandra
Kot Jarosław

Spis treści

1. Opis projektu

Tematem naszego projektu jest aplikacja służąca do prowadzenia i zarządzania hurtownią sprzętu RTV/AGD. Projekt został stworzony jako projekt Gradle z wykorzystaniem Intellij IDEA jako środowiska programistycznego. Aplikacja jest przeznaczona dla pracowników hurtowni. Założyliśmy, że pracownik dostaje swoje dane do logowania od pracodawcy i nie musi już zakładać swojego indywidualnego konta. Dane takie jak informacje o loginie i haśle do konta czy o dostępnym sprzęcie są przechowywane w bazie danych. Została ona utworzona z wykorzystaniem programu XAMPP Control Panel w języku MySQL. W momencie uruchomienia aplikacji i prawidłowego zalogowania się na konto baza zostaje wczytana i wyświetlona w odpowiedni sposób. Naszym celem było, aby każdy użytkownik mógł swobodnie się poruszać i obsługiwać aplikację. Zawiera ona podstawowe funkcje, które są niezbędne do działania takiej hurtowni sprzętów. Ich funkcjonalność została przedstawiona w dalszej części.

- **Technologie**

Przy tworzeniu projektu wykorzystaliśmy technologie takie jak:

- Baza danych MariaDB (MySQL);
- Java 8+;
- JavaFX
- JDK16

- **Frameworki**

- Gradle
- Intellij IDEA
- JUnit 5
- XAMPP Control Panel

- **Biblioteki**

```
javafx.application.Application;
javafx.fxml.FXMLLoader;
javafx.scene.Parent;
javafx.scene.Scene;
javafx.stage.Stage;
javafx.stage.StageStyle;
java.io.IOException;
com.mysql.jdbc.PreparedStatement;
javafx.collections.FXCollections;
javafx.collections.ObservableList;
javafx.fxml.FXML;
javafx.fxml.Initializable;
javafx.scene.control.*;
javafx.scene.control.cell.PropertyValueFactory;
```

- `javax.swing.*;`
- `java.net.URL;`
- `java.sql.Connection;`
- `java.sql.ResultSet;`
- `java.sql.SQLException;`
- `java.sql.Statement;`
- `java.util.Objects;`
- `java.util.ResourceBundle;`
- `javafx.scene.control.Label;`
- `javafx.scene.control.PasswordField;`
- `javafx.scene.control.TextField;`
- `java.sql.Statement;`
- `javafx.application.Platform;`
- `javafx.scene.control.Button;`
- `java.sql.DriverManager;`
- `javafx.scene.effect.*;`
- `javafx.scene.layout.*;`
- `javafx.scene.text.*;`
- `javafx.scene.image.*;`
- `import org.junit.jupiter.api.Test;`
- `import static org.junit.jupiter.api.Assertions.*;`

2. Funkcjonalność projektu

Aplikacja zawiera niezbędna funkcje to prawidłowego działania i zarządzania hurtownią sprzętu RTV/AGD. W skład funkcjonalności wchodzą:

- Logowanie do konta;
- Wyświetlenie wszystkich sprzętów będących w hurtowni, z podziałem na kategorie takie jak: AGD duże, AGD małe, komputery, konsole i gry, smartfony oraz telewizory i urządzenia audio;
- Wyświetlenie zamówień;
- Dodawanie sprzętu do hurtowni – należy podać takie informacje o nazwie sprzętu i nazwie producenta, cenę i raty oraz krótki opis na jego temat;
- Dodawanie zamówień – należy podać informacje o kliencie, jego imię, nazwisko oraz adres zamieszkania, adres e-mail, sposób płatności oraz datę złożonego zamówienia;
- Usuwanie – użytkownik wybiera wpis, który chce usunąć;
- Edycja – tak jak przy usuwaniu, należy wybrać wpis a dane zostaną wyświetlone i gotowe do modyfikacji;
- Sortowanie alfabetyczne lub numeryczne po wybranym polu;
- Wyszukiwanie po nazwie urządzenia, producencie oraz ilości rat;
- Działanie na bazie danych;

3. Sposób uruchomienia i obsługa

Projekt został stworzony jako projekt Gradle z wykorzystaniem Intellij IDEA jako środowiska programistycznego w wersji 2020.3.2. Użytkownik z zewnątrz, aby skorzystać z naszego projektu potrzebuje plik AplikacjaSklep-1.0-SNAPCHOT.jar. Należy go otworzyć przy pomocy programu Intelilij IDEAl. Znajduje się w folderze /build/libs. Natomiast, aby uruchomić także kod programu należy skorzystać z pliku build.gradle, a wraz z nim z pozostałymi plikami potrzebnymi do działania aplikacji. Wszystkie niezbędne pliki znajdują się w folderze 'AplikacjaSklep'. Przed włączeniem aplikacji należy najpierw skorzystać z program XAMPP i uruchomić w nim dwa moduły: Apache oraz MySQL. Plik sklepртv.sql zawiera potrzebne skrypty do importowania bazy danych.

Przed uruchomieniem pliku .jar należy dodać folder 'javafx-sdk-16' do /Program Files/Java. Następnie w programie Intelij należy przejść do Run->Edit Configurations -> Modify options -> Add VM options. W polu trzeba podać następującą ścieżkę: *--module-path "C:\Program Files\Java\javafx-sdk-16\lib" --add-modules javafx.controls,javafx.fxml* . Po wykonaniu tych kroków wystarczy odnaleźć plik .jar i uruchomić.

Aplikacja została stworzona w taki sposób, aby każdy użytkownik mógł swobodnie się nią posługiwać. Po uruchomieniu aplikacji należy się zalogować. Na potrzeby projektu stworzyliśmy przykładowego użytkownika o danych *Admin/admin*. W panelu logowania pojawi się komunikat w przypadku błędnych danych.

W oknie menu głównego mamy do wyboru sześć kategorii sprzętów, zamówienia oraz opcja wyjścia z menu, czyli zamknięcie aplikacji i opcja wyloguj, czyli powrót do panelu logowania.

Dla przykładu po wybraniu w menu opcji *'AGD duże'* pojawi się okno, w którym zostają wyświetlone wszystkie sprzęty znajdujące się w hurtowni mające kategorie AGD duże. Użytkownik po lewej stronie okna ma szereg opcji, które może wykonać:

- W pierwszym polu tekstowym znajduje się wyszukiwanie. Użytkownik podaje część frazy, której szuka a prawidłowe wpisy zostają wyświetlone. Wyszukiwać można po polach takich jak: nazwa urządzenia, nazwa producenta oraz ilość rat.
- Aby dodać nowy wpis należy uzupełnić wszystkie potrzebne pola a następnie nacisnąć przycisk *'DODAJ'*. W ten sposób dołączy do hurtowni nowy sprzęt o tej kategorii.
- Aby usunąć lub edytować wpis, należy z listy wyświetlonych wybrać jeden sprzęt. Jego dane zostaną wyświetlone w tym samym miejscu, w którym dodajemy urządzenie. Następnie wystarczy wybrać odpowiedni przycisk, w zależności czy chcemy usunąć wpis z bazy czy go edytować.

W przypadku pozostałych kategorii sprzętów dostępnych w menu głównym obsługa wygląda tak samo.

4. Funkcjonalność oraz przeznaczenie stworzonych klas, metod i funkcji

- **Klasa *Aplikacja***

Jest to główna klasa, która stanowi podstawę całego programu. Z użyciem bloku try..catch jest uruchamiane okno, w którym użytkownik się loguje.

- **Klasa *agd1* i klasa *zamówienia***

Obie klasy służą do zdefiniowania potrzebnych zmiennych i ich typów danych do prawidłowego działania aplikacji sklepu. Znajdują się w nich konstruktory oraz gettery i settery, które pozwalają na pobranie i modyfikację wartości zmiennej.

- **Klasa *Polaczenie***

Klasa ta służy do uzyskania połączenia z bazą danych. Przy użyciu bloku *try…catch*, który służy do obsługi wyjątków zostaje podejmowana próba nawiązania połączenia. Na początku zostają zarejestrowane sterowników przy użyciu metody *Class.forName()*. Pozwala na dynamiczne załadowanie plików klasy sterownika do pamięci, która automatycznie go rejestruje. Metoda *DriverManager.getConnection()* pozwala na utworzenie obiektu połączenia. Jako argument zostaje przekazany adres URL do bazy danych. W przypadku niemożliwości nawiązania połączenia zostają wywołane metody *printStackTrace()* i *getCause()*, które informują co jest przyczyną błędu.

- **Klasa *KontrolerLogowania***

Służy ona do logowania się do aplikacji sklepu. Znajdują się w niej odwołania do pliku *main.fxml*, w którym jest graficzny projekt panelu logowania.
W funkcji *loginButtonOnAction()* w instrukcji *if* jest sprawdzany warunek, czy pola przeznaczone na login i hasło zostały uzupełnione. Jeśli warunek jest nie spełniony przy użyciu metody *setText()* zostanie wyświetlony komunikat informujący o tym. W przypadku, gdy pola zostaną uzupełnione, zostajemy przeniesieni do funkcji *validateLogin()*, gdzie jest nawiązywane połączenie z bazą danych. Następnie wpisane dane zostają zweryfikowane z tymi, które są wprowadzone w bazie przy użyciu *getText()* służącą do pobierania danych z pola *TextField* . Jeśli dane są poprawne do zostajemy przeniesieni do funkcji *kontrolerMenu()*. W przeciwnym wypadku pojawia się informacja o nieprawidłowych danych.
Funkcja *kontrolerMenu()* tworzy nowe okno, którego graficzny projekt znajduje się w pliku *Menu.fxml*. Przy użyciu metod *initStyle(), setScene(), setTitle()* zostają dodane styl, rozmiar okna i tytuł. Zostaje także wywołana metoda *show(),* która służy do uruchomienia okna.
Funkcja *wyjscieButtionOnAction()* służy do zamknięcia okna. Wykorzystuje do tego metodę *close()*.

- **Klasa *KontrolerMenu***

Klasa służy obsługi głównego menu aplikacji. Tak jak w przypadku poprzedniej klasy, znajdują się odwołania do graficznego projektu okna, którego implementacja znajduje się w pliku *Menu.fxml*.
Funkcja *zamknijButtonOnAction()* służy do całkowitego zamknięcia aplikacji. Wykorzystuje ona metodę *close()* i *exit()*.
Funkcja *wylogujButtonOnAction()* ma podobne zachowanie do poprzedniej funkcji. Tyle że, zamyka ona jedynie menu główne przy użyciu metody *close()* i wylogowuje użytkownika przenosząc go do panelu logowania.
Kolejne funkcje znajdujące się w tej klasie mają bardzo podobne działanie. Zajmują się one tworzeniem nowych okien odpowiadających wybranej opcji. Każde z tych okien ma przypisaną swoją nazwę i osobny plik graficzny. Przy użyciu metod *initStyle(), setScene()*, *setTitle()* zostają dodane styl, rozmiar okna i tytuł. Zostaje także wywołana metoda show(), która służy do uruchomienia okna.
Funkcje, których nazwa kończy się na *ButtonOnAction(),* wywołują odpowiednie metody. Dla przykładu *AgdDButtonOnAction()* wywołuje *KontrolerAgdD()*. Ta funkcja zostaje użyta w pliku *Menu.fxml*. Jest ona przypisana do guzika, po naciśnięciu którego otwiera się okno dotyczące sprzętu o kategorii AGD duże.
Każda z funkcji zawiera blok *try..catch*. W przypadku napotkania błędu zostają wywołane metody *printStackTrace()* i *getCause()*, które informują co jest przyczyną nieprawidłowego działania.

- **Klasa *KontrolerAgdD***

Klasa służy do obsługi sprzętów o kategorii AGD duże. Zawiera w sobie kilka funkcji, które są odpowiedzialne za działanie sklepu. Każda z nich na początku nawiązuje połączenie z bazą danych.
Funkcja *initialize()* wywołuje dwie funkcje: wyszukiwanie() oraz odswiezania().
Funkcja odswiezanie() służy do wczytania danych z bazy z tabeli *agd_duze*. Po prawidłowym połączeniu wykonana zostaje instrukcja *select*, która wyświetla rekordy znajdujące się w tej tabeli. W pętli *while* dane zostają pobrane i wpisane do odpowiednich pól, przy użyciu metod *getInt()* i *getString()*. Następnie do utworzonego obiektu sklep1 zostają wpisane te dane i wyświetlone na ekranie.
Funkcja wyszukiwanie() służy do wyświetlenia wpisów, które zawierają określoną frazę. Zostają wyświetlone wszystkie wpisy, a następnie są filtrowane. Dane są sprawdzane pod względem zgodności nazwy sprzętu, nazwy producenta i ilości rat. Zostało tutaj wykorzystane wyrażenie lambda.
Funkcja *DodawanieProduktu()* zajmuje się dodawaniem nowych sprzętów do hurtowni. W bloku *try…catch*, przy użyciu *getText()* zostają pobrane wpisane przez użytkownika informacje. Następnie korzystając z funkcji *execute()* instrukcja zostaje uruchomiona. Jeśli wszystko pójdzie bezbłędnie i każde z pól zostanie prawidłowo uzupełnione to przy wykorzystaniu *JOptionPane.showMessageDialog()* zostanie wyświetlone okno dialogowe informujące, że wpis został dodany pomyślnie. W przeciwnym wypadku w oknie zobaczymy komendę o błędzie.

Funkcja *edytowanie()* zajmuje się aktualizowaniem istniejącego już wpisu. W bloku *try…catch* przy pomocy *getText()* zostają pobrane i wpisane dane sprzętu, który zostanie poddany edycji. Instrukcją w MySQL, która pozwala na taką czynność jest update. Przy użyciu *execute*() zostaje wykonana instrukcja, a efekt zostaje wyświetlony w oknie dialogowym w postaci komunikatu. W przypadku błędu także użytkownik zostanie poinformowany.
Funkcja *usuwanie()* służy to usunięcia wybranego wpisu z bazy danych. Tak jak w poprzednich funkcjach, zostaje wykonana odpowiednia instrukcja w języku MySQL służąca do usuwania rekordu. W bloku *try…catch* zostaje wykonana funkcja *execute()* uruchamiająca komendy w bazie. Użytkownik dostanie odpowiedni komunikat zależny od efektu działania usunięcia.
Kolejne funkcje takie jak *EdytujButtonOnActionEvent()*, *DodajButtonOnActionEvent()*, *UsunButtonOnActionEvent()* zajmują się uruchomieniem wyżej opisanych funkcji, a następnie zamykają okna przy użyciu metody *close()*. Są one przypisane do odpowiednich guzików, po naciśnięciu których wykonują się odpowiednie instrukcje.
Ostatnia funkcja znajdująca się w tej klasie to *getSelected()*. Umożliwia ona wybór konkretnego wpisu z listy i wyświetlenie odpowiednich informacji w wyznaczonych do tego miejscach.

- **Klasy:** *KontrolerAgdM*, *KontrolerKomputer*, *KontrolerKonsoleGry*, *KontrolerSmartfony*, *KontrolerTvAudio*, *KontrolerZamowienia*

Pozostałe klasy posiadają takie same funkcję i metody jak klasa *KontrolerAgdD*. Ich działanie polega na tym samym. Różnią się jedynie nazwą tabeli, w której znajdują się dane.

- **Klasa *walidacja***

Została stworzona aby sprawdzać czy podane wartości są zgodne z wymaganiami. Każda z użytych funkcji jest typu *boolean* i wartość takiego typu jest zwracana. Funkcje sprawdzają takie rzeczy jak: czy pole nie jest puste, czy podana wartość liczbowa nie jest równa bądź mniejsza od 0, czy długość numeru telefonu lub numeru zamówienia jest odpowiednia.

- **Testy jednostkowe**

Testy jednostkowe zostały stworzone po to, aby sprawdzić czy aplikacja w sytuacji, gdy zostanie podana błędna wartość zostanie poprawnie obsłużona. Zostało stworzone 38 testy i każdy z nich jest wykonywany poprawnie. Do sprawdzenia zostały wykorzystane metody assertFalse() i assertTrue().

5. Podział prac

Podział prac był równomierny. Każdy z członków wykonał taką samą część projektu.
Jarosław Kot -> ustawienie połączenia z bazą danych; wykonanie wyświetlania, dodawania i edycji;
Aleksandra Kwiatkowska -> wykonanie usuwania i wyszukiwania wpisów oraz sprawozdania wraz z dokumentacją JavaDoc;