P O L I T E C H N I K A Ś W I Ę T O K R Z Y S K A
Wydział Elektrotechniki, Automatyki i Informatyki

Projekt: Programowanie obiektowe 2	
Temat: Aplikacja (standalone) do prowadzenia oraz zarządzania sklepem komputerowym	
Mikołaj Pacierz Bartłomiej Pawlik	Rok studiów: **2**
	Grupa: **2ID13B**

Opis projektu:

Aplikacja służy do obsługi sklepu komputerowego.
Pozwala na zarządzaniu klientami, pracownikami, zamówieniami, dostawcami, produktami i usługami.

Użyte technologie:
- Java 21 + Maven
- Interfejs: JavaFX 21
- Baza danych: PostgreSQL
- Testy: JUnit 5, Mockito
- Dokumentacja: Javadoc
- Biblioteki: Lombok, SLF4J
- IDE: IntelliJ IDEA Ultimate

Dane do bazy danych / do logowania:
Nazwa użytkownika: admin
Hasło:

Sposób uruchomienia:
1. Utwórz nową bazę danych o nazwie ***sklepKomputerowy***
2. Zaimportuj dane do bazy z pliku ***sklepKomputerowy.sql***
2. Uruchom klasę ***App*** przez IDE.

Obsługa programu:
- Po uruchomieniu aplikacji należy zalogować się do systemu używając danych zapisanych powyżej.
- Pasek **Menu** na górze ekranu odpowiada za wybór odpowiedniej tabeli z danymi.
- Kliknięcie na przycisk **Logout** spowoduje wylogowanie i cofnięcie do ekranu logowania.
- Po kliknięciu na którąkolwiek z opcji wyświetli się tabela, oraz dwa przyciski.
- Dwukrotne kliknięcie na komórkę tabeli spowoduje edycję tekstu w komórce.
- Po kliknięciu w przycisk **Add**, wyświetli się okno na którym można wprowadzać dane do stworzenia nowego obiektu w bazie danych.
- Po kliknięciu w przycisk **Delete**, zostanie usunięty obiekt który jest podświetlony w tabeli.

Opis klas i metod:

App:
Klasa główna - służy do uruchomienia aplikacji.

DatabaseConnection:
Klasa odpowiadająca za połączenie użytkownika z bazą danych.
Zawiera metody ***connect()*** i ***disconnect()***;

SceneSwitch:
Klasa posiada jedną metodę - ***switchScene***, która odpowiada za zmianę sceny.

Modele:
- Client
- Delivery
- Order
- OrderItem
- Product
- Service
- Worker

Modele to klasy będące odpowiednikiem tabeli w bazie danych. Każdy model posiada ***gettery***, ***settery***, oraz metody ***of*** i ***validate***.

Metoda ***validate*** sprawdza poprawność danych (m.in. czy imie klienta nie posiada cyfr). Jest wywoływana w metodzie ***of***.
Metoda ***of*** służy do stworzenia nowego obiektu danej klasy.
Zwraca nowy obiekt.

Wyjątki:
- AddException
- DeleteException
- GetException
- UpdateException

Wszystkie wyjątki rozszerzają klasę RuntimeException.

Kontrolery:
- LoginController
Klasa która odpowiada za obsługę ekranu logowania.

- MenuBarController
Klasa która odpowiada za obsługę przycisków na pasku menu na górze ekranu.

- AppController
Klasa która odpowiada za obsługę ekranu który pojawia się po zalogowaniu.

- ClientsController
- DeliveryController
- OrderItemsController
- OrdersController
- ProductsController
- ServicesController
- WorkersController

Klasy które odpowiadają za obsługę tabeli i przycisków na poszczególnych ekranach.

Interfejsy:
- ClientRepository
- DeliveryRepository
- OrderItemRepository
- OrderRepository
- ProductRepository
- WorkerRepository

Każdy z interfejsów posiada statyczną metodę ***getDefaultImplementation()*** która zwraca implementację interfejsu.

Implementacje interfejsów:
- ClientRepositoryImplementation
- DeliveryRepositoryImplementation
- OrderItemRepositoryImplementation
- OrderRepositoryImplementation
- ProductRepositoryImplementation
- WorkerRepositoryImplementation

Klasy które są implementacją interfejsów. Każda z nich posiada metody: ***get***, ***add***, ***edit***, ***update***, ***delete*** i ***sortById***. Funkcje te odpowiadają za operowanie na obiektach z bazy i tym samym na tabeli z danymi.