System Zarządzania Promocjami
Opis oprogramowania

Antoni Blicharz
Szymon Dybał
Jakub Koszorz
Mikołaj Mroczek
Diego Ostoja-Kowalski

16 czerwca 2025

1 Wstęp

1.1 Cel dokumentu

Ten dokument opisuje architekturę oprogramowania Systemu Zarządzania Promocjami. Tłumaczy on zastosowany wzorzec architekturalny oraz to, które części oprogramowania realizują którą jego część.

1.2 Definicje

Panel – wyświetlana wersja intefejsu graficznego. Panele mogą zawierać inne panele oraz być przełączane na inne panele realizujące inne funkcjonalności.

Dla definicji innych terminów zobacz sekcję „Definicje” w dokumencie `Software Requirements Specification.pdf`.

1.3 Szczegóły implementacji

Szczegóły implementacji Systemu można znaleźć w folderze `src` oraz dokumencie `Diagram klas.png`, gdzie znajdują się informacje na temat kodu napisanego w języku Java, oraz w pliku `Diagram bazy danych.png`, gdzie jest przedstawiona struktura zdalnej bazy danych.

2 Wzorzec MVCS

Podczas gdy bardziej popularny wzorzec MVC (Model-View-Controller) ma jeden ustalony kształt, taka zgoda nie istnieje co do jego modyfikacji MVCS (Model-View-Controller-Service). Opierając się na idei takiej jak przedstawiona tutaj, w naszej architekturze cztery warstwy wzorca MVCS pełnią następujące funkcje:

- **Model** – warstwa zawierająca encje reprezentujące dane Systemu. Zaliczają się do tego wszystkie klasy `*Model.java` zdefiniowane w `src/main/java/app/model`, jak również struktura bazy danych. Klasy służą reprezentacji tych danych w interfejsie graficznym, a baza danych przechowuje końcowy stan operacji użytkownika.
- **Service** – warstwa służąca realizacji logiki biznesowej. Zaliczają się do niej klasy zdefiniowane w `src/main/java/app/service` poza `SceneManager.java`. Służy ona dwustronnej wymianie danych pomiędzy bazą danych, a interfejsem graficznym użytkownika, to w niej następuje sprawdzanie poprawności danych i wywoływanie powiadomień o powodzeniu lub niepowodzeniu operacji wykonywanych przez użytkowników.
- **Controller** – warstwa służąca transformacji danych przekazanych z warstwy Service na potrzeby interfejsu graficznego oraz przekazywaniu danych wprowadzonych przez użytkownika do warstwy Service. Należą do niej klasy zdefiniowane w `src/main/java/app/controllers`.
- **View** – warstwa odpowiadająca za faktyczne wyświetlanie interfejsu graficznego, komunikująca się z warstwą `Controller`. Należą do niej panele `.fxml` zawarte w

`src/main/java/app/resources`, jak również klasy `PanelList.java` i `SceneManager.java`.

Większość plików oprogramowania w prosty sposób realizuje tę architekturę przez powiązanie ze sobą odpowiadających sobie plików `*Service.java`, `*Controller.java` oraz `*_panel.fxml`, gdzie pierwsze dwa pliki są nazwane w CamelCase, a ostatni w snake_case. Pliki `*Model.java` są z kolei używane uniwersalnie, w zależności od funkcjonalności wymaganych w konkretnych panelach. Pozostałe pliki, które nie zaliczają się do tego schematu, są opisane dokładniej w dalszej sekcji.

3 Opis szczegółowy

3.1 Baza danych

Klasa `MySQLConnection.java` jest odpowiedzialna za nawiązywanie i kończenie połączenia z bazą danych MySQL. Klasa `RepositorySQL.java` jest odpowiedzialna za wykonywanie odpowiednich poleceń SQL w celu wyświetlenia obecnych danych lub modyfikacji bazy danych.

3.2 Połączenie z systemem kasowym

Klasy odpowiedzialne za realizację tego połączenia oraz symulację takowego połączenia na potrzeby wykazania funkcjonalności znajdują się w `src/main/java/app/service/branch_panel/ClientSimulation`.

3.3 Funkcjonalności pracownika punktu sprzedaży

Część klas warstwy Service obsługująca pracownika punktu sprzedaży została połączona, gdyż następowała zbędna duplikacja kodu.

3.4 Tworzenie raportów

Istnieją dodatkowe klasy służące tworzeniu raportów `.pdf`, zawarte w `src/main/java/app/service/business_panel`. Służą one używaniu tego samego kodu do generacji podobnych graficznie raportów zamiast powielania logiki w różnych panelach odpowiadających wyborowi różnych raportów.

3.5 Sprawdzanie poprawności danych i wyświetlanie powiadomień

Klasy `LoginValidation.java` i `TypeValidation.java` służą sprawdzaniu poprawności wprowadzanych danych, a klasy `Alerts.java` i `Dialogs.java` zawierają kilkukrotnie używane typy okienek z powiadomieniami oraz okienek dialogowych.

3.6 Wyświetlanie interfejsu graficznego

Klasa `Main.java` jest odpowiedzialna za rozpoczęcie działania interfejsu graficznego. Klasa `SceneManager.java` służy ładowaniu, usuwaniu i przełączaniu między panelami,

wykorzystując do tego mapę ich nazw w klasie `PanelList.java`. Klasy zawarte w `src/main/java/app/controllers/shared` oraz powiązane z nimi pliki `.fxml` służą wyświetlaniu bocznego panelu użytkownika i zapewnianiu miejsca na podpanele realizujące wymagania funkcjonalne.