POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

PROJEKT Z BAZ DANYCH

System obsługi wypożyczalni kaset oparty o relacyjną bazę danych

AUTOR:

Karol Wojtachnia
Indeks: 243702

PROWADZĄCY ZAJĘCIA:

Dr inż. Robert Wójcik, K30W04D03

OCENA PRACY:

Wrocław, 2021

Spis treści

Spis tabel 4
Spis rysunków 5
1. Wstęp 6
1.1 Cel projektu 6
1.2 Zakres i układ projektu 6
2. Analiza wymagań 7
2.1 Opis działania i funkcje systemu 7
2.2 Schemat systemu 7
2.3 Założenia architektoniczne przyjęte podczas realizacji systemu 7
2.4 Wykorzystywane technologie, narzędzia projektowania oraz implementacji systemu 7
2.5 Analiza wymagań funkcjonalnych 8
2.5.1 Diagram przypadków użycia 8
2.5.2 Definicje aktorów 9
2.5.3 Scenariusze przypadków użycia 10
2.5.4 Diagram wymagań 14
3. Projekt systemu 15
3.1 Projekt bazy danych 15
3.1.1 Uproszczony model konceptualny 15
3.1.2 Model logiczny i normalizacja 16
3.1.3 Model fizyczny i ograniczenia integralności danych 17
3.1.4 Inne elementy bazy danych 18
3.1.5 Projekt zabezpieczeń na poziomie bazy danych 18
3.2 Projekt aplikacji 19
3.2.1 Interfejs graficzny i struktura menu 19
3.2.2 Metoda podłączenia do bazy danych – integracja z bazą danych 21
3.2.3 Projekt zabezpieczeń na poziomie aplikacji 22
4. Implementacja systemu 23
4.1 Realizacja bazy danych 23
4.1.1 Tworzenie tabel i widoków oraz definiowanie ograniczeń 23
4.1.2 Implementacja trigger-ów 24
4.1.3 Implementacja mechanizmów zabezpieczeń 25
4.2 Realizacja elementów aplikacji 25
4.2.1 Przykład operacji INSERT 26
4.2.2 Przykład operacji DELETE 26
4.2.3 Przykład operacji SELECT 26
5. Testowanie systemu 27
5.1 Test operacji INSERT 27
5.2 Test operacji DELETE 28
5.3 Test operacji UPDATE 29
5.4 Test operacji SELECT 30

5.5 Test walidacji danych w aplikacji 31
5.6 Wnioski z testów 33
6. Podsumowanie 34
Literatura 35

Spis tabel

Tabela 1 Definicje aktorów 9

Spis rysunków

Rysunek 1Schemat projektowanego systemu 7
Rysunek 2 Diagram przypadków użycia 8
Rysunek 3 Diagram wymagań 14
Rysunek 4 Konceptualny model bazy – diagram 15
Rysunek 5 Logiczny model bazy - diagram 16
Rysunek 6 Fizyczny model bazy - diagram 17
Rysunek 7 Wygląd okna logowania 19
Rysunek 8 Wygląd głównego menu 19
Rysunek 9 Wygląd okna "KLIENCI" 20
Rysunek 10 Wygląd okna "KASETY" 20
Rysunek 11 Wygląd okna "RACHUNKI" 21
Rysunek 12 Przykład połączenia z bazą danych 21
Rysunek 13 Przykład wyrażeń regularnych użytych do walidacji danych 22
Rysunek 14 Obiekty typu java.util.regex.Matcher 22
Rysunek 15 Inicjalizacja Matcher-ów 22
Rysunek 16 Przykład walidacji numeru telefonu 22
Rysunek 17 Tabla KLIENCI przed testem INSERT 27
Rysunek 18 Dane do testu INSERT 27
Rysunek 19 Tabela KLIENCI po teście INSERT 27
Rysunek 20 Tabela KLIENCI przed testem DELETE 28
Rysunek 21 Dane testowe do testu DELETE 28
Rysunek 22 Tabela KLIENCI po teście DELETE 28
Rysunek 23 Tablela KASETY przed testem UPDATE 29
Rysunek 24 Dane testowe do testu UPDATE 29
Rysunek 25 Tabela KASETY po teście UPDATE 30
Rysunek 26 Tabela KLIENCI dla testu SELECT 30
Rysunek 27 Tabela RACHUNKI dla testu SELECT 31
Rysunek 28 Tabela w oknie aplikacji dla testu SELECT 31
Rysunek 29 Test walidacji - niepoprawne imię 31
Rysunek 30 Test walidacji - niepoprawne nazwisko 32
Rysunek 31 Test walidacji - niepoprwny numer telefony 32
Rysunek 32 Test walidacji - niepoprawny adres email 32

1. Wstęp

1.1 Cel projektu

Celem projektu było zaprojektowanie oraz implementacja bazy danych oraz prostego interfejsu użytkownika przeznaczonych do obsługi i zarządzania wypożyczalnią kaset z poziomu aplikacji okienkowej.

1.2 Zakres i układ projektu

Rozdział drugi zawiera analizę wymagań funkcjonalnych oraz niefunkcjonalnych. Można w nim znaleźć informacje odnośnie wstępnych założeń co do budowy i działania projektowanego systemu. Diagramy i opisy słowne pomagają zobrazować przypadki użycia, wymagania funkcjonalne oraz zdefiniować aktorów w systemie.

W rozdziale trzecim mieści się projekt bazy danych i aplikacji. Baza danych została zobrazowana jako trzy diagramy encji oraz opis słowny pozostałych jej elementów. Projekt aplikacji pokazuje budowę interfejsu graficznego, sposób połączenia z bazą danych i zaimplementowane zabezpieczenia działające na jej poziomie.

Kolejny rozdział opisuje sposób implementacji bazy danych. Zawiera kod w języku SQL, za pomocą którego baza oraz wszystkie jej pozostałe elementy zostały stworzone. Na końcu przedstawiono implementację operacji bazodanowych CRUD w aplikacji.

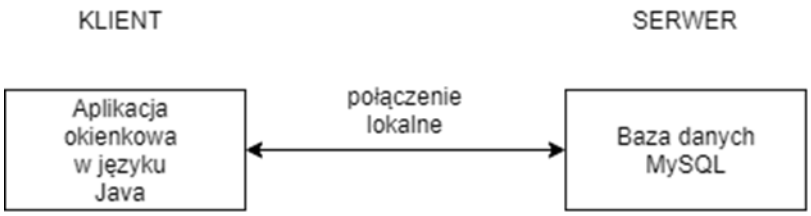
W ostatnim rozdziale skupiono się na testowaniu działania całego systemu. Zostały sprawdzone wszystkie operacje CRUD oraz sposób walidacji danych wprowadzanych przez użytkownika do aplikacji.

2. Analiza wymagań

2.1 Opis działania i funkcje systemu

System będzie umożliwiać zarządzanie wypożyczalnią kaset w oparciu o relacyjną bazę danych (tabele opisujące dane kaset i klientów). Dostęp do bazy danych będzie możliwy z poziomu aplikacji okienkowej (klienta). Aplikacja będzie umożliwiała wykonywanie różnych operacji na bazie danych (np. dodawanie nowych kaset, wyszukiwanie kaset, tworzenie konta osoby wypożyczającej, wypożyczanie kaset).

2.2 Schemat systemu



Rysunek 1Schemat projektowanego systemu

2.3 Założenia architektoniczne przyjęte podczas realizacji systemu

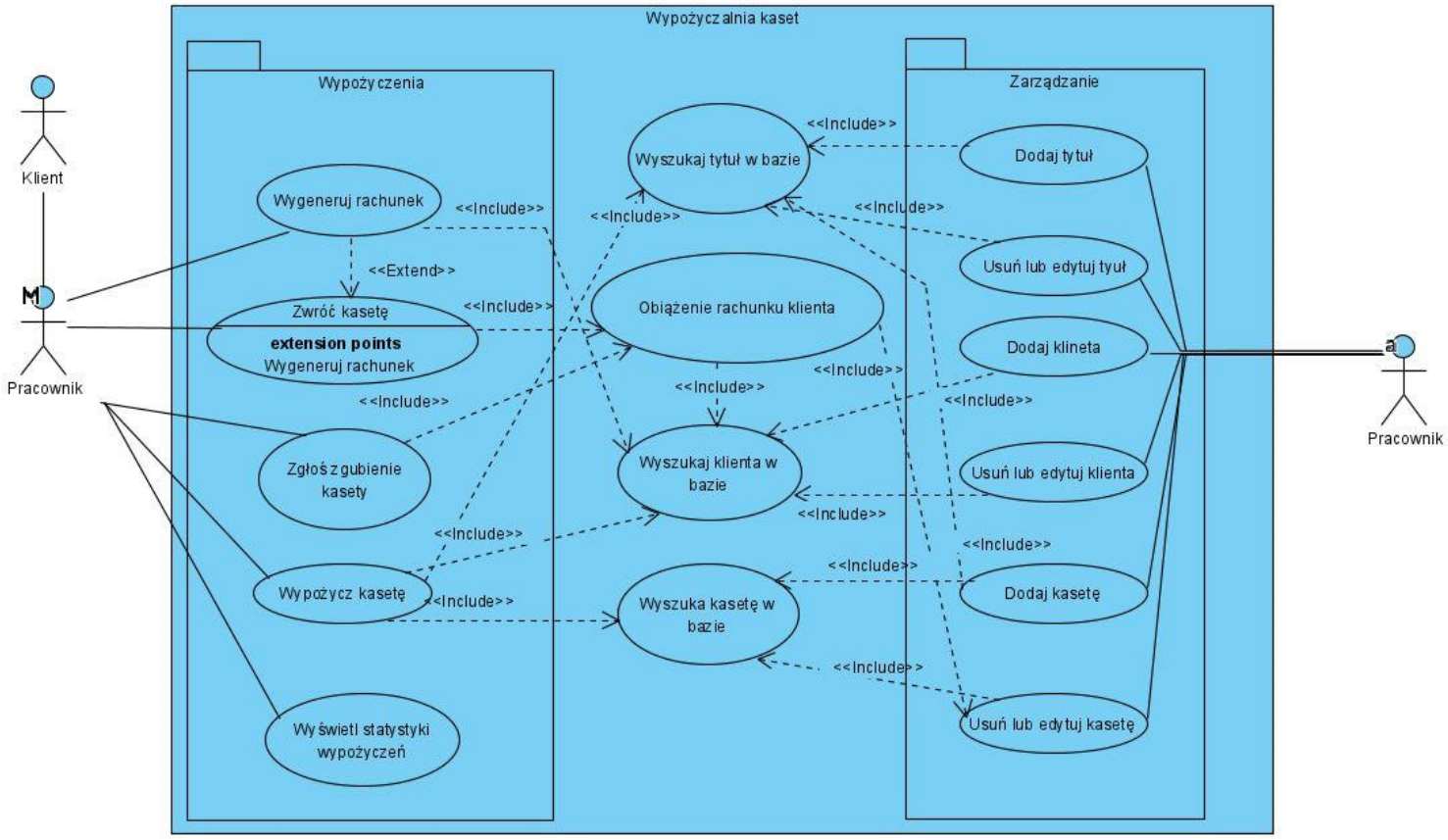
Projektowany system będzie się składał z 2 warstw: aplikacji okienkowej(klienta) oraz serwera bazy danych [1], [2], [3]. W zastosowanej architekturze serwer ma za zadanie przechowywanie danych oraz przetwarzanie zapytań wysyłanych z aplikacji. Klient w postaci aplikacji okienkowej służy do przedstawienia graficznego informacji wysyłanych z serwera oraz umożliwia użytkownikowi wysyłanie zapytań do serwera

2.4 Wykorzystywane technologie, narzędzia projektowania oraz implementacji systemu

Baza danych będzie obsługiwana przez serwer MySQL [6], [7]. Interfejs użytkownika zostanie zrealizowany w postaci aplikacji okienkowej napisanej w języku Java [4], [5]. Model systemu zostanie zaprezentowany z wykorzystaniem języka modelowania UML [1] przy pomocy programu Visual Paradigm [8].

2.5 Analiza wymagań funkcjonalnych

2.5.1 Diagram przypadków użycia



Rysunek 2 Diagram przypadków użycia

2.5.2 Definicje aktorów

Tabela 1 Definicje aktorów

AKTOR	OPIS	PRZYPADKI UŻYCIA
Klient	Klient jest aktorem biernym w systemie. Wszystkie akcje są wykonywane przez pracownika wypożyczalni.	
Pracownik	Sprzedawca wykonuje wszystkie czynności w systemie np. dodawanie kaset, zarządzanie wypożyczeniami, rozliczanie klienta.	1. PU Wygeneruj rachunek powiązane przez <<include>> z PU Wyszukaj klienta w bazie 2. PU Zwróć kasetę powiązane przez <<extend>> z PU Wygeneruj rachunek i przez <<include>> z PU Obciążenie rachunku klienta 3. PU Zgłoś zgubienie kasety powiązane przez <<include>> z PU Obciążenie rachunku klienta 4. PU Wypożycz kasetę powiązane przez <<include>> z PU Wyszukaj klienta w bazie oraz przez <<include>> PU Wyszukaj kasetę w bazie 5. PU Dodaj tytuł powiązane przez <<include>> z PU Wyszukaj tytuł w bazie 6. PU Usuń lub edytuj tytuł powiązane przez <<include>> z PU Wyszukaj tytuł w bazie 7. PU Dodaj klienta powiązane przez <<include>> z PU Wyszukaj klienta w bazie 8. PU Usuń lub edytuj klienta powiązane przez <<include>> z PU Wyszukaj klienta w bazie 9. PU Dodaj kasetę powiązene przez <<include>> z PU Wyszukaj kasetę w bazie 10. 9. PU Usuń lub edytuj kasetę powiązene przez <<include>> z PU Wyszukaj kasetę w bazie

2.5.3 Scenariusze przypadków użycia

PU Dodaj tytuł

OPIS
CEL: Dodanie nowego tytułu.
Warunki Wstępne: Wywołuje go pracownik za pomocą aplikacji.
Warunki Końcowe: Podanie tytułu o następujących atrybutach obowiązkowych: tytuł, reżyser, studio filmowe oraz cena.
PRZEBIEG:
1. Wywołuje się PU Wyszukaj tytuł w bazie, gdzie sprawdza się czy istnieje już tytuł o takich samych atrybutach: tytuł, reżyser, studio.
2. Jeśli znaleziono, zwracana jest informacja o istnieniu takiego tytułu w bazie danych, w przeciwnym wypadku, generuje się identyfikator i tytuł jest dodawany do bazy.

PU Wyszukaj tytuł w bazie

OPIS
CEL: Szukanie tytułu w bazie.
WS: Wywołują go PU Dodaj tytuł, PU Usuń tytuł i PU Edytuj tytuł.
WK: Podanie identyfikatora tytułu lub parametrów: tytuł, reżyser, studio filmowe.
PRZEBIEG:
1. Sprawdza się czy w bazie istnieje tytuł o podanych parametrach.
2. Jeśli znaleziono, zwracana jest informacja o istnieniu takiego tytułu w bazie danych, w przeciwnym wypadku, zwracana jest informacja o jego braku.

PU Usuń lub edytuj tytuł

OPIS
CEL: Usunięcie lub edycja istniejącego tytułu.
WS: Wywołuje go pracownik za pomocą aplikacji.
WK: Podanie identyfikatora tytułu lub parametrów: tytuł, reżyser, studio filmowe.
PRZEBIEG:
1. Wywołuje się PU Wyszukaj tytuł w bazie, gdzie sprawdza się czy istnieje już tytuł o takich samych atrybutach.
2. Jeśli znaleziono, pracownik wybiera, czy chce go edytować, czy usunąć, w przeciwnym wypadku, zwracana jest informacja o jego braku.
3. Jeżeli wybrano opcję usuń, pracownik jest pytany o potwierdzenie, a następnie dane są usuwane z bazy. Jeżeli wybrano opcję edycji, pracownik przystępuje do edytowania danych w bazie.

PU Dodaj klienta

OPIS
CEL: Dodanie nowego klienta.
WS: Wywołuje go pracownik za pomocą aplikacji.
WK: Podanie informacji o kliencie o następujących atrybutach obowiązkowych: imię, nazwisko, data urodzenia, adres e-mail.
PRZEBIEG:
1. Wywołuje się PU Wyszukaj klienta w bazie, gdzie sprawdza się czy istnieje już klient o podanych atrybutach.
2. Jeśli znaleziono, zwracana jest informacja o istnieniu takiego klienta w bazie danych, w przeciwnym wypadku, generuje się identyfikator i klient jest dodawany do bazy.

PU Wyszukaj klienta w bazie

OPIS

CEL: Szukanie klienta w bazie.

WS: Wywołują go PU Dodaj klienta, PU Usuń klienta i PU Edytuj klienta.

WK: Podanie danych klienta o następujących atrybutach obowiązkowych: imię, nazwisko, data urodzenia, adres e-mail lub identyfikator.

PRZEBIEG:

1. Sprawdza się czy w bazie istnieje klient o takich samych atrybutach
2. Jeśli znaleziono, zwracana jest informacja o istnieniu takiego klienta w bazie danych, w przeciwnym wypadku, zwracana jest informacja o jego braku.

PU Usuń lub edytuj klienta

OPIS

CEL: Usunięcie lub edycja danych istniejącego klienta.

WS: Wywołuje go pracownik za pomocą aplikacji.

WK: Podanie danych klienta o następujących atrybutach obowiązkowych: imię, nazwisko, data urodzenia, adres e-mail lub identyfikator.

PRZEBIEG:

1. Wywołuje się PU Wyszukaj klienta w bazie, gdzie sprawdza się czy istnieje już klient o takich samych atrybutach.
2. Jeśli znaleziono, pracownik wybiera, czy chce go edytować, czy usuną , w przeciwnym wypadku, zwracana jest informacja o jego braku.
3. Jeżeli wybrano opcję usuń, pracownik jest pytany o potwierdzenie, a następnie dane są usuwane z bazy. Jeżeli wybrano opcję edycji, pracownik przystępuje do edytowania danych w bazie.

PU Dodaj kasetę

OPIS

CEL: Dodanie nowej kasety.

WS: Wywołuje go pracownik za pomocą aplikacji.

WK: Podanie numeru identyfikacyjnego kasety i informacji o filmie jaki jest na niej nagrany – tytułu, reżysera i studia filmowego.

PRZEBIEG:

1. Wywołuje się PU Wyszukaj kasetę w bazie, gdzie sprawdza się czy istnieje już kaseta o podanym numerze identyfikacyjnym.
2. Jeśli znaleziono, zwracana jest informacja o istnieniu takiej kasety w bazie danych i proces kończy się na tym punkcie.
3. Jeśli nie znaleziono, wywołuje się PU Wyszukaj tytuł w bazie, gdzie sprawdza się czy istnieje już tytuł o podanych parametrach. Jeśli istnieje, zwiększana jest informacja o liczbie egzemplarzy filmu o jeden, a identyfikator kasety jest dodawany do listy identyfikatorów, która jest atrybutem tytułu. W przeciwnym wypadku pojawia się komunikat o braku tytułu w bazie.
4. Kaseta jest dodawana do bazy. Przydzielana jest jej lokalizacja i informacja o stanie niewypożyczonym.

PU Wyszukaj kasetę w bazie

OPIS

CEL: Szukanie kasety w bazie.

WS: Wywołują go PU Dodaj kasetę, PU Usuń kasetę i PU Edytuj kasetę.

WK: Podanie identyfikatora kasety.

PRZEBIEG:

1. Sprawdza się czy w bazie istnieje kaseta o podanym identyfikatorze.
2. Jeśli znaleziono, zwracana jest informacja o istnieniu takiej kasety w bazie danych, w przeciwnym wypadku, zwracana jest informacja o jej braku.

PU Usuń lub edytuj kasetę

OPIS
CEL: Usunięcie lub edycja istniejącej kasety.
WS: Wywołuje go pracownik za pomocą aplikacji.
WK: Podanie identyfikatora kasety.
PRZEBIEG:
1. Wywołuje się PU Wyszukaj kasetę w bazie, gdzie sprawdza się czy istnieje już kaseta o takim numerze identyfikacyjnym.
2. Jeśli nie znaleziono, zwracana jest informacja o braku takiej kasety w bazie danych i proces kończy się na tym punkcie, w przeciwnym wypadku pracownik wybiera, czy chce edytować, czy usunąć dane.
3. Jeśli pracownik wybierze opcję usunięcia, wywołuje się PU Wyszukaj tytuł w bazie, gdzie sprawdza się czy istnieje już tytuł odpowiadający kasecie. Jeśli tak, zmniejszana jest informacja o liczbie egzemplarzy filmu o jeden, a identyfikator kasety jest usuwany z listy identyfikatorów, która jest atrybutem tytułu. W przeciwnym wypadku pojawia się komunikat o błędzie. Jeżeli wybrana została opcja edycji danych, pracownik przystępuje do edycji.

PU Wypożycz kasetę

OPIS
CEL: Wypożyczenie kasety klientowi przez pracownika i zapisanie tej informacji w systemie.
WS: Wywołuje go pracownik za pomocą aplikacji.
WK: Podanie informacji o filmie jaki ma zostać wypożyczony – tytułu, reżysera i studia filmowego.
PRZEBIEG:
1. Wywołuje się PU Wyszukaj tytuł w bazie, gdzie sprawdza się czy istnieje taki tytuł.
2. Jeśli nie znaleziono, proces kończy się w tym miejscu.
3. Jeśli znaleziono, sprawdza się dostępność kaset o identyfikatorach zawartych w liście identyfikatorów za pomocą wywoływania PU Wyszukaj kasetę w bazie, dopóki nie znajdzie się dostępnej kasety.
4. Pracownik otrzymuje informację o lokalizacji kasety i uaktualniane są dane o jej stanie wypożyczenia za pomocą PU Edytuj kasetę.

PU Obciążenie rachunku klienta

OPIS
CEL: Obciążenie rachunku klienta.
WS: Wywołują go PU Zgłoś zgubienie kasety i PU Zwróć kasetę.
WK: Podanie informacji o kwocie jaką należy dodać do rachunku klienta i identyfikatora klienta.
PRZEBIEG:
1. Wywołanie PU Wyszukaj klienta w bazie klientów. Jeśli nie znaleziono, wyświetlony zostaje komunikat o nieprawidłowym identyfikatorze klienta.
2. Dodanie do rachunku klienta kwoty jaką ma on uiścić.

PU Zgłoś zgubienie kasety

OPIS
CEL: Obciążenie rachunku klienta opłatą za zgubioną kasetę.
WS: Wywołuje go pracownik za pomocą aplikacji.
WK: Podanie informacji o kwocie jaką należy dodać do rachunku klienta i dane zgubionej kasety.
PRZEBIEG:
1. Wywołanie PU Wyszukaj klienta w bazie klientów. Jeśli nie znaleziono, wyświetlony zostaje komunikat o nieprawidłowym identyfikatorze klienta.
2. Dodanie do rachunku klienta kwoty jaką ma on uiścić.

3. Wywołanie PU Usuń kasetę w celu usunięcia zgubionego egzemplarza.

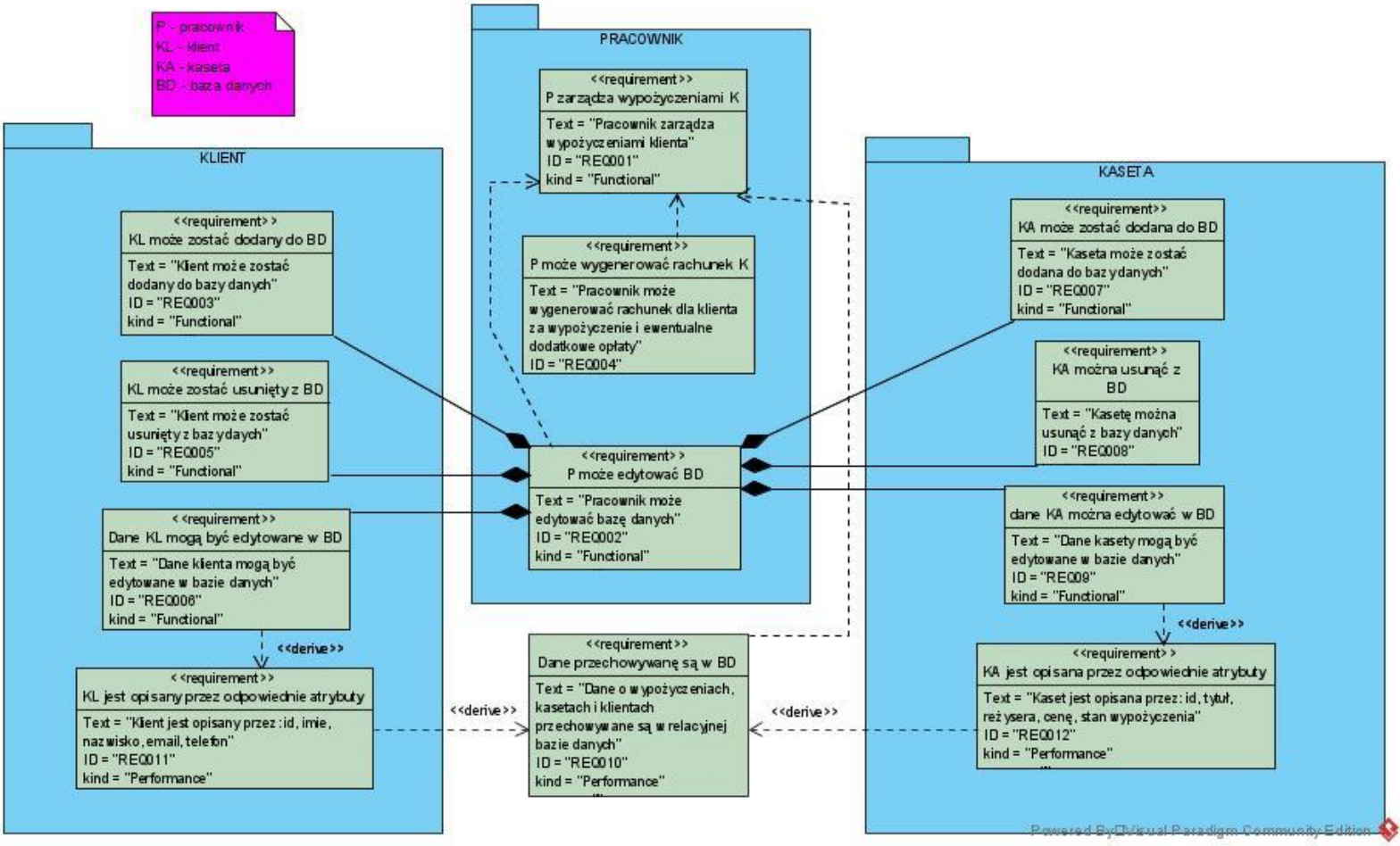
PU Zwróć kasetę

OPIS
CEL: Obciążenie rachunku klienta opłatą za wypożyczoną kasetę.
WS: Wywołuje go pracownik za pomocą aplikacji.
WK: Podanie informacji o kwocie jaką należy dodać do rachunku klienta i dane kasety.
PRZEBIEG:
1. Wywołanie PU Wyszukaj klienta w bazie klientów. Jeśli nie znaleziono, wyświetlony zostaje komunikat o nieprawidłowym identyfikatorze klienta.
2. Dodanie do rachunku klienta kwoty jaką ma on uiścić.
3. Wywołanie PU Edytuj kasetę w celu zmiany informacji o stanie wypożyczenia egzemplarza.

PU Wygeneruj rachunek

OPIS
Cel: Przekazanie klientowi informacji o tym, ile musi zapłacić za swoje wypożyczenia
WS: Wywoływany jest przez pracownika w aplikacji
WK: Podanie danych klienta, którego rachunek ma zostać wyświetlony
PRZEBIEG:
1. Wywołanie PU Wyszukaj klienta w bazie. Jeśli nie znaleziono, wyświetlony zostaje komunikat o nieprawidłowym identyfikatorze klienta.
2. Wyświetlenie na ekranie informacji o wysokości kwoty jaką klient ma uiścić.

2.5.4 Diagram wymagań

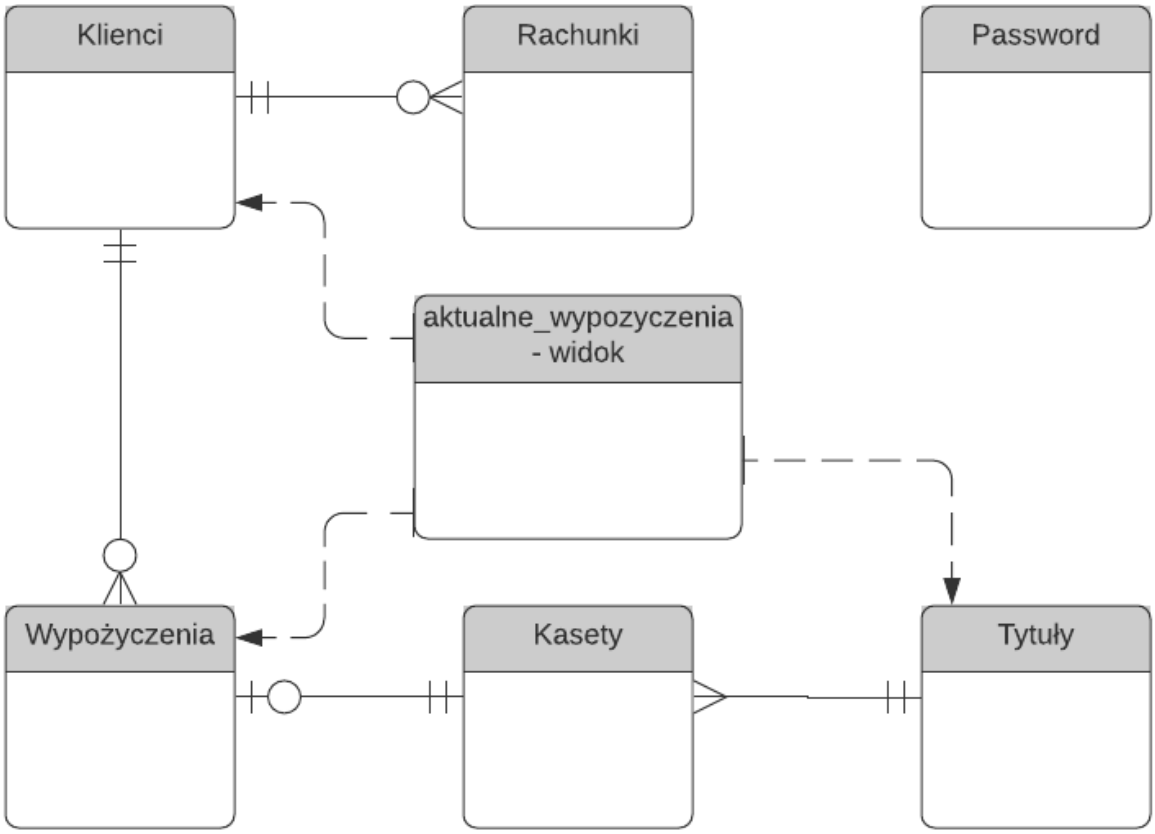


Rysunek 3 Diagram wymagań

3. Projekt systemu

3.1 Projekt bazy danych

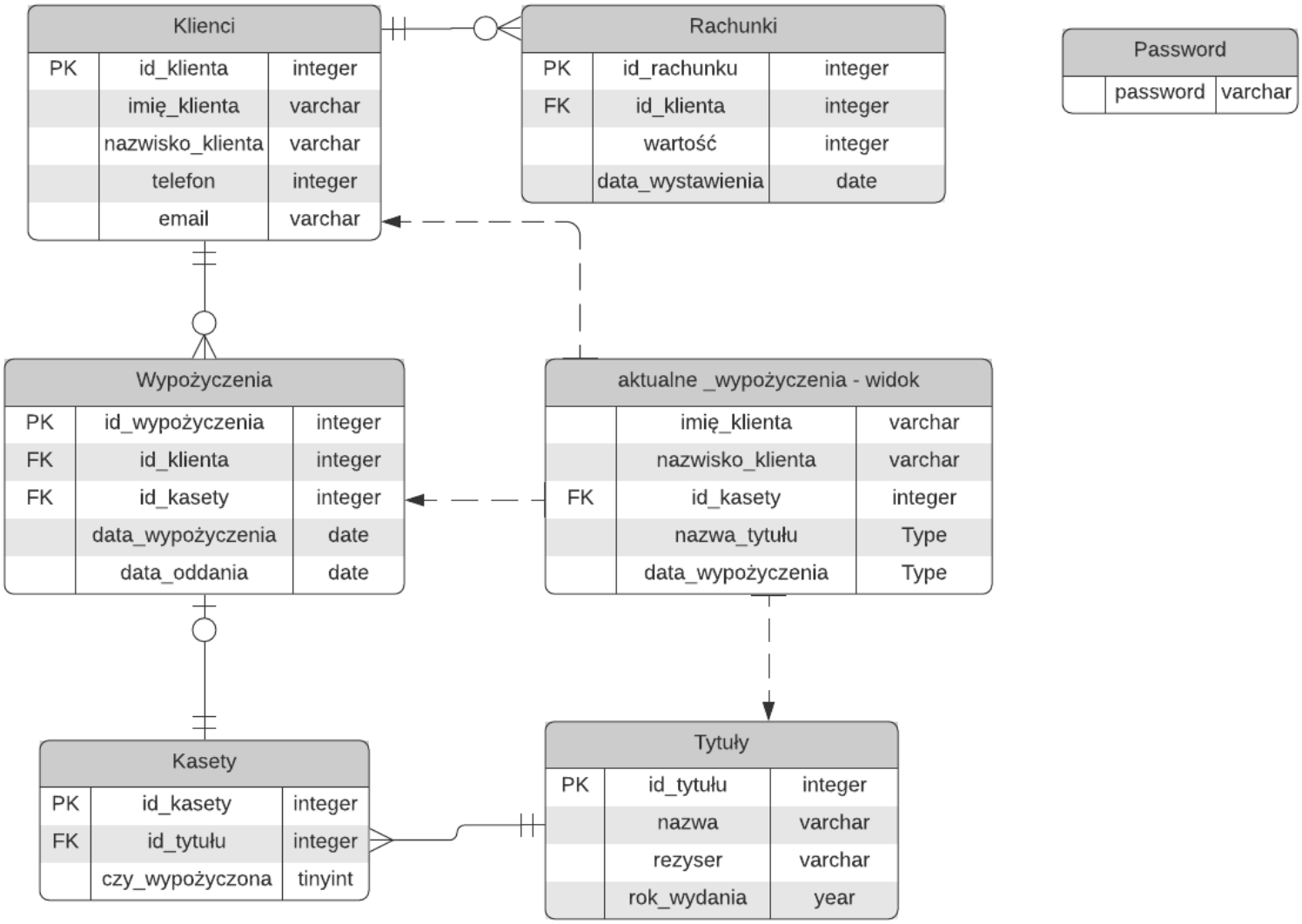
3.1.1 Uproszczony model konceptualny



Rysunek 4 Konceptualny model bazy – diagram

Model konceptualny bazy danych pokazuje jedynie związki encji w bazie danych. Baza składa się z sześciu tabel oraz jednego widoku.

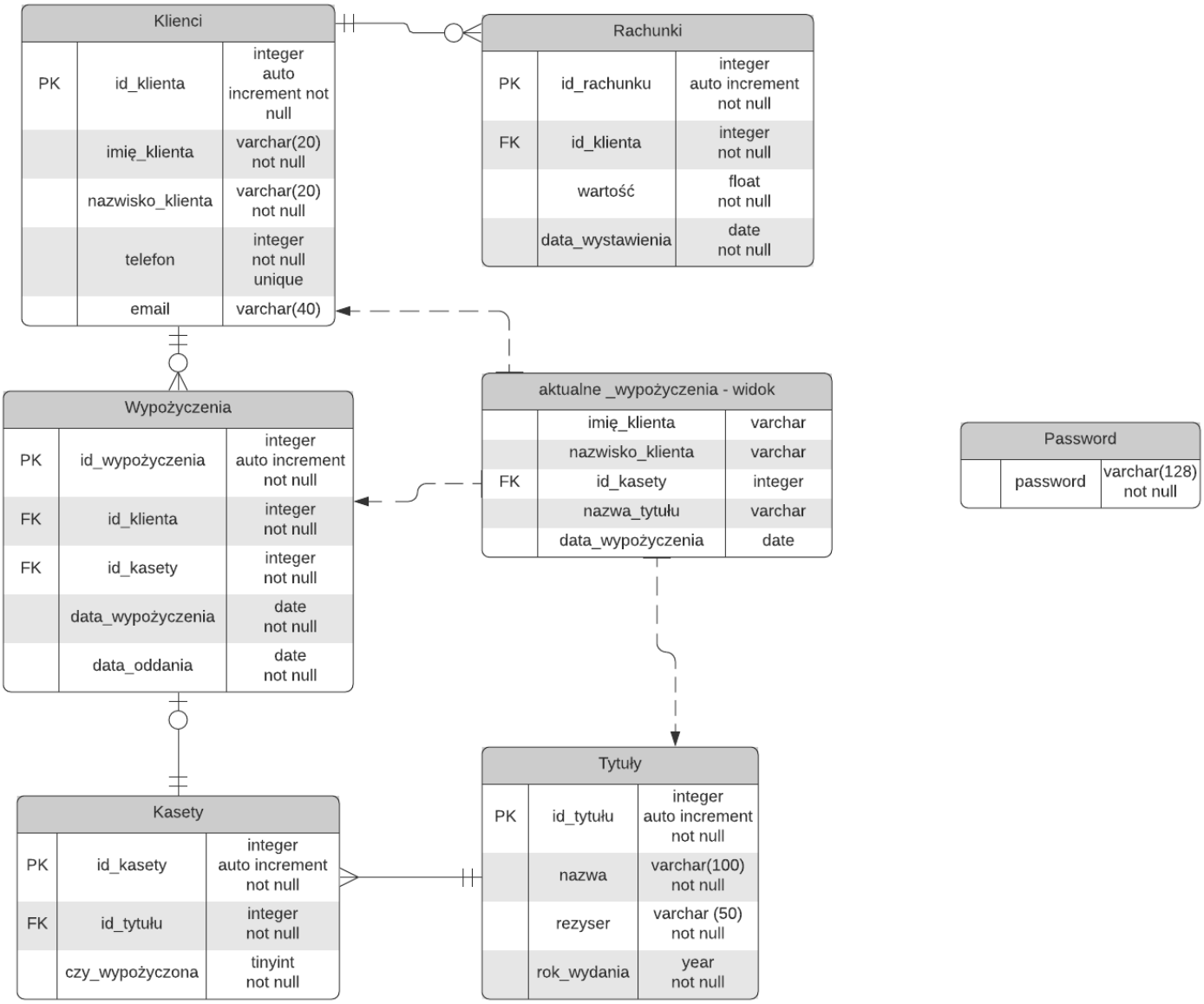
3.1.2 Model logiczny i normalizacja



Rysunek 5 Logiczny model bazy - diagram

Diagram rozwija schemat z poprzedniej strony i prezentuje logiczny model bazy. Przekazywane są informacje o tym, jakie kolumny zawiera każda tabela. Oprócz tego, zapisano format danych zapisanych w każdej kolumnie, oraz zaznaczono klucze prywatne i obce.

3.1.3 Model fizyczny i ograniczenia integralności danych



Rysunek 6 Fizyczny model bazy - diagram

Ostatni diagram encji jest najbardziej szczegółowy. Model logiczny został poszerzony o ograniczenia odnośnie danych zawartych w każdej kolumnie.

3.1.4 Inne elementy bazy danych

Baza danych zawiera także trzy triggery:

- setDates – przed dodaniem rekordu do tabeli ‘Wypożyczenia’ ustawia pola ‘data_wypozyczenia’ oraz ‘data_oddania’ dla wstawianego rekordu jako aktualną datę systemową oraz datę o tydzień późniejszą
- setKasetaWypozyczona – po dodaniu rekordu do tabeli ‘Wypożyczenia’, ustawia w tabeli ‘Kasety’ pole ‘czy_wypożyczona’ dla rekordu o ‘id_kasety’ równym ‘new.id_kasety’ na wartość ‘1’
- setKasetaNiewypozyczona – po usunięciu rekordu z tabeli ‘Wypożyczenia’, ustawia w tabeli ‘Kasety’ pole ‘czy_wypożyczona’ dla rekordu o ‘id_kasety’ równym ‘old.id_kasety’ na wartość ‘0’
- utwórzRachunek – po usunięciu rekordu z tabeli ‘Wypożyczenia’, wstawia nowy rekord do tabeli ‘Rachunki’. Trigger automatycznie wylicza wartość rachunku oraz sprawdza, czy termin oddania został przekroczony. Tym samym tworzenie rachunków w systemie zostało zautomatyzowane.

3.1.5 Projekt zabezpieczeń na poziomie bazy danych

W bazie danych przechowywane jest hasło dostępu do aplikacji. Tabela ‘Password’ zawiera jedną kolumnę i jeden rekord. Służy ona do przechowywania hasła. Hasło jest zapisane w postaci wyniku działania funkcji zaimplementowanej w bazie MySql: SHA2() w wariancie 512 bit-owym. Pozwala ona na hashowanie hasła do postaci 128 znakowego ciągu znaków zgodnie z algorytmem SHA2.

3.2 Projekt aplikacji

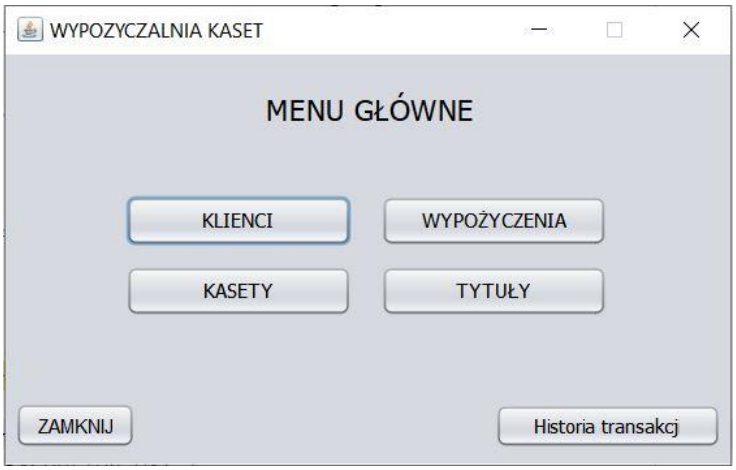
3.2.1 Interfejs graficzny i struktura menu

LOGOWANIE

Rysunek 7 Wygląd okna logowania

Strona logowania jest pierwszym oknem, które użytkownik widzi po uruchomieniu aplikacji. Należy wpisać hasło dostępu, a następnie wcisnąć przycisk „LOGIN”, aby przejść do głównego menu. W razie błędu zostanie wyświetlone odpowiednie okno ze szczegółami błędów, które wystąpiły. Zostanie to pokazane podczas testów aplikacji

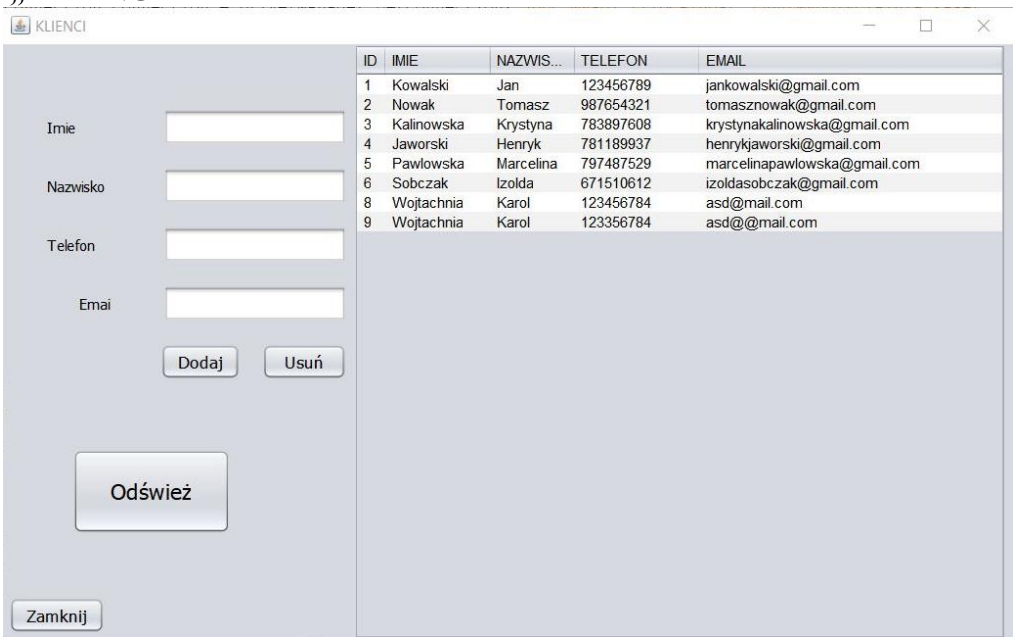
MENU GŁÓWNE



Rysunek 8 Wygląd głównego menu

Menu główne pozwala przejść do poszczególnych funkcjonalności aplikacji. W aplikacji nie została zaimplementowana funkcjonalność przycisku „Tytuły”. Po jego wciśnięciu nie jest wykonywana żadna akcja.

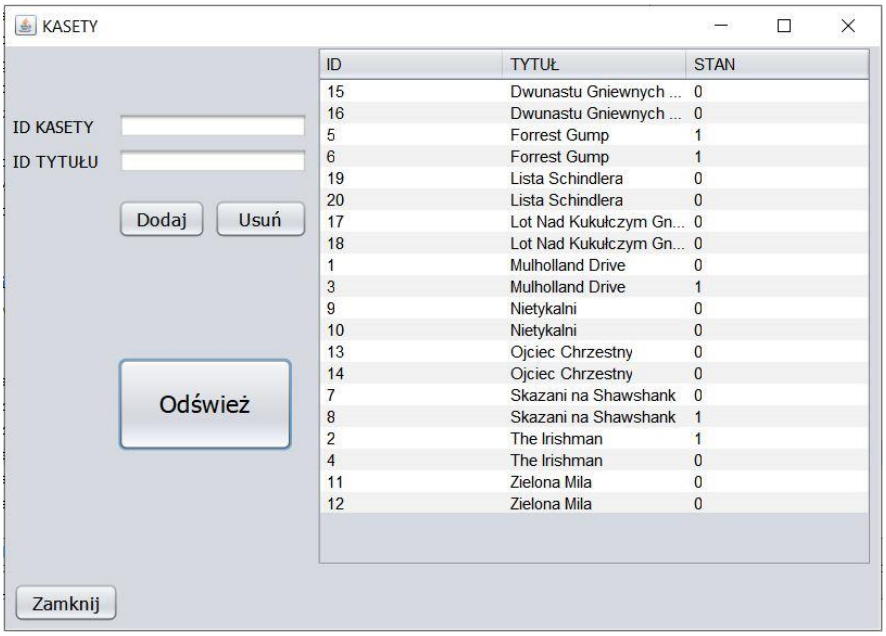
OKNO „KLIENCI”



Rysunek 9 Wygląd okna "KLIENCI"

Po wciśnięciu przycisku „Odśwież” pobierane są informacje z bazy i wyświetleni zostają wszyscy klienci znajdujący się w bazie danych. Okno pozwala także na dodawanie i usuwanie klientów.

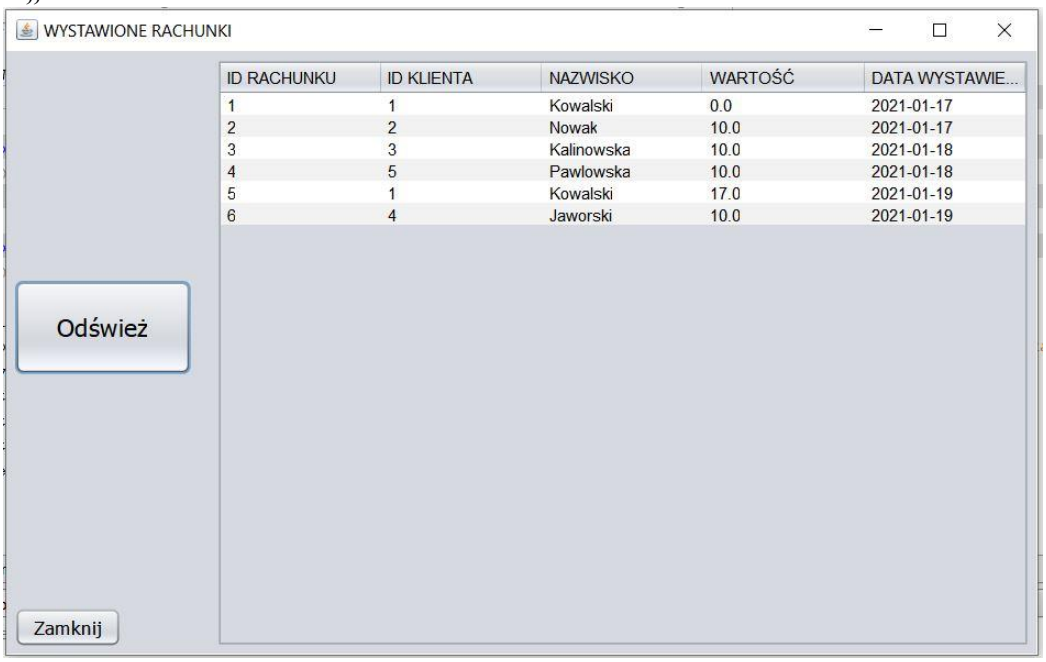
OKNO „KASETY”



Rysunek 10 Wygląd okna "KASETY"

Okno wyświetla informacje o wszystkich kasetach z bazy danych. Pozwala także na dodawanie i usuwanie rekordów z bazy.

OKNO „RACHUNKI”



Rysunek 11 Wygląd okna "RACHUNKI"

Okno zawiera informacje zawarte w tabeli „Rachunki” wyświetla historię wystawionych rachunków, a więc także wszystkich wypożyczeń, jakie zostały zakończone.

3.2.2 Metoda podłączenia do bazy danych – integracja z bazą danych

Integracja aplikacji z bazą danych została osiągnięta poprzez interfejs Java DataBase Connection. Przykład implementacji połączenia znajduje się na rysunku poniżej.

```
103        try{
104            Class.forName("com.mysql.cj.jdbc.Driver");
105            Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3308/wypozyczalnia_kaset","[redacted]","[redacted]");
106            System.out.println("polaczono z baza");
107            String password = passwordField.getText();
108            Statement statement = connection.createStatement();
109            String sql = "SELECT * from password where password = sha2('"+password+"',512)" ;
110            ResultSet resultSet = statement.executeQuery(sql);
111              if(resultSet.next()){
112
113                 dispose();
114                 MainMenu homepage = new MainMenu();
115                 homepage.show();
116             }
```

Rysunek 12 Przykład połączenia z bazą danych

Przykład został zaczerpnięty z kodu implementującego logowanie do aplikacji.

3.2.3 Projekt zabezpieczeń na poziomie aplikacji

Zabezpieczenia po stronie aplikacji obejmują jedynie walidację wprowadzanych danych. Do implementacji tej funkcjonalności użyte zostały wyrażenia regularne (ang. regular expresion – regex) i pakietu „java.util.regex". Przykład zastosowania znajduje się poniżej. Oprócz tego w kodzie sprawdza się, czy wszystkie potrzebne pola zostały wypełnione.

```
24      private final Pattern emailPattern = Pattern.compile("^.+@.+\\..+$");
25      private final Pattern lettersPattern = Pattern.compile("^[a-zA-Z]+$");
26      private final Pattern numberPattern = Pattern.compile("^[0-9]*$");
```

Rysunek 13 Przykład wyrażeń regularnych użytych do walidacji danych

Przykład został wzięty z kodu opisującego okno klientów. Obiekty klasy „java.util.regex.Pattern" opisują wyrażenia regularne. Linia 24 pozwala na walidację emaila. Linia 25 sprawdza, czy ciąg znaków zawiera tylko litery. Linia 26 pozwala na walidację pola, które powinno zawierać 9 cyfr, czyli numer telefonu.

```
27      private Matcher nameMatcher;
28      private Matcher surnameMatcher;
29      private Matcher numberMatcher;
30      private Matcher emailMatcher;
```

Rysunek 14 Obiekty typu java.util.regex.Matcher

```
272         numberMatcher = numberPattern.matcher(phone.getText());
273         nameMatcher = lettersPattern.matcher(name.getText());
274         surnameMatcher = lettersPattern.matcher(surname.getText());
275         emailMatcher = emailPattern.matcher(email.getText());
```

Rysunek 15 Inicjalizacja Matcher-ów

```
282         else if(!numberMatcher.find()){
283             JOptionPane.showMessageDialog(null, "Numer telefonumusi zawierać tylko litery");
284         }
285         else if(!(phone.getText().length()==9)){
286             JOptionPane.showMessageDialog(null, "Numer telefonu musi zawierać 9 cyfr");
```

Rysunek 16 Przykład walidacji numeru telefonu

Podczas próby dodania nowego klienta, następuje utworzenie obiektów porównujących wpisane ciągi znaków z wyrażeniami regularnymi. Powyższy przykład obrazuje walidację wprowadzonego numeru telefonu.

4. Implementacja systemu

4.1 Realizacja bazy danych

4.1.1 Tworzenie tabel i widoków oraz definiowanie ograniczeń

TABELA „KLIENCI”

```
CREATE TABLE Klienci(
id_klienta int auto_increment primary key,
imie_klienta varchar(20) not null,
nazwisko_klienta varchar(20) not null,
telefon integer not null,
email varchar(40))
```

TABELA „RACHUNK**I”**

```
CREATE TABLE Rachunki(
id_rachunku int auto_increment primary key,
id_klienta int not null references klienci(id_klienta),
wartosc float not null,
data_wystawienia date not null)
```

TABELA „TYTULY”

```
CREATE TABLE Tytuly(
id_tytulu int auto_increment primary key,
nazwa varchar(100) not null,
reżyser varchar(50) not null,
rok_wydania date not null)
```

TABELA „KASETY”

```
CREATE TABLE Kasety(
id_kasety int auto_increment primary key,
id_tytulu int not null references tytuly(id_tytulu),
czy_wypozyczona tinyint not null)
```

TABELA „WYPOZYCZENIA”

```
CREATE TABLE Wypozyczenia(
id_wypozyczenia int auto_increment primary key,
id_klienta int not null references klienci(id_klienta),
id_kasety int not null references kasety(id_kasety),
data_wypozyczenia date not null,
```

data_oddania date not null)

WIDOK „AKTUALNE_WYPOZYCZENIA”

```
CREATE VIEW aktualne_wypozyczenia AS
SELECT imie_klienta, nazwisko_klienta,
            wypozyczenia.id_kasety, nazwa,
            data_wypozyczenia
FROM
wypozyczenia JOIN klienci
ON wypozyczenia.id_klienta = klienci.id_klienta
JOIN kasety
ON wypozyczenia.id_kasety = kasety.id_kasety
JOIN tytuly
ON kasety.id_tytulu = tytuly.id_tytulu
```

4.1.2 Implementacja trigger-ów

TRIGGER „SETDATES”

```
CREATE TRIGGER `setDates`
BEFORE INSERT
ON `wypozyczenia`
FOR EACH ROW
BEGIN
SET new.data_wypozyczenia = curdate(),
    new.data_oddania = date_add(curdate(),interval 7 day);
END
```

TRIGGER „SETKASETAWYPOZYCZONA”

```
CREATE TRIGGER `setKasetaWypozyczona`
AFTER INSERT
ON `wypozyczenia`
FOR EACH ROW
UPDATE kasety
SET czy_wypozyczona=1
WHERE kasety.id_kasety = new.id_kasety
```

TRIGGER „SETKASETANIEWYPOZYCZONA”

```
CREATE
TRIGGER `setKasetaNiewypozyczona`
AFTER DELETE
ON `wypozyczenia`
FOR EACH ROW
UPDATE kasety
SET czy_wypozyczona=0
WHERE kasety.id_kasety = old.id_kasety
```

TRIGGER „UTWORZRACHUNEK”

```
CREATE TRIGGER `utworzRachunek`
AFTER DELETE
ON `wypozyczenia`
FOR EACH ROW
BEGIN
INSERT INTO rachunki (id_klienta,wartosc,data_wystawienia)
VALUES (old.id_klienta,
        10+7*datediff(curdate(), old.data_wypozyczenia)+
                                    if(datediff(curdate(), old.data_wypozyczenia)>7,50,0),
        curdate());
END
```

4.1.3 Implementacja mechanizmów zabezpieczeń

Tabela „Password” zawiera jedną kolumnę i jeden rekord będący hashem hasła i wynikiem funkcji SHA2():

```
CREATE TABLE Password(
Password varchar(128) primary key)
```

Dodanie hasła następuje w taki sposób:

```
INSERT into Password (password)
VALUES (sha2(‘tutaj haslo jako ciag znakow’,512))
```

4.2 Realizacja elementów aplikacji

Aplikacja została stworzona z pomocą biblioteki graficznej Swing.

- Wszystkie okienka są obiektami klasy javax.swing.JFrame
- Pola do wprowadzania tekstu zostały stworzone jako obiekty klasy javax.swing.JTextField
- Przyciski są obiektami klasy javax.swing.JButton
- Tabele są obiektami klasy javax.swing.JTable

4.2.1 Przykład operacji INSERT

```
if(!resultSet.next()){
    sql = "insert into klienci (imie_klienta,nazwisko_klienta,telefon,email) values"
            + " ('"+name.getText()+"','"+surname.getText()+"','"+phone.getText()+",'"+email.getText()+"')";
    statement.executeUpdate(sql);
    JOptionPane.showMessageDialog(null,"Dodano klienta");
```

Przykład pokazuje operację INSERT, która odpowiada za dodanie nowego klienta do tabeli KLIENCI.

4.2.2 Przykład operacji DELETE

```
Class.forName("com.mysql.cj.jdbc.Driver");
Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3308/wypozyczalnia_kaset","root","root");
Statement statement = connection.createStatement();
String sql = "delete from klienci where telefon = "+phone.getText();
statement.executeUpdate(sql);
JOptionPane.showMessageDialog(null,"Klient usunięty");
```

Przykład pokazuje operację DELETE, która odpowiada za usunięcie klienta z tabeli KLIENCI. Usuwany klient ma taki sam numer telefonu, jak wprowadzony przez użytkownika do aplikacji.

4.2.3 Przykład operacji SELECT

```
Class.forName("com.mysql.cj.jdbc.Driver");
Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3308/wypozyczalnia_kaset","root","root");
System.out.println("polaczono z baza");
Statement statement = connection.createStatement();
String sql = "SELECT id_rachunku, rachunki.id_klienta, nazwisko_klienta, wartosc, data_wystawienia\n" +"FROM wypozyczalnia_kaset.rachunki\n" +
        "JOIN klienci ON rachunki.id_klienta = klienci.id_klienta";
ResultSet resultSet = statement.executeQuery(sql);
DefaultTableModel tableModel = (DefaultTableModel)table.getModel();
tableModel.setRowCount(0);
while(resultSet.next()){
    String idRachunku = String.valueOf(resultSet.getInt("id_rachunku"));
    String idKlienta = String.valueOf(resultSet.getInt("rachunki.id_klienta"));
    String nazwisko = resultSet.getString("nazwisko_klienta");
    String wartosc = resultSet.getString("wartosc");
    String data = resultSet.getString("data_wystawienia");
    String dane[] = {idRachunku,idKlienta,nazwisko,wartosc,data};
    tableModel.addRow(dane);
}
connection.close();
```

Przykład pokazuje złożoną operację SELECT. Dane z tabel RACHUNKI i KLIENCI są łączone za pomocą operacji JOIN i wyświetlane w oknie „Wystawione rachunki”.

5. Testowanie systemu

5.1 Test operacji INSERT

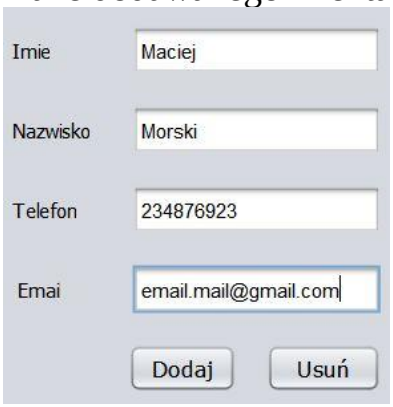
Jako test zostało przeprowadzone dodanie nowego klienta do bazy.

Zawartość tabeli KLIENCI przed dodaniem:

id_klienta	imie_klienta	nazwisko_klienta	telefon	email
1	Jan	Kowalski	123456789	jankowalski@gmail.com
2	Tomasz	Nowak	987654321	tomasznowak@gmail.com
3	Krystyna	Kalinowska	783897608	krystynakalinowska@gmail.com
4	Henryk	Jaworski	781189937	henrykjaworski@gmail.com
5	Marcelina	Pawlowska	797487529	marcelinapawlowska@gmail.com
6	Izolda	Sobczak	123456784	12 izoldasobczak@gmail.com
8	Karol	Wojtachnia	123456784	asd@mail.com
NULL	NULL	NULL	NULL	NULL

Rysunek 17 Tabla KLIENCI przed testem INSERT

Dane dodawanego klienta:



Rysunek 18 Dane do testu INSERT

Zawartość tabeli KLIENCI po dodaniu:

1	Jan	Kowalski	123456789	jankowalski@gmail.com
2	Tomasz	Nowak	987654321	tomasznowak@gmail.com
3	Krystyna	Kalinowska	783897608	krystynakalinowska@gmail.com
4	Henryk	Jaworski	781189937	henrykjaworski@gmail.com
5	Marcelina	Pawlowska	797487529	marcelinapawlowska@gmail.com
6	Izolda	Sobczak	671510612	izoldasobczak@gmail.com
8	Karol	Wojtachnia	123456784	asd@mail.com
10	Maciej	Morski	234876923	email.mail@gmail.com

Rysunek 19 Tabela KLIENCI po teście INSERT

Klient został poprawnie dodany do bazy danych. W tabeli pojawił się rekord o takich samych danych, jakie zostały wpisane w aplikacji.

5.2 Test operacji DELETE

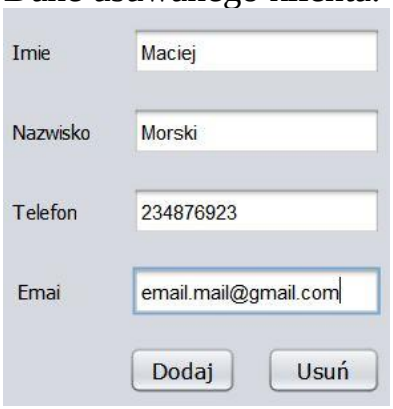
Jako test zostało przeprowadzone usunięcie klienta z bazy.

Zawartość tabeli KLIENCI po dodaniem:

1	Jan	Kowalski	123456789	jankowalski@gmail.com
2	Tomasz	Nowak	987654321	tomasznowak@gmail.com
3	Krystyna	Kalinowska	783897608	krystynakalinowska@gmail.com
4	Henryk	Jaworski	781189937	henrykjaworski@gmail.com
5	Marcelina	Pawlowska	797487529	marcelinapawlowska@gmail.com
6	Izolda	Sobczak	671510612	izoldasobczak@gmail.com
8	Karol	Wojtachnia	123456784	asd@mail.com
10	Maciej	Morski	234876923	email.mail@gmail.com

Rysunek 20 Tabela KLIENCI przed testem DELETE

Dane usuwanego klienta:



Rysunek 21 Dane testowe do testu DELETE

Zawartość tabeli KLIENCI po usunięciu:

id_klienta	imie_klienta	nazwisko_klienta	telefon	email
1	Jan	Kowalski	123456789	jankowalski@gmail.com
2	Tomasz	Nowak	987654321	tomasznowak@gmail.com
3	Krystyna	Kalinowska	783897608	krystynakalinowska@gmail.com
4	Henryk	Jaworski	781189937	henrykjaworski@gmail.com
5	Marcelina	Pawlowska	797487529	marcelinapawlowska@gmail.com
6	Izolda	Sobczak	123456784	12 izoldasobczak@gmail.com
8	Karol	Wojtachnia	123456784	asd@mail.com
NULL	NULL	NULL	NULL	NULL

Rysunek 22 Tabela KLIENCI po teście DELETE

Test przebiegł poprawnie. Odpowiedni klient został usunięty z bazy danych.

5.3 Test operacji UPDATE

Jako test zostało przeprowadzone usunięcie wypożyczenia, gdyż aktywacja triggera SETNIEWYPOZYCZONA powoduje UPDATE na tabeli KASET i ustawia stan wypożyczenia kasety o danym id.

Zawartość tabeli KASETY przed usunięciem wypożyczenia:

id_kasety	id_tytulu	czy_wypozyczona
1	1	0
2	2	1
3	1	1
4	2	0
5	3	1
6	3	1
7	4	0
8	4	1
9	5	0
10	5	0
11	6	0
12	6	0
13	7	0
14	7	0
15	8	0
16	8	0
17	9	0
18	9	0
19	10	0
20	10	0
NULL	NULL	NULL

Rysunek 23 Tablela KASETY przed testem UPDATE

Dane usuwanego wypożyczenia:

Rysunek 24 Dane testowe do testu UPDATE

Zawartość tabeli KASETY po usunięciu wypożyczenia:

id_kasety	id_tytulu	czy_wypozyczona
1	1	0
2	2	1
3	1	0
4	2	0
5	3	1
6	3	1
7	4	0
8	4	1
9	5	0
10	5	0
11	6	0
12	6	0
13	7	0
14	7	0
15	8	0
16	8	0
17	9	0
18	9	0
19	10	0
20	10	0
NULL	NULL	NULL

Rysunek 25 Tabela KASETY po teście UPDATE

Test przebiegł poprawnie. Kaseta o ID równym 3 zmieniła stan swojego wypożyczenia z 1 na 0, co oznacza, że jest wolna i można ją wypożyczyć

5.4 Test operacji SELECT

Do testów użyto okna „WYSTAWIONE RACHUNKI”. Po naciśnięciu przycisku ODŚWIEŻ, tabela w oknie powinna pokazać zawartość tabeli RACHUNKI z bazy danych połączonej z nazwiskiem klienta, do którego rachunek jest przypisany poprzez id.

Zawartość bazy daych:

1	Jan	Kowalski	123456789	jankowalski@gmail.com
2	Tomasz	Nowak	987654321	tomasznowak@gmail.com
3	Krystyna	Kalinowska	783897608	krystynakalinowska@gmail.com
4	Henryk	Jaworski	781189937	henrykjaworski@gmail.com
5	Marcelina	Pawlowska	797487529	marcelinapawlowska@gmail.com
6	Izolda	Sobczak	671510612	izoldasobczak@gmail.com
8	Karol	Wojtachnia	123456784	asd@mail.com
10	Maciej	Morski	234876923	email.mail@gmail.com

Rysunek 26 Tabela KLIENCI dla testu SELECT

id_rachunku	id_klienta	wartosc	data_wystawienia
1	1	0	2021-01-17
2	2	10	2021-01-17
3	3	10	2021-01-18
4	5	10	2021-01-18
5	1	17	2021-01-19
6	4	10	2021-01-19
7	1	10	2021-01-26
NULL	NULL	NULL	NULL

Rysunek 27 Tabela RACHUNKI dla testu SELECT

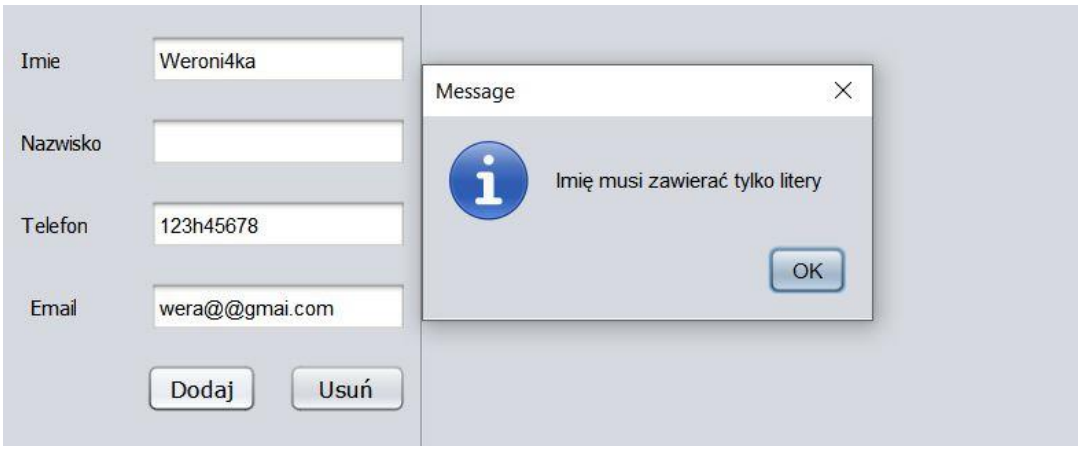
Zawartość tabeli w oknie aplikacji

Rysunek 28 Tabela w oknie aplikacji dla testu SELECT

5.5 Test walidacji danych w aplikacji

Test obejmował próbę dodania klienta z niepoprawnymi danymi. Testowy klient posiadał cyfrę w polu imienia, literę w numerze telefonu, puste pole nazwiska oraz dwa znaki ‘@’ w polu na adres mailowy

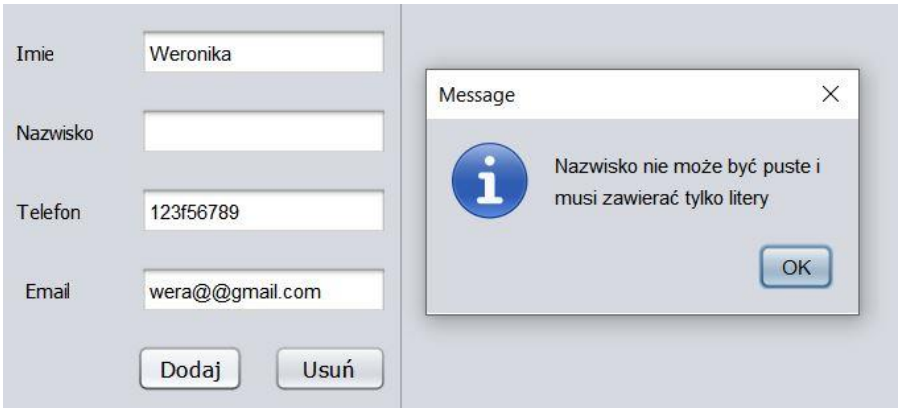
Test imienia:



Rysunek 29 Test walidacji - niepoprawne imię

Test zakończył się poprawnym wykryciem błędu.

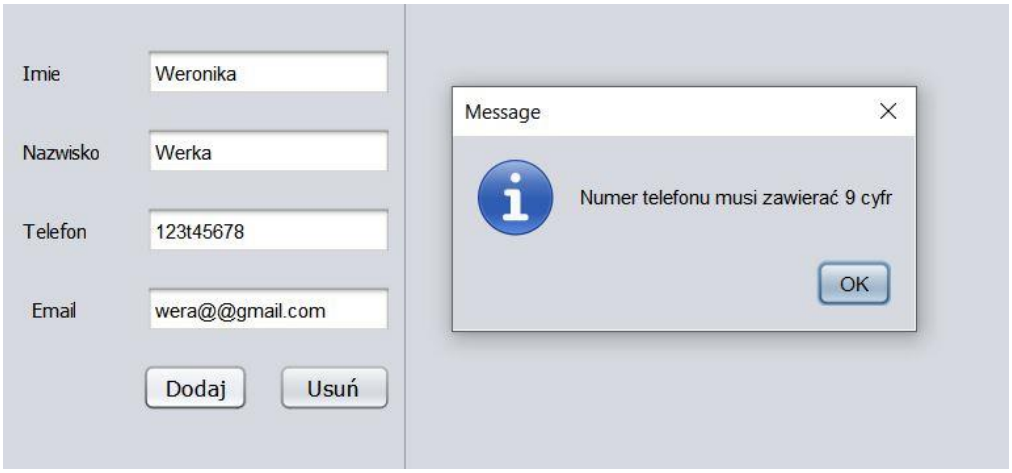
Test nazwiska:



Rysunek 30 Test walidacji - niepoprawne nazwisko

Test zakończył się poprawnym wykryciem błędu.

Test numeru telefonu:



Rysunek 31 Test walidacji - niepoprawny numer telefonu

Test zakończył się poprawnym wykryciem błędu.

Test maila:

Rysunek 32 Test walidacji - niepoprawny adres email

Test zakończył się wykryciem błędu w aplikacji. System nie powinien przyjąć maila zawierającego dwa razy znak '@'. Najprawdopodobniej zostało użyte złe wyrażenie regularne.

5.6 Wnioski z testów

Prawie wszystkie testy przebiegły pomyślnie. Jedynie walidacja wprowadzania emaila nie funkcjonuje w zaplanowany sposób. Wprowadzenie maila zawierającego dwa znaki ‘@’ nie będzie powodować błędów krytycznych dla działania aplikacji.

6. Podsumowanie

W projekcie udało się zrealizować wszystkie wymagania niefunkcjonalne oraz wszystkie najważniejsze wymagania funkcjonalne postawione na początku projektowania całego systemu.
W trackie implementacji bazy danych zrezygnowano z tabeli opisującej reżyserów tytułów. Zamiast tego dodano w tabeli TYTULY pole zawierające imię oraz nazwisko reżysera. Poza tą modyfikacją baza danych spełnia wszystkie postawione wymagania i jest zgoda jej projektem.
Stworzona aplikacja nie pozwala na edycję danych w bazie (np. danych klientów lub kaset oraz hasła dostępu do aplikacji). Przypadki użycia związane z modyfikacją danych są jedynymi, których brakuje w implementacji aplikacji. Pozostałe przypadki użycia znajdują swoje odzwierciedlenie w funkcjach aplikacji i możliwościach, jakie stworzona aplikacja stawia przed użytkownikiem.
W trakcie testowania systemu jedynym znalezionym błędem było akceptowanie przez aplikację emaila, który zawiera dwa znaki ‘@’. Teoretycznie taki adres mailowy jest dopuszczalny, jednak niespotykany. Zastosowaną implementację można pozostawić w aktualnym stanie, jednak dobrze byłoby dodać komunikat potwierdzający dwa znaki tego typu, aby użytkownik sprawdził jeszcze raz poprawność wpisanego maila.

Literatura

[1] Górski J., „Inżynieria oprogramowania w projekcie informatycznym”, Mikom, Warszawa 2000.
[2] Beyon-Davies P., „Systemy baz danych”, WNT, Warszawa, 2000.
[3] Garcia-Molina H., Ullman J.D., Widom J., „Systemy baz danych. Kompletny podręcznik.” Wydanie II,
Prentice Hall, New Jersey, 2011.
[4] Deitel Paul, Deitel Harvey, „Programowanie w Javie. Solidna wiedza w praktyce”. Wydanie XI, Helion,
2018.
[5] Dokumentacja języka Java, https://docs.oracle.com/en/java/
[6] Strona internetowa MySQL Workbench, https://www.mysql.com/products/workbench/
[7] Dokumentacja MySQL, https://dev.mysql.com/doc/
[8] Visual Paradigm, https://www.visual-paradigm.com/