



Politechnika Wrocławska

WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI

Bazy Danych 2 – Projekt

Temat:

„Bazodanowy system obsługi sklepu sieciowego zajmującego się sprzedażą elektroniki”

Termin zajęć:

Piątek, 11:15 – 13:00

Autorzy:

Kamil Dywan, 254049

Magdalena Komorowska, 256497

Prowadzący zajęcia:

Dr inż. Roman Ptak

Wrocław, 2021 r.

Spis treści

1. Wstęp	3
1.1. Cel projektu.....	3
1.2. Zakres projektu	3
2. Analiza wymagań	3
2.1. Opis działania systemu	3
2.2. Opis zasobów ludzkich.....	3
2.3. Wymagania funkcjonalne.....	3
2.4. Wymagania niefunkcjonalne	4
2.4.1. Wymagania dotyczące bezpieczeństwa systemu	4
2.4.2. Wymagania dotyczące rozmiaru bazy danych	4
2.4.3. Wykorzystywane technologie i narzędzia.....	4
3. Projekt systemu	5
3.1. Projekt bazy danych.....	5
3.1.1. Uproszczony model konceptualny	5
3.1.2. Model logiczny	5
3.1.3. Model fizyczny	6
3.1.4. Inne elementy schematu – mechanizmy przetwarzania danych	6
3.1.5. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych.....	15
3.2. Modelowanie aplikacji dostępowej	16
3.2.1. Diagram przypadków użycia.....	16
3.2.2. Interfejs graficzny	17
3.2.3. Metoda podłączania do bazy danych – integracja z bazą danych	22
3.2.4. Projekt zabezpieczeń na poziomie aplikacji	22
4. Implementacja systemu baz danych	23
4.1. Tworzenie tabel i definiowanie ograniczeń na podstawie modelu	23
4.1.1. Tworzenie tabel, ograniczeń check oraz kluczy głównych.....	23
4.1.2. Klucze obce	25
4.2. Implementacja mechanizmów przetwarzania danych	25
4.3. Implementacja uprawnień i innych zabezpieczeń.....	32
4.4. Testowanie bazy danych na przykładowych danych	33
4.4.1. Generowanie danych testowych	33
4.4.2. Testy wydajnościowe.....	34
4.4.3. Testy na tak i na nie	41

4.4.4.	Testy uprawnień	44
5.	Implementacja i testy aplikacji	47
5.1.	Instalacja i konfigurowanie systemu	47
5.1.1.	Perspektywa użytkownika	47
5.1.2.	Perspektywa programisty	47
5.1.3.	Perspektywa wdrożeniowca	48
5.2.	Instrukcja użytkowania aplikacji	48
5.3.	Testowanie opracowanych funkcji systemu	56
5.3.1.	Rejestracja	56
5.3.2.	Logowanie	58
5.3.3.	Zmiana danych pracowniczych	59
5.3.4.	Dodawanie produktu	61
5.3.5.	Edytowanie produktu	63
5.3.6.	Usuwanie produktu	64
5.3.7.	Wyszukiwanie produktów	65
5.4.	Omówienie wybranych rozwiązań programistycznych	67
5.4.1.	Implementacja interfejsu dostępu do bazy danych	67
5.4.2.	Implementacja wybranych funkcjonalności systemu	68
5.4.3.	Implementacja mechanizmów bezpieczeństwa	69
6.	Podsumowanie i wnioski	69
6.1.	Literatura	70
6.2.	Spis rysunków	70
6.3.	Spis tabel	72
7.	Link do plików projektu	73

1. Wstęp

1.1.Cel projektu

Celem projektu jest implementacja bazy danych oraz aplikacji dostępowej dla pracownika i klienta sklepu internetowego.

1.2. Zakres projektu

Firma zajmuje się jedynie sprzedażą produktów. Projekt nie obejmuje kwestii związanych z magazynowaniem produktów.

2. Analiza wymagań

2.1.Opis działania systemu

Głównym powodem stworzenia systemu jest usprawnienie pracy firmy oraz zapewnienie wygodnej i szybkiej obsługi klienta. System będzie się składał z bazy danych przechowującej dane o produktach i transakcjach oraz aplikacji dla pracownika i klienta do wygodnej edycji zasobów sklepu. System będzie zarządzał listą produktów oraz automatycznie "obsługiwał" transakcje. Możliwe będzie zalogowanie się do aplikacji na konto pracownika lub klienta. W zależności od wybranego konta dostępne będą inne opcje korzystania z bazy danych sklepu. System wykonuje automatycznie transakcję wyliczając ceny produktów oraz ich promocję po uzupełnieniu odpowiednich danych oraz zaakceptowaniu zakupu przez klienta. System przechowuje dane o transakcjach. Po dokonaniu transakcji baza danych jest automatycznie aktualizowana.

2.2.Opis zasobów ludzkich

Pracownik ma możliwość edytowania, usuwania i dodawania produktów do bazy produktów. Dodatkowo ma wgląd do historii dokonanych zakupów. Każdy produkt reprezentowany jest przez nazwę, cenę, promocję od ceny, id produktu, liczbę w magazynie, obrazek poglądowy i nazwę producenta. Klient może dokonać zakupu poprzez dodanie produktów do "koszyka" i zaakceptowaniu transakcji oraz może także przeglądać historię swoich zakupów. Klient identyfikowany jest za pomocą numeru karty, z której skorzystał w trakcie zakupów. Po wybraniu danego produktu klient może wybrać jego ilość.

2.3.Wymagania funkcjonalne

- pracownik może edytować, dodawać i usuwać produkty,
- dane klientów przechowywane są w systemie (numer karty),
- w bazie danych przechowywane są dane o produktach, promocjach, oraz transakcjach,
- pracownik aby dokonać jakichkolwiek zmian oraz mieć dostęp do zasobów sklepu z perspektywy firmy, musi być zalogowany,
- system automatycznie obsługuje transakcje wyliczając wartość rachunku wraz z uwzględnieniem promocji,
- klient samodzielnie dokonuje wyboru produktów, może edytować zawartość "koszyka",
- klient może dokonać transakcji poprzez uzupełnienie odpowiednich danych oraz zaakceptowanie zakupu,
- klient może przeglądać historię swoich transakcji.

2.4. Wymagania niefunkcjonalne

2.4.1. Wymagania dotyczące bezpieczeństwa systemu

- w razie konieczności uprawnieni pracownicy mogą zmienić dane dokonanych transakcji,
- klient nie może zakupić większej liczby produktów od ich liczby znajdujących się w magazynie,
- klient aby dokonać zakupu musi być zalogowany,
- każdy pracownik ma swoje własne konto, dzięki czemu można sprawdzić, kto zrobił jakąś zmianę, dlatego aby pracownik mógł dokonać jakichkolwiek zmian oraz mieć dostęp do zasobów sklepu z perspektywy firmy, musi być zalogowany.

2.4.2. Wymagania dotyczące rozmiaru bazy danych

- firma posiada 2000 zarejestrowanych klientów,
- firma jest nakierowana tylko na rynek w ramach Polski
- przy czym oszacowano, że w jednym momencie system jest w stanie obsłużyć 1000 klientów,
- liczba dostępnych produktów wynosi 1000 podzielonych na 50 kategorii,
- oszacowano, że dzienna liczba transakcji wynosi 200,
- przy czym największy ruch jest w godzinach wieczornych.

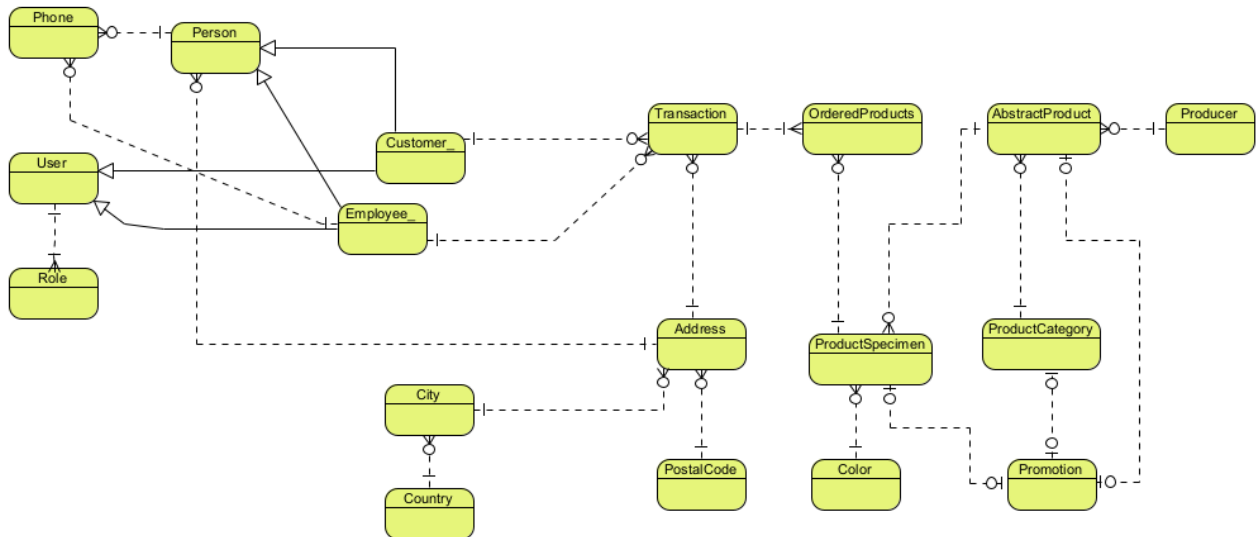
2.4.3. Wykorzystywane technologie i narzędzia

- DBMS - Oracle Database
- Backend - Java (JDBC)
- Frontend (GUI) - Java (Swing)
- Windows

3. Projekt systemu

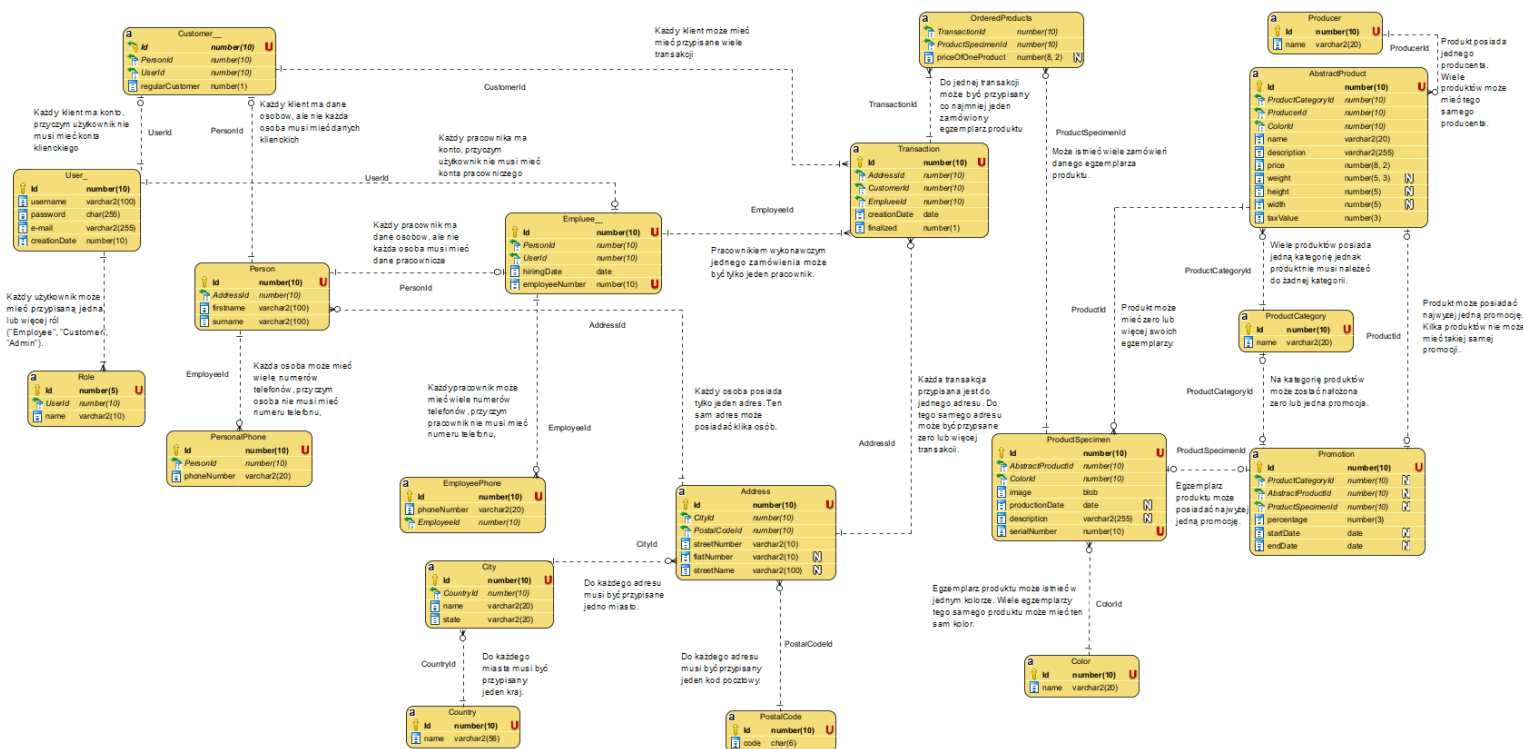
3.1. Projekt bazy danych

3.1.1. Uproszczony model konceptualny



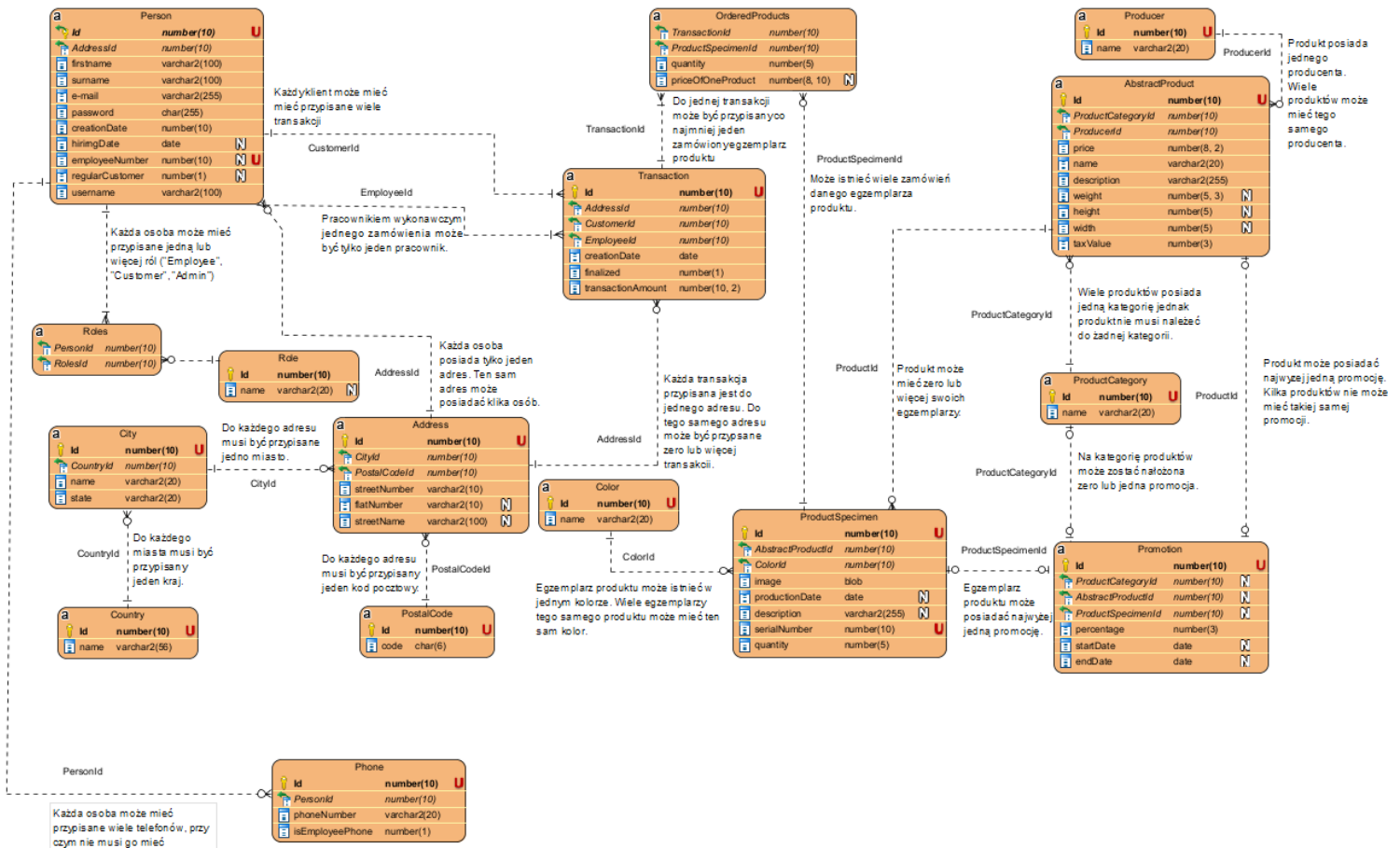
Rysunek 1 Uproszczony model konceptualny bazy danych

3.1.2. Model logiczny



Rysunek 2 Model logiczny bazy danych

3.1.3. Model fizyczny



Rysunek 3 Model fizyczny bazy danych

3.1.4. Inne elementy schematu – mechanizmy przetwarzania danych

Sekwencje

Dla każdego klucza głównego ze wszystkich tabel będzie tworzona sekwencja, której minimalna wartość wynosi 1, maksymalna w zależności od maksymalnej wielkości klucza głównego oraz jej wartość będzie zwiększana o 1. Przykładowa sekwencja dla klucza głównego z tabeli *Person* jest przedstawiona poniżej.

```
CREATE SEQUENCE BD_2.COUNTRY_ID_SEQ
MINVALUE 1
INCREMENT BY 1;
```

Triggery

Triggery zostaną wykorzystane do generowania kluczy głównych korzystając z wcześniej utworzonych sekwencji. Przed wstawieniem nowego wiersza do danej tabeli, do klucza głównego w tym wierszu przypisywana jest kolejna wartość

z sekwencji. Przykładowy trigger z przedstawionym zastosowaniem znajduje się poniżej.

```
CREATE OR REPLACE TRIGGER PERSON_ID_T
BEFORE INSERT ON BD_2.Person
FOR EACH ROW
    WHEN (NEW.Id is NULL)
BEGIN
    :NEW.Id := test_seq.NEXTVAL;
END;
```

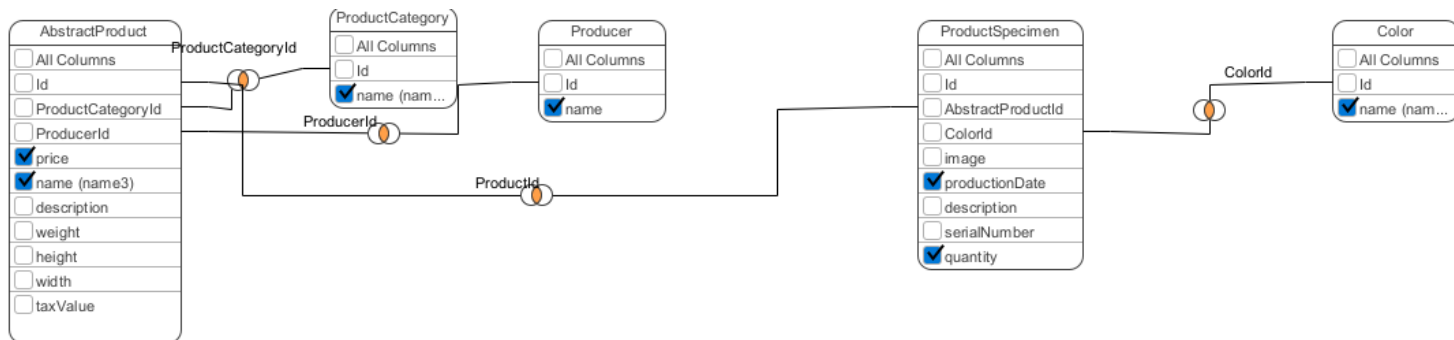
Kolejnym zastosowaniem triggerów jest aktualizowanie liczby produktów po dokonanej transakcji. Po wstawieniu zamówionego przedmiotu z danego zamówienia, zmniejszana jest liczba produktu o liczbę zamówionych sztuk tego produktu. Kod triggera znajduje się poniżej.

```
CREATE OR REPLACE TRIGGER UPDATE_PRODUCT
AFTER INSERT ON BD_2.ORDERED_PRODUCTS
FOR EACH ROW
BEGIN
    UPDATE BD_2.PRODUCT_SPECIMEN p
    SET p.QUANTITY = p.QUANTITY - :new.QUANTITY;
END;
```

Trigger nie pozwalający stworzyć promocji w sytuacji, kiedy 3 klucze obce miałyby być równe null.

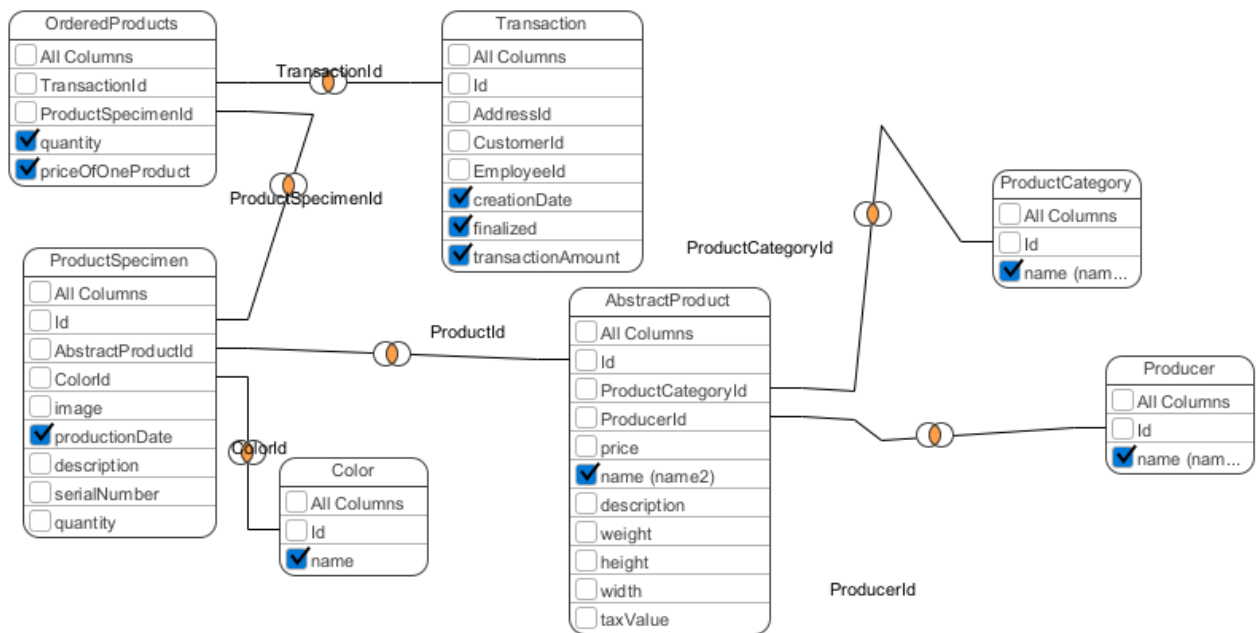
```
CREATE OR REPLACE TRIGGER INSERT_PROMOTION
BEFORE INSERT ON BD_2.PROMOTION
FOR EACH ROW
BEGIN
    IF (:new.PRODUCT_CATEGORY_ID IS NULL AND
        :new.ABSTRACT_PRODUCT_ID IS NULL AND
        :new.PRODUCT_SPECIMEN_ID IS NULL)
    THEN
        RAISE_APPLICATION_ERROR( -20000, 'Promocja
musi mieć kategorię' );
    END IF;
END;
```

Widoki



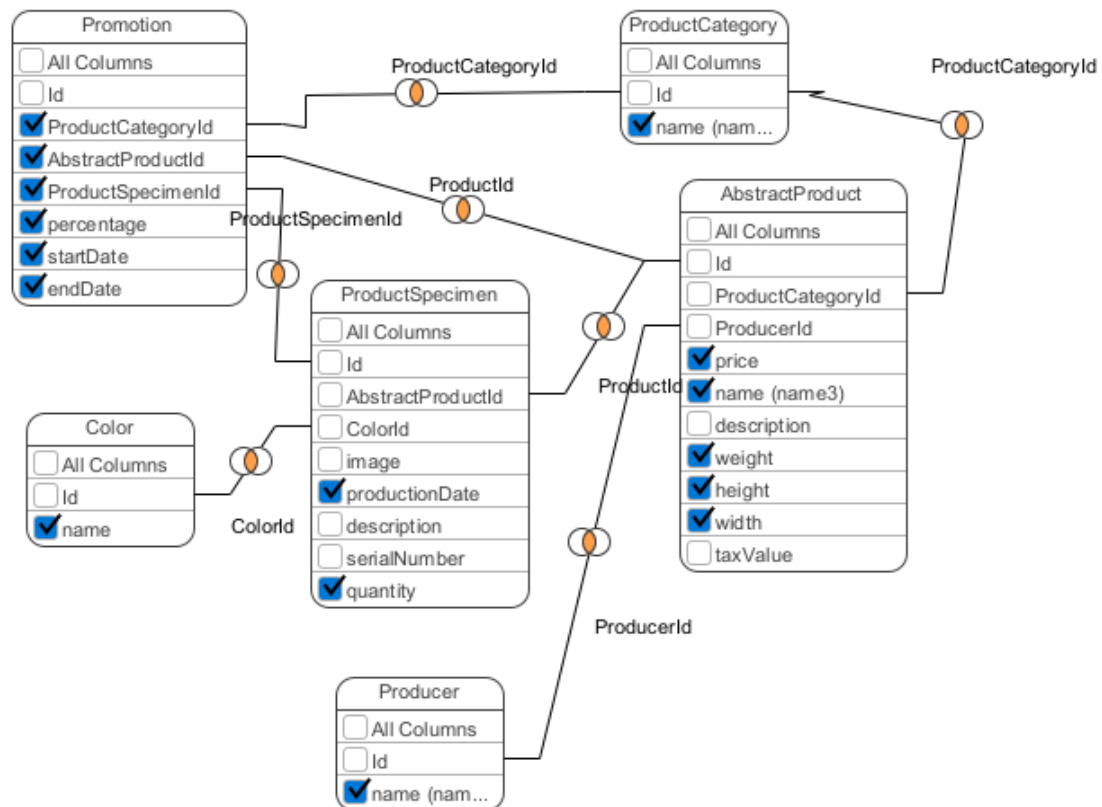
Rysunek 4 Widok produktów z ceną niższą niż 100

```
CREATE VIEW BD_2.PRODUCTS_WITH_PRICE_LESS_100 AS
SELECT
    PRODUCER.name Producer,
    P_C.name "Product category",
    A_P.name "Abstract Product",
    A_P.price,
    COLOR.name "Color",
    P_S.quantity,
    P_S.PRODUCTION_DATE
FROM BD_2.ABSTRACT_PRODUCT A_P
INNER JOIN BD_2.PRODUCT_SPECIMEN P_S ON A_P.Id =
P_S.ABSTRACT_PRODUCT_ID
INNER JOIN BD_2.PRODUCT_CATEGORY P_C ON
A_P.PRODUCT_CATEGORY_ID = P_C.Id
INNER JOIN BD_2.PRODUCER ON A_P.PRODUCER_ID = PRODUCER.Id
INNER JOIN BD_2.COLOR ON P_S.COLOR_ID = COLOR.Id
WHERE A_P.price < 100
ORDER BY A_P.price DESC;
```



Rysunek 5 Widok zamówień oraz transakcji

```
CREATE VIEW BD_2.ALL_ORDERED_PRODUCTS AS
SELECT
    O_P.quantity,
    O_P.price_Of_One_Product,
    Color.name "Color name",
    t.finalized,
    t.transaction_Amount,
    t.creation_Date,
    p_s.production_Date,
    a_p.name "Abstract product name",
    Producer.name "Producer name",
    p_c.name "Product category name"
FROM BD_2.ORDERED_PRODUCTS O_P
INNER JOIN BD_2.TRANSACTION_T t ON O_P.Transaction_Id =
t.Id
INNER JOIN BD_2.PRODUCT_SPECIMEN p_s ON
O_P.Product_Specimen_Id = p_s.Id
INNER JOIN BD_2.Color ON p_s.Color_Id = Color.Id
INNER JOIN BD_2.ABSTRACT_PRODUCT a_p ON
p_s.Abstract_Product_Id = a_p.Id
INNER JOIN BD_2.Producer ON a_p.Producer_Id = Producer.Id
INNER JOIN BD_2.PRODUCT_CATEGORY p_c ON
a_p.Product_Category_Id = p_c.Id;
```



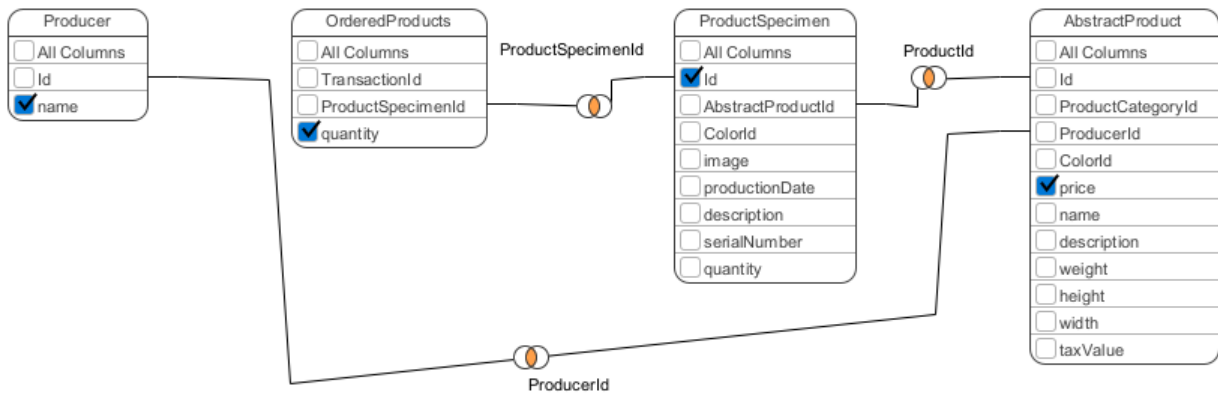
Rysunek 6 Widok egzemplarzy produktów objęte promocjom

```
CREATE VIEW BD_2.PRODUCTS_ON_PROMOTIONS AS
SELECT
    Color.name,
    Producer.name "Producer",
    a_p.name "Abstract product name",
    a_p.price,
    p.Product_Category_Id,
    p.Abstract_Product_Id,
    p.Product_Specimen_Id,
    p.start_Date,
    p.end_Date,
    p.percentage,
    p_s.quantity,
    p_s.production_Date,
    a_p.weight,
    a_p.height,
    a_p.width,
    p_c.name AS name4
FROM BD_2.Promotion p
INNER JOIN BD_2.Product_Category p_c ON
p.Product_Category_Id = p_c.Id
INNER JOIN BD_2.Abstract_Product a_p ON
p.Abstract_Product_Id = a_p.Id AND p_c.Id =
a_p.Product_Category_Id
```

```

INNER JOIN BD_2.Product_Specimen p_s ON
p.Product_Specimen_Id = p_s.Id AND a_p.Id =
p_s.Abstract_Product_Id
INNER JOIN BD_2.Producer ON a_p.Producer_Id = Producer.Id
INNER JOIN BD_2.Color ON p_s.Color_Id = Color.Id
ORDER BY p.percentage DESC;

```

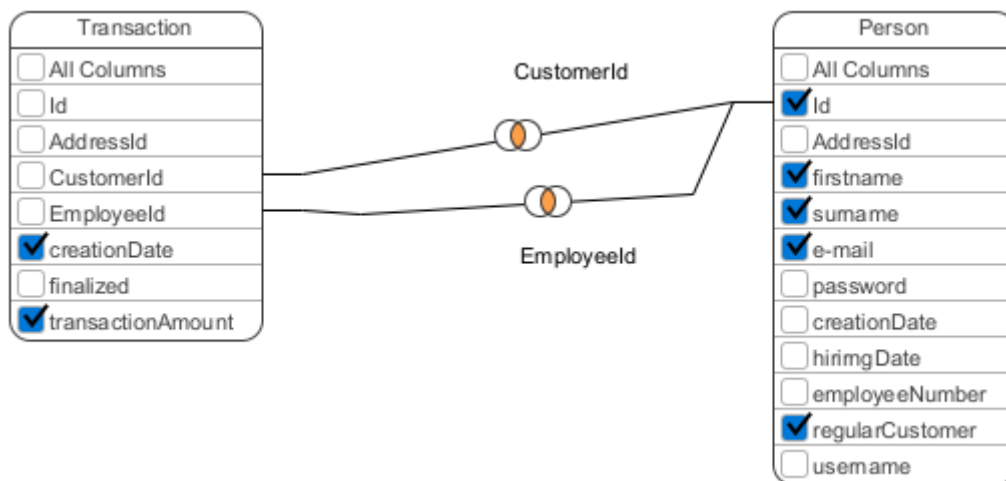


Rysunek 7 Widok najlepiej sprzedających się egzemplarzy produktów

```

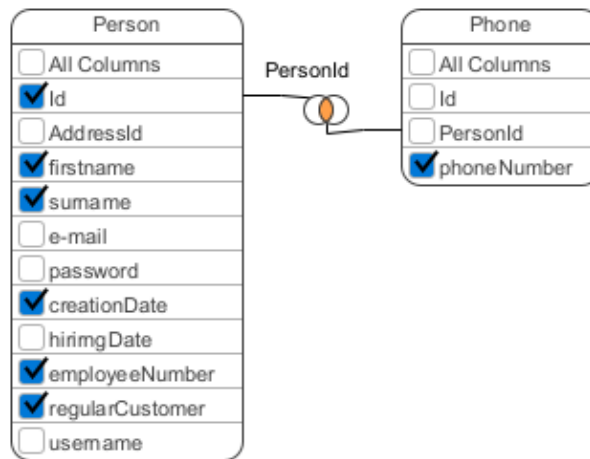
CREATE VIEW BD_2.Best_Selling_Products AS
SELECT
    p_s.production_Date,
    Color.name "Color",
    Producer.name "Producer",
    a_p.name AS "Abstract product name",
    o_p.price_Of_One_Product,
    SUM(o_p.quantity) sold
FROM BD_2.Ordered_Products o_p
INNER JOIN BD_2.Product_Specimen p_s ON p_s.Id =
o_p.Product_Specimen_Id
INNER JOIN BD_2.Abstract_Product a_p ON
p_s.Abstract_Product_Id = a_p.Id
INNER JOIN BD_2.Color ON p_s.Color_Id = Color.Id
INNER JOIN BD_2.Producer ON a_p.Producer_Id = Producer.Id
GROUP BY o_p.quantity, p_s.production_Date, Color.name,
Producer.name, a_p.name, o_p.price_Of_One_Product
ORDER BY o_p.quantity DESC;

```



Rysunek 8 Widok sfinalizowanych zamówień i przypisanych do nich klientów

```
CREATE VIEW BD_2.finalized_Orders AS
SELECT
    t.transaction_Amount,
    t.creation_Date,
    p.firstname,
    p.surname,
    p.Id,
    p.regular_Customer,
    p.e_mail
FROM BD_2.Transaction_t t
INNER JOIN BD_2.Person p ON t.Customer_Id = p.Id AND
t.Employee_Id = p.Id
WHERE p.employee_Number IS NULL;
```



Rysunek 8 Numery telefonów danej osoby

```
CREATE VIEW BD_2.Person_Numbers AS
SELECT
    Phone.phone_Number,
    Person.firstname,
    Person.surname,
    Person.Id,
    Person.regular_Customer,
    Person.employee_Number,
    Person.creation_Date
FROM BD_2.Person
INNER JOIN BD_2.Phone ON Person.Id = Phone.Person_Id;
```

Indeksy

```
CREATE INDEX BD_2.transaction_creation_date_done
ON BD_2.Transaction_t (creation_Date, finalized);

CREATE INDEX BD_2.product_quantity
ON BD_2.Product_Specimen (ABSTRACT_Product_Id, quantity);

CREATE INDEX BD_2.product_price_desc
ON BD_2.Abstract_Product (price DESC, Id);

CREATE INDEX BD_2.promotion_start_date
ON BD_2.PROMOTION (start_date DESC);

CREATE INDEX BD_2.promotion_end_date
ON BD_2.PROMOTION (end_date ASC);

CREATE INDEX BD_2.promotion_perc_end_start_date
ON BD_2.PROMOTION (percentage, end_Date, start_Date);
```

```

CREATE INDEX BD_2.product_name
ON BD_2.Abstract_Product (name);

CREATE INDEX BD_2.person_regular_Customer
ON BD_2.Person(regular_Customer);

CREATE INDEX BD_2.person_employee_Number
ON BD_2.Person (employee_Number);

CREATE INDEX BD_2.product_Producent_Id
ON BD_2.Abstract_Product (Producer_Id);

CREATE INDEX BD_2.product_production_Date
ON BD_2.Product_Specimen (production_Date);

CREATE INDEX BD_2.ordered_prod_Prod_Specimen_Id
ON BD_2.Ordered_Products (Product_Specimen_Id);

CREATE INDEX BD_2.loc_street_name_number_flat
ON BD_2.Address(street_Name, street_Number, flat_Number);

CREATE INDEX BD_2.phone__phone_number
ON BD_2.PHONE(PHONE_NUMBER);

```

Procedury składowane

Procedura wykonująca utworzenie transakcji w stanie początkowym (*finalized = 0*):

```

CREATE OR REPLACE PROCEDURE BD_2.INSERT_TRANSACTION
(creation_Date DATE,
  Customer_Id NUMBER, Employee_Id NUMBER, Address_Id
NUMBER, transaction_Amount NUMBER)
AS
BEGIN
  INSERT INTO BD_2.TRANSACTION_t (Address_Id,
  Customer_Id, Employee_Id, creation_Date, finalized,
  transaction_Amount)
  VALUES(Address_Id, Customer_Id, Employee_Id,
  creation_Date, 0, transaction_Amount);
END;

```

Procedura wykonująca utworzenie zamówionego produktu:

```
CREATE OR REPLACE PROCEDURE BD_2.INSERT_ORDERED_PRODUCTS
(Transaction_Id NUMBER,
  Product_Specimen_Id NUMBER, quantity NUMBER)
AS
BEGIN
  INSERT INTO BD_2.ORDERED_PRODUCTS(Transaction_Id,
    Product_Specimen_Id, quantity)
    VALUES(Transaction_Id, Product_Specimen_Id, quantity);
END;
```

Procedura obliczająca liczbę różnych produktów z danego zamówienia:

```
CREATE OR REPLACE PROCEDURE
BD_2.NUMBER_OF_ORDERED_PRODUCTS (Transaction_Id NUMBER,
sum_of_products OUT NUMBER)
AS
BEGIN
  SELECT COUNT(*) "Sum of diffrent products"
  INTO sum_of_products
  FROM BD_2.Ordered_Products o
  WHERE o.Transaction_Id = Transaction_Id;
END;
```

Zostanie zaimplementowana procedura wyznaczająca liczbę dni do końca promocji:

```
CREATE OR REPLACE PROCEDURE BD_2.PROMOTION_DAYS_LEFT
(PROMOTION_ID NUMBER, DAYS_LEFT OUT NUMBER)
AS
BEGIN
  SELECT trunc(start_date,'day') -
trunc(end_date,'day') "Days left"
  INTO DAYS_LEFT
  FROM BD_2.PROMOTION p
  WHERE p.ID = PROMOTION_ID;
END;
```

3.1.5. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych

Bezpieczeństwo na poziomie bazy danych będzie zapewnione poprzez utworzenie użytkowników o różnych dostępach do danych w bazie (admin, pracownik, klient). Przy logowaniu się do bazy danych użytkownik uzyska uprawnienia zgodne z jego typem konta (rolą). Poniżej znajduje się projekt uprawnień w zależności od typu użytkownika.

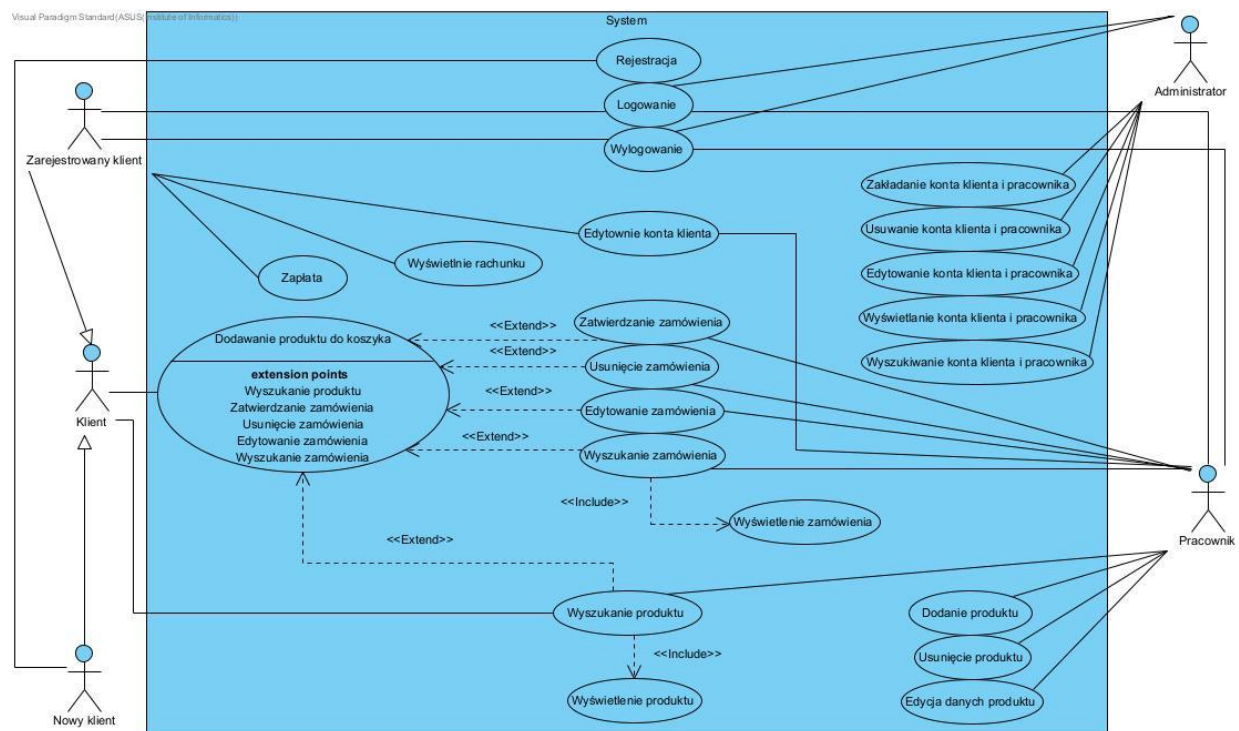
Tabela 1 Tabela uprawnień aktorów

Upewnienia / Rola	Administrator	Pracownik	Klient	Niealogowany
Person	A	S	S, I, U	-
Role	A	S	S	S
Country	A	S	S	-
City	A	S	S, I	-
PostalCode	A	S	S, I	-
Address	A	A	S, I	-
OrderedProducts	S	S	S	-
Transaction	S	S, I, U	S	-
AbstractProduct	S	A	S	S
ProductSpecimen	S	A	S	S
Producer	S	A	S	S
ProductCategory	S	A	S	S
Promotion	S	A	S	S
Color	A	A	S	S
Phone	A	A	A	-

Legenda: A – all, S – select, I – insert, U – update

3.2. Modelowanie aplikacji dostępowej

3.2.1. Diagram przypadków użycia



Rysunek 4 Diagram przypadków użycia systemu

3.2.2. Interfejs graficzny

Sklep z elektroniką

Przejdźcie do sklepu bez logowania

Logowanie

Rejestracja

Zamknij

Rysunek 5 Strona startowa aplikacji

Rejestracja

Typ konta:

Klient

Zatwierdź

Rysunek 6 Wybór rodzaju konta do utworzenia

Rejestracja klienta

Login:	<input type="text"/>	Data urodzenia:	<input type="text" value="11/28/2021"/>
Hasło:	<input type="text"/>	E-mail:	<input type="text"/>
Imię:	<input type="text"/>	Numer telefonu:	<input type="text"/>
Nazwisko:	<input type="text"/>		

Państwo	Województwo/Stan	Miasto
---------	------------------	--------

Ulica	Kod-pocztowy	Numer budynku	Numer mieszkania/lokalu
-------	--------------	---------------	-------------------------

Zarejestruj

Rysunek 7 Rejestracja w systemie dla klienta

Rejestracja pracownika

Login:	<input type="text"/>	Data urodzenia:	<input type="text" value="11/28/2021"/>
Hasło:	<input type="text"/>	E-mail:	<input type="text"/>
Imię:	<input type="text"/>	Numer telefonu:	<input type="text"/>
Nazwisko:	<input type="text"/>	Numer pracownika:	<input type="text"/>

[Zarejestruj](#)



Rysunek 8 Rejestracja w systemie dla pracownika

Do utworzenia konta pracownika wymagane jest wprowadzenie unikatowego dla danego pracownika „numeru pracownika”. Numer ten jest udostępniany w postaci papierowej podczas pierwszej wizyty w firmie pracownika. Numer ten widnieje w tabeli *Employee* pod nazwą *employeeNumber*.

Sklep z elektroniką

[Koszyk](#)


Peryferia [Wyślij](#)

	Myszka komputerowa XYZ 2 120,00zł Dodaj do koszyka
	Klawiatura ABC 4 300,00zł Dodane do koszyka >

Rysunek 9 Widok zasobów sklepu z perspektywy zalogowanego/niezalogowanego klienta

Detale produktu

[Koszyk](#)

	Myszka komputerowa XYZ 2 120,00zł
Rozmiar: 8x5	
Waga: 100g	
Data produkcji: 28.10.2021	Dodaj do koszyka
Liczba przycisków: 2	
Typ: optyczna	

Rysunek 10 Detale produktu

Przy czym detale „Liczba przycisków” oraz „Typ” znajdują się w tabeli *Product* pod atrybutem *description*.

Koszyk(2)

	Myszka komputerowa XYZ 2 120,00zł	<input type="text" value="2"/>	<input type="button" value="Usuń >"/>	Łączna kwota: 540,00zł <input type="button" value="Przejdź do zamówienia"/>
	Klawiatura ABC 4 300,00zł	<input type="text" value="1"/>	<input type="button" value="Usuń >"/>	

Rysunek 11 Widok koszyka

W koszyku można edytować liczbę produktów, usunąć liczbę danego produktu lub złożyć zamówienie.

Zasoby sklepu

	Myszka komputerowa XYZ 2 120,00zł	<input type="button" value="Edytuj"/>	<input type="button" value="Usuń"/>
	Klawiatura ABC 4 300,00zł	<input type="button" value="Edytuj"/>	<input type="button" value="Usuń"/>

Rysunek 12 Widok zasobów sklepu z perspektywy zalogowanego pracownika

Zmiana danych produktu

 <input type="button" value="Wybierz obraz"/>	Nazwa:	<input type="text"/>	Cena:	<input type="text"/>
	Rozmiar:	<input type="text"/>		
	Waga:	<input type="text"/>		
	Data produkcji:	<input type="text"/>		
	Opis:	<input type="text"/>		
<input type="button" value="Zapisz"/>				

Rysunek 13 Widok edycji danych produktu przez pracownika

Wybór lokalizacji dostawy

Użyj adresu z konta

Wprowadz inny adres

Anuluj

Rysunek 14 Wybór lokalizacji dostawy

Historia transakcji

28.11.2021

200,00zł

Zobacz szczegóły

28.11.2021

200,00zł

Zobacz szczegóły

28.11.2021

200,00zł

Zobacz szczegóły

Rysunek 15 Historia transakcji klienta

Klient ma dostęp do wszystkich wykonanych przez niego transakcji.

28.11.2021

200,00zł

Myszka komputerowa XYZ 2 x2 200,00zł

Klawiatura ABC 4 x1 100,00zł

Łącznie: 300,00zł

Rysunek 16 Detale danej transakcji klienta

Aktualizacja danych

Login:	<input type="text" value="aaaaaa"/>	Data urodzenia:	<input type="text" value="11/28/2021"/>
Hasło:	<input type="text" value="aaaaa"/>	E-mail:	<input type="text" value="cccc"/>
Imię:	<input type="text" value="bbb"/>	Numer telefonu:	<input type="text" value="aaaaa"/>
Nazwisko:	<input type="text" value="cccccc"/>		

Państwo	Województwo/Stan	Miasto
---------	------------------	--------

Ulica	Kod pocztowy	Numer budynku	Numer mieszkania/lokalu
-------	--------------	---------------	-------------------------

Zapisz

Rysunek 17 Aktualizacja danych klienta

Aktualizacja danych

Login:	<input type="text" value="aaaaaa"/>	Data urodzenia:	<input type="text" value="11/29/2021"/>
Hasło:	<input type="text" value="aaaaa"/>	E-mail:	<input type="text" value="cccc"/>
Imię:	<input type="text" value="bbb"/>	Numer telefonu:	<input type="text" value="aaaaa"/>
Nazwisko:	<input type="text" value="cccccc"/>		

Zapisz

Rysunek 18 Aktualizacja danych pracownika

3.2.3. Metoda podłączania do bazy danych – integracja z bazą danych

Do połączenia z bazą danych zostanie wykorzystana biblioteka JDBC Aby się połączyć z bazą danych należy skorzystać ze specjalnego sterownika, który tłumaczy odwołania z poziomu Javy na odwołania właściwe dla danego RDBMS (w tym przypadku Oracle Database). Połączenie polega na załadowaniu sterownika JDBC (w tym przypadku *"oracle.jdbc.driver.OracleDriver"*) i zażądaniu od sterownika połączenia. Proces ten może być wykonany następującym kodem:

```
Class.forName("com.mysql.cj.jdbc.Driver");

try{
Connection connection = DriverManager
    .getConnection("jdbc:oracle:thin:@//host:1521/d_b"
", "login", "password")
}
catch(Exception e){
...
}
```

3.2.4. Projekt zabezpieczeń na poziomie aplikacji

Jednym z zabezpieczeń jest zapisywanie zaszyfrowanego hasła do bazy danych. Kolejnym zapewnieniem bezpieczeństwa jest proste logowanie przy pomocy unikalnego loginu i hasła. Jeśli użytkownik o podanych danych istnieje w bazie danych, wtenczas użytkownik uzyskuje dostęp do systemu zgodnie z przypisaną do konta rolą. Jeśli użytkownik wprowadził niepoprawne dane, wtenczas na ekranie zostanie pokazany komunikat, że wprowadzono niepoprawne dane.

Logowanie

Login:

Hasło:

Zaloguj

Rysunek 19 Okno logowania

Logowanie

Login:

Hasło:

Zaloguj

Zalogowano pomyślnie

Rysunek 20 Udane logowanie

Logowanie

Login:

Hasło:

Zaloguj

Wprowadzono błędne dane

Rysunek 21 Nieudane logowanie

4. Implementacja systemu baz danych

4.1. Tworzenie tabel i definiowanie ograniczeń na podstawie modelu

4.1.1. Tworzenie tabel, ograniczeń check oraz kluczy głównych

Tabele zostały stworzone od początku, bez dodatkowego generatora. Ograniczeniom integralności (CONSTRAINT) nadano unikatowe nazwy, w celu ich łatwiej identyfikacji. Do ograniczeń integralności w tym przypadku wliczają się UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK (NOT NULL i inne). Poniżej zostaną przedstawione najciekawsze fragmenty kodu tworzenia tabel, w którym znajdują się ograniczenia integralności.


```

CREATE TABLE BD_2.PERSON(
  ID NUMBER(10) CONSTRAINT PERSON_ID_PK PRIMARY KEY,
  ADDRESS_ID NUMBER(10) NOT NULL,
  FIRSTNAME VARCHAR2(100) NOT NULL,
  SURNAME VARCHAR2(100) NOT NULL,
  E_MAIL VARCHAR2(255) NOT NULL,
  USERNAME VARCHAR2(100) NOT NULL UNIQUE,
  PASSWORD CHAR(255) NOT NULL,
  CREATION_DATE DATE NOT NULL,
  HIRING_DATE DATE,
  EMPLOYEE_NUMBER NUMBER(10) UNIQUE,
  REGULAR_CUSTOMER NUMBER(1),
  CONSTRAINT CHECK_E_MAIL CHECK(E_MAIL LIKE '%_@%._%'),
  CONSTRAINT CHECK_REGULAR_CUSTOMER CHECK(REGULAR_CUSTOMER
  IN (0,1)));

```

Powyżej widać, że tworzona jest tabela PERSON w pakiecie BD_2.

Widać klucz główny (ID), któremu nadano typ NUMBER o rozmiarze 10 oraz nazwę PERSON_ID_PK. Wprowadzono ADDRESS_ID, który nie może przyjmować wartości null. Następnie znajduje się pole EMPLOYEE_NUMBER, które jest unikalne. Wprowadzono również ograniczenie na e-mail, aby nie można było wprowadzić do bazy adresu, który jest niezgodny ze wzorem i np. prawidłowy jest ab@cd.ef, ale @cd.ef, już nie. Można zauważyć również ograniczenie CHECK_REGULAR_CUSTOMER na pole REGULAR_CUSTOMER, które pozwala na wartości 0 oraz 1, gdyż atrybut ten stanowi wartość logiczną: prawda, fałsz.

```

CREATE TABLE BD_2.ABSTRACT_PRODUCT(
...
  PRICE NUMBER(8,2) NOT NULL,
...
  WEIGHT NUMBER(5,3),
  HEIGHT NUMBER(5),
...
  TAX_VALUE NUMBER(3) NOT NULL,
  CONSTRAINT CHECK_PRICE CHECK(PRICE > 0),
  CONSTRAINT CHECK_WEIGHT CHECK(WEIGHT > 0 OR WEIGHT IS
  NULL),
...
  CONSTRAINT CHECK_TAX_VALUE CHECK(TAX_VALUE > 0 AND
  TAX_VALUE <= 100));

```

Pierwszym widocznym powyżej atrybutem jest cena PRICE, która została zapisana jako liczba z dwoma miejscami po przecinku. Zostało na nią założone również ograniczenie, że musi być większa od 0.

Zaimplementowano wagę WEIGHT z trzema miejscami po przecinku, gdyż waga jest przechowywana w kilogramach np. 0,232 kg. Wprowadzono ograniczenie, że jeśli wprowadzana jest waga różna od null, wtenczas nie może być ona mniejsza lub równa 0.

Wprowadzono również atrybut reprezentujący podatek TAX_VALUE, który może przyjmować wartości od 1 do 100.

4.1.2. Klucze obce

Dodawanie kluczy obcych zostało zrealizowane w osobnym skrypcie aby mieć większą swobodę w ich modyfikowaniu oraz aby umożliwić wygenerowanie tabel bez tych kluczy. Zostanie przedstawiony jedynie fragment skryptu do generowania wszystkich kluczy obcych, który jest dostępny poniżej.

```
ALTER TABLE
    BD_2.PERSON
ADD
    CONSTRAINT PERSON_ADDRESS_FK FOREIGN KEY(ADDRESS_ID)
        REFERENCES BD_2.ADDRESS(ID);

ALTER TABLE
    BD_2.PHONE
ADD
    CONSTRAINT PHONE_PERSON_FK FOREIGN KEY(PERSON_ID)
        REFERENCES BD_2.PERSON(ID) ON DELETE CASCADE;
```

W powyższym fragmencie kodu widać modyfikowanie dwóch tabel PERSON i PHONE poprzez dodanie do nich kluczy obcych. Widać również, że dodany do tabeli PHONE klucz obcy posiada właściwość ON DELETE CASCADE i dzięki temu przy usunięciu danej osoby, zostaną automatycznie usunięte jego numery telefonu.

4.2.Implementacja mechanizmów przetwarzania danych

Widoki

Widoki zostały zaimplementowane zgodnie z opisem znajdującym się w punkcie 3.1.4 widoki. Poniżej znajduje się kod do ich generowania.

```
CREATE VIEW BD_2.PRODUCTS_WITH_PRICE_LESS_100 AS
SELECT
    PRODUCER.name Producer,
    P_C.name "Product category",
    A_P.name "Abstract Product",
    A_P.price,
    COLOR.name "CoLoR",
    P_S.quantity,
    P_S.PRODUCTION_DATE
```

```

FROM BD_2.ABSTRACT_PRODUCT A_P
INNER JOIN BD_2.PRODUCT_SPECIMEN P_S ON A_P.Id =
P_S.ABSTRACT_PRODUCT_ID
INNER JOIN BD_2.PRODUCT_CATEGORY P_C ON
A_P.PRODUCT_CATEGORY_ID = P_C.Id
INNER JOIN BD_2.PRODUCER ON A_P.PRODUCER_ID = PRODUCER.Id
INNER JOIN BD_2.COLOR ON P_S.COLOR_ID = COLOR.Id
WHERE A_P.price < 100
ORDER BY A_P.price DESC;

```

```

CREATE VIEW BD_2.ALL_ORDERED_PRODUCTS AS
SELECT

```

```

    O_P.quantity,
    O_P.price_Of_One_Product,
    Color.name "Color name",
    t.finalized,
    t.transaction_Amount,
    t.creation_Date,
    p_s.production_Date,
    a_p.name "Abstract product name",
    Producer.name "Producer name",
    p_c.name "Product category name"
FROM BD_2.ORDERED_PRODUCTS O_P
INNER JOIN BD_2.TRANSACTION_T t ON O_P.Transaction_Id =
t.Id
INNER JOIN BD_2.PRODUCT_SPECIMEN p_s ON
O_P.Product_Specimen_Id = p_s.Id
INNER JOIN BD_2.Color ON p_s.Color_Id = Color.Id
INNER JOIN BD_2.ABSTRACT_PRODUCT a_p ON
p_s.Abstract_Product_Id = a_p.Id
INNER JOIN BD_2.Producer ON a_p.Producer_Id = Producer.Id
INNER JOIN BD_2.PRODUCT_CATEGORY p_c ON
a_p.Product_Category_Id = p_c.Id;

```

```

CREATE VIEW BD_2.PRODUCTS_ON_PROMOTIONS AS
SELECT

```

```

    Color.name,
    Producer.name "Producer",
    a_p.name "Abstract product name",
    a_p.price,
    p.Product_Category_Id,
    p.Abstract_Product_Id,
    p.Product_Specimen_Id,
    p.start_Date,
    p.end_Date,
    p.percentage,
    p_s.quantity,
    p_s.production_Date,
    a_p.weight,

```

```

        a_p.height,
        a_p.width,
        p_c.name AS name4
FROM BD_2.Promotion p
INNER JOIN BD_2.Product_Category p_c ON
p.Product_Category_Id = p_c.Id
INNER JOIN BD_2.Abstract_Product a_p ON
p.Abstract_Product_Id = a_p.Id AND p_c.Id =
a_p.Product_Category_Id
INNER JOIN BD_2.Product_Specimen p_s ON
p.Product_Specimen_Id = p_s.Id AND a_p.Id =
p_s.Abstract_Product_Id
INNER JOIN BD_2.Producer ON a_p.Producer_Id = Producer.Id
INNER JOIN BD_2.Color ON p_s.Color_Id = Color.Id
ORDER BY p.percentage DESC;

```

```

CREATE VIEW BD_2.Best_Selling_Products AS
SELECT
    p_s.production_Date,
    Color.name "Color",
    Producer.name "Producer",
    a_p.name AS "Abstract product name",
    o_p.price_Of_One_Product,
    SUM(o_p.quantity) sold
FROM BD_2.Ordered_Products o_p
INNER JOIN BD_2.Product_Specimen p_s ON p_s.Id =
o_p.Product_Specimen_Id
INNER JOIN BD_2.Abstract_Product a_p ON
p_s.Abstract_Product_Id = a_p.Id
INNER JOIN BD_2.Color ON p_s.Color_Id = Color.Id
INNER JOIN BD_2.Producer ON a_p.Producer_Id = Producer.Id
GROUP BY o_p.quantity, p_s.production_Date, Color.name,
Producer.name, a_p.name, o_p.price_Of_One_Product
ORDER BY o_p.quantity DESC;

```

```

CREATE VIEW BD_2.finalized_Orders AS
SELECT
    t.transaction_Amount,
    t.creation_Date,
    p.firstname,
    p.surname,
    p.Id,
    p.regular_Customer,
    p.e_mail
FROM BD_2.Transaction_t t
INNER JOIN BD_2.Person p ON t.Customer_Id = p.Id AND
t.Employee_Id = p.Id
WHERE p.employee_Number IS NULL;

```

```

CREATE VIEW BD_2.Person_Numbers AS
SELECT
    Phone.phone_Number,
    Person.firstname,
    Person.surname,
    Person.Id,
    Person.regular_Customer,
    Person.employee_Number,
    Person.creation_Date
FROM BD_2.Person
INNER JOIN BD_2.Phone ON Person.Id = Phone.Person_Id;

```

Sekwencje

Sekwencje zostały użyte dla kluczy głównych ze wszystkich tabel, które je posiadają. Kod wszystkich sekwencji jest bardzo podobny i różni się jedynie nazwą i nazwą tabeli, także zostanie przedstawiony tylko jeden przykładowy.

```

CREATE SEQUENCE BD_2.COUNTRY_ID_SEQ
MINVALUE 1
INCREMENT BY 1;

```

W powyższym kodzie początkową wartością sekwencji jest 1 i jest ona inkrementowana o 1.

Indeksy

Indeksy wykorzystano dla atrybutów, które najprawdopodobniej będą najczęściej wykorzystywane.

Poniżej znajduje się indeks, który wskazuje na dwa atrybuty:

```

CREATE INDEX BD_2.transaction_creation_date_done
ON BD_2.Transaction_t (creation_Date, finalized);

CREATE INDEX BD_2.product_quantity
ON BD_2.Product_Specimen (ABSTRACT_Product_Id, quantity);

CREATE INDEX BD_2.product_price_desc
ON BD_2.Abstract_Product (price DESC);

CREATE INDEX BD_2.product_price_asc
ON BD_2.Abstract_Product (price ASC);

```

Indeksy zostały również wykorzystane do wydajnego wyszukiwania posortowanych atrybutów, czego przykładem jest poniższy indeks, który został stworzony dla malejącej (DESC) daty rozpoczęcia promocji.

```

CREATE INDEX BD_2.promotion_start_date
ON BD_2.PROMOTION (start_date DESC);

CREATE INDEX BD_2.promotion_end_date
ON BD_2.PROMOTION (end_date ASC);

CREATE INDEX BD_2.promotion_perc_end_start_date
ON BD_2.PROMOTION (percentage, end_Date, start_Date);

CREATE INDEX BD_2.product_name
ON BD_2.Abstract_Product (name);

CREATE INDEX BD_2.person_regular_Customer
ON BD_2.Person(regular_Customer);

CREATE INDEX BD_2.person_employee_Number
ON BD_2.Person (employee_Number);

CREATE INDEX BD_2.product_Producent_Id
ON BD_2.Abstract_Product (Producer_Id);

CREATE INDEX BD_2.product_production_Date
ON BD_2.Product_Specimen (production_Date);

CREATE INDEX BD_2.ordered_prod_Prod_Specimen_Id
ON BD_2.Ordered_Products (Product_Specimen_Id);

CREATE INDEX BD_2.loc_street_name_number_flat
ON BD_2.Address(street_Name, street_Number, flat_Number);

CREATE INDEX BD_2.phone__phone_number
ON BD_2.PHONE(PHONE_NUMBER);

```

Triggery

Triggery zostały użyte przede wszystkim do generowania kluczy podstawowych na podstawie sekwencji. Kod tych triggerów jest bardzo podobny, dlatego zostanie przedstawiony tylko jeden przykład:

```

CREATE OR REPLACE TRIGGER BD_2.COUNTRY_ID_T
BEFORE INSERT ON BD_2.COUNTRY
FOR EACH ROW
    WHEN (NEW.Id is NULL)
BEGIN
    :NEW.Id := COUNTRY_ID_SEQ.NEXTVAL;
END;

```

Fragment kodu między BEGIN i END wstawia następną wartość sekwencji COUNTRY_ID_SEQ do klucza głównego Id nowo wstawianego wiersza.

Kolejne triggerzy w skrypcie muszą być oddzielone znakiem „/”.

Kolejnym zastosowaniem triggerów jest aktualizowanie liczby produktów po dokonanej transakcji. Po wstawieniu zamówionego przedmiotu z danego zamówienia, zmniejszana jest liczba produktu o liczbę zamówionych sztuk tego produktu. Kod triggera znajduje się poniżej:

```
CREATE OR REPLACE TRIGGER UPDATE_PRODUCT
  AFTER INSERT ON BD_2.ORDERED_PRODUCTS
  FOR EACH ROW
BEGIN
  UPDATE BD_2.PRODUCT_SPECIMEN p
  SET p.QUANTITY = p.QUANTITY - :new.QUANTITY;
END;
```

Kolejny trigger, tym razem nie pozwalający stworzyć promocji w sytuacji, kiedy 3 klucze obce miałyby być równe null:

```
CREATE OR REPLACE TRIGGER INSERT_PROMOTION
  BEFORE INSERT ON BD_2.PROMOTION
  FOR EACH ROW
BEGIN
  IF (:new.PRODUCT_CATEGORY_ID IS NULL AND
      :new.ABSTRACT_PRODUCT_ID IS NULL AND
      :new.PRODUCT_SPECIMEN_ID IS NULL)
  THEN
    RAISE_APPLICATION_ERROR( -20000, 'Promocja
musi mieć kategorię' );
  END IF;
END;
```

Procedury składowane

Procedura wykonująca utworzenie transakcji w stanie początkowym (*finalized=0*):

```
CREATE OR REPLACE PROCEDURE BD_2.INSERT_TRANSACTION
  (creation_Date DATE,
   Customer_Id NUMBER, Employee_Id NUMBER, Address_Id
NUMBER, transaction_Amount NUMBER)
AS
BEGIN
  INSERT INTO BD_2.TRANSACTION_t (Address_Id,
  Customer_Id, Employee_Id, creation_Date, finalized,
  transaction_Amount)
  VALUES(Address_Id, Customer_Id, Employee_Id,
  creation_Date, 0, transaction_Amount);
END;
```

Procedura obliczająca liczbę różnych produktów z danego zamówienia:

```
CREATE OR REPLACE PROCEDURE
BD_2.NUMBER_OF_ORDERED_PRODUCTS (Transaction_Id NUMBER,
sum_Of_products OUT NUMBER)
AS
BEGIN
    SELECT COUNT(*) "Sum of different products"
    INTO sum_Of_products
    FROM BD_2.Ordered_Products o
    WHERE o.Transaction_Id = Transaction_Id;
END;
```

Procedura wyznaczająca liczbę dni do końca promocji:

```
CREATE OR REPLACE PROCEDURE BD_2.PROMOTION_DAYS_LEFT
(PROMOTION_ID NUMBER, DAYS_LEFT OUT NUMBER)
AS
BEGIN
    SELECT trunc(start_date,'day') -
    trunc(end_date,'day') "Days left"
    INTO DAYS_LEFT
    FROM BD_2.PROMOTION p
    WHERE p.ID = PROMOTION_ID;
END;
```

Procedura dodająca nowy numer telefonu danej osoby:

```
CREATE OR REPLACE PROCEDURE BD_2.INSERT_PHONE_NUMBER
(PERSON_ID NUMBER,
PHONE_NUMBER VARCHAR, IS_EMPLOYEE_PHONE NUMBER)
AS
BEGIN
    INSERT INTO BD_2.PHONE(PERSON_ID, PHONE_NUMBER,
IS_EMPLOYEE_PHONE)
VALUES(PERSON_ID, PHONE_NUMBER, IS_EMPLOYEE_PHONE);
END;
```

Procedura dodająca nowego pracownika.

```
CREATE OR REPLACE PROCEDURE BD_2.INSERT_EMPLOYEE
(
    FIRSTNAME VARCHAR2, SURNAME VARCHAR2, E_MAIL
VARCHAR2, USERNAME VARCHAR2, PASSWORD CHAR,
HIRING_DATE DATE, EMPLOYEE_NUMBER NUMBER
)
AS
BEGIN
```



```

        INSERT INTO BD_2.PERSON(FIRSTNAME, SURNAME, E_MAIL,
        USERNAME, PASSWORD, CREATION_DATE,
        HIRING_DATE, EMPLOYEE_NUMBER)
        VALUES(FIRSTNAME, SURNAME, E_MAIL, USERNAME, PASSWORD,
        SYSDATE, HIRING_DATE, EMPLOYEE_NUMBER);
    END;

```

Procedura dodająca nowy abstrakcyjny produkt:

```

CREATE OR REPLACE PROCEDURE BD_2.INSERT_ABSTRACT_PRODUCT
(
    PRODUCT_CATEGORY_ID NUMBER, PRODUCER_ID NUMBER,
    IMAGE BLOB,
    PRICE NUMBER, NAME VARCHAR, DESCRIPTION VARCHAR,
    WEIGHT NUMBER, HEIGHT
    NUMBER, WIDTH NUMBER, TAX_VALUE NUMBER
)
AS
BEGIN
    INSERT INTO BD_2.ABSTRACT_PRODUCT(PRODUCT_CATEGORY_ID,
    PRODUCER_ID,
    IMAGE, PRICE, NAME, DESCRIPTION, WEIGHT, HEIGHT,
    WIDTH, TAX_VALUE)
    VALUES(PRODUCT_CATEGORY_ID, PRODUCER_ID,
    IMAGE, PRICE, NAME, DESCRIPTION, WEIGHT, HEIGHT,
    WIDTH, TAX_VALUE);
END;

```

4.3.Implementacja uprawnień i innych zabezpieczeń

Założono, że potrzebne będą 4 role:

- administrator,
- pracownik,
- klient,
- niezalogowany użytkownik

Kod:

```

CREATE ROLE ADMINISTRATOR;
CREATE ROLE EMPLOYEE;
CREATE ROLE CUSTOMER;
CREATE ROLE NOT_LOGGED_IN;

```

Dla utworzonych ról dodano odpowiednie uprawnienia w zależności od tabel. Poniżej znajduje się fragment kodu.

```
GRANT SELECT, INSERT, UPDATE ON BD_2.PERSON  
TO EMPLOYEE, CUSTOMER;
```

Powyższy kod wprowadza możliwość użytkownikom o rolach EMPLOYEE i CUSTOMER na wykonywanie zapytań SELECT, INSERT, UPDATE na tabeli PERSON.

Po utworzeniu odpowiednich ról należało utworzyć użytkowników, do których przypisane są odpowiednie role. Przyjęto, że jedno konto o danej roli będzie wykorzystywane przez wszystkich użytkowników, którzy mają przypisaną taką rolę, np. wszyscy zarejestrowani klienci korzystają z jednego konta użytkownika po stronie bazy danych o roli CUSTOMER, ale za to, aby dany klient miał dostęp do systemu, musi się zalogować na swoje konto po stronie aplikacji. Poniżej znajduje się kod, który odpowiada za stworzenie wszystkich wymaganych użytkowników o przypisanych do nich przykładowych haseł.

```
CREATE USER ADMINISTRATOR_U IDENTIFIED BY a_pass;  
GRANT ADMINISTRATOR TO ADMINISTRATOR_U;
```

```
CREATE USER EMPLOYEE_U IDENTIFIED BY e_pass;  
GRANT EMPLOYEE TO EMPLOYEE_U;
```

```
CREATE USER CUSTOMER_U IDENTIFIED BY c_pass;  
GRANT CUSTOMER TO CUSTOMER_U;
```

```
CREATE USER NOT_LOGGED_IN_U IDENTIFIED BY n_l_pass;  
GRANT NOT_LOGGED_IN TO NOT_LOGGED_IN_U;
```

4.4. Testowanie bazy danych na przykładowych danych

4.4.1. Generowanie danych testowych

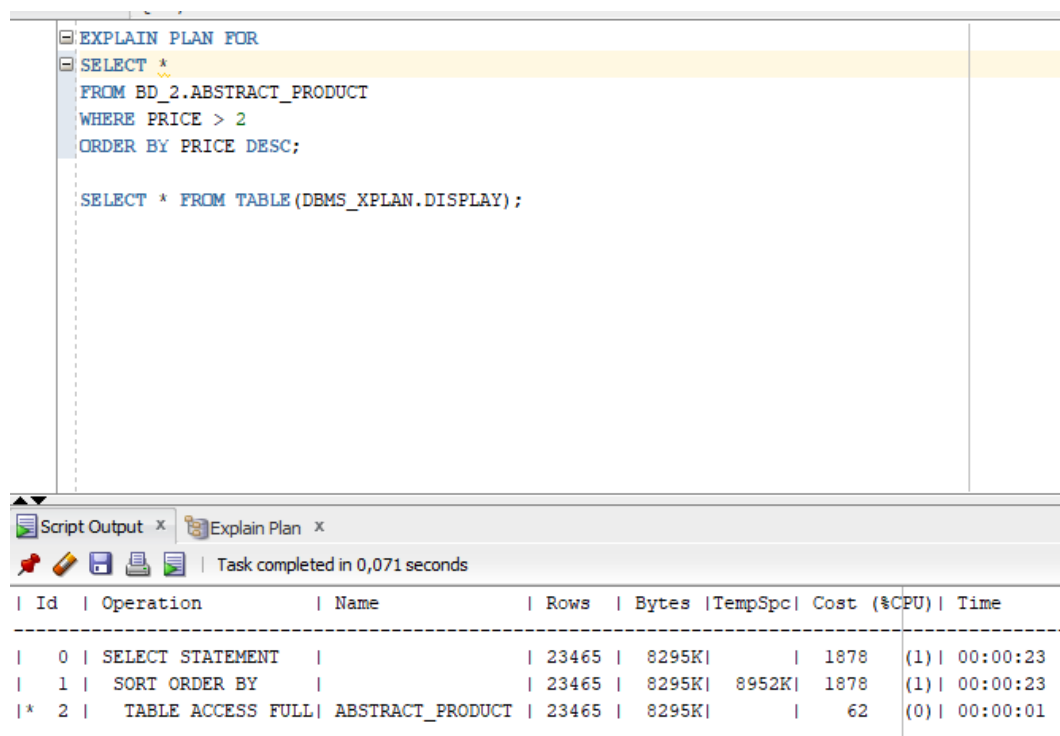
Dane zostały wygenerowane przy użyciu strony <https://www.mockaroo.com/>. Aby wygenerować dane dla danej tabeli trzeba było zbudować jej model na stronie a następnie wybrać liczbę generowanych wierszy. Dane testowe były generowane w postaci kodu SQL – INSERT INTO(...) VALUES(...). Przykładowe wstawienie wiersza znajduje się poniżej.

```
INSERT INTO BD_2.PHONE  
(PERSON_ID, phone_number, is_employee_phone)  
VALUES (2400, '1-801-274-6851', 1);
```

Powyższy kod odpowiada za wstawienie nowego numeru telefonu służbowego pracownika.

4.4.2. Testy wydajnościowe

W ramach testów wydajnościowych przetestowano 2 najważniejsze tabele: ABSTRACT_PRODUCT i PRODUCT_SPECIMEN. Zmierzono czas wykonywania zapytań SELECT w zależności od liczby wierszy w tabeli oraz sprawdzono wpływ indeksów. Czas zapytań został zmierzony przy użyciu EXPLAIN PLAN w środowisku Oracle SQL Developer.



The screenshot shows the Oracle SQL Developer interface. The top pane contains the following SQL code:

```
EXPLAIN PLAN FOR
SELECT *
FROM BD_2.ABSTRACT_PRODUCT
WHERE PRICE > 2
ORDER BY PRICE DESC;

SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

The bottom pane shows the execution plan output, titled "Explain Plan". It indicates the task completed in 0,071 seconds. The execution plan is as follows:

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		23465	8295K		1878	(1) 00:00:23
1	SORT ORDER BY		23465	8295K	8952K	1878	(1) 00:00:23
* 2	TABLE ACCESS FULL	ABSTRACT_PRODUCT	23465	8295K		62	(0) 00:00:01

Rysunek 22 Pomiar czasu dla przykładowego zapytania wraz z wynikiem czasowym

Powyższy kod przedstawia przykładowe mierzenie czasu dla zapytania SELECT dla tabeli ABSTRACT_PRODUCT. Widać, że łączny czas wykonania zapytania wynosi 47s, a łączny koszt pamięciowy to 3818B.

Tabela ABSTRACT_PRODUCT

Zapytanie:

```
SELECT *
FROM BD_2.ABSTRACT_PRODUCT
WHERE PRICE > 2
ORDER BY PRICE DESC;
```

Instancja 10000:

Bez indeksów: 50s

Tabela 2 Wynik czasowy 1

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		13786	4873K		1680	(1) 00:00:21
1	SORT ORDER BY		13786	4873K	5264K	1680	(1) 00:00:21
* 2	TABLE ACCESS FULL	ABSTRACT_PRODUCT	13786	4873K		612	(1) 00:00:08

Z indeksami: 3s

Tabela 3 Wynik czasowy 2

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		13786	4873K	7	(0) 00:00:01
1	TABLE ACCESS BY INDEX ROWID	ABSTRACT_PRODUCT	13786	4873K	7	(0) 00:00:01
* 2	INDEX RANGE SCAN	PRODUCT_PRICE_DESC	6		2	(0) 00:00:01

Instancja 50000:

Bez indeksów: 2m 4s

Tabela 4 Wynik czasowy 3.

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		54393	18M		4818	(1) 00:00:58
1	SORT ORDER BY		54393	18M	20M	4818	(1) 00:00:58
* 2	TABLE ACCESS FULL	ABSTRACT_PRODUCT	54393	18M		612	(1) 00:00:08

Z indeksami: 3s

Tabela 5 Wynik czasowy 4

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		54393	18M	26	(0) 00:00:01
1	TABLE ACCESS BY INDEX ROWID	ABSTRACT_PRODUCT	54393	18M	26	(0) 00:00:01
* 2	INDEX RANGE SCAN	PRODUCT_PRICE_DESC	24		3	(0) 00:00:01

Instancja 100000:

Bez indeksów: 3m 23s

Tabela 6 Wynik czasowy 5

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		105K	36M		8733	(1) 00:01:45
1	SORT ORDER BY		105K	36M	39M	8733	(1) 00:01:45
* 2	TABLE ACCESS FULL	ABSTRACT_PRODUCT	105K	36M		613	(1) 00:00:08

Z indeksami: 3s

Tabela 7 Wynik czasowy 6

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		105K	36M	49	(0) 00:00:01
1	TABLE ACCESS BY INDEX ROWID	ABSTRACT_PRODUCT	105K	36M	49	(0) 00:00:01
* 2	INDEX RANGE SCAN	PRODUCT_PRICE_DESC	47		4	(0) 00:00:01

Instancja 150000:

Bez indeksów: 5m 12s

Tabela 8 Wynik czasowy 7

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		155K	53M		12643	(1) 00:02:32
1	SORT ORDER BY		155K	53M	57M	12643	(1) 00:02:32
* 2	TABLE ACCESS FULL	ABSTRACT_PRODUCT	155K	53M		613	(1) 00:00:08

Z indeksami: 3s

Tabela 9 Wynik czasowy 8

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		155K	53M	73	(0) 00:00:01
1	TABLE ACCESS BY INDEX ROWID	ABSTRACT_PRODUCT	155K	53M	73	(0) 00:00:01
* 2	INDEX RANGE SCAN	PRODUCT_PRICE_DESC	70		5	(0) 00:00:01

Instancja 200000:

Tabela 10 Wynik czasowy 9

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		216K	74M		17324	(1) 00:03:28
1	SORT ORDER BY		216K	74M	80M	17324	(1) 00:03:28
* 2	TABLE ACCESS FULL	ABSTRACT_PRODUCT	216K	74M		614	(1) 00:00:08

Bez indeksów: 7m 4s

Tabela 11 Wynik czasowy 10

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		216K	74M	96	(0) 00:00:02
1	TABLE ACCESS BY INDEX ROWID	ABSTRACT_PRODUCT	216K	74M	96	(0) 00:00:02
* 2	INDEX RANGE SCAN	PRODUCT_PRICE_DESC	97		6	(0) 00:00:01

Z indeksami: 5s

Widać sporą różnicę w czasie wykonania zapytania bez indeksów i z. Można zauważyć również sporą różnicę w koszcie wykonania zapytania na korzyść wykonania z indeksami. Im większa instancja tym ta różnica jest większa.

Instancja 250000:

Bez indeksów: 8m 33s

Tabela 12 Wynik czasowy 11

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		258K	89M		20882	(1) 00:04:11
1	SORT ORDER BY		258K	89M	96M	20882	(1) 00:04:11
* 2	TABLE ACCESS FULL	ABSTRACT_PRODUCT	258K	89M		878	(1) 00:00:11

Z indeksami: 5s

Tabela 13 Wynik czasowy 12

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		258K	89M	121	(0) 00:00:02
1	TABLE ACCESS BY INDEX ROWID	ABSTRACT_PRODUCT	258K	89M	121	(0) 00:00:02
* 2	INDEX RANGE SCAN	PRODUCT_PRICE_DESC	117		8	(0) 00:00:01

Instancja 300000:
Bez indeksów: 14m 11s

Tabela 14 Wynik czasowy 13

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		441K	152M		34997	(1) 00:07:00
1	SORT ORDER BY		441K	152M	164M	34997	(1) 00:07:00
* 2	TABLE ACCESS FULL	ABSTRACT_PRODUCT	441K	152M		880	(1) 00:00:11

Z indeksami: 5s

Tabela 15 Wynik czasowy 14

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		297K	102M	144	(0) 00:00:02
1	TABLE ACCESS BY INDEX ROWID	ABSTRACT_PRODUCT	297K	102M	144	(0) 00:00:02
* 2	INDEX RANGE SCAN	PRODUCT_PRICE_DESC	134		9	(0) 00:00:01

Tabela PRODUCT SPECIMEN

Zapytanie:

```
SELECT *
FROM BD_2.PRODUCT_SPECIMEN
WHERE ABSTRACT_PRODUCT_ID = 504700 AND QUANTITY > 2;
```

Instancja: 100000
Bez indeksów:

Tabela 16 Wynik czasowy 15

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		36	79380	1020	(1) 00:00:13
* 1	TABLE ACCESS FULL	PRODUCT_SPECIMEN	36	79380	1020	(1) 00:00:13

Z indeksami:

Tabela 17 Wynik czasowy 16

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		36	79380	6	(0) 00:00:01
1	TABLE ACCESS BY INDEX ROWID	PRODUCT_SPECIMEN	36	79380	6	(0) 00:00:01
* 2	INDEX RANGE SCAN	PRODUCT_QUANTITY	3		2	(0) 00:00:01

Instancja: 200000

Bez indeksów:

Tabela 18 Wynik czasowy 17

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		104	223K	1022	(1) 00:00:13
* 1	TABLE ACCESS FULL	PRODUCT_SPECIMEN	104	223K	1022	(1) 00:00:13

Z indeksami:

Tabela 19 Wynik czasowy 18

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		104	223K	11	(0) 00:00:01
1	TABLE ACCESS BY INDEX ROWID	PRODUCT_SPECIMEN	104	223K	11	(0) 00:00:01
* 2	INDEX RANGE SCAN	PRODUCT_QUANTITY	7		3	(0) 00:00:01

Instancja: 300000

Bez indeksów:

Tabela 20 Wynik czasowy 19

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		132	284K	1022	(1) 00:00:13
* 1	TABLE ACCESS FULL	PRODUCT_SPECIMEN	132	284K	1022	(1) 00:00:13

Z indeksami:

Tabela 21 Wynik czasowy 20

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		132	284K	14	(0) 00:00:01
1	TABLE ACCESS BY INDEX ROWID	PRODUCT_SPECIMEN	132	284K	14	(0) 00:00:01
* 2	INDEX RANGE SCAN	PRODUCT_QUANTITY	9		3	(0) 00:00:01

Instancja 400000:

Bez indeksów:

Tabela 22 Wynik czasowy 21

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		208	447K	1024	(1) 00:00:13
* 1	TABLE ACCESS FULL	PRODUCT_SPECIMEN	208	447K	1024	(1) 00:00:13

Tabela 23 Wynik czasowy 22

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		208	447K	18	(0) 00:00:01
1	TABLE ACCESS BY INDEX ROWID	PRODUCT_SPECIMEN	208	447K	18	(0) 00:00:01
* 2	INDEX RANGE SCAN	PRODUCT_QUANTITY	15		3	(0) 00:00:01

Z indeksami:

Instancja 500000:

Bez indeksów:

Tabela 24 Wynik czasowy 23

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		260	559K	1025	(1) 00:00:13
* 1	TABLE ACCESS FULL	PRODUCT_SPECIMEN	260	559K	1025	(1) 00:00:13

Z indeksami:

Tabela 25 Wynik czasowy 24

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		260	559K	21	(0) 00:00:01
1	TABLE ACCESS BY INDEX ROWID	PRODUCT_SPECIMEN	260	559K	21	(0) 00:00:01
* 2	INDEX RANGE SCAN	PRODUCT_QUANTITY	19		3	(0) 00:00:01

Widać sporą różnicę w czasie wykonania zapytania bez indeksów i z. Można zauważyć również sporą różnicę w koszcie wykonania zapytania na korzyść wykonania z indeksami. Im większa instancja tym ta różnica jest większa.

4.4.3. Testy na tak i na nie

```
INSERT INTO BD_2.PERSON(ADDRESS_ID, FIRSTNAME, SURNAME, E_MAIL,
    PASSWORD, CREATION_DATE,
    HIRING_DATE, EMPLOYEE_NUMBER, USERNAME)
VALUES(3, 'aa2aa2', 'aaaa22', null, 'pass23', sysdate, sysdate,
5, 'aaa22');
```

Operacja się nie uda, gdyż wprowadzono e-mail jako null, a atrybut ten jest NOT NULL.

```
Error starting at line : 2 in command -
INSERT INTO BD_2.PERSON(ADDRESS_ID, FIRSTNAME, SURNAME, E_MAIL, PASSWORD, CREATION_DATE,
    HIRING_DATE, EMPLOYEE_NUMBER, USERNAME)
VALUES(3, 'aa2aa2', 'aaaa22', null, 'pass23', sysdate, sysdate, 5, 'aaa22')
Error report -
ORA-01400: cannot insert NULL into ("BD_2"."PERSON"."E_MAIL")
```

Rysunek 23 Wynik zapytania 1

```
INSERT INTO BD_2.PERSON(ADDRESS_ID, FIRSTNAME, SURNAME, E_MAIL,
    PASSWORD, CREATION_DATE,
    HIRING_DATE, EMPLOYEE_NUMBER, USERNAME)
VALUES(3, 'aa2aa2', 'aaaa22', 'a243@a.a', 'pass23', sysdate,
sysdate, 5, 'aaa22');
```

Operacja się nie uda, gdyż wprowadzono numer pracownika, który znajduje się już w bazie -> UNIQUE

```

Error starting at line : 4 in command -
INSERT INTO BD_2.PERSON(ADDRESS_ID, FIRSTNAME, SURNAME, E_MAIL, PASSWORD, CREATION_DATE,
      HIRING_DATE, EMPLOYEE_NUMBER, USERNAME)
VALUES(3, 'aa2aa2', 'aaaa22', 'a243@a.a', 'pass23', sysdate, sysdate, 5, 'aaa22')
Error report -
ORA-00001: unique constraint (BD_2.SYS_C0016701) violated

```

Rysunek 24 Wynik zapytania 2

```

INSERT INTO BD_2.PERSON(ADDRESS_ID, FIRSTNAME, SURNAME, E_MAIL,
      PASSWORD, CREATION_DATE,
      HIRING_DATE, EMPLOYEE_NUMBER, USERNAME)
VALUES(3, 'aa2aa2', 'aaaa22', 'a243.a', 'pass23', sysdate,
      sysdate, 4, 'aaa22');

```

Operacja się nie uda, gdyż wprowadzono niepoprawny e-mail.

```

Error starting at line : 4 in command -
INSERT INTO BD_2.PERSON(ADDRESS_ID, FIRSTNAME, SURNAME, E_MAIL, PASSWORD, CREATION_DATE,
      HIRING_DATE, EMPLOYEE_NUMBER, USERNAME)
VALUES(3, 'aa2aa2', 'aaaa22', 'a243.a', 'pass23', sysdate, sysdate, 4, 'aaa22')
Error report -
ORA-02290: check constraint (BD_2.CHECK_E_MAIL) violated

```

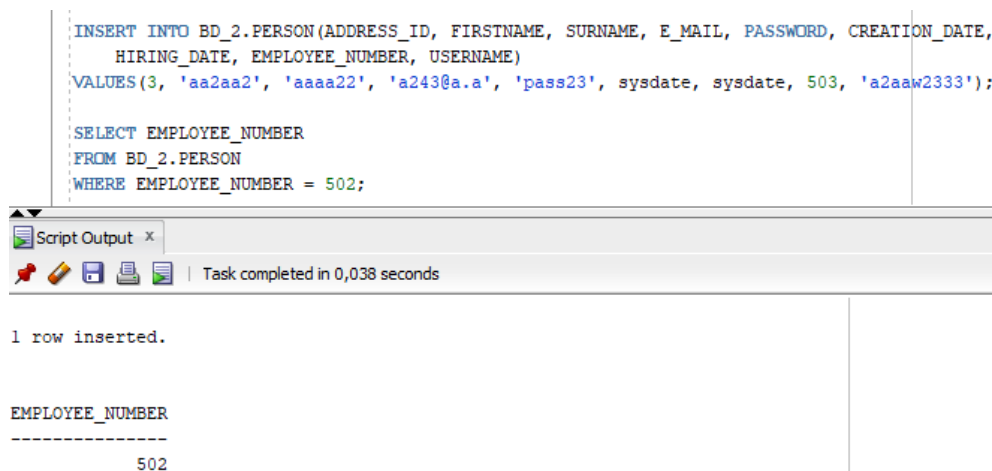
Rysunek 25 Wynik zapytania 3

```

INSERT INTO BD_2.PERSON(ADDRESS_ID, FIRSTNAME, SURNAME,
      E_MAIL, PASSWORD, CREATION_DATE,
      HIRING_DATE, EMPLOYEE_NUMBER, USERNAME)
VALUES(3, 'aa2aa2', 'aaaa22', 'a243@a.a', 'pass23', sysdate,
      sysdate, 5, 'aaa22');

```

Operacja się uda.



The screenshot shows the SQL Developer interface. The top pane displays the following SQL commands:

```

INSERT INTO BD_2.PERSON(ADDRESS_ID, FIRSTNAME, SURNAME, E_MAIL, PASSWORD, CREATION_DATE,
      HIRING_DATE, EMPLOYEE_NUMBER, USERNAME)
VALUES(3, 'aa2aa2', 'aaaa22', 'a243@a.a', 'pass23', sysdate, sysdate, 503, 'a2aaw2333');

SELECT EMPLOYEE_NUMBER
FROM BD_2.PERSON
WHERE EMPLOYEE_NUMBER = 502;

```

The bottom pane shows the 'Script Output' window with the message: 'Task completed in 0,038 seconds'. Below this, the query result is displayed:

```

1 row inserted.

EMPLOYEE_NUMBER
-----
502

```

Rysunek 26 Wynik zapytania 4

```
INSERT INTO BD_2.CITY(COUNTRY_ID, NAME, STATE_V)
VALUES(0, 'POZNAŃ', 'STATE');
```

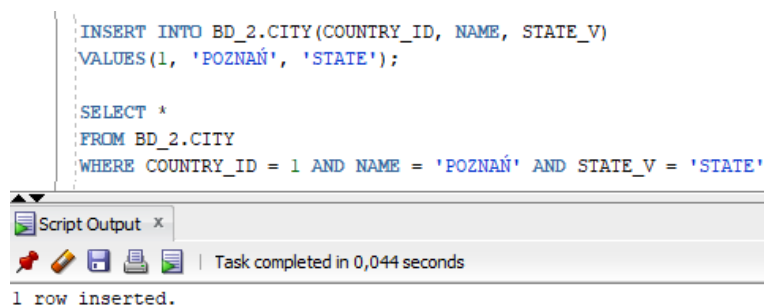
Nie, gdyż jest wprowadzony niepoprawny klucz obcy COUNTRY_ID -> 0

```
Error starting at line : 4 in command -
INSERT INTO BD_2.CITY(COUNTRY_ID, NAME, STATE_V)
VALUES(0, 'POZNAŃ', 'STATE')
Error report -
ORA-02291: integrity constraint (BD_2.CITY_COUNTRY_FK) violated - parent key not found
```

Rysunek 27 Wynik zapytania 5

```
INSERT INTO BD_2.CITY(COUNTRY_ID, NAME, STATE_V)
VALUES(1, 'POZNAŃ', 'STATE');
```

Operacja się uda.



ID	COUNTRY_ID	NAME	STATE_V
102	1	POZNAŃ	STATE

Rysunek 28 Wynik zapytania 6

```
INSERT INTO BD_2.ROLES_T(PERSON_ID, ROLE_ID)
VALUES(1, 0);
```

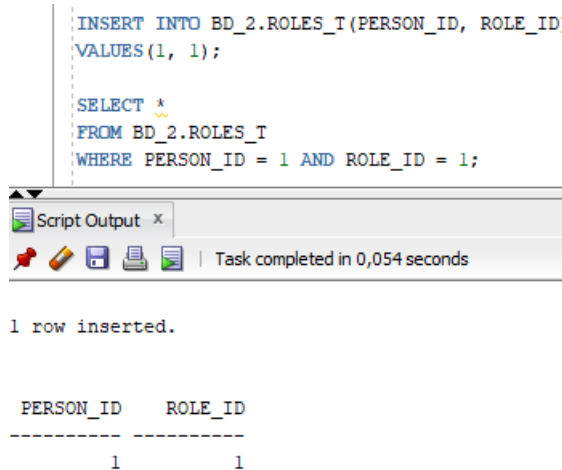
Operacja się nie uda, gdyż wprowadzono niepoprawny klucz obcy ROLE_ID.

```
Error starting at line : 4 in command -
INSERT INTO BD_2.ROLES_T(PERSON_ID, ROLE_ID)
VALUES(1, 0)
Error report -
ORA-02291: integrity constraint (BD_2.ROLES_T_ROLE_FK) violated - parent key not found
```

Rysunek 29 Wynik zapytania 7

```
INSERT INTO BD_2.ROLES_T(PERSON_ID, ROLE_ID)
VALUES(1, 1);
```

Operacja się uda.



Rysunek 30 Wynik zapytania 8

4.4.4. Testy uprawnień

Administrator (ADMINISTRATOR_U):

```
SQL> SELECT COUNT(*) FROM BD_2.PERSON;

COUNT(*)
-----
2304
```

Rysunek 31 Wynik testu uprawnień 1

```
SQL> INSERT INTO BD_2.PERSON(ADDRESS_ID, FIRSTNAME, SURNAME, E_MAIL, PASSWORD, CREATION_DATE, HIRING_DATE, EMPLOYEE_NUMBER,
, USERNAME) VALUES(3, 'aa2aa2', 'aaaa22', 'a243@a.a', 'pass23', sysdate, sysdate, 520, 'aa3123a22');

1 row created.

SQL>
SQL>
SQL>
```

Rysunek 32 Wynik testu uprawnień 2

```
SQL> SELECT COUNT(*) FROM BD_2.TRANSACTION_T;

COUNT(*)
-----
748
```

Rysunek 33 Wynik testu uprawnień 3

```
SQL> insert into BD_2.TRANSACTION_T (ADDRESS_ID, CUSTOMER_ID, EMPLOYEE_ID, CREATION_DATE, FINALIZED, TRANSACTION_AMOUNT)
values (861, 932, 2240, '2020-12-15', 1, 6866);
insert into BD_2.TRANSACTION_T (ADDRESS_ID, CUSTOMER_ID, EMPLOYEE_ID, CREATION_DATE, FINALIZED, TRANSACTION_AMOUNT) values
(861, 932, 2240, '2020-12-15', 1, 6866)
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

Rysunek 34 Wynik testu uprawnień 4

Pracownik (EMPLOYEE_U):

```
SQL> SELECT COUNT(*) FROM BD_2.PROMOTION;

COUNT(*)
-----
178
```

Rysunek 35 Wynik testu uprawnień 5

```
SQL> insert into BD_2.PROMOTION (PRODUCT_CATEGORY_ID, PERCENTAGE, START_DATE, END_DATE) values (47, 41, '2021-05-09', '2021-05-29');
1 row created.
```

Rysunek 36 Wynik testu uprawnień 6

```
SQL> SELECT COUNT(*) FROM BD_2.ORDERED_PRODUCTS;

COUNT(*)
-----
255
```

Rysunek 37 Wynik testu uprawnień 7

```
SQL> insert into BD_2.ORDERED_PRODUCTS (TRANSACTION_ID, PRODUCT_SPECIMEN_ID, quantity, price_of_one_product) values (50, 247, 623, 720);
insert into BD_2.ORDERED_PRODUCTS (TRANSACTION_ID, PRODUCT_SPECIMEN_ID, quantity, price_of_one_product) values (50, 247, 623, 720)
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

Rysunek 43 Wynik testu uprawnień 8.

Klient (CUSTOMER_U):

```
SQL> SELECT COUNT(*) FROM BD_2.TRANSACTION_T;

COUNT(*)
-----
749
```

Rysunek 38 Wynik testu uprawnień 9

```
SQL> insert into BD_2.TRANSACTION_T (ADDRESS_ID, CUSTOMER_ID, EMPLOYEE_ID, CREATION_DATE, FINALIZED, TRANSACTION_AMOUNT)
values (861, 932, 2240, '2020-12-15', 1, 6866);
insert into BD_2.TRANSACTION_T (ADDRESS_ID, CUSTOMER_ID, EMPLOYEE_ID, CREATION_DATE, FINALIZED, TRANSACTION_AMOUNT) valu
es (861, 932, 2240, '2020-12-15', 1, 6866)
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

Rysunek 39 Wynik testu uprawnień 10

```
SQL> SELECT COUNT(*) FROM BD_2.PHONE;

COUNT(*)
-----
2300
```

Rysunek 40 Wynik testu uprawnień 11

```
SQL> INSERT INTO BD_2.PHONE (PERSON_ID,phone_number,is_employee_phone)
2 VALUES (2406,'1-362-845-5557',1);

1 row created.
```

Rysunek 41 Wynik testu uprawnień 12

Niezałogowany (NOT_LOGGED_IN_U):

```
SQL> SELECT COUNT(*) FROM BD_2.PERSON;
SELECT COUNT(*) FROM BD_2.PERSON
*
ERROR at line 1:
ORA-00942: table or view does not exist
```

Rysunek 42 Wynik testu uprawnień 13

```
SQL> SELECT COUNT(*) FROM BD_2.ABSTRACT_PRODUCT;

COUNT(*)
-----
467
```

Rysunek 43 Wynik testu uprawnień 14

```
SQL> insert into BD_2.ABSTRACT_PRODUCT (Product_Category_Id, Producer_id, price, name, description, weight, height, width, tax_Value) values (14, 20, 833.82, 'Cheese - Brie, Triple Creme', 'Yogurt - Cherry, 175 Gr', 56, 23, 57, 48);
insert into BD_2.ABSTRACT_PRODUCT (Product_Category_Id, Producer_id, price, name, description, weight, height, width, tax_Value) values (14, 20, 833.82, 'Cheese - Brie, Triple Creme', 'Yogurt - Cherry, 175 Gr', 56, 23, 57, 48)
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

Rysunek 44 Wynik testu uprawnień 15

```
SQL> SELECT COUNT(*) FROM BD_2.PRODUCT_SPECIMEN;

COUNT(*)
-----
344
```

Rysunek 45 Wynik testu uprawnień 16

```
SQL> insert into BD_2.PRODUCT_SPECIMEN (ABSTRACT_PRODUCT_ID, COLOR_ID, PRODUCTION_DATE, description, serial_number, quantity) values (168, 9, '2021-10-21', 'Mousse - Passion Fruit', '318', 99);
insert into BD_2.PRODUCT_SPECIMEN (ABSTRACT_PRODUCT_ID, COLOR_ID, PRODUCTION_DATE, description, serial_number, quantity) values (168, 9, '2021-10-21', 'Mousse - Passion Fruit', '318', 99)
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

Rysunek 46 Wynik testu uprawnień 17

5. Implementacja i testy aplikacji

5.1. Instalacja i konfigurowanie systemu

5.1.1. Perspektywa użytkownika

Wymagania programowe:

- Java (JDK 15.0.2)

```
java version "15.0.2" 2021-01-19
Java(TM) SE Runtime Environment (build 15.0.2+7-27)
Java HotSpot(TM) 64-Bit Server VM (build 15.0.2+7-27, mixed mode, sharing)
```

W klasycznym podejściu, w którym aplikacja byłaby dostępna publicznie, użytkownik mógłby pobrać aplikację ze strony internetowej i jej od razu używać. W tym projekcie baza jest postawiona lokalnie, zatem użytkownik mógłby uruchomić aplikację jedynie na komputerze programisty lub wdrożeniowca. Jeśli jest postawiona odpowiednia baza danych, to wystarczy mieć zainstalowaną Javę i zostaje już tylko uruchomić aplikację, gdyż jest ona w wersji wykonywalnej (jar).

5.1.2. Perspektywa programisty

Wymagania programowe:

- Java (JDK 15.0.2)

```
java version "15.0.2" 2021-01-19
Java(TM) SE Runtime Environment (build 15.0.2+7-27)
Java HotSpot(TM) 64-Bit Server VM (build 15.0.2+7-27, mixed mode, sharing)
```

- Maven (3.6.3)
- Oracle Database (11g Express Edition)

```
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
PL/SQL Release 11.2.0.2.0 - Production
CORE 11.2.0.2.0 Production
TNS for 64-bit Windows: Version 11.2.0.2.0 - Production
NLSRTL Version 11.2.0.2.0 - Production
```

Należy postawić bazę poprzez uruchomienie wszystkich skryptów z punktu 4-4.3. Do zarządzania projektem w Javie jest używany Maven, zatem aby uruchomić aplikację należy pobrać zależności i wygenerować plik .jar poprzez `mvn clean install`, a następnie wystarczy uruchomić aplikację poprzez `java -jar target/com.BD_2-0.0.1-SNAPSHOT-jar-with-dependencies.jar`.

Do testowania służy `mvn test`.

Do kompilowania służy `mvn compile`.

5.1.3. Perspektywa wdrożeniowca

Wymagania programowe:

- Java (JDK 15.0.2)

```
java version "15.0.2" 2021-01-19
Java(TM) SE Runtime Environment (build 15.0.2+7-27)
Java HotSpot(TM) 64-Bit Server VM (build 15.0.2+7-27, mixed mode, sharing)
```

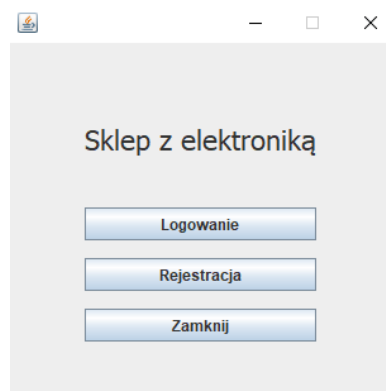
- Maven (3.6.3)
- Oracle Database (11g Express Edition)

```
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
PL/SQL Release 11.2.0.2.0 - Production
CORE      11.2.0.2.0      Production
TNS for 64-bit Windows: Version 11.2.0.2.0 - Production
NLSRTL Version 11.2.0.2.0 - Production
```

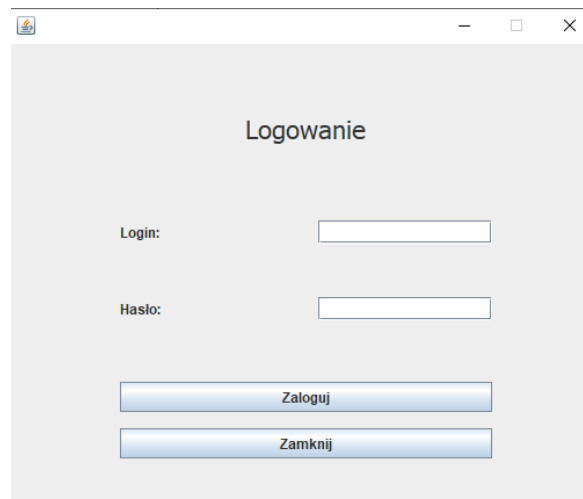
Należy postawić bazę poprzez uruchomienie wszystkich skryptów z punktu 4-4.3. Do zarządzania projektem w Javie jest używany Maven, zatem aby uruchomić aplikację należy pobrać zależności i wygenerować plik `.jar` poprzez `mvn clean install`, a następnie wystarczy uruchomić aplikację poprzez `java -jar target/com.BD_2-0.0.1-SNAPSHOT-jar-with-dependencies.jar`.

5.2. Instrukcja użytkownika aplikacji

Zaimplementowano aplikację jedynie dla pracownika. W menu głównym sklepu należy się zarejestrować lub zalogować.

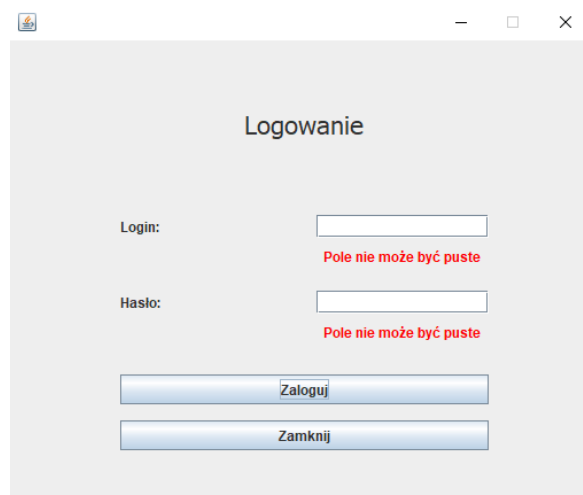


Rysunek 47 Menu główne

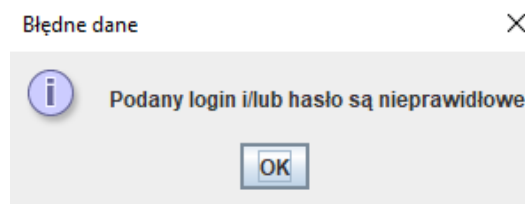


Rysunek 48 Logowanie

W przypadku wprowadzonych błędnych danych, zostanie wyświetlony odpowiedni komunikat.

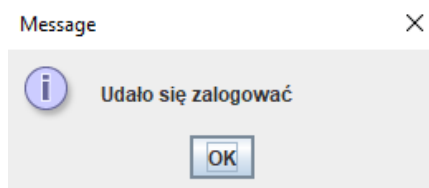


Rysunek 49 Komunikat o błędnych wprowadzonych danych logowania cz.1



Rysunek 50 Komunikat o błędnych wprowadzonych danych logowania cz.2

W przypadku pomyślnego logowania zostanie wyświetlony odpowiedni komunikat.



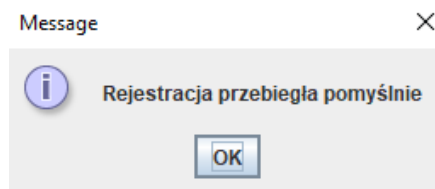
Rysunek 51 Komunikat o pomyślnym zalogowaniu

Rysunek 52 Rejestracja

W celu rejestracji należy podać wszystkie dane w poprawnym formacie. W przypadku wystąpienia błędu zostanie wyświetlony odpowiedni komunikat, np.

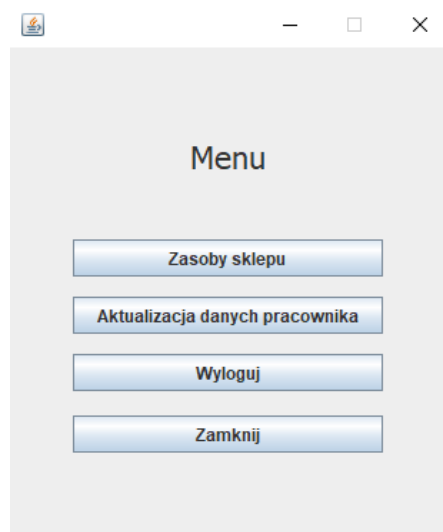
Rysunek 53 Komunikat o błędnych wprowadzonych danych

W przypadku pomyślnej rejestracji zostanie wyświetlony odpowiedni komunikat.



Rysunek 54 Komunikat o pomyślnej rejestracji

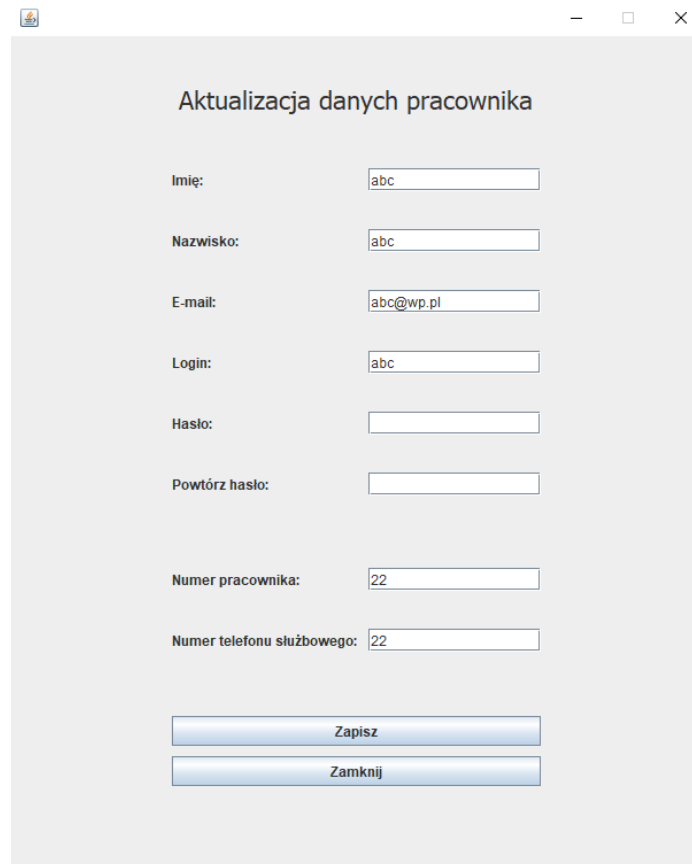
Po pomyślnej rejestracji lub logowaniu nastąpi przejście do strony głównej dostępnej dla zalogowanego użytkownika.



Rysunek 55 Menu po zalogowaniu

Następnie można przejść do pożądaney sekcji.

W przypadku chęci edycji danych pracowniczych należy zmienić pożądanę dane i kliknąć „Zapisz” i tylko zmienione dane zostaną zaktualizowane.



Aktualizacja danych pracownika

Imię: abc

Nazwisko: abc

E-mail: abc@wp.pl

Login: abc

Hasło:

Powtórz hasło:

Numer pracownika: 22

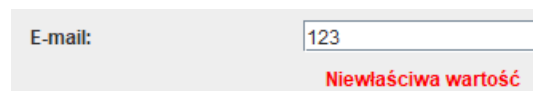
Numer telefonu służbowego: 22

Zapisz

Zamknij

Rysunek 56 Zmiana danych pracowniczych

W przypadku wprowadzenia niepoprawnych danych, wyświetlony zostanie adekwatny komunikat.

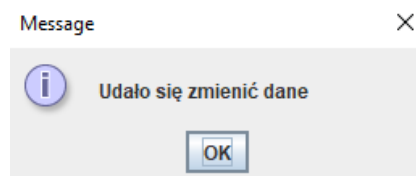


E-mail: 123

Niewłaściwa wartość

Rysunek 57 Komunikat o niepoprawnych wprowadzonych wartości przy zmianie danych pracownika

W przypadku pomyślnej zmiany danych zostanie wyświetlony odpowiedni komunikat.



Message

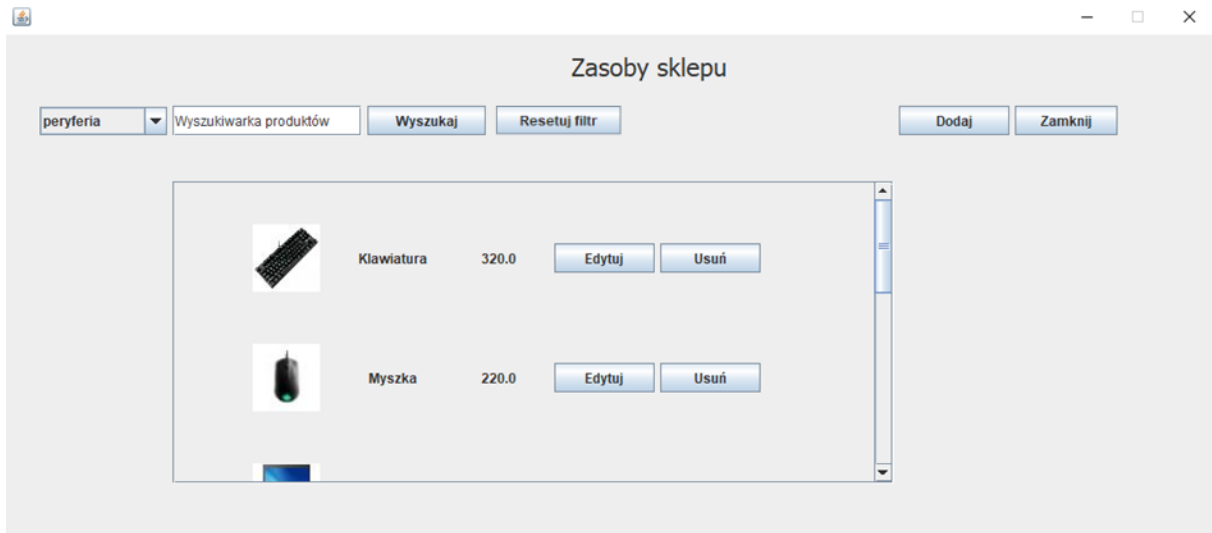
Udało się zmienić dane

OK

Rysunek 58 Komunikat o pomyślnej zmianie danych pracownika

Można wylogować się z konta poprzez przycisk „Wyloguj”, a z kolei wyjście z aplikacji może być uzyskane poprzez kliknięcie „X” w prawym górnym rogu lub kliknięcie przycisku „Zamknij” w oknie głównym przez zalogowaniem i w oknie głównym po zalogowaniu.

Po przejściu do zasobów sklepu widoczne będzie następujące okno.



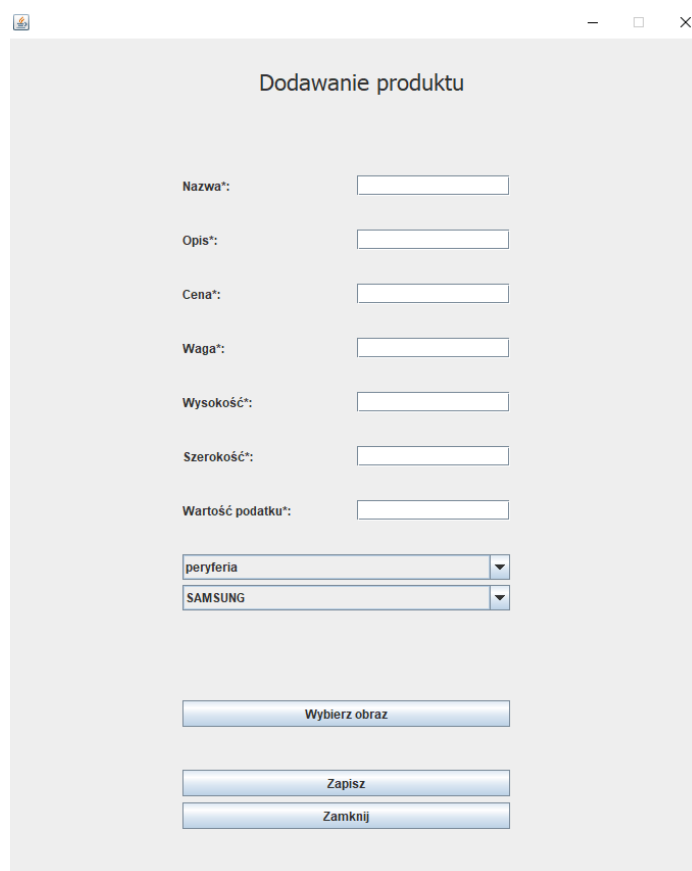
Rysunek 59 Okno z zasobami sklepu

Aby wyświetlić detale produktu, należy wliście z produktami kliknąć na obrazek (lub napis informujący o braku obrazka) danego produktu.



Rysunek 60 Okno z detalami produktu

W celu dodania nowego produktu należy kliknąć przycisk „Dodaj”. Jedynym nieobowiązkowym polem jest pole z wybieraniem obrazka produktu.



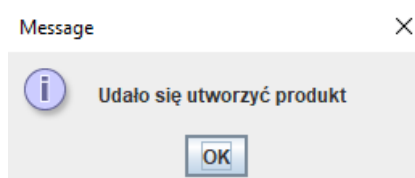
Rysunek 61 Okno z dodawaniem produktu

W przypadku błędnych wprowadzonych danych, zostanie wyświetlony odpowiedni komunikat.



Rysunek 62 Komunikat o błędnych wprowadzonych danych podczas tworzenia produktu

W przypadku pomyślnego dodania produktu, zostanie wyświetlony odpowiedni komunikat.



Rysunek 63 Komunikat pomyślnym dodaniu produktu

W przypadku przypadku edytowania produktu, należy kliknąć przycisk „Edytuj” przy odpowiednim produkcie. Jest możliwość zmiany tak dużej liczby pól, jak jest to pożądane.

Rysunek 64 Okno z edycją produktu

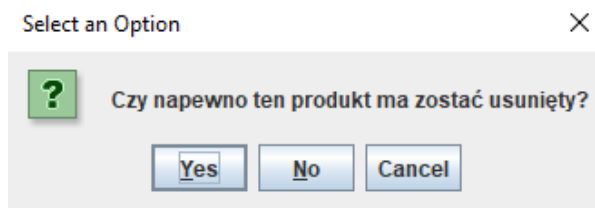
W przypadku błędnych wprowadzonych danych, zostanie wyświetlony odpowiedni komunikat.

Rysunek 65 Komunikat o błędnych wprowadzonych danych przy edycji produktu

W przypadku pomyślnego edytowania produktu, zostanie wyświetlony odpowiedni komunikat.

Rysunek 66 Komunikat o błędnych wprowadzonych danych przy edycji produktu

Aby usunąć dany produkt należy kliknąć przycisk „Usuń” i potwierdzić wykonanie operacji usunięcia.



Rysunek 67 Potwierdzenie usunięcia produktu

System pozwala również na wyszukiwanie produktów po kategorii i po nazwie. Nie jest możliwe wyszukiwanie po samej nazwie i trzeba podać przy tym odpowiednią kategorię. Możliwe jest za to wyszukiwanie tylko po kategorii. W obu przypadkach należy kliknąć przycisk „Wyszukaj” w celu wyświetlenia wyników wyszukiwania. W celu zresetowania filtrów (wyszukanie wszystkich produktów) należy kliknąć przycisk „Resetuj filtr”. Opcja wyszukiwanie po nazwie nie uwzględnia wielkości liter, ale bierze pod uwagę nazwę produktu od pierwszej litery.

5.3. Testowanie opracowanych funkcji systemu

5.3.1. Rejestracja

W aplikacji należy wypełnić wszystkie pola odpowiednimi danymi.

A screenshot of a registration form titled "Rejestracja". It contains several input fields with labels: "Imię:" (Name), "Nazwisko:" (Surname), "E-mail:", "Login:", "Hasło:" (Password), "Powtórz hasło:" (Repeat password), "Numer pracownika:" (Employee number), and "Numer telefonu służbowego:" (Office phone number). Each field has a sample value entered: "123", "123", "123@wp.pl", "123", "...", "...", "321", and "321". At the bottom, there are two buttons: "Zarejestruj" (Register) and "Zamknij" (Close).

Rysunek 68 Wprowadzenie danych do rejestracji

W przypadku wprowadzonych błędnych danych, został wyświetlony następujący komunikat:

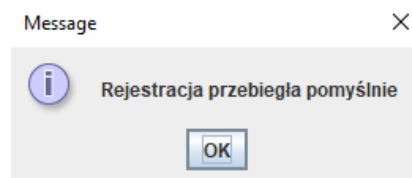


E-mail: Niewłaściwa wartość

Login: Pole nie może być puste

Rysunek 69 Komunikat o błędnych wprowadzonych danych

Po pomyślnym zarejestrowaniu (dane z rysunku 58) został wyświetlony komunikat.



Rysunek 70 Komunikat o pomyślnej rejestracji

Aby sprawdzić czy rzeczywiście zostało utworzone konto, można zobaczyć to w bazie danych.

```
SQL> SELECT FIRSTNAME, USERNAME, E_MAIL FROM BD_2.PERSON;
FIRSTNAME
-----
USERNAME
-----
E_MAIL
-----
abc
abc
abc@wp.pl

kamil
kamil
kamil@wp.pl

FIRSTNAME
-----
USERNAME
-----
E_MAIL
-----

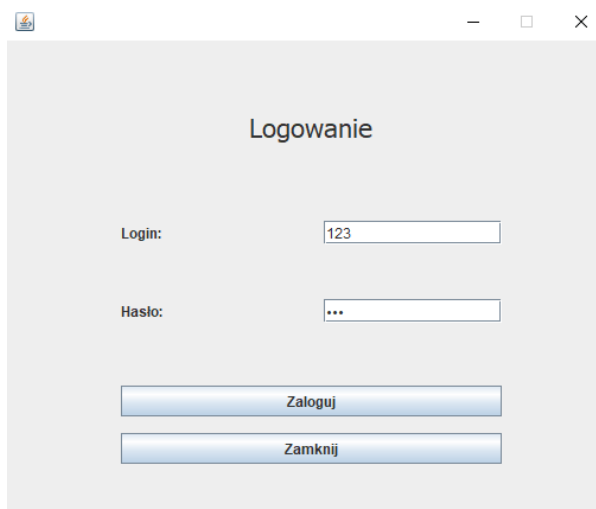
123
123
123@wp.pl
```

Rysunek 71 Zawartość tabeli Person po rejestracji

Można zauważyć, że został utworzony użytkownik o loginie 123, co zgadza się z wprowadzonymi danymi.

5.3.2. Logowanie

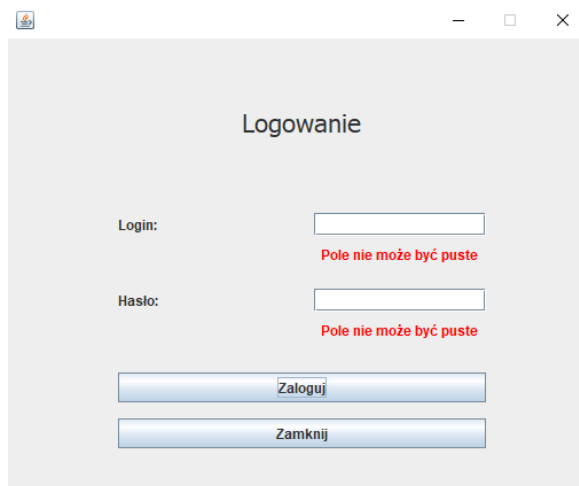
Do logowania należy podać login i hasło.



The screenshot shows a standard Windows-style window titled "Logowanie". It contains two input fields: "Login:" with the value "123" and "Hasło:" with three dots indicating a password. Below the fields are two buttons: "Zaloguj" and "Zamknij".

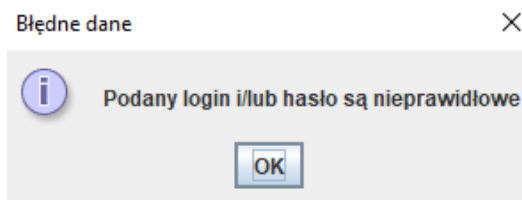
Rysunek 72 Okno logowania

W przypadku wprowadzonych błędnych danych, został wyświetlony odpowiedni komunikat.



This screenshot shows the same "Logowanie" window, but the input fields are empty. Below each field is a red error message: "Pole nie może być puste" (Field cannot be empty). The "Zaloguj" and "Zamknij" buttons are still present.

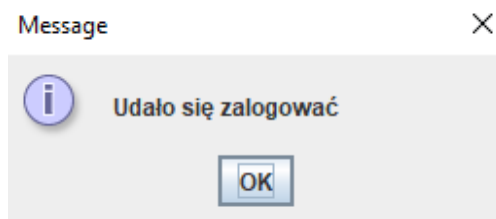
Rysunek 73 Komunikat błędnych wprowadzonych danych logowania cz.1



The screenshot shows a small error dialog box titled "Błędne dane" with a close button (X). It contains an information icon (i) and the text "Podany login i/lub hasło są nieprawidłowe" (The provided login and/or password are incorrect). There is an "OK" button at the bottom.

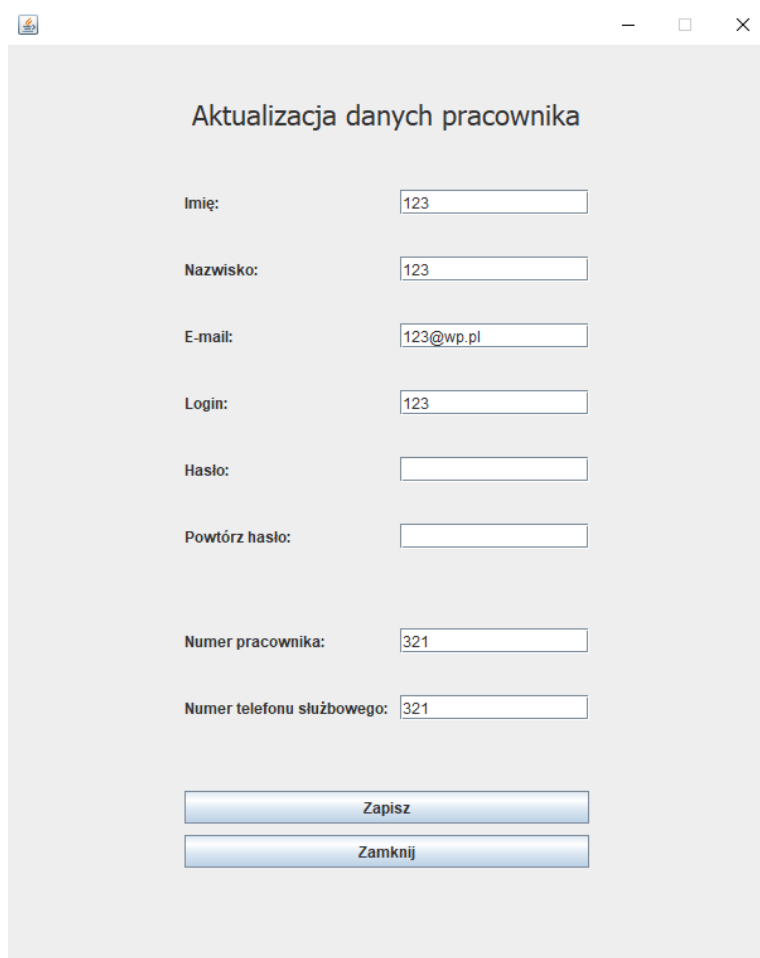
Rysunek 74 Komunikat błędnych wprowadzonych danych logowania cz.2

Po pomyślnym zalogowaniu (login: 123 oraz hasło: 123) został wyświetlony następujący komunikat:



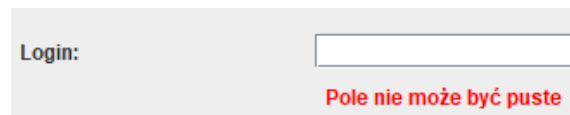
Rysunek 75 Komunikat o pomyślnym logowaniu

5.3.3. Zmiana danych pracowniczych



Rysunek 76 Dane pracownicze przed zmianą

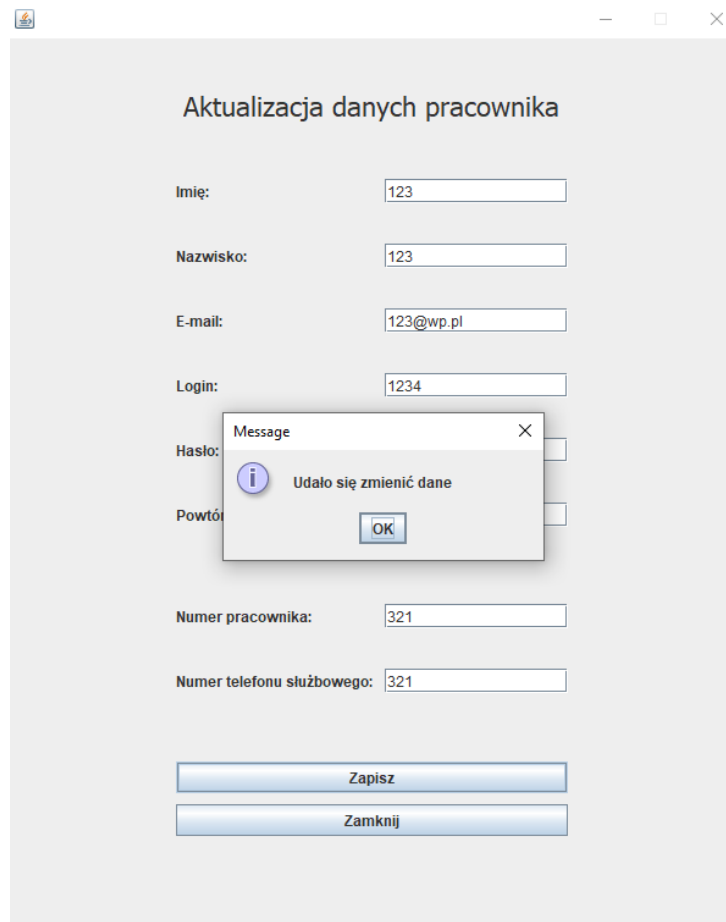
W przypadku błędnych wprowadzonych danych, został wyświetlony odpowiedni komunikat.



A screenshot of a login form. It has a label "Login:" followed by a text input field. Below the input field, the text "Pole nie może być puste" is displayed in red, indicating a validation error for an empty field.

Rysunek 77 Komunikat o błędnych wprowadzonych danych przy aktualizacji danych pracowniczych

Dla przetestowania tej funkcji zostanie zmieniony login na „1234”.



A screenshot of a web application window titled "Aktualizacja danych pracownika". The form contains several input fields: "Imię:" (123), "Nazwisko:" (123), "E-mail:" (123@wp.pl), "Login:" (1234), "Hasło:" (empty), "Powtórz:" (empty), "Numer pracownika:" (321), and "Numer telefonu służbowego:" (321). At the bottom are two buttons: "Zapisz" and "Zamknij". A modal dialog box titled "Message" is overlaid on the form, displaying an information icon and the text "Udało się zmienić dane" (Data was successfully changed), with an "OK" button.

Rysunek 78 Komunikat o pomyślnej zmianie danych

Na rysunku 78 można zauważyć, że zmiana danych przebiegła pomyślnie.

Aby sprawdzić czy rzeczywiście zostały zaktualizowane dane, można zobaczyć to w bazie danych.

```
SQL> SELECT FIRSTNAME, USERNAME, E_MAIL FROM BD_2.PERSON WHERE USERNAME='1234';
```

FIRSTNAME	USERNAME	E_MAIL
123	1234	123@wp.pl

Rysunek 79 Zmienione dane użytkownika '123'

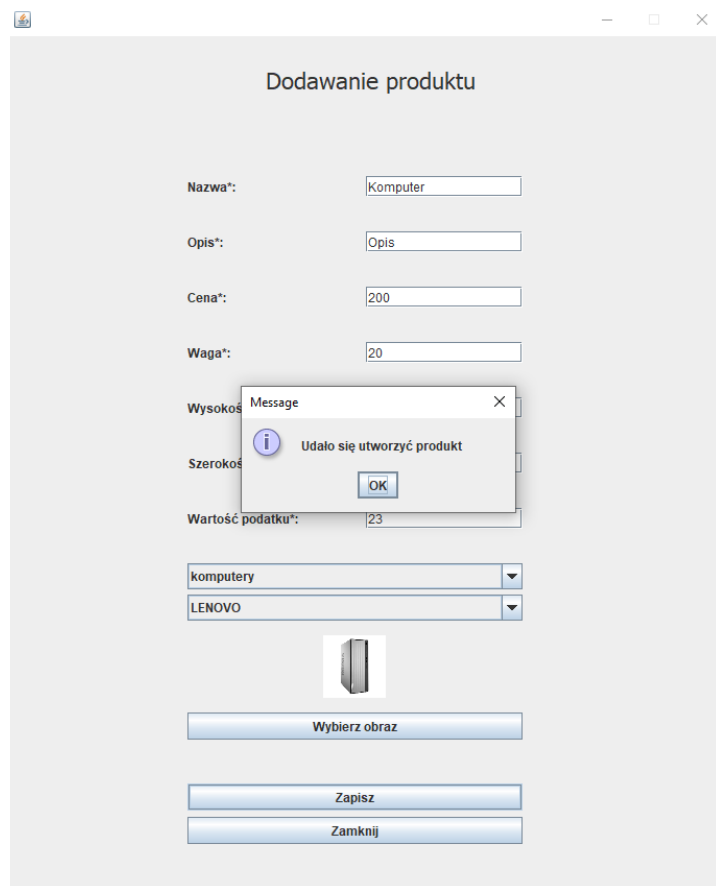
Na rysunku 67 widać, że login utworzonego wcześniej użytkownika „123” jest teraz „1234”, co zgadza się oczekiwaniami.

5.3.4. Dodawanie produktu

Rysunek 80 Testowanie dodawania produktu

W przypadku błędnych wprowadzonych danych, został wyświetlony odpowiedni komunikat.

Rysunek 81 Testowanie dodawania produktu – błędne wprowadzone dane



Rysunek 82 Testowanie dodawania produktu – pomyślne utworzenie produktu

Na rysunku można zauważyć, że dodanie produktu przebiegło pomyślnie.

Efekt dodania produktu można zauważyć w oknie z zasobami sklepu. Widać, że rzeczywiście został dodany nowy produkt o danych równymi tymi wprowadzonymi.



Rysunek 83 Testowanie dodawania produktu – rezultat utworzenia produktu

Detale produktu



Nazwa: Komputer

Opis: Opis

Cena: 200.0 zł

Waga: 20.0

Wysokość: 21.0

Szerokość: 23.0

Kategoria: komputery

Producent: LENOVO

Zamknij

Rysunek 84 Testowanie dodawania produktu – rezultat utworzenia produktu – detale produktu

5.3.5. Edytowanie produktu

Zmiana danych produktu

Nazwa:

Opis:

Cena:

Waga:


Wysokość:

Szerokość:

Wartość podatku:

▼

▼



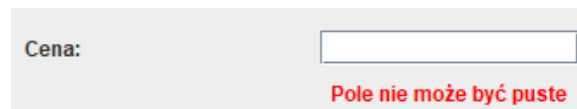
Wybierz obraz

Zapisz

Zamknij

Rysunek 85 Testowanie edytowania produktu – dane edytowanego produktu

W przypadku błędnych wprowadzonych danych, został wyświetlony odpowiedni komunikat.

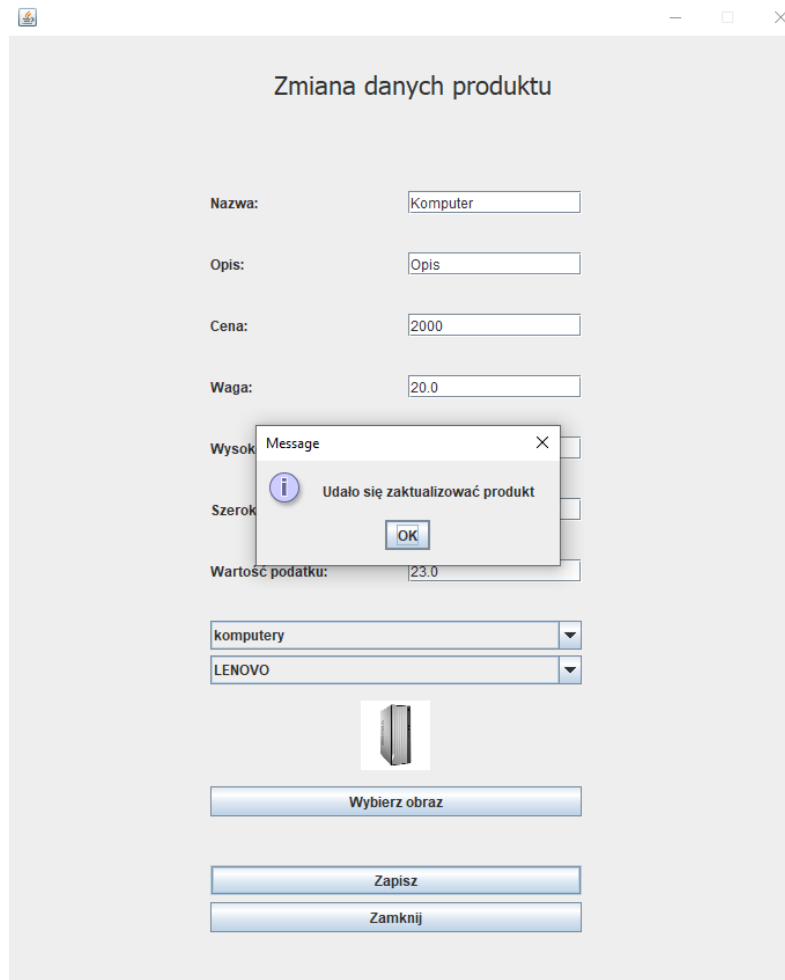


Cena:

Pole nie może być puste

Rysunek 86 Komunikat o błędnych wprowadzonych danych przy aktualizacji produktu

Dla przetestowania tej funkcji zostanie zmieniona cena na „2000”.



Zmiana danych produktu

Nazwa:

Opis:

Cena:

Waga:


Wysokość:

Szerokość:

Wartość podatku:

komputery

LENOVO



Wybierz obraz

Zapisz

Zamknij

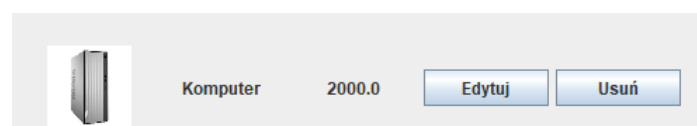
Message


Udało się zaktualizować produkt

OK

Rysunek 87 Pomyślne aktualizowanie produktu

Efekt edytowania produktu można zauważyć w oknie z zasobami sklepu. Widać, że rzeczywiście została zmieniona cena produktu.



 Komputer 2000.0

Rysunek 88 Rezultat zaktualizowania produktu

5.3.6. Usuwanie produktu

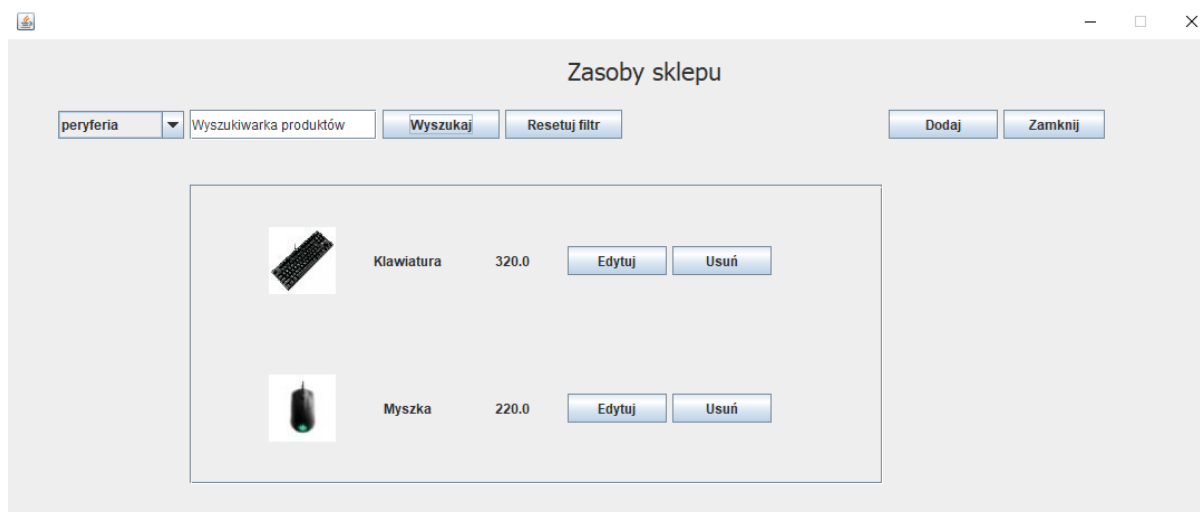
W ramach testów zostanie usunięty poprzednio dodany produkt (np. rysunek 88).



Rysunek 89 Rezultat usunięcia produktu

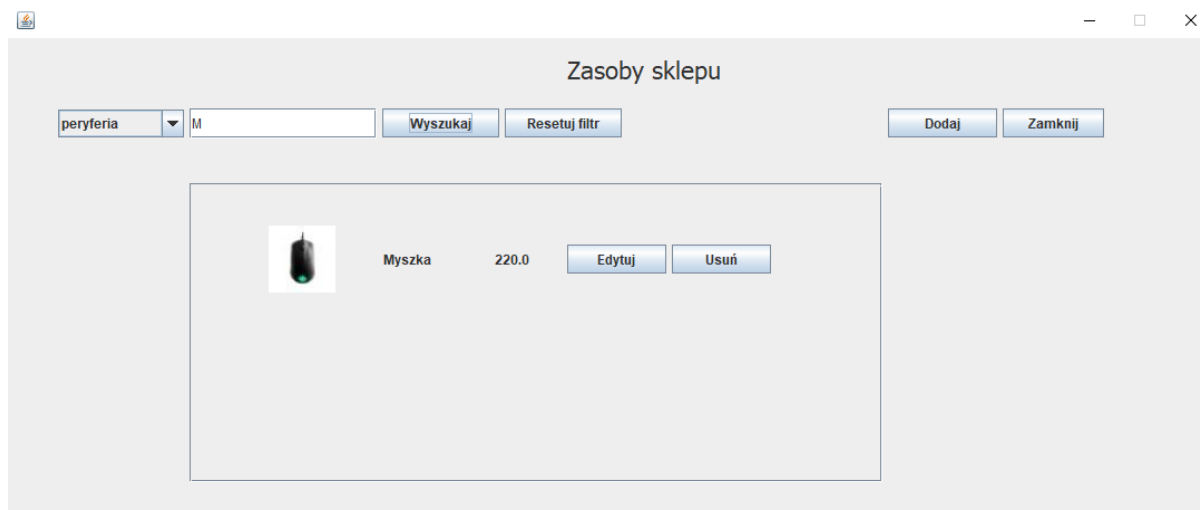
Na rysunku 89 można zauważyć, że produkt ten został usunięty.

5.3.7. Wyszukiwanie produktów



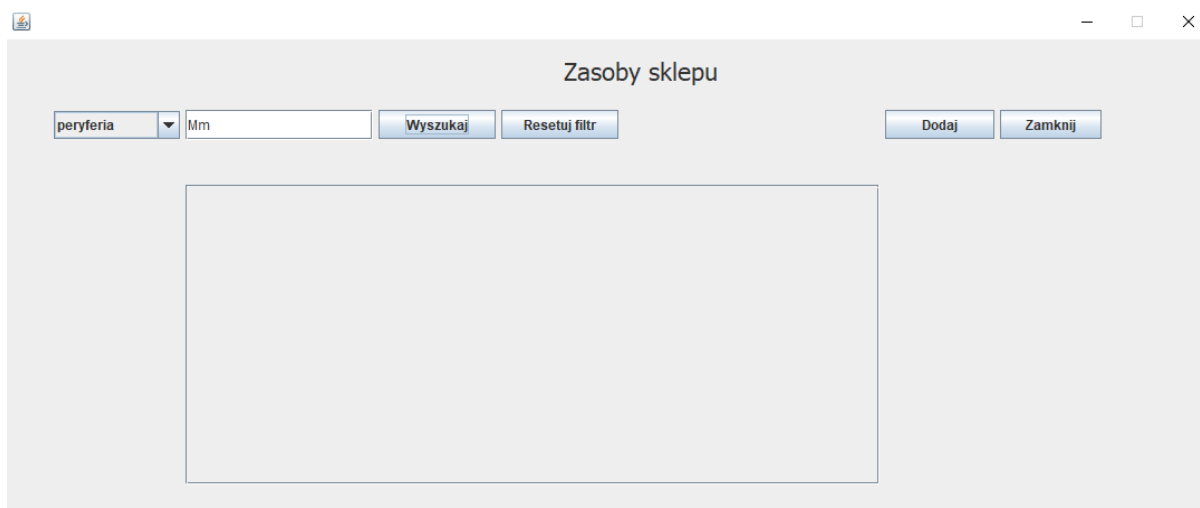
Rysunek 90 Rezultat wyszukiwania peryferii

Na rysunku 90 można zauważyć, że zostały wyszukane tylko peryferia, co zgadza się z wprowadzoną kategorią produktów.



Rysunek 91 Rezultat wyszukiwania peryferii o danej nazwie

Na rysunku 91 można zauważyć, że opcja wyszukiwania po nazwie również działa, gdyż otrzymano produkt zaczynający się na „M”.



Rysunek 92 Rezultat wyszukiwania nieistniejącego produktu

Na podstawie rysunku 92 widać, że jak wprowadzono nazwę nieistniejącego produktu, to nie został on wyszukany.



Rysunek 93 Rezultat zresetowania filtrów wyszukiwania

Na podstawie rysunku 93 można zauważyć, że opcja „Resetuj filtr” spowodowała wyszukanie wszystkich produktów.

5.4. Omówienie wybranych rozwiązań programistycznych

5.4.1. Implementacja interfejsu dostępu do bazy danych

Do połączenia z bazą danych została wykorzystana biblioteka JDBC. Aby się połączyć z bazą danych należy skorzystać ze specjalnego sterownika, który tłumaczy odwołania z poziomu Javy na odwołania właściwe dla danego RDBMS (w tym przypadku Oracle Database). Połączenie polega na załadowaniu sterownika JDBC (w tym przypadku „*oracle.jdbc.driver.OracleDriver*”) i zażądaniu od sterownika połączenia. Połączenie jest tworzone w następujący sposób:

```
public static Connection createConnection(String username, char[] password) {
    try {
        Class.forName(driverName);

        connection = DriverManager.getConnection(url, username,
            String.valueOf(password));
    }
    catch (ClassNotFoundException e) {
        System.out.println("Driver not found");
    }
    catch (SQLException e) {
        e.printStackTrace();

        System.out.println("Failed to create the database connection");
    }

    return connection;
}
```

Do połączenia z bazą danych używane jest konto dostępu do bazy danych o odpowiednich uprawnieniach (w tym przypadku jedynie konto pracownika).

5.4.2. Implementacja wybranych funkcjonalności systemu

W ramach aplikacji udało się zaimplementować logowanie, rejestrację, zmianę danych pracowniczych oraz operacje CRUD na produktach.

W aplikacji można wyróżnić różne warstwy w zależności od ich odpowiedzialności i są one następujące:

- Warstwa bazy danych – jak nazwa wskazuje, jest to warstwa, której zadaniem jest przechowywanie informacji w bazie danych (Oracle Database).
- Warstwa dostępu do danych (DAO – data access object) – stanowi interfejs komunikacji między aplikacją a źródłem danych i w tym przypadku źródłem danych są pliki tekstowe, obrazy jpg oraz baza danych.
- Warstwa serwisowa – stanowi interfejs komunikacji między DAO a warstwą GUI
- Warstwa interfejsu użytkownika (GUI – graphical user interface) – stanowi graficzny interfejs użytkownika, która jest zaimplementowana jako desktopowe GUI w Swingu.

Aplikacja została zaimplementowana jako jedno okno „JFrame”, w którym wyświetlane są różne panele „JPanel” np. panel logowania.

Rejestracja

```
String query = "{call BD_2.INSERT_EMPLOYEE(?,?,?,?,?,?,?)}";
```

```
CallableStatement statement;
```

```
statement = ConnectionDAO.connection.prepareCall(query);
```

```
fillStatement(statement, employee);
```

```
statement.executeQuery();
```

Rejestracja jest uzyskiwana przy pomocy JDBC poprzez uruchomienie procedury składowanej BD_2.INSERT_EMPLOYEE razem z wymaganymi parametrami np. dane osobowe pracownicze i login i hasło.

Logowanie

```
PreparedStatement statement =
```

```
ConnectionDAO.connection.prepareStatement("SELECT * FROM BD_2.PERSON WHERE  
username = ?");
```

```
statement.setString(1, username);
```

```
ResultSet results = statement.executeQuery();
```

```
Employee employee = getOneFromResults(results);
```

```
results.close();
```

```
statement.close();
```

Logowanie jest uzyskiwane poprzez wybranie z bazy danych konta o wprowadzonym loginie i porównaniu hasła z tego konta do wprowadzonego hasła. Powyżej znajduje się kod odpowiadający za wyszukanie konta o danym loginie.

Zmiana danych pracowniczych

```
PreparedStatement statement = ConnectionDAO.connection.prepareStatement(  
    "UPDATE BD_2.PERSON "  
    + "SET firstname=?, surname=?, e_mail=?,  
    "username=?, password=?, hiring_date=?, employee_number=? "  
    + "WHERE ID = ?");  
  
fillStatement(statement, employee);  
statement.setLong(8, id);  
  
statement.executeUpdate();  
statement.close();
```

Zmiana danych jest możliwa poprzez wywołanie instrukcji UPDATE bazy danych z wprowadzonymi zmienionymi parametrami.

5.4.3. Implementacja mechanizmów bezpieczeństwa

W aplikacji są używane odpowiednie konta dostępu do bazy danych, aby dany użytkownik miał dostęp tylko do niezbędnych dla niego funkcji systemu. Dane do logowania do bazy danych są zapisane w pliku tekstowym „Users.txt” aby zwiększyć bezpieczeństwo (wydaje się bardziej bezpieczne niż wprowadzenie po prostu danych w kodzie).

Po stronie aplikacji jedynym mechanizmem bezpieczeństwa jest logowanie przy pomocy loginu i hasła. Dane do logowania są przechowywane w bazie danych i przy logowaniu są one porównywane z tymi wprowadzonymi przez użytkownika.

6. Podsumowanie i wnioski

- W początkowych założeniach projektu było stworzenie osobnej aplikacji webowej przeznaczonej dla klienta jednak udało się stworzyć jedynie aplikację desktopową przeznaczoną dla pracownika.
- Aplikacja korzysta tylko z następujących procedur składowanych: "INSERT_PHONE_NUMBER", "INSERT_EMPLOYEE", "INSERT_ABSTRACT_PRODUCT".
- W trakcie implementacji aplikacji dostępowej wystąpiły problemy uprawnieniami dla użytkowników bazy danych, gdyż okazało się, że aby mogli oni utworzyć połączenie z bazą danych, to muszą mieć przypisane uprawnienie do tworzenia sesji. Dodatkowym problemem było nie przypisanie uprawnień dla użytkownika do używania określonych procedur składowanych. Ostatecznie udało się naprawić wymienione problemy.

6.1.Literatura

Dokumentacje:

Java:

<https://docs.oracle.com/en/java/javase/15/docs/api/index.html>

Swing:

<https://docs.oracle.com/en/java/javase/15/docs/api/java.desktop/javax/swing/package-summary.html>

Oracle database:

https://docs.oracle.com/cd/E11882_01/index.htm

Maven:

<https://maven.apache.org/guides/>

JDBC:

<https://docs.oracle.com/en/java/javase/15/docs/api/java.sql/java/sql/package-summary.html>

6.2.Spis rysunków

RYSUNEK 1 UPROSZCZONY MODEL KONCEPTUALNY BAZY DANYCH	5
RYSUNEK 2 MODEL LOGICZNY BAZY DANYCH	5
RYSUNEK 3 MODEL FIZYCZNY BAZY DANYCH	6
RYSUNEK 4 DIAGRAM PRZYPADKÓW UŻYCIA SYSTEMU	16
RYSUNEK 5 STRONA STARTOWA APLIKACJI	17
RYSUNEK 6 WYBÓR RODZAJU KONTA DO UTWORZENIA	17
RYSUNEK 7 REJESTRACJA W SYSTEMIE DLA KLIENTA	17
RYSUNEK 8 REJESTRACJA W SYSTEMIE DLA PRACOWNIKA	18
RYSUNEK 9 WIDOK ZASOBÓW SKLEPU Z PERSPEKTYWY ZAŁOGOWANEGO/NIEZAŁOGOWANEGO KLIENTA	18
RYSUNEK 10 DETALE PRODUKTU	18
RYSUNEK 11 WIDOK KOSZYKA	19
RYSUNEK 12 WIDOK ZASOBÓW SKLEPU Z PERSPEKTYWY ZAŁOGOWANEGO PRACOWNIKA	19
RYSUNEK 13 WIDOK EDYCJI DANYCH PRODUKTU PRZEZ PRACOWNIKA	19
RYSUNEK 14 WYBÓR LOKALIZACJI DOSTAWY	20
RYSUNEK 15 HISTORIA TRANSAKCJI KLIENTA	20
RYSUNEK 16 DETALE DANEJ TRANSAKCJI KLIENTA	20
RYSUNEK 17 AKTUALIZACJA DANYCH KLIENTA	21
RYSUNEK 18 AKTUALIZACJA DANYCH PRACOWNIKA	21
RYSUNEK 19 OKNO LOGOWANIA	22
RYSUNEK 20 UDANE LOGOWANIE	23
RYSUNEK 21 NIEUDANE LOGOWANIE	23
RYSUNEK 22 POMIAR CZASU DLA PRZYKŁADOWEGO ZAPYTANIA WRAZ Z WYNIKIEM CZASOWYM	34
RYSUNEK 23 WYNIK ZAPYTANIA 1	41
RYSUNEK 24 WYNIK ZAPYTANIA 2	42

RYSUNEK 25 WYNIK ZAPYTANIA 3	42
RYSUNEK 26 WYNIK ZAPYTANIA 4	42
RYSUNEK 27 WYNIK ZAPYTANIA 5	43
RYSUNEK 29 WYNIK ZAPYTANIA 7	43
RYSUNEK 28 WYNIK ZAPYTANIA 6	43
RYSUNEK 30 WYNIK ZAPYTANIA 8	44
RYSUNEK 31 WYNIK TESTU UPRAWNIEŃ 1.....	44
RYSUNEK 32 WYNIK TESTU UPRAWNIEŃ 2.....	44
RYSUNEK 33 WYNIK TESTU UPRAWNIEŃ 3.....	44
RYSUNEK 34 WYNIK TESTU UPRAWNIEŃ 4.....	45
RYSUNEK 35 WYNIK TESTU UPRAWNIEŃ 5.....	45
RYSUNEK 36 WYNIK TESTU UPRAWNIEŃ 6.....	45
RYSUNEK 37 WYNIK TESTU UPRAWNIEŃ 7.....	45
RYSUNEK 38 WYNIK TESTU UPRAWNIEŃ 9.....	45
RYSUNEK 39 WYNIK TESTU UPRAWNIEŃ 10	46
RYSUNEK 40 WYNIK TESTU UPRAWNIEŃ 11	46
RYSUNEK 41 WYNIK TESTU UPRAWNIEŃ 12	46
RYSUNEK 42 WYNIK TESTU UPRAWNIEŃ 13	46
RYSUNEK 43 WYNIK TESTU UPRAWNIEŃ 14.....	46
RYSUNEK 44 WYNIK TESTU UPRAWNIEŃ 15.....	46
RYSUNEK 45 WYNIK TESTU UPRAWNIEŃ 16.....	46
RYSUNEK 46 WYNIK TESTU UPRAWNIEŃ 17	47
RYSUNEK 47 MENU GŁÓWNE.....	48
RYSUNEK 48 LOGOWANIE	49
RYSUNEK 49 KOMUNIKAT O BŁĘDNYCH WPROWADZONYCH DANYCH LOGOWANIA CZ.1	49
RYSUNEK 50 KOMUNIKAT O BŁĘDNYCH WPROWADZONYCH DANYCH LOGOWANIA CZ.2	49
RYSUNEK 51 KOMUNIKAT O POMYŚLNYM ZALOGOWANIU.....	50
RYSUNEK 52 REJESTRACJA.....	50
RYSUNEK 53 KOMUNIKAT O BŁĘDNYCH WPROWADZONYCH DANYCH	50
RYSUNEK 54 KOMUNIKAT O POMYŚLNEJ REJESTRACJI	51
RYSUNEK 55 MENU PO ZALOGOWANIU.....	51
RYSUNEK 56 ZMIANA DANYCH PRACOWNICZYCH	52
RYSUNEK 57 KOMUNIKAT O NIEPOPRAWNYCH WPROWADZONYCH WARTOŚCI PRZY ZMIANIE DANYCH PRACOWNIKA.....	52
RYSUNEK 58 KOMUNIKAT O POMYŚLNEJ ZMIANIE DANYCH PRACOWNIKA	52
RYSUNEK 59 OKNO Z ZASOBAMI SKLEPU	53
RYSUNEK 60 OKNO Z DETALAMI PRODUKTU.....	53
RYSUNEK 61 OKNO Z DODAWANIEM PRODUKTU	54
RYSUNEK 62 KOMUNIKAT O BŁĘDNYCH WPROWADZONYCH DANYCH PODCZAS TWORZENIA PRODUKTU	54
RYSUNEK 63 KOMUNIKAT POMYŚLNYM DODANIU PRODUKTU.....	54
RYSUNEK 64 OKNO Z EDYCJĄ PRODUKTU	55

RYSUNEK 65 KOMUNIKAT O BŁĘDNYCH WPROWADZONYCH DANYCH PRZY EDYCJI PRODUKTU.....	55
RYSUNEK 66 KOMUNIKAT O BŁĘDNYCH WPROWADZONYCH DANYCH PRZY EDYCJI PRODUKTU.....	55
RYSUNEK 67 POTWIERDZENIE USUNIĘCIA PRODUKTU	56
RYSUNEK 68 WPROWADZENIE DANYCH DO REJESTRACJI.....	56
RYSUNEK 69 KOMUNIKAT O BŁĘDNYCH WPROWADZONYCH DANYCH	57
RYSUNEK 70 KOMUNIKAT O POMYŚLNEJ REJESTRACJI	57
RYSUNEK 71 ZAWARTOŚĆ TABELI PERSON PO REJESTRACJI	57
RYSUNEK 72 OKNO LOGOWANIA	58
RYSUNEK 73 KOMUNIKAT BŁĘDNYCH WPROWADZONYCH DANYCH LOGOWANIA CZ.1.....	58
RYSUNEK 74 KOMUNIKAT BŁĘDNYCH WPROWADZONYCH DANYCH LOGOWANIA CZ.2.....	58
RYSUNEK 75 KOMUNIKAT O POMYŚLNYM LOGOWANIU	59
RYSUNEK 76 DANE PRACOWNICZE PRZED ZMIANĄ.....	59
RYSUNEK 77 KOMUNIKAT O BŁĘDNYCH WPROWADZONYCH DANYCH PRZY AKTUALIZACJI DANYCH PRACOWNICZYCH.....	60
RYSUNEK 78 KOMUNIKAT O POMYŚLNEJ ZMIANIE DANYCH	60
RYSUNEK 79 ZMIENIONE DANE UŻYTKOWNIKA '123'	61
RYSUNEK 80 TESTOWANIE DODAWANIA PRODUKU	61
RYSUNEK 81 TESTOWANIE DODAWANIA PRODUKU – BŁĘDNE WPROWADZONE DANE.....	61
RYSUNEK 82 TESTOWANIE DODAWANIA PRODUKU – POMYŚLNE UTWORZENIE PRODUKTU	62
RYSUNEK 83 TESTOWANIE DODAWANIA PRODUKU – REZULTAT UTWORZENIA PRODUKTU.....	62
RYSUNEK 84 TESTOWANIE DODAWANIA PRODUKU – REZULTAT UTWORZENIA PRODUKTU – DETALE PRODUKTU	63
RYSUNEK 85 TESTOWANIE EDYTOWANIA PRODUKU – DANE EDYTOWANEGO PRODUKTU.....	63
RYSUNEK 86 KOMUNIKAT O BŁĘDNYCH WPROWADZONYCH DANYCH PRZY AKTUALIZACJI PRODUKTU	64
RYSUNEK 87 POMYŚLNE AKTUALIZOWANIE PRODUKTU.....	64
RYSUNEK 88 REZULTAT ZAKTUALIZOWANIA PRODUKTU	64
RYSUNEK 89 REZULTAT USUNIĘCIA PRODUKTU	65
RYSUNEK 90 REZULTAT WYSZUKIWANIA PERYFERII	65
RYSUNEK 91 REZULTAT WYSZUKIWANIA PERYFERII O DANEJ NAZWIE	66
RYSUNEK 92 REZULTAT WYSZUKIWANIA NIEISTNIEJĄCEGO PRODUKTU	66
RYSUNEK 93 REZULTAT ZRESETOWANIA FILTRÓW WYSZUKIWANIA.....	67

6.3.Spis tabel

TABELA 1 TABELA UPRAWNIEŃ AKTORÓW	16
TABELA 2 WYNIK CZASOWY 1.....	35
TABELA 3 WYNIK CZASOWY 2.....	35
TABELA 4 WYNIK CZASOWY 3.....	35
TABELA 5 WYNIK CZASOWY 4.....	35
TABELA 6 WYNIK CZASOWY 5.....	36

TABELA 7 WYNIK CZASOWY 6.....	36
TABELA 8 WYNIK CZASOWY 7.....	36
TABELA 9 WYNIK CZASOWY 8.....	36
TABELA 10 WYNIK CZASOWY 9.....	37
TABELA 12 WYNIK CZASOWY 11.....	37
TABELA 13 WYNIK CZASOWY 12.....	37
TABELA 11 WYNIK CZASOWY 10.....	37
TABELA 15 WYNIK CZASOWY 14.....	38
TABELA 16 WYNIK CZASOWY 15.....	38
TABELA 14 WYNIK CZASOWY 13.....	38
TABELA 17 WYNIK CZASOWY 16.....	39
TABELA 19 WYNIK CZASOWY 18.....	39
TABELA 20 WYNIK CZASOWY 19.....	39
TABELA 18 WYNIK CZASOWY 17.....	39
TABELA 21 WYNIK CZASOWY 20.....	40
TABELA 22 WYNIK CZASOWY 21.....	40
TABELA 24 WYNIK CZASOWY 23.....	40
TABELA 23 WYNIK CZASOWY 22.....	40
TABELA 26 WYNIK CZASOWY 24.....	41

7. Link do plików projektu

https://github.com/kamil20020/BD_2