|  |
| --- |
| C:\Users\ASUS\AppData\Local\Temp\Rar$DIa18116.39973\logo PWr kolor poziom  bez tla.png 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 2](#_Toc89412560)

[1.1. Cel projektu 2](#_Toc89412561)

[1.2. Zakres projektu 2](#_Toc89412562)

[2. Analiza wymagań 2](#_Toc89412563)

[2.1. Opis działania systemu 2](#_Toc89412564)

[2.2. Opis zasobów ludzkich 2](#_Toc89412565)

[2.3. Wymagania funkcjonalne 2](#_Toc89412566)

[2.4. Wymagania niefunkcjonalne 3](#_Toc89412567)

[2.4.1. Wymagania dotyczące bezpieczeństwa systemu 3](#_Toc89412568)

[2.4.2. Wymagania dotyczące rozmiaru bazy danych 3](#_Toc89412569)

[2.4.3. Wykorzystywane technologie i narzędzia 3](#_Toc89412570)

[3. Projekt systemu 3](#_Toc89412571)

[3.1. Projekt bazy danych 3](#_Toc89412572)

[3.1.1. Uproszczony model konceptualny 3](#_Toc89412573)

[3.1.2. Model logiczny 4](#_Toc89412574)

[3.1.3. Model fizyczny 4](#_Toc89412575)

[3.1.4. Inne elementy schematu – mechanizmy przetwarzania danych 5](#_Toc89412576)

[3.1.5. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych 9](#_Toc89412577)

[4. Modelowanie aplikacji dostępowej 10](#_Toc89412578)

[4.1. Diagram przypadków użycia 10](#_Toc89412579)

[4.2. Interfejs graficzny 10](#_Toc89412580)

[4.3. Metoda podłaczania do bazy danych – integracja z bazą danych 15](#_Toc89412581)

[4.4. Projekt zabezpieczeń na poziomie aplikacji 15](#_Toc89412582)

# Wstęp

## Cel projektu

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

## Zakres projektu

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

# Analiza wymagań

## 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.

## 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.

## 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.

## Wymagania niefunkcjonalne

### 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.

### 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 kategori,
* oszacowano, że dzienna liczba transakcji wynosi 200,
* przy czym największy ruch jest w godzinach wieczornych.

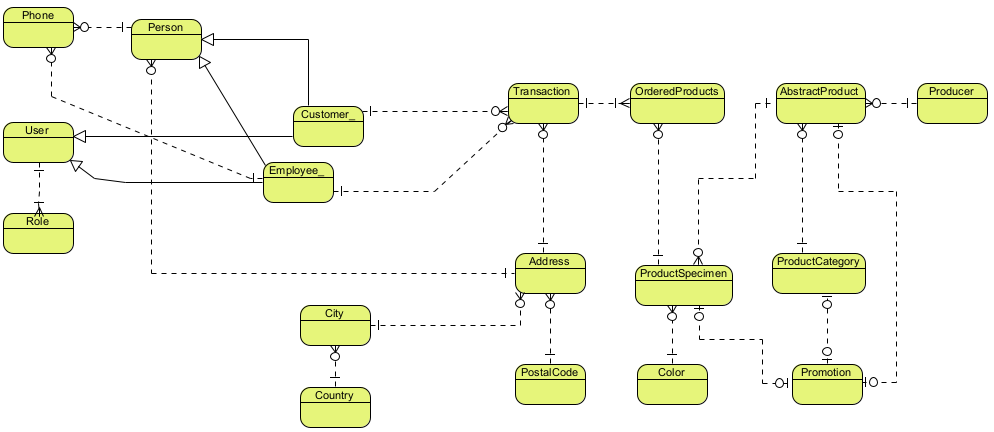
### Wykorzystywane technologie i narzędzia

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

# Projekt systemu

## Projekt bazy danych

### Uproszczony model konceptualny



Rysunek 1 Uproszczony model konceptualny bazy danych

### Model logiczny

Rysunek 2 Model logiczny bazy danych

### Model fizyczny

Rysunek 3 Model fizyczny bazy danych

### 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)

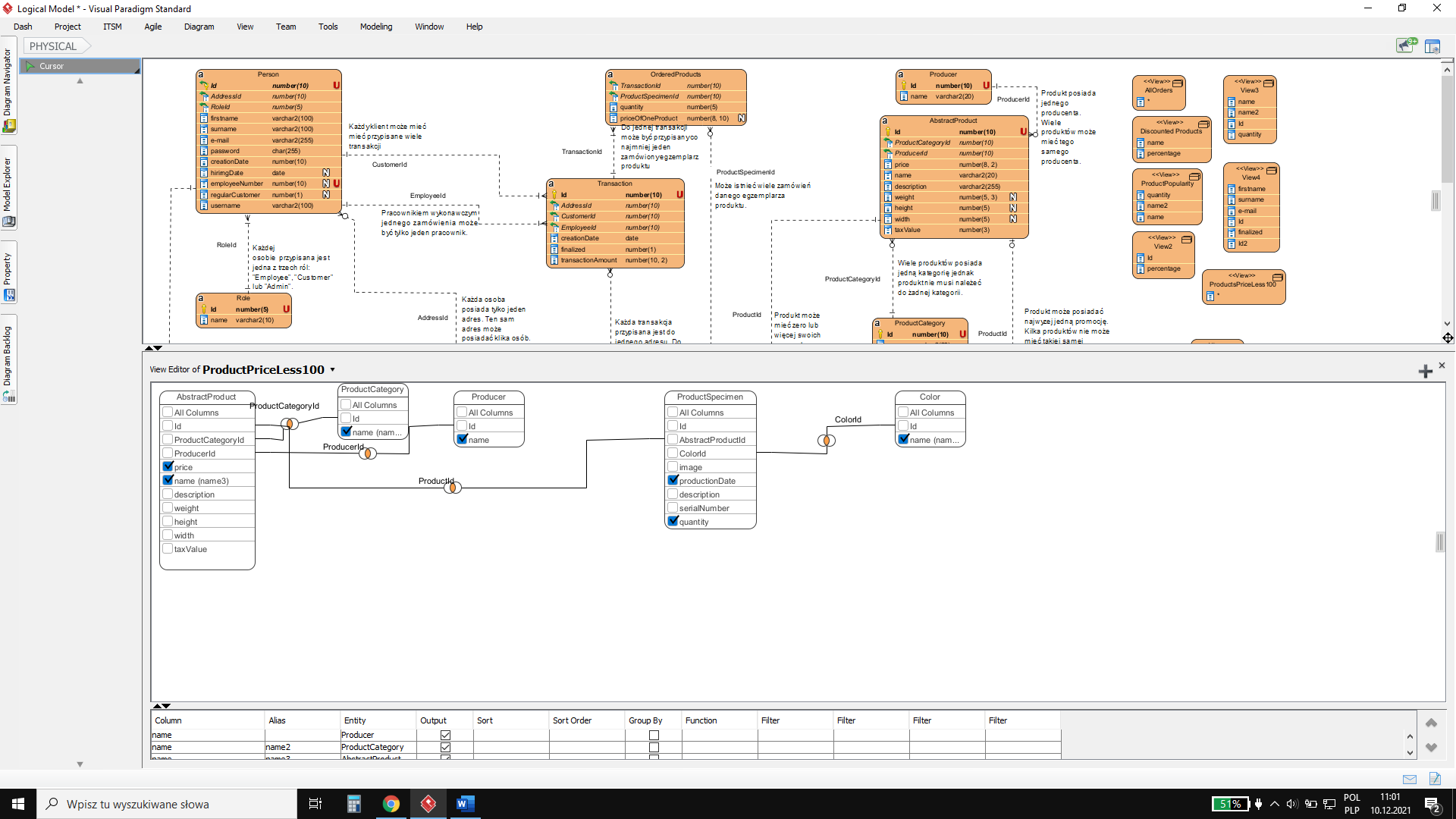
THEN

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

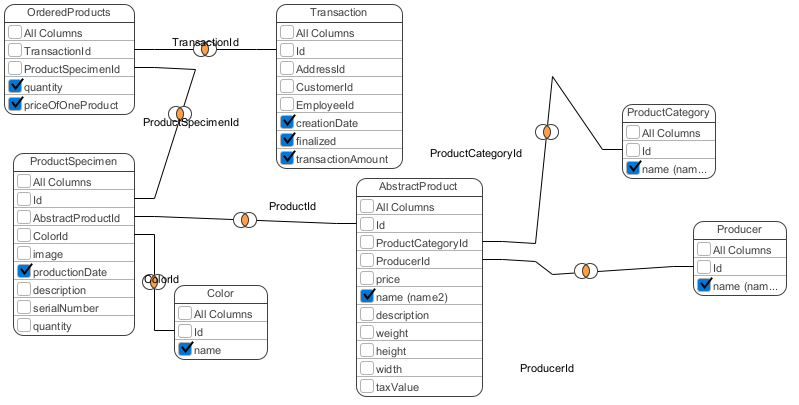
INNER JOIN BD\_2.PRODUCT\_CATEGORY P\_C ON A\_P.PRODUCT\_CATEGORY\_ID = P\_C.Id

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 transakcjii

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

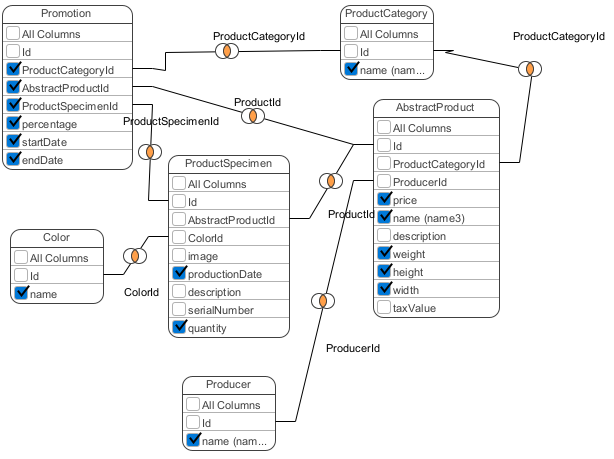
INNER JOIN BD\_2.PRODUCT\_SPECIMEN p\_s ON O\_P.Product\_Specimen\_Id = p\_s.Id

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

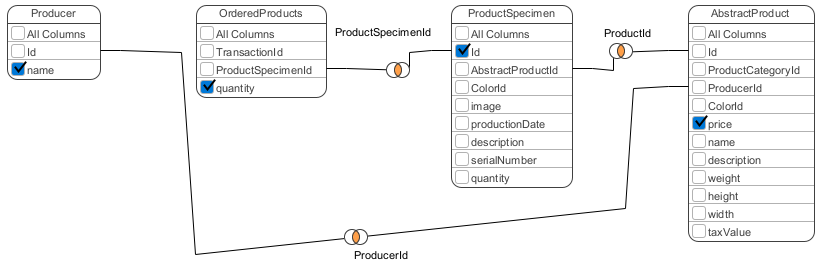
INNER JOIN BD\_2.Abstract\_Product a\_p ON p.Abstract\_Product\_Id = a\_p.Id AND p\_c.Id = a\_p.Product\_Category\_Id

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

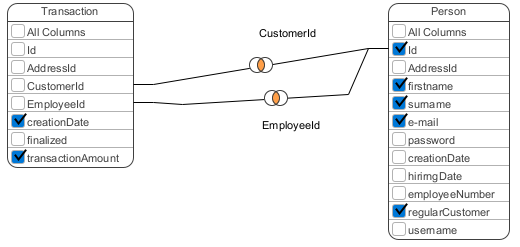
INNER JOIN BD\_2.Abstract\_Product a\_p ON p\_s.Abstract\_Product\_Id = a\_p.Id

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,

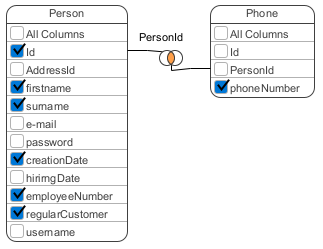
p.regular\_Customer,

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 obliczająca cenę danego produktu z uwzględnieniem podatku oraz promocji.

Zostanie zaimplementowana procedura obliczająca łączną wielkość transakcji.

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;

### 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

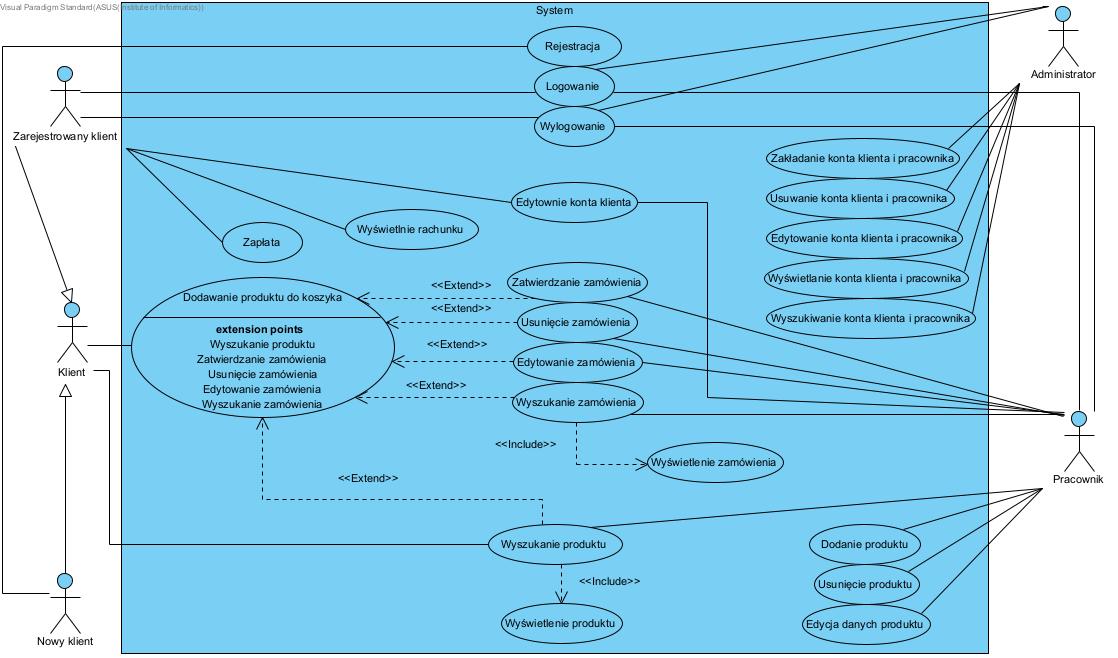
Obraz zawierający stół

Opis wygenerowany automatycznie

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

# Modelowanie aplikacji dostępowej

## Diagram przypadków użycia

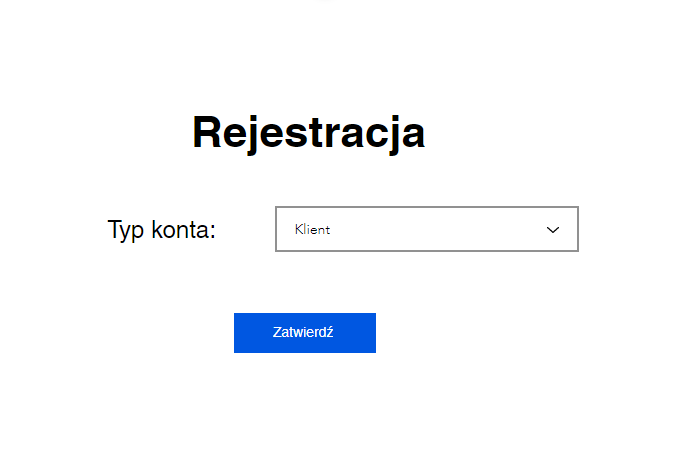


Rysunek 9 Diagram przypadków użycia systemu

## Interfejs graficzny



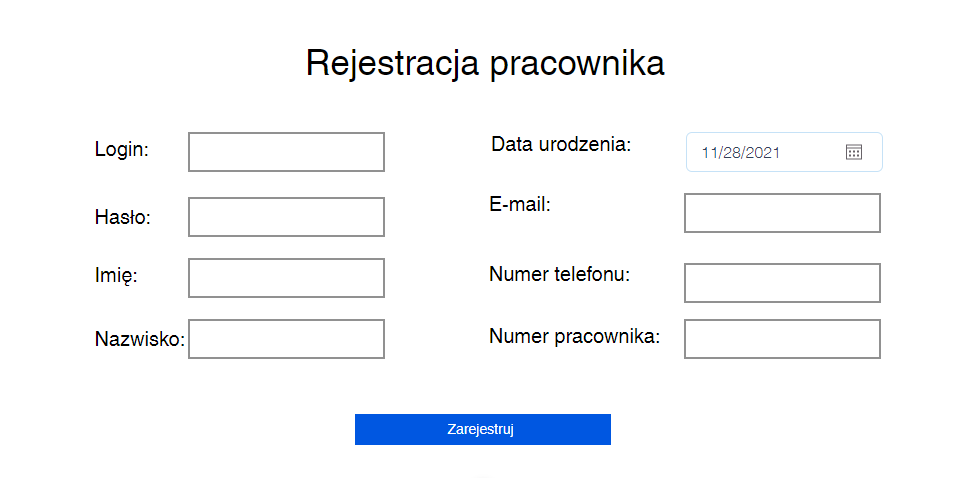
Rysunek 10 Strona startowa aplikacji



Rysunek 11 Wybór rodzaju konta do utworzenia

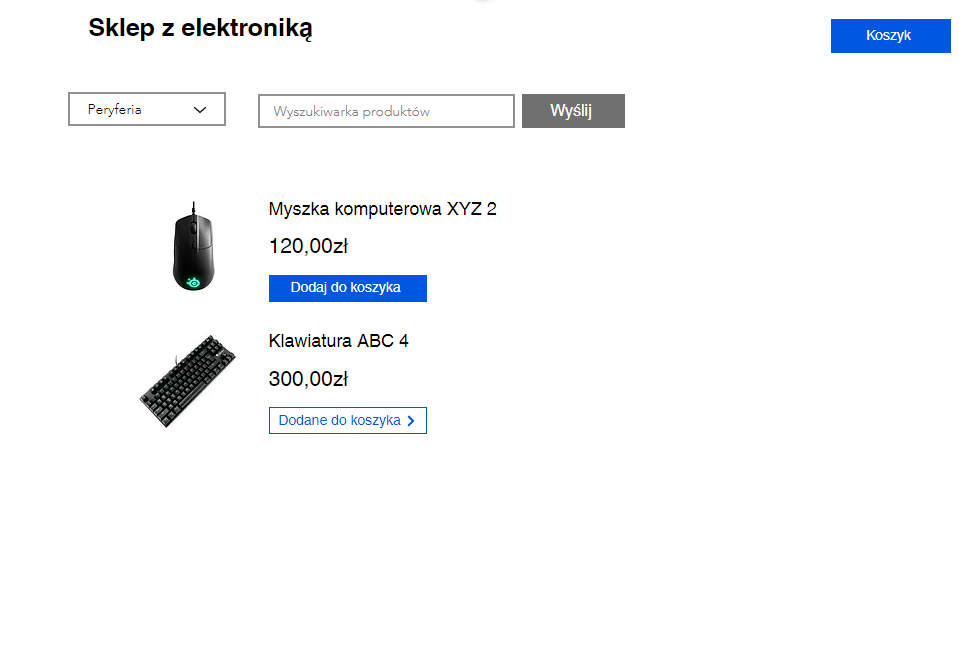


Rysunek 12 Rejestracja w systemie dla klienta

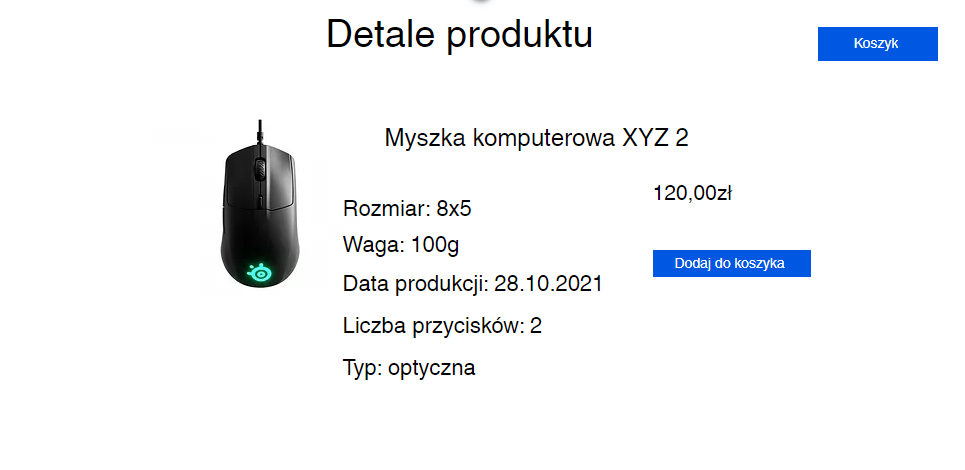


Rysunek 13 Rejestracja w systemie dla pracownika

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.*

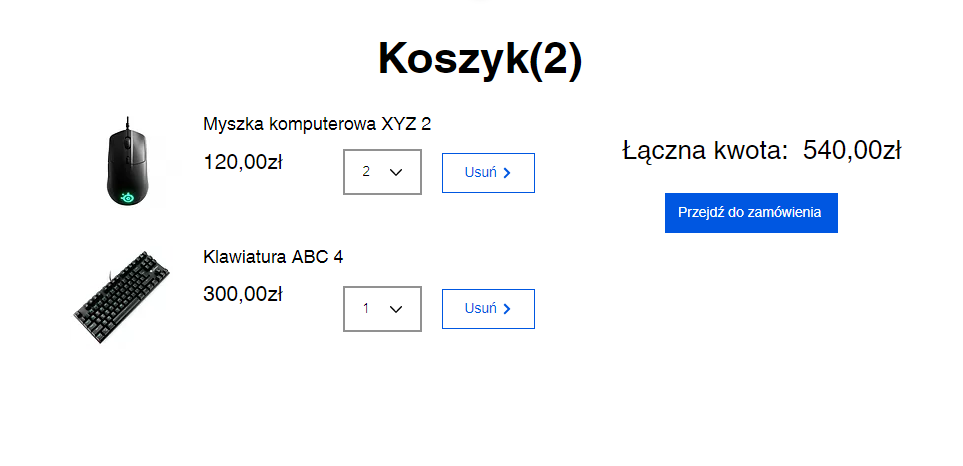


Rysunek 14 Widok zasobów sklepu z perspektywy zalogowanego/niezalogowanego klienta



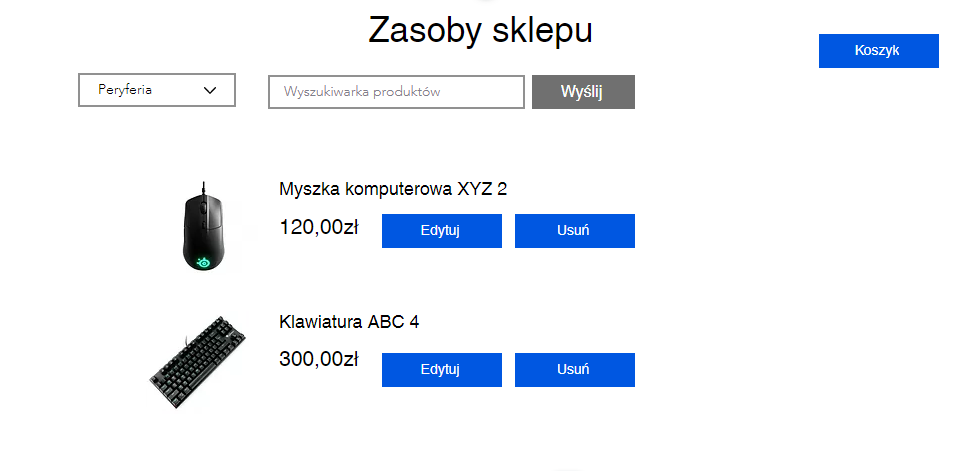
Rysunek 15 Detale produktu

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

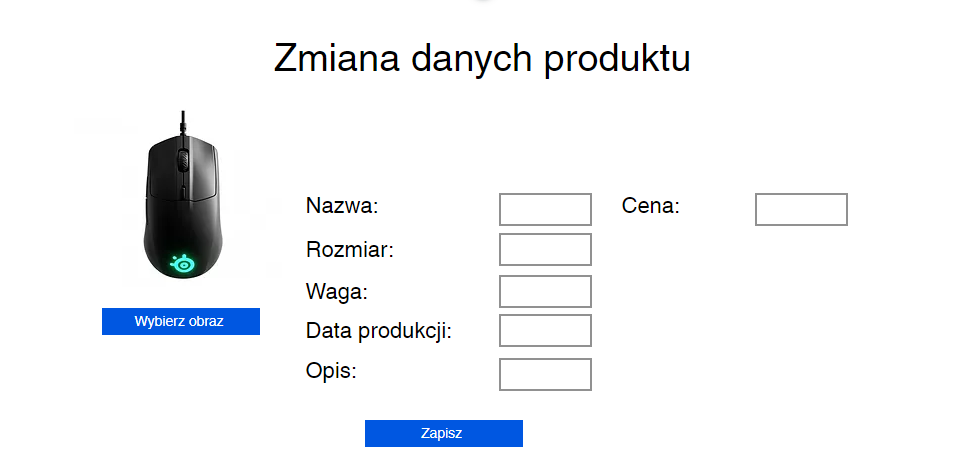


Rysunek 16 Widok koszyka

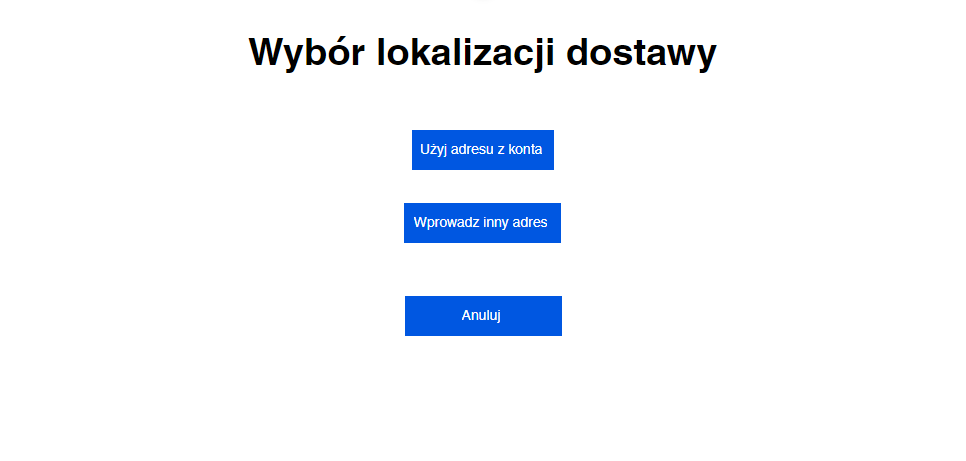
W koszyku można edytować liczbę produktów, usunąć liczbę danego produktu lub złożyć zamówienie.



Rysunek 17 Widok zasobów sklepu z perspektywy zalogowanego pracownika



Rysunek 18 Widok edycji danych produktu przez pracownika

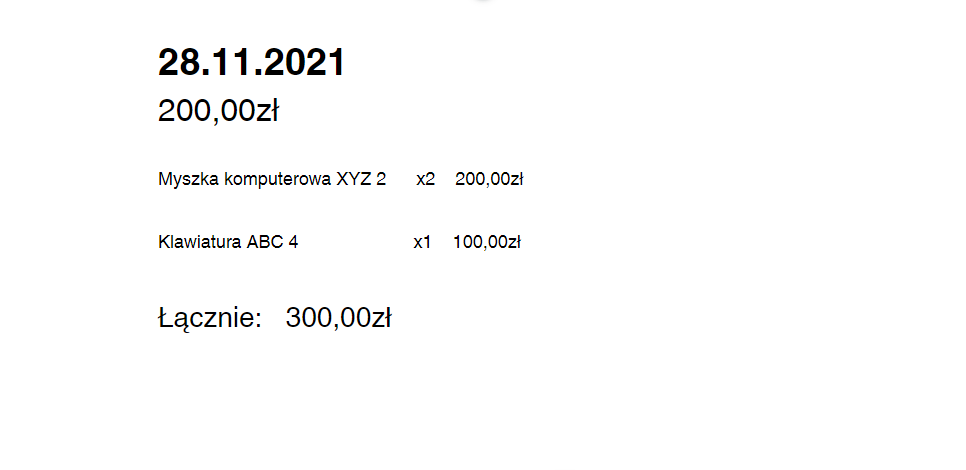


Rysunek 19 Wybór lokalizacji dostawy

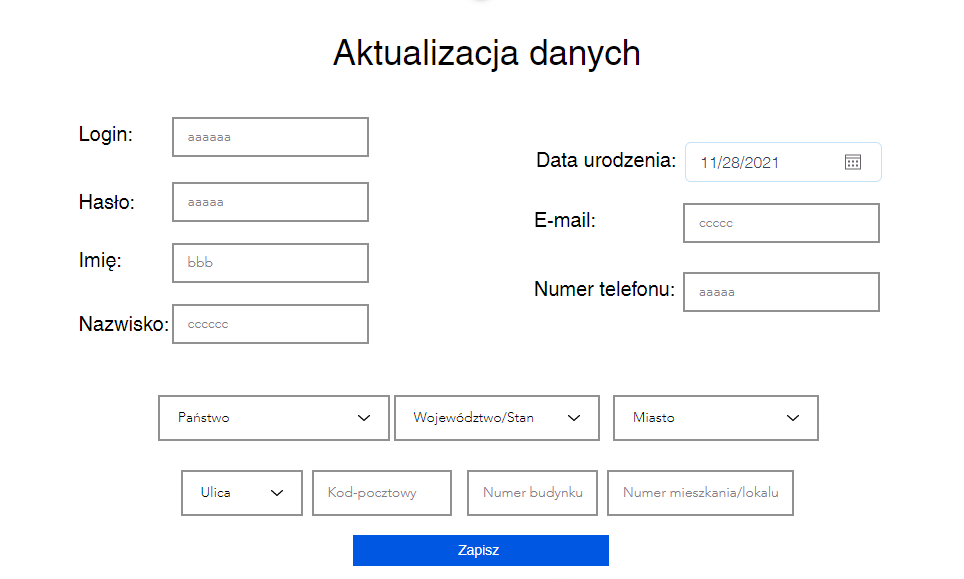


Rysunek 20 Historia transakcji klienta

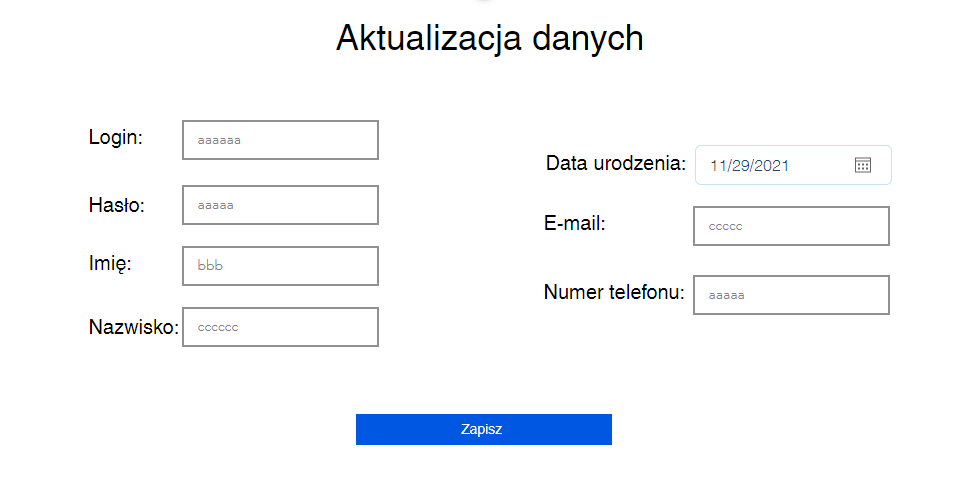
Klient ma dostęp do wszystkich wykonanych przez niego transakcjii.



Rysunek 21 Detale danej transakcji klienta



Rysunek 22 Aktualizacja danych klienta



Rysunek 23 Aktualizacja danych pracownika

## Metoda podłaczania 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")

}

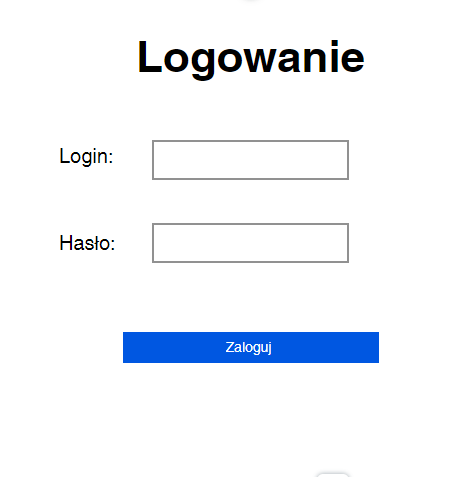
catch(Exception e){

...

}

## 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.



Rysunek 24 Okno logowania



Rysunek 25 Udane logowanie



Rysunek 26 Nieudane logowanie

1. **Implementacja systemu baz danych**
   1. **Tworzenie tabel i definiowanie ograniczeń**
   2. **Implementacja mechanizmów przetwarzania danych**
   3. **Implementacja uprawnień i innych zabezpieczeń**
   4. **Testowanie bazy danych na przykładowych danych**