

**UNIwersYTET RZESZOWSKI**  
**WYDZIAŁ NAUK ŚCISŁYCH I TECHNICZNYCH**  
**INSTYTUT INFORMATYKI**



*Wojciech Knapik*  
134923

*Informatyka*

*System zarządzania Biurem Podróży napisany w języku Java*

Praca projektowa

Praca wykonana pod kierunkiem  
mgr Ewa Żesławska

Rzeszów 2025



## **Spis treści**

# 1. Wprowadzenie

Niniejszy dokument stanowi dokumentację techniczną projektu desktopowej aplikacji wspomagającej pracę biura podróży. System pozwala użytkownikom na przeglądanie ofert wycieczek, dokonywanie rezerwacji oraz zarządzanie danymi w przyjaznym, graficznym interfejsie. Aplikacja została wykonana w języku Java z wykorzystaniem technologii Swing, JDBC oraz relacyjnej bazy danych PostgreSQL.

## 1.1. Cel i zakres projektu

Celem projektu było stworzenie funkcjonalnej aplikacji, która umożliwia:

- przeglądanie i filtrowanie dostępnych ofert wycieczek według daty i ceny,
- rejestrację oraz logowanie użytkowników wraz z walidacją danych,
- dokonywanie i przeglądanie rezerwacji,
- administrowanie danymi użytkowników i ofert przez dedykowany panel administracyjny.

Projekt obejmuje również przygotowanie szczegółowej dokumentacji wykonawczej w systemie  $\text{\LaTeX}$ .

## 1.2. Wymagania systemowe

Minimalne środowisko niezbędne do uruchomienia aplikacji obejmuje:

- system operacyjny: Windows 10 lub nowszy,
- zainstalowane środowisko Java Development Kit 17 lub wyższe,
- relacyjną bazę danych PostgreSQL 15 wraz z narzędziem pgAdmin 4,
- zainstalowany sterownik PostgreSQL JDBC,
- minimum 4 GB pamięci operacyjnej RAM.

## 1.3. Zastosowane technologie

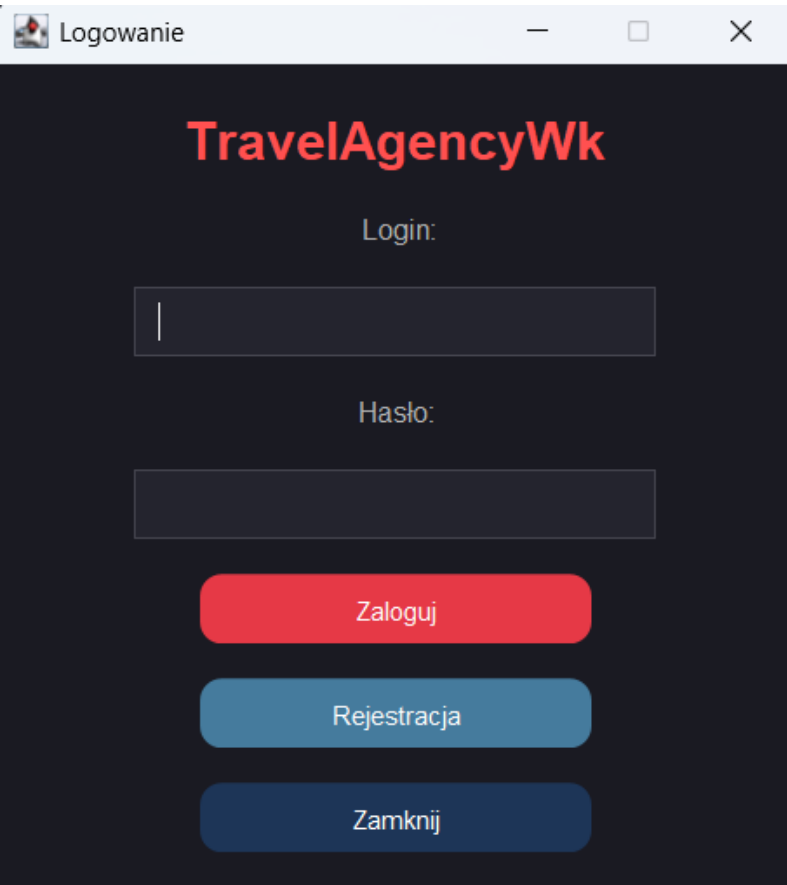
W projekcie wykorzystano następujące rozwiązania technologiczne:

- **Java SE 17** — do implementacji logiki aplikacji,
- **Swing** — do tworzenia graficznego interfejsu użytkownika,
- **JDBC** — jako warstwę połączeniową z bazą danych,
- **PostgreSQL 15** — jako system zarządzania bazą danych,
- $\text{\LaTeX}$  — do opracowania dokumentacji technicznej.

## 2. Interfejs użytkownika i funkcjonalności

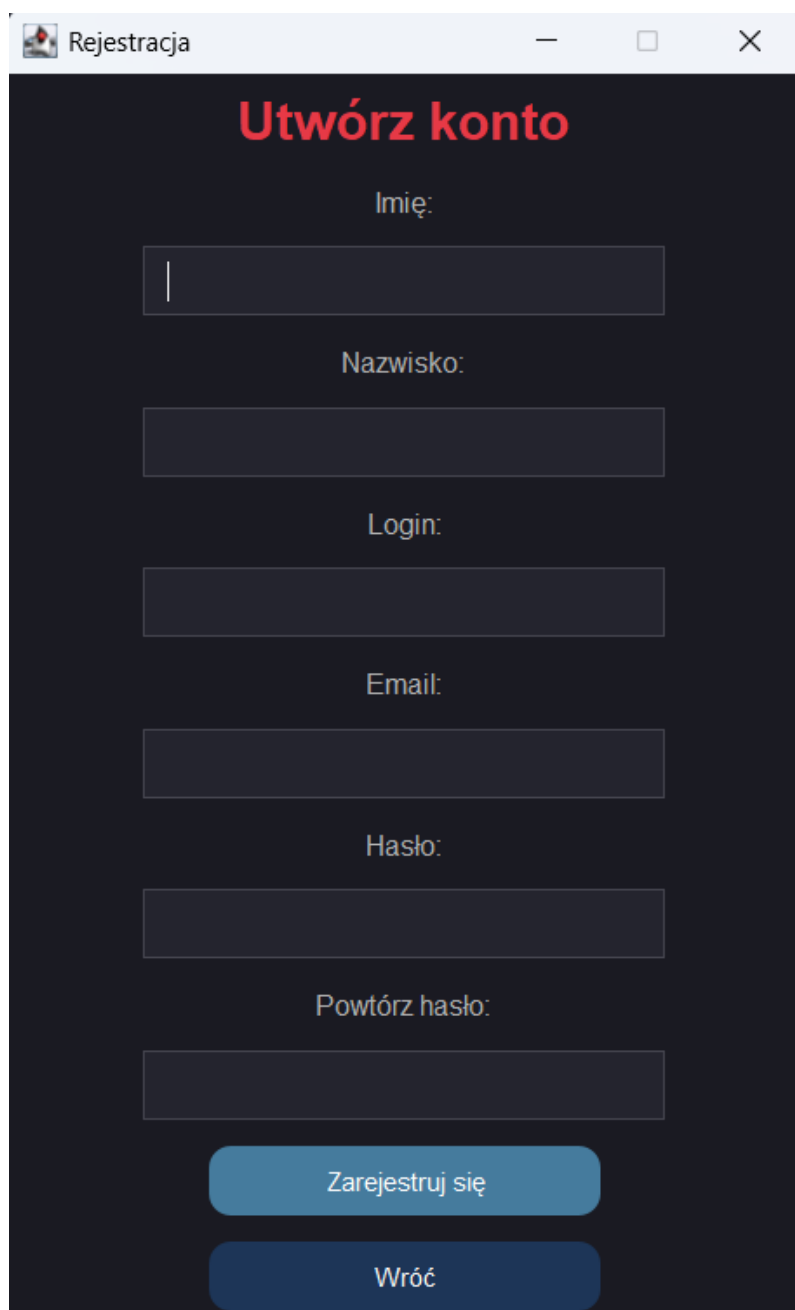
### 2.1. Logowanie i rejestracja użytkownika

Po uruchomieniu aplikacji wyświetlany jest ekran logowania (rys. ??). Użytkownik nieposiadający konta może przejść do rejestracji. Formularze logowania i rejestracji zawierają mechanizmy walidacyjne, weryfikujące poprawność adresu e-mail, długość oraz powtórzenie hasła, a także unikalność loginu na poziomie bazy danych.



The screenshot shows a web application window titled "Logowanie". The main content area has a dark blue background. At the top, the text "TravelAgencyWk" is displayed in a large, red, sans-serif font. Below this, there are three labels: "Login:", "Hasło:", and a third label that is not explicitly shown but corresponds to the third input field. Each label is followed by a dark gray rectangular input field. Below the input fields, there are three buttons stacked vertically. The first button is red and labeled "Zaloguj". The second button is blue and labeled "Rejestracja". The third button is dark blue and labeled "Zamknij".

Rys. 2.1. Ekran logowania

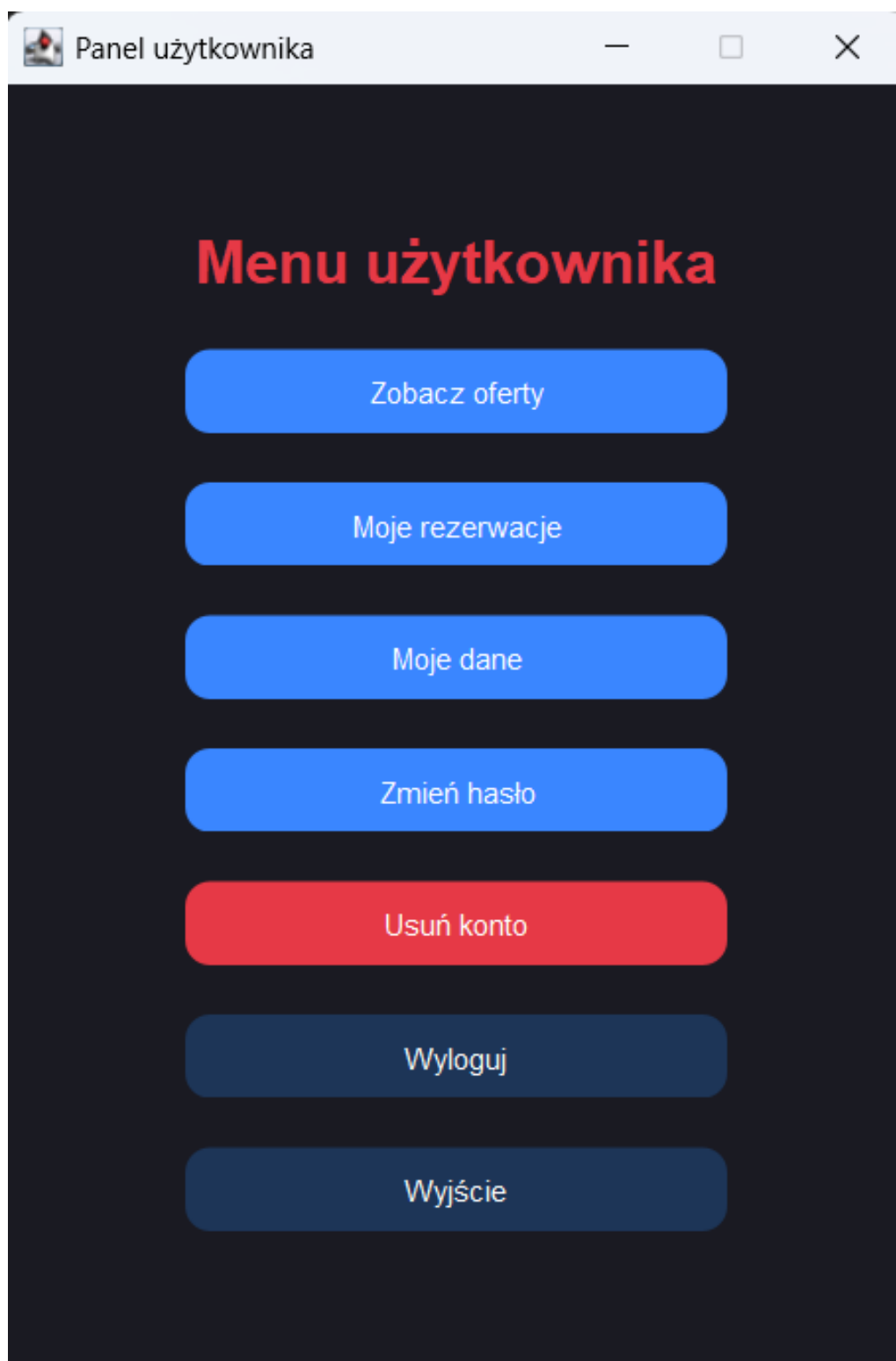


The image shows a web browser window titled "Rejestracja" (Registration). The main heading is "Utwórz konto" (Create account) in red. Below it are six input fields with labels: "Imię:" (First name), "Nazwisko:" (Last name), "Login:", "Email:", "Hasło:" (Password), and "Powtórz hasło:" (Repeat password). At the bottom are two buttons: "Zarejestruj się" (Register) in light blue and "Wróć" (Go back) in dark blue.

Rys. 2.2. Formularz rejestracyjny

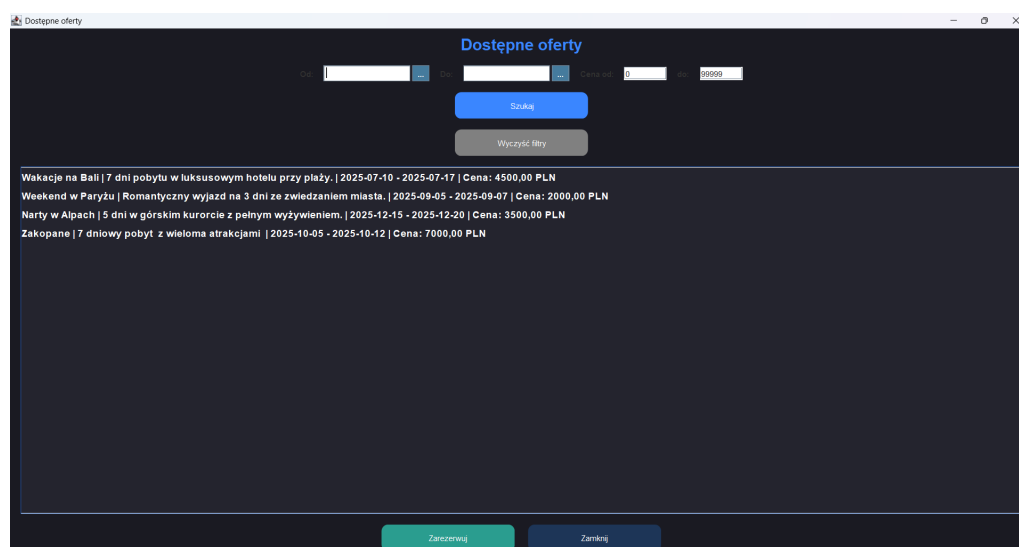
## 2.2. Panel użytkownika i oferty turystyczne

Po zalogowaniu użytkownik trafia do głównego panelu (rys. ??), w którym dostępne są wszystkie możliwe akcje — m.in. podgląd rezerwacji, przeglądanie ofert wycieczek i ich filtrowanie.



**Rys. 2.3.** Panel użytkownika

Wybór opcji „Zobacz oferty” otwiera listę wycieczek (rys. ??), które można przefiltrować według daty oraz zakresu cenowego. Domyślnie prezentowane są wszystkie aktywne oferty.



Rys. 2.4. Lista ofert z możliwością filtrowania

### 2.2.1. Filtrowanie według daty i ceny

Na rysunku ?? przedstawiono fragment klasy `OffersPanel.java`, realizującej filtrowanie listy ofert zgodnie z kryteriami cenowymi i daty:

Listing 2.1. Metoda filtrowania ofert po dacie i cenie

```

1 private void filterOffersByDate() {
2     LocalDate start = this.startDatePickerField.getDate();
3     LocalDate end = this.endDatePickerField.getDate();
4     boolean filterByDate = start != null && end != null;
5
6     BigDecimal min;
7     BigDecimal max;
8     try {
9         String minText = this.minPriceField.getText().trim();
10        String maxText = this.maxPriceField.getText().trim();
11        min = minText.isEmpty() ? BigDecimal.ZERO : new BigDecimal(minText);
12        max = maxText.isEmpty() ? BigDecimal.valueOf(Double.MAX_VALUE) : new
13        BigDecimal(maxText);
14    } catch (NumberFormatException var9) {
15        JOptionPane.showMessageDialog(this, "Nieprawidłowy format ceny!", "Błąd", 0)
16    };
17    return;
18
19    List<Offer> filtered = this.allOffers.stream().filter((o) -> !filterByDate || !o
20    .getStartDate().isBefore(start) && !o.getEndDate().isAfter(end)).filter((o) -> o
21    .getPrice().compareTo(min) >= 0 && o.getPrice().compareTo(max) <= 0).toList();
22    this.model.clear();
23    if (filtered.isEmpty()) {
24        this.model.addElement("Brak ofert w wybranym zakresie dat.");
25    } else {
26        filtered.forEach((o) -> this.model.addElement(String.format("%s | %s | %s -
27        %s | Cena: %.2f PLN", o.getName(), o.getDescription(), o.getStartDate(), o.
28        getEndDate(), o.getPrice())));
29    }
30 }

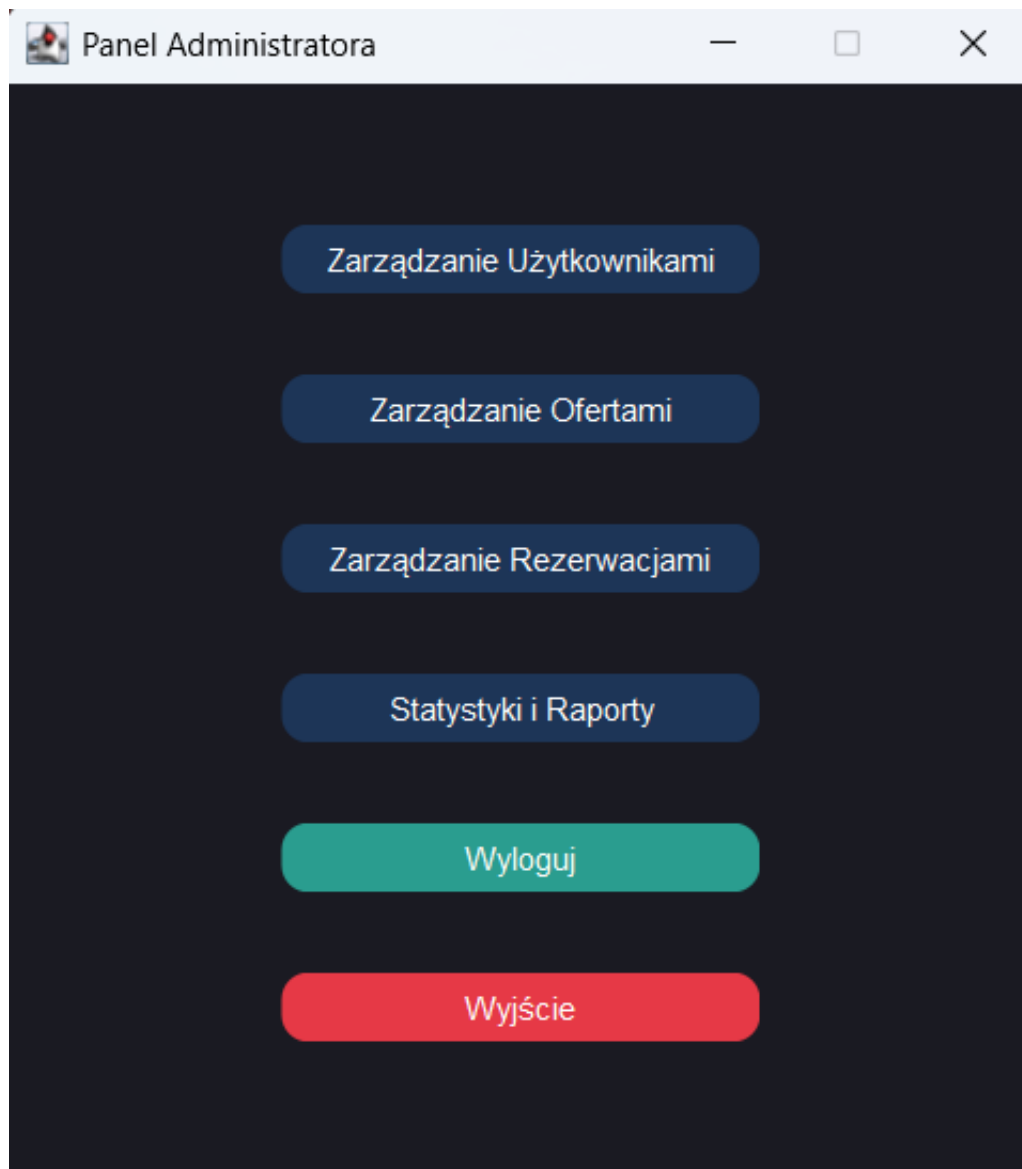
```



```
25  
26 }
```

## 2.3. Panel administratora

Użytkownik o uprawnieniach administratora po zalogowaniu uzyskuje dostęp do rozszerzonego menu systemu, umożliwiającego zarządzanie kontami użytkowników, ofertami oraz przeglądanie rezerwacji (rys. ??).



Rys. 2.5. Panel administratora z listą użytkowników

## 2.4. Walidacja danych

System przeprowadza walidację danych wejściowych zarówno po stronie klienta (Java Swing), jak i po stronie bazy danych. Przykładowe mechanizmy walidacyjne obejmują:

- sprawdzenie poprawności adresu e-mail,

- weryfikację długości oraz zgodności hasła,
- unikalność loginu użytkownika (na poziomie bazy),
- poprawność danych liczbowych (np. zakres cenowy oferty).

## 3. Struktura systemu i baza danych

### 3.1. Architektura aplikacji

Aplikacja została opracowana zgodnie z wzorcem projektowym MVC (Model–View–Controller), co zapewnia przejrzysty podział odpowiedzialności pomiędzy poszczególne komponenty systemu:

- **Model** – klasy reprezentujące struktury danych (np. użytkownik, oferta, rezerwacja),
- **View** – komponenty interfejsu graficznego zbudowane w oparciu o bibliotekę Swing,
- **Controller** – klasy pośredniczące pomiędzy logiką aplikacji, bazą danych a interfejsem graficznym.

### 3.2. Baza danych

System przechowuje dane w relacyjnej bazie **PostgreSQL 15**. Projekt bazy został przygotowany w środowisku pgAdmin 4, a komunikacja z aplikacją odbywa się za pośrednictwem sterownika PostgreSQL JDBC w wersji 42.7.5.

#### 3.2.1. Tabela **users**

Tabela przechowuje podstawowe informacje rejestracyjne użytkowników:

Kolumna	Typ	Opis
id	SERIAL PRIMARY KEY	ID użytkownika
login	VARCHAR(30) UNIQUE	Nazwa logowania
email	VARCHAR(50) UNIQUE	Adres e-mail
password	VARCHAR(100)	Zaszyfrowane hasło
role	VARCHAR(10)	Rola systemowa (user lub admin)

**Tabela 3.1.** Struktura tabeli **users**

#### 3.2.2. Tabela **offers**

Zawiera dane związane z ofertami turystycznymi:

Kolumna	Typ	Opis
id	SERIAL PRIMARY KEY	ID oferty
title	VARCHAR(100)	Nazwa wycieczki
description	TEXT	Szczegóły oferty
price	DECIMAL(10,2)	Cena brutto
start_date	DATE	Data rozpoczęcia
end_date	DATE	Data zakończenia

**Tabela 3.2.** Struktura tabeli **offers**

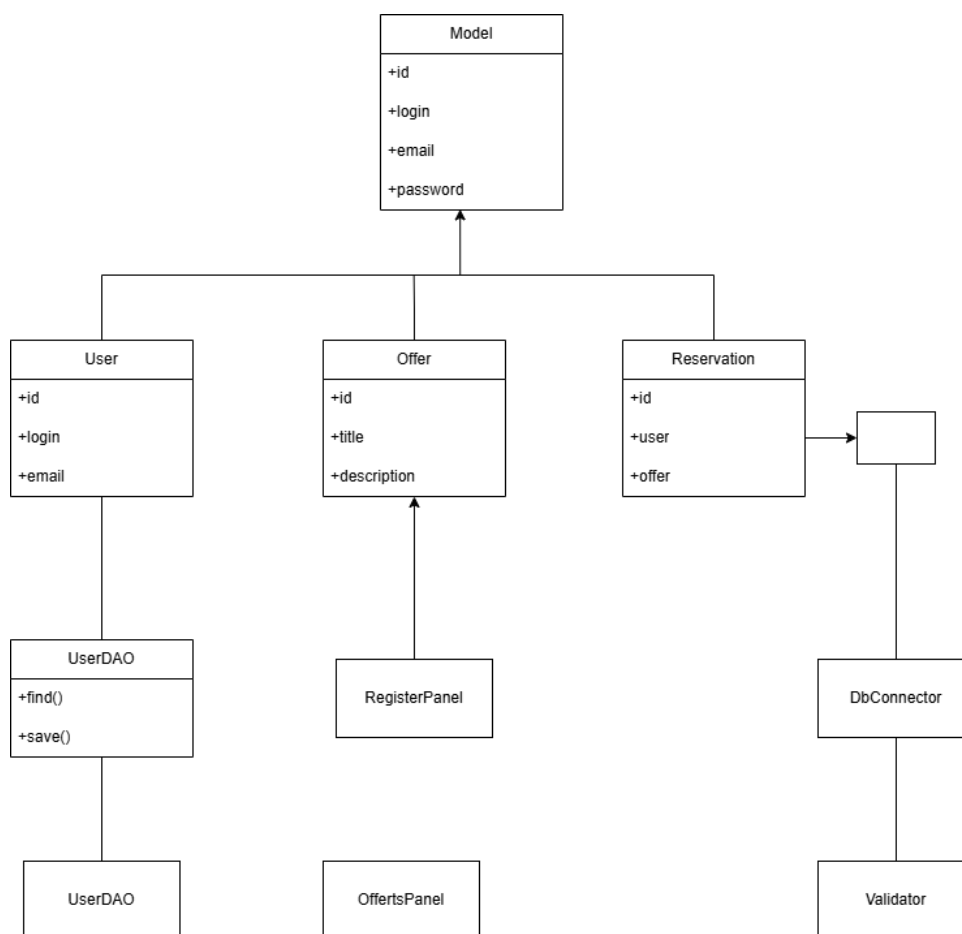


### 3.4. Organizacja pakietów i klas aplikacji

Struktura katalogowa projektu została podzielona zgodnie z zasadami separacji odpowiedzialności:

- model – klasy reprezentujące dane: `User.java`, `Offer.java`, `Reservation.java`,
- dao – warstwa dostępu do danych: `UserDAO.java`, `OfferDAO.java`,
- gui – komponenty interfejsu użytkownika: `LoginPanel.java`, `RegisterPanel.java`, `OffersPanel.java`,
- util – klasy pomocnicze: `DbConnector.java`, `Validator.java`.

Uproszczony diagram klas zaprezentowano na rysunku ??.



Rys. 3.2. Diagram klas aplikacji

## 4. Harmonogram, testy i podsumowanie

### 4.1. Harmonogram realizacji projektu

Projekt został zrealizowany w okresie od maja do czerwca 2025 roku. Całość prac została podzielona na trzy główne etapy:

- **maj 2025** — zaprojektowanie struktury bazy danych oraz graficznego interfejsu użytkownika,
- **maj–czerwiec 2025** — implementacja logiki aplikacji oraz integracja z bazą danych PostgreSQL,
- **czerwiec 2025** — przeprowadzenie testów, wprowadzenie poprawek oraz przygotowanie dokumentacji technicznej.

Wizualny przebieg realizacji przedstawiono na diagramie Gantta (rys. ??).

	Maj			
	Tydzień 1	Tydzień 2	Tydzień 3	Czerwiec
Analiza/zaprojektowanie struktury bazy danych				
Implementacja interfejsu graficznego w Swing				
Testowanie, dokumentacja, poprawki	Testowanie, dokumentacja, poprawki			

Rys. 4.1. Diagram Gantta przedstawiający harmonogram projektu

### 4.2. Repozytorium i wersjonowanie

Do kontroli wersji zastosowano system `Git`, natomiast kod źródłowy przechowywano w publicznym repozytorium serwisu GitHub:

- **Repozytorium:** [https://github.com/Arioch2/Biuro\\_Podr-y](https://github.com/Arioch2/Biuro_Podr-y)
- **Główna gałąź:** `main`

Rozwiązanie to umożliwia łatwą kontrolę nad postępem prac oraz historią zmian.

## 4.3. Testowanie aplikacji

Testy zostały przeprowadzone zarówno manualnie, jak i poprzez proste testy jednostkowe dla wybranych fragmentów logiki biznesowej. Sprawdzone poprawność działania formularzy, walidacji danych oraz połączenia z bazą danych.

### 4.3.1. Walidacja danych wejściowych

Walidacja danych odbywa się głównie po stronie klienta z wykorzystaniem klasy `Validator.java`. Przykładowy fragment tej klasy przedstawiono na listingu ??.

**Listing 4.1.** Fragment klasy `Validator`

```
1 package Services;
2
3 import java.util.regex.Pattern;
4
5 public class Validator {
6     private static final String EMAIL_REGEX = "^[\\w-\\.]+@([\\w-]+\\.){1,4}$";
7
8     public Validator() {
9     }
10
11     public static boolean isValidEmail(String email) {
12         return Pattern.matches("^[\\w-\\.]+@([\\w-]+\\.){1,4}$", email);
13     }
14 }
```

### 4.3.2. Połączenie z bazą danych

Kluczową rolę w obsłudze połączenia z bazą danych pełni klasa `DatabaseConnector`. Listing ?? prezentuje fragment tej klasy, odpowiedzialny za inicjalizację połączenia JDBC.

**Listing 4.2.** Fragment klasy `DatabaseConnector`

```
1 public class DatabaseConnector {
2     private static final String URL = "jdbc:postgresql://localhost:5432/postgres";
3     private static final String USER = "postgres";
4     private static final String PASSWORD = "qwerty";
5     private static String currentUserLogin;
6
7     public DatabaseConnector() {
8     }
9
10    public static Connection connect() {
11        try {
12            return DriverManager.getConnection("jdbc:postgresql://localhost:5432/
13postgres", "postgres", "qwerty");
14        } catch (SQLException e) {
15            e.printStackTrace();
16            return null;
17        }
18    }
19 }
```

## 4.4. Podsumowanie i wnioski

Zaprojektowana i zaimplementowana aplikacja spełnia wszystkie zakładane cele funkcjonalne. Użytkownik końcowy ma możliwość rejestracji, logowania, przeglądania i rezerwacji ofert. Z kolei administrator systemu otrzymuje dostęp do rozszerzonych funkcji zarządzania.

Potencjalne kierunki dalszego rozwoju systemu obejmują:

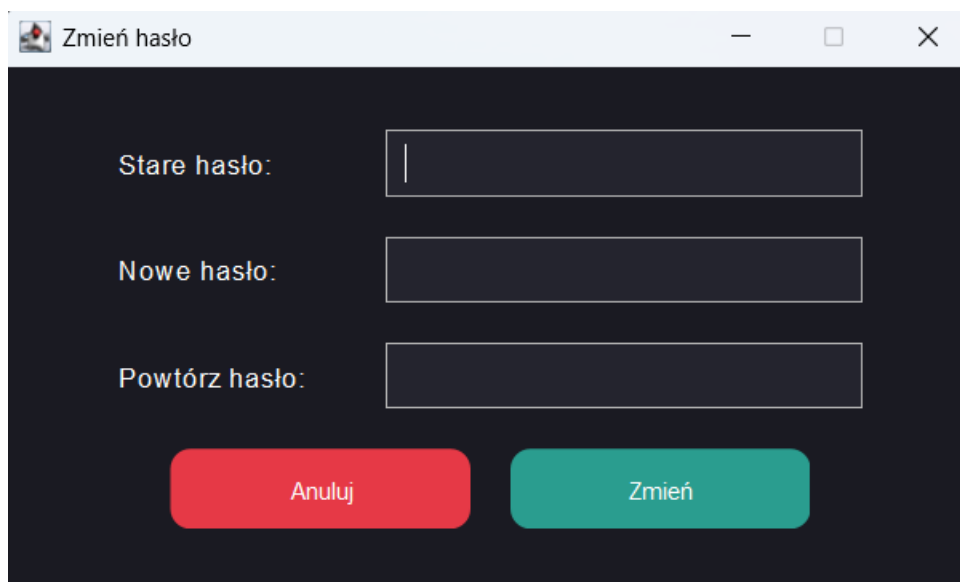
- implementację mechanizmu wysyłania potwierdzeń e-mailowych po dokonaniu rezerwacji,
- rozszerzenie systemu o możliwość eksportowania danych do plików PDF,
- integrację z zewnętrznymi usługami, takimi jak mapy lub prognozy pogody,
- dostosowanie interfejsu użytkownika do urządzeń mobilnych.



## 4.5. Pozostałe widoki aplikacji

Poza głównym panelem użytkownika i administracyjnym menu systemu, aplikacja udostępnia szereg dodatkowych interfejsów, wspierających funkcjonalność związaną z kontem użytkownika oraz zarządzaniem systemem.

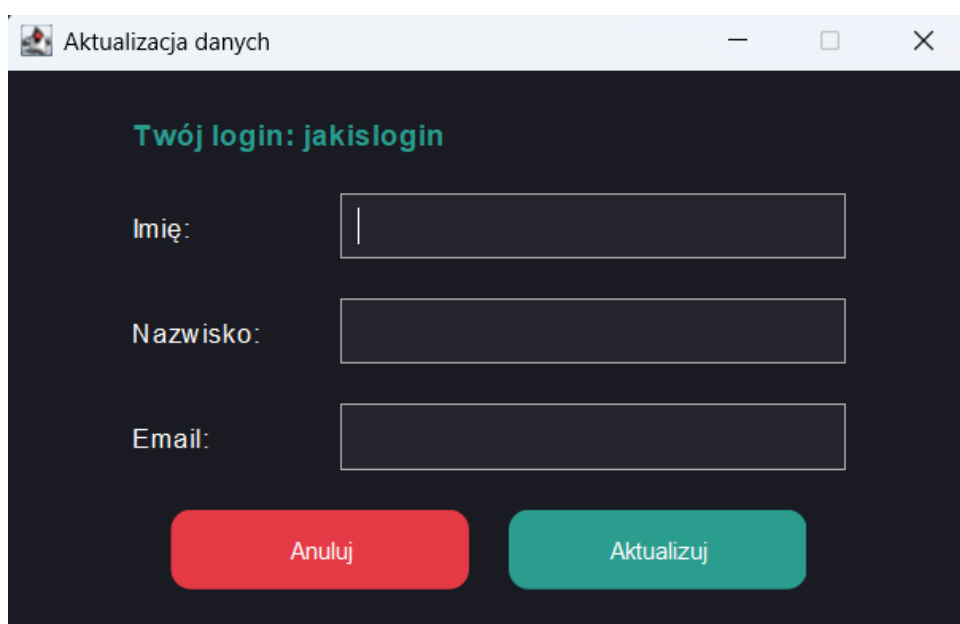
Na rysunku ?? przedstawiono panel umożliwiający zmianę hasła użytkownika.



Panel zmiany hasła użytkownika. Okno ma tytuł "Zmień hasło". Wewnątrz znajdują się trzy pola tekstowe: "Stare hasło:", "Nowe hasło:" i "Powtórz hasło:". Poniżej pól znajdują się dwa przyciski: "Anuluj" (czerwony) i "Zmień" (zielony).

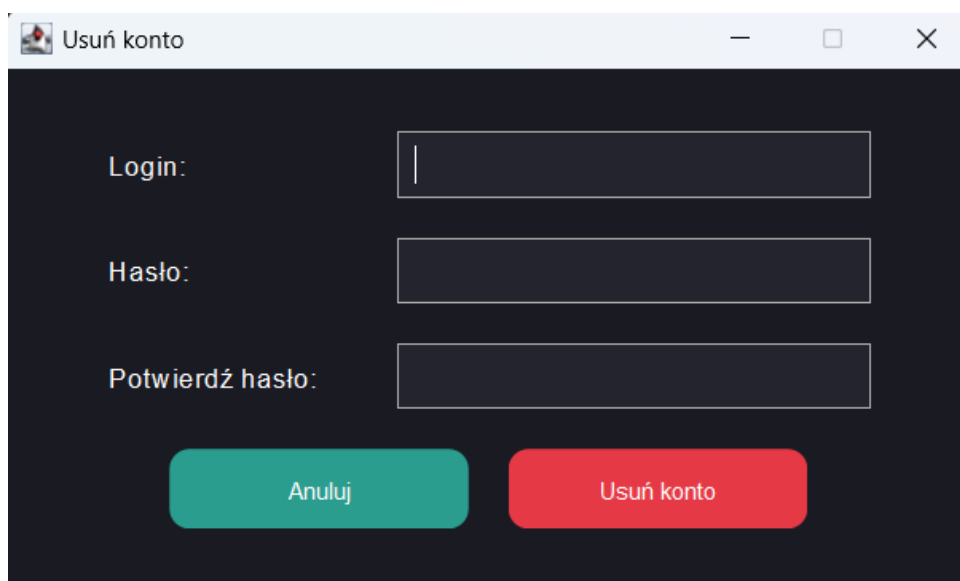
Rys. 4.2. Panel zmiany hasła użytkownika

Dane użytkownika mogą być edytowane za pośrednictwem dedykowanego formularza (rys. ??), a konto może zostać usunięte w panelu z rysunku ??.



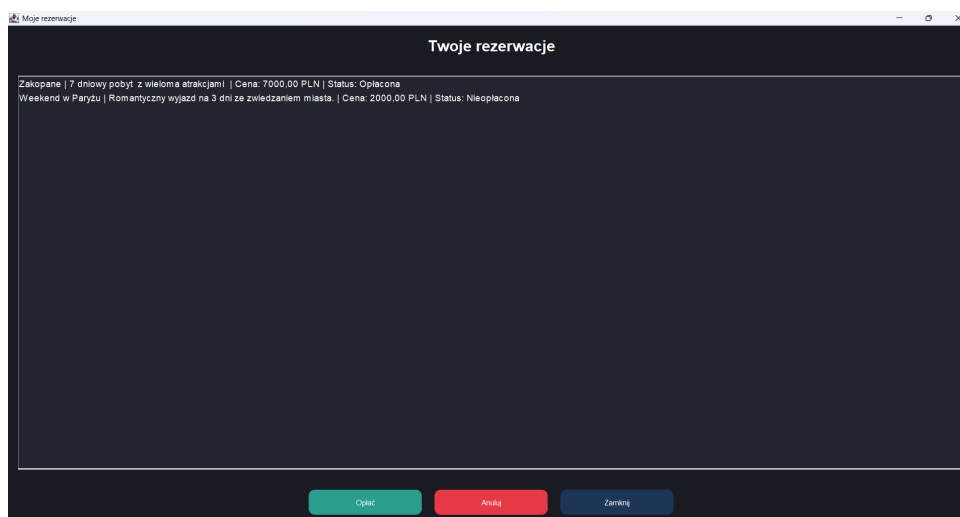
Panel edycji danych użytkownika. Okno ma tytuł "Aktualizacja danych". Wewnątrz znajduje się komunikat "Twój login: jakislogin". Poniżej znajdują się trzy pola tekstowe: "Imię:", "Nazwisko:" i "Email:". Poniżej pól znajdują się dwa przyciski: "Anuluj" (czerwony) i "Aktualizuj" (zielony).

Rys. 4.3. Panel edycji danych użytkownika



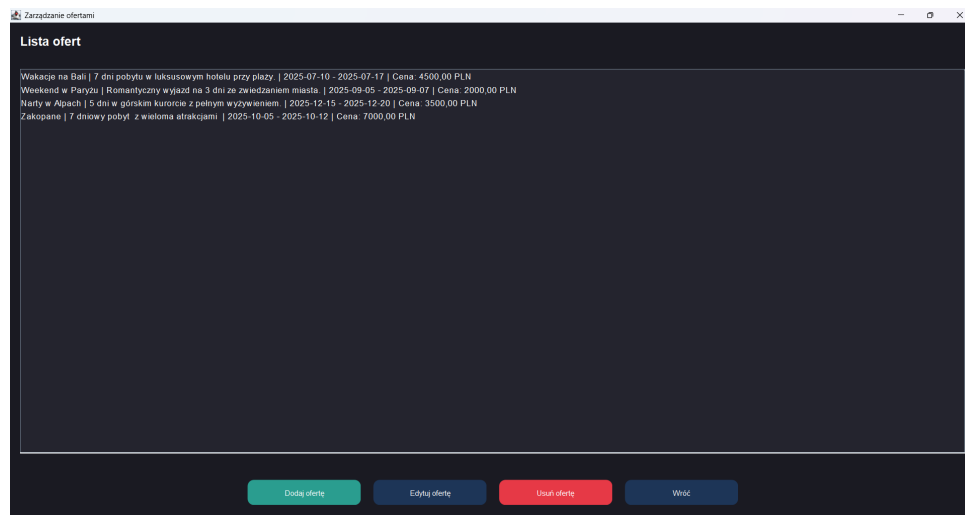
Rys. 4.4. Panel usuwania konta

Lista dokonanych rezerwacji dostępna jest z poziomu panelu użytkownika (rys. ??).

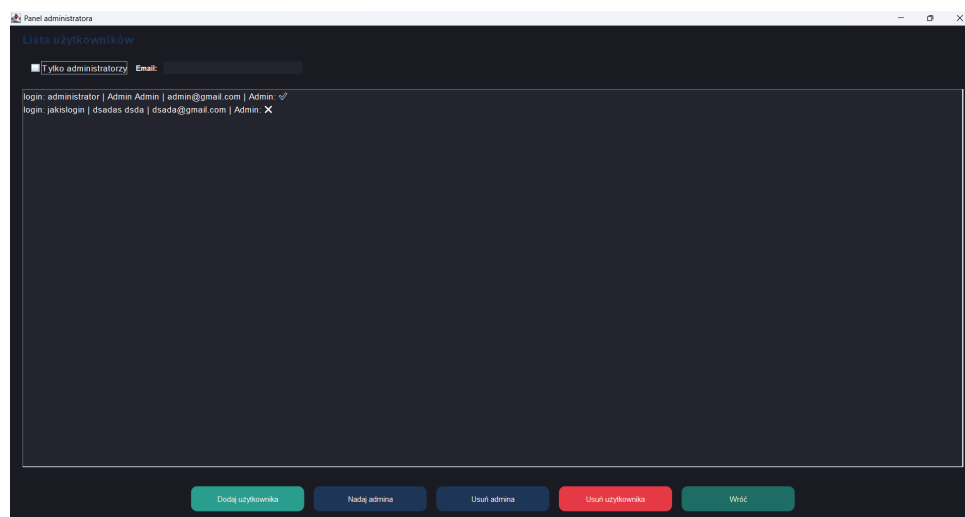


Rys. 4.5. Panel rezerwacji użytkownika

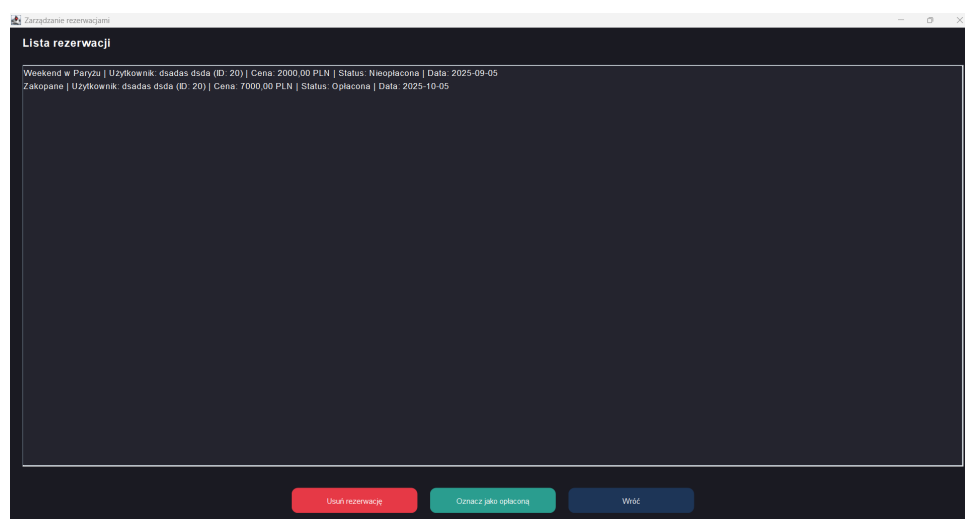
Administrator posiada również dostęp do rozszerzonych interfejsów zarządzania systemem. Na rysunkach ??–?? zaprezentowano odpowiednio panele zarządzania ofertami, użytkownikami, rezerwacjami oraz statystykami.



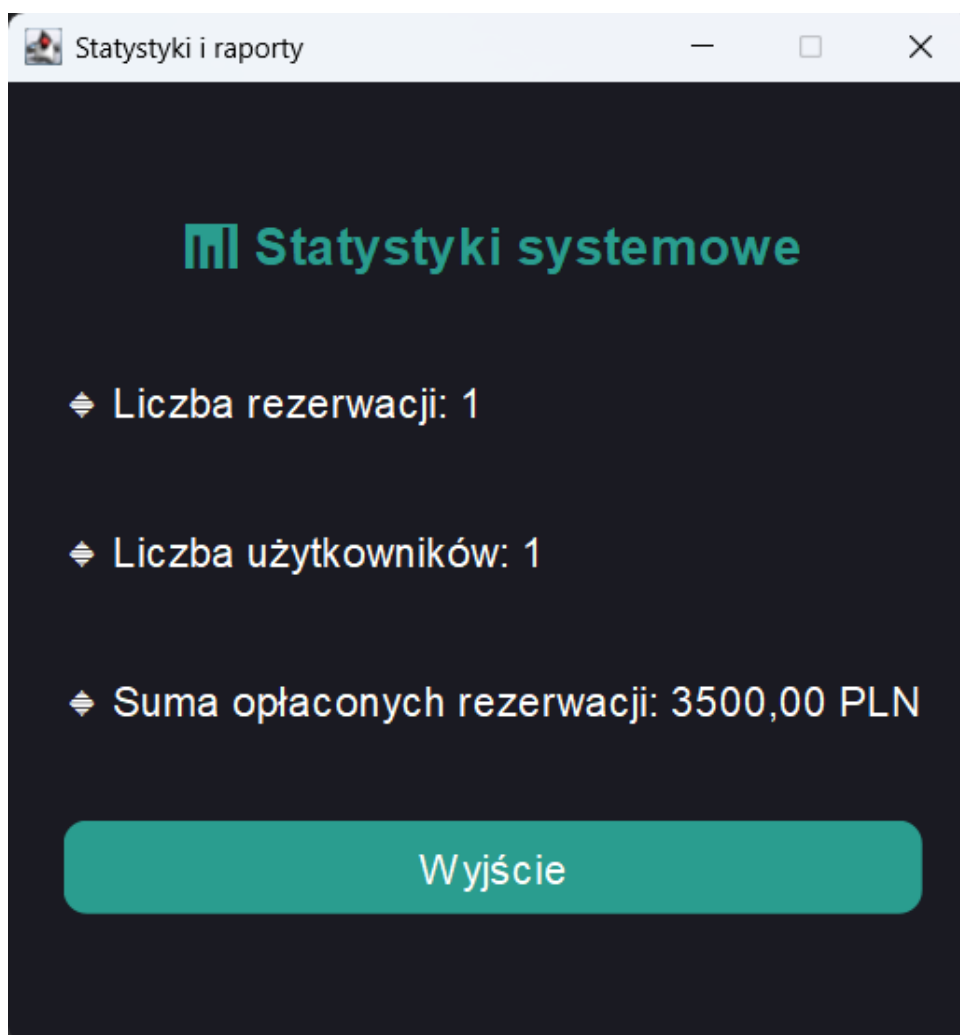
Rys. 4.6. Panel zarządzania ofertami (administrator)



Rys. 4.7. Panel zarządzania użytkownikami (administrator)



Rys. 4.8. Panel zarządzania rezerwacjami (administrator)



**Rys. 4.9.** Panel statystyk systemowych — liczba rezerwacji i aktywnych użytkowników

## **Bibliografia**

# Spis rysunków

## **Spis tabel**

## Spis listingów

2.1	Metoda filtrowania ofert po dacie i cenie . . . . .	10
4.1	Fragment klasy <code>Validator</code> . . . . .	17
4.2	Fragment klasy <code>DatabaseConnector</code> . . . . .	17



Załącznik nr 2 do Zarządzenia nr 228/2021 Rektora Uniwersytetu Rzeszowskiego z dnia 1 grudnia 2021 roku w sprawie ustalenia procedury antyplagiatowej w Uniwersytecie Rzeszowskim

### **OŚWIADCZENIE STUDENTA O SAMODZIELNOŚCI PRACY**

.....**Wojciech Knapik**.....  
Imię (imiona) i nazwisko studenta

Wydział Nauk Ścisłych i Technicznych

.....**Informatyka**.....  
Nazwa kierunku

.....**134923**.....  
Numer albumu

1. Oświadczam, że moja praca projektowa pt.: System zarządzania Biurem Podróży napisany w języku Java