UNIWERSYTET RZESZOWSKI

WYDZIAŁ NAUK ŚCISŁYCH I TECHNICZNYCH

INSTYTUT INFORMATYKI

Artur Jurkowski

134916

Informatyka

Projekt i implementacja systemu rezerwacji biletów lotniczych z wykorzystaniem języka Java i bazy danych SQLite

Praca projektowa

Praca wykonana pod kierunkiem
mgr inż.Ewa Żeszławska

Rzeszów 2025

Spis treści

0.1. Streszczenie

0.1.1. Streszczenie w języku polskim

System rezerwacji lotów to zaawansowana aplikacja umożliwiająca zarządzanie rezerwacjami, lotami i pasażerami. Aplikacja została zaimplementowana w języku Java z wykorzystaniem technologii Swing dla interfejsu użytkownika oraz bazy danych MySQL do przechowywania informacji o lotach, pasażerach oraz rezerwacjach. W projekcie wykorzystano klasy takie jak 'FlightDAO', 'PassengerDAO', 'ReservationDAO' do obsługi danych oraz technologie opisane w [1] oraz [2]. System oferuje dwie główne ścieżki interakcji: dla użytkowników (przeglądanie dostępnych lotów, rezerwacje, zarządzanie użytkownikami) oraz dla administratorów (zarządzanie lotami, pasażerami i rezerwacjami, analiza transakcji). Aplikacja zapewnia również system logowania i rejestracji użytkowników oraz funkcjonalność zarządzania bazą danych.

Kod źródłowy projektu jest dostępny w repozytorium GitHub: *https://github.com/Dzurek007/Repozytorium-Programowanie_Obiektowe*

0.1.2. Summary in English

The flight reservation system is an advanced application that enables the management of bookings, flights, and passengers. The application was implemented in Java using Swing for the user interface and MySQL database for storing information about flights, passengers, and reservations. The project utilizes classes such as 'FlightDAO', 'PassengerDAO', 'ReservationDAO' for handling data, as well as technologies described in [1] and [2]. The system offers two main interaction paths: for users (browsing available flights, making reservations, user management) and for administrators (managing flights, passengers, and reservations, transaction analysis). The application also provides a user login and registration system, as well as database management functionality.

The source code of the project is available in the GitHub repository: *https://github.com/Dzurek007/Repozytorium-Programowanie_Obiektowe*

0.2. Opis założeń projektu

0.2.1. Cel projektu

Głównym celem projektu było stworzenie systemu rezerwacji lotów, który:

- Symuluje rzeczywiste operacje związane z rezerwacjami lotów w biurze podróży, w tym operacje zarządzania lotami, pasażerami i rezerwacjami.
- Umożliwia łatwe przeglądanie dostępnych lotów przez użytkowników, dzięki interfejsowi graficznemu.
- Zapewnia wygodny sposób dokonywania rezerwacji oraz zarządzania rezerwacjami z poziomu panelu użytkownika.
- Umożliwia zarządzanie lotami i rezerwacjami przez użytkowników systemu (np. pasażerów).

0.2.2. Wymagania funkcjonalne

- Przeglądanie dostępnych lotów z podziałem na różne kierunki, daty i dostępność, co umożliwia panel użytkownika (klasa `FlightPanel.java`).

- Możliwość składania rezerwacji na wybrane loty przez pasażerów, zintegrowana z bazą danych rezerwacji (klasa `ReservationDAO.java`). (klasa `Reservation.java`).
- Rejestracja użytkowników i autentykacja (klasa `LoginPanel.java`, `UserDAO.java`).
- Zarządzanie stanem lotów (dodawanie nowych lotów, edytowanie, usuwanie) za pomocą klas `FlightDAO.java` i `Flight.java`.
- Zarządzanie danymi pasażerów (dodawanie, edytowanie, usuwanie) przez klasy `PassengerDAO.java`, `Passenger.java`.
- Generowanie raportów dotyczących rezerwacji i finansów (np. raporty transakcji, dostępność lotów) (klasa `ReservationDAO.java`).
- Obsługa rezerwacji poprzez odpowiednią logikę w klasach `Reservation.java` i `ReservationPanel.java`.

0.2.3. Wymagania niefunkcjonalne

- Wydajność: Czas odpowiedzi systemu na zapytania użytkownika powinien być poniżej 2 sekund. Zapewnienie szybkiego przetwarzania danych i interakcji z bazą danych.
- Bezpieczeństwo: Szyfrowanie danych użytkowników, ochrona danych transakcji oraz dostępność autentykacji użytkowników za pomocą bezpiecznych metod (klasy `UserDAO.java`, `LoginPanel.java`).
- Kompatybilność: Aplikacja działa na systemach Windows, Linux i MacOS z Java 8+. Zastosowanie Swing zapewnia wsparcie dla różnych platform.
- Użyteczność: Przejrzysty i intuicyjny interfejs użytkownika z wykorzystaniem Swing, w tym klasy `MainFrame.java`, `FlightPanel.java`, `ReservationPanel.java`.
- Niezawodność: Odporność systemu na błędy użytkowników, łatwe odzyskiwanie danych w przypadku awarii dzięki solidnemu zarządzaniu połączeniem z bazą danych (`DBConnection.java`).

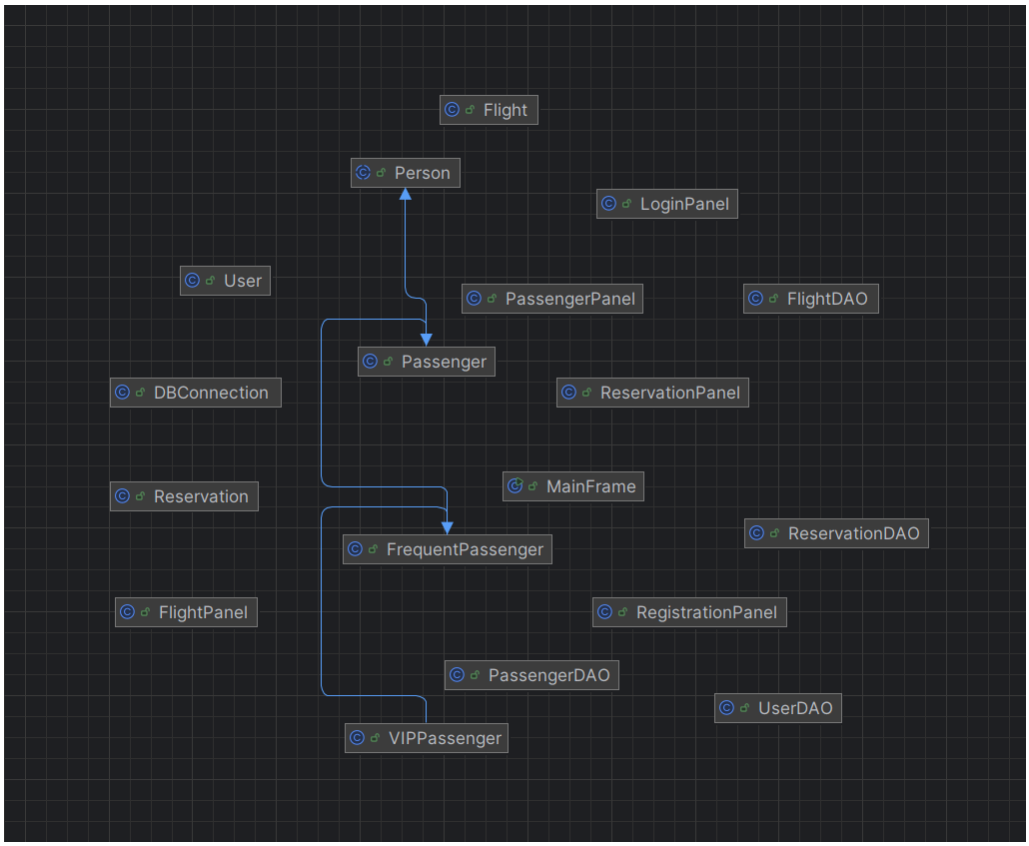
0.3. Opis struktury projektu

0.3.1. Architektura systemu

System został zbudowany w oparciu o wzorzec MVC (Model-Widok-Kontroler):

- **Model**: Baza danych SQLite, klasy dziedziny takie jak Lot, Rezerwacja, Pasażer, Użytkownik.
- **Widok**: Interfejs użytkownika zrealizowany w Swing, z odpowiednimi formularzami do przeglądania dostępnych lotów, rezerwacji oraz zarządzania kontem użytkownika.
- **Kontroler**: Logika biznesowa aplikacji, odpowiedzialna za obsługę rezerwacji, płatności oraz interakcje z bazą danych.

0.3.2. Diagram klas



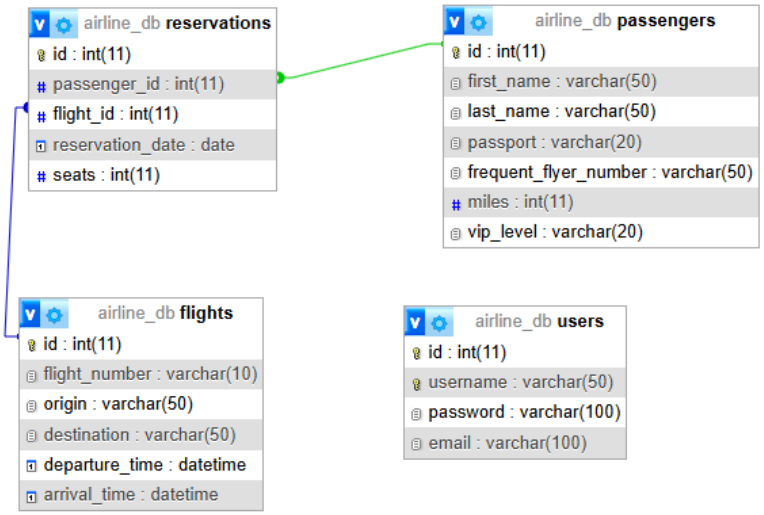
Rysunek 1. Diagram głównych klas systemu

Kluczowe klasy systemu:

- **MainFrame**: Główne okno aplikacji, które uruchamia interfejs użytkownika i umożliwia interakcję z systemem.
- **Flight**: Klasa reprezentująca dane o locie, takie jak numer lotu, miejsce początkowe, miejsce docelowe oraz godziny odlotu i przylotu.
- **Passenger**: Klasa reprezentująca dane pasażera, takie jak imię, nazwisko, numer paszportu oraz historia rezerwacji.
- **Reservation**: Klasa odpowiadająca za dane o rezerwacji, powiązana z pasażerem oraz lotem.
- **User**: Klasa reprezentująca użytkownika aplikacji (dane logowania, poziom uprawnień).
- **DatabaseManager**: Klasa odpowiedzialna za dostęp do bazy danych i wykonywanie operacji na tabelach.

0.3.3. Baza danych

System wykorzystuje bazę danych SQLite, w której przechowywane są dane o lotach, rezerwacjach, pasażerach i użytkownikach. Baza danych składa się z następujących tabel:



Rysunek 2. Diagram ERD bazy danych

0.4. Harmonogram realizacji projektu

Projekt był realizowany zgodnie z poniższym planem:

- **Analiza wymagań** Określenie funkcjonalności systemu oraz wymagań użytkowników.
- **Projekt interfejsu użytkownika** Opracowanie graficznego interfejsu użytkownika (GUI) w technologii Swing.
- **Implementacja podstawowej funkcjonalności** : Implementacja kluczowych funkcji systemu, takich jak rejestracja użytkowników i rezerwacja lotów.
- **Testowanie i debugowanie** Przeprowadzenie testów jednostkowych oraz integracyjnych w celu zapewnienia poprawności działania systemu.
- **Przygotowanie dokumentacji** Opracowanie dokumentacji projektowej i użytkowej, w tym opis systemu, kodu i testów.

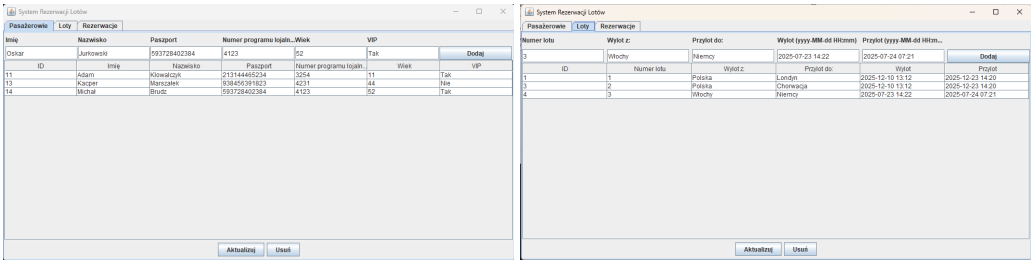
Kod źródłowy projektu jest dostępny w repozytorium GitHub: *https://github.com/Dzurek007/Repozytorium-Programowanie_Obiektowe*

0.5. Prezentacja warstwy użytkowej projektu

0.5.1. Interfejs użytkownika

System rezerwacji lotów oferuje trzy główne interfejsy:

- **Ekran logowania i rejestracji** - dla użytkowników
- **Ekran klienta** - do przeglądania dostępnych lotów, rezerwacji biletów
- **Panel klienta** - do zarządzania rezerwacjami oraz przeglądania danych

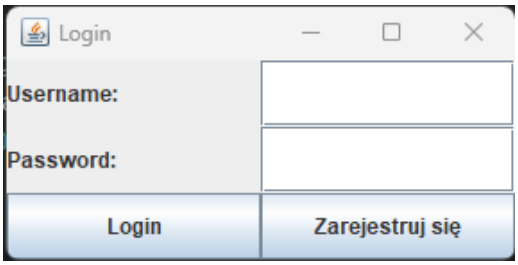


(a) Ekran główny - przegląd pasażerów (b) Ekran z listą lotów

Rysunek 3. Podstawowe ekrany systemu rezerwacji lotów

0.5.2. Interfejs logowania

System oferuje ekran logowania, na którym użytkownik wprowadza swój login oraz hasło. W przypadku braku konta, użytkownik może przejść do ekranu rejestracji, gdzie może stworzyć nowe konto.

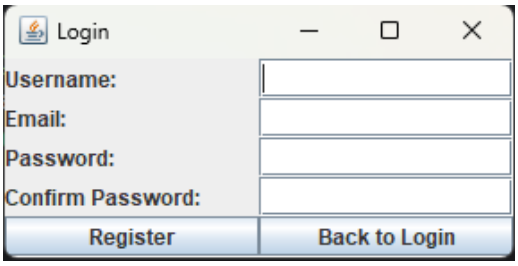


Rysunek 4. Ekran logowania

Po pomyślnym zalogowaniu, użytkownik zostaje przeniesiony do ekranu głównego, gdzie może przeglądać dostępne loty, rezerwować bilety oraz zarządzać swoimi danymi.

0.5.3. Rejestracja użytkownika

W przypadku braku konta, użytkownik może zarejestrować się poprzez formularz rejestracyjny, gdzie wprowadza dane takie jak login, adres e-mail, hasło oraz jego potwierdzenie.



Rysunek 5. Ekran rejestracji użytkownika

0.5.4. Przepływ użytkownika

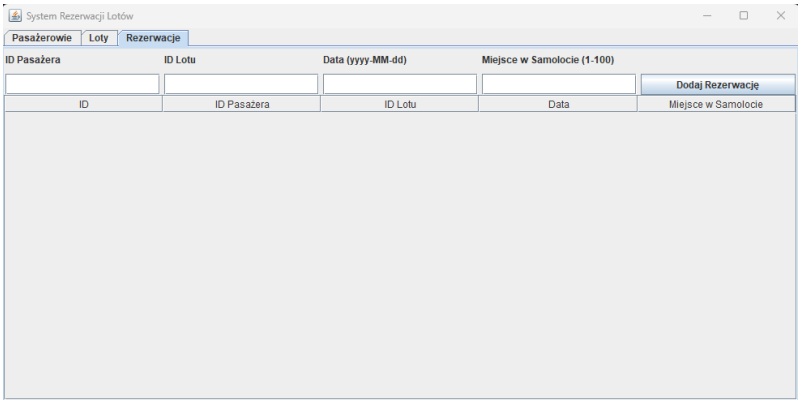
Typowy scenariusz użycia systemu:

1. Użytkownik uruchamia aplikację i loguje się na swoje konto.
2. Po zalogowaniu użytkownik przegląda dostępne loty.
3. Wybiera lot, uzupełnia dane pasażera, a następnie rezerwuje bilet.

4. Użytkownik może również edytować swoje dane lub sprawdzić historię rezerwacji.

0.5.5. Panel klienta

Panel klienta jest dostępny po zalogowaniu się na konto użytkownika. W panelu klienta użytkownik może zarządzać swoimi rezerwacjami, przeglądać historię oraz edytować swoje dane.



Rysunek 6. Panel klienta systemu rezerwacji

Funkcjonalności panelu klienta:

- **Zarządzanie rezerwacjami** - przeglądanie, edytowanie i anulowanie rezerwacji.
- **Przegląd danych osobowych** - możliwość edytowania danych pasażera.
- **Historia rezerwacji** - monitorowanie rezerwacji dokonanych przez użytkownika.

0.6. Testowanie systemu

0.6.1. Strategia testowania

System został przetestowany na trzech poziomach:

- Testy jednostkowe (JUnit) - testowanie poszczególnych metod w klasach.
- Testy integracyjne - testowanie współdziałania komponentów systemu.
- Testy systemowe - testowanie całej aplikacji jako całości.

0.6.2. Przykładowe scenariusze testowe

Tabela 1. Przykładowe przypadki testowe

Scenariusz	Oczekiwany wynik	Status
Logowanie użytkownika	Użytkownik zalogowany do systemu	PASS
Rezerwacja lotu	Rezerwacja poprawnie zapisana w systemie	PASS
Dodanie użytkownika	Poprawne dodanie i dodanie do bazy danych	PASS

0.6.3. Wyniki testów wydajnościowych

- Średni czas ładowania ekranu: 0.3s

- Maksymalna liczba równoległych rezerwacji: 10
- Średni czas przetwarzania rezerwacji: 0.4s

0.7. Przykłady dziedziczenia w systemie rezerwacji lotów

0.7.1. Klasa bazowa `Person` i klasy dziedziczące

W systemie rezerwacji lotów, klasa `Person` pełni rolę klasy bazowej dla wszystkich osób zaangażowanych w systemie (np. pasażerów). Klasa ta zawiera ogólne dane dotyczące osoby, takie jak imię, nazwisko oraz identyfikator. Na podstawie tej klasy zostały utworzone klasy dziedziczące, takie jak `Passenger` (Pasażer) i `VIPPassenger` (Pasażer VIP).

Kod klasy `Person`

Klasa `Person` zawiera podstawowe dane osobowe, takie jak identyfikator, imię i nazwisko, które są wspólne dla różnych typów użytkowników systemu.

Listing 1. Klasa Person

```
package com.example.airlinereservation.models;

public class Person {
    protected int id;
    protected String firstName;
    protected String lastName;

    public Person(int id, String firstName, String lastName) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
    }

    public int getId() { return id; }
    public String getFirstName() { return firstName; }
    public String getLastName() { return lastName; }
}
```

Kod klasy `FrequentPassenger`

Klasa `FrequentPassenger` dziedziczy po klasie `Person`, rozszerzając ją o właściwości związane z częstymi podróżnikami, takie jak numer frequent flyer i liczba mil.

Listing 2. Klasa FrequentPassenger

```
package com.example.airlinereservation.models;

public class FrequentPassenger extends Person {
    private String frequentFlyerNumber;
    private int miles;

    public FrequentPassenger(int id, String firstName, String lastName, String frequentFlyerNumber, int miles)
        super(id, firstName, lastName);
        this.frequentFlyerNumber = frequentFlyerNumber;
        this.miles = miles;
    }
```

```
    public String getFrequentFlyerNumber() { return frequentFlyerNumber; }
    public int getMiles() { return miles; }
}
```

Kod klasy `Flight`

Klasa `Flight` reprezentuje dane dotyczące lotu, takie jak numer lotu, miejsce startu, miejsce docelowe i data lotu. To klasa, która będzie używana w kontekście rezerwacji i przypisania pasażerów do określonych lotów.

Listing 3. Klasa Flight

```
package com.example.airlinereservation.models;

import java.time.LocalDate;

public class Flight {
    private String flightNumber;
    private String origin;
    private String destination;
    private LocalDate date;

    public Flight(String flightNumber, String origin, String destination, LocalDate date) {
        this.flightNumber = flightNumber;
        this.origin = origin;
        this.destination = destination;
        this.date = date;
    }

    public String getFlightNumber() { return flightNumber; }
    public String getOrigin() { return origin; }
    public String getDestination() { return destination; }
    public LocalDate getDate() { return date; }
}
```

Kod klasy `InternationalFlight`

Klasa `InternationalFlight` dziedziczy po klasie `Flight` i dodaje specyficzne właściwości, takie jak wymaganie paszportu dla pasażerów.

Listing 4. Klasa InternationalFlight

```
package com.example.airlinereservation.models;

public class InternationalFlight extends Flight {
    private boolean passportRequired;

    public InternationalFlight(String flightNumber, String origin, String destination, LocalDate date, b
        super(flightNumber, origin, destination, date);
        this.passportRequired = passportRequired;
    }

    public boolean isPassportRequired() { return passportRequired; }
}
```

0.7.2. Zastosowanie dziedziczenia w systemie

System rezerwacji lotów w pełni wykorzystuje dziedziczenie, aby uprościć zarządzanie różnymi typami pasażerów oraz lotów. Klasy takie jak `FrequentPassenger` i `VIPPassenger` rozszerzają klasę bazową `Person`, aby dodać szczegółowe informacje dotyczące frequent flyer i statusu VIP, co pozwala na łatwe zarządzanie różnymi rodzajami pasażerów.

Podobnie, klasy takie jak `Flight` i `InternationalFlight` dzielą wspólne cechy, ale umożliwiają dodanie specyficznych właściwości, takich jak wymaganie paszportu, co zwiększa elastyczność systemu i pozwala na łatwiejsze dodawanie nowych typów lotów.

0.7.3. Podsumowanie

Dziedziczenie w systemie rezerwacji lotów pozwala na stworzenie przejrzystej hierarchii klas, co sprawia, że kod jest bardziej modularny, łatwiejszy do rozbudowy i utrzymania. Dzięki dziedziczeniu możliwe jest stworzenie ogólnych klas bazowych, które mogą być rozszerzane o specyficzne właściwości, co upraszcza zarządzanie systemem rezerwacji i umożliwia łatwiejszą modyfikację w przyszłości.

0.8. Podsumowanie

0.8.1. Osiągnięte cele

Projekt został zrealizowany zgodnie z założeniami. System rezerwacji biletów lotniczych, umożliwiający użytkownikom dokonywanie rezerwacji, został zaprojektowany z wykorzystaniem wzorca MVC (Model-View-Controller) oraz bazy danych MySQL. Projekt zawiera kluczowe funkcjonalności, takie jak rejestracja użytkowników, logowanie, przegląd lotów, rezerwacja biletów oraz przegląd rezerwacji użytkowników. Całość działa stabilnie i zapewnia podstawowe funkcje umożliwiające łatwą obsługę rezerwacji.

0.8.2. Problemy napotkane podczas realizacji

W trakcie realizacji projektu napotkano kilka trudności:

- Problemy z konfiguracją bazy danych MySQL, które wymagały dostosowania połączeń oraz mapowania danych.
- Optymalizacja interfejsu użytkownika pod kątem responsywności, co wiązało się z kilkoma iteracjami projektowymi.
- Problemy z integracją różnych elementów systemu, co wymagało rozwiązań iteracyjnych i dodatkowych testów.

0.8.3. Kierunki rozwoju

Projekt może zostać rozwinięty o dodatkowe funkcjonalności:

- Integracja z systemami płatności online.
- Rozwój aplikacji mobilnej umożliwiającej rezerwację biletów.
- Rozbudowa raportów i analiz rezerwacji.

0.8.4. Wnioski

Bibliografia

[1] https://www.w3schools.com/. w3schools documentation. *https://www.w3schools.com/js/default.asp*.

[2] sqlite.org. Sqlite documentation. *https://www.sqlite.org/docs.html*.

[1] [2]

Spis rysunków

Spis listingów

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

……Artur Jurkowski……
Imię (imiona) i nazwisko studenta

Wydział Nauk Ścisłych i Technicznych

……Informatyka……
Nazwa kierunku

……134916……
Numer albumu

1. Oświadczam, że moja praca projektowa pt.: Projekt i implementacja systemu rezerwacji biletów lotniczych z wykorzystaniem języka Java i bazy danych SQLite
   1) została przygotowana przeze mnie samodzielnie*,
   2) nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2021 r., poz. 1062) oraz dóbr osobistych chronionych prawem cywilnym,
   3) nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
   4) nie była podstawą otrzymania oceny z innego przedmiotu na uczelni wyższej ani mnie, ani innej osobie.
2. Jednocześnie wyrażam zgodę/nie wyrażam zgody** na udostępnienie mojej pracy projektowej do celów naukowo–badawczych z poszanowaniem przepisów ustawy o prawie autorskim i prawach pokrewnych.

(miejscowość, data) (czytelny podpis studenta)

* Uwzględniając merytoryczny wkład prowadzącego przedmiot

** – niepotrzebne skreślić