



Uniwersytet Rzeszowski
Kolegium Nauk Przyrodniczych
Instytut Informatyki

Praca projektowa programowanie obiektowe

Elektroniczny system rezerwacji miejsc w samolocie

Prowadzący:

mgr inż. Ewa Żesławska

Autor:

Dominik Żak

nr albumu: 131398

Kierunek: Informatyka, grupa lab 1

Rzeszów 2024

Spis treści

1.	Opis założeń projektu.....	3
2.	Opis struktury projektu	11
3.	Harmonogram realizacji projektu	18
4.	Prezentacja warstwy użytkowej projektu	19
5.	Podsumowanie	27
6.	Literatura	28

1. Opis założeń projektu

Celem projektu jest stworzenie intuicyjnej i funkcjonalnej aplikacji do rezerwacji miejsc w samolocie, przeznaczonej do użytku przez obsługę lotniska. Aplikacja ma na celu usprawnienie procesów rezerwacji i anulowania rezerwacji oraz umożliwić personelowi łatwe i szybkie sprawdzenie dostępnych lotów.

Problem:

Współczesne lotniska obsługują tysiące pasażerów dziennie, co wiąże się z koniecznością efektywnego zarządzania rezerwacjami miejsc w samolocie. Obecne systemy rezerwacji mogą być przestarzałe, niewydajne lub trudne w obsłudze, co prowadzi do problemów takich jak opóźnienia w procesie wchodzenia na pokład samolotu lub trudności w szybkim reagowaniu na zmiany w rezerwacji. Tego typu problemy nie tylko negatywnie wpływają na satysfakcję pasażerów, ale również obciążają personel lotniska, zwiększając ryzyko błędów i obniżając ogólną efektywność operacyjną.

Źródło problemu:

Podstawowym źródłem problemu jest brak nowoczesnego, zintegrowanego systemu do rezerwacji miejsc w samolotach, który byłby intuicyjny i łatwy w obsłudze dla personelu lotniska. Obecne systemy mogą nie oferować możliwości szybkiej aktualizacji danych w czasie rzeczywistym, co prowadzi do sytuacji, gdzie informacje o dostępności miejsc mogą być nieaktualne lub niekompletne. Ponadto, brak automatyzacji w przydzielaniu miejsc sprawia, że obsługa lotniska musi polegać na ręcznych procedurach co jest czasochłonne i podatne na błędy.

Znaczenie problemu:

Efektywne zarządzanie rezerwacjami miejsc w samolotach jest kluczowe dla funkcjonowania każdego lotniska. Problemy związane z błędnym przydzieleniem miejsc oraz opóźnienia w procesie wchodzenia na pokład samolotu mogą prowadzić do poważnych konsekwencji zarówno dla pasażerów, jak i personelu lotniska. Oto, dlaczego ten problem jest ważny:

- **Satysfakcja pasażerów:** Błędy w rezerwacjach i długie opóźnienia mogą znacząco obniżać satysfakcję pasażerów, co może wpłynąć negatywnie na reputację lotniska oraz linii lotniczych.
- **Efektywność operacyjna:** Niewydajne systemy rezerwacyjne zwiększają obciążenie pracowników, co prowadzi do mniejszej produktywności i większej liczby błędów.
- **Bezpieczeństwo:** Problemy z zarządzaniem miejscami mogą prowadzić do chaosu podczas procesu wchodzenia na pokład samolotu, co może wpłynąć na bezpieczeństwo operacji lotniczych.
- **Koszty operacyjne:** Opóźnienia i błędy w rezerwacjach mogą prowadzić do zwiększonych kosztów operacyjnych, związanych m.in. z koniecznością zatrudnienia personelu czy rekompensat dla niezadowolonych pasażerów.

Dowodem istnienia problemu są tutaj opinie ekspertów z branży lotniczej, którzy regularnie zwracają uwagę na potrzebę usprawnienia procesów rezerwacji oraz ankiety i opinie od personelu lotnisk którzy często zgłaszają problemy z obecnymi systemami rezerwacyjnymi.

Niezbędne elementy do rozwiązania problemu:

- **Analiza wymagań i planowanie:**
 - **Co jest potrzebne:** Przeprowadzenie warsztatów z użytkownikami, analiza istniejących systemów, zdefiniowanie wymagań funkcjonalnych i нефункциональных.
 - **Dlaczego:** Zrozumienie potrzeb użytkowników końcowych (personelu lotniska) oraz specyfikacji technicznych jest kluczowe, aby stworzyć aplikację spełniającą oczekiwania i rozwiązującą realne problemy. Dokładna analiza wymagań pozwoli na identyfikację kluczowych funkcji oraz potencjalnych wyzwań.
- **Projektowanie intuicyjnego interfejsu użytkownika (UI):**
 - **Co jest potrzebne:** Prototypowanie, testowanie interfejsu z udziałem użytkowników, poprawki na podstawie feedbacku.
 - **Dlaczego:** Interfejs musi być łatwy w obsłudze, aby personel lotniska mógł szybko i efektywnie zarządzać rezerwacjami bez potrzeby długiego szkolenia. Intuicyjny interfejs zmniejsza ryzyko błędów i przyspiesza proces rezerwacji.
- **Funkcje zarządzania rezerwacjami:**
 - **Co jest potrzebne:** Implementacja mechanizmów aktualizacji danych w czasie rzeczywistym i możliwość dokonywania i odwoływania rezerwacji.
 - **Dlaczego:** Aktualne informacje o dostępności miejsc i możliwość szybkiej modyfikacji są kluczowe dla sprawnego zarządzania i minimalizowania błędów.
- **Testowanie i zapewnienie jakości:**
 - **Co jest potrzebne:** Testy programu, regularne przeglądy kodu, testy funkcjonalne
 - **Dlaczego:** Testowanie jest niezbędne, aby upewnić się, że aplikacja działa poprawnie i jest wolna od błędów i spełnia wymagania użytkowników.
- **Szkolenie i wsparcie dla użytkowników:**
 - **Co jest potrzebne:** Przygotowanie materiałów szkoleniowych, przeprowadzanie szkoleń dla personelu, zapewnienie wsparcia technicznego po wdrożeniu.
 - **Dlaczego:** Nawet najlepsza aplikacja nie będzie skuteczna, jeśli pracownicy nie będą wiedzieli, jak jej używać. Szkolenie i wsparcie pomagają w pełnym wykorzystaniu możliwości aplikacji.
- **Monitorowanie i ciągłe doskonalenie:**
 - **Co jest potrzebne:** Monitorowanie wydajności, zbieranie feedbacku od użytkowników, regularne aktualizacje i poprawki.
 - **Dlaczego:** Środowisko lotniskowe jest dynamiczne, a wymagania mogą się zmieniać. Ciągłe monitorowanie i doskonalenie aplikacji pozwoli na szybkie reagowanie na nowe wyzwania i potrzeby.

Kroki realizacji projektu:

1. Analiza i planowanie:

- **Działania:**
 - Przeprowadzenie warsztatów z użytkownikami końcowymi (personel lotniska) w celu zebrania wymagań.
 - Analiza istniejących systemów i identyfikacja kluczowych funkcji aplikacji.
 - Opracowanie dokumentacji
- **Wynik:** Dokumentacja wymagań projektowych.

2. Projektowanie:

- **Działania:**

- Stworzenie prototypu interfejsu (UI) z uwzględnieniem intuicyjności i łatwości obsługi.
- Testowanie prototypu z użytkownikami i wprowadzanie poprawek.
- Opracowanie architektury systemu.

- **Wynik:** Prototyp UI, dokumentacja techniczna projektu.

3. Implementacja:

- **Działania:**

- Rozwój aplikacji zgodnie z ustalonymi wymaganiami i architekturą.
- Implementacja funkcji zarządzania rezerwacjami.

- **Wynik:** Działająca wersja aplikacji z podstawowymi funkcjonalnościami.

4. Testowanie i zapewnienie jakości:

- **Działania:**

- Przeprowadzenie testów jednostkowych.
- Testy użyteczności z udziałem personelu lotniska.
- Regularne przeglądy kodu i wprowadzanie niezbędnych poprawek.

- **Wynik:** Stabilna i wolna od błędów wersja aplikacji.

5. Szkolenie i wdrożenie:

- **Działania:**

- Przygotowanie materiałów szkoleniowych i podręczników dla użytkownika.
- Przeprowadzenie szkoleń dla personelu lotniska w zakresie obsługi aplikacji.
- Wdrożenie aplikacji na lotnisko i zapewnienie wsparcia technicznego.

- **Wynik:** Przeszkolony personel i wdrożona aplikacja na lotnisku.

6. Monitorowanie i wsparcie po wdrożeniu:

- **Działania:**

- Zbieranie feedbacku od użytkowników końcowych.
- Regularne aktualizacje i doskonalenie aplikacji na podstawie zebranych danych i opinii.

- **Wynik:** Ciągłe doskonalenie aplikacji i bieżące wsparcie dla użytkowników.

Oczekiwany wynik projektu:

Wynikiem prac będzie nowoczesna aplikacja do rezerwacji miejsc w samolocie, przeznaczona do użytku przez obsługę lotniska. Aplikacja będzie posiadać następujące cechy:

- Intuicyjny interfejs użytkownika.
- Wysoka wydajność i stabilność, zapewnione dzięki rygorystycznym testom i zapewnieniu jakości.
- Dokumentacja techniczna oraz materiały szkoleniowe dla użytkowników.

Aplikacja przyczyni się do poprawy efektywności operacyjnej lotniska, zwiększenia satysfakcji pasażerów oraz redukcji błędów w procesie wchodzenia na pokład samolotu.

Wymagania funkcjonalne:

1. Zarządzanie rezerwacją

- **Opis funkcji:** Umożliwia personelowi dodawanie i usuwanie rezerwacji miejsc w samolotach.
- **Jak funkcja powinna działać:**
 - **Dodawanie rezerwacji:** Personel wybiera lot z dostępnymi miejscami, wybiera opcje rezerwacji miejsca a następnie wprowadza dane pasażera.
 - **Anulowanie rezerwacji:** Personel wybiera lot, w którym pasażer posiada rezerwację, wybierając pasażera z listy anuluje jego rezerwację.
- **Przypadki użycia:**
 - Dodawanie nowej rezerwacji dla pasażera.
 - Usuwanie rezerwacji na życzenie pasażera lub w przypadku odwołania lotu.
- **Wynik działania użytkownika:** Zaktualizowana baza danych rezerwacji, zgodnie z wykonanymi operacjami.
- **Wymagania użytkownika:** Umożliwienie zarządzania rezerwacjami pasażerów w sposób łatwy i intuicyjny.
- **Weryfikacja:** Przeprowadzenie testów funkcjonalnych, aby upewnić się, że wszystkie operacje na rezerwacjach są realizowane poprawnie i bez błędów.

2. Aktualizacja danych w czasie rzeczywistym

- **Opis funkcji:** Zapewnia aktualizację informacji o dostępności miejsc.
- **Jak funkcja powinna działać:**
 - **Dane:** Personel powinien dostawać aktualne informacje o dostępnych lotach i dostępności miejsc.
 - **Synchronizacja:** Dane o rezerwacjach powinny być automatycznie dodane do bazy i zaktualizowane.
- **Przypadki użycia:**
 - Wyświetlanie aktualnego stanu dostępności miejsc w samolocie.
- **Wynik działania użytkownika:** Zawsze aktualne informacje o dostępności miejsc.
- **Wymaganie użytkownika:** Umożliwienie dostępu do aktualnych informacji o dostępnych lotach i miejscach w samolocie.

- **Weryfikacja:** Testy integracyjne z bazą danych w celu sprawdzenia poprawności i szybkości aktualizacji danych.

3. Przydzielenie miejsc pasażerom

- **Opis funkcji:** Umożliwia personelowi lotniska automatyczne lub ręczne przydzielenie miejsc pasażerom.
- **Jak funkcja powinna działać:**
 - **Automatyczne przydzielanie miejsc:** Aplikacja automatycznie wybiera wolne miejsce w samolocie.
 - **Ręczne przydzielanie miejsc:** Personel wybiera miejsce z listy dostępnych miejsc i przypisuje je pasażerowi.
- **Wynik działania użytkownika:** Przydzielone miejsce dla pasażera zaktualizowane z bazą danych.
- **Wymaganie użytkownika:** Umożliwienie szybkiego i elastycznego przydzielania miejsc pasażerom, zgodnie z ich preferencjami.
- **Przypadki użycia:**
 - Automatyczne przypisanie miejsca w momencie rezerwacji miejsca w samolocie.
 - Ręczne przypisanie miejsca przez personel na życzenie pasażera.
- **Weryfikacja:** Testy użyteczności z udziałem personelu lotniska, aby upewnić się, że proces przydzielania miejsc jest intuicyjny i niezawodny.

4. Wyszukiwanie dostępnych lotów

- **Opis funkcji:** Umożliwia wyświetlanie aktualnych lotów wraz z ilością dostępnych miejsc.
- **Jak funkcja powinna działać:**
 - **Wyszukiwanie lotów:** Aplikacja wyszukuje wszystkie loty które spełniają warunki zdefiniowane przez użytkownika.
- **Wynik działania użytkownika:** Wyświetlenie listy lotów spełniających warunki.
- **Wymagania użytkownika:** Umożliwienie znalezienia odpowiedniego lotu zgodnie z podanymi warunkami.
- **Przypadki użycia:**
 - Wyszukiwanie lotów według miejsca startu, miejsca końcowego, daty lub kodu.
 - Wyszukiwanie wszystkich lotów.
- **Weryfikacja:** Testy integracyjne z bazą danych, aby zapewnić poprawność wyświetlanych informacji.

5. Interfejs użytkownika (UI)

- **Opis funkcji:** Zapewnia intuicyjny i łatwy w obsłudze interfejs dla personelu lotniska.
- **Jak funkcja powinna działać:**
 - **Przeglądanie:** Personel może przeglądać listę lotów i rezerwacji.
 - **Dostęp:** Łatwy dostęp do najważniejszych funkcji.
 - **Informacje o pasażerach:** Możliwość podejrzania informacji o pasażerach takich jak ich wiek, imię, nazwisko, dane o ubezpieczeniu i ich aktualne rezerwacje.
- **Przypadki użycia:**
 - Przeglądanie listy pasażerów i informacji o dostępnych miejscach.
 - Szybkie wyszukanie lotów według różnych kryteriów.
 - Łatwy dostęp do funkcji dodawania i anulowania rezerwacji.
 - Sprawdzenie informacji o pasażerze.
- **Wynik działania użytkownika:** Szybkie i bezproblemowe zarządzanie rezerwacjami dzięki intuicyjnemu interfejsowi.
- **Wymagania użytkownika:** Umożliwienie efektywnego zarządzania rezerwacjami przez łatwy w obsłudze interfejs.
- **Weryfikacja:** Testy użyteczności z udziałem personelu lotniska oraz zbieranie feedbacku w celu wprowadzenia poprawek UI.

6. Integracja z bazą danych lotniska

- **Opis funkcji:** Umożliwia synchronizację danych o rezerwacjach z bazą danych używaną przez lotnisko.
- **Jak funkcja powinna działać:**
 - **Wprowadzanie danych:** Aplikacja pozwala wprowadzać dane dotyczące rezerwacji do bazy danych.
 - **Odbieranie danych:** Aplikacja otrzymuje i przetwarza dane otrzymane z bazy we właściwy sposób.
- **Przypadki użycia:**
 - Wprowadzenie i pobranie aktualizacji o rezerwacjach.
 - Pobranie aktualizacji statusu lotu w czasie rzeczywistym.
 - Przechowywanie danych o lotach i pasażerach.
- **Wynik działania użytkownika:** Zsynchronizowane i aktualne dane rezerwacyjne w systemie lotniskowym.
- **Wymagania użytkownika:** Umożliwienie przechowywania danych i ich bezproblemowej wymiany.
- **Weryfikacja:** Testy integracyjne z bazą danych w celu zapewnienia poprawności i niezawodności wymiany danych.

7. Wypisywanie ceny za rezerwacje

- **Opis funkcji:** Wyświetla cenę za zarezerwowane miejsce.
- **Jak funkcja powinna działać:**
 - **Wyświetlanie ceny:** Aplikacja powinna wyświetlać cenę w momencie rezerwacji miejsca. Powinna istnieć możliwość zrezygnowania, jeśli cena nie będzie spełniała oczekiwań.
 - **Obliczanie ceny:** Cena powinna być odpowiednio niższa dla osób poniżej 18 roku życia i dla osób od 65 roku życia.
- **Przypadki użycia:**
 - Dodanie rezerwacji pasażera i wyświetlenie jego kwoty do zapłaty
- **Wynik działania użytkownika:** Wyświetlenie ceny za zarezerwowane miejsce.
- **Wymagania użytkownika:** Umożliwienie obliczenia odpowiedniej kwoty do zapłaty przez pasażera.
- **Weryfikacja:** Testy jednostkowe sprawdzające czy podana cena jest prawidłowa i czy uwzględnia wiek pasażera.

Wymaganie niefunkcjonalne:

1. Użyteczność produktu

- **Oczekiwania dotyczące użyteczności:**
 - **Intuicyjny interfejs:** Aplikacja powinna być łatwa w obsłudze dla personelu lotniska, nawet dla tych którzy nie mają zaawansowanego przeszkolenia technicznego.
 - **Prosta nawigacja:** Menu i funkcje aplikacji powinny być logicznie zorganizowane, umożliwiając łatwe znalezienie potrzebnych opcji.
 - **Przyjazny dla użytkowników design:** Kolory, czcionki i układ graficzny powinny być elastyczne i nie męczyć wzroku podczas długotrwałego użytkowania.

2. Dostępność aplikacji

- **Dostępność:** Aplikacja powinna być dostępna i działać bez przerw 24/7/365. W przypadku planowanych prac serwisowych należy wcześniej informować użytkowników.

3. Środowisko aplikacji:

- **Środowisko:** Aplikacja powinna się uruchamiać na sprzęcie komputerowym z systemem Windows nie starszym niż Windows 10 na którym możliwe jest uruchamianie aplikacji języka Java.

4. Wydajność systemu

- **Czas odpowiedzi:**
 - **Przeglądanie listy lotów:** Lista powinna ładować się w ciągu maksymalnie 3 sekund.
 - **Przeglądanie listy pasażerów:** Lista powinna ładować się w ciągu maksymalnie 2 sekund.
 - **Dodawanie/Odwoływanie rezerwacji:** Operacje dodawania i odwoływania rezerwacji powinny być realizowane w ciągu maksymalnie 5 sekund.

5. Kontrola danych

- **Kontrola informacji o pasażerze:** Aplikacja powinna automatycznie sprawdzać czy:
 - Pesel ma 10 cyfr i nie zaczyna się od 0
 - Wiek nie jest mniejszy od 0
 - Pola ubezpieczenia są wypełnione lub pozostawione puste w przypadku braku ubezpieczenia.
 - Data ubezpieczenia jest faktyczną datą.
- **Kontrola dodawanych informacji:** Aplikacja powinna sprawdzać czy:
 - Lot już się odbył i wyświetlać odpowiednie informacji przed rezerwacją.
 - Pasażer jest już w bazie danych, jeśli go nie ma, powinien zostać dodany.
- **Kontrola usuwania rezerwacji:** Aplikacja powinna pozwalać na usuwanie pasażera nawet jeśli lot się odbył, wyświetlając odpowiednie komunikaty.
- **Kontrola wyświetlanych informacji:** Aplikacja powinna ostrzegać użytkownika, gdy próbuje sprawdzić pustą listę pasażerów.

6. Kontrola działań użytkownika

- **Nawigacja:** Aplikacja powinna ostrzegać użytkownika, gdy próbuje zamknąć aplikację z innego miejsca niż ekran główny.

7. Baza danych:

- **Baza danych:** Powinna istnieć skonfigurowana baza danych do której program będzie się łączył. W przypadku braku wykrycia połączenia z bazą danych powinien zostać wyświetlony odpowiedni komunikat.

Rozwinięcie wymagań niefunkcjonalnych:

- Aplikacja powinna uruchamiać się w oknie minimum 1280x800

2. Opis struktury projektu

W tym rozdziale znajdują się informacje dotyczące wykorzystanych narzędzi, struktury systemu oraz podstawowe wymagania sprzętowe.

1. Wykorzystany język programowania i narzędzia

Język programowania: Java

IDE: IntelliJ IDEA

System kontroli wersji: Git

System graficzny: Swing

Baza danych: MySQL

Moduł JDBC: mysql-connector 8.4.0 (Dostępny pod tym [linkiem](#))

2. Minimalne wymagania sprzętowe

- **Procesor:** Intel Core i3 lub lepszy
- **RAM:** 4 GB (zalecane 8 GB lub więcej)
- **Dysk twardy:** 50 MB wolnego miejsca
- **System operacyjny:** Windows 10+
- **Java Development Kit (JDK):** JDK 22 lub nowszy

3. Struktura bazy danych

Nazwa bazy danych: flightsdatabase

Typ bazy danych: Relacyjna baza danych

- Tabela **clients**: Zawiera informacje o klientach
 - ``CleintID` (INT)` – Klucz główny
 - ``IDSocial` (VARCHAR)` – Unikalny numer użytkownika (np. Pesel)
 - ``Name` (VARCHAR)` – Imię
 - ``Surname` (VARCHAR)` – Nazwisko
 - ``Age` (INT)` – Wiek
 - ``InsuranceProvider` (VARCHAR)` – Ubezpieczyciel (może być puste)
 - ``InsuranceExpirationDate` (DATE)` – Data wygaśnięcia ubezpieczenia (może być puste)
 - ``InsuranceCode` (VARCHAR)` – Unikalny kod ubezpieczenia (może być puste)

- Tabela **flights**: Zawiera informacje o lotach
 - `FlightID` (INT) – Klucz główny
 - `Code` (VARCHAR) – Unikalny kod lotu (np. WAW-00011160)
 - `StartCountry` (VARCHAR) – Kraj początkowy lotu
 - `StartLocation` (VARCHAR) – Miasto początkowe lotu
 - `FinishLocation` (VARCHAR) – Miasto końcowe lotu
 - `FinishCountry` (VARCHAR) – Kraj końcowy lotu
 - `Date` (DATE) – Data lotu
 - `Time` (TIME) – Godzina lotu
 - `TotalSeats` (INT) – Liczba miejsc
 - `Price` (DOUBLE) – Cena lotu
- Tabela **reservations**:
 - `ReservationID` (INT) – Klucz główny
 - `FlightID` (INT) – Klucz obcy z tabeli flights
 - `ClientID` (INT) – Klucz obcy z tabeli clients
 - `SeatCol` (INT) – Kolumna wybranego miejsca
 - `SeatRow` (INT) – Rząd wybranego miejsca

Pole `FlightID` z tabeli flights jest w relacji jeden do wielu z polem `FlightID` z tabeli reservations, podobnie `ClientID` z tabeli clients jest w relacji jeden do wielu z polem `ClientID` z tabeli reservations. Pełną strukturę i powiązania bazy danych można zobaczyć na rysunku Rysunek Obraz-Baza-Danych na końcu tej sekcji.

4. Struktura programu

Główne klasy i ich opis:

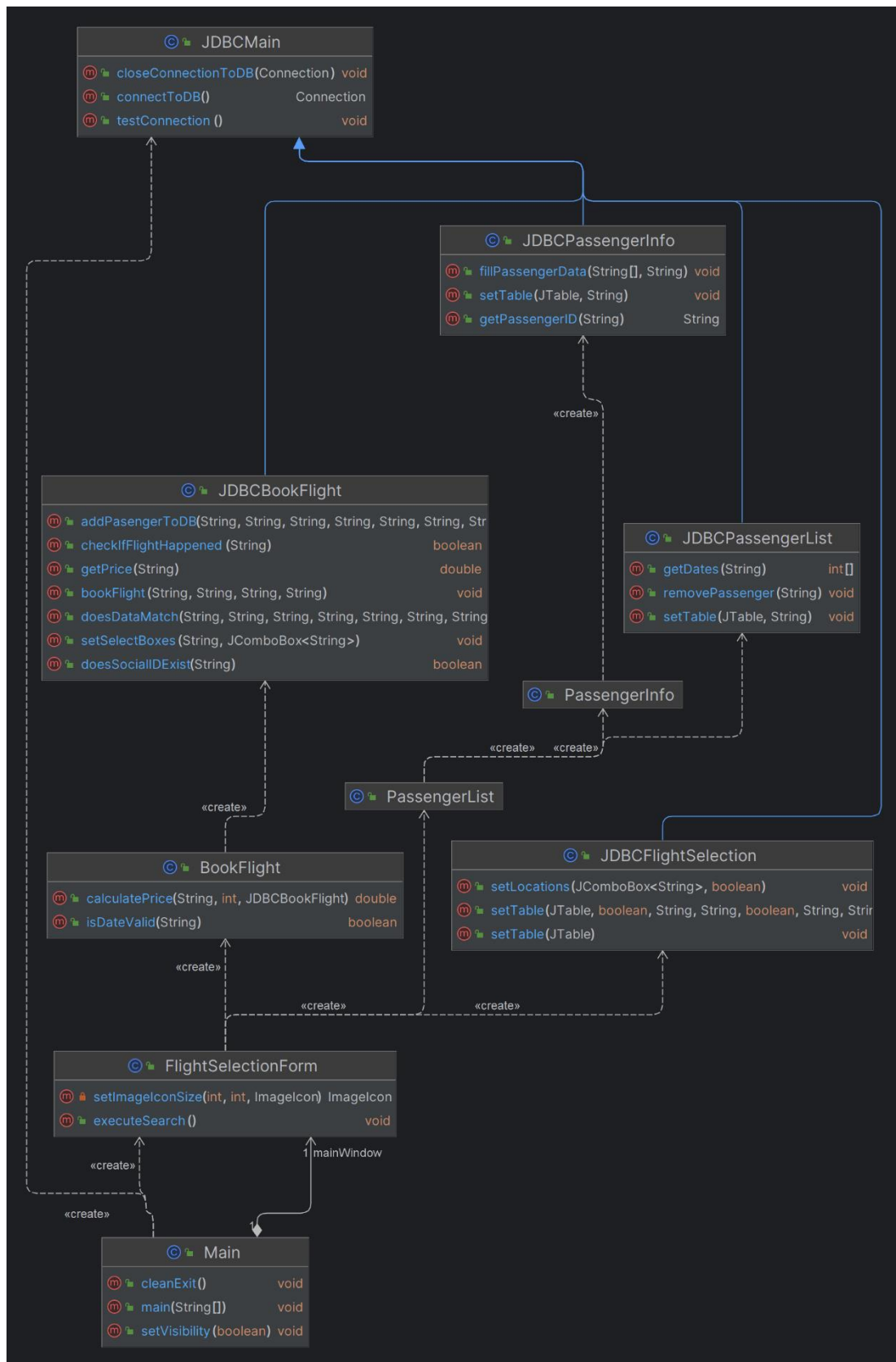
- **`Main`** – Klasa główna, odpowiada za uruchomienie programu i uruchomienie klasy okna głównego. Jest również wykorzystywana przy przełączaniu pomiędzy oknami. Najważniejsze metody:
 - **`setVisibility(boolean)`** – Odpowiada za zmianę widoczności okna głównego, służy przy przejściu z okna głównego do okien pomocniczych i z powrotem, zapewnia, że okno główne nigdy nie zostanie zamknięte przed końcem działania programu. Dodatkowo odświeża listę wyszukanych lotów.
 - **`cleanExit()`** – Odpowiada za znalezienie wszystkich aktywnych okien i ich zamknięcie. Metoda kończy działanie programu.

- **`FlightSelectionForm`** – Klasa rozszerzająca klasę **`JFrame`**, jest odpowiedzialna za wyświetlanie i obsługę okna głównego aplikacji. Większość funkcjonalności klasy jest zawarta w nadpisanych metodach **`actionPerformed(ActionEvent)`** a ich zadaniem jest zmiana informacji wyświetlanych na ekranie. Najważniejsze metody:
 - **`executeSearch()`** – Odpowiada za pobranie informacji i aktualizacje danych w tabeli. Wykorzystuje metody klasy **`JDBCFlightSelection`**
- **`PassengerList`** – Klasa rozszerzająca klasę **`JFrame`**, jest odpowiedzialna za wyświetlanie i obsługę okna listy pasażerów lotu. Większość funkcjonalności klasy jest zawarta w nadpisanych metodach **`actionPerformed(ActionEvent)`** a ich zadaniem jest wywołanie metod innej klasy.
- **`PassengerInfo`** – Klasa rozszerzająca klasę **`JFrame`**, jest odpowiedzialna za wyświetlanie i obsługę okna informacji o pasażerze. Większość funkcjonalności klasy jest zawarta w nadpisanych metodach **`actionPerformed(ActionEvent)`** a ich zadaniem jest wywołanie metod innej klasy.
- **`BookFlight`** – Klasa rozszerzająca klasę **`JFrame`**, jest odpowiedzialna za wyświetlanie i obsługę rezerwacji miejsca. Większość funkcjonalności klasy jest zawarta w nadpisanych metodach **`actionPerformed(ActionEvent)`** a ich zadaniem jest wywołanie metod innej klasy. Najważniejszą z nich jest metoda obsługi przycisku „Zarezerwuj” (w kodzie „SendButton”), przed wywołaniem metody z innej klasy sprawdza czy wszystkie dane są poprawne i weryfikuje je.
- **`JDBCMain`** – Główna klasa łącząca się z bazą danych.
- **`JDBCFlightSelection`** – Klasa rozszerzająca klasę **`JDBCMain`**, odpowiada za wykonanie operacji dla klasy **`FlightSelectionForm`**. Posiada metody łączące się z bazą danych i pobierające z niej dane. Najważniejsze metody:
 - **`setLocations(JComboBox<String>, boolean)`** – Odpowiada za ustawienie modelu dla listy rozwijanej wyboru miast w oknie głównym. Wykonuje odpowiednie zapytanie w zależności od wartości **`boolean`**.
 - **`setTable(JTable, boolean, String, String, boolean, String, String, String)`** – Odpowiada za pobranie informacji o lotach z bazy danych a następnie ustawia te dane w tabeli wyświetlanej w oknie głównym. Dane są pobierane z uwzględnieniem parametrów podanych przez użytkownika w oknie głównym.

- **`JDBCPassengerList`** – Klasa rozszerzająca klasę **`JDBCMain`**, odpowiada za wykonanie operacji dla klasy **`PassengerList`**. Posiada metody łączące się z bazą danych i pobierające lub usuwające z niej dane. Najważniejsze metody:
 - **`setTable(JTable, String)`** – Odpowiada za pobranie informacji o pasażerach lotu z bazy danych a następnie ustawia te dane w tabeli wyświetlanej na ekranie listy pasażerów.
 - **`removePassenger(String)`** – Odpowiada za wysłanie do bazy danych zapytania o usunięcie rekordu. Metoda usuwa rekord zgodnie z przekazanym ID rezerwacji.
 - **`getDates(String)`** – Odpowiada za wysłanie do bazy danych zapytania o datę, aby sprawdzić czy lot już się odbył. Baza danych powinna tutaj zwracać tylko 1 rekord, zgodny ze wskazanym ID rezerwacji.
- **`JDBCPassengerInfo`** – Klasa rozszerzająca klasę **`JDBCMain`**, odpowiada za wykonanie operacji dla klasy **`PassengerInfo`**. Posiada metody łączące się z bazą danych i pobierające z niej dane. Najważniejsze metody:
 - **`getPassengerID(String)`** – Odpowiada za pobranie z bazy danych informacji o ID pasażera wykorzystując podane ID Rezerwacji.
 - **`fillPassengerData(String[], String)`** – Odpowiada za pobranie z bazy danych informacji o pasażerze rozpoznany przez jego ID pasażera, następnie wstawia te dane w przekazaną listę i kończy metodę.
 - **`setTable(JTable, String)`** – Odpowiada za pobranie z bazy danych informacji o lotach pasażera i wstawienie ich do przekazanej tabeli.
- **`JDBCBookFlight`** – Klasa rozszerzająca klasę **`JDBCMain`**, odpowiada za wykonanie operacji dla klasy **`BookFlight`**. Posiada metody łączące się z bazą danych i pobierające z niej dane i wstawiające dane do bazy. Najważniejsze metody:
 - **`setSelectBoxes(String, JComboBox<String>)`** – Odpowiada za pobranie informacji i aktualnie dostępnych miejscach w samolocie a następnie wstawia wolne miejsca do listy rozwijanej, która wyświetla się na ekranie rezerwacji lotu. Metoda jest uruchamiana jeden raz przy uruchomieniu okna rezerwacji, a następnie za każdym razem, kiedy dodana zostanie nowa rezerwacja.

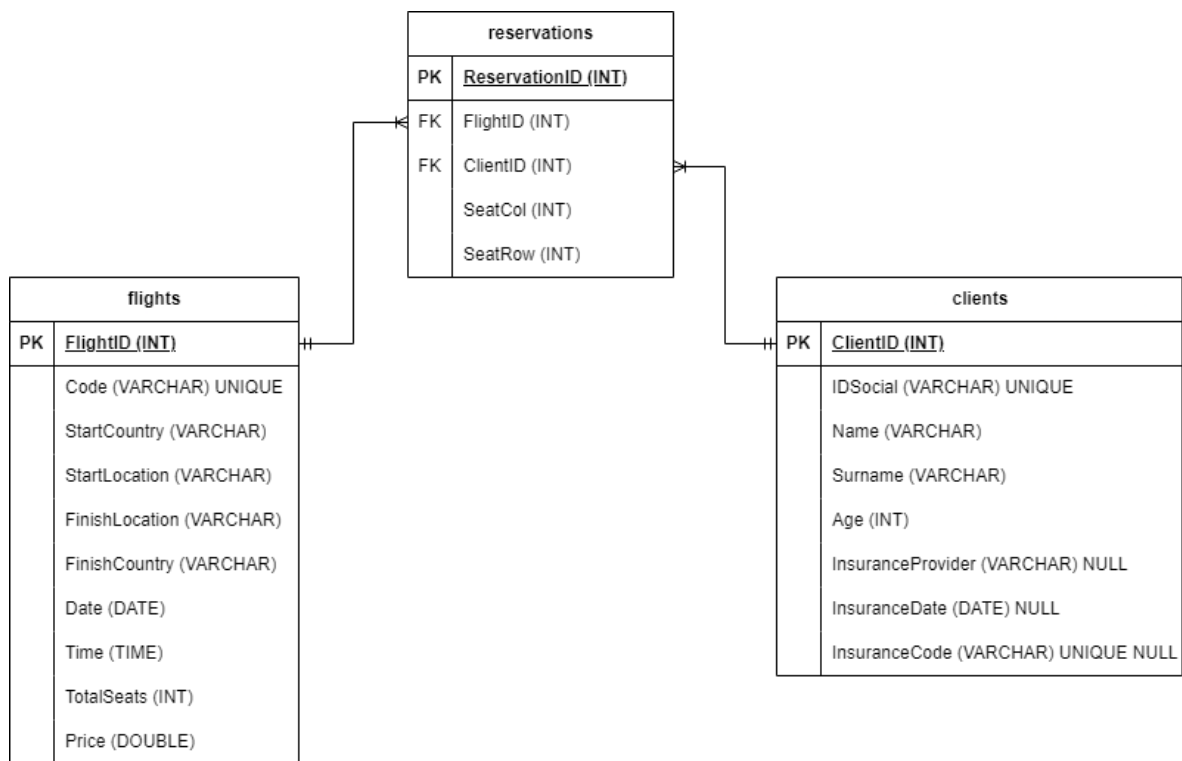
- **`doesSocialIDExist(String)`** – Odpowiada za pobranie informacji o danym ID pasażera (np. Pesel), metoda jedynie sprawdza czy ID istnieje, jeśli tak to zwraca true, w przeciwnym razie false.
- **`doesDataMatch(String, String, String, String, String, String, String)`** – Odpowiada za pobranie informacji o danym pasażerze celem sprawdzenia czy istnieje on już w bazie danych. Jest to później wykorzystane do pominięcia kroku dodawania pasażera do bazy danych przy rezerwacji lotu.
- **`bookFlight(String, String, String, String)`** – Odpowiada za wysłanie do bazy danych zapytania o dodanie pasażera do listy pasażerów lotu.
- **`addPasengerToDB(String, String, String, String, String, String, String)`** – Odpowiada za wysłanie do bazy danych zapytania o dodanie pasażera do listy klientów lotniska.
- **`getPrice(String)`** – Odpowiada za wysłanie do bazy danych zapytania o aktualną cenę lotu. Metoda zwraca wartość typu double.
- **`checkIfFlightHappened(String)`** – Odpowiada za pobranie informacji o dacie, kiedy lot ma się odbyć, jeśli lot podany przez kod już się odbył metoda zwraca true, w przeciwnym razie false.

Na poniższym rysunku Rysunek Diagram-Klas przedstawiony jest diagram klas aplikacji wygenerowany przez program IntelliJ IDEA:



Rysunek Diagram-Klas

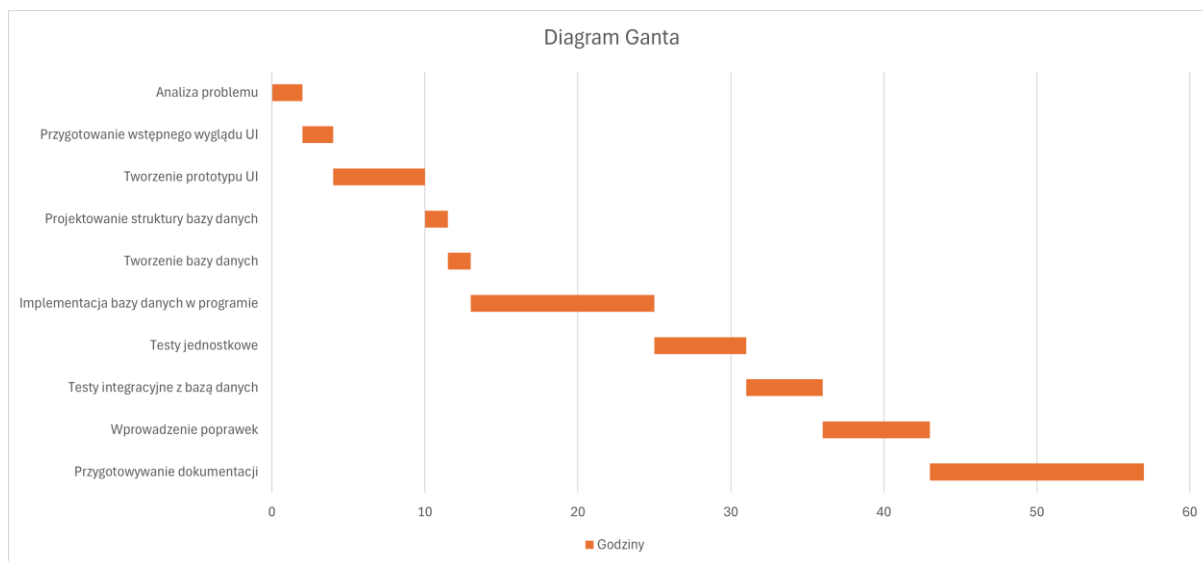
Na poniższym rysunku Rysunek Obraz-Baza-Danych przedstawiona jest struktura bazy danych aplikacji:



Rysunek Obraz-Baza-Danych

3. Harmonogram realizacji projektu

Harmonogram realizacji projektu przedstawiony na diagramie Gantta (Rysunek Diagram-Ganta) przedstawia plan działań z podzieleniem na godziny jakie zostały przeznaczone na poszczególne zadania. Większość czasu poświęcona na projekt to czynności związane z implementacją metod i interfejsu.



Rysunek Diagram-Ganta

Projekt realizowany był z wykorzystaniem systemu kontroli wersji Git, wszystkie pliki źródłowe projektu znajdują się pod adres: [Dominik398/System-Rezerwacji-Miejsc-W-Samolocie \(github.com\)](https://github.com/Dominik398/System-Rezerwacji-Miejsc-W-Samolocie) i będą dostępne do 01.07.2025. Na repozytorium można również zobaczyć historię commitów.

4. Prezentacja warstwy użytkowej projektu

Warstwa użytkowa programu do rezerwacji miejsc w samolocie jest kluczowym elementem aplikacji, ponieważ odpowiada za interakcję pomiędzy użytkownikiem a systemem. Intuicyjny i przyjazny interfejs jest kluczowy w efektywnym zarządzaniu rezerwacjami czy przeglądaniu lotów.

Warstwa użytkowa została zaprojektowana z myślą o prostocie i użyteczności, aby minimalizować czas potrzebny na szkolenie personelu i zapewnić szybki dostęp do najważniejszych funkcji systemu. W tej sekcji omówione zostaną kluczowe komponenty interfejsu użytkownika, a także sposób ich działania.

Ponadto, szczegółowo zostaną opisane interfejsy poszczególnych ekranów aplikacji a także funkcje ułatwiające pracę użytkownikom, takie jak wyszukiwarka lotów. Dzięki temu uzyskamy pełen obraz, tego jak warstwa użytkownika wspiera procesy biznesowe oraz jakie korzyści przynosi użytkownikom końcowym.

Na rysunku Warstwa-Uzytkowa-1 przedstawiono główne okno aplikacji. Użytkownik po uruchomieniu aplikacji otrzymuje ekran startowy na którym może:


- Wyszukać lot według kodu, miejsca startu, miejsca lądowania lub daty wylotu. Użytkownik może też wybrać opcję pokazania lotów które są już pełne oraz pokazania lotów ignorując datę wylotu.
 - Miejsce start i miejsce lądowania są ograniczone do miejsc istniejących w bazie danych (tzn. Jeśli nie ma żadnego lotu do Krakowa, to nie będzie możliwe jego wybranie), pole można pozostawić puste, wtedy wyświetlone zostaną wszystkie loty które spełniają pozostałe wymagania.
 - Data jest ograniczona odpowiednio, dzień do wartości od 1-31 oraz rok od 2020 do aktualnego roku + 20 (aktualnie dla roku 2024, ograniczenie będzie do 2044).
- Z tego ekranu można również zamknąć aplikację.
- Ten ekran zapisuje swój stan przy przejściu w inne okno (np. z ekranu głównego na ekran listy pasażerów i z powrotem).

Kod	Start	Cel	Data wylotu	Wolne Miejsca	Wszystkie Miejsca
WAR-00011150	Warszawa	Berlin	20-06-2024 19:00	2	50
WAR-00012650	Warszawa	Dublin	05-07-2024 19:00	6	50
WAR-00012740	Warszawa	Ateny	06-07-2024 15:30	8	40
WAR-00012830	Warszawa	Budapeszt	07-07-2024 11:15	10	30
WAR-00012950	Warszawa	Wiedeń	08-07-2024 17:00	1	50
WAR-00013040	Warszawa	Mediolan	09-07-2024 13:30	40	40
BER-00013130	Berlin	Warszawa	10-07-2024 09:00	30	30
PAR-00013240	Paryż	Madryt	11-07-2024 11:30	40	40
LON-00013350	Londyn	Berlin	12-07-2024 15:00	50	50
ROM-00013430	Rzym	Wiedeń	13-07-2024 07:45	30	30
MAD-00013540	Madryt	Lizbona	14-07-2024 14:00	39	40
VIE-00013650	Wiedeń	Zurych	15-07-2024 18:45	50	50
AMS-00013730	Amsterdam	Bruksela	16-07-2024 08:00	30	30
BRU-00013840	Bruksela	Dublin	17-07-2024 12:00	40	40
STO-00013950	Sztokholm	Kopenhaga	18-07-2024 10:00	50	50
CPH-00014030	Kopenhaga	Oslo	19-07-2024 19:00	30	30
WAR-00014130	Warszawa	Berlin	15-04-2024 09:00	30	30
WAR-00014240	Warszawa	Paryż	18-04-2024 11:30	40	40
WAR-00014350	Warszawa	Londyn	20-08-2024 15:00	9	50
WAR-00014540	Warszawa	Madryt	25-08-2024 14:00	20	40

Warstwa-Uzytkowa-1

Na rysunku Warstwa-Uzytkowa-2 przedstawiono główne okno aplikacji po wybraniu opcji z tabeli lotów. Wyświetlony zostanie nowy panel boczny na którym pojawią się szczegółowe informacje dotyczące wybranego lotu. Dodatkowo ważniejsze informacje zostaną zaznaczone innym kolorem. Na tym ekranie użytkownik może:

- Zminimalizować i zmaksymalizować panel boczny (odpowiada za to kontrolka, która dotąd była niewidoczna).
- Zarezerwować miejsce w samolocie.
- Obejrzyć listę pasażerów.



Kod: WAR-00012230

Miejsce odlotu:
Warszawa,
Polska

Miejsce docelowe:
Kopenhaga,
Dania

Data odlotu: 01-07-2024

Godzina odlotu: 08:00

Zajęte miejsca: 30

Wolne miejsca: 0

Liczba miejsca: 30

Zarezerwuj

Lista pasażerów

Kod:

Miejsce Startu:

Miejsce Lądowania:

Szukaj

Dzień Miesiąc Rok

Od 30 Czerwiec 2024 Do 31 Czerwiec 2024

☒ Pokaż pełne ☒ Ignoruj Datę

Kod	Start	Cel	Data wylotu	Wolne Miejsca	Wszystkie Miejsca
WAR-00011150	Warszawa	Berlin	20-06-2024 19:00	2	50
WAR-00011240	Warszawa	Parvz	21-06-2024 14:30	0	40
WAR-00011330	Warszawa	Londyn	22-06-2024 10:00	0	30
WAR-00011450	Warszawa	Rzym	23-06-2024 08:15	0	50
WAR-00011540	Warszawa	Madryt	24-06-2024 12:00	0	40
WAR-00011630	Warszawa	Wiedeń	25-06-2024 07:45	0	30
WAR-00011750	Warszawa	Amsterdam	26-06-2024 17:00	0	50
WAR-00011840	Warszawa	Praha	27-06-2024 13:30	0	40
WAR-00011930	Warszawa	Sztokholm	28-06-2024 09:15	0	30
WAR-00012050	Warszawa	Oslo	29-06-2024 15:00	0	50
WAR-00012140	Warszawa	Helsinki	30-06-2024 11:30	0	40
WAR-00012230	Warszawa	Kopenhaga	01-07-2024 08:00	0	30
WAR-00012350	Warszawa	Bruksela	02-07-2024 18:45	0	50
WAR-00012440	Warszawa	Zurich	03-07-2024 14:00	0	40
WAR-00012530	Warszawa	Lizbona	04-07-2024 09:45	0	30
WAR-00012650	Warszawa	Dublin	05-07-2024 19:00	6	50
WAR-00012740	Warszawa	Ateny	06-07-2024 15:30	8	40
WAR-00012830	Warszawa	Budapeszt	07-07-2024 11:15	10	30
WAR-00012950	Warszawa	Wiedeń	08-07-2024 17:00	1	50
WAR-00013040	Warszawa	Mediolan	09-07-2024 13:30	40	40
BER-00013130	Berlin	Warszawa	10-07-2024 09:00	30	30
PAR-00013240	Parvz	Madryt	11-07-2024 11:30	40	40
LON-00013350	Londyn	Berlin	12-07-2024 15:00	50	50
ROM-00013430	Rzym	Wiedeń	13-07-2024 07:45	30	30
MAD-00013540	Madryt	Lizbona	14-07-2024 14:00	39	40
VIE-00013650	Wiedeń	Zurich	15-07-2024 18:45	50	50
AMS-00013730	Amsterdam	Bruksela	16-07-2024 08:00	30	30
BRU-00013840	Bruksela	Dublin	17-07-2024 12:00	40	40
STO-00013950	Sztokholm	Kopenhaga	18-07-2024 10:00	50	50
CPH-00014030	Kopenhaga	Oslo	19-07-2024 19:00	30	30
WAR-00014130	Warszawa	Berlin	15-04-2024 09:00	30	30
WAR-00014240	Warszawa	Parvz	18-04-2024 11:30	40	40
WAR-00014350	Warszawa	Londyn	20-08-2024 15:00	9	50
WAR-00014430	Warszawa	Rzym	23-08-2024 07:45	0	30
WAR-00014540	Warszawa	Madryt	25-08-2024 14:00	20	40


Zamknij

Warstwa-Uzytkowa-2

Okno główne dostarcza również alerty, w zależności od zachowania użytkownika. Przykładowo:

- Gdy użytkownik próbuje zobaczyć listę pasażerów lotu, w którym nie ma żadnych pasażerów, program wyświetli powiadomienie, że lista jest pusta (ale mimo to ją otworzy). Powiadomienie można zobaczyć na rysunku Warstwa-Uzytkowa-3.

System Rezerwacji



Kod: **VIE-00013650**
Miejsce odlotu: **Wiedeń, Austria**
Miejsce docelowe: **Zurych, Szwajcaria**
Data odlotu: **15-07-2024**
Godzina odlotu: **18:45**
Zajęte miejsca: **0**
Wolne miejsca: **50**
Liczba miejsca: **50**

Zarezerwuj
Lista pasażerów

Kod:

Miejsce Startu:

Miejsce Lądowania:

Szukaj

Dzień: Miesiąc: Rok: Dzień: Miesiąc: Rok:
Od: Do:
☐ Pokaż pełne ☐ Ignoruj Datę

Kod	Start	Cel	Data wylotu	Wolne Miejsca	Wszystkie Miejsca
WAR-00011150	Warszawa	Berlin	20-06-2024 19:00	1	50
WAR-00011240	Warszawa	Paryz	21-06-2024 14:30	0	40
WAR-00011330	Warszawa	Londyn	22-06-2024 10:00	0	30
WAR-00011450	Warszawa	Rzym	23-06-2024 08:15	0	50
WAR-00011540	Warszawa	Madryt	24-06-2024 12:00	0	40
WAR-00011630	Warszawa	Wiedeń	25-06-2024 07:45	0	30
WAR-00011750	Warszawa	Amsterdam	26-06-2024 17:00	0	50
WAR-00011840	Warszawa	Praga	27-06-2024 13:30	0	40
WAR-00011930	Warszawa	Sztokholm	28-06-2024 09:15	0	30
WAR-00012050	Warszawa	Oslo	29-06-2024 15:00	0	50
WAR-00012140	Warszawa	Wiedeń	30-06-2024 11:30	0	40
WAR-00012230	Warszawa	Wiedeń	01-07-2024 08:00	0	30
WAR-00012350	Warszawa	Wiedeń	02-07-2024 18:45	0	50
WAR-00012440	Warszawa	Wiedeń	03-07-2024 14:00	0	40
WAR-00012530	Warszawa	Wiedeń	04-07-2024 09:45	0	30
WAR-00012650	Warszawa	Wiedeń	05-07-2024 19:00	6	50
WAR-00012740	Warszawa	Wiedeń	06-07-2024 15:30	8	40
WAR-00012830	Warszawa	Wiedeń	07-07-2024 11:15	10	30
WAR-00012950	Warszawa	Wiedeń	08-07-2024 17:00	1	50
WAR-00013040	Warszawa	Mediolan	09-07-2024 13:30	40	40
BER-00013130	Berlin	Warszawa	10-07-2024 09:00	30	30
PAR-00013240	Paryz	Madryt	11-07-2024 11:30	40	40
LON-00013350	Londyn	Berlin	12-07-2024 15:00	50	50
ROM-00013430	Rzym	Wiedeń	13-07-2024 07:45	30	30
MAD-00013540	Madryt	Lizbona	14-07-2024 14:00	39	40
VIE-00013650	Wiedeń	Zurych	15-07-2024 18:45	50	50
AMS-00013730	Amsterdam	Bruksela	16-07-2024 08:00	30	30
BRU-00013840	Bruksela	Dublin	17-07-2024 12:00	40	40
STO-00013950	Sztokholm	Kopenhaga	18-07-2024 10:00	50	50
CPH-00014030	Kopenhaga	Oslo	19-07-2024 19:00	30	30
WAR-00014130	Warszawa	Berlin	15-04-2024 09:00	30	30
WAR-00014240	Warszawa	Paryz	18-04-2024 11:30	40	40
WAR-00014350	Warszawa	Londyn	20-08-2024 15:00	9	50
WAR-00014430	Warszawa	Rzym	23-08-2024 07:45	0	30
WAR-00014540	Warszawa	Madryt	25-08-2024 14:00	20	40

Zamknij

Warstwa-Uzytkowa-3

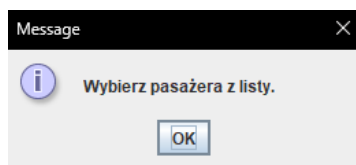
Na rysunku Warstwa-Uzytkowa-4 przedstawiono okno listy pasażerów do którego można wejść wybierając jeden z lotów z ekranu głównego a następnie klikając przycisk Lista pasażerów. Jeśli lista nie jest pusta, to wyświetlona zostanie tabela pasażerów z podstawowymi informacjami. Na tym ekranie użytkownik może:

- Anulować rezerwację (Użytkownik zostanie zapytany o potwierdzenie, rysunek Powiadomienie-4). Dodatkowo w przypadku próby anulowania rezerwacji zostanie wyświetlone powiadomienie, że następuje próba anulowania rezerwacji dla lotu, który już się odbył (Użytkownik zostanie zapytany o potwierdzenie rysunek Powiadomienie-2).
- Wrócić do poprzedniego ekranu przyciskiem wróć.
- Zamknąć aplikację całkowicie (Użytkownik zostanie zapytany o potwierdzenie, rysunek Powiadomienie-3) przyciskiem zamknij.
- W przypadku braku wyboru pasażera z listy program wypisze odpowiednie powiadomienie rysunek Powiadomienie-1.

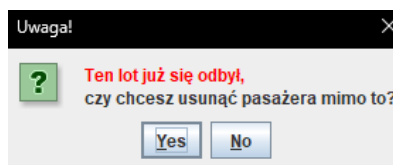
Lista Pasażerów					
ID Rezerwacji	Kod	Nazwisko	Kolumna	Rząd	
4	WAR-00011150	Hoshino	0	3	
5	WAR-00011150	Ishida	0	4	
6	WAR-00011150	Sato	0	5	
7	WAR-00011150	Nakano	0	6	
8	WAR-00011150	Suzuki	0	7	
9	WAR-00011150	Kobayashi	0	8	
10	WAR-00011150	Watanabe	0	9	
11	WAR-00011150	Yamamoto	1	0	
12	WAR-00011150	Nakamura	1	1	
13	WAR-00011150	Kato	1	2	
14	WAR-00011150	Yoshida	1	3	
15	WAR-00011150	Inoue	1	4	
16	WAR-00011150	Kinoshita	1	5	
17	WAR-00011150	Murakami	1	6	
18	WAR-00011150	Abe	1	7	
19	WAR-00011150	Matsumoto	1	8	
20	WAR-00011150	Okada	1	9	
21	WAR-00011150	Kaneko	2	0	
22	WAR-00011150	Takahashi	2	1	
23	WAR-00011150	Ono	2	2	
24	WAR-00011150	Hayashi	2	3	
25	WAR-00011150	Fujii	2	4	
26	WAR-00011150	Takagi	2	5	
27	WAR-00011150	Ueda	2	6	
28	WAR-00011150	Endo	2	7	
29	WAR-00011150	Nagai	2	8	
30	WAR-00011150	Yoshikawa	2	9	
31	WAR-00011150	Fukuda	3	0	
32	WAR-00011150	Sakamoto	3	1	
33	WAR-00011150	Saito	3	2	
34	WAR-00011150	Kojima	3	3	
35	WAR-00011150	Yamashita	3	4	
36	WAR-00011150	Sakamoto	3	5	
37	WAR-00011150	Matsuo	3	6	
38	WAR-00011150	Kuwahara	3	7	
39	WAR-00011150	Kondo	3	8	
40	WAR-00011150	Yamashiro	3	9	
41	WAR-00011150	Oda	4	0	
42	WAR-00011150	Saeki	4	1	
43	WAR-00011150	Mizuno	4	2	
44	WAR-00011150	Miyamoto	4	3	
45	WAR-00011150	Nakata	4	4	
46	WAR-00011150	Teraoka	4	5	

Anuluj Rezerwacje
Pokaż Informacje o pasażerze
Wróć
Zamknij

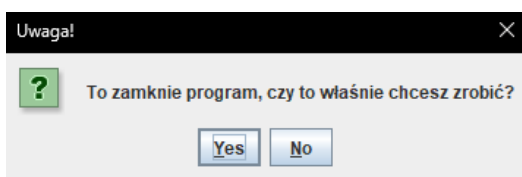
Warstwa-Uzytkowa-4



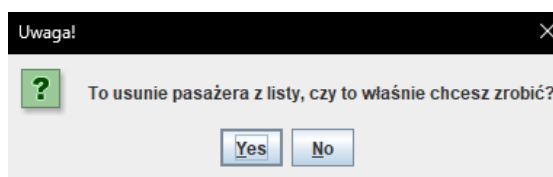
Powiadomienie-1



Powiadomienie-2



Powiadomienie-3



Powiadomienie-4

Na rysunku Warstwa-Uzytkowa-5 przedstawiono okno informacji o pasażerze, do którego można wejść wybierając przycisk pokaż informacje o pasażerze. Okno listy pasażerów pozostanie otwarte, a użytkownik może otworzyć wiele okien z informacją o pasażerze. Należy jednak uważać, ponieważ zamknięcie okna listy pasażerów lub powrót do okna głównego spowoduje zamknięcie wszystkich okien z informacjami o pasażerze. W tym oknie nie ma specjalnych funkcji, poza możliwością zamknięcia okna. Użytkownik może tutaj zobaczyć szczegółowe informacje o pasażerze m.in. jego imię i nazwisko, pesel, wiek, informacje o ubezpieczeniu oraz jego loty.

Informacje

Dane:

Brak ubezpieczenia!

Imie i nazwisko: Olivia Hoshino

Pesel: 4962831704

Wiek: 22

Loty pasażera:

Kod	Start	Cel	Data
WAR-00011150	Warszawa	Berlin	2024-06-20

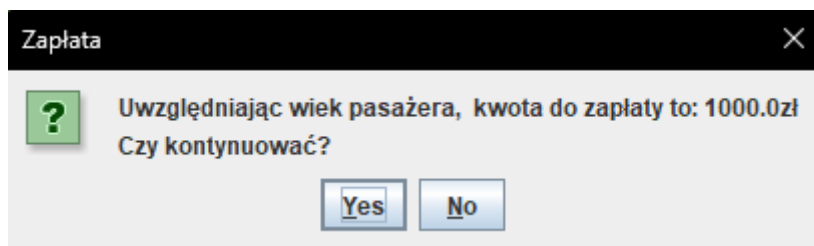
Zamknij Okno

Warstwa-Uzytkowa-5

Na rysunku Warstwa-Uzytkowa-6 przedstawiono okno rezerwacji. Można do niego przejść z okna głównego, wybierając lot z wolnymi miejscami a następnie klikając w przycisk zarezerwuj w panelu bocznym. To okno pozwala użytkownikowi na dodanie nowego pasażera do wybranego lotu. W przypadku gdy dane nie będą prawidłowe, na dolnym panelu wyświetlona zostanie odpowiednia informacja. Użytkownik powinien tutaj wpisać dane pasażera i gdy będzie gotowy kliknąć przycisk dodaj. Wspierane są takie funkcje jak:

- Sprawdzenie czy numer pesel jest 10 cyfrowy i nie zaczyna się od 0.
- Sprawdzenie czy imię i nazwisko zostało wypełnione
- Sprawdzenie czy wiek jest liczbą i czy jest większy od 0
- Sprawdzenie czy ubezpieczyciel, data ubezpieczenia i numer ubezpieczenia są wypełnione, lub pozostawione puste.
 - Sprawdzana jest również data, w przypadku błędu, wypisywany jest błąd i podpowiedź, jaki powinien zostać wprowadzony format daty.
- Automatyczne lub ręczne wybranie miejsca, kolumna i rząd.
- Możliwość wyczyszczenia formularza.
- Możliwość powrotu do poprzedniego okna.
- Możliwość zamknięcia programu (Podobnie jak w oknie listy pasażerów, wypisana zostanie prośba o potwierdzenie).
- Wyświetlenie powiadomienia o próbie rezerwacji lotu po dacie odlotu (prośba użytkownika o potwierdzenie operacji).
- Możliwość automatycznego wykrycia czy pasażer jest w bazie i dodanie go w razie takiej potrzeby.

Dodatkowo w momencie, gdy wszystkie dane będą spełniały oczekiwania wyświetlona zostanie cena za przelot z uwzględnieniem zniżki dla osób poniżej 18 roku życia lub dla osób od 65 roku życia. Użytkownik zostanie wtedy poproszony o potwierdzenie operacji, w przypadku rezygnacji, dodanie pasażera zostanie anulowane. Zaprezentowano na rysunku Powiadomienie-5.



Powiadomienie-5

The screenshot shows a web application window titled "Rezerwacja". The form has a green background and contains the following fields and controls:

- Pesel:
- Imie:
- Nazwisko:
- Wiek:
- Ubezpieczyciel:
- Data wygaśnięcia:
- Numer ubezpieczenia:
- Kolumna i Rząd: (dropdown arrow)

Below the fields are two buttons: "Wyczyść" and "Dodaj".

A large light gray area below the buttons contains the text "Czekam na dane..." (Waiting for data...).

At the bottom right of the window are two buttons: "Wróć" and "Zamknij".

Warstwa-Uzytkowa-6

Ikona wykorzystana na ekranie głównym i jako ikona programu pochodzi z tej [strony](#), i jest autorstwa [BomSymbols](#).

5. Podsumowanie

Aplikacja do rezerwacji miejsc w samolocie, przeznaczona do użytku przez obsługę lotniska, została zaprojektowana z myślą o usprawnieniu procesu rezerwacji i odwoływania rezerwacji.

Wprowadzenie do opisu technicznego szczegółowo przedstawiło strukturę aplikacji, wykorzystane narzędzia oraz minimalne wymagania sprzętowe. Opisano również strukturę bazy danych, zaprojektowaną hierarchię klas oraz najważniejsze metody.

W opisie warstwy użytkowej programu uwzględniono kluczowe komponenty interfejsu użytkownika oraz sposób ich działania. Szczegółowo opisano interfejsy poszczególnych ekranów aplikacji, takie jak ekran główny, ekran listy pasażerów, ekran informacji o pasażerze oraz ekran rezerwacji.

Ostatecznie, aplikacja ta ma na celu zapewnienie intuicyjnego, szybkiego i wydajnego narzędzia do obsługi rezerwacji miejsc w samolocie, wspierając procesy biznesowe i zwiększając efektywność operacyjną na lotniskach.

Plany na przyszłość

W dalszych planach rozwoju aplikacji przewidziane są następujące działania:

1. Rozszerzenie funkcjonalności aplikacji:

- Wprowadzenie opcji edycji już istniejących rezerwacji.
- Dodanie możliwości rezerwacji grupowej.
- Dodanie obsługi bagażu pasażera.

2. Udoskonalenie interfejsu użytkownika:

- Dodanie możliwości zaznaczania kilku rekordów tabeli jednocześnie

3. Poprawa bezpieczeństwa:

- Wprowadzenie mechanizmu logowania
- Szyfrowanie danych pasażerów

4. Analiza i raportowanie:

- Dodanie automatycznych systemów raportowania

Realizacja tych planów na przyszłość zapewni dalszy rozwój i udoskonalenie aplikacji, dostosowując ją do zmieniających się potrzeb użytkowników i dynamicznego rynku lotniczego.

6. Literatura

1. Bruce Eckel, Thinking in Java, wyd. IV, Helion, 2006.
2. Cay S. Horstmann, Core Java: Fundamentals Volume 1 wyd. Pearson, Edycja 11, 2020.