# D O K U M E N T A C J A

## Cel projektu i jego idee (opis)

**Strona internetowa linii lotniczych**

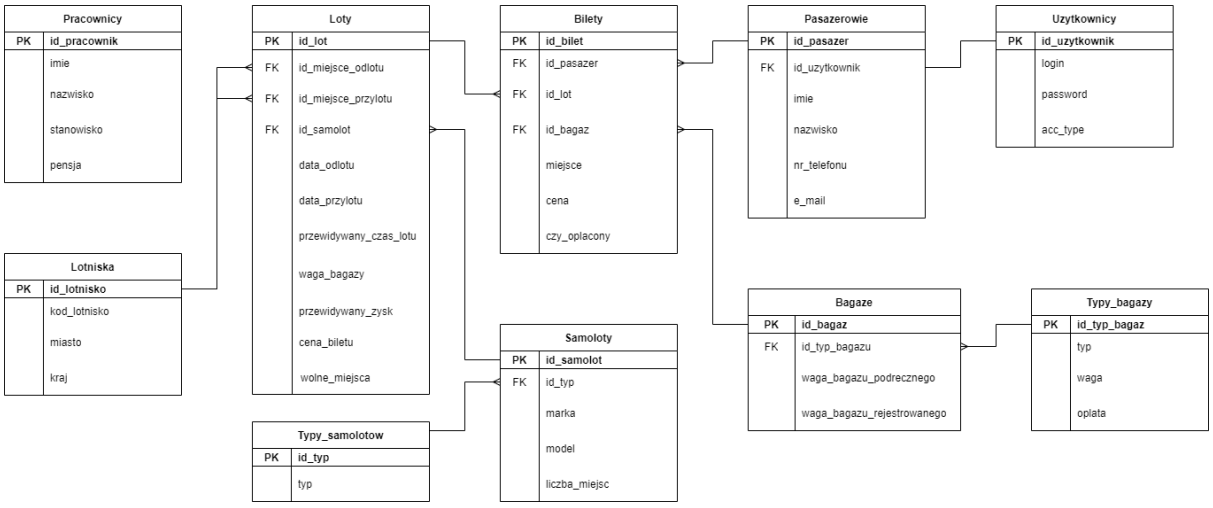
**Cele szczegółowe:**

- Stworzenie w pełni funkcjonalnej aplikacji oferującej użytkownikom łatwy dostęp do zakupienia biletów online.
- Stworzenie panelu administracyjnego umożliwiającego łatwe wprowadzanie bądź edytowanie nowych lotów.
- Eleganckie okienko logowania używające technologii AJAX oraz PHP do komunikacji z serwerem oraz walidowania danych bez przekierowania na oddzielną stronę.
- Responsywny panel administracyjny korzystający w całości w technologii AJAX.
- Zabezpieczenie wrażliwych danych (hashowanie haseł).

**Funkcjonalności:**

- Strona będzie zawierać formularz logowania (ten sam dla użytkownika i administratora) oraz formularz rejestracji.
- Użytkownik będzie mógł składać rezerwacje na dostępne loty w wybranym przez siebie terminie.
- Użytkownikowi będą się wyświetlać powiadomienia o nadchodzących lotach.
- Wyszukiwanie lotów.
- Administratorzy będą mogli dodawać, edytować, usuwać nowe loty, samoloty, lotniska.
- Drukowanie eleganckiego biletu.

## Projekt bazy danych



**Opis projektu bazy danych**

- Każdy pasażer może mieć tylko jedno konto użytkownika
- Pasażerowie mogą kupić wiele biletów
- Jeden typ bagażu może należeć do wielu bagaży
- Do każdego lotu może być przypisane wiele biletów
- Na każdym lotnisku może odbywać się wiele lotów
- Każdy typ samolotu może być przypisany do wielu samolotów
- Każdy samolot może odbywać wiele lotów

## Opis CRUD

Przykładowy package CRUD (istnieją do większości tabel, posiadają zaimek _DIU. Tu BILETY_DIU)

**Deklaracja procedur:**

- Usuwania biletu
- Dodawania biletu
- Aktualizacji biletu
- Pobierania ilości rekordów
- Wyświetlania określonej ilości biletów
- Opłacania biletu

```
create or replace NONEDITIONABLE PACKAGE BILETY_DIU AS
    PROCEDURE BILETY_DELETE(
        in_id_bilet IN bilety.id_bilet%TYPE);
    PROCEDURE BILETY_INSERT(
        in_pasazer IN bilety.id_pasazer%TYPE,
        in_lot IN bilety.id_lot%TYPE,
        in_bagaz IN bilety.id_bagaz%TYPE,
        in_miejsce IN bilety.cena%TYPE,
        in_cena IN bilety.cena%TYPE,
        in_oplacony IN bilety.czy_oplacony%TYPE);
    PROCEDURE BILETY_UPDATE(
        in_id_bilet IN bilety.id_bilet%TYPE,
        in_pasazer IN bilety.id_pasazer%TYPE,
        in_lot IN bilety.id_lot%TYPE,
        in_bagaz IN bilety.id_bagaz%TYPE,
        in_miejsce IN bilety.cena%TYPE,
        in_cena IN bilety.cena%TYPE,
        in_oplacony IN bilety.czy_oplacony%TYPE);
    PROCEDURE GET_NUMBER_OF_ROWS(wyjscie OUT NUMBER);
    PROCEDURE GET_FROM_TABLE(
        c1 OUT SYS_REFCURSOR,
        in_start IN NUMBER,
        in_stop IN NUMBER,
        in_search IN NUMBER);

    PROCEDURE OPLAC_BILET(
    in_id_bilet IN bilety.id_bilet%TYPE);
END BILETY_DIU;
```

PROCEDURE BILETY_DELETE – odpowiada za usuwanie biletu dla wskazanego id.

```
    PROCEDURE BILETY_DELETE(
        in_id_bilet IN bilety.id_bilet%TYPE)
        IS
        BEGIN
            DELETE FROM bilety WHERE id_bilet = in_id_bilet;
    END BILETY_DELETE;
```

PROCEDURE BILETY_INSERT – odpowiada za wprowadzanie nowego rekordu. Wartość ID pobiera z odpowiadającej tabeli sekwencji.

```
    PROCEDURE BILETY_INSERT(
        in_pasazer IN bilety.id_pasazer%TYPE,
        in_lot IN bilety.id_lot%TYPE,
        in_bagaz IN bilety.id_bagaz%TYPE,
        in_miejsce IN bilety.cena%TYPE,
        in_cena IN bilety.cena%TYPE,
        in_oplacony IN bilety.czy_oplacony%TYPE)
        IS
        BEGIN
            INSERT INTO bilety
            VALUES(BILETY_SEQ.nextval, in_pasazer, in_lot, in_bagaz, in_miejsce, in_cena, in_oplacony)
;
    END BILETY_INSERT;
```

PROCEDURE BILETY_UPDATE – odpowiada za aktualizacje rekordu. Przyjmuje wszystkie dane, sprawdza w IF-ach które zostały zmienione, a które pozostały puste (nie zmienione) i w razie potrzeby je aktualizuje.

```
    PROCEDURE BILETY_UPDATE(
        in_id_bilet IN bilety.id_bilet%TYPE,
        in_pasazer IN bilety.id_pasazer%TYPE,
        in_lot IN bilety.id_lot%TYPE,
        in_bagaz IN bilety.id_bagaz%TYPE,
        in_miejsce IN bilety.cena%TYPE,
        in_cena IN bilety.cena%TYPE,
        in_oplacony IN bilety.czy_oplacony%TYPE)
        IS
        BEGIN
            IF in_pasazer IS NOT NULL THEN
                UPDATE bilety SET id_pasazer = in_pasazer WHERE id_bilet = in_id_bilet;
            END IF;
            IF in_lot IS NOT NULL THEN
                UPDATE bilety SET id_lot = in_lot WHERE id_bilet = in_id_bilet;
            END IF;
            IF in_bagaz IS NOT NULL THEN
                UPDATE bilety SET id_bagaz = in_bagaz WHERE id_bilet = in_id_bilet;
```

```
            END IF;
            IF in_miejsce IS NOT NULL THEN
                UPDATE bilety SET miejsce = in_miejsce WHERE id_bilet = in_id_bilet;
            END IF;
            IF in_cena IS NOT NULL THEN
                UPDATE bilety SET cena = in_cena WHERE id_bilet = in_id_bilet;
            END IF;
            IF in_oplacony IS NOT NULL THEN
                UPDATE bilety SET czy_oplacony = in_oplacony WHERE id_bilet = in_id_bilet;
            END IF;
    END BILETY_UPDATE;
```

PROCEDURE GET_NUMBER_OF_ROWS – pobiera liczbę rekordów w tabeli. Potrzebne w celu wykonania paginacji w panelu administracyjnym

```
    PROCEDURE GET_NUMBER_OF_ROWS(wyjscie OUT NUMBER)
        AS
        BEGIN
        SELECT
        COUNT(*) INTO wyjscie
        FROM bilety;
    END GET_NUMBER_OF_ROWS;
```

PROCEDURE GET_FROM_TABLE – pobiera z tabeli wszystkie rekordy z określonego przedziału (in_start, in_stop), zawiera opcję wyszukiwania określonych rekordów po przez wprowadzenie zmiennej in_search

```
    PROCEDURE GET_FROM_TABLE(
        c1 OUT SYS_REFCURSOR,
        in_start IN NUMBER,
        in_stop IN NUMBER,
        in_search IN NUMBER)
        AS
        BEGIN
        OPEN c1 FOR
            SELECT
                rn, id_bilet, id_pasazer,(IMIE || ' ' || NAZWISKO) pasazer, id_lot, (miejsce_odlotu ||
 ' ' || miejsce_przylotu || ' ' || do_eleganckiej_daty(data_odlotu)) lot, id_bagaz, (WAGA_BAGAZU_REJES
TROWANEGO || ' ' || TYP) bagaz, miejsce, cena, czy_oplacony
            FROM
                (SELECT rownum  rn, id_bilet, bilety.id_pasazer, imie, nazwisko, bilety.id_lot, l_odl.
kod_lotnisko miejsce_odlotu, l_p.kod_lotnisko miejsce_przylotu, data_odlotu, bagaze.id_bagaz, bagaze.w
aga_bagazu_rejestrowanego, typy_bagazy.typ , miejsce, cena, czy_oplacony FROM
                ((((((bilety INNER JOIN PASAZEROWIE ON bilety.id_pasazer=pasazerowie.id_pasazer)
                INNER JOIN loty ON bilety.id_lot=loty.id_lot)
                INNER JOIN lotniska l_odl ON loty.id_miejsca_odlotu=l_odl.id_lotnisko)
                INNER JOIN lotniska l_p ON loty.id_miejsca_przylotu=l_p.id_lotnisko)
                INNER JOIN bagaze ON bilety.id_bagaz=bagaze.id_bagaz)
                INNER JOIN typy_bagazy ON bagaze.id_typ_bagazu=typy_bagazy.id_typ_bagaz)
                WHERE id_bilet LIKE '%' || in_search || '%'
```

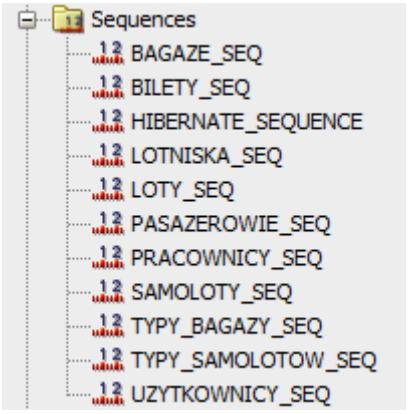
```
            ORDER BY id_bilet)
        WHERE
            rn BETWEEN in_start AND in_stop;
    END GET_FROM_TABLE;
```

`PROCEDURE OPLAC_BILET` – niekiedy pakiety CRUD-owe posiadają procedury ekskluzywne dla określonej tabeli. To jest jedna z nich. Odpowiada za opłacenie biletu dla określonego id.

```
    PROCEDURE OPLAC_BILET(
        in_id_bilet IN bilety.id_bilet%TYPE)
    IS
    BEGIN
    UPDATE bilety SET czy_oplacony = 1 WHERE id_bilet=in_id_bilet;
    COMMIT;
END;
```

## Sekwencje

Dla każdej tabeli są ustawione sekwencje pod nazwą [NAZWA_TABELI]_SEQ (dla kolumny id)



## Funkcje PL/SQL

FUNCTION xd - sprawdza które pola są uzupełnione, oraz zwraca numer odpowiadający (o użyciu jej później).

```
create or replace NONEDITIONABLE FUNCTION xd(
in_odlot_kraj IN lotniska.kraj%TYPE,
in_odlot_miasto IN lotniska.miasto%TYPE,
in_przylot_kraj IN lotniska.kraj%TYPE,
in_przylot_miasto IN lotniska.miasto%TYPE)
RETURN number
IS
choice number;
BEGIN
IF ((in_odlot_kraj is not null or in_odlot_miasto is not null) and (in_przylot
_kraj is not null or in_przylot_miasto is not null)) THEN
choice := 1;
ELSIF ((in_odlot_kraj is not null or in_odlot_miasto is not null) and (in_przy
lot_kraj is null or in_przylot_miasto is null)) THEN
choice := 2;
ELSIF ((in_odlot_kraj is null or in_odlot_miasto is null) and (in_przylot_kraj
 is not null or in_przylot_miasto is not null)) THEN
choice := 3;
ELSE
choice := 4;
END IF;
RETURN choice;
END;
```

FUNCTION do_eleganckiej_daty – przyjmuję datę oraz zwraca ją w formacie 'DD-MM-YY HH24:MI'

```
create or replace NONEDITIONABLE FUNCTION do_eleganckiej_daty(
    in_date IN DATE)
RETURN STRING
IS
   out_date STRING(20);
BEGIN
    out_date := to_char(in_date, 'DD-MM-YY HH24:MI');
RETURN out_date;
END;
```

# Procedury PL/SQL

**Pakiet `USERUTILS`** – służy głównie do obsługi rejestracji oraz logowania.

- `ADD_USER` – dodaje nowego użytkownika

- `CHECK_IF_EXISTS` – sprawdza czy podany użytkownik już istnieje

- `FIND_USER` – pobiera użytkownika o danym username oraz zwraca jego dane do weryfikacji

**Pakiet `USER_PROFILE`** - służy do obsługi profilu użytkownika

- `USER_FIND` – pobranie i wyświetlenie danych pasażera z tabeli pasażerowie

- `USER_EDIT` – edycja danych użytkownika obsługująca niewprowadzone dane oraz błędy związane z brakiem danych użytkownika w pasażerach

- `USER_DEL` – usuwanie użytkownika

- `USER_BOOKING` – procedura wyświetlająca nieopłacone loty użytkownika

- `USER_FLIGHTS` – procedura wyświetlająca opłacone loty użytkownika do dwóch tygodni do przodu

**Pakiet `SEARCH_FLIGHT`**

- `Countries` – pobiera kraje z tabeli lotniska

- `Cities` – pobiera miasta z tabeli lotniska

**Pakiet STATYSTYKA_OPISOWA**

- `najczesciej_wybierane_bagaze` – pobiera liczbę wszystkich bagaży, procent oraz typ bagażu

- `najczesciej_wybierane_miejsca` – pobiera 5 najczęściej wybieranych przez klientów miejsc docelowych

- `top_5_klientow` – pobiera top 5 klientów którzy wydali najwięcej, wraz z kwota którą wydali

-`pracownicy_zarobki` – kategoryzuje pracowników względem ich zarobków

- `ostatnie_zarobki` – pobiera dane przychodu firmy z ostatniego tygodnia, miesiąca oraz kwartalu

## Pozostałe procedury (bez pakietowe)

Procedura `Magiczny_insert_bagazu` - służy do wprowadzenia bagażu do listy bagaży

Procedura `Elegancki_insert_biletu` – wprowadza bilet do tabeli bilety i jednocześnie korzystając z procedury `Magiczny_insert_bagazu` dodaje bagaż

Procedura `FIND_FLIGHT` – odwołuje się do funkcji `xd` i na podstawie zwróconej przez nią wartości wyszukuje lot.

Procedura `Get_data_for_PDF` - pobieranie danych pasażera, miejsca w samolocie, daty odlotu, miejsca odlotu i przylotu oraz typu bagażu w celu utworzenia biletu w PDF

Procedura `Oplac_bilet` – opłaca bilet

## Triggery PL/SQL

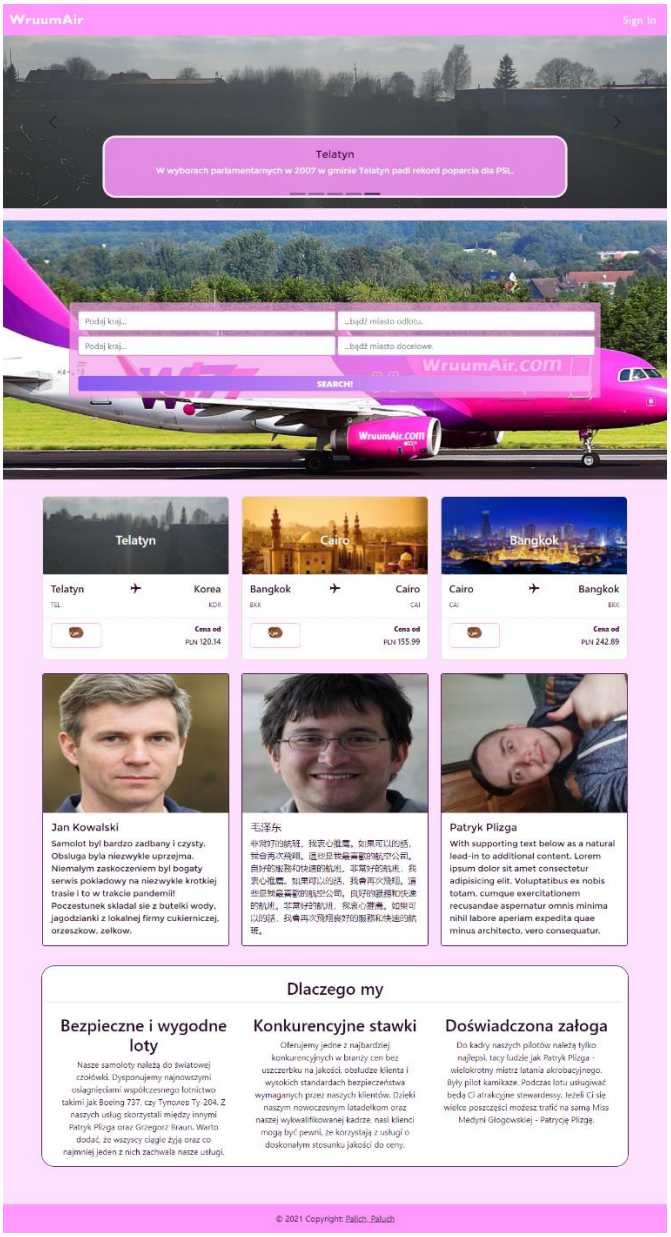
1. `CREATE_NEW_USER_PROFILE_TRG` – compoud trigger tworzący użytkownika korzystającego z formularza rejestracji
2. `CZAS_CENA_LOTU_TRG` – oblicza czas lotu na podstawie data_przylotu oraz data_odlotu. Oblicza podstawowa cene biletu ze wzoru czas_lotu * 3.27 * 0.75, gdzie 3.27 jest wymyślona przeze mnie liczba, a 0.75 to zysk po odjęciu wszystkich opłat(podatki, opłaty itp.)
3. `LOTNISKA_UPPERCASE` trigger zamieniający nowe dane w tabeli lotniska na uppercase
4. `LOTY_BAGAZE_ZYSK` – w przypadku insertu do biletów zmniejsza ilość dostępnych miejsc w samolocie. W przypadku usuwania rekordu z biletów dodaje dostępne miejsca. W przypadku opłacenia rezerwacji (`updating('Czy_oplacony')`) pobiera stara łączna wagę bagaży, oraz łączna cenę wszystkich biletów oraz dodaje do nich wagę, oraz cenę z nowego biletu.
5. `SET_BAGAZ_TYP_TRG` – w przypadku dodania lub aktualizacji bagażu sprawdza oraz zwraca typ bagażu
6. `WOLNE_MIEJSCA_TRG` – po dodaniu nowego lotu wprowadza liczbę dostępnych miejsc na podstawie wybranego samolotu

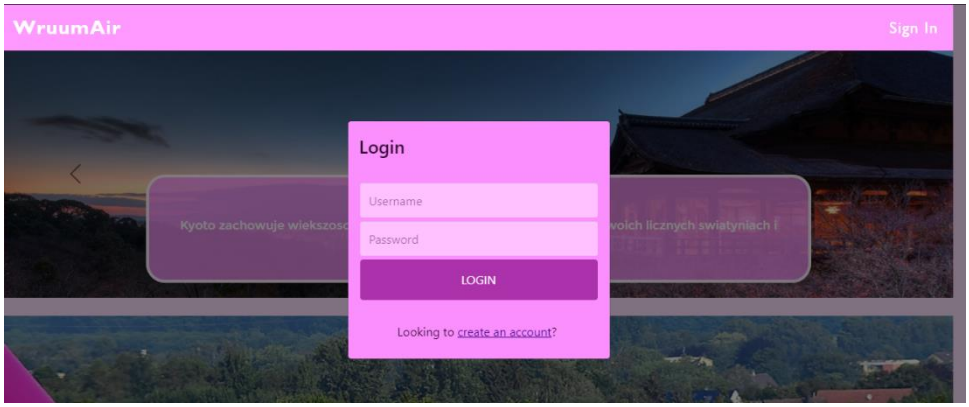
## Opis techniczny interface (+ Zrzuty ekranu)

Wykorzystywane technologie:
HTML, JS, CSS, BOOTSTRAP, AJAX, JQUERY, PHP, ORACLE SQL, PL/SQL

`config.php` oraz `config2.php` - pliki konfiguracyjne, podaje się w nim dane logowania do bazy danych `config2.php` od `config.php` różni się tylko tym ze nie ustawia sesji w PHP.

1. `Index.php` - używa styli `new.css, styles.css, passtrength.css`. Używa zewnętrznych skryptów `login-register.js` oraz `jquery.passtrength.js`
- `Jquery.passtrength.js` - sprawdza siłę hasła podczas rejestracji
- `Login-register.js` - obsługuje modal rejestracji oraz logowania. Korzysta ze skryptu `checkUser.php` oraz `checkRegister.php`
- `CheckRegister.php` oraz `checkUser.php` - łączą się z bazą danych oraz obsługują procedury rejestracji oraz logowania






2. `Dashboard.php` - panel administracyjny

   Wykorzystane style:

   - `dashboard.css`

   Wykorzystane skrypty:

   - `dashboard.js`

   - `utils.php` - zbiór wszystkich funkcji php potrzebnych do wyciągania danych z bazy

   Główne zakładki korzystające z plików o tych samych nazwach z rozszerzeniem .php w podfolderze prefab, w każdej z zakładek jest możliwość dodania, edycji i usunięcia odpowiedniego rekordu:

   - `lotniska` – wszystkie dostępne lotniska

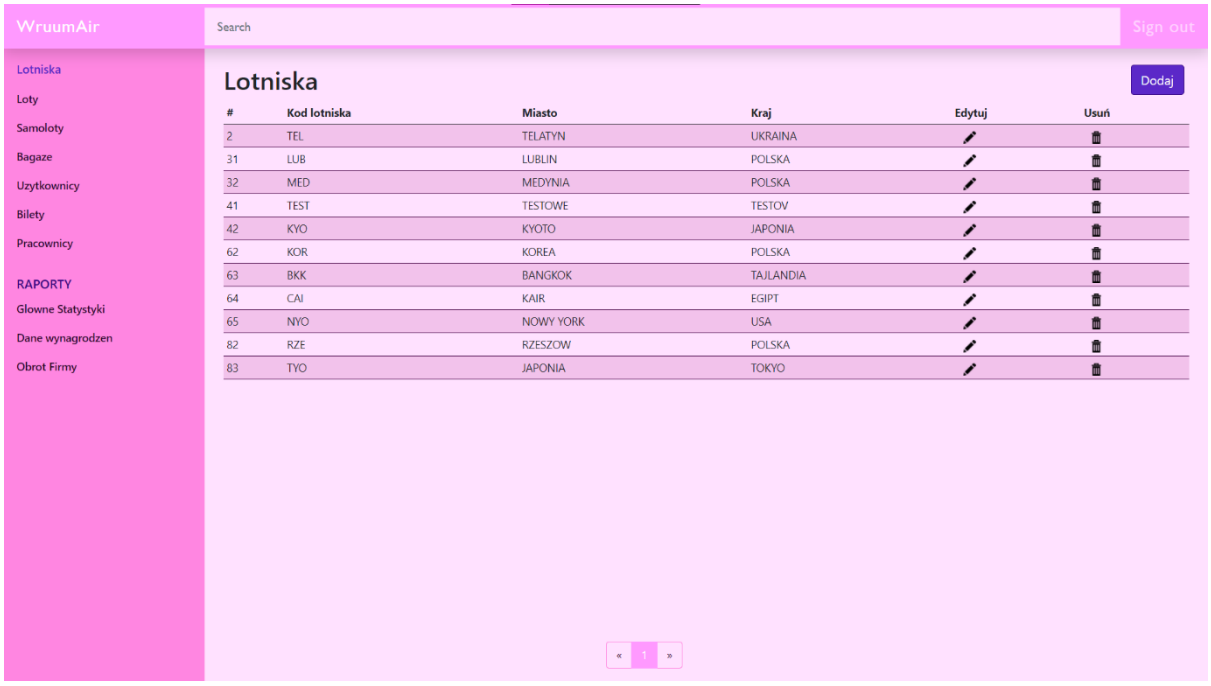
   -`loty` – informacje o aktualnych lotach

   -`samoloty` – dane o samolotach

   -`użytkownicy` – zarejestrowani użytkownicy i typ konta

   -`bilety` – informacje o rezerwacjach

   -`pracownicy` – dane, stanowisko i zarobki pracowników





3. `glowne_statystkie.php, pensje_pracownikow.php , obrot.php` - `odpowiadaja` kolejno zakladkom w RAPORTY.
4. `user.php` - panel użytkownika

Wykorzystane style:

`-login-register.css`

`-styles.css`

`-user.css`

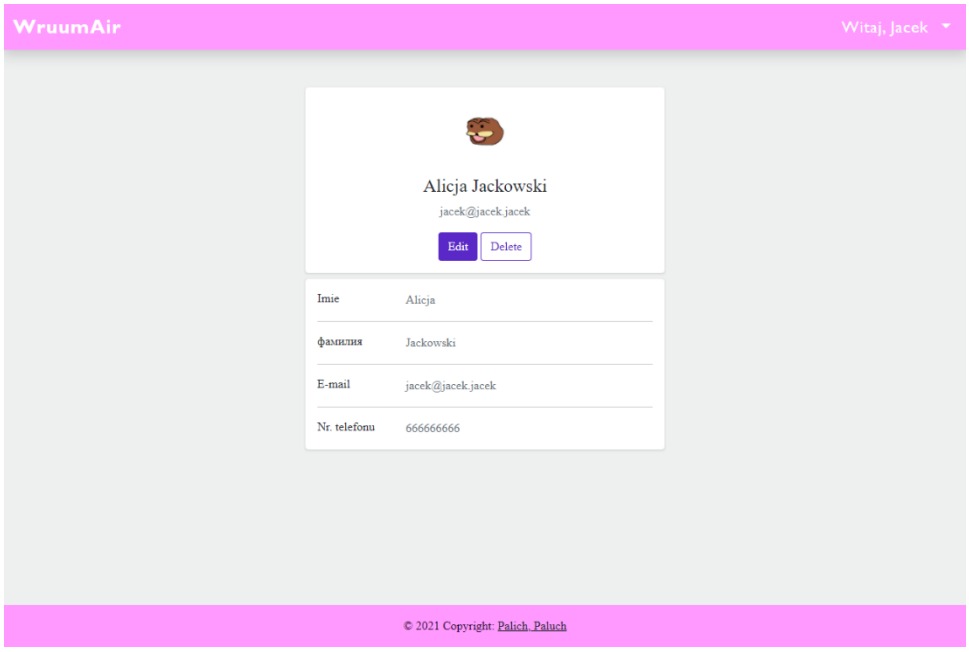
Wykorzystane skrypty:

`-user.js`

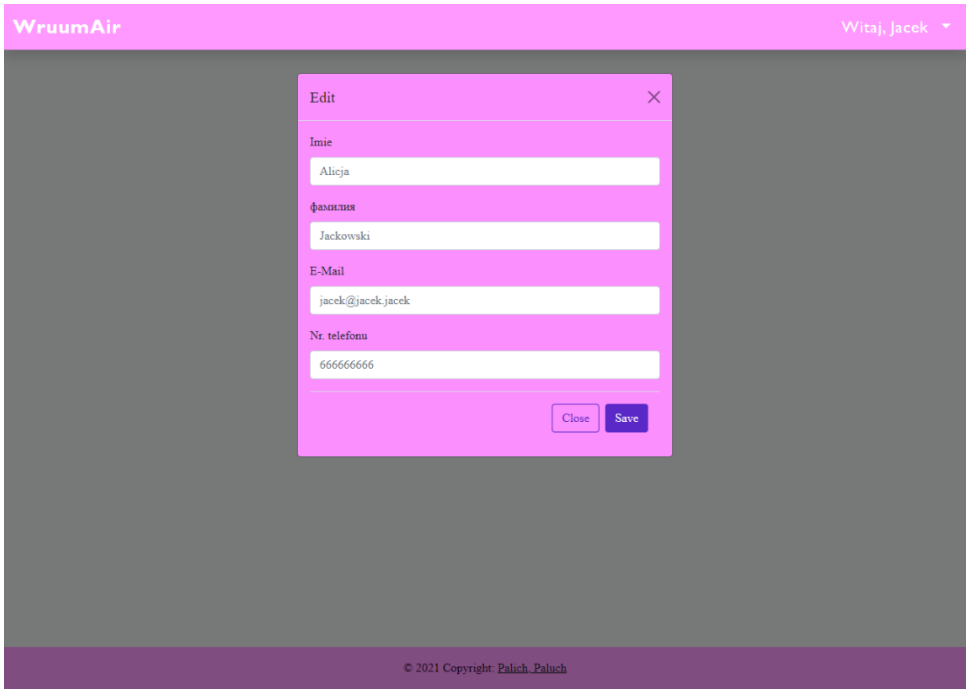
Podstrony użytkownika:

Strona główna użytkownika z powiadomieniami:

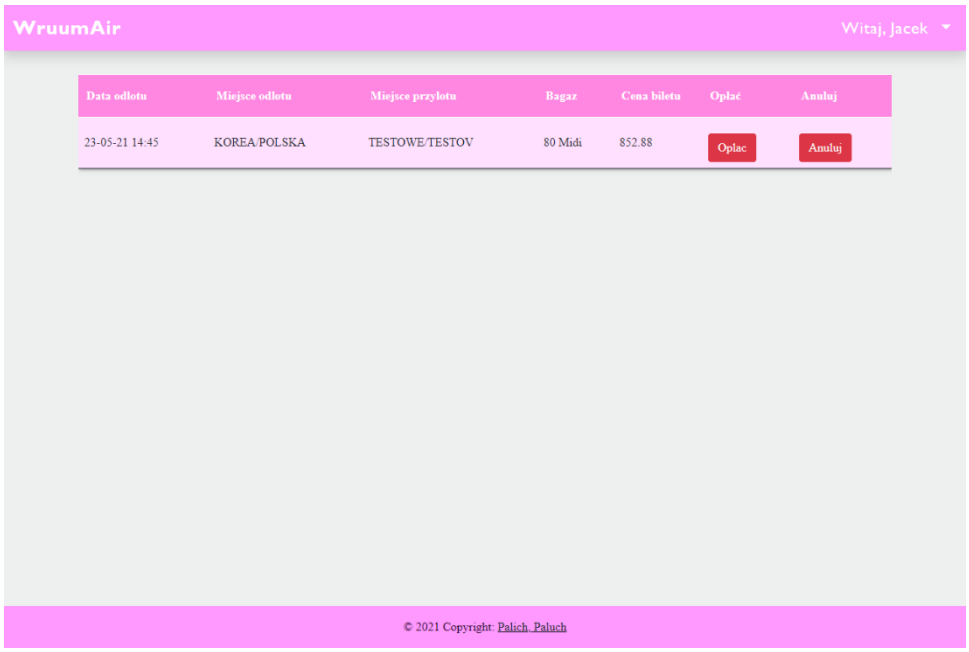

Profil:



Edycja danych:

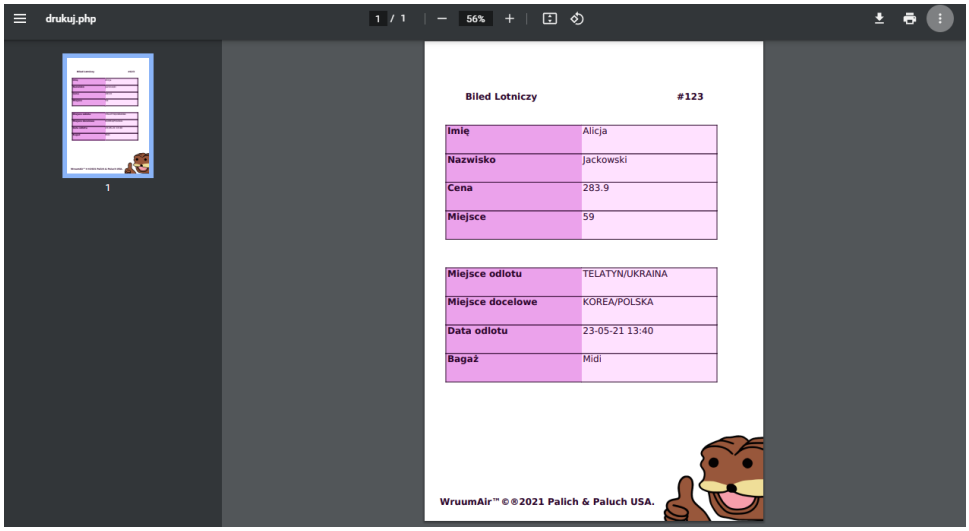



Zarezerwowane loty z możliwością opłacenia lub anulowania:

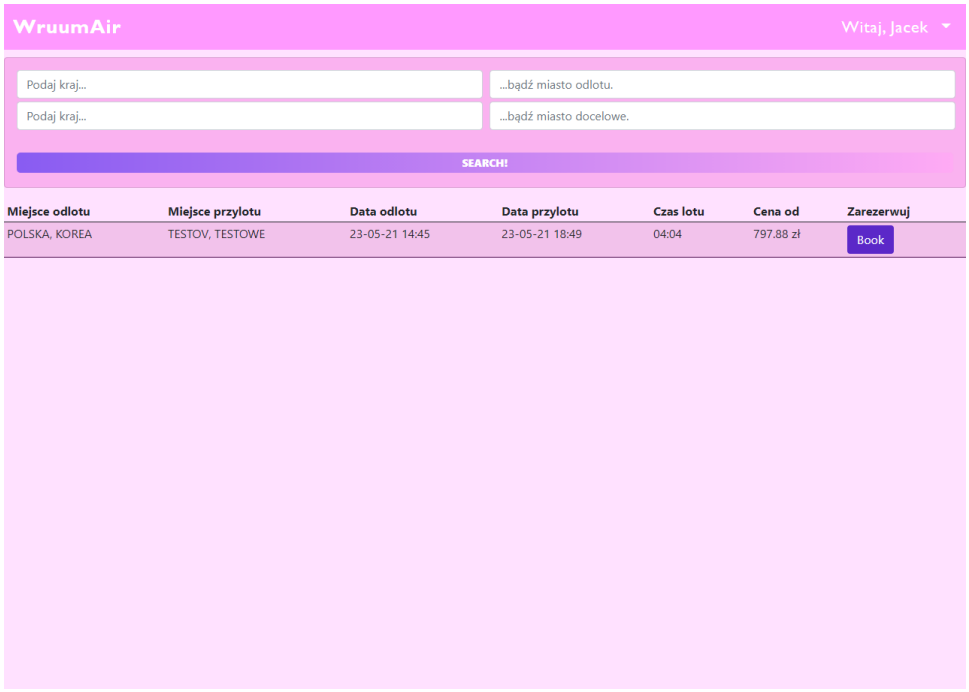


Nadchodzące loty z możliwością wydrukowania biletu:

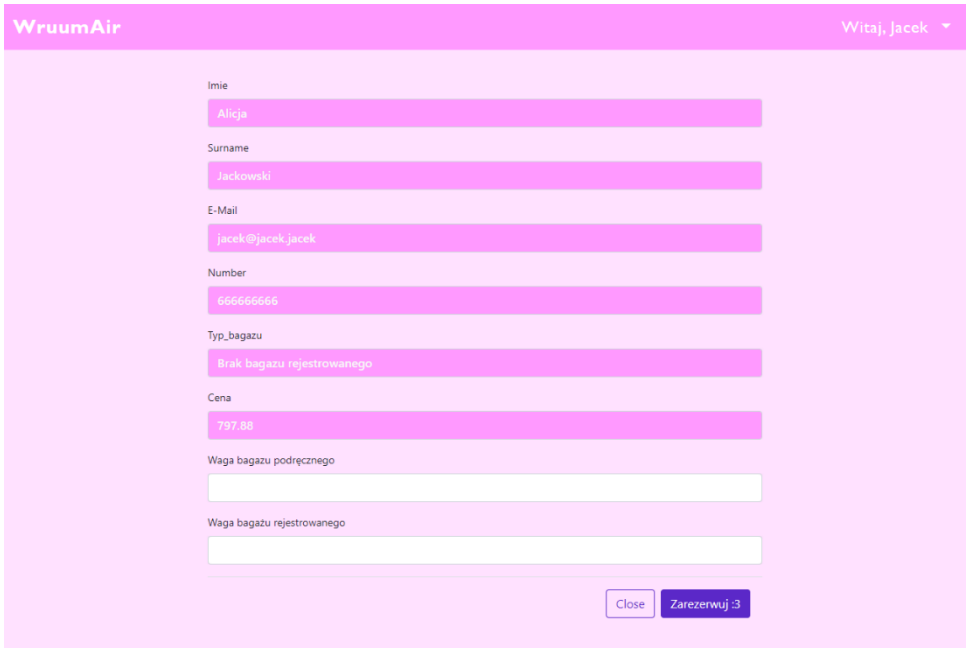

Przykładowy bilet:



5. `search.php` - służy do wyszukiwania dostępnych lotów.




6. `test10.php` - służy do rezerwacji wybranego lotu. Po rezerwacji lot pojawia się w zakładce użytkownika w '`Bookings`'. Gdzie może zostać opłacony



## Najciekawsze procedury

1. statystyka_opisowa.pracownicy_zarobki

```
PROCEDURE pracownicy_zarobki(c1 OUT SYS_REFCURSOR)

        IS
        BEGIN
            OPEN c1 FOR
        SELECT CASE
         WHEN pensja <= 1000 THEN '1-1000'
         WHEN pensja <= 3000 THEN '1001-3000'
         WHEN pensja <= 6000 THEN '3001-6000'
         ELSE '6001+'
       END AS pensja,
       COUNT(*) AS n
        FROM pracownicy
GROUP BY CASE
         WHEN pensja <= 1000 THEN '1-1000'
         WHEN pensja <= 3000 THEN '1001-3000'
         WHEN pensja <= 6000 THEN '3001-6000'
         ELSE '6001+'
         END;
END pracownicy_zarobki;
```

Procedura ta operuje na CASE-ach. Pobiera ona pensje pracowników, a następnie sprawdza w jakich przedziałach się one znajdują oraz zlicza ilość wystąpień danego przedziału. Pobrane dane umieszcza w kursorze c1.

2. statystyka_opisowa.ostatnie_zarobki

```
PROCEDURE ostatnie_zarobki(ostatni_tydzien OUT NUMBER,
      ostatni_miesiac OUT NUMBER, ostatni_kwartal OUT NUMBER)
    IS
    zysk NUMBER(10, 2);
    data_o DATE;

    teraz DATE;

    cursor zarobki is
        SELECT przewidywany_zysk, data_odlotu from loty;
    BEGIN
        ostatni_tydzien := 0;
        ostatni_miesiac := 0;
        ostatni_kwartal := 0;

        select CURRENT_DATE into teraz from dual;
        OPEN zarobki;
        LOOP
            FETCH zarobki into zysk, data_o;
            IF teraz-data_o <= 7 THEN
                ostatni_tydzien := ostatni_tydzien + zysk;
            END IF;
            IF teraz-data_o <= 30 THEN
                ostatni_miesiac := ostatni_miesiac + zysk;
            END IF;
            IF teraz-data_o <= 90 THEN
                ostatni_kwartal := ostatni_kwartal + zysk;
            END IF;
            EXIT WHEN zarobki%notfound;
        END LOOP;
        CLOSE zarobki;

END ostatnie_zarobki;
```

Procedura ta pobiera do kursora zarobki zapytanie zwracające przewidywany zysk oraz datę odlotu z tabeli loty. Następnie deklaruje zmienne odpowiadające poszczególnym przedziałom czasowym. Później następuje iteracja kursora zarobki. W każdej iteracji pobiera zysk oraz datę odlotu. Na podstawie ich w następujących IF-ach przypisuje zysk do odpowiadającej mu zmiennej. Po zakończeniu pętli zwraca przychód z ostatniego tygodnia, miesiąca i kwartału.

3. CREATE_NEW_USER_PROFILE_TRG

```
create or replace NONEDITIONABLE TRIGGER CREATE_NEW_USER_PROFILE_TRG
  FOR INSERT
  ON uzytkownicy
    COMPOUND TRIGGER
    username uzytkownicy.username%TYPE;
    acc_type uzytkownicy.typ_konta%TYPE;
    AFTER EACH ROW IS
        BEGIN
            username := :new.username;
            acc_type := :new.typ_konta;
        END AFTER EACH ROW;
    AFTER STATEMENT IS
        BEGIN
            IF acc_type = 0 THEN
                user_profile.user_edit(username, username, '', '', '');
            END IF;
    END AFTER STATEMENT;
END CREATE_NEW_USER_PROFILE_TRG;
```

Ten trigger kompozytowy w momencie wprowadzenia danych pobiera nowy username oraz przypisany mu typ konta. Po pobraniu tych danych sprawdza czy typ konta jest równy zero (Czy jest użytkownikiem). Jeżeli jest równy 0, wywołuje się procedura `user_edit`. Która wstawia podstawowe wartości do tabeli pasażerowie.