Dokumentacja projektu z przedmiotu Bazy Danych I

Temat: Organizacja turnieju sportowego

Autor: Krystian Molenda

1 Projekt koncepcji oraz założenia

1.1 O Projekcie

> Projekt ma na celu stworzenie aplikacji bazodanowej pomocnej w organizacji zawodów łuczniczych rangi ogólnopolskiej, zarządzanych przez Związek Łuczniczy. Aplikacja ułatwia proces zapisów na zawody, a podczas zawodów pozwala na bieżąco gromadzić dane i udostępniać wyniki rywalizacji, a po zawodach wygenerować protokoły z ich przebiegu. W aplikacji skupiam się na organizacji strzelań kwalifikacyjnych.

O przebiegu zawodów

> Zawodnicy startujący w zawodach występują jako reprezentanci swoich klubów lub jako tzw. niezrzeszeni (nie są członkami żadnego klubu). Zawodnik, oraz sędzia muszą posiadać ważną licencję, która uprawnia do udziału w zawodach. Rozróżniany jest podział według płci -- kobiet i mężczyzn.

Przed zawodami

> Zawodnicy po rejestracji, przejściu pozytywnie kontroli ważności licencji są zapisywani na listę startową dotyczącą danej odległości. Lista startowa zawiera informację o zawodniku, stanowisku do którego jest przypisany, odległości na której startuje oraz jego wyniku na danej odległości. Na jednym stanowisku znajduje się maksymalnie dwóch zawodników z jednej kategorii. Ponadto do każdych czterech stanowisk przypisany jest jeden sędzia posiadający licencję. Odpowiada on za rozwiązywanie sytuacji spornych oraz korekty po wprowadzeniu wyniku.

Przebieg strzelań kwalifikacyjnych (informacje o rzeczywistych zawodach)

> Strzelanie odbywa się do tarcz, których wielkość zależy od odległości. Punkty na tarczy numerowane są od 1 do 10 (X jest liczony jako 10pkt.). Nie trafienie w tarczę jest równoważne z otrzymaniem 0 punktów za dany strzał. Każdy zawodnik na etapie kwalifikacji oddaje 12 serii, po 6 strzał na określonych odległościach. Odległości nie muszą być różne. Mogą się różnić zależnie od kategorii. Wyniki kolejnych strzał w kolejnych seriach zapisuje sędzia tarczowy w specjalnym dokumencie nazywanym metryczką. Maksymalną liczbą punktów możliwych do uzyskania na jednej odległości jest 720. W celu uproszczenia danych notowane są tylko wyniki końcowe wraz z liczbą trafionych najwyżej punktowanych obszarów.

1.2 Analiza wymagań uzytkownika

1. Wprowadzanie usuwanie oraz edycja danych organizacyjnych dotyczących:

    - Klubów
    - Zawodników
    - Sędziów
    - Trenerów
    - Relacji Trenerów z Klubami
    - Relacji Zawodników z Klubami

2. Rejestrowanie usuwanie oraz edycja zawodników na liście startowej

    - Przypisanie zawodnika do listy startowej na daną odległość
        - Zawodnik może występować w liście startowej wielokrotnie oraz na różnych stanowiskach pod warunkiem, że startuje na różnych odległościach
    - Walidacja wprowadzanych danych
        - Zawodnik nie może być dwa razy przypisany do tej samej odległości
        - Zawodnik nie może być na stanowisku z innym zawodnikiem z tego samego klubu
        - Zawodnik nie może być na stanowisku z innym zawodnikiem z innej kategorii
        - Zawodnik musi posiadać ważną licencję

3. Wprowadzanie oraz edycja wyników

    - Możliwość wprowadzania wyników zawodników na danych odległościach
    - Możliwość wprowadzania ewentualnych korekt w przypadku błędnego wprowadzenia wyniku

4. Wyświetlanie statystyk oraz rankingów

    - Wyswietlanie rankingów zawodników w danych kategoriach na danych odległościach
    - Wyświetlanie rankingów zawodników w danych kategoriach ogólnych
    - Wyświetlanie rankingów klubowych
    - Wyświetlanie statystyk danego zawodnika

---

2. Projekt konceptualny

2.1 Założenia dotyczące budowy bazy danych - zdefiniowanie encji

Dane organizacyjne

1. Klub
    - numer licencji klubu
    - Nazwa
    - Miasto

- Kluby zrzeszają zawodników i trenerów. Nadzór nad klubami, zawodnikami, trenerami, sędziami prowadzi Związek Łuczniczy -- udzielając licencji.
- Numer licencji klubu jest wartością unikalną nadawaną automatycznie z chwilą zarejestrowania klubu w Związku. W toku eksploatacji bazy danych nie jest możliwa jego zmiana.

---

1. Zawodnik

    - numer licencji zawodnika
    - Imię
    - Nazwisko
    - Płeć
    - Licencja ważna

- Numer licencji zawodnika jest wartością unikalną nadawaną automatycznie z chwilą zarejestrowania zawodnika w Związku. W toku eksploatacji bazy danych nie jest możliwa jego zmiana.
- Zawodnicy przynależą do klubów, każdy zawodnik w danym momencie może należeć do nie więcej niż jednego klubu.
- Zawodnik niezrzeszony nie jest członkiem klubu

---

1. Trener
    - numer licencji trenera
    - Imię
    - Nazwisko

- Trenerzy przynależą do klubów, trener (w danym momencie oraz w ujęciu historycznym) może reprezentować wiele klubów. Do klubu może należeć wielu trenerów. (wiele do wielu).
- Trener może reprezentować zawodnika.
    - Jeśli zawodnik przynależy do klubu, jego trenerem może być tylko dowolny trener aktualnie zatrudniony w klubie. W przypadku zawodników niezrzeszonych brak informacji o trenerze.

---

1. Sędzia
    - numer licencji sędziego
    - Imię
    - Nazwisko
    - Licencja ważna

- Osoba -- w ramach swej sportowej aktywności -- może być jednocześnie zawodnikiem, trenerem, sędzią . Ze względu na rozdzielenie tych ról na encje identyfikowane różnego typu licencjami, fakt ten nie ma znaczenia. Podczas konkretnych zawodów osoba taka może pełnić tylko jedną z ról i sprawdza to ich organizator.

Rejestracja zawodów

5. Lista startowa
    - zawodnik
    - stanowisko
    - odległość
    - wynik

- Lista startowa łączy zawodnika ze stanowiskiem oraz informacją o osiąganych wynikach na danych odległościach. Zawodnik może być wpisany kilkukrotnie na listę startową, jednak tylko raz przypisany do danej odległości.

Przebieg turnieju

6. Stanowisko
    - numier stanowiska

- Zawodnicy strzelają na stanowiskach. Do każdego stanowiska przypisanych są maksymalnie 2 metryczki (2 zawodników) oraz sędzia główny.
- Łączone są z zawodnikiem poprzez listę startową.

---

7. Wynik
    - id metryczki
    - wynik X
    - liczba 10
    - liczba 9
    - wynik

- Reprezentacja wyników poszczególnych zawodników. Łączona z zawodnikiem poprzez listę startową

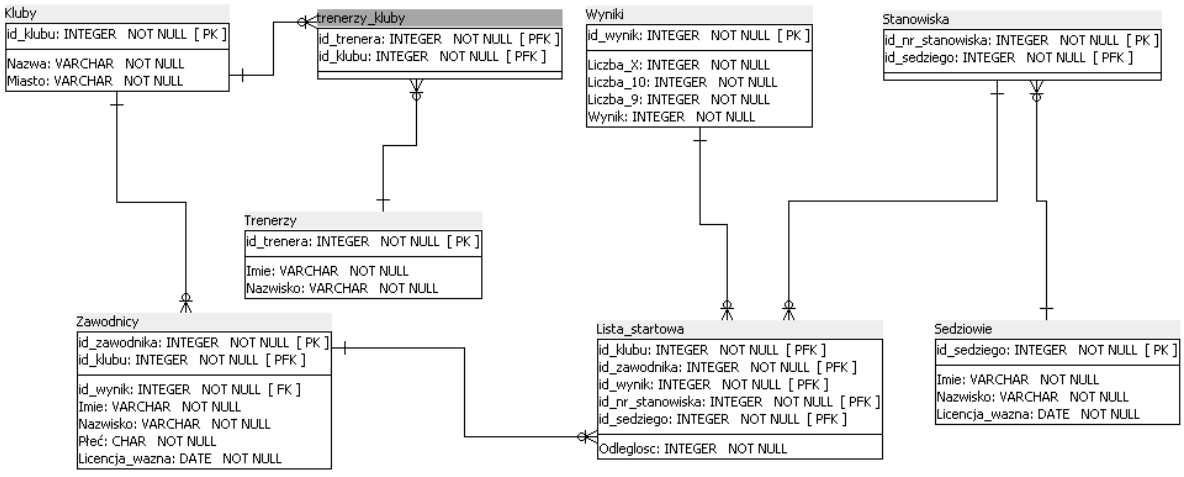
---

1. Sędzia
    - numer licencji sędziego
    - imie
    - nazwisko

- Do każdego stanowiska musi być przypisany sędzia. Nie wszyscy sędziowie musza mieć przypisane stanowiska (np. Sędziowie mogą mieć inne ustalone rol, sędzia główny, kierownik strzelań).

2.2 Zdefiniowanie relacji między encjami - diagram ERD

> Schemat budowy bazy danych oraz powiązania pomiędzy tabelami prezentuje poniższy diagram ERD.



Schemat ERD bazy danych

---

3 Projekt Logiczny

3.1 Projektowanie tabel kluczy oraz indeksów:

> Baza danch dzieli się na dwie części.
> Pierwsza część składa się z tabel Kluby, Zawodnicy, Trenerzy, Sedziowie, trener_klub .
> Odpowiadają one za zgromadzenie danych niezbędnych do dalszego przeprowadzenia turnieju.
>
> Druga część projektu opiera się na tabelach Stanowiska, Wyniki, Lista_startowa. Odpowiadają one za przebieg turnieju. Zawodnicy trafiający na listę startową są dopuszczani do zawodów, otrzymują własne metryczki (rekordy w tabeli Wyniki) oraz są przypisywani do istniejących stanowisk w tabeli Stanowiska.

Szczegóły techniczne

> Poniżej przestawiony jest kod SQL odpowiadający za utworzenie tabel oraz wykorzystywanych domen.
> Operacje muszą zostać wykonane w prezentowanej kolejności.

> Tworzenie tabeli Kluby

```
CREATE TABLE public.Kluby (
                id_klubu SERIAL ,
                Nazwa VARCHAR NOT NULL,
                Miasto VARCHAR ,
                CONSTRAINT id_klubu PRIMARY KEY (id_klubu)
);
```

> Tworzenie tabeli Zawodnicy oraz domen w niej wykorzystywanych.
> Zawodnik nie może istnieć bez klubu. Zawodnik niezrzeszony reprezentowany jest pod id_klubu = 0.

```
CREATE DOMAIN lic_zawodnik_date AS date CHECK((VALUE >'01-01-2000'::date) AND
(VALUE < (CURRENT_DATE + 365)::date));
COMMENT ON DOMAIN lic_zawodnik_date IS 'Niepoprawna data wazności licencji
zawodnika';

CREATE DOMAIN kat_char AS CHAR CHECK((VALUE = 'M'::CHAR) OR (VALUE = 'K'::CHAR));
COMMENT ON DOMAIN kat_char IS 'kategoria';

CREATE TABLE public.Zawodnicy (
                id_zawodnika SERIAL,
                id_klubu INTEGER,
                Imie VARCHAR NOT NULL,
                Nazwisko VARCHAR NOT NULL,
                Plec kat_char NOT NULL,
                Licencja_wazna lic_zawodnik_date,
                CONSTRAINT id_zawodnika PRIMARY KEY (id_zawodnika, id_klubu)
);
```

Tworzenie tabeli Trenerzy

```
CREATE TABLE public.Trenerzy (
                id_trenera SERIAL,
                Imie VARCHAR NOT NULL,
                Nazwisko VARCHAR NOT NULL,
                CONSTRAINT id_trenera PRIMARY KEY (id_trenera)
);
```

Tworzenie tabeli Sędziowie oraz domen w niej wykorzystywanych

```
CREATE DOMAIN lic_sedziowie_date AS date CHECK((VALUE >'01-01-2000'::date) AND
     (VALUE <  (CURRENT_DATE + (5*365))::date));
COMMENT ON DOMAIN lic_sedziowie_date IS 'Niepoprawna data wazności licencji
Sedziego';

CREATE TABLE public.Sedziowie (
                id_sedziego SERIAL,
                Imie VARCHAR NOT NULL,
                Nazwisko VARCHAR NOT NULL,
                Licencja_wazna lic_sedziowie_date ,
                CONSTRAINT id_sedziego PRIMARY KEY (id_sedziego)
);
```

Tworzenie tabeli Wyniki oraz domen w niej wykorzystywanych

```
CREATE DOMAIN wynik_int AS INTEGER CHECK((VALUE >=0::integer) AND (VALUE
<=720::integer));
COMMENT ON DOMAIN wynik_int IS 'wynik';

CREATE DOMAIN l_x_10_int AS INTEGER CHECK((VALUE >=0::integer) AND (VALUE
<=72::integer));
COMMENT ON DOMAIN l_x_10_int IS 'liczba X lub 10';

CREATE DOMAIN l_9_int AS INTEGER CHECK((VALUE >=0::integer) AND (VALUE
<=80::integer));
COMMENT ON DOMAIN l_9_int IS 'liczba 9';

CREATE TABLE public.Wyniki (
                id_wynik SERIAL,
                Liczba_X l_x_10_int ,
                Liczba_10 l_x_10_int ,
                Liczba_9 l_9_int,
                Wynik wynik_int ,
                CONSTRAINT id_wynik PRIMARY KEY (id_wynik)

);
```

Tworzenie tabeli Stanowiska oraz sekwencji odpowiadającej za numer stanowiska.
Stanowisko nie może istnieć bez przypisanego sędziego

```
CREATE SEQUENCE seq_nr_stan_int START 1  MAXVALUE 16;

CREATE TABLE public.Stanowiska (
                id_nr_stanowiska INTEGER DEFAULT nextval('seq_nr_stan_int') NOT
NULL,
                id_sedziego INTEGER NOT NULL,
                CONSTRAINT id_nr_stanowiska PRIMARY KEY (id_nr_stanowiska,
id_sedziego)
);
```

Relacja trenerzy_kluby (wiele do wielu) Trener może być przypisany do wielu klubów, a klub do wielu trenerów.

```
CREATE TABLE public.trenerzy_kluby (
                id_trenera INTEGER NOT NULL,
                id_klubu INTEGER,
                CONSTRAINT trenerzy_kluby_pk PRIMARY KEY (id_trenera, id_klubu)
);
```

Tworzenie tabeli Lista_startowa

```
CREATE DOMAIN odl_int AS INTEGER CHECK((VALUE =90::integer) OR (VALUE
=70::integer) OR
    (VALUE =60::integer) OR (VALUE =50::integer) OR (VALUE =30::integer));
COMMENT ON DOMAIN odl_int IS 'odleglosc';
CREATE TABLE public.Lista_startowa (
                id_nr_stanowiska INTEGER NOT NULL,
                id_sedziego INTEGER NOT NULL,
                id_klubu INTEGER,
                id_zawodnika INTEGER NOT NULL,
                id_wynik INTEGER NOT NULL,
                Odleglosc INTEGER NOT NULL,
                CONSTRAINT lista_startowa_pk PRIMARY KEY (id_zawodnika,
id_nr_stanowiska, id_sedziego, id_wynik)
);
```

Ustanawianie zależności między tabelami

```
/*
Warning: Relationship has no columns to map:
*/
ALTER TABLE public.Zawodnicy ADD CONSTRAINT kluby_zawodnicy_fk
FOREIGN KEY (id_klubu)
REFERENCES public.Kluby (id_klubu)
```

```
ON DELETE CASCADE
ON UPDATE CASCADE
NOT DEFERRABLE;

ALTER TABLE public.Stanowiska ADD CONSTRAINT sedziowie_stanowiska_fk
FOREIGN KEY (id_sedziego)
REFERENCES public.Sedziowie (id_sedziego)
ON DELETE CASCADE
ON UPDATE CASCADE
NOT DEFERRABLE;

ALTER TABLE public.trenerzy_kluby ADD CONSTRAINT trenerzy_trenerzy_kluby_fk
FOREIGN KEY (id_trenera)
REFERENCES public.Trenerzy (id_trenera)
ON DELETE CASCADE
ON UPDATE CASCADE
NOT DEFERRABLE;

ALTER TABLE public.Lista_startowa ADD CONSTRAINT stanowiska_lista_startowa_fk
FOREIGN KEY (id_nr_stanowiska, id_sedziego)
REFERENCES public.Stanowiska (id_nr_stanowiska, id_sedziego)
ON DELETE CASCADE
ON UPDATE CASCADE
NOT DEFERRABLE;

ALTER TABLE public.trenerzy_kluby ADD CONSTRAINT kluby_trenerzy_kluby_fk
FOREIGN KEY (id_klubu)
REFERENCES public.Kluby (id_klubu)
ON DELETE CASCADE
ON UPDATE CASCADE
NOT DEFERRABLE;

ALTER TABLE public.Lista_startowa ADD CONSTRAINT zawodnicy_lista_startowa_fk
FOREIGN KEY (id_zawodnika, id_klubu)
REFERENCES public.Zawodnicy (id_zawodnika, id_klubu)
ON DELETE CASCADE
ON UPDATE CASCADE
NOT DEFERRABLE;

ALTER TABLE public.Lista_startowa ADD CONSTRAINT wyniki_lista_startowa_fk
FOREIGN KEY (id_wynik)
REFERENCES public.Wyniki (id_wynik)
ON DELETE CASCADE
ON UPDATE CASCADE
NOT DEFERRABLE;
```

3.2 Słownik danych

> Opis nieoczywistych danych oraz ich ograniczeń

- Kluby
    - id_klubu -unikalny numer licencji klubu

    - Nazwa - nazwa klubu
    - Miasto - miasto w którym klub się znajduje

- Zawodnicy

    - id_zawodnika - unikalny numer licencji zawodnika
    - Plec - kategoria w której zawodnik startuje - M - mężczyźni, K - kobiety
    - Licencja_wazna - Data ważności licencji. Licencja nie może być ważna dłużej niż rok od aktualnej daty co wynika z ograniczeń związku Łuczniczego

- Trenerzy

    - id_trenera - unikalny numer licencji trenera

- Sedziowie

    - id_sedziego - unikalny numer id sędziego
    - Licencja_wazna - Data ważności licencji. Nie może być ważna dłużej niż do 5 lat od aktualnej daty co wynika z ograniczeń związku Łuczniczego. Podczas wprowadzania sędziego, sędzia musi mieć ważną licencję. Wynika to z obowiązku przedstawienia sędziów nadzorujących turniej przed rozpoczęciem zapisów zawodników.

- Wyniki - Wyniki uzyskane na odległościach przez danych zawodników

    - id_wynik - unikalne id metryczki
    - Liczba_X - liczba trafionych X-ów (X jest liczony jako 10pkt) . Suma 10-tek nie może przekraczać maksymalnego wyniku (720pkt)
    - Liczba_10 - liczba trafionych 10-tek. Suma 10-tek nie może przekraczać maksymalnego wyniku (720pkt)
    - Liczba_9 - liczba trafionych 9-tek. Suma 9-tek nie może przekraczać maksymalnego wyniku (720pkt)
    - Wynik - Wynik na danej odległości. Nie może być większy niż 720pkt, Wynik nie może przekraczać sumy wprowadzonych X, 10 oraz 9.

- Stanowiska

    - in_nr_stanowiska - numer stanowiska. Stanowisk nie może być więcej niż 16. Numerowane są kolejno podczas wprowadzania od 1 do 16

- Lista_startowa

    - Odleglosc - odleglosc, której dotyczą dane zawodnika (wynik, stanowisko). Możliwe są odległości: 90, 70, 60, 50, 30

3.3 Operacje na danych

> Przedstawiam wybrane operacje wykonywane na danych

Triggery oraz Funkcje

Lista_startowa

Trigger odpowiadający za walidację danych wprowadzanych do listy startowej.
Pozwala na zapisanie zawodnika na listę startową jeśli spełnione są warunki :
- Zawodnik ma ważną licencję
- Na stanowisku na danej odległości nie znajduje się zawodnik z innej kategorii lub tego samego klubu
- na stanowisko możliwe jest dodanie zawodnika (nie jest przekroczony limit zawodników)
- zawodnik nie jest jeszcze zapisany na danej odległości

Podczas dodawania zawodnika do listy startowej automatycznie generowana i przydzielana jest jego metryczka na daną odległość (rekord w tabeli wynik)

```
CREATE OR REPLACE FUNCTION zawodnik_do_startowej() RETURNS TRIGGER
AS '
DECLARE
aktualna_data DATE;
waznosc_lic DATE;
kat_zawodnika CHAR;
tmp INTEGER;
BEGIN
    IF EXISTS(SELECT 1 FROM Lista_startowa WHERE id_zawodnika = NEW.id_zawodnika
AND Odleglosc = NEW.Odleglosc) THEN
        RAISE NOTICE ''Zawodnik jest juz przypisany do tej odleglosci'';
        RETURN NULL;
    END IF;

    IF((SELECT COUNT(id_nr_stanowiska) FROM Lista_startowa WHERE id_nr_stanowiska
= NEW.id_nr_stanowiska AND Odleglosc = NEW.Odleglosc) >= 2 ) THEN
        RAISE NOTICE ''Za duzo zawodnikow na stanowisku'';
        RETURN NULL;
    END IF;

    IF EXISTS(SELECT 1 FROM Lista_startowa WHERE id_klubu = NEW.id_klubu AND
id_nr_stanowiska = NEW.id_nr_stanowiska AND Odleglosc = NEW.Odleglosc) THEN
        RAISE NOTICE ''Zawodnicy z tego samego klubu nie moga byc na
stanowisku'';
        RETURN NULL;
    END IF;

    SELECT Plec INTO kat_zawodnika FROM Zawodnicy WHERE id_zawodnika =
NEW.id_zawodnika;
    IF EXISTS(SELECT 1 FROM Lista_Startowa ls WHERE ls.id_nr_stanowiska =
NEW.id_nr_stanowiska AND (SELECT Plec FROM Zawodnicy WHERE
    id_zawodnika = ls.id_zawodnika ) != kat_zawodnika  AND ls.Odleglosc =
NEW.Odleglosc) THEN
        RAISE NOTICE ''Zawodnicy z Roznych kategorii nie moga byc na
stanowisku'';
        RETURN NULL;
    END IF;

    SELECT CURRENT_DATE INTO aktualna_data;
    SELECT Licencja_wazna INTO waznosc_lic FROM Zawodnicy WHERE id_zawodnika =
NEW.id_zawodnika;
    IF (waznosc_lic < aktualna_data) THEN
```

```
        RAISE NOTICE ''Zawodnik nie ma waznej licencji'';
        RETURN NULL;
    END IF;
    IF (NEW.id_wynik = 0) THEN
        INSERT INTO Wyniki VALUES(DEFAULT);
        SELECT MAX(id_wynik )INTO tmp FROM Wyniki;
        NEW.id_wynik = tmp;
    END IF;
    RETURN NEW;
END;
' LANGUAGE 'plpgsql';

CREATE TRIGGER zawodnik_do_startowej_trig BEFORE INSERT ON Lista_startowa
FOR EACH ROW EXECUTE PROCEDURE zawodnik_do_startowej();
```

> Trigger odpowiadający za walidację danych aktualizowanych w liście startowej.
> Kontroluje czy zmienione nie zostały dane wykluczające zawodnika ze startu jak ważność licencji, czy też zapisanie zawodnika na stanowisku z zawodnikiem z z innej kategorii lub tego samego klubu.

```
CREATE OR REPLACE FUNCTION zawodnik_do_startowej_update() RETURNS TRIGGER
AS '
DECLARE
aktualna_data DATE;
waznosc_lic DATE;
kat_zawodnika CHAR;
tmp INTEGER;
BEGIN


    IF EXISTS(SELECT 1 FROM Lista_startowa WHERE id_klubu = NEW.id_klubu AND
id_nr_stanowiska = NEW.id_nr_stanowiska AND Odleglosc = NEW.Odleglosc) THEN
        RAISE NOTICE ''Zawodnicy z tego samego klubu nie moga byc na
stanowisku'';
        RETURN NULL;
    END IF;

    SELECT Plec INTO kat_zawodnika FROM Zawodnicy WHERE id_zawodnika =
NEW.id_zawodnika;
    IF EXISTS(SELECT 1 FROM Lista_Startowa ls WHERE ls.id_nr_stanowiska =
NEW.id_nr_stanowiska AND (SELECT Plec FROM Zawodnicy WHERE
    id_zawodnika = ls.id_zawodnika ) != kat_zawodnika  AND ls.Odleglosc =
NEW.Odleglosc) THEN
        RAISE NOTICE ''Zawodnicy z Roznych kategorii nie moga byc na
stanowisku'';
        RETURN NULL;
    END IF;

    SELECT CURRENT_DATE INTO aktualna_data;
    SELECT Licencja_wazna INTO waznosc_lic FROM Zawodnicy WHERE id_zawodnika =
NEW.id_zawodnika;
    IF (waznosc_lic < aktualna_data) THEN
```

```
        RAISE NOTICE ''Zawodnik nie ma waznej licencji'';
        RETURN NULL;
    END IF;

    RETURN NEW;
END;
' LANGUAGE 'plpgsql';

CREATE TRIGGER zawodnik_do_startowej_update_trig BEFORE UPDATE ON Lista_startowa
FOR EACH ROW EXECUTE PROCEDURE zawodnik_do_startowej_update();
```

> Funkcja upraszczająca dodawanie danych do Listy startowej.
> Wymaga podania jedynie:
> -id_zawodnika
> -id_nr_stanowiska
> -Odległość

```
CREATE OR REPLACE FUNCTION dodaj_do_listy(INTEGER, INTEGER, INTEGER) RETURNS void
AS '
DECLARE
id_zaw ALIAS FOR $1;
id_stan ALIAS FOR $2;
odl ALIAS FOR $3;

id_k INTEGER;
id_s INTEGER;


BEGIN
    SELECT id_klubu INTO id_k FROM Zawodnicy WHERE id_zawodnika = id_zaw;
    SELECT id_sedziego INTO id_s FROM Stanowiska WHERE id_nr_stanowiska =
id_stan;
    INSERT INTO Lista_startowa VALUES(id_stan, id_s, id_k, id_zaw, 0, odl);

END;
'LANGUAGE 'plpgsql';
```

> Funkcja upraszczająca edycję danych w Liście startowej.
> Wymaga podania jedynie:
> -id_zawodnika
> -id_nr_stanowiska
> -Odległość poprzednia
> -Odległość nowa

```
CREATE OR REPLACE FUNCTION aktualizuj_liste(INTEGER,INTEGER, INTEGER, INTEGER)
RETURNS void
AS '
```

```
DECLARE
id_zaw ALIAS FOR $1;
id_stan ALIAS FOR $2;
odl_stara ALIAS FOR $3;
odl ALIAS FOR $4;

id_k INTEGER;
id_s INTEGER;


BEGIN
    SELECT id_sedziego INTO id_s FROM Stanowiska WHERE id_nr_stanowiska =
id_stan;
    UPDATE Lista_startowa SET id_nr_stanowiska= id_stan, id_sedziego = id_s,
Odleglosc = odl WHERE
    id_zawodnika = id_zaw AND Odleglosc = odl_stara ;

END;
'LANGUAGE 'plpgsql';
```

Sędziowie oraz stanowiska

> Trigger odpowiadający za sprawdzenie czy dodawany sędzia ma ważną licencję.

```
CREATE OR REPLACE FUNCTION nowy_sedzia() RETURNS TRIGGER
AS '
DECLARE
    aktualna_data DATE;
    wazn_lic DATE;
BEGIN
    SELECT CURRENT_DATE INTO aktualna_data;
    wazn_lic := NEW.Licencja_wazna;
    IF (wazn_lic > aktualna_data) THEN
        RETURN NEW;
    ELSE
        RAISE NOTICE ''Sedzia nie ma waznej licencji'';
        RETURN NULL;
    END IF;
END;
' LANGUAGE 'plpgsql';

CREATE TRIGGER nowy_sedzia_trig BEFORE INSERT OR UPDATE ON Sedziowie
FOR EACH ROW EXECUTE PROCEDURE nowy_sedzia();
```

> Trigger odpowiadający za sprawdzenie czy dodawany sędzia może zostać przypisany do danego stanowiska. (Czy nie jest już przypisany do zbyt dużej liczby stanowisk).

```
CREATE OR REPLACE FUNCTION sedzia_do_stan() RETURNS TRIGGER
```

```
AS '
BEGIN
    IF EXISTS(SELECT 1 FROM Stanowiska WHERE id_nr_stanowiska =
NEW.id_nr_stanowiska) THEN
        RAISE NOTICE ''Stanowisko istnieje z przypisanym innym sędzią'';
        RETURN NULL;
    END IF;
    IF ( ( SELECT COUNT(id_sedziego) FROM Stanowiska WHERE id_sedziego =
NEW.id_sedziego) < 4 ) THEN
      RETURN NEW;
    ELSE
        RAISE NOTICE ''Za duzo stanowisk na jednego sedziego'';
        RETURN NULL;
    END IF;
END;
'LANGUAGE 'plpgsql';

CREATE TRIGGER sedzia_do_stan BEFORE INSERT OR UPDATE ON Stanowiska
FOR EACH ROW EXECUTE PROCEDURE sedzia_do_stan();
```

Kluby

> Trigger orpowiadający za przepisanie zawodników jako niezrzeszonych po usunięciu klubu. Zawodnicy zostają dalej na liście startowej a ich wyniki są zachowywane.

```
CREATE OR REPLACE FUNCTION usun_klub() RETURNS TRIGGER
AS '
BEGIN
UPDATE Zawodnicy SET id_klubu = 0 WHERE id_klubu = old.id_klubu;
RETURN OLD;
END;
' LANGUAGE 'plpgsql';

CREATE TRIGGER usun_klub BEFORE DELETE ON Kluby
FOR EACH ROW EXECUTE PROCEDURE usun_klub();
```

Wyniki

> Trigger odpowiadający za walidację wprowadzanego wyniku. Suma z trafień w punktowane obszary tarczy nie może przekraczać mozliwego maksymalnego wyniku

```
CREATE OR REPLACE FUNCTION dodaj_wynik() RETURNS TRIGGER
AS '
BEGIN
    IF((New.Liczba_X + New.Liczba_10 + New.Liczba_9) > NEW.Wynik) THEN
        RAISE NOTICE ''Niemozliwy wynik'';
        RETURN NULL;
    ELSE
```

```
        RETURN NEW;
    END IF;
END;
'LANGUAGE 'plpgsql';

CREATE TRIGGER nowy_wynik BEFORE INSERT OR UPDATE ON Wyniki
FOR EACH ROW EXECUTE PROCEDURE dodaj_wynik();

CREATE OR REPLACE FUNCTION sumuj_wyniki(INTEGER) RETURNS INTEGER
AS '
DECLARE
id_zaw ALIAS FOR $1;
sum INTEGER;
tmp INTEGER;
id_w INTEGER;
BEGIN
    sum := 0;
    FOR id_w IN SELECT id_wynik FROM Lista_startowa WHERE id_zaw = id_zawodnika
    LOOP
        SELECT wynik INTO tmp FROM Wyniki WHERE id_wynik = id_w;
        IF tmp IS NULL THEN
            UPDATE Wyniki SET Liczba_X = 0, Liczba_10 = 0, Liczba_9 = 0, Wynik =
0 WHERE id_wynik = id_w;
            SELECT wynik INTO tmp FROM Wyniki WHERE id_wynik = id_w;
        END IF;
        sum := sum + tmp;
    END LOOP;
    RETURN sum;
END;
'LANGUAGE 'plpgsql';
```

Funkcja pomagająca we wprowadzaniu wyników zawodników

```
CREATE OR REPLACE FUNCTION dodaj_aktualizuj_wynik(INTEGER, INTEGER, INTEGER,
INTEGER, INTEGER, INTEGER) RETURNS void
AS '
DECLARE
id_zaw ALIAS FOR $1;
odl ALIAS FOR $2;
wyn ALIAS FOR $3;
l_X ALIAS FOR $4;
l_10 ALIAS FOR $5;
l_9 ALIAS FOR $6;

BEGIN
   UPDATE Wyniki SET Wynik = wyn, Liczba_X = l_X, Liczba_10 = l_10, Liczba_9 =
l_9 WHERE
         id_wynik =(SELECT id_wynik FROM Lista_startowa WHERE Odleglosc = odl AND
id_zawodnika = id_zaw ) ;
```

```
END;
'LANGUAGE 'plpgsql';
```

> Funkcja odpowiadająca za sumowanie wyników zawodnika ze wszystkich odległości na których startował. Zwraca otrzymany wynik.

```
CREATE OR REPLACE FUNCTION sumuj_wyniki(INTEGER) RETURNS INTEGER
AS '
DECLARE
id_zaw ALIAS FOR $1;
sum INTEGER;
tmp INTEGER;
id_w INTEGER;
BEGIN
    sum := 0;
    FOR id_w IN SELECT id_wynik FROM Lista_startowa WHERE id_zaw = id_zawodnika
    LOOP
        SELECT wynik INTO tmp FROM Wyniki WHERE id_wynik = id_w;
        IF tmp IS NULL THEN
            UPDATE Wyniki SET Liczba_X = 0, Liczba_10 = 0, Liczba_9 = 0, Wynik =
0 WHERE id_wynik = id_w;
            SELECT wynik INTO tmp FROM Wyniki WHERE id_wynik = id_w;
        END IF;
        sum := sum + tmp;
    END LOOP;
    RETURN sum;
END;
'LANGUAGE 'plpgsql';
```

> Funkcja odpowiadająca za sumowanie wyników zawodników z tego samego klubu ze wszystkich odległości. Zwraca otrzymany wynik.

```
CREATE OR REPLACE FUNCTION sumuj_wyniki_klub(INTEGER) RETURNS INTEGER
AS '
DECLARE
id_k ALIAS FOR $1;
id_z INTEGER;
sum INTEGER;
tmp INTEGER;
BEGIN
    sum := 0;
    FOR id_z IN SELECT DISTINCT id_zawodnika FROM Lista_startowa WHERE id_klubu =
id_k
    LOOP
        tmp := sumuj_wyniki(id_z);
        sum := sum + tmp;
    END LOOP;
    RETURN sum;
END;
'LANGUAGE 'plpgsql';
```

Widoki

> Wynik grupujący dane do wyników na odległościach, tworzący ranking zawodników posortowanych po odległościach, kategoriach i statystykach na danej odległości.

```
CREATE VIEW Wyniki_odleglosci AS SELECT row_number() over(partition by
ls.Odleglosc, z.Plec ORDER BY z.Plec, ls.Odleglosc DESC, w.wynik DESC, w.Liczba_X
DESC,
 w.Liczba_10 DESC, w.Liczba_9 DESC) as lp, z.Plec AS Plec, z.imie AS Imie,
z.nazwisko AS Nazwisko, k.nazwa AS Nazwa, k.miasto AS Miasto, ls.Odleglosc AS
Odleglosc,
  w.wynik AS Wynik FROM Kluby k, Zawodnicy z, Wyniki w, Lista_startowa ls WHERE
ls.id_zawodnika = z.id_zawodnika  AND ls.id_wynik = w.id_wynik AND ls.id_klubu =
k.id_klubu ;
```

> Zestawienie informacji o zawodnikach oraz ich wynikach na wszystkich odległościach

```
CREATE VIEW zestawienie AS SELECT z.id_zawodnika, z.Plec, z.Imie, z.Nazwisko,
k.nazwa, k.miasto, ls.Odleglosc, w.wynik, w.Liczba_X, w.Liczba_10, w.Liczba_9,
sumuj_wyniki(z.id_zawodnika)
FROM Zawodnicy z , Wyniki w, Kluby k, Lista_startowa ls WHERE z.id_zawodnika =
ls.id_zawodnika AND z.id_klubu = k.id_klubu AND ls.id_wynik = w.id_wynik;
```

> Wyniki zawodnika ogólnie, tworzy ranking zawodników posortowanych po kategoriach i wynikach ogólnych. Korzysta z funkcji sumuj_wynik()

```
CREATE VIEW Wyniki_ogolne_tmp AS SELECT DISTINCT ON(id_zawodnika) row_number()
over(partition by Plec ORDER BY Plec DESC,sumuj_wyniki(id_zawodnika) DESC) AS lp,
Plec, Imie, Nazwisko, nazwa, miasto,
 sumuj_wyniki FROM zestawienie ;

CREATE VIEW Wyniki_ogolne AS SELECT DISTINCT  lp, Plec, Imie, Nazwisko, nazwa,
miasto,
 sumuj_wyniki FROM Wyniki_ogolne_tmp ORDER BY Plec, lp;
```

> Wyniki teamowe, tworzy ranking klubów ze wzgledu na sumy wyników

```
CREATE VIEW Wyniki_team AS SELECT row_number() over( ORDER BY
sumuj_wyniki_klub(k.id_klubu) DESC) as lp, k.nazwa, k.miasto,
sumuj_wyniki_klub(k.id_klubu) FROM Kluby k WHERE k.id_klubu != 0 ;
```

> Widok sortujący zawodników ze względu na klub i wyświetlający podstawowe informacje na ich temat.

```
CREATE VIEW Sklady_klubowe_zarejestrowane AS SELECT k.nazwa, k.miasto, z.imie,
z.nazwisko, z.id_zawodnika
    FROM Kluby k, Zawodnicy z WHERE k.id_klubu = z.id_klubu ORDER BY k.nazwa;
```

> Widok prezentujący rozstawienie zawodników bedących na liście startowej na stanowiskach posortowanych ze względu na odległość oraz numer stanowiska

```
CREATE VIEW Rozstawienie_zawodnikow_dopuszczonych AS SELECT ls.id_nr_stanowiska,
ls.Odleglosc, z.imie, z.nazwisko, k.nazwa, k.miasto
    FROM Lista_startowa ls, Zawodnicy z, Kluby k WHERE ls.id_zawodnika =
z.id_zawodnika AND z.id_klubu = k.id_klubu
    ORDER BY ls.Odleglosc DESC, ls.id_nr_stanowiska;
```

> Widok pokazujący zawodników jeszcze nie przypisanych do listy startowej na żadnej odległości.

```
REATE VIEW zawodnicy_niezapisani AS SELECT z.id_zawodnika, z.Imie, z.Nazwisko,
z.Plec FROM Zawodnicy z
    WHERE z.id_zawodnika NOT IN (SELECT id_zawodnika FROM Lista_startowa);
```

Inne

> Ponadto w kodzie aplikacji znajdują się liczne kwerendy typu SELECT, INSERT, UPDATE, DELETE odpowiadające za podstawowe operacje na bazie danych z posiomu aplikacji. Często wykorzystują one opisane powyżej widoki oraz funkcje.

---

3 Projekt aplikacji

Ogólne

> Aplikacja wykonana została przy użyciu języka PHP z wykorzystaniem HTML oraz CSS. Język PHP odpowiada za połączenie oraz wykonywanie poleceń w bazie danych.
> Za połączenie z bazą danych odpowiada plik db_connect.php oraz funkcja connect na samej górze pliku index.php. Operacje pracujące na danych wprowadzonych poprzez formularze wykonywane są poprzez skrypty znajdujące się w katalogach kluby, sedziowie, trenerzy, wyniki, zawodnicy. Katalogi te zawierają odpowiednio pogrupowane skrypty. Tak przykładowo skrypt odpowiadający za dodanie nowego sędziego znajduje się w katalogu sedziowie a za aktualizację wyniku w katalogu wyniki.
> Wszystkie funkcje wykonujące kwerendy SELECT znajdują się w pliku index.php.

Połączenie z bazą danych

> W celu połączenia z bazą danych inną niż domyślnie ustawiona należy zmienić w pliku db_connect.php oraz funkcji connect w pliku index.php odpowiednio nazwa_hosta, port, dbname, user, password na własne.

```
$host = "host = nazwa_hosta";
$port = "port = numer_portu";
$dbname = "dbname = nazwa_bazy";
$credentials = "user = nazwa_użytkownika password=haslo";
```

Kod

> Kod PostgreSQL znajduje się w katalogu SQL. W celu zbudowania bazy oraz zaimportowania przykładowych danych należy wczytać z plik scripts.sql.

- SQL/create_tables.sql - Tworzenie tabel
- SQL/pop_functions.sql - Stworzenie Triggerów, funkcji, widoków
- SQL/pop_tables.sql - Wczytanie przykładowych danych
- SQL/drop_tables.sql - Usunięcie bazy wraz z triggerami, funkcjami, widokami
- SQL/scripts.sql - Wykonanie zestawu poleceń budujących bazę i wypełniających przykładowymi danymi

> Kod aplikacji znajduje się w katalogu APP

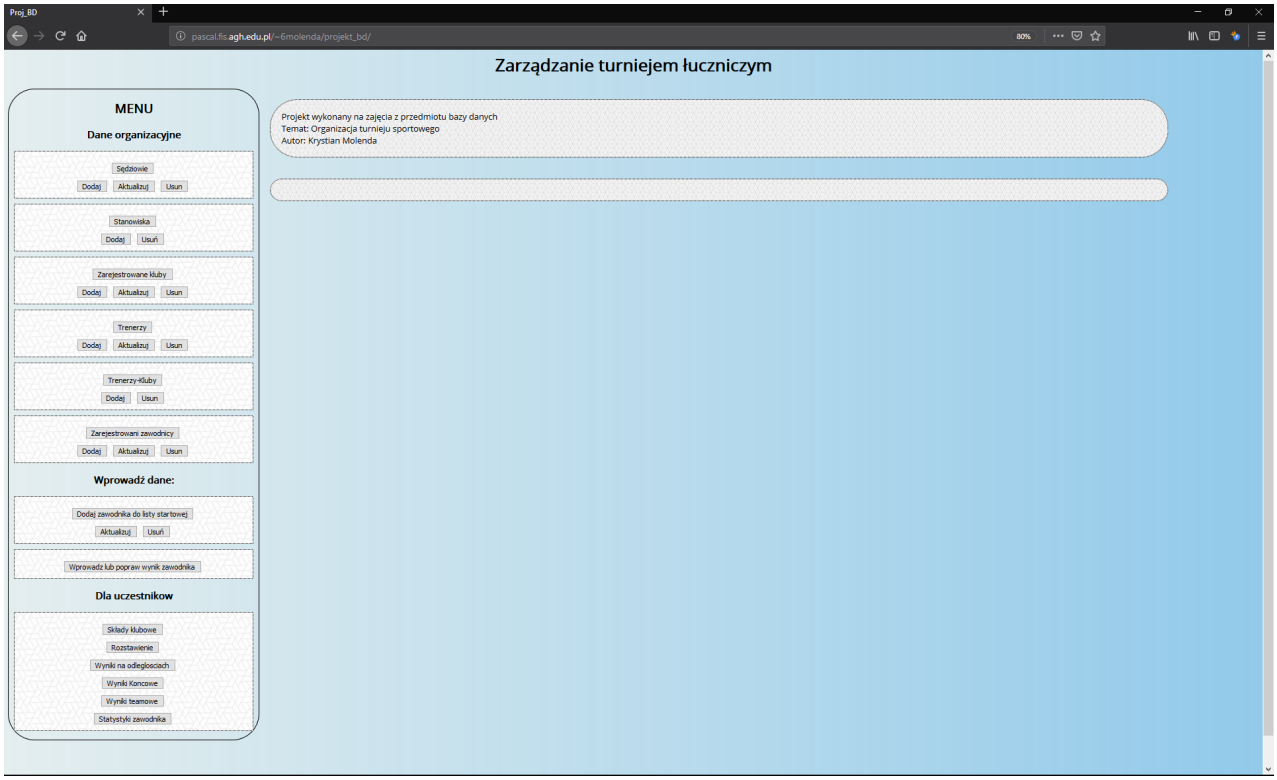
- APP/kluby - skrypty obsługujące formularze dotyczące klubów
  - aktualizuj_klub.php - aktualizacja danych klubu
  - nowy_klub.php - dodawanie klubu
  - usun_klub.php - usuwanie klubu z bazy
- APP/sedziowie - skrypty obsługujące formularze dotyczące sędziów oraz stanowisk
  - aktualizuj_sedziego.php - aktualizacja danych sędziego
  - nowe_stanowisko.php - dodawanie stanowiska
  - nowy_sedzia.php - dodawanie sędziego
  - usun_sedziego.php - usuwanie sędziego z bazy
  - usun_stanowisko.php - usuwanie stanowiska z bazy
- APP/trenerzy - skrypty obsługujące formularze dotyczące trenerów
  - aktualizuj_trenera.php - aktualizacja danych trenera
  - nowy_trener - dodawanie trenera
  - nowy_trener_klub.php - nowa relacja trener-klub
  - usun_trener_klub.php - usuwanie wybranej relacji trener-klub
  - usun_trenera.php - usuwanie trenera z bazy
- APP/wyniki - skrytpy obsługujące formularze dotyczące wyników
  - nowy_wynik.php - dodawanie/poprawianie wyniku dla danego zawodnika na danej odległości
- APP/zawodnicy - skrypty obsługujące formularze dotyczące zawodników
  - aktualizuj_liste.php - aktualizacja danych na liscie startowej
  - aktualizj_zawodnika.php - aktualizacja danych zawodnika
  - dodaj_do_listy_zaw.php - dodawanie zawodnika do listy startowej
  - nowy_zawodnik.php - dodawanie zawodnika
  - staty.php - Wyswietlanie statystyk wybranego zawodnika
  - usun_z_listy.php - usuwanie zawodnika z listy startowej

    - usun_zawodnika.php - usuwanie zawodnika
- APP/db_connect.php - skrypt obsługujący połączenie z bazą postgreSQL
- APP/index.css - css
- APP/index.php - aplikacja

---

4 Dokumentacja

> Po uruchomieniu aplikacji (index.php) poprzez przeglądarkę (zalecana Google Chrome lub Firefox) po lewej stronie widzimy menu aplikacji a po prawej część, w której wyświetlane będą dane lub formularze do wprowadzania danych.



Widok po otwarciu aplikacji

> Jak widać powyżej menu po lewej stronie podzielone jest na trzy sekcje.

- Dane organizacyjne - Podstawowe dane potrzebne do rozpoczęcia turnieju
- Wprowadź dane - Rejestrowanie zawodników na listę startową oraz wprowadzanie wyników
- Dla zawodników - Wyswietlanie Rozstawień, wyników oraz sprawdzanie statystyk wybranego zawodnika

Wprowadzanie danych organizacyjnych

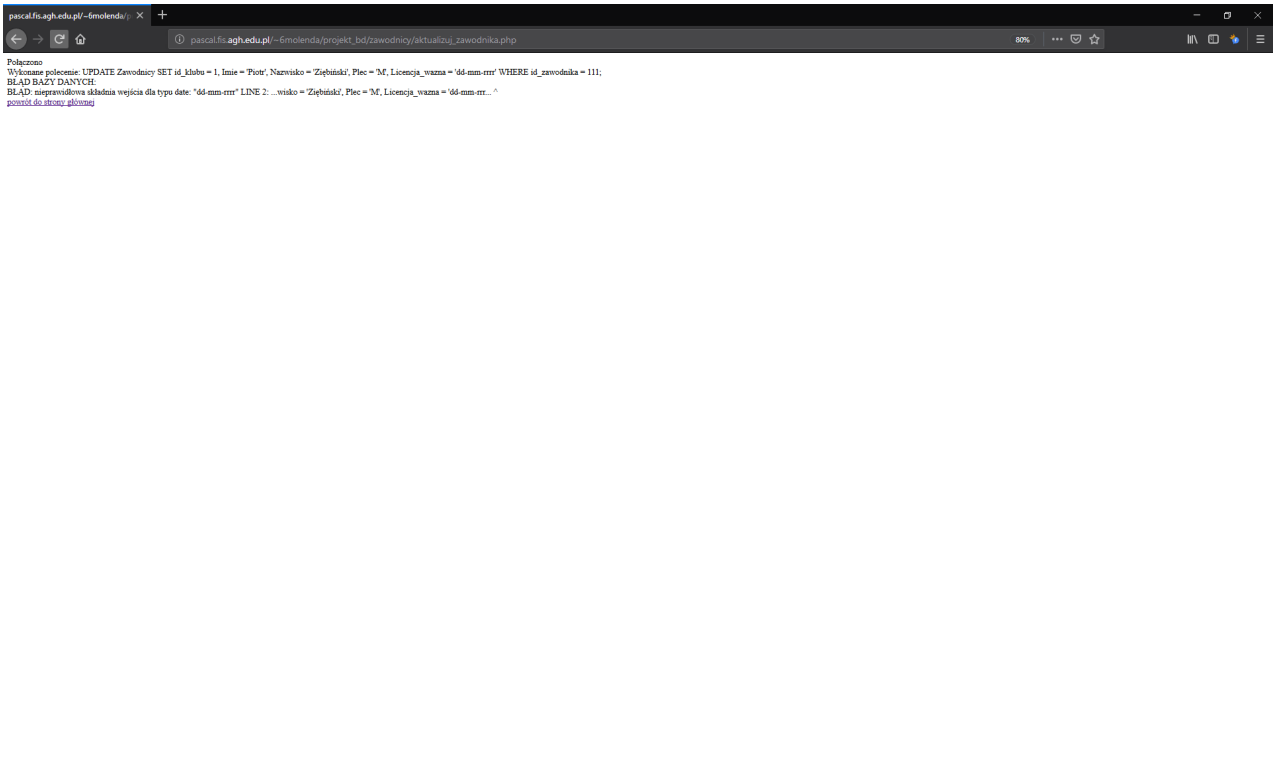
> Do każdej tabeli danych istnieje możliwość wprowadzenia nowych danych, ich usuwania, późniejszej edycji (z wyjątkiem id, błędne wprowadzenie id wymusza usunięcie rekordu i stworzenie go od nowa) oraz oczywiście wyświetlania.
> Tabela Stanowiska nie posiada możliwości edycji, ponieważ sędzia jest przypisany do stanowiska na czas trwania całych zawodów.
> Operację, którą chcemy wykonać należy wybrać klikając odpowiedni guzik w menu. Następnie po wypełnieniu pól danymi klikamy przycisk zatwierdzający polecenie. Następuje powrót do ekranu, który widzielismy zaraz po uruchomieniu aplikacji. Oznacza to poprawne wykonanie polecenia. Możemy

sprawdzić czy wynik został dodany sprawdzając informacje o odpowiedniej tabeli. W przypadku niepowodzenia otwarte zostanie nowe okno, gdzie pojawi się informacja o błędzie zwróconym przez bazę danych. Po zapoznaniu się z kompunikatem klikamy "powróć do strony głównej" aby wrócić do aplikacji i spróbować wykonać polecenie jeszcze raz.

Dodawanie zawodnika

UWAGA: Klub powinien zostać dodany przed dodaniem zawodnika. W przypadku dodawania zawodnika niezrzeszonego należy w numer licencji klubu wpisać 0. W przypadku dodania klubu po dodaniu zawodnika, można zaktualizować tą informację w Zawodnik->aktualizuj.

TODO WPISANY ZAWODNIK LISTA

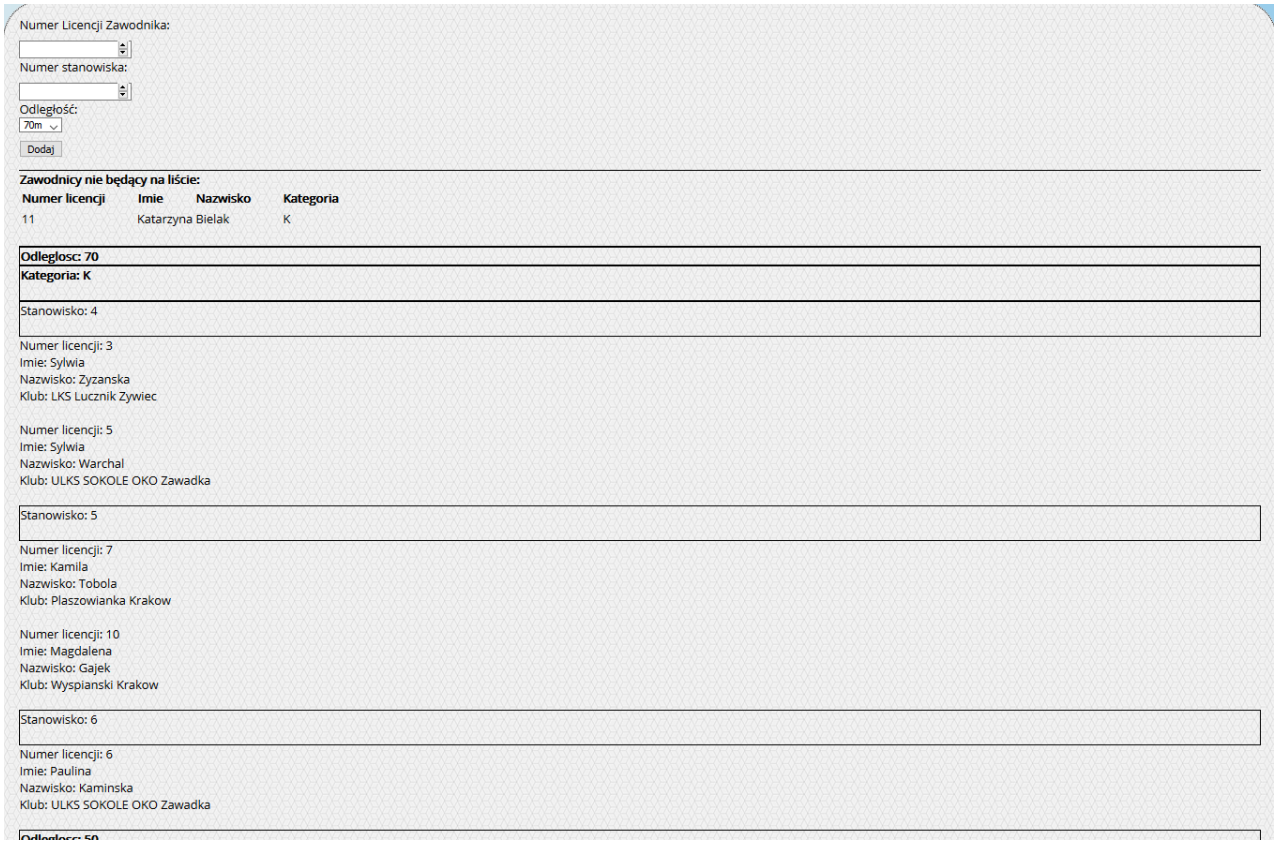


Próba aktualizacji danych zawodnika i zmiany daty ważności licencji na błędny format. W przypadku niepowodzenia otwarte zostanie nowe okno, gdzie pojawi się informacja o błędzie zwróconym przez bazę danych. Po zapoznaniu się z kompunikatem klikamy "powróć do strony głównej" aby wrócić do aplikacji i spróbować wykonać polecenie jeszcze raz.

Wprowadzanie danych turniejowych

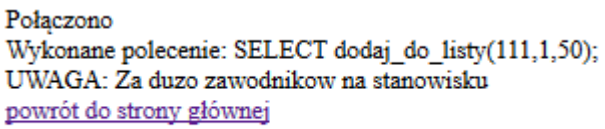
Rejestracja zawodników

Jeżeli dane organizacyjne zostały poprawnie wprowadzone można przystąpić do przypisywania zawodników do listy startowej.



Dodawanie zawodnika do listy startowej.

Możliwe jest niepowodzenie w dodawaniu zawodnika do listy startowej. Pojawi się wtedy komunikat z informacją, dlaczego nie jest możliwe zapisanie zawodnika. Przyczynami mogą być: brak ważnej licencji, zawodnik już jest zapisany na daną odległość, na stanowisku do którego próbujemy przypisać zawodnika jest już maksymalna ich liczba, na stanowisku jest już zawodnik z danego klubu, kategorie zawodników na stanowiskach mogą być różne



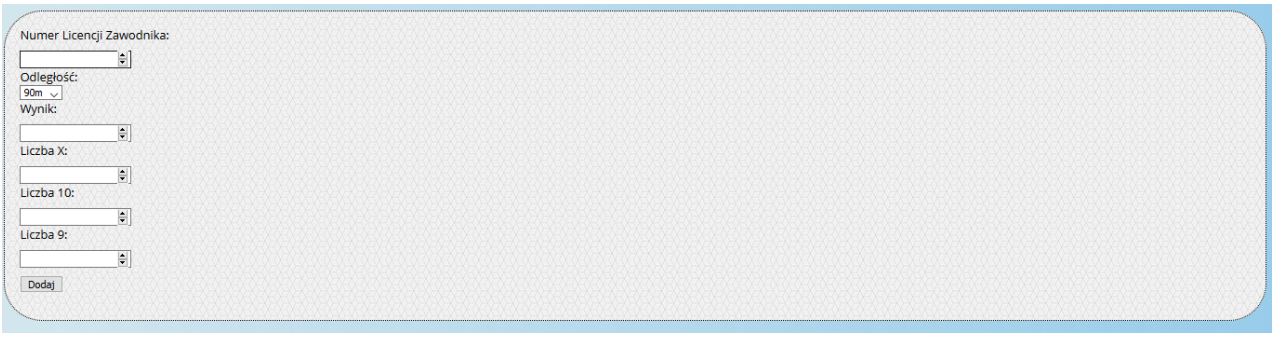
Próba przypisania zbyt dużej liczby zawodnikow do stanowiska

Po dodaniu zawodników do listy startowej pojawią się oni w informacjach dla uczestników-Rostawieniu, oraz wynikach (przy czym wszyscy będą mieli wyniki równe 0pkt).

Fragment ekranu prezentującego rozstawienia zawodników na stanowiskach na danych odległościach

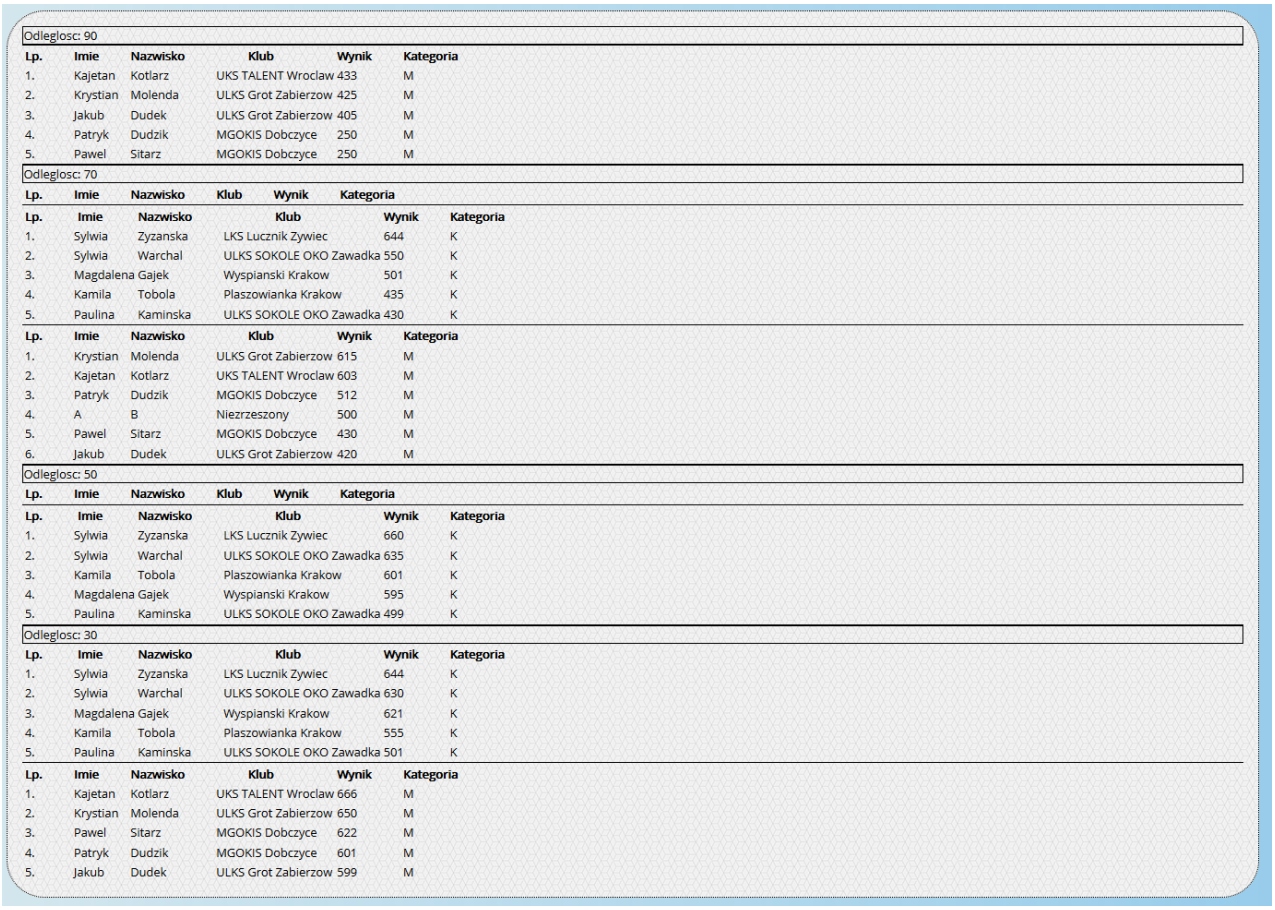
Wpisywanie wyników oraz ich odczyt

Można również w tym momencie przystąpić do wpisywania zawodnikom uzyskanych wyników. W informacjach dla zawodników- Wyniki na odległosciach, Wyniki Końcowe, teamowe oraz statystykach zawodnika obserwować możemy zmiany. Rankingi ustalane są automatycznie.



Wprowadzanie przykładowego wyniku dla wprowadzonego zawodnika

Po wprowadzeniu możemy sprawdzić rankingi zawodników na odległościach lub też rankingi teamowe klikając odpowiednio zakładki Wyniki na odległościach, Wyniki Koncowe, Wyniki teamowe



Odleglosc: 90

Lp.	Imie	Nazwisko	Klub	Wynik	Kategoria
1.	Kajetan	Kotlarz	UKS TALENT Wroclaw	433	M
2.	Krystian	Molenda	ULKS Grot Zabierzow	425	M
3.	Jakub	Dudek	ULKS Grot Zabierzow	405	M
4.	Patryk	Dudzik	MGOKIS Dobczyce	250	M
5.	Pawel	Sitarz	MGOKIS Dobczyce	250	M

Odleglosc: 70

Lp.	Imie	Nazwisko	Klub	Wynik	Kategoria

Lp.	Imie	Nazwisko	Klub	Wynik	Kategoria
1.	Sylwia	Zyzanska	LKS Lucznik Zywiec	644	K
2.	Sylwia	Warchal	ULKS SOKOLE OKO Zawadka	550	K
3.	Magdalena	Gajek	Wyspianski Krakow	501	K
4.	Kamila	Tobola	Plaszowianka Krakow	435	K
5.	Paulina	Kaminska	ULKS SOKOLE OKO Zawadka	430	K

Lp.	Imie	Nazwisko	Klub	Wynik	Kategoria
1.	Krystian	Molenda	ULKS Grot Zabierzow	615	M
2.	Kajetan	Kotlarz	UKS TALENT Wroclaw	603	M
3.	Patryk	Dudzik	MGOKIS Dobczyce	512	M
4.	A	B	Niezrzeszony	500	M
5.	Pawel	Sitarz	MGOKIS Dobczyce	430	M
6.	Jakub	Dudek	ULKS Grot Zabierzow	420	M

Odleglosc: 50

Lp.	Imie	Nazwisko	Klub	Wynik	Kategoria

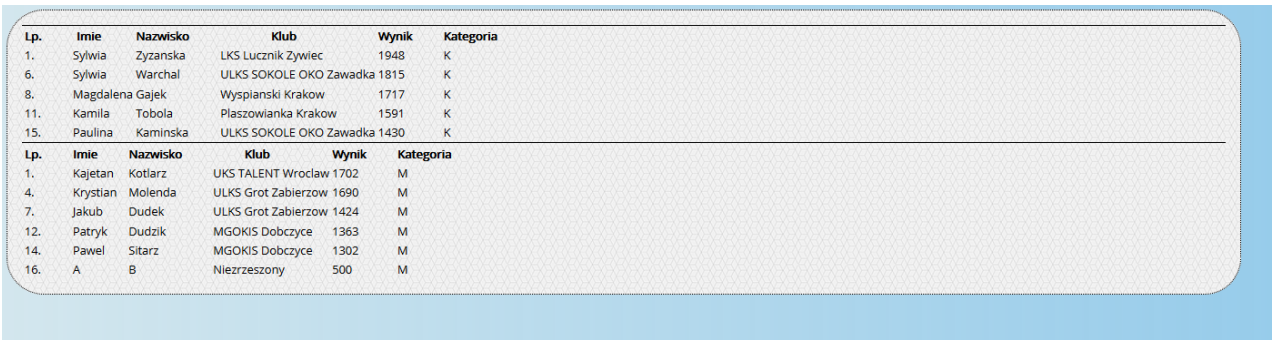
Lp.	Imie	Nazwisko	Klub	Wynik	Kategoria
1.	Sylwia	Zyzanska	LKS Lucznik Zywiec	660	K
2.	Sylwia	Warchal	ULKS SOKOLE OKO Zawadka	635	K
3.	Kamila	Tobola	Plaszowianka Krakow	601	K
4.	Magdalena	Gajek	Wyspianski Krakow	595	K
5.	Paulina	Kaminska	ULKS SOKOLE OKO Zawadka	499	K

Odleglosc: 30

Lp.	Imie	Nazwisko	Klub	Wynik	Kategoria
1.	Sylwia	Zyzanska	LKS Lucznik Zywiec	644	K
2.	Sylwia	Warchal	ULKS SOKOLE OKO Zawadka	630	K
3.	Magdalena	Gajek	Wyspianski Krakow	621	K
4.	Kamila	Tobola	Plaszowianka Krakow	555	K
5.	Paulina	Kaminska	ULKS SOKOLE OKO Zawadka	501	K

Lp.	Imie	Nazwisko	Klub	Wynik	Kategoria
1.	Kajetan	Kotlarz	UKS TALENT Wroclaw	666	M
2.	Krystian	Molenda	ULKS Grot Zabierzow	650	M
3.	Pawel	Sitarz	MGOKIS Dobczyce	622	M
4.	Patryk	Dudzik	MGOKIS Dobczyce	601	M
5.	Jakub	Dudek	ULKS Grot Zabierzow	599	M

Wyniki na odległościach



Lp.	Imie	Nazwisko	Klub	Wynik	Kategoria
1.	Sylwia	Zyzanska	LKS Lucznik Zywiec	1948	K
6.	Sylwia	Warchal	ULKS SOKOLE OKO Zawadka	1815	K
8.	Magdalena	Gajek	Wyspianski Krakow	1717	K
11.	Kamila	Tobola	Plaszowianka Krakow	1591	K
15.	Paulina	Kaminska	ULKS SOKOLE OKO Zawadka	1430	K

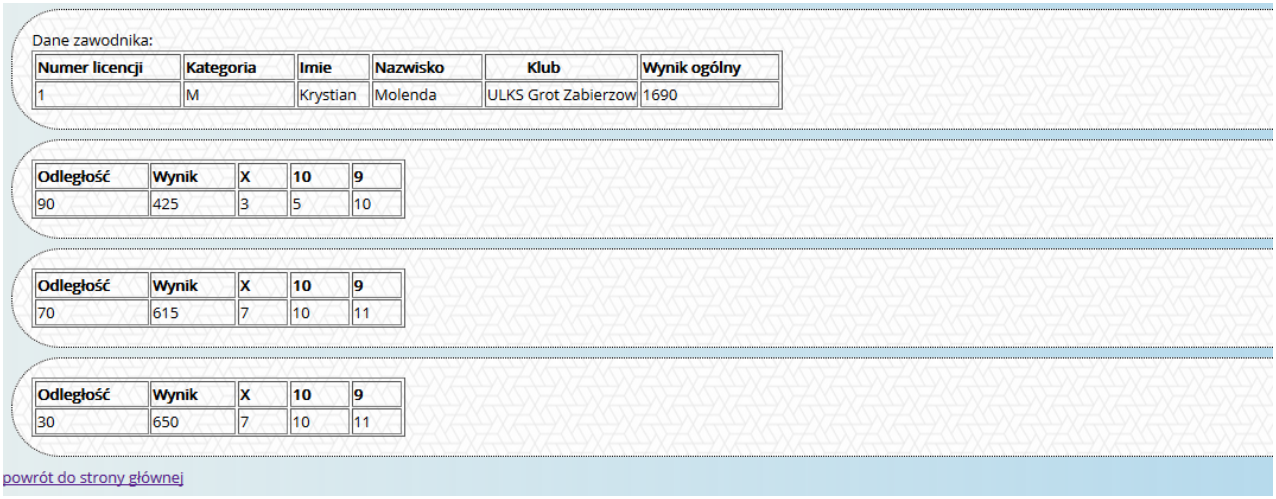
Lp.	Imie	Nazwisko	Klub	Wynik	Kategoria
1.	Kajetan	Kotlarz	UKS TALENT Wroclaw	1702	M
4.	Krystian	Molenda	ULKS Grot Zabierzow	1690	M
7.	Jakub	Dudek	ULKS Grot Zabierzow	1424	M
12.	Patryk	Dudzik	MGOKIS Dobczyce	1363	M
14.	Pawel	Sitarz	MGOKIS Dobczyce	1302	M
16.	A	B	Niezrzeszony	500	M

Wyniki koncowe, tj. sumy wyników na wszystkich odległościach

Lp.	Klub	Wynik
1	ULKS SOKOLE OKO Zawadka	3245
2	ULKS Grot Zabierzow	3114
3	MGOKIS Dobczyce	2665
4	LKS Lucznik Zywiec	1948
5	Wyspianski Krakow	1717
6	UKS TALENT Wroclaw	1702
7	Plaszowianka Krakow	1591

Wyniki teamowe

Istnieje opcja wyszukania informacji o statystykach jednego zawodnika. W tym celu wchodzimy w zakładkę Statystyki zawodnika i podajemy numer licencji zawodnika, którego osiągnięcia i dane chcemy oglądać. W przypadku podania nieprawidłowego numeru licencji nie otrzymamy żadnych danych.



Dane zawodnika:

Numer licencji	Kategoria	Imie	Nazwisko	Klub	Wynik ogólny
1	M	Krystian	Molenda	ULKS Grot Zabierzow	1690

Odległość	Wynik	X	10	9
90	425	3	5	10

Odległość	Wynik	X	10	9
70	615	7	10	11

Odległość	Wynik	X	10	9
30	650	7	10	11

powrót do strony głównej

Statystyki zawodnika o numerze licencji 111. Możemy dokładnie sprawdzić liczbę trafień w najbardziej punktowane obszary, oraz wyniki na wszystkich odległościach.