POLITECHNIKA POZNAŃSKA
WYDZIAŁ ELEKTRYCZNY, INFORMATYKA
SEMESTR VII

BAZY DANYCH - Projekt

RAFAŁ JENDRASZAK
INDEKS: 121286

1 CHARAKTERYSTYKA PROJEKTU

1.1 Opis

Celem projektu jest zaimplementowanie systemu zarządzania meczami dla sędziów piłki ręcznej. Zadaniami do wykonania są zaprojektowanie bazy danych, implementacja logiki aplikacji po stronie bazy danych oraz stworzenie prostego interfejsu dla użytkownika, który będzie umożliwiał wykorzystanie funkcji stworzonych po stronie bazy danych.

1.2 Wykorzystane technologie

Do realizacji wykorzystano język Python wraz z biblioteką Django oraz baza danych PostgreSQL.

Język Python został wybrany, ponieważ jest łatwy do nauczenia, a kod napisany w tym jeżyku jest przejrzysty i czytelny. Biblioteka Django zostanie wykorzystana do stworzenia interfejsu graficznego w postaci strony internetowej, umożliwia ona w prosty sposób manipulację na obiektach bazodanowych, np. dodawanie, edycja, usuwanie, czytanie danych z tabel.

Początkowo wybrana została baza danych MySQL, jednak na etapie tworzenia projektu, napotkano przeszkodę w postaci połączenia kodu aplikacji z bazą danych, dlatego zdecydowano na zmianę na PostgreSQL. Jest to jeden z najpopularniejszych otwartych systemów zarządzania relacyjnymi bazami danych.

1.3 Architektura oprogramowania

Poniższy diagram przedstawia w prosty sposób architekturę aplikacji.

Użytkownik (Klient) łączy się ze stroną internetową, która jest generowana dla niego przez aplikację (Aplikacja w Django), korzystając z danych dostępnych w bazi (Baza danych PostgreSQL)

Rysunek 1: Architektura oprogramowania

2 Baza Danych

2.1 Skrypty tworzące strukturę SZBD

2.1.1 Drużyna

```
create table referee_team
    (
        id        serial       not null
            constraint referee_team_pkey
                primary key,
        name      varchar(100) not null,
        city      varchar(50)  not null,
        telephone varchar(11)
    );
```

2.1.2 Sędziowie szczegóły

```
create table referee_refereedetails
(
    id            serial  not null
        constraint referee_refereedetails_pkey
            primary key,
    referee_level integer not null,
    city          varchar(50),
    telephone     varchar(11),
    user_id       integer not null
        constraint referee_refereedetails_user_id_key
            unique
        constraint referee_refereedetails_user_id_c9200df3_fk_auth_user_id
            references auth_user
            deferrable initially deferred
);
```

2.1.3 Status meczu

```
create table referee_matchstatus
    (
        id          serial       not null
            constraint referee_matchstatus_pkey
                primary key,
        name        varchar(100) not null,
        next_status integer
    );
```

2.1.4 Kategoria meczu

```
create table referee_matchcategory
(
    id   serial      not null
        constraint referee_matchcategory_pkey
            primary key,
    name varchar(50) not null
);
```

2.1.5 Mecz

```
create table referee_match
(
    id                serial                   not null
        constraint referee_match_pkey
            primary key,
    match_number      varchar(10)              not null,
    date_time         timestamp with time zone not null,
    away_team_id      integer
        constraint referee_match_away_team_id_fkey
            references referee_team
            deferrable initially deferred,
    home_team_id      integer
        constraint referee_match_home_team_id_fkey
            references referee_team
            deferrable initially deferred,
    match_category_id integer
        constraint referee_match_match_category_id_fkey
            references referee_matchcategory
            deferrable initially deferred,
    referee_a_id      integer
        constraint referee_match_referee_a_id_fkey
            references auth_user
            deferrable initially deferred,
    referee_b_id      integer
        constraint referee_match_referee_b_id_fkey
            references auth_user
            deferrable initially deferred,
    match_status_id   integer
        constraint referee_match_match_status_id_fkey
            references referee_matchstatus
            deferrable initially deferred
);
```

2.1.6 Wyniki meczu

```
create table referee_matchresult
(
    id              serial  not null
        constraint referee_matchresult_pkey
            primary key,
    home_team_goals integer,
    away_team_goals integer,
    winner          varchar(1),
    match_id        integer not null
        constraint referee_matchresult_match_id_ef3a6230_fk_referee_match_id
            references referee_match
            deferrable initially deferred
);
```

2.2 Diagramy ER

2.2.1 Dodanie meczu

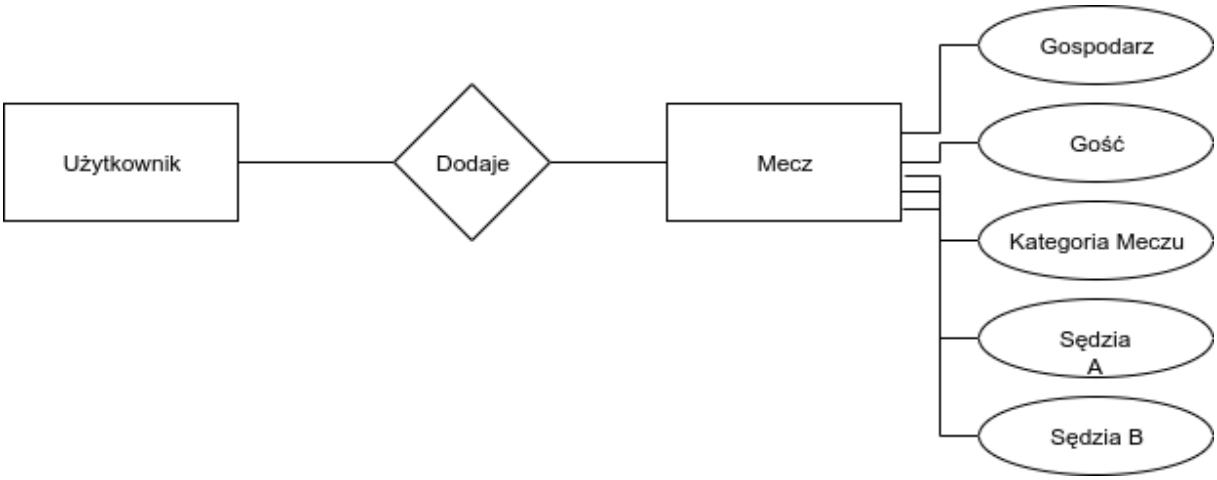


Diagram 1

2.2.2 Zmiana statusu meczu

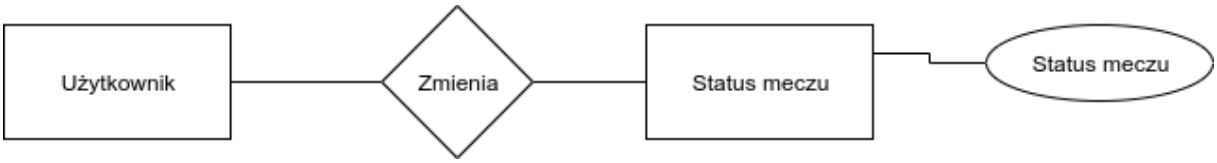


Diagram 2

2.2.3 Dodanie wyniku meczu

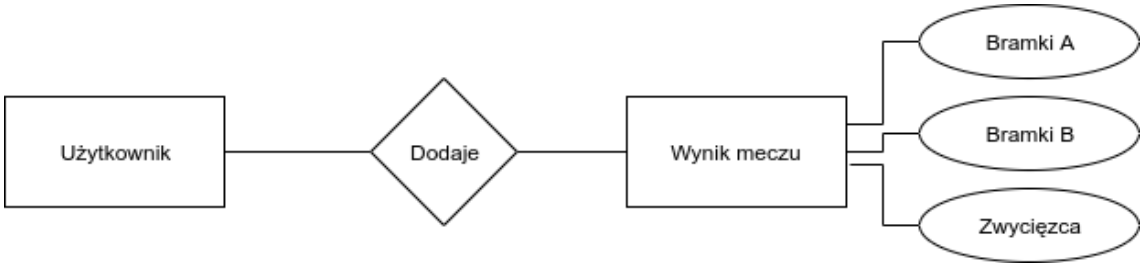
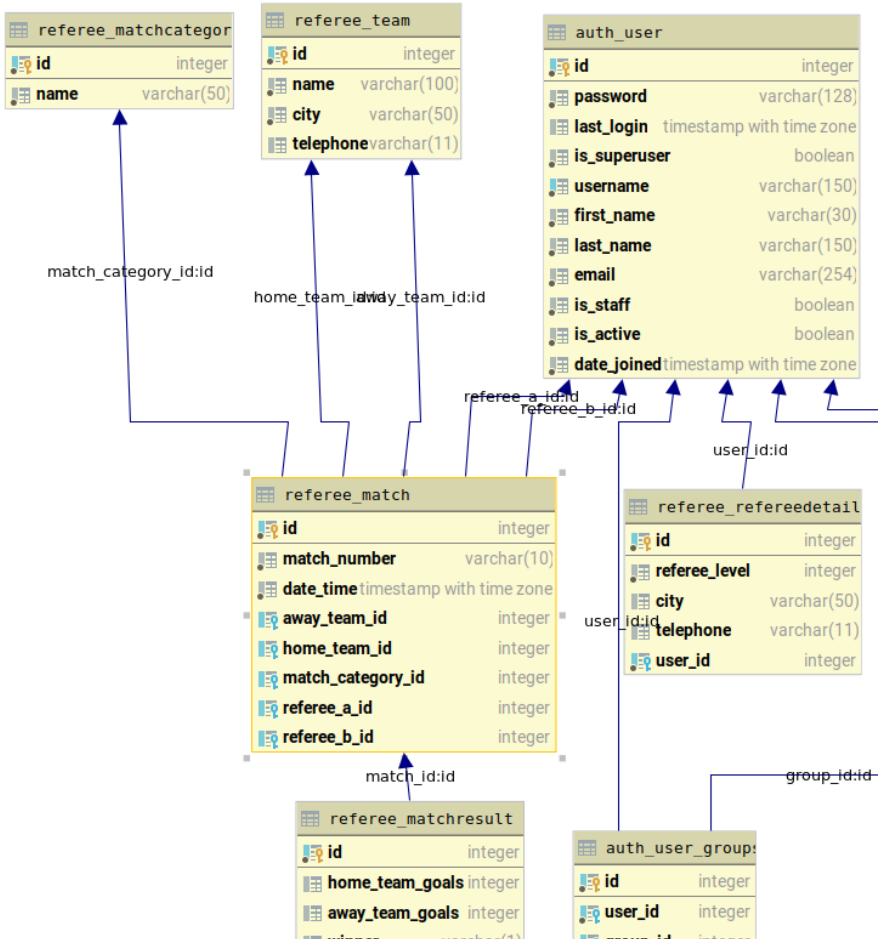


Diagram 3

2.3 Diagram DB



3 PRZYKŁADY ROZWIĄZAŃ

3.1 Triggery

3.1.1 Sprawdzenie wyniku i wpisanie do tabeli zwycięzcy meczu

```
create trigger t_insert_winner after insert or update on referee_matchresult
    for each row
    execute procedure f_insert_winner();

drop trigger t_insert_winner on referee_matchresult;

create or replace function f_insert_winner()
returns trigger as
    $BODY$
    BEGIN
        if new.home_team_goals > new.away_team_goals then
            update referee_matchresult set winner = 'A' where id = new.id;
        else
            update referee_matchresult set winner = 'B' where id = new.id;
        end if;

        return new;
    end;
    $BODY$

language plpgsql volatile;
```

Podczas prezentacji projektu na zajęciach, powyższy trigger działał tylko w przypadku, gdy dokonano edycji meczu i dodania wyniku. Aktualnie działa również w przypadku dodania nowego rekordu do bazy danych.

3.1.2 Ustawienie statusu meczu na ‘Nowy’, jeśli status nie został przypisany podczas dodawania meczu

```
create trigger t_set_new_status after insert on referee_match
    for each row
    execute procedure f_set_new_status();

create or replace function f_set_new_status()
returns trigger as
    $BODY$
    BEGIN
        if new.match_status_id isnull then
            update referee_match set match_status_id = 1 where id = new.id;
        end if;

        return new;
    end;
    $BODY$
language plpgsql volatile;
```

3.2 Widoki

3.2.1 Wyświetlenie meczy wraz z wynikami oraz sędziami

```
create or replace view v_match_details as
select
        rm.match_number AS MATCH_NUMBER,
        rm.date_time AS DATE_TIME,
        rmc.name AS MATCH_CATEGORY,
        rtA.name AS HOME_TEAM,
        rtB.name as AWAY_TEAM,
        auA.last_name as REFEREE_A,
        auB.last_name as REFEREE_B,
        rmr.home_team_goals as HOME_TEAM_GOALS,
        rmr.away_team_goals as AWAY_TEAM_GOALS,
        null as id,
        rm.id as match_id,
        rms.name as match_status,
        rmr.winner as match_winner
from
     referee_match rm
join referee_matchcategory rmc on rm.match_category_id = rmc.id
join referee_team rtA on rm.home_team_id = rtA.id
join referee_team rtB on rm.away_team_id = rtB.id
left join referee_matchresult rmr on rm.id = rmr.match_id
join auth_user auA on rm.referee_a_id = auA.id
join auth_user auB on rm.referee_b_id = auB.id
left join referee_matchstatus rms on rm.match_status_id = rms.id;
```

3.2.2 Wyświetlanie danych sędziego

```
create or replace view v_ref_details as
select
 au.first_name,
 au.last_name,
 rr.city,
 rr.referee_level,
 rr.telephone
from
 auth_user au
join referee_refereedetails rr on au.id = rr.user_id;
```

3.3 Procedury

3.3.1 Wyświetlenie drużyn z danego miasta

```
create or replace function f_show_teams_from_city(t_city varchar)
returns setof referee_team
    language sql
    as
    $$
        select * from referee_team rt where rt.city = t_city;
    $$;
```

3.3.2 Wyświetlenie sędziów z danego miasta

```
create or replace function f_show_ref_from_city(r_city varchar)
returns setof v_ref_details
language sql
as
    $$
        select * from v_ref_details vrd where vrd.city = r_city;
    $$;
```

3.3.3 Wyświetlenie meczy, gdzie dana drużyna jest gospodarzem/gościem

```
create or replace function f_show_matches_where_team_is_home(f_team varchar)
returns setof v_match_details
language sql
as
    $$
```

```
        select * from v_match_details vmd where vmd.home_team = f_team;
    $$;

create or replace function f_show_matches_where_team_is_away(f_team varchar)
returns setof v_match_details
language sql
as
    $$
        select * from v_match_details vmd where vmd.away_team = f_team;
    $$;
```

3.3.4 Zmiana statusu meczu

```
create or replace procedure f_change_match_status(f_match_id integer)
language plpgsql
as
    $$
    BEGIN
        update
            referee_match
        set
            match_status_id = (
                select next_status from referee_matchstatus where id = (select
match_status_id from referee_match where id = f_match_id)
                )
        where
        id = f_match_id;

        commit;
    end;
    $$;
```

3.4 Transakcje

3.4.1 Dodanie nowego meczu

```
BEGIN;
INSERT INTO referee_match
VALUES ((SELECT MAX(id) + 1 FROM referee_match),
        'IMB/60',
        '2019-11-30 18:00:00.000000',
        6,
        5,
        1,
        1,
        5,
        2);
COMMIT;
```

3.4.2 Dodanie nowego zespołu

```
BEGIN;
INSERT INTO referee_team
VALUES (
        (SELECT MAX(id) + 1 FROM referee_team),
        'Sambor Tczew',
        'Tczew'
       );
COMMIT;
```