Kalkulator v2 - baza danych do obsługi wyborów

Dominik Kasperski

22 lutego 2018

1. Projekt koncepcji, założenia.

1.1. Zdefiniowanie tematu projektu.

Tematem projektu jest przygotowanie bazy danych do obsługi wyborów. W systemie który ona opisuje, głosowanie i liczenie głosów odbywa się w obwodach. W obwodzie jeden z członków komisji wprowadza dane do systemu. Następnie w okręgach składających się z obwodów odbywa się sumowanie głosów. Kandydat może być wystawiony w okręgu. Obwody dodaje się w okręgu. Na podstawie protokołów z okręgów i obwodów PKW dokonuje podziału mandatów i publikuje dane o wynikach. PKW także ma strzec poprawności wyborów. Wyznacza okręgi wyborcze i komisarzy.
W czasie najbliższych wyborów samorządowych w Polsce protokoły z okręgowych komisji wyborczych mają być skanowane i ładowane na stronę PKW - koszt zakupu skanerów to ok. 150 mln zł. W aplikacji ma dziać się odwrotnie - protokoły generowane są przez system i dostępne na stronie a później mogą być wydrukowane. Baza danych ma przechowywać informacje o obwodach, komisjach, okręgach, kandydatach i wynikach. Rozdział mandatów to osobne zagadnienie.

1.2. Analiza wymagań użytkownika.

Użytkownik oczekuje prostego narzędzia do przeprowadzania wyborów, skalowalnego, bezpiecznego. W bazie musi być mozliwość wprowadzania i edycji okręgów, obwodów, kandydatów i członków komisji. Powinny istnieć tabele z informacjami o możliwych nadużyciach uzupełniane automatycznie. Powinny istnieć możliwości wyświetlania protokołu dla obwodu i okręgu. Powinna istnieć gradacja dostępu do funkcjonalności.

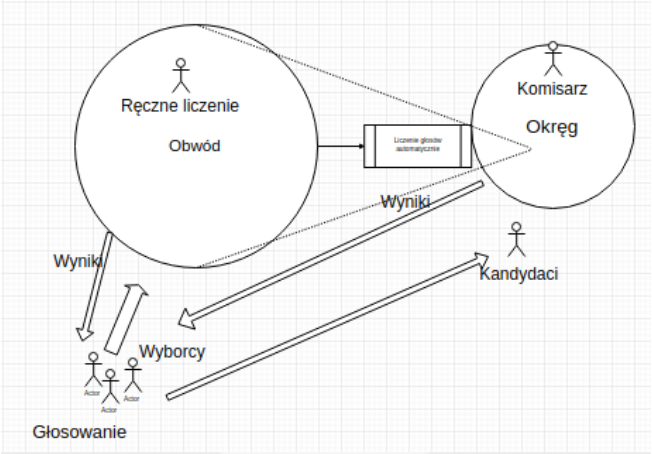
1.3. Zaprojektowanie funkcji.

Podstawowe funkcje bazy danych to przechowywanie i gromadzenie informacji o wynikach wyborów na różnych szczeblach. Powinna istnieć funkcja przeliczenia głosów w obwodach w ramach jednego okręgu oraz widoki umożliwiające łatwą generację raportów. Dodatkowo należy zaprojektować wyzwalacze monitorujące licznie głosów. Informacje o kandydacie powinny zawierać jego przynależność do komitetu wyborczego (w celu obsługi systemów proporcjonalnych), informacje o okręgu i obwodach w nim się znajdujących, informację o tym kto odpowiada za licznie głosów.

2. Projekt diagramów konceptualny.

2.1. Budowa i analiza diagramu przepływu danych.

Przepływ informacji odbywa się w wielu kierunkach. Wyborcy i głosują i oczekują wyników, kandydaci kandydują w całym okręgu ale nie w całym kraju, głosy liczone sa w obwodach, ale potrzebne są wyniki okręgu. Zależności pokazuje diagram (2.1). Wyróżnione są na nim 4 grupy osób - komisarze, kandydaci, wyborcy, członkowie komisji. Komisarz jest jeden w okręgu, kandydat kandyduje w jednym okręgu. Każdy obwód to odbicie okręgu, z tym, że to w obwodzie odbywa się głosowanie i



Rys. 1: *Data Flow Diagram*

liczenie ręczne. Z obwodu i okręgu muszą iść informacje o wynikach kandydatów i komitetów oraz o frekwencji liczonej tym ile osób głosowało. Muszą istnieć wejścia takie jak: wprowadzanie okręgów, obwodów, kandydatów i głosów, wyjścia będące raportami z okręgu i obwodu. Procesy to liczenie głosów w okręgu, pilnowanie przebiegu głosowania. Przechowywanie danych o listach wyborczych, kandydatach, członkach komisji, komisarzach i liczbiw głosów w obwodzie i okregu.

2.2. Zdefiniowanie encji oraz ich atrybutów.

W bazie danych muszą istnieć encje opisujące:

- Okręg (lokalizacja, komisarz).
- Kandydata (imię, nazwisko).
- Komitety wyborcze (nazwa).
- Obwód (lokalizacja, przynależność do okręgu, komisja wyborcza).
- Przebieg głosowania (komu ile głosów przybyło).
- Członków komisji (imię, nazwisko).
- Liczbę głosów w obwodzie.
- Liczbę głosów w okręgu.

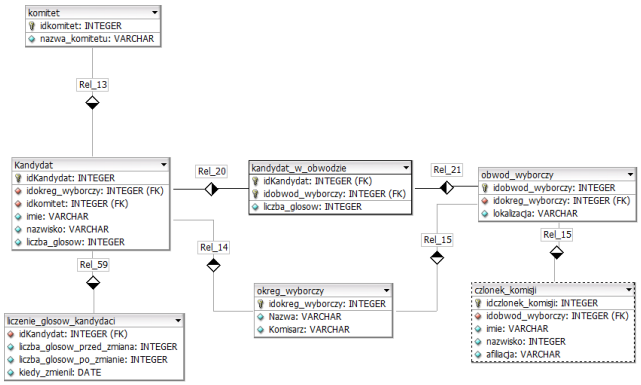
2.3. Zaprojektowanie relacji pomiędzy encjami.

Powinny występować następujące klucze główne:

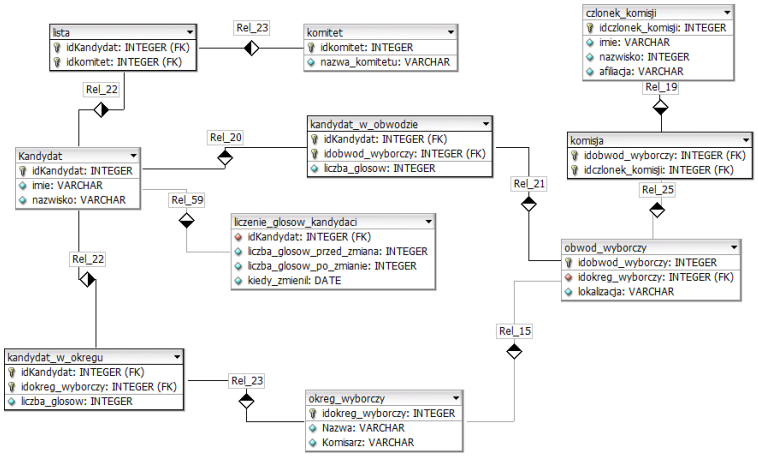
- Numer okręgu.
- Numer obwodu.
- ID kandydata.
- Numer listy wyborczej.
- ID członka komisji.

Powiązania między encjami:

- W okręgu jest wielu kandydatów którzy są tylko w nim: **relacja 1:n**.



Rys. 2: Optymalny diagram encji.



Rys. 3: Diagram encji po produkcji.

- Każdy kandydat może przynależeć do jednej listy: **relacja 1:n**.
- W okręgu jest wiele obwodów które są tylko w nim: **relacja 1:n**.
- W jednym obwodzie może dochodzić do wielu nadużyć: **relacja 1:n**.
- W wielu obwodach kandydat ma wiele wyników: **relacja n:m**.
- W okręgu kandydat ma jeden wynik: **relacja 1:n**.

2.4. Zaprojektowanie relacji pomiędzy encjami.

Diagram (2.4) przedstawia optymalny diagram encji wynikający z powyższych założeń.

3. Projekt logiczny

3.1. Projektowanie tabel, kluczy, indeksów.

W rzeczywistości zastosowany diagram wyglądał nieco inaczej, jak na rysunku (3.1). Wynikało to z ewolucji projektu w trakcie realizacji: kilka błędnych rozwiązań zostało zaimplementowanych, a część zmian ma na celu optymalizację. Dodatkowo istniała także *symetryczna* część dla obsługi referendów jednak nie została wzięta pod uwagę. Przede wszystkim pojawiło się kilka encji słownikowych takich jak *lista, komisja* czy *kandydat w okręgu*. Skrypt instalujący całą bazę wygląda następująco:

```
CREATE TABLE okreg_wyborczy (
  idokreg_wyborczy SERIAL ,
  Nazwa VARCHAR    ,
  Komisarz VARCHAR      ,
PRIMARY KEY(idokreg_wyborczy));


CREATE TABLE komitet (
  idkomitet SERIAL ,
  nazwa_komitetu VARCHAR      ,
PRIMARY KEY(idkomitet));


CREATE TABLE czlonek_komisji (
  idczlonek_komisji SERIAL ,
  imie VARCHAR    ,
  nazwisko VARCHAR    ,
  afiliacja VARCHAR      ,
PRIMARY KEY(idczlonek_komisji));



CREATE TABLE obwod_wyborczy (
  idobwod_wyborczy SERIAL ,
  idokreg_wyborczy INTEGER   NOT NULL ,
  lokalizacja VARCHAR      ,
PRIMARY KEY(idobwod_wyborczy),
  FOREIGN KEY(idokreg_wyborczy)
    REFERENCES okreg_wyborczy(idokreg_wyborczy) ON DELETE CASCADE
    ON UPDATE CASCADE);


CREATE INDEX IFK_Rel_15 ON obwod_wyborczy (idokreg_wyborczy);


CREATE TABLE Kandydat (
  idKandydat SERIAL ,
  imie VARCHAR    ,
  nazwisko VARCHAR      ,
PRIMARY KEY(idKandydat));


CREATE TABLE lista (
  idKandydat INTEGER   NOT NULL ,
  idkomitet INTEGER   NOT NULL ,
PRIMARY KEY(idKandydat, idkomitet),
  FOREIGN KEY(idKandydat)
    REFERENCES Kandydat(idKandydat) ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY(idkomitet)
    REFERENCES komitet(idkomitet) ON DELETE CASCADE
    ON UPDATE CASCADE);
```

```
CREATE INDEX IFK_Rel_33 ON lista (idKandydat);
CREATE INDEX IFK_Rel_34 ON lista (idkomitet);



CREATE TABLE komisja (
  idobwod_wyborczy INTEGER   NOT NULL ,
  idczlonek_komisji INTEGER  NOT NULL    ,
PRIMARY KEY(idczlonek_komisji, idobwod_wyborczy),
  FOREIGN KEY(idobwod_wyborczy)
    REFERENCES obwod_wyborczy(idobwod_wyborczy) ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY(idczlonek_komisji)
    REFERENCES czlonek_komisji(idczlonek_komisji) ON DELETE CASCADE
     ON UPDATE CASCADE);

CREATE INDEX IFK_Rel_255 ON komisja (idobwod_wyborczy);
CREATE INDEX IFK_Rel_199 ON komisja (idczlonek_komisji);


CREATE TABLE kandydat_w_okregu (
  idKandydat INTEGER   NOT NULL ,
  idokreg_wyborczy INTEGER   NOT NULL ,
  liczba_glosow INTEGER      ,
PRIMARY KEY(idKandydat, idokreg_wyborczy),
  FOREIGN KEY(idKandydat)
    REFERENCES Kandydat(idKandydat) ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY(idokreg_wyborczy)
    REFERENCES okreg_wyborczy(idokreg_wyborczy) ON DELETE CASCADE
    ON UPDATE CASCADE);


CREATE INDEX IFK_Rel_22 ON kandydat_w_okregu (idKandydat);
CREATE INDEX IFK_Rel_23 ON kandydat_w_okregu (idokreg_wyborczy);


CREATE TABLE kandydat_w_obwodzie (
  idKandydat INTEGER   NOT NULL ,
  idobwod_wyborczy INTEGER   NOT NULL ,
  liczba_glosow INTEGER      ,
PRIMARY KEY(idKandydat, idobwod_wyborczy),
  FOREIGN KEY(idKandydat)
    REFERENCES Kandydat(idKandydat) ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY(idobwod_wyborczy)
    REFERENCES obwod_wyborczy(idobwod_wyborczy) ON DELETE CASCADE
    ON UPDATE CASCADE);


CREATE INDEX IFK_Rel_220 ON kandydat_w_obwodzie (idKandydat);
CREATE INDEX IFK_Rel_221 ON kandydat_w_obwodzie (idobwod_wyborczy);
```

```
CREATE TABLE liczenie_glosow_kandydaci (
  idKandydat INTEGER   NOT NULL ,
  liczba_glosow_przed_zmiana INTEGER    ,
  liczba_glosow_po_zmianie INTEGER    ,
  kiedy_zmienil DATE    ,
  FOREIGN KEY(idKandydat)
    REFERENCES Kandydat(idKandydat) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE INDEX IFK_Rel_59 ON liczenie_glosow_kandydaci (idKandydat);
```

3.2. Słownik danych

$komitet = nazwakomitet + \underline{idkomitet}$
$lista = \underline{idkomitet} + \underline{idkandydat}$
$kandydat = \underline{idkandydat} + imie + nazwisko$
$kandydatwokregu = \underline{idkandydat} + \underline{idokregwyborczy} + liczbaglosow$
$okregwyborczy = \underline{idokregwyborczy} + lokalizacja + komisarz$
$obwodwyborczy = \underline{idobwodwyborczy} + \underline{idokregwyborczy} + lokalizacja$
$kandydatwobwodzie = \underline{idkandydat} + \underline{idobwodwyborczy} + liczbaglosow$
$czlonekkomisji = \underline{idczlonekkomisji} + imie + nazwisko + (afiliacja)$
$komisja = \underline{idczlonekkomisji} + \underline{idobwodwyborczy}$
$liczenieglosowkandydaci = \underline{idkandydat} + liczbaglosowprzedzmiana + liczbaglosowpozmianie$
Wszystkie klucze to liczby całkowite nieujemne i większe od zera, reszta to VARCHARy.

3.3. Zaprojektowanie operacji na danych.

Dodano także widoki: *protokół w okręgu, okw, protokół w obwodzie, frekwencja w obwodach.* Utworzono także funkcję *policz głosy w okręgu* która uzupełnia tabelę kandydat w okręgu sumą głosów ze wszystkich obwodów dla każdego kandydata z okręgu. Na końcu stworzono trigger wraz z funkcją który wypełnia tabelę *liczenie głosów kandydaci*, dodając do niej *idkandydata*, liczbę głosów przed zmianą, liczbę głosów po zmianie i czas zmiany. Działa w momencie gdy ktoś zmieni niezerową liczbę głosów w obwodzie.

```
CREATE OR REPLACE FUNCTION policz_glosy_w_okregu(idokreg integer)
  RETURNS void AS
  $BODY$
      BEGIN
        DELETE FROM kandydat_w_okregu where idokreg_wyborczy = idokreg;
        INSERT INTO kandydat_w_okregu (idKandydat, idokreg_wyborczy, liczba_glosow)
        select k.idkandydat as idKandydat, idokreg as idokreg_wyborczy,
        sum(liczba_glosow) as liczba_glosow from kandydat_w_obwodzie kwo
        inner join kandydat k on k.idkandydat=kwo.idkandydat
        inner join obwod_wyborczy ob on ob.idobwod_wyborczy=kwo.idobwod_wyborczy
        where ob.idokreg_wyborczy=idokreg group by ob.idokreg_wyborczy, k.idkandydat;
      END;
  $BODY$
  LANGUAGE 'plpgsql';

CREATE OR REPLACE VIEW protokol_w_obwodzie AS
SELECT kom.idkomitet, k.idkandydat, kom.nazwa_komitetu, k.imie, k.nazwisko,
kwo.liczba_glosow, kwo.idobwod_wyborczy, o.idokreg_wyborczy
```

```
FROM kandydat_w_obwodzie kwo
INNER JOIN kandydat k on kwo.idkandydat=k.idkandydat
INNER JOIN lista l on k.idkandydat=l.idkandydat
INNER JOIN komitet kom on kom.idkomitet=l.idkomitet
INNER JOIN obwod_wyborczy o on kwo.idobwod_wyborczy=o.idobwod_wyborczy
ORDER BY kom.idkomitet;

CREATE OR REPLACE VIEW okw AS
SELECT obw.idokreg_wyborczy, obw.idobwod_wyborczy, obw.lokalizacja, czk.idczlonek_komisji,
        czk.imie, czk.nazwisko FROM czlonek_komisji czk
        inner join  komisja k on k.idczlonek_komisji=czk.idczlonek_komisji
        inner join  obwod_wyborczy obw on obw.idobwod_wyborczy=k.idobwod_wyborczy


CREATE OR REPLACE VIEW frekwencja_w_obwodach AS
select idobwod_wyborczy, sum(liczba_glosow), idokreg_wyborczy from protokol_w_obwodzie
where idobwod_wyborczy in
(select idobwod_wyborczy from obwod_wyborczy)
group by idobwod_wyborczy, idokreg_wyborczy;


CREATE OR REPLACE VIEW protokol_w_okregu AS
SELECT kom.idkomitet, k.idkandydat, kom.nazwa_komitetu, k.imie, k.nazwisko,
kwo.liczba_glosow, kwo.idokreg_wyborczy FROM kandydat_w_okregu kwo
INNER JOIN kandydat k on kwo.idkandydat=k.idkandydat
INNER JOIN lista l on k.idkandydat=l.idkandydat
INNER JOIN komitet kom on kom.idkomitet=l.idkomitet
ORDER BY kom.idkomitet;


CREATE OR REPLACE FUNCTION policz_glosy_w_okregu(idokreg integer)
  RETURNS void AS
  $BODY$
      BEGIN
        DELETE FROM kandydat_w_okregu where idokreg_wyborczy = idokreg;
        INSERT INTO kandydat_w_okregu (idKandydat, idokreg_wyborczy, liczba_glosow)
        select k.idkandydat as idKandydat, idokreg as idokreg_wyborczy,
         sum(liczba_glosow) as liczba_glosow from kandydat_w_obwodzie kwo
        inner join kandydat k on k.idkandydat=kwo.idkandydat
        inner join obwod_wyborczy ob on ob.idobwod_wyborczy=kwo.idobwod_wyborczy
        where ob.idokreg_wyborczy=idokreg group by ob.idokreg_wyborczy, k.idkandydat;
      END;
  $BODY$
  LANGUAGE 'plpgsql';

CREATE OR REPLACE FUNCTION kontrola() RETURNS TRIGGER AS $kontrola$
    BEGIN

        INSERT INTO liczenie_glosow_kandydaci (idkandydat,
        liczba_glosow_przed_zmiana,
        liczba_glosow_po_zmianie, kiedy_zmienil)
        Values (OLD.idkandydat, OLD.liczba_glosow,
        NEW.liczba_glosow, (select * from now()) );
```

```
        RETURN NEW;
    END;
$kontrola$ LANGUAGE plpgsql;

CREATE TRIGGER kontrola
BEFORE  UPDATE  ON kandydat_w_obwodzie
FOR EACH ROW
    WHEN (OLD.liczba_glosow IS DISTINCT FROM NEW.liczba_glosow
    AND OLD.liczba_glosow IS NOT NULL)
    EXECUTE PROCEDURE kontrola();
```

4. Projekt funkcjonalny

4.1. Interfejsy do prezentacji, edycji i obsługi danych

Z racji planowanego stopniowania dostępu zaimplementowano interfejsy dla PKW, komisarza i członka komisji. W interfejsie dla PKW istnieje możliwość wyświetlania, dodawania i edycji okręgów, dodawania i edycji list wyborczych, wyświetlania listy wszystkich kandydatów, oraz interaktywne poruszanie się po protokołach okręgów. W interfejsie komisarza można dodać i przeglądać obwody (widok okw), dodawać, edytować i przeglądać kandydatów w danym okręgu oraz wygenerować protokół z okręgu. Protokół ten zawiera: dane okręgu, liczbę głosów na listy, liczbę głosów na kandydatów, dane o frekwencji. Wygląda w sposób zbliżony do widoków ze stron PKW. Interfejs członka komisji umożliwia dodawanie głosów kandydatom oraz generację protokołu obwodowej komisji wyborczej. Protokół zawiera dane jak dla okręgu tylko, że dla obwodu. Na stronie głównej prezentowane są dane pomocne przy logowaniu, takie jak okręgi czy członkowie komisji.

4.2. Wizualizacja danych

Zdecydowano się na tabele ze względu na ich porzadkujący charakter oraz liczbę wyświetlanych danych. Ponadto dane wyświetlane są jako różne detale. Raporty występują jako protokoły i mają postać: liczba głosów według list, liczba głosów na kandydatów, frekwencja. Starano się osiągnąć podobieństwo do serwisów PKW.

4.3. Panel sterowania aplikacji

Strona główna aplikacji to nieukończony ekran logowania. Można zalogować się jako PKW, komisarz okręgu lub członek OKW. Następnie przechodzi się do kolejnych podmenu, właściwych dla wybranej funkcji. Aby "przelogować się" należy powrócić na stronę główną, tak samo jeśli użytkownik zechce przejść między interfejsami. Dzięki wykorzystaniu usługi *ElephantSQL* właściciel bazy danych dysponuje możliwością jej zdalnej edycji, np. korzystając z przeglądarki zapytań udostępnianej przez serwis albo przez API. Drugie podejście oferuje wiele możliwości, takich jak: uzupełnienie kandydatów za pomocą skryptu itp.

4.4. Makropolecenia

Funkcja licząca głosy w okręgu jest wywoływana za każdym wyświetleniem raportu z okręgu. Trigger *kontrola* uzupełnia tabelę *liczenie głosów kandydaci*.

5. Dokumentacja

5.1. Wprowadzanie danych

Dane można wprowadzić i edytować przez formularze. Przy wykorzystaniu klucza API i *psycopg2* napisano skrypty w języku Python które same generują zadaną liczbę kandydatów [1] oraz uzupełniają liczbę głosów losowymi wartościami, w zakresie od 0 do 200. Pakiet *psycopg2* zapewnia oczyszczanie danych wejściowych z wszelkich niepożądanych znaków (powstrzymanie przed atakami SQLInjection). Formularze zawierają ukryte pole przez co aplikacja jest chroniona przed atakami *CSRF*. Klucz do bazy danych jest zmienną systemową, przez co nie jest ujawniony w kodzie w repozytoriach.

5.2. Dokumentacja użytkownika

Strona serwisu: `https://kalkulatorv2.herokuapp.com/`
Aby zalogować się jako PKW należy kliknąć w hiperłącze. Aby zalogować się jako komisarz okręgu należy wpisać we formularzu numer okręgu. Aby zalogować się jako członek komisji należy kliknąć odpowiednie ID członka komisji. Interfejs PKW umożliwia: dodanie i edycję okręgów i komitetów, przeglądanie wszystkich kandydatów, przeglądanie wyników wyborów, przeglądanie informacji o nadużyciach. Interfejs komisarza umożliwia dodanie i edycję okręgów, członków komisji i kandydatów. Kolejność ma znaczenie! Można także wyświetlić prototyp karty do głosowania: wystarczy zobaczyć kandydatów. Można także wygenerować protokół i przeglądać informacje o wynikach w obwodach przy pomocy odnośników w protokole. Interfejs członka komisji umożliwia dodanie głosów kandydatom z danego obwodu, wielokrotne, w celu biezącego poprawiania pomyłek. O każdej zmianie informowane jest PKW w specjalnej tabeli! Można także wygenerować protokół OKW.

5.3. Dokumentacja techniczna

W załączniku, kod do wglądu na stronie `https://github.com/dodek08/kalkulator_v2`

5.4. Wykaz literatury

- `http://newton.fis.agh.edu.pl/~antek/index.php?sub=dbase`
- `https://www.elephantsql.com/docs/python.html`
- `http://flask.pocoo.org/docs/0.12/`
- `https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world`
- `https://www.postgresql.org/docs/`
- `https://devcenter.heroku.com/articles/python-gunicorn`
- `http://fakty.interia.pl/raporty/raport-wybory-samorzadowe-2018/aktualnosci/news-szefowa-k nId,2523031`
- `http://samorzad2014.pkw.gov.pl/321_protokol_komisji_obwodowej/22942/rdw_3`
- `http://samorzad2014.pkw.gov.pl/357_rady_woj/0/2604`
- `http://egov.pl/kim/wp-content/uploads/cw4_lato.pdf`

[1] W generatorze wykorzystano 20 najpopularniejszych imion męskich i żeńskich, ulubione imiona autora, imiona postaci z serialu *Korona Królów* oraz kilkadziesiąt "nieodmiennych" nazwisk (wszystkie nazwiska w języku polskim są odmienne) m.in. ministrów z III RP, nazwisk sportowców oraz znajomych autora