DOKUMENTACJA PROJEKTU
„System Zarządzania Partią Polityczną”

Bartosz Sobocki

SPIS TREŚCI

PRZEZNACZENIE API

- API zostało stworzone z myślą o partii potrzebującej systemu zarządzania nią, do prowadzenia rejestru działań rządowych i samorządowych, które wspiera lub przeciw którym protestuje.
- Partią zarządza Zespół Liderów będących jej członkami.
- Członkowie mogą proponować akcje oraz głosować za i przeciw ich prowadzeniu.
- Członkiem partii zostaje się poprzez aktywny udział w jej życiu. Po roku nieaktywności, rozumianej jako brak wywołań funkcji API autoryzowanych danymi danej osoby, traci się prawa członka partii, a takie konto jest trwale zamrażane (tzn. dany członek nie może wykonać żadnej czynności, ale wszystkie informacje na jego temat są dalej przechowywane i raportowane). Zasada ta dotyczy wszystkich członków partii, również liderów.

URUCHOMIENIE

Niezbędne komponenty

Do uruchomienia programu niezbędny jest PostgreSQL, python3.x oraz biblioteka psycopg2 niezbędna do połączenia się z Bazą Danych.

Uruchomienie w Systemie Linux

Uruchomienie programu następuje poprzez uruchomienie interpretera python poleceniem ‘python3’ wraz parametrem ‘app.py’:

~$ python3 app.py

Wraz z uwzględnieniem przekierowania informacji z pliku, z którego aplikacja ma odczytać kolejne wywołania funkcji, np.:

~$ python3 app.py < example.txt
(API pobierze wywołania funkcji z pliku example.txt i wykona je)

Możliwe jest również wywoływanie na bieżąco poszczególnych funkcji API:

```
~$ python3 app.py
{"open":{"database":"student", "login":"app", "password":"qwerty"}}
{"status":"OK"}
```

Pierwsze uruchomienie powinno dodatkowo zawierać parametr ***-- init*** np.:

~$ python3 app.py --init < open_example.txt

Zawiera w pierwszym wierszu wywołanie funkcji open z następującymi danymi login: init, password: qwerty, w kolejnych wierszach wywołania funkcji leader.

Każde kolejne uruchomienie programu powinno zaczynać się od wywołania funkcji ‘’open’’ z loginem ‘’app’’ oraz hasłem ‘’qwerty’’, natomiast kolejne funkcje wywołane nie mogą być funkcją *leader* oraz *open*.

SPECYFIKACJA WYWOŁAŃ FUNKCJI API

Wywołania funkcji API

Wywołania funkcji API powinny być obiektami JSON oraz zawierać niezbędne elementy do uruchomienia funkcji (np. funkcja ‘open’ powinna zawierać nazwę bazy danych, login użytkownika i hasło).

Wszystkie funkcje wraz z parametrami:

- **open :**
  - **‘’database‘’ (string)**
  - **‘’login’’ (string)**
  - **‘’password’’ (string)**
- **leader:**
  - **‘’timestamp’’ (long integer)**
  - **‘’password‘’ (string)**
  - **‘’member’’ (integer)**
- **support:**
  - **‘’timestamp’’ (long integer)**
  - **‘’member’’ (integer)**
  - **‘’password’’ (string)**
  - **‘’action’’ (integer)**
  - **‘’project’’ (integer)**
  - **[‘’authority’’ (integer)]**
- **protest:**
  - **‘’timestamp’’ (long integer)**
  - **‘’member’’ (integer)**
  - **‘’password’’ (string)**
  - **‘’action’’ (integer)**
  - **‘’project’’ (integer)**
  - **[‘’authority’’ (integer)]**
- **upvote:**
  - **‘’timestamp’’ (long integer)**
  - **‘’member’’ (integer)**
  - **‘’password’’ (integer)**
  - **‘’action’’ (integer)**
- **downvote:**
  - **‘’timestamp’’ (long integer)**
  - **‘’member’’ (integer)**
  - **‘’password’’ (integer)**
  - **‘’action’’ (integer)**
- **actions:**
  - **‘’timestamp’’ (long integer)**
  - **‘’member’’ (integer)**
  - **‘’password’’ (string)**
  - **[‘’type’’ -> ‘’support’’/’’protest’’]**
  - **[‘’project’’/’’authority’’]**
- **projects:**
  - **‘’timestamp’’ (long integer)**
  - **‘’member’’ (integer)**
  - **‘’password’’ (string)**
  - **[‘’authority’’ (integer)]**
- **votes:**
  - **‘’timestamp’’ (string)**
  - **‘’member’’ (integer)**
  - **‘’password’’ (string)**
  - **[‘’action’’/’’project’’ (integer)]**
- **trolls:**
  - **‘’timestamp’’ (integer)**

{ "•" : { ∘ : value, ∘ : value,... }}

Wygląd wywołania funkcji API

timestamp -> UNIX time
member, action, project, authority -> id

Pliki z wywołaniami kolejnych uruchomień programu (bez parametru --init)

Każdy plik, z którego przekierowane będą dane wywołania powinien zawierać jako pierwszy obiekt JSON wywołanie funkcji **'open'** np.:

{ "open": { "database": "student", "login": "app", "password": "qwerty"}}

a następnie wywołania dowolnych funkcji API za wyjątkiem funkcji open i leader.

Działanie poszczególnych funkcji API

Każda z funkcji w której podany jest <member> i <password> sprawdza, czy podany członek partii istnieje.

Jeśli istnieje, to program sprawdza hasło i gdy jest ono poprawne (oraz dla funkcji wypisujących: ***votes***, ***projects***, ***actions*** sprawdzi czy członek jest *liderem*) wykona funkcje. W przeciwnym przypadku zwróci błąd.

Jeśli jednak użytkownik nie istnieje w bazie to dodaje go jako zwykłego użytkownika (poza funkcjami wypisującymi).

-- init - inicjalizuje bazę danych, tworzy potrzebne: tabele, więzy, indeksy, klucze, akcje referencyjne, funkcje, perspektywy i wyzwalacze, a także użytkownika app z odpowiednimi uprawnieniami.

open - otwiera bazę danych i loguje użytkownika bazy

leader - nadaje funkcje lidera członkowi o podanym id

suport - dodaje akcje <support> przeciwko podanemu projektowi (do głosowania) do bazy danych

protest - dodaje akcje <protest> przeciwko podanemu projektowi (do głosowania) do bazy danych

upvote - dodaje głos *'za'* oddany przez <member> na <action> do bazy danych

downvote - dodaje głos *'przeciw'* oddany przez <member> na <action> do bazy danych

actions - zwraca listę wszystkich akcji wraz z typem, id projektu, id organu władzy oraz z liczbami głosów za i przeciw akcji z następującymi zastrzeżeniami:
- jeśli podano <type> w postaci tekstu <support> albo <protest> ograniczy się do akcji podanego typu
- jeśli podano <project> ograniczy się do akcji dotyczących danego <project>
- jeśli podano <authority> ograniczy się do akcji dotyczących działań danego <authority>

<password> to hasło członka <member> będącego liderem.

projects - zwraca listę wszystkich działań wraz z id organu władzy prowadzącego dane działanie.
- jeśli <authority> jest podane to zwraca wyłącznie działania podanego <authority>

<password> to hasło członka <member> będącego liderem.

votes - zwraca listę wszystkich członków (w tym tych, którzy do tej pory nie głosowali) wraz z sumarycznymi liczbami oddanych przez nich głosów za (<upvotes>) i przeciw (downvotes)
- jeśli podano <action> to wynik powinien ograniczyć się do głosów dotyczących akcji <action>
- jeśli podano <project> to wynik powinien ograniczyć się do głosów dotyczących akcji związanych z projektem <project>

<password> to hasło członka <member> będącego liderem

trolls - zwraca listę wszystkich użytkowników, którzy zaproponowali akcje mające sumarycznie więcej downvotes niż upvotes (troll'i)

pusty wiersz- kończy działanie programu

MODEL KONCEPTUALNY