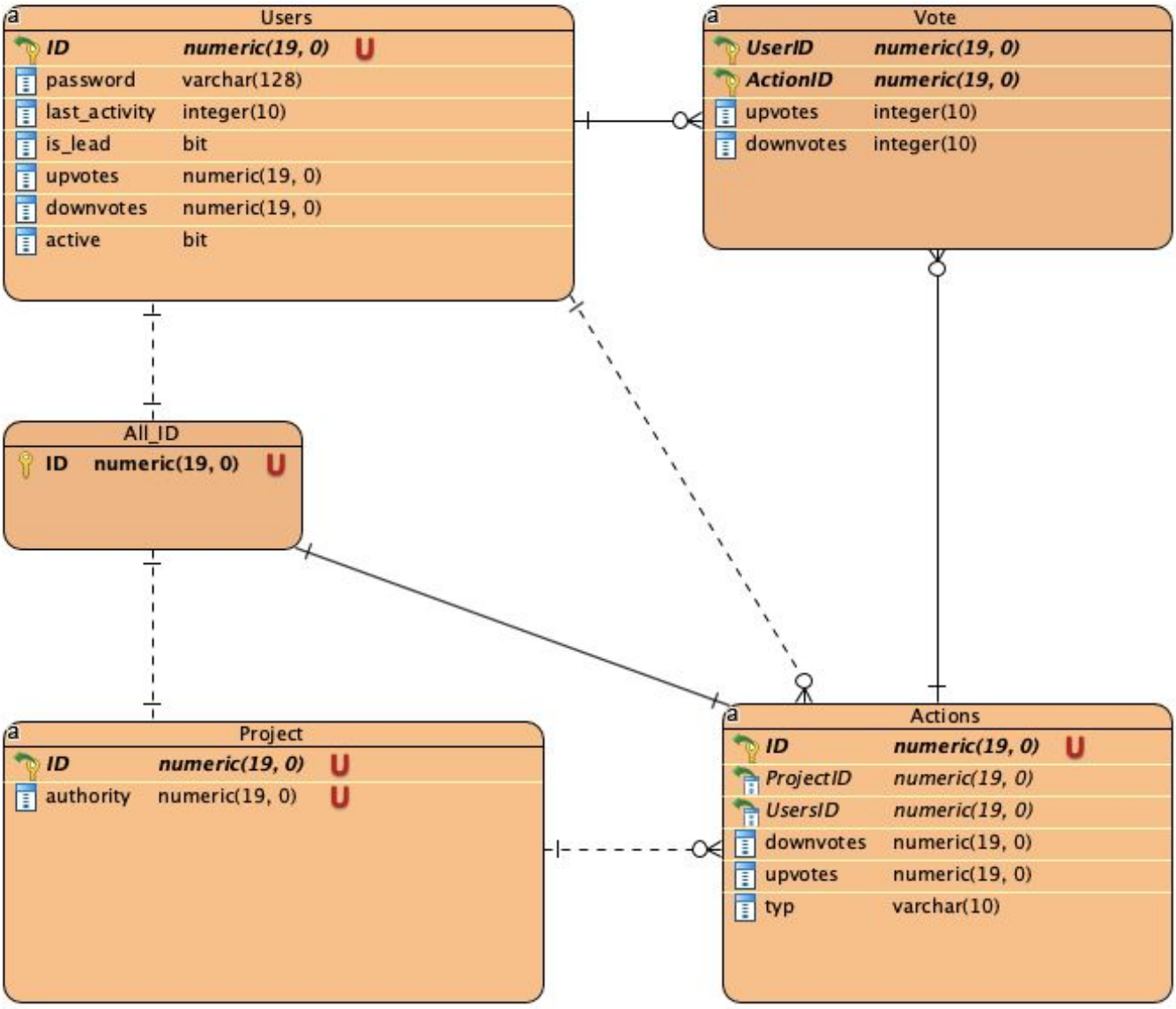
DOKUMENTACJA - BAZY DANYCH 2019

Izabela Strumecka



Uprawnienia init i app

Init - pierwsze uruchomienie, tworzenie leaderów, funkcja *open*

App - kolejne uruchomienia, wszystkie funkcje API oprócz *leader*

Baza - tabele:

1. ALL_ID - zawiera wszystkie ID, zapewnia globalną unikalność wszystkich ID w bazie.
2. Users - zawiera informacje o członkach partii - id, hasło, datę ostatniej aktywności, liczbę oddanych upvotes i liczbę oddanych downvotes, informację czy jest leaderem oraz pole active wykorzystywane przy wywołaniu funkcji trolls.
3. Project - zawiera id projektu oraz id organu władzy organizującego dany projekt.
4. Actions - zawiera informacje o zgłoszonych przez użytkowników akcjach - id akcji, id projektu, do którego należy akcja, id użytkownika, który zgłosił akcję, liczbę upvotes i liczbę downvotes na daną akcję oraz typ akcji.
5. Vote - zawiera głosy oddane przez konkretnego użytkownika na konkretną akcję - id akcji, na którą zagłosował użytkownik, id tego użytkownika, upvotes/downvotes ustawione na 0 lub 1 w zależności jaki głos oddał członek (zapewnienie, że dana osoba odda tylko jeden głos na daną akcję).

Idea

Open - funkcja z poziomu języka programowania, łączy z bazą danych.

Leader - dodaję do tabeli User użytkownika z wartością 1 w kolumnie is_leader.

Support i *protest* - w tabeli User sprawdzam, czy użytkownik istnieje, jeżeli nie - dodaję go, jeżeli tak - sprawdzam czas ostatniej aktywności. Jeżeli użytkownik w ciągu roku był aktywny, sprawdzam w tabeli Project, czy projekt dodawanej akcji już istnieje. Jeżeli nie - dodaję go z odpowiednim authority. Do tabeli akcji dodaję akcję. Dbam o to, aby ID akcji i użytkownika były unikalne globalnie, wykorzystując tabelę All_ID.

Upvote i *downvote* - jeżeli członek o podanym id nie istnieje - dodaję go do tabeli User a jego ID do tabeli All_ID (unikalne globalnie!). Sprawdzam ostatnią jego aktywność. Jeżeli wszystko idzie pomyślnie, sprawdzam w tabeli Vote, czy użytkownik nie głosował już na daną akcję. Jeżeli nie inkrementuję upvotes/downvotes w tabeli Action akcji podanej w argumencie , jednocześnie inkrementując upvotes/downvotes w tabeli Vote przy głosującym użytkowniku na daną akcję oraz inkrementuję upvotes/downvotes w tabli User przy odpowiednim użytkowniku.

Actions - funkcja odczytuje dane z bazy zwracając listę akcji wraz z typem, id projektu, id organu władzy oraz liczbą głosów za i przeciw. Wypisanie krotek tabeli Action złączonej z tabelą Project. Odpowiednie ograniczenia już na tym złączeniu. Wywoływanie funkcji tylko przez leadera.

Projects - funkcja odczytuje dane z bazy zwracając listę wszystkich członków wraz z id organu władzy. Wypisanie krotek tabeli Project. Odpowiednie ograniczenie już na tej tabeli. Wywoływanie funkcji tylko przez leadera.

Votes - funkcja odczytuje dane z bazy zwracając listę wszystkich członków wraz z sumarycznymi liczbami oddanych przez nich głosów za i przeciw. Funkcja działa na tabeli User poprzez odpowiedni SELECT. Ograniczając się do jednej akcji informację możemy wziąć z tabeli Vote (użytkownik może oddać tylko jeden głos na akcję). Ograniczenie do konkretnego projektu wymaga korzystania z tabeli Action, z której wybieramy akcje przypisane do danego projektu, oraz tabeli Vote. Sumujemy głosy wszystkie głosy za danego użytkownika po wszystkich wybranych akcjach oraz wszystkie głosy przeciw. Wywoływanie funkcji tylko przez leadera.

Trolls - zwraca listę wszystkich użytkowników, którzy zaproponowali akcje mające w chwili sumarycznie więcej downvotes niż upvotes. Operujemy na tabeli User. i Actions. Wywoływanie funkcji tylko przez leadera.

Implementacja

Projekt zawiera dwa pliki:

1. baza.sql - model fizyczny bazy, plik uruchamiany podczas wywołania funkcji open przy uruchomieniu z parametrem --init
2. main.py - aplikacja zarządzająca bazą - łącznie się z bazą, pobieranie danych wejściowych, wszystkie funkcje API;
    a. main() - odpowiada za wywołanie kolejnych funkcji w zależności od parametru (--init, --reset, brak parametru)

b. reset() - czyści bazę danych
c. readjson() - modyfikuje plik JSON na listę słowników
d. start_init() - odpowiada za obsługę danych wejściowych wywołanych z parametrem --init
e. start_app() - odpowiada za obsługę danych wejściowych wywołanych bez parametru
f. statusOK() - tworzy status OK
g. statusER() - tworzy status ERROR
h. status() - tworzy wynikowy obiekt JSON
i. not_exist_id() - sprawdza czy w konkretnej tabeli znajduje się konkretne id
j. not_lead() - sprawdza czy członek jest leaderem
k. validate_password_and_active() - sprawdza, czy podano dobre hasło i czy członek jest aktywny
l. open_init() - tworzy połączenie z bazą danych przy uruchomieniu z parametrem --init
m. open_app() - tworzy połączenie z bazą danych przy uruchomieniu bez parametru
n. leader() - tworzy lidera i dodaje go do bazy
o. support() - tworzy akcję poparcia
p. protest() - tworzy akcję protestu
q. upvote() - tworzy głos “za” daną akcją
r. downvote() - tworzy głos “przeciw” danej akcji
s. actions() - wyświetla wszystkie akcje wraz z informacjami o nich - typ, projekt, organ organizujący, głosy za, głosy przeciw
t. projects() - wyświetla wszystkie projekty wraz z organem organizującym to przedsięwzięcie
u. votes() - zwraca listę wszystkich członków wraz z sumarycznymi liczbami oddanych przez nich głosów za i przeciw
v. trolls() - zwraca listę trolli

Uruchomienie aplikacji

Każde uruchomienie wymaga dodania nazwy pliku JSON.

Uruchomienie z parametrem --reset resetuje bazę danych.

python main.py --reset test.json

Uruchomienie z parametrem --init - JSON zawiera w pierwszym wierszu wywołanie funkcji open z następującymi danymi login: init, password: qwerty, w kolejnych wierszach wywołania funkcji leader.

python main.py --init testinit.json

Uruchomienie bez parametrem - JSON zawiera w pierwszym wierszu wywołanie funkcji open z następującymi danymi login: app, password: qwerty, a następnie wywołania dowolnych funkcji API za wyjątkiem funkcji open i leader

python main.py testapp.json