Generator ruchu Google Analytics
Iteracja II architektura systemu

Bartłomiej Dalak Bartłomiej Karwowski
Bartosz Gromek Tomasz Kanas

23 maja 2018

Wstęp

Dokument architektury systemu ma na celu przedstawienie wizji architektury. Opisana architektura może ulec zmianom w fazie implementacji.

Aplikacja

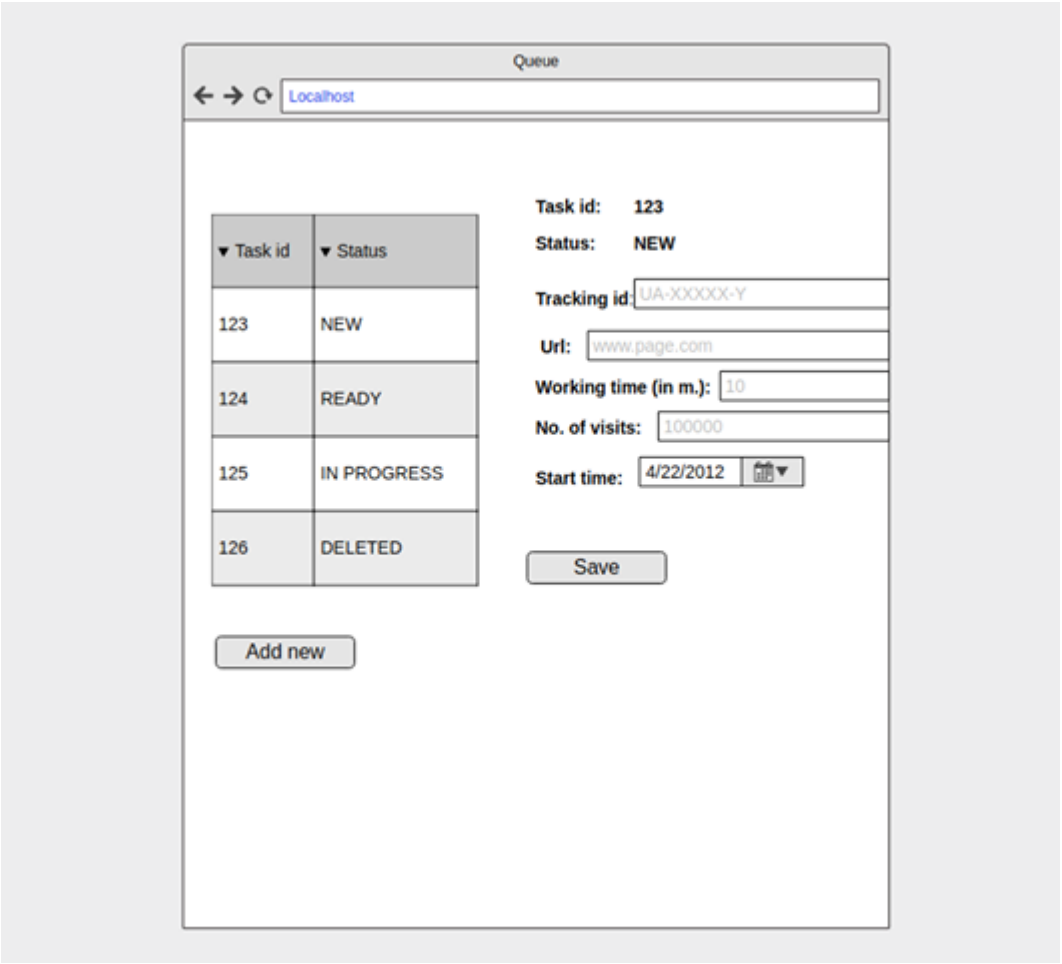
Aplikacją będzie system kolejkowania wysyłania wejść na podaną stronę użytkownika do Google Analytics.

Opis elementów architektury

UI

Użytkownik po wejściu na stronę zobaczy po lewej stronie listę wszystkich zadań dodanych do systemu, podzieloną na 2 kolumny. W pierwszej będzie id zadania, w drugiej stan w jakim się znajduje. W dalszej częsci dokumentu będzie opisany każdy stan. Pod listą znajduje się przycisk **Add new** pozwalający dodać nowe zadanie ze statusem **NEW**. Automatycznie po prawej stronie pokaże się formularz gotowy do uzupełnienia. Po uzupełnieniu go, będzie mógł go zapisać naciskając przycisk **Save**. Zmieni się wtedy stan na **READY**. Dodatkowo użytkownik naciskając na wiersz listy, będzie mógł zobaczyć szczegóły wybranego zadania. Jeśli będzie to zadanie w stanie **READY**, będzie można edytować dane. Dodatkowo pojawi się przycisk **Delete** obok przycisku **Save** pozwalający usunąć zadanie do wykonania. Jeśli będzie to zadanie w stanie **IN PROGRESS**, to pojawi się przycisk **Cancel** pozwalający przerwać wykonywanie zadania.

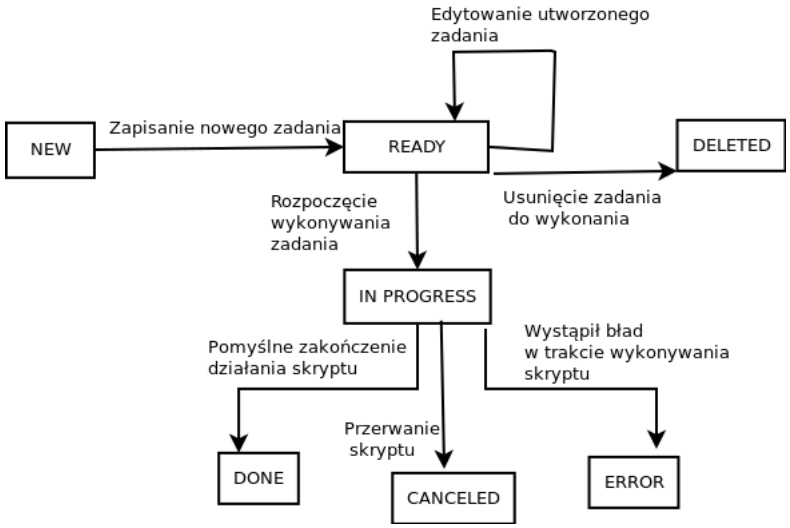
Przykładowy wygląd



Stany zadań

- **NEW**: zadanie dodane do bazy
- **READY**: zadanie zapisane, czekające na wykonanie
- **IN PROGRESS**: zadanie w trakcie wykonywania
- **CANCELED**: zadanie zatrzymane
- **DELETED**: zadanie usunięte, zanim zaczęło się wykonywać
- **DONE**: zadanie zostało wykonane
- **ERROR**: wystąpiły problemy w trakcie wysyłania

Przejścia stanów



Niedozwolone przejścia między stanami

Może się zdarzyć, że użytkownik spróbuje przejść między stanami, dla których nie ma połączenia. Zostanie wtedy on poinformowany, że danej akcji nie można wykonać. Niedozwolone przejścia:

- **IN PROGRESS**, a **READY**: może się to zdarzyć, gdy użytkownik utworzy zadanie, zacznie ono się wykonywać, a on spróbuje je edytować, ponieważ nie odświeżył widoku. Wtedy zostanie on poinformowany, że nie mozna edytować trwającego zadania
- **IN PROGRESS**, a **DELETED**: analogiczna sytuacja jak wyżej tylko tym razem użytkownik próbuje usunąć zadanie

Baza danych

Jako systemem do zarządzania bazą użyjemy SQLite. Baza będzie zawierała dwie tabele:

- **state**, która będzię trzymała stany w jakich może znajdować się zadanie. Będzie sie składała z 2 kolumn:
  - **id: INT**: id statusu
  - **name: TEXT**: nazwa statusu
- **tasks**, która będzie trzymała dodane zadania. Kolumny z jakich będzie sie składać:
  - **task_id: INT, PK**: id zadania

- **tracking_id: TEXT**: tracking id użytkownika
- **url: TEXT**: url strony na jaką chcemy dodawać użytkowników
- **time: INT**: czas przez jaki ma działać skrypt
- **visits: INT**: liczba użytkowników do wygenerowania
- **start_time: DATE**: data kiedy ma się wykonać skrypt
- **state: FK do state.id**: klucz obcy do tabeli stanów oznaczający stan w jakim aktualnie znajduje się zadanie

Backend aplikacji

Słuzy do komunkacji między **UI**, a **bazą**. Użyjemy do tego frameworka **Flask**. Będziemy używać widoków:

- widok pod adresem "/tasks" z metodą **GET** wyświetlający listę wszystkich zadań
- widok pod adresem "/tasks" z metodą **POST**, który będzie dodawał nowe zadanie do bazy, odświeżał listę wyświetlonych zadań i wyświetlał formularz
- widok pod adresem "/tasks/save" z metodą **POST**, który będzie aktualizował wybrane zadanie i dodatkowo walidował czy został podany poprawny **tracking_id**

System kolejkowania

Język

Python 3.6.

Użyte biblioteki

- **multiprocessing**: Tworzenie nowych procesów wysyłających dane do GA, oraz zarządzanie nimi, w szczególności udostępnia metodę przerywającą dany proces (wysyła SIGTERM).

Opis działania

Skrypt, który co 1 minutę będzie wysyłał zapytania do bazy pobierając dane. Skrypt będzie sprawdzał, czy należy uruchomić kolejny proces (czy rozpoczęło się właśnie nowe zadanie), czy jakiś działający proces należy przerwać (użytkownik go anulował), oraz monitorował działanie wszystkich procesów i aktualizował w bazie danych ich stan w przypadku jego zmiany — zakończenia działania (sukces lub błąd). Przerywanie będzie wykonywane poprzez wysłanie do procesu sygnału SIGTERM (proces będzie obsługiwał ten sygnał aby nie dopuścić do wycieku zasobów).

Skrypt do wysyłania zapytań do GA

Język

Wykorzystany zostanie Python w wersji 3.6.

Użyte biblioteki

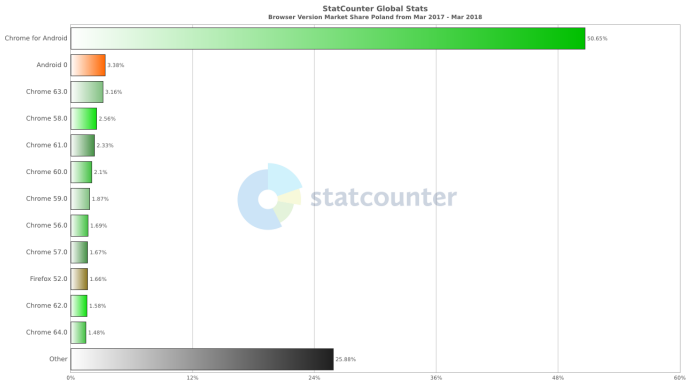
- **requests**: wysyłanie zapytań do GA i Measurement Protocol Validation Server
- **csv**: do obsługi pliku 'browser.csv', w którym mamy rozkład przeglądarek na terenie Polski.

send_requests_api

API służące do komunkacji z GA, generuje potrzebne dane oraz je wysyła. Dostępne metody:

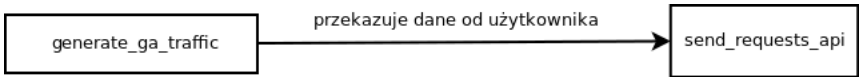
- **send (tracking_id, url, visits_no, time)**: Przekazuje dane do niżej opisanej funkcji **generate_data**. Po odebraniu wygenerowanych danych, próbuje przesłać je bezpośrednio do GA. Przykładowe wysłanie danych: **requests.post ("https://www.google-analytics.com/ collect", data="v": 1, "t": "pageview", "tid": tracking_id, "cid": 1, "dp": url)**. W ten sposób będziemy wysyłać w pętli kolejne wejścia z wygenerowanych danych. Zwraca kod **OK**, po wygenerowaniu wszystkich danych.
- **generate_data (visits_no)**: metoda wołana przez **send ()**, generujące odpowiednie dane do wysłania. Po odebraniu informacji przekazanych przez użytkownika, do odpowiedniej ilości zapytań przypisuje dane przygotowane z wiarygodnym rozkładem. Informacje do tego potrzebne zostaną zczytane z pliku 'browser.csv', który zostanie pobrany ze strony GlobalStats StatCounter (dane dot. oprogramowania użytkowników witryny). Informacje te zostaną przypisane na zmienną **distribution_informations** będącą typu DataFrame. Zostanie to wykonane tylko raz.

Wykres użytkowania przeglądarek na terenie Polski

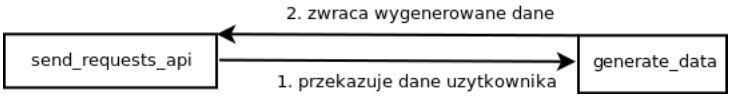


Schemat działania

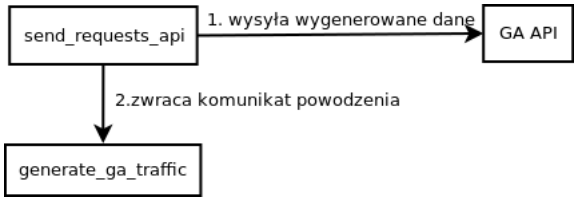
- **przekazanie danych**: Użytkownika wywołuję funkcję **send (tracking_id, url, visits_no, time)** udostępnioną przez **send_requests_api**. W rezultacie otrzymuje komunikat tego czy udało się pomyślnie wysłać żądanie.



- **wygenerowanie danych** Po otrzymaniu danych od użytkownika zostaje wywołana funkcja **generate_data**, która generuje i zwraca dane.



- **wysyłanie danych i zwrócenie komunikatu** Po otrzymaniu wygenerowanych danych zostają one wysyłane do Google Analytics, a następnie zostaje zwrócony komunikat powodzenia.



Komunikacja między elementami