

# Generator ruchu Google Analytics

## Iteracja I architektura systemu

Bartłomiej Dalak      Bartłomiej Karwowski  
Bartosz Gromek      Tomasz Kanas

11 kwietnia 2018

### Wstęp

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

### Opis elementów architektury

#### Aplikacja

Aplikacją w naszym przypadku będzie skrypt “generate\_ga\_traffic.py”, generujący ruch według podanych przez użytkownika wartości. Będzie on uruchamiany przez konsolę.

#### Język

Wykorzystany zostanie Python w wersji 3.6.

#### Użyte biblioteki

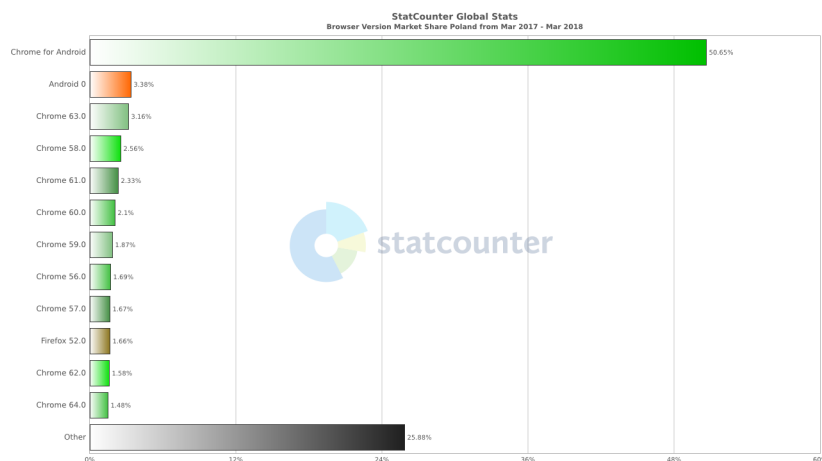
- **requests**: wysyłanie zapytań do GA i Measurement Protocol Validation Server
- **pandas**: do obsługi pliku ‘browser.csv’, w którym mamy rozkład przeglądarek na terenie Polski.

#### send\_requests\_api

API służące do komunikacji z GA, sprawdza dane od użytkownika, generuje potrzebne dane oraz je wysyła.

## Metody

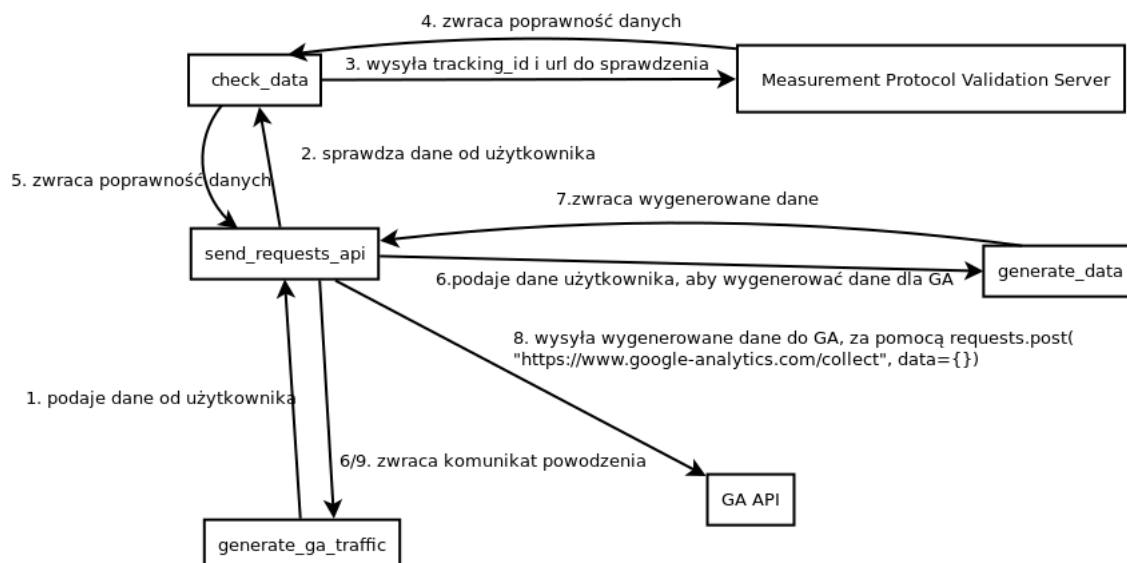
- **send(tracking\_id, url, visits\_no, time)**: sprawdza czy dane użytkownika mogą zostać wygenerowane, za pomocą funkcji **check\_data**. Jeśli dane przejdą weryfikację to przekazuje je do niżej opisanej funkcji **generate\_data**. Wpp. zwraca kod WRONG\_DATA. Po odebraniu wygenerowanych danych, próbuje 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 jeden z 2 możliwych kodów OK, CONNECTION PROBLEM, w zależności od tego czy udało się wysłać wszystkie dane, czy utracono połączenie.
- **check\_data(tracking\_id, url)**: metoda, która wyśle requesta do Measurement Protocol Validation Server za pomocą **requests.post("https://www.google-analytics.com/debug/collect", data='tid': tracking\_id, 'dp': url, 'v': 1)** i jeśli otrzymany response w formacie JSON w polu ["hitParsingResult"][0]["valid"] zawiera wartość true to zwróci tuple(true, []), wpp. zwróci tuple(false, parameters), gdzie parameters to lista błędnych parametrów, które otrzymamy z pól ["hitParsingResult"][0]["parserMessage"][i]["parameter"], gdzie i to indeks złego parametru.
- **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 czytane z pliku 'browser.csv', który zostanie pobrany ze strony GlobalStats StatCounter (dane dot. oprogramowania użytkowników witryny). Wykres użytkowania przeglądarek na terenie Polski.



## Schemat działania

### Generowanie i wysyłanie danych

Dane otrzymane przez użytkownika zostaną przekazane do napisanego przez nas API: **send\_requests\_api**. API sprawdzi czy otrzymane dane od użytkownika są poprawne, wygeneruje zestaw wejść, które będą wysyłane do GA, a następnie za pomocą metody post z biblioteki **requests** wyśle je.



### Wykrywanie braku połączenia z internetem

Jeśli **send\_requests\_api** przez 10 min nie uda się wysłać żadnego requesta do GA API, bądź nie uzyskamy odpowiedzi, w takim wypadku zostaje zatrzymane wysyłanie oraz zwrócony do **generate\_ga\_traffic** komunikat **CONNECTION\_PROBLEM**. Wtedy użytkownik zostaje o tym poinformowany i może wybrać jedną z trzech możliwości co chce robić dalej.