Politechnika Poznańska

Wydział Elektryczny

Projekt Semestralny

Agregator informacji dotyczących filmów z wykorzystaniem web scrapingu

MILAN SAWICKI

JAKUB ANIOŁA

1. Wstęp

1. 1. Opis projektu

Web scraping, screen scraping, mówiąc potocznie *skrobanie WWW*, metoda pozyskiwania informacji za pomocą symulacji przeglądania stron internetowych jaką wykonuje człowiek, analiza ich pod kątem składni i ekstrakcja danych za pomocą atrybutów CSS lub za pomocą odpowiednich zapytań w języku XPath.

Projekt polega na w/w scrapingu stron internetowych zawierających informacje na temat filmów, np *imdb.com*, przetworzenie ich oraz zwrócenie odpowiedzi za pomocą REST API. Umożliwiono również użytkownikowi eksport pozyskanych danych do bazy danych NoSQL MongoDB.

Rys. 1 Logo

1. 2. Uzasadnienie wyboru

Zdecydowaliśmy się podjąć ten temat, ponieważ zainteresowała alternatywna metoda pozyskiwania danych. Pozwala ona na masowe ich gromadzenie. Dodatkowym atutem jest możliwość i okazja poznania nowego języka programowania (Python), a także praktyczność zdobytej przy wykonywaniu projektu wiedzy, jak np. tworzenie REST API w Pythonie, tworzenie aplikacji mobilnej na system Android, zapoznanie się z atutami i wadami baz danych NoSQL.

2. Podział i harmonogram pracy

Nasz podział prac został stanowczo zachwiany z uwagi na wypadek losowy (konieczność odejścia jednej osoby z naszego zespołu). Niemniej jednak kolejność zadań została nie zmieniona, co zaprezentowano poniżej:

1. Zapoznanie się z zagadnieniem scrapingu
2. Wybór technologii oraz bibliotek
3. Zainicjowanie programu służącego do scrapingu
4. Nauka narzędzia *Scrapy* ozyskanie pierwszych danych
5. Konfiguracja strumienia (*ang. pipeline*) do eksportu pozyskanych danych do bazy danych
6. Zapoznanie się z podstawami tworzenia REST API w Pythonie
7. Stworzenie widoków i wyglądu aplikacji mobilnej
8. Stworzenie pierwszej wersji API
9. Obsługa API w aplikacji mobilnej
10. Dostosowanie programu do scrapingu tak, aby umożliwiał dynamiczne przekazywanie informacji do API (bez użycia bazy danych)
11. Testy oraz ulepszenia widoków
12. Oddanie projektu

Jest to zarys skrótowy, nie uwzględnia trudności napotkanych na drodze tworzenia projektu. Trudności zostały opisane w rozdziale 6.

Za pracę nad częścią *back-endową* (scraper i API) projektu jak i wszelkie zadania wchodzące w jej skład czuwał Milan Sawicki.
Za wizualną część i aplikację mobilną odpowiada Jakub Anioła.

3. Wykorzystane technologie oraz wiedza

3. 1. Trello

Trello to narzędzie służące do pracy w zespole osób. Daje możliwość tworzenia tablice a w nich umieszczania zadania oraz notatki. Ułatwia prowadzenie rejestru zadań wykonanych i takich które jeszcze muszą zostać wykonane. Jest to narzędzie darmowe i pozwala na wprowadzenie do projektu elementów metodyki Scrum.

3. 4. Git

System kontroli wersji. Umożliwia przechowywanie historii zmian i wersjonowanie kodu.

3. 5. Środowisko programistyczne PyCharm

Zintegrowane środowisko programistyczne (IDE) dla języka programowania Python firmy JetBrains.

3. 6. Środowisko programistyczne Android Studio

Zintegrowane środowisko programistyczne (IDE) do tworzenia aplikacji mobilnych na platformę Android firmy JetBrains.

3. 7. Framework Scrapy

Scrapy to open-sourcowe'owy framework służący do ekstrakcji danych ze stron internetowych napisany w języku Python.

3. 9. Framework Flask

Framework służący do tworzenia aplikacji REST w języku Python. Bazuje na bibliotece *Werkzeug*.

3. 8. MongoDB

Otwarty, nierelacyjny system zarządzania bazą danych napisany w języku C++. Charakteryzuje się dużą skalowalnością, wydajnością oraz brakiem ściśle zdefiniowanej struktury obsługiwanych baz danych. Zamiast tego dane składowane są jako dokumenty w konwencji JSON, co umożliwia aplikacjom bardziej naturalne ich przetwarzanie, przy zachowaniu możliwości tworzenia hierarchii oraz indeksowania.

3. 9. Język programowania Python oraz Java

4. Zastosowane standardy komunikacji

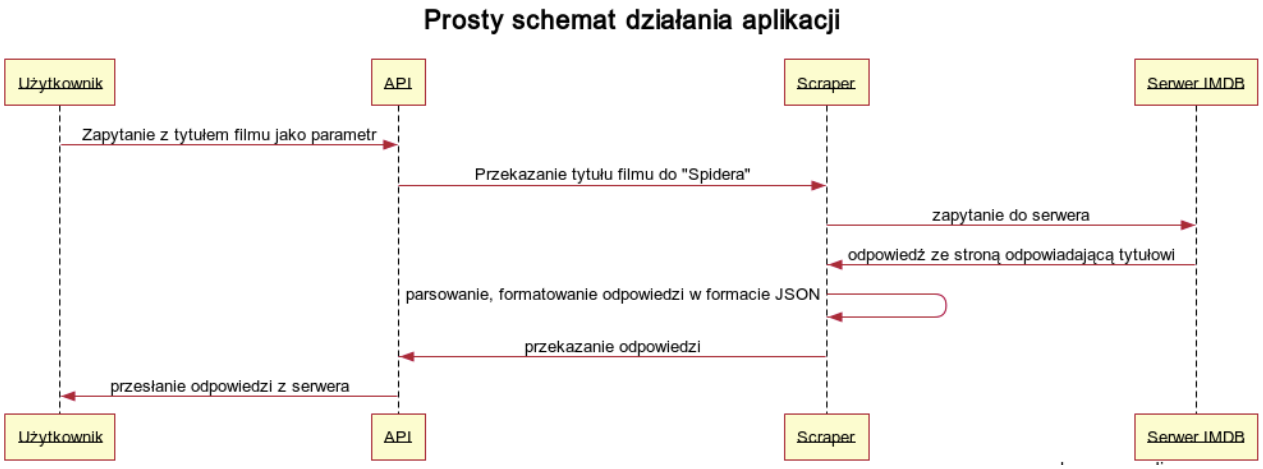
4. 1. REST

REST jest zbiorem reguł, których powinien przestrzegać programista. Jest to wzorzec architektury oprogramowania, który opisuje jak operować zapytaniami do API i wprowadza zestaw dobrych praktyk. REST ułatwia obsługę żądań i odpowiedzi bez konieczności odwoływania się do złożonych dokumentacji. W ciągu ostatnich kilku lat REST stał się wiodącym standardem architektury sieciowej.

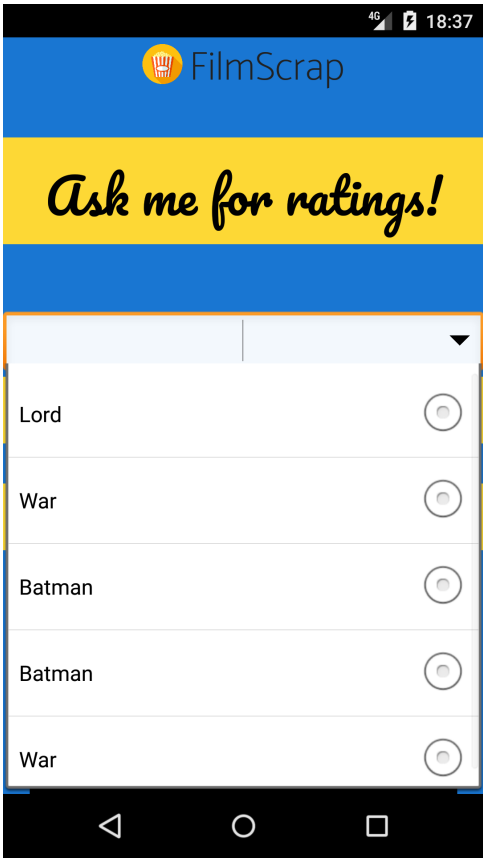
4.2 JSON

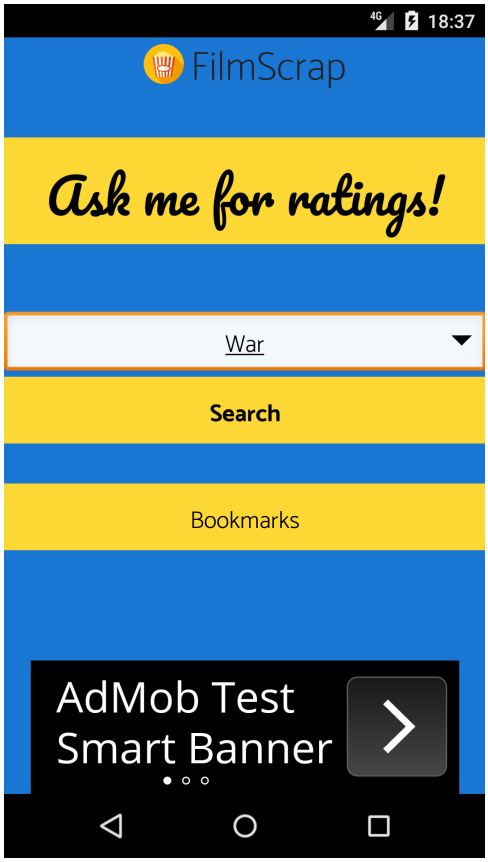
JavaScript Object Notation – lekki format wymiany danych komputerowych. JSON jest formatem tekstowym, bazującym na podzbiorze języka JavaScript. Zastosowany do komunikacji z API.

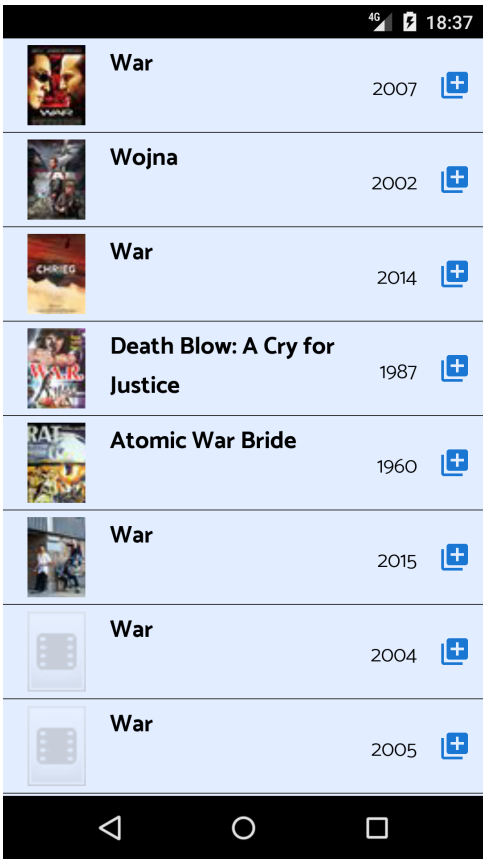
4.3 Schemat działania

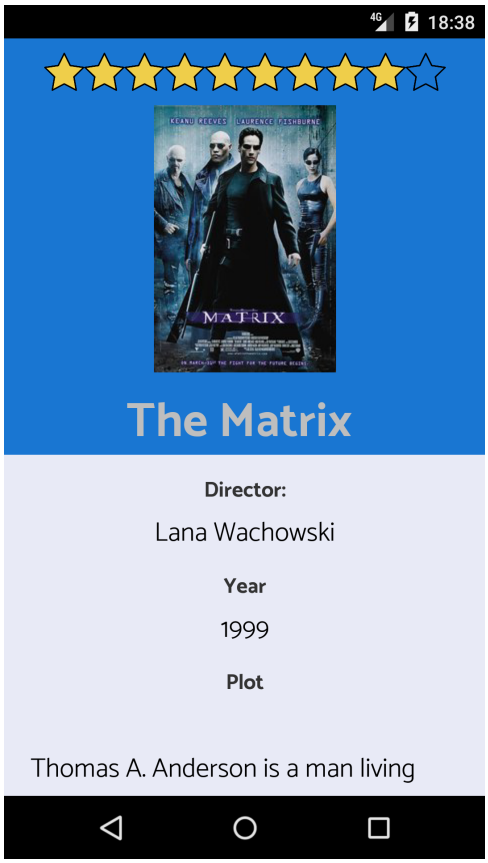


5. Interfejs użytkownika aplikacji mobilnej









6. Problemy i rozwiązania

6.1. Problem z elementami strony generowanymi w JavaScript'cie

Próbując pozyskać informacje ze strony *Filmweb.pl* istotnym problemem było uzyskanie informacji o ocenie wyszukanego filmu. Framework *Scrapy* nie jest w stanie wygenerować strony po wykonaniu skryptów Javascript. W tym celu należało użyć narzędzia *Splash*, które jest w stanie wykonać to zadanie. Rozwiązanie to jednak nie spełniało naszych oczekiwań ze względu na stopień skomplikowania i opóźnienia generujące w przetwarzaniu danych, dlatego nie znalazło się w finalnej implementacji.

6. 2. Paginacja

Problem powiązany jest z problemem 6.1. Lista, po której można przechodzić po kolejnych pozycjach jest generowana za pomocą JavaScript. Rozwiązaniem było znalezienie znacznika przycisku, "następny", który znajdował się w szkielecie strony niezależnie od generowanej odpowiedzi.

6. 3. Zmiana technologii implementacji API

Początkowo zamierzaliśmy zaimplementować API w języku Java z użyciem frameworka Spring. Od tej decyzji odsunął nas fakt, iż znacznie komplikowałoby to stworzenie aplikacji, która dynamicznie zwracałaby pozyskane informacje.

6. 4. Chaotyczność i nieczytelna struktura "scrapowanych" stron

7. Wymagania systemowe

W celu uruchomienia *scrapera* konieczna jest instalacja interpretera Python w wersji 2.7.3 lub wyższej. Przydatne będzie też narzędzie *pip*, czyli system zarządzania "paczkami" napisanymi w Pythonie. Następnie w lokalizacji projektu należy stworzyć środowisko testowe za pomocą `$ virtualenv scrapy_env,` aktywować je wykonać następujące komendy:

```
$ pip install scrapy
$ pip install klein
```

7. 1. Uruchomienie serwera lokalnie

Serwer lokalny uruchamiamy komendą `python $ app.py` w głownym folderze aplikacji.

7. 2. Uruchomienie aplikacji mobilnej

Program zbudować w środowisku Android Studio i wgrać na dowolne urządzenie z systemem Android.

9. Konkluzja

Niestety nie udało się zrealizować wszystkich celów, jakie założyliśmy na etapie planowania. Problem okazał się być nietrywialny, a nasze początkowe założenia okazały się być błędne w zderzeniu z rzeczywistością. Niemałym utrudnieniem był też fakt, że od połowy semestru prace musieliśmy wykonywać w pomniejszonym składzie.

9. 1. Wizje rozwoju projektu:

- rozszerzenie o dodatkowe strony
- aplikacja mobilna na inne platformy
- rozszerzenie o dodatkowe pola
- rozszerzenie kwerend o inne informacje (np. newsy, aktorzy, plotki itd.)