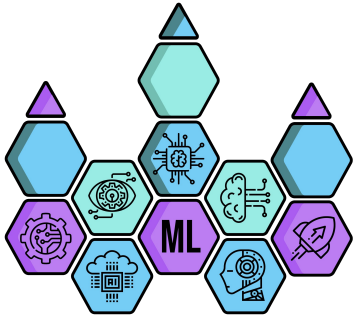
DATA
WORKSHOP



DW Poznań - projekt filmweb-rekomendacje #3

2019-11-26

Pierwsza struktura projektu

http://bit.ly/2qM66LR

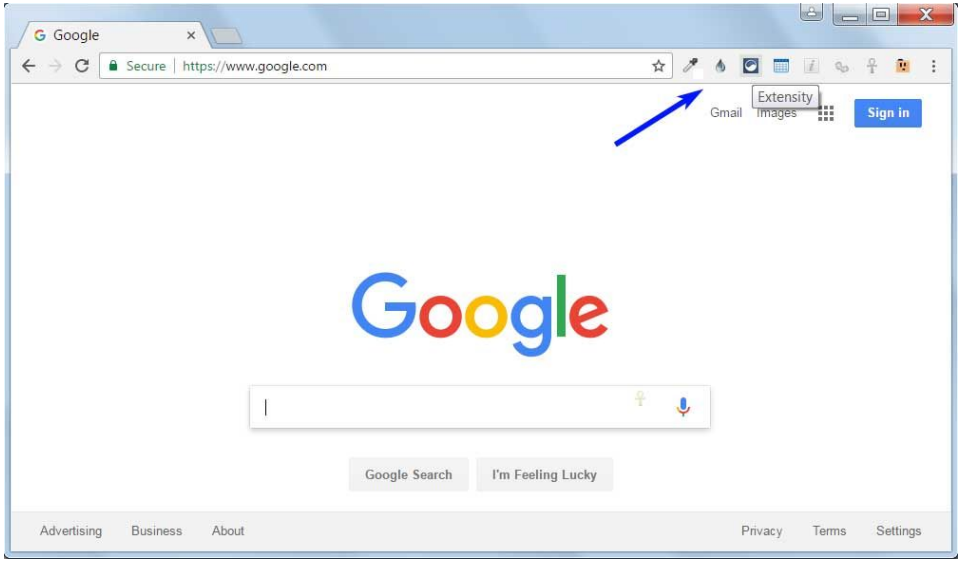
Agenda

01. **Chrome Extension** - Czyli jak rozszerzyć stronę filmweb o nowe możliwości
02. **Flask Server** - Jak przekazywać dane chrome extension do serwera
03. **IMDB** - Jak z naszych wcześniejszych spotkań stworzyć narzędzie do łączenia dwóch różnych baz danych.
04. **Wykresy** - Jak pokazać użytkownikowi przy pomocy `chart.js` pokazaż interesujące wykresy

https://github.com/alexiej/filmweb-rekomendacje

01. Chrome Extension

- Pozwala na rozszerzenie możliwości Google Chrome o własny kod HTML/JS który ma możliwość działania niezależnie od strony takie jak:
  - Zarządzanie hasłami
  - Blokowaniem reklam
  - Zarządzanie mailami w gmail i innych programach pocztowych
  - Zarządzanie zadaniami
  - Łatwiejsze kopiowanie tekstu ze stron



Zalety

- Piszę się tak samo jak każdą inną stronę HTML/CSS
- Łatwość instalacji i bogaty zbiór w bibliotece Google (Google Chrome WebStore)
- Brak wymagań co do instalowania dodatkowych komponentów (wystarczy tylko spakować katalog z rozszerzeniem)

Wady

- Działa tylko na Google Chrome
- Słabe zabezpieczenia. Po instalacji i dania dostępu pozwala na praktycznie dowolne działania
- Dużo aplikacji w Google Chrome extension to malware albo adware

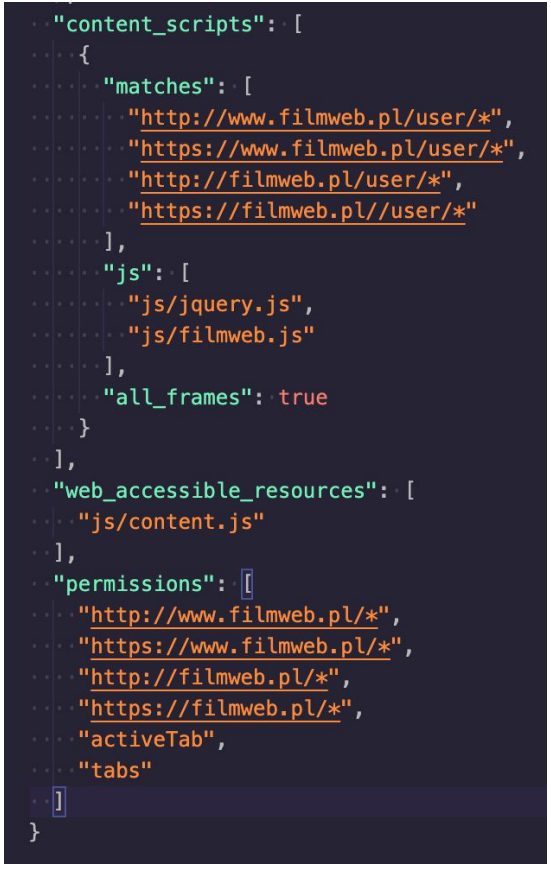
Struktura `manifest.js`

```
{
  "name": "Filmweb Rekomendacje",
  "version": "0.0.1",
  "manifest_version": 2,
  "description": "Pobierz najnowsze
  "homepage_url": "https://filmweb.p
  "icons": {
    "16": "icons/icon16.png",
    "48": "icons/icon48.png",
    "128": "icons/icon128.png"
  },
  "browser_action": {},
  "background": {
    "scripts": [
      "js/background.js"
    ],
    "persistent": true
  },
```

- **browser_action** - co ma się dziać po kliknięciu na rozszerzenie może pojawić się nasz html https://developer.chrome.com/extensions/browserAction
- **background** - kod wykonywany w tle niezależnie od strony. Można dodać różne akcje takie jak `OnClicked`. Po kliknięciu na nasze rozszerzenie otwiera się nowa strona https://developer.chrome.com/extensions/background_pages

```
chrome.browserAction.onClicked.addListener(function (tab) {
    chrome.tabs.create({url: 'https://www.filmweb.pl/'})
})
```

Struktura `manifest.js`

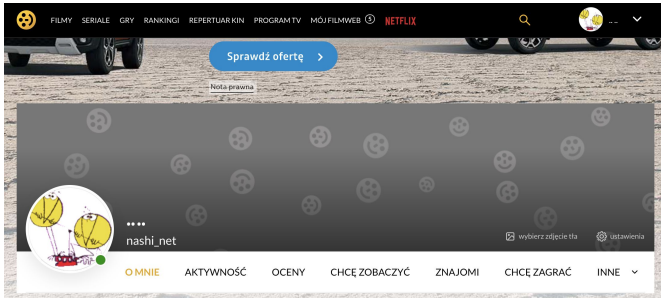


```
"content_scripts": [
  {
    "matches": [
      "http://www.filmweb.pl/user/*",
      "https://www.filmweb.pl/user/*",
      "http://filmweb.pl/user/*",
      "https://filmweb.pl//user/*"
    ],
    "js": [
      "js/jquery.js",
      "js/filmweb.js"
    ],
    "all_frames": true
  }
],
"web_accessible_resources": [
  "js/content.js"
],
"permissions": [
  "http://www.filmweb.pl/*",
  "https://www.filmweb.pl/*",
  "http://filmweb.pl/*",
  "https://filmweb.pl/*",
  "activeTab",
  "tabs"
]
}
```

- **content_scripts -** To skrypty które uruchamiają się na wskazanych stronach internetowych
  https://developer.chrome.com/extensions/content_scripts
- **web_accessible_resources -** dodatkowe dostępne zasoby stron które będą dodawane
  https://developer.chrome.com/extensions/manifest/web_accessible_resources
- **permissions -** lista pozwoleń wymagana przez rozszerzenie
  - **activeTab -** dostęp do zarządzania aktualną zakładką
  - **tabs -** dostęp do zakładek

Jak działa rozszerzenie

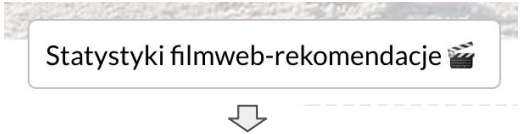
strona: filmweb.pl pasuje do matches w **content_script**



Uruchamiany jest `js/query.js` oraz `js/filmweb.js`

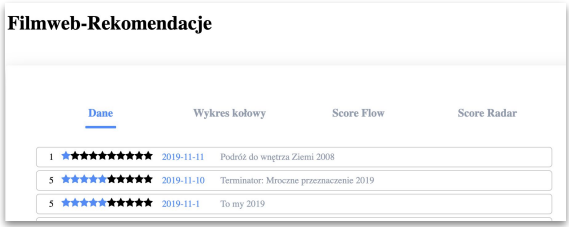
Filmweb.js wstrzykuje kod `js/content.js` do sekcji `body` strony (która jest zasobem w **web_accessible_resources**)

Za pomocą funkcji **Reset()** dodawany jest przycisk który po kliknięciu uruchomi **PrzygotowanieDoPobierania()**



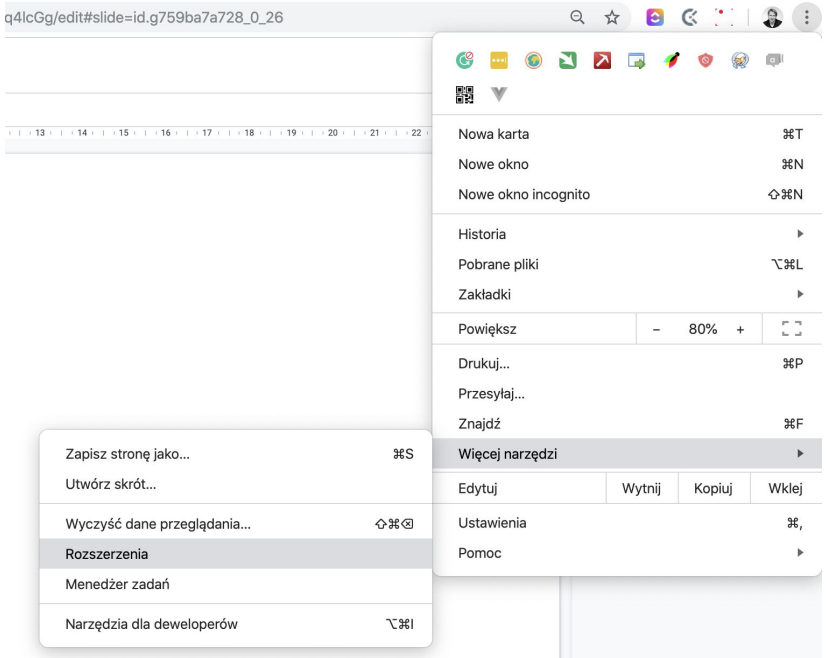
Tworzony jest niewidoczny obiekt **<table>** i za pomocą **PobierzOceny()** zostaje zescrapowana lista ocen. Po zakończeniu zostaje uruchomiona: **PokazWyniki()**

Wysyłany jest za pomocą POST i funkcji: **openWindowWithPost** informacja w postaci JSON do serwera Flask do serwera wskazanego w zmiennej **URL_SERVER** Otwierane jest nowe okono z wynikami



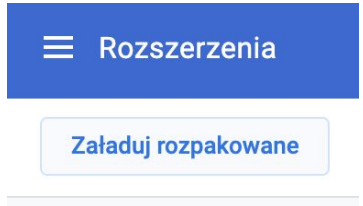
Instalowanie

1. Klikamy rozszerzenia



2. Zaznaczamy tryb dewelopera

3. Klikamy załaduj rozpakowane



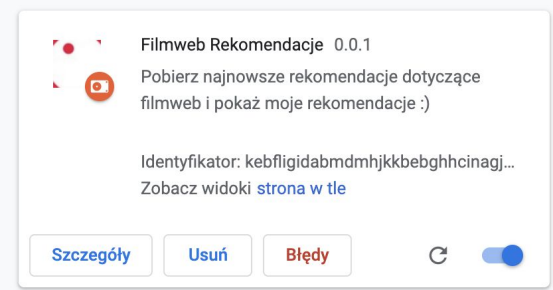
3. Wybieramy folder `chrome_extension` i klikamy ok

Instalowanie, cz. 2

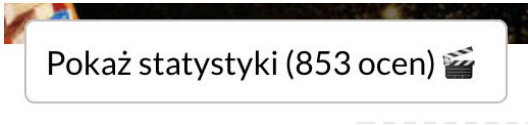
SERWER FLASK MUSI BYC URUCHOMIONY
python server.py

```
(dw-poznan) MacBook-Pro-Arkadiusz:filmweb-rekomendacje klemenka$ python server.py
 * Serving Flask app "server.server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
Called only once, when the first request comes in
```

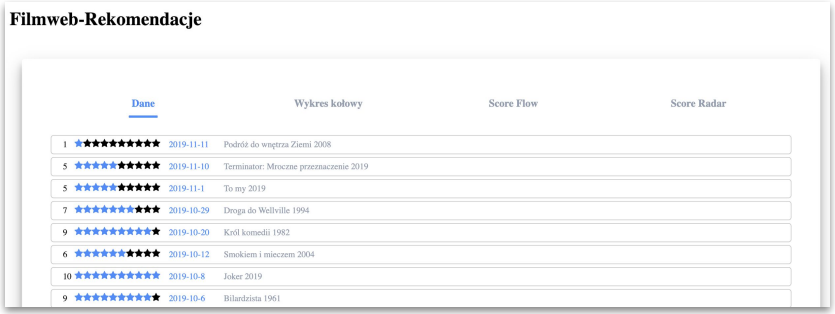
4. Nasze rozszerzenie pojawia się na liście

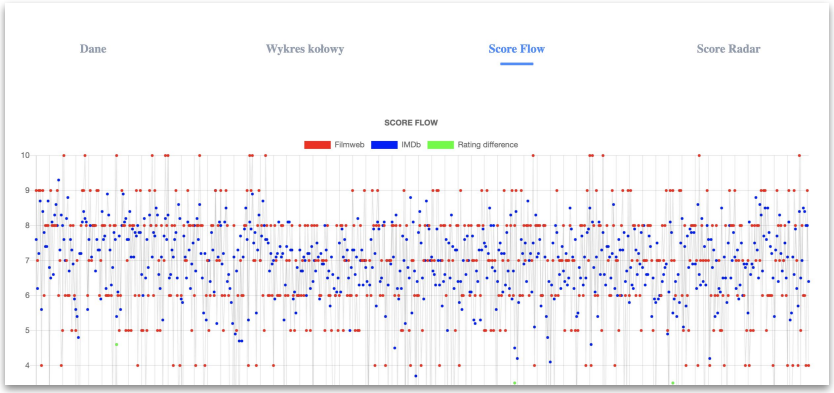


4. Na stronie filmweb swojej oceny klikamy `statystyki filmweb-rekomendacje` oraz po zakończeniu klikamy



5. Pojawia się nowa strona ze statystykami





02. Flask - Co to?

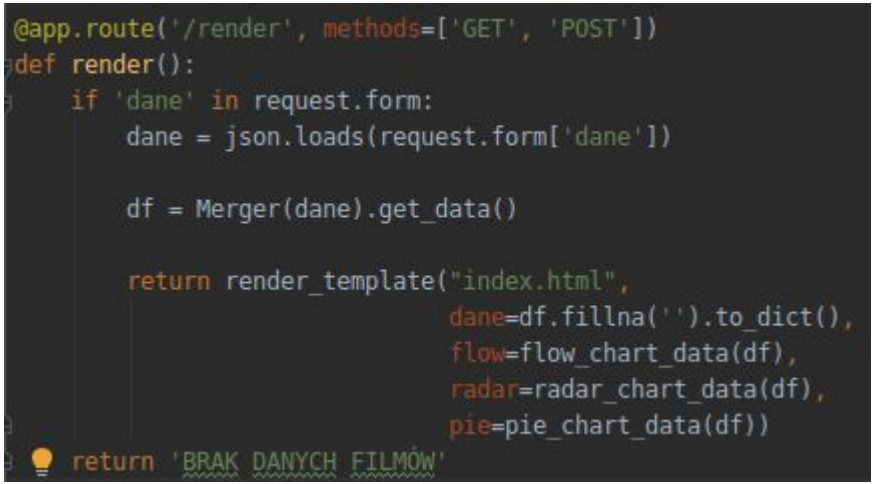
- Flask to mikro web framework napisany w Pythonie
- Pozwala łatwo tworzyć punkty serwera przez dekoratory w Pythonie
- Jest bardzo szybki

```python
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

Wady:

- Wszystkie zadania są wykonywane po kolei, czyli im więcej zapytań tym dłuższa odpowiedź.
- Do większych zastosowań wymaga dodatkowych modułów, nie obsługuje baz danych, użytkowników

Serwer Flask



```python
@app.route('/render', methods=['GET', 'POST'])
def render():
    if 'dane' in request.form:
        dane = json.loads(request.form['dane'])

        df = Merger(dane).get_data()

        return render_template("index.html",
                               dane=df.fillna('').to_dict(),
                               flow=flow_chart_data(df),
                               radar=radar_chart_data(df),
                               pie=pie_chart_data(df))
    return 'BRAK DANYCH FILMÓW'
```

- Jest podstawowy punkt render który przyjmuje przez **POST** zmienną JSON w postaci tekstu (**dane)**
- Klasa Merger dostarcza pandas.Dataframe z danymi o filmach
- **Zwracana jest strona** render_template(‘index.html’,...) z poszczególnymi danymi na każdą zakładkę, **dane, flow, radar, pie**

Inżynieria i łączenie danych

```
class Merger(object):

    def __init__(self, json):
        self.df = json_normalize(json)

    def get_data(self):
        df = Filmweb(self.df).get_dataframe(True)
        return Imdb().merge(df)
```

- Klasa Merger łączy otrzymane dane z Filmweb ze statystykami IMDb
- **json_normalize** - normalizuje nam strukturę json do płaskiej struktury, dzięki czemu każdy wiersz to jedna ocena
- **Filmweb(df).get_dataframe(True)** - powoduje przygotowanie dataframe do połączenia z imdb i uzupełnienia takich informacji jak **group, budget, boxoffice**
- **Imdb().merge(df)** - łączy tabelkę filmweb z globalną tabelą **Imdb()** na podstawie tytułu oryginalnego, roku, i kategorii
- Zwracany jest **pandas.Dataframe**

Przygotowanie danych IMDb

```
class Imdb(object):

    def __init__(self):
        self.imdb = pd.read_pickle(IMDB_MOVIES_PICLE)

    @staticmethod
    def prepare():
        imdb_title = pd.read_csv(IMDB_TITLE_GZIP, sep='\t', dtype='str', index_col='tconst', engine='c')
        imdb_title = imdb_title[imdb_title['titleType']=='movie']
        imdb_title = imdb_title.dropna(subset=['startYear', 'originalTitle'])

        pd.merge(
            imdb_title,
            pd.read_csv(IMDB_RATING_GZIP, sep='\t', dtype='str', index_col='tconst', engine='c'),
            how='left',
            left_index=True,
            right_index=True).to_pickle(IMDB_MOVIES_PICLE)
        pd.read_csv(IMDB_COVERS_CSV).to_pickle(IMDB_COVERS_PICLE)
```

- Podczas instalacji (**prepare()**) ogólne dane o filmach i ocenach z IMDb są filtrowane i zapisywane jednorazowo do pickle
- Uzyskano znaczący wzrost wydajności poprzez odpowiednie filtrowanie danych (interesują nas tylko rekordy typu ***movie)*** oraz połączenie danych po indeksach
- Łącząc dane Filmweb z IMDb dane są odczytywane z lokalnie z pickle

Uruchomienie serwera

```
#!/usr/bin/env bash

docker build -t myflask .
```

```
FROM python:3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
RUN python setup.py
ENTRYPOINT ["python"]
CMD ["server.py"]
```

```
#!/usr/bin/env bash

docker run -i -p 5000:5000 myflask
```

- **Docker** to narzędzie, które pozwala umieścić program oraz jego zależności w lekkim, przenośnym, wirtualnym kontenerze
- Instalacja serwera sprowadza się do:
    - zbudowania kontenera myflask (**./build.sh**), w którym rozwiązywane są zależności z pliku r**equirements.txt**
    - uruchomienia serwera **(./up.sh)**
- **Instalacja bez dockera:**

```
pip install -r requirements.txt
python server.py
```

https://www.chartjs.org/

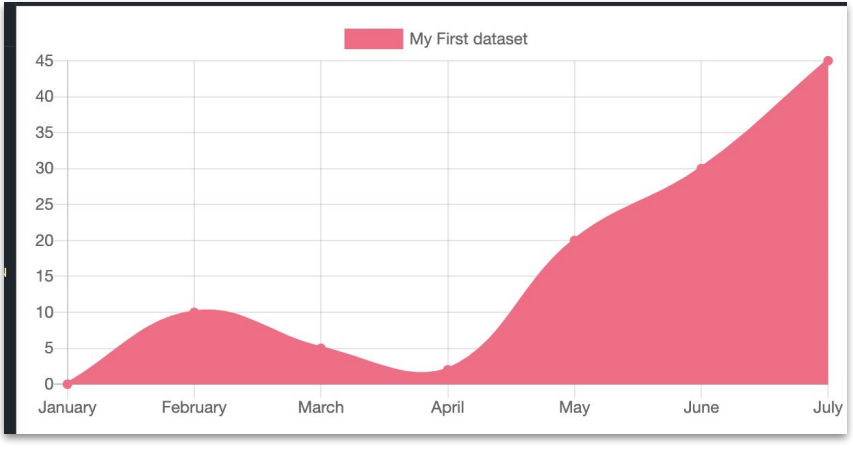
04. chart.js

- Chart.js to biblioteka JavaScript która w łatwy sposób pozwala wyświetlać wykresy

```
<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
```

```
<canvas id="myChart"></canvas>
```

```
var ctx = document.getElementById('myChart').getContext('2d');
var chart = new Chart(ctx, {
    // The type of chart we want to create
    type: 'line',

    // The data for our dataset
    data: {
        labels: ['January', 'February', 'March', 'April', 'May', 'June', 'July'],
        datasets: [{
            label: 'My First dataset',
            backgroundColor: 'rgb(255, 99, 132)',
            borderColor: 'rgb(255, 99, 132)',
            data: [0, 10, 5, 2, 20, 30, 45]
        }]
    },

    // Configuration options go here
    options: {}
});
```



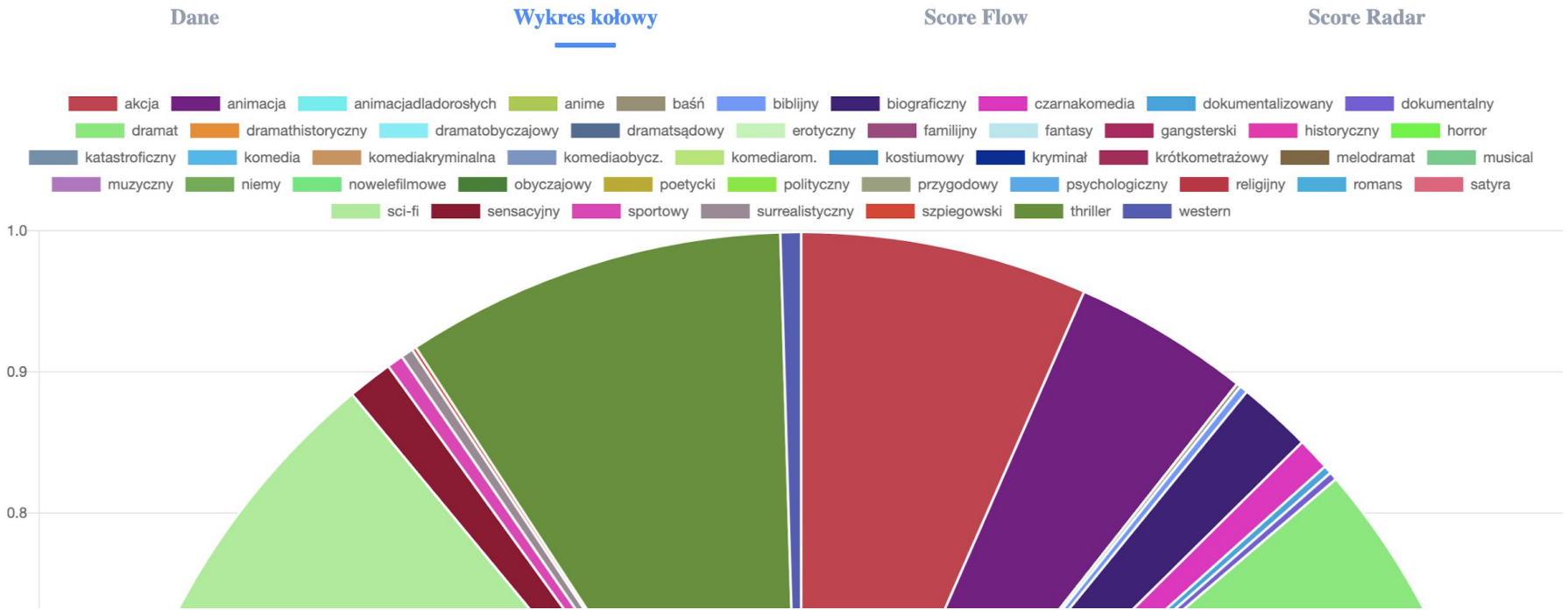
04a. Zakładka Dane

Dane | Wykres kołowy

1	2019-11-11	Podróż do wnętrza Ziemi 2008
5	2019-11-10	Terminator: Mroczne przeznaczenie 2019
5	2019-11-1	To my 2019
7	2019-10-29	Droga do Wellville 1994
9	2019-10-20	Król komedii 1982
6	2019-10-12	Smokiem i mieczem 2004
10	2019-10-8	Joker 2019
9	2019-10-6	Bilardzista 1961
8	2019-10-5	Chłopcy z ferajny 1990
7	2019-9-30	Ad Astra 2019
8	2019-9-28	Green Book 2018
6	2019-9-19	Agentka specjalnej troski 2016

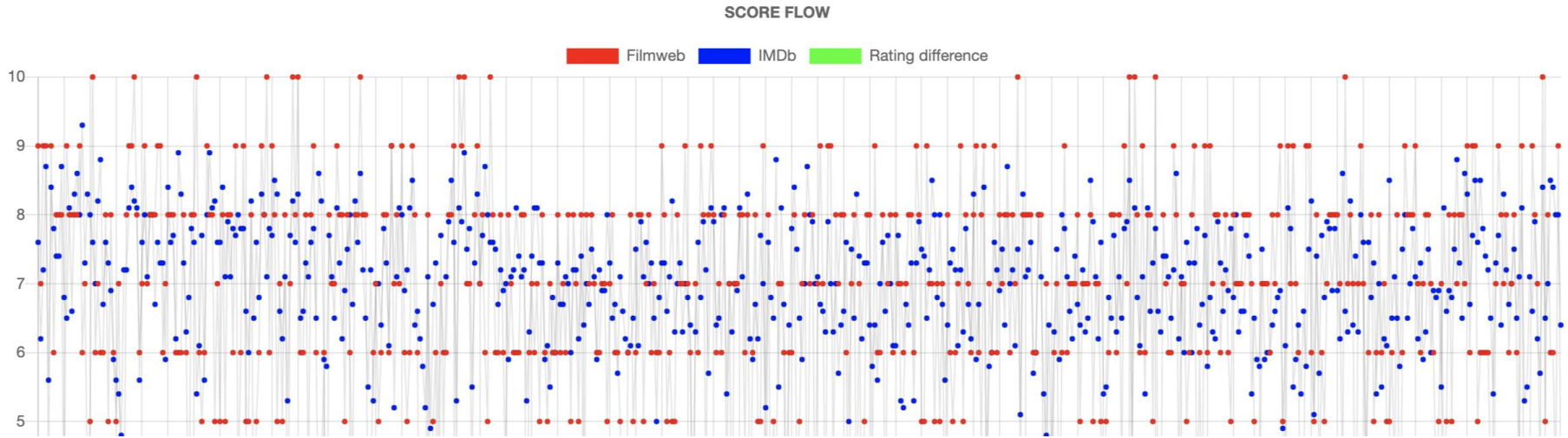
```
<section>
    <div id="example" class="dane">
        <div v-for="m in dane" class="row">
                <div class="ocena">[[m.ocena]]</div>
                <span class="fa fa-star" :class="{'checked': m.ocena>0}"></span>
                <span class="fa fa-star" :class="{'checked': m.ocena>1}"></span>
                <span class="fa fa-star" :class="{'checked': m.ocena>2}"></span>
                <span class="fa fa-star" :class="{'checked': m.ocena>3}"></span>
                <span class="fa fa-star" :class="{'checked': m.ocena>4}"></span>
                <span class="fa fa-star" :class="{'checked': m.ocena>5}"></span>
                <span class="fa fa-star" :class="{'checked': m.ocena>6}"></span>
                <span class="fa fa-star" :class="{'checked': m.ocena>7}"></span>
                <span class="fa fa-star" :class="{'checked': m.ocena>8}"></span>
                <span class="fa fa-star" :class="{'checked': m.ocena>9}"></span>

            <div class="data"> [[m.data]] </div>
            <div class="info">[[m.tytułpolski]] [[m.rokprodukcji]]</div>
        </div>
    </div>
</section>
```

04b. Zakładka wykres kołowy

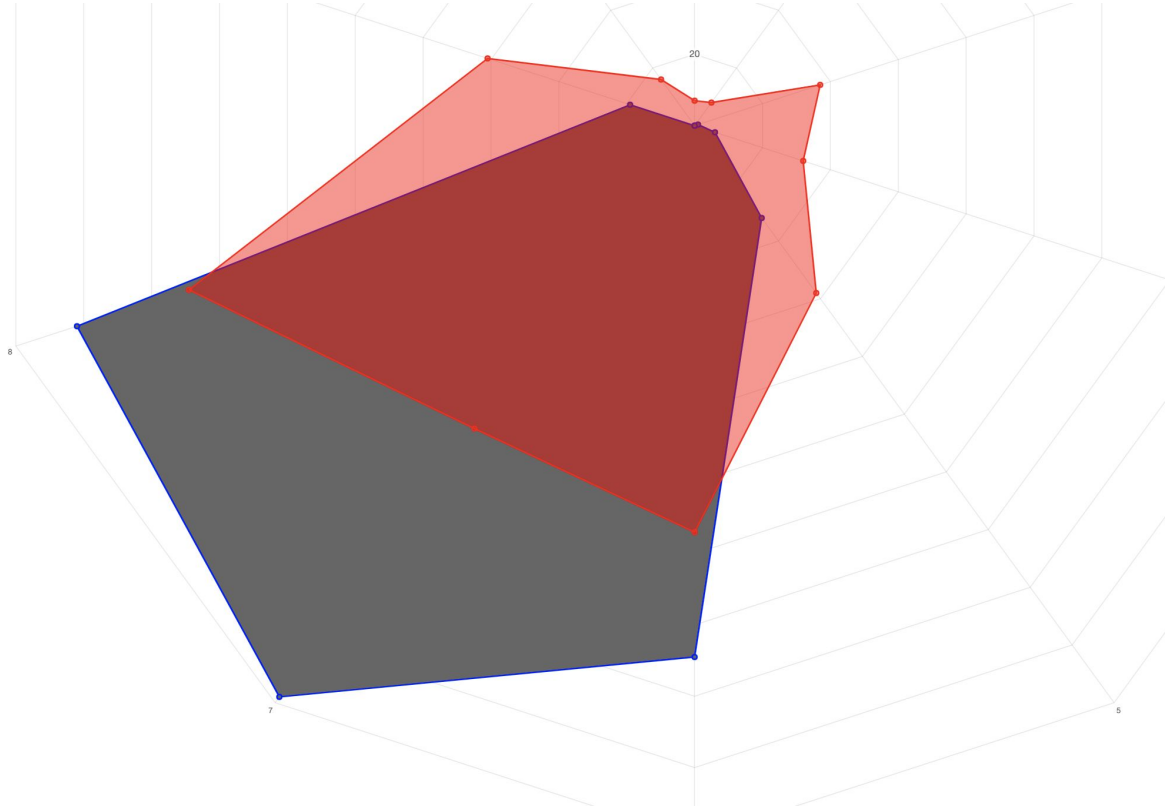


04c. Zakładka score flow

Dane | Wykres kołowy | Score Flow | Score Radar



04d. Zakładka score-radar



Podsumowanie

- Za pomocą czystego js/html można zrobić bardzo fajne rozszerzenie do chrome-a które pokazuje statystyki naszych wyborów
- Dzięki flask możemy w łatwy sposób zbudować serwer. Za pomocą kilku linijek można zrobić prostą stronę zwracającą wykresy.
- Dzięki `vue.js` i `chartjs` możemy w łatwy sposób manipulować danymi i wyświetlać je na ekranie.

Kolejne kroki

- Jeszcze więcej wykresów :)
- Użycie biblioteki http://surpriselib.com/ SURPRISE do rekomendacji ulubionych filmów
- Użycie ImdbPy do pobierania **Cover** zdjęć plakatów filmów i wyświetlanie ich użytkownikowi.

https://github.com/dataworkshop/dw-poznan-project

Dziękuję