

# TASS - dokumentacja

## TREŚĆ

Analiza współ publikowania z autorami patentów: dla wskazanej grupy patentów<sup>+</sup> (dziedziny, zakresu czasu) sprawdzić, czy sieć badaczy generowana przez patenty (na zasadzie wspólnego patentowania) różni się istotnie od sieci wynikającej z wspólnego publikowania.

## DANE

### ŹRÓDŁO DANYCH

Dla patentów:

- <https://polon.nauka.gov.pl/opi/aa/pid/patent>
- <https://radon.nauka.gov.pl/opendata/polon/products>

Dla publikacji:

- <https://radon.nauka.gov.pl/opendata/polon/publications>

Dane dodatkowe:

- <https://radon.nauka.gov.pl/opendata/polon/employees>

### MECHANIZM POBIERANIA DANYCH

Dla patentów:

- Ograniczymy się do typu “wynałazek”
- Aby usunąć stare patenty, będą nas interesować te które zostały zarejestrowane po 2016r.
- Korzystamy z rest api - polon

Dla publikacji:

- Użyjemy rest api - radon.

---

## SPOSOBY PRZECHOWYWANIA DANYCH

Dla patentów:

- Dla powyższych ograniczeń, pobierzemy wszystkie patenty i zapiszemy w relacyjnej bazie danych

Dla patentów powstaną trzy tabele w których zostaną zapisane:

- Data
- Kraj / Region
- Nazwa produktu
- Rodzaj produktu (zapisujemy ale będzie to Wynalazek)
- Twórcy - w osobnej tabeli
- Jednostki - w osobnej tabeli

Dla publikacji:

Po pobraniu danych dotyczących patentów i zapisaniu jej do bazy danych, dla uzyskanej w ten sposób listy osób będą dokonywane zapytania do odpowiednich serwisów dotyczące publikacji danego autora.

Dla publikacji będziemy zapisywać:

- Data publikacji
- Nazwa publikacji
- Autorów - w osobnej tabeli

Część ze wspomnianych zapisywanych danych (np. Kraj, rodzaj produktu) nie będą wyświetlane bezpośrednio w końcowym zestawieniu, a będą pełniły rolę pomocniczą przy łączeniu danych.

## METODA ŁĄCZENIA DANYCH

Najpierw pobierane są wspomniane informacje dotyczące patentów oraz dane pracowników naukowych zarejestrowanych w systemie polon.

Potem następuje łączenie danych pracowników z autorami patentów na podstawie nazwiska, imion, instytucji zatrudniających.

---

Ten krok jest czyniony w celu uzyskania większej ilości informacji o danej osobie. Jeżeli autor patentu nie zostanie połączony z żadnym pracownikiem, to tworzony jest nowy profil i dodawany do zbioru danych.

Kolejnym krokiem jest pobranie publikacji związanych z danym autorem patentu.

Odgrywa się to poprzez wysłanie zapytania przez api o publikację autora o danym nazwisku, a następnie wszystkie uzyskane publikacje są dopasowywane do autora patentu.

Informacje które odgrywają tutaj rolę to:

- Imię - często zdarza się, że przy autorze publikacji widnieje jedynie inicjał imienia, co jest odpowiednio wykrywane
- Inne osoby powiązane z patentem - jeżeli na liście współautorów publikacji widnieją osoby powiązane przez wspólny patent, to zwiększany jest współczynnik powiązania danego autora patentu z daną publikacją

Z kolei informacje, które nie odgrywają roli przy łączeniu z publikacjami, choć mailowo zadeklarowaliśmy, że będą brane pod uwagę:

- Instytuty osób związanych z patentami
- dziedziny nauki, którymi zajmują się osoby związane z patentami
- słowa kluczowe przy publikacjach

Dzieje się tak dlatego że nie dysponowaliśmy skutecznym pomysłem na wiązanie dziedzin nauki i instytutów z słowami kluczowymi przy publikacjach. Występują tam często zupełnie różne słowa, których powiązanie przy użyciu odległości Levenshteina nie dawało większego efektu, a nie znamy bibliotek porównujących semantyki wyrazów.

Uznaliśmy, że lepszym rozwiązaniem będzie kierowanie się tymi parametrami które wymieniliśmy wcześniej.

Następnie jeżeli współczynnik powiązania przekroczy pewną zadaną wartość to przypisuje się publikację do autora patentu

Ostatecznie w bazie danych będziemy mieli trzy najważniejsze tabele:

- Autorzy
- Patenty
- Publikacje

---

Gdzie tabele: Autorzy - Patenty i Autorzy - Publikacje będą połączone relacją wiele do wielu. Po pobraniu i zapisaniu danych powinniśmy dysponować pełnym zbiorem informacji potrzebnych do odpowiedzenia na pytanie z treści projektu.

## METODA ANALIZY DANYCH

Informacje dotyczące uzyskanych danych będą podzielone na dwie części:

- Informacje analityczne, dotyczące takich zagadnień jak
  - średni stopień wierzchołka
  - średnia gęstość
  - współczynnik klasteryzacji
- Graficzna reprezentacja powiązań pomiędzy autorami - Tutaj wierzchołki odpowiadałyby autorom, a krawędzie wspólnym patentom/publikacjom - jeżeli jakaś grupa autorów jest autorami tego samego patentu, lub tej samej publikacji, to każdy z wierzchołków ich reprezentujących będzie posiadał krawędź z pozostałymi. W rezultacie, każda grupa współautorów będzie tworzyła klikę.

Krawędzie dotyczące publikacji i patentów będą reprezentowane przez różne kolory.

Na interfejsie użytkownika będzie opcja ukrycia krawędzi dotyczących patentów/publikacji i wyświetlania tylko tych drugich, albo obu na raz.

Waga krawędzi będzie odwrotnie proporcjonalna do ilości współprac między autorami, czyli w przypadku pojedynczej publikacji/patentowania waga wynosi 1, a w przypadku 5 publikacji waga osiągnie wartość  $\frac{1}{5}$ . W ten sposób chcielibyśmy uzyskać różne grubości krawędzi pozwalające ocenić intensywność współpracy z innymi autorami.

W interfejsie użytkownika będzie opcja wyboru zakresu wyświetlanego grafu. Można wyświetlić zarówno graf przedstawiający wszystkich autorów i ich powiązania, lub skupić się tylko na wybranej osobie. W tym drugim przypadku zostanie wyświetlone tylko najbliższe sąsiedztwo danego wierzchołka - osoby o wspólnym patencie/publikacji.

## APLIKACJA

### STOS TECHNOLOGICZNY

- Python 3
- Angular 11

- 
- MySQL - uruchomiona w dockerze

Dodatkowe biblioteki

- Requests: HTTP for Humans™ - <https://requests.readthedocs.io/en/master/>
- GraphQL
- fuzzywuzzy - biblioteka do wykrywania powiązanych imion

Dodatkowe biblioteki Angular:

- Bootstrap
- Angular Material
- ngx-graph - do rysowania grafów
- ngx-spinner

## CEL I UŻYCIE APLIKACJI

### Użycie aplikacji

Aplikacja będzie składać się z 3 modułów. Części frontend'owej - Angulara, części backend'owej - Pythona i bazy danych.

Użytkownik będzie mógł wchodzić w interakcje z systemem poprzez przeglądarkę internetową.

Będzie mógł wprowadzać wielorakie filtry, aby otrzymać wynik. Ponadto na stronie internetowej zostanie dodany przycisk, który jest wyzwalaczem do pobierania danych przez backend. Tj. po naciśnięciu przycisku, backend pobierze wszelkie niezbędne do działania dane.

### Odstępstwa od koncepcji

Aplikacja składa się z części frontendowej (angular), ale także z dwóch części backendowych tj. Java + Spring oraz Python - jest to odstępstwo od koncepcji, ponieważ miał być sam Python.

Uzasadnieniem tej decyzji, jest to, że chcieliśmy poznać nową technologię tworzenia webowych aplikacji dlatego zadaniem części Pythona jest pobranie i uzupełnienie bazy danych MySQL, natomiast moduł Java + Spring ma za zadanie pobierać dane z bazy danych, a następnie przekazywać je do frontendu.

Kolejnym technologicznym odstępstwem jest użycie **dockera** do uruchomienia bazy danych, a także phpMyAdmina - w koncepcji nic o tym nie wspominaliśmy.

Ostatnim odstępstwem jest sposób łączenia danych autorów publikacji z autorami patentów, co zostało opisane wyżej, w rozdziale **Metoda łączenia danych**.

---

## Instrukcja instalacji i obsługi aplikacji

Instrukcja instalacji i obsługi aplikacji, zależy od modułu.

### Moduł bazy danych

W celu uruchomienia bazy danych należy przejść do katalogu `/backend_spring/tass-spring`, a następnie wykonać polecenie:

```
docker-compose -f docker-compose-local.yml up
```

Po wykonaniu polecenia, zostanie uruchomiona baza danych MySQL w wersji 8 oraz phpMyAdmin, który od razu będzie z nią połączony.

Aby “zajrzeć” do bazy danych, używamy w tym celu phpMyAdmin. Po wykonaniu uprzednio opisanego polecenia, możemy przejść na <http://localhost:8185>. Na stronie wpisujemy dane, znajdujące się w pliku **mysql.env** w katalogu **docker**.

### Moduł Java + Spring

Moduł jest niezbędny do poprawnego działania części frontendowej, ponieważ dostarcza do niej dane.

W celu zbudowania modułu, należy przejść do katalogu `/backend_spring/tass-spring`, a następnie wykonać polecenie:

```
gradlew clean build
```

W celu uruchomienia modułu, należy przejść do katalogu `/backend_spring/tass-spring/build/libs`, a następnie wykonać polecenie:

```
java -jar tass-spring-0.0.1-SNAPSHOT.jar
```

Teraz możemy przejść do uruchomienia modułu frontendowego.

### Moduł Frontendowy (Angular)

Po uruchomieniu modułu (Java + Spring), możemy uruchomić moduł frontendowy.

W tym celu należy przejść do katalogu `frontend/`, a następnie uruchomić polecenie `ng serve`.

Angular zostanie uruchomiony na domyślnym porcie 4200, czyli pod adresem:

<http://localhost:4200>

## Moduł Python

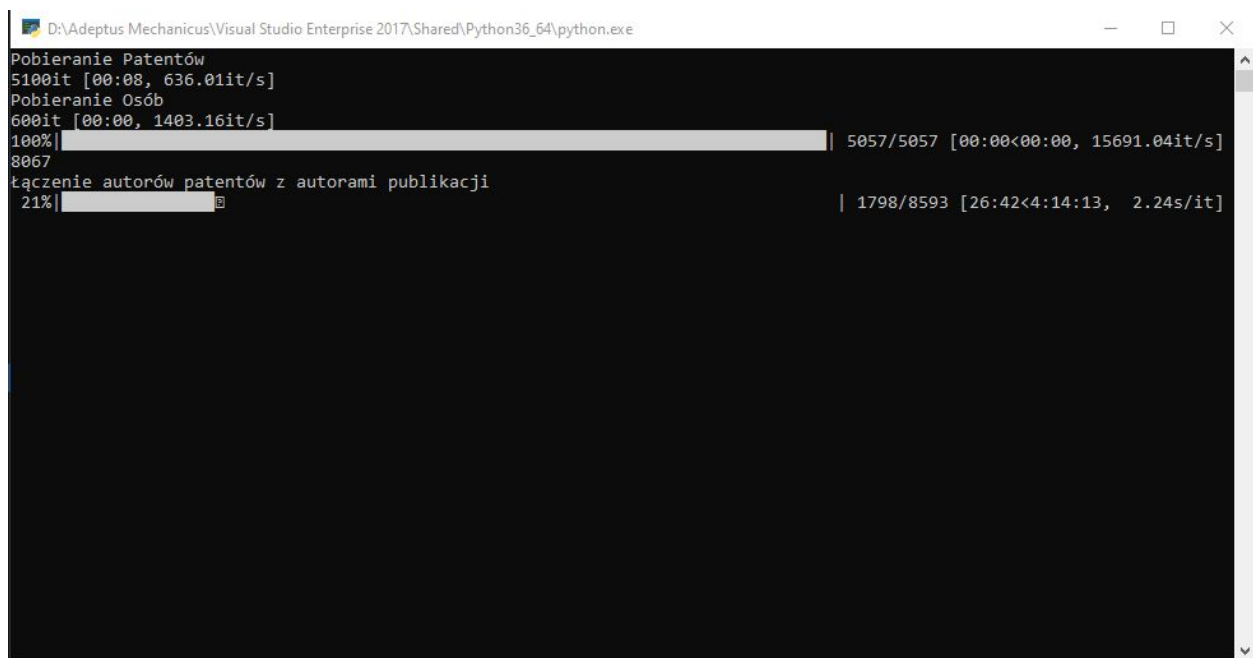
Python odpowiada za pobranie danych z serwisów zewnętrznych, przetworzeniu ich, a następnie zapisaniu w Bazie danych.

Uruchomienie programu polega na wywołaniu funkcji Main.py poleceniem

```
python Main.py
```

Proces pobierania powinien się od razu rozpocząć, a informacje o tym co jest aktualnie wykonywane będzie wypisywane w konsoli.

Domyślnie łączy się z bazą danych uruchomioną lokalnie (dokładne informacje w pliku: MySQL\_connection.py), a pobierane dane są specyfikowane w pliku: Downloading\_manager.py



```
D:\Adeptus Mechanicus\Visual Studio Enterprise 2017\Shared\Python36_64\python.exe
Pobieranie Patentów
5100it [00:08, 636.01it/s]
Pobieranie Osób
600it [00:00, 1403.16it/s]
100% | 5057/5057 [00:00<00:00, 15691.04it/s]
8067
Łączenie autorów patentów z autorami publikacji
21% | 1798/8593 [26:42<4:14:13, 2.24s/it]
```

Powyżej przykładowe uruchomienie programu napisanego w pythonie zbierającego dane.

## ŹRÓDŁA

- <https://pypi.org/>

- 
- <https://www.wikipedia.org/>
  - <https://pypi.org/project/scholarly/>