Programowanie I R

Projekt II: ORCID.

Bartłomiej Zglinicki

1. Wprowadzenie

ORCID (ang. *Open Researcher and Contributor ID*) jest systemem służącym do identyfikowania autorów publikacji naukowych. Każdemu z zarejestrowanych w systemie naukowców przypisany jest unikalny numer, który jednoznacznie identyfikuje tę osobę (imię i nazwisko mogą się przecież powtarzać). Na przykład identyfikator prof. Dariusza Wasika, Dziekana Wydziału Fizyku UW, to 0000-0001-5173-8230. Wchodząc na stronę `https://orcid.org/identyfikator-orcid` (a zatem, dla przykładu, w przypadku prof. Wasika na stronę `https://orcid.org/0000-0001-5173-8230`), możemy zobaczyć podstawowe informacje o badaczu i listę publikacji, których jest autorem lub współautorem (o ile dane te zostały uzupełnione i nie są w całości lub częściowo ukryte).

System ORCID udostępnia też API (ang. *Application Programming Interface – interfejs programowania aplikacji*), pozwalając aplikacjom na łatwe przetwarzanie zawartych w nim danych. Najprostszy sposób wykorzystania tego API, niewymagający autoryzacji i pozwalający wyłącznie na odczyt danych, polega na połączeniu się z adresem `https://pub.orcid.org/identyfikator-orcid` (notabene innym niż poprzednio – pojawia się subdomena `pub`), pozwalającym na pobranie zawartych w systemie ORCID informacji w formacie XML. W języku Python możemy to zrobić, korzystając np. z modułu `urllib`:

```
import urllib.request
with urllib.request.urlopen("https://pub.orcid.org/identyfikator-orcid") as f:
   info = f.read().decode("utf-8")
   # operacje na danych zawartych w zmiennej info...
```

Uzyskane informacje zdają się mieć dosyć skomplikowaną postać, jednak to tylko pozory; wręcz przeciwnie, mają one czytelną i łatwą do analizy strukturę. Co więcej, Python umożliwia bardzo łatwe i wygodne przetwarzanie danych w formacie XML – można w tym celu wykorzystać np. moduł `xml.dom.minidom` lub moduł **xml.etree.ElementTree**. Opis i pouczające przykłady stosowania tych modułów dostępne są m.in. pod adresem

`https://www.edureka.co/blog/python-xml-parser-tutorial/`

I tak, imiona analizowanego naukowca poznamy, odczytując wartość elementu `personal-details:given-names`, jego nazwisko – sprawdzając wartość elementu `personal-details:family-name`. Informacje o jego publikacjach zawarte są w elemencie `activities:works`. W przypadku każdej publikacji podany jest m.in. jej tytuł, zawarty w elemencie `common:title`, oraz, zazwyczaj, identyfikator DOI (ang. *digital object identifier*), który można znaleźć w jednym z elementów `common:external-id` i/lub w elemencie `common:url` (w postaci odnośnika do strony `https://doi.org/identyfikator-doi`).

DOI jednoznacznie identyfikuje opublikowane w internecie prace naukowe (oraz inne dokumenty elektroniczne), podobnie jak ORCID jednoznacznie identyfikuje ich autorów. Znając identyfikator DOI publikacji, możemy wejść na adres `https://doi.org/identyfikator-doi`, który natychmiast przeniesie nas na stronę wydawcy, zawierającą informacje o tej publikacji. System DOI, podobnie jak ORCID, udostępnia API, pozwalający łatwo pozyskać dane o publikacji w formacie XML (oraz kilku innych) – wystarczy wykorzystać ten sam

adres `https://doi.org/identyfikator-doi` dołączając nagłówek `Accept: application/rdf+xml`. W języku Python można to zrobić m.in. w ten sposób:

```
import urllib.request   # o ile ta instrukcja nie pojawiła się wcześniej
req = urllib.request.Request("https://doi.org/identyfikator-doi")
req.add_header("Accept", "application/rdf+xml")
with urllib.request.urlopen(req) as f:
   info = f.read().decode("utf-8")
   # operacje na danych zawartych w zmiennej info...
```

Wśród uzyskanych w ten sposób informacji znajdziemy m.in. pełną listę autorów publikacji, datę jej wydania, wydawcę oraz dane czasopisma, w którym została opublikowana.

Niektóre publikacje nie mają numeru DOI – dotyczy to głównie preprintów. W takiej sytuacji wśród informacji dostarczonych przez system ORCID (w jednym z elementów `common:external-id`) znajdziemy zazwyczaj identyfikator arXiv – obszernego elektronicznego archiwum prac naukowych, gromadzącego publikacje z dziedziny nauk ścisłych, w tym także preprinty. Baza arXiv również posiada wygodny API umożliwiający pozyskanie informacji o publikacji w formacie XML – tym razem należy wykorzystać adres `http://export.arxiv.org/api/query?id_list=identyfikator-arXiv`. W języku Python wystarczy zatem, jak zwykle, skorzystać np. z modułu `urllib`:

```
import urllib.request   # o ile ta instrukcja nie pojawiła się wcześniej
with urllib.request.urlopen("http://export.arxiv.org/api/
                                        query?id_list=identyfikator-arXiv") as f:
   info = f.read().decode("utf-8")
   # operacje na danych zawartych w zmiennej info...
```

Poza listą autorów i datą publikacji uzyskamy w ten sposób m.in. streszczenie pracy.

2. Opis projektu

Celem projektu jest napisanie programu `orcid`, przyjmującego jako argument wywołania identyfikator ORCID i wypisującego na ekranie imię i nazwisko osoby reprezentowanej przez ten identyfikator oraz listy jej publikacji. Przy każdej publikacji powinny zostać podane następujące informacje na jej temat: tytuł, pełna lista autorów, data publikacji, nazwa czasopisma, jego wydanie i wydawca, identyfikator DOI i/lub arXiv oraz adres pliku PDF z pracą, przy czym dana informacja powinna się pojawić tylko wówczas, gdy można ją pozyskać na podstawie informacji bezpośrednio zawartych w bazie ORCID lub dzięki identyfikatorom DOI i/lub arXiv zawartym w bazie ORCID. Jeśli wśród argumentów wywołania programu znajdzie się dodatkowo sekwencja `-o nazwapliku.txt`, informacje nie powinny być wypisywane na ekranie, ale zapisywane w pliku tekstowym `nazwapliku.txt`.

W projekcie nie można wykorzystywać gotowych modułów służących do komunikacji z API systemów ORCID, DOI i arXiv (np. `orcid`, `python-doi`).

Warto podkreślić, że istnieją też inne bazy danych dotyczące pracowników nauki i ich publikacji oraz sieć powiązań między nimi. W tym projekcie dla uproszczenia ograniczamy się jednak do informacji, do których da się łatwo dotrzeć dzięki danym udostępnianym przez ORCID.