Standard techniczny

STANDARDY OTWARTOŚCI DANYCH

Fundusze Europejskie Polska Cyfrowa
Ministerstwo Cyfryzacji
OTWARTE DANE
Unia Europejska Europejski Fundusz Rozwoju Regionalnego

Spis treści

Wstęp

Dokument definiuje zestaw minimalnych wymagań technicznych, które muszą spełniać dane publiczne, w szczególności udostępniane w Centralnym Repozytorium Informacji Publicznej (portal danepubliczne.gov.pl), w tym za pośrednictwem API.

Użyte w dokumencie sformułowania wymaga się, zaleca się, dopuszcza się, nie zaleca się, nie dopuszcza się oznaczają odpowiednio:

- **wymaga się** – oznacza, że aby zachować zgodność ze standardem, trzeba bezwarunkowo spełnić opisane kryteria/zasady,
- **zaleca się** – oznacza, że w uzasadnionych przypadkach można nie spełnić danego kryterium, pod warunkiem, że opisująca zasoby dokumentacja (np. dokumentacja API, opisana w Standardzie API[1]) zawiera uzasadnienie decyzji o odstępstwie,
- **dopuszcza się** – oznacza, że można zastosować opisane kryterium lub go nie zastosować, a dokumentacja tej decyzji nie jest wymagana (choć można ją zamieścić),
- **nie zaleca się** – oznacza, że w wyjątkowych i uzasadnionych przypadkach można postąpić w opisany sposób pod warunkiem, że opisująca zasoby dokumentacja (np. dokumentacja API, opisana w Standardzie API) zawiera uzasadnienie decyzji o odstępstwie od danego zalecenia,
- **nie dopuszcza się** – oznacza, że aby zachować zgodność ze standardem, nie wolno postąpić w opisany sposób.

[1] Dotyczy Standardu API przygotowanego w ramach projektu „Otwarte dane – dostęp, standard, edukacja", dofinansowanego z poddziałania 2.3.1 Programu Operacyjnego Polska Cyfrowa
https://danepubliczne.gov.pl/article/standard-interfejsu-programistycznego-aplikacji-api-final

Poziomy otwartości danych

Schemat pięciu poziomów otwartości danych, zaproponowany przez Tima Bernersa-Lee, to użyteczne narzędzie klasyfikowania zasobów pod kątem ich otwartości. Model otwartości opisany tym schematem został przyjęty w Programie Otwierania Danych Publicznych[2]. Więcej informacji o tym schemacie można znaleźć na stronach W3C[3].

Im wyższy poziom otwartości danych, tym dane są lepiej przygotowane do dalszego przetwarzania. Wszystkie otwarte dane są udostępniane bez żadnych ograniczeń do dowolnych celów komercyjnych i niekomercyjnych. Schemat pięciu poziomów otwartości danych jest opisany w tabeli 1.

UWAGI:

Dane na poziomach otwartości 3 ★★★ i wyższych zaleca się udostępniać przez API, co pozwala na ich maszynowe przetwarzanie i opatrzenie dodatkowymi metadanymi.

Nie zaleca się publikowania danych na pierwszym poziomie otwartości.

[2] https://danepubliczne.gov.pl/article/program-otwierania-danych-publicznych

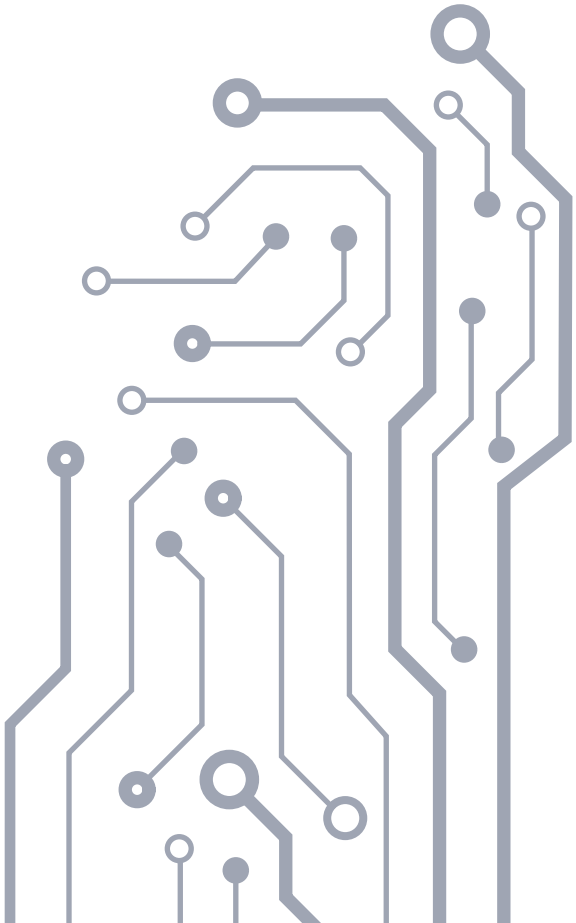
[3] https://www.w3.org/DesignIssues/LinkedData.html

Tabela 1. Schemat pięciu poziomów otwartości danych

Poziom	Oznaczenie	Charakterystyka poziomu	Dodatkowe informacje	Opis
1	★	Dane w dowolnym formacie udostępniane są bez ograniczeń licencyjnych (Więcej na ten temat w **Standardzie prawnym**)	Dopuszczalne jest publikowanie danych w formie plików graficznych, skanów z obrazem danych i tekstu, plików tekstowych	Mogą to być pliki JPEG z zeskanowanymi dokumentami, wygenerowane z różnych programów pliki PDF lub pliki tekstowe zawierające dane nieustrukturyzowane albo o strukturach niejednorodnych. Większość ludzi może je odczytać, ale jakiekolwiek dalsze ich wykorzystywanie wymaga dodatkowej pracy, polegającej na zidentyfikowaniu, odczytaniu i przeniesieniu danych (często ręcznie bądź ze wspomaganiem programów typu OCR).
2	★★	Dane są udostępniane w postaci ustrukturyzowanej	Dane publikowane są w postaci sformatowanego arkusza kalkulacyjnego	Dane mają już określoną strukturę, którą można odczytać komputerowo, na przykład poprzez zastosowanie pliku w formacie arkusza kalkulacyjnego lub edytora tekstu. Nie są to zeskanowane w niedającej się przeszukiwać formie obrazy. Natomiast dane te są w formacie zamkniętym (własnościowym), dla którego stosowanie w oprogramowaniu jest ograniczone przez restrykcje patentowe, licencyjne lub podobne.

Poziom	Oznaczenie	Charakterystyka poziomu	Dodatkowe informacje	Opis
3	★★★	Używanie formatów otwartych	Dane są publikowane w otwartym ustrukturyzowanym formacie, np. CSV lub JSON	Dane są w otwartym formacie , ale ich zrozumienie na potrzeby przetwarzania maszynowego wymaga każdorazowej analizy danych i ustalenia, bądź odnalezienia w dokumentacji, jeśli taka istnieje, znaczeń poszczególnych pól. Czy „nazwisko” oznacza samo nazwisko, czy imię i nazwisko? Czy „kod” to kod pocztowy czy terytorialny? Czy „odległość” jest podana w metrach, czy w kilometrach? Czy „1/12/2018” to pierwszy grudnia, czy dwunasty stycznia? Dane udostępniane są w niezastrzeżonym formacie.
4	★★★★	Używanie URI do identyfikacji obiektów zgodnie z RDF	Dane publikowane są w formacie umożliwiającym oznaczenie ich struktury znaczeniowej	Wejście na czwarty poziom otwartości pozwala jednoznacznie określić znaczenie udostępnianych danych. Technicznym sposobem wyrażania takiego znaczenia w sposób zrozumiały dla maszyn jest identyfikacja konkretnych właściwości danych za pomocą zrozumiałych dla maszyny URI zgodnie z modelem RDF (model opisu danych). Dane udostępniane są w niezastrzeżonym formacie.

Poziom	Oznaczenie	Charakterystyka poziomu	Dodatkowe informacje	Opis
5	★★★★★	Łączenie danych z innymi danymi za pomocą linków – budowanie kontekstu	Dane zawierają połączenia strukturalne online do innych zbiorów informacji	Piąty poziom zaś dodatkowo ułatwia przetwarzanie, jawnie wskazując relacje między danymi w formie linków. Dzięki temu możliwe jest odnajdywanie połączeń pomiędzy różnymi zbiorami danych. Dane udostępniane są w niezastrzeżonym formacie.



1.

Formaty plików

1.1. Format CSV

CSV to prosty format przechowywania danych tabelarycznych. Zastosowanie dodatkowych standardów, takich jak *Model for Tabular Data and Metadata on the Web*[4], pozwala na właściwe wyrażenie metadanych również na najwyższych poziomach otwartości. W związku z tym **format CSV jest zalecany do udostępniania otwartych danych na wszystkich poziomach otwartości**.

Szczegółowe wymagania związane z formatem CSV znajdują się w **punkcie 3** „Szczegółowe wymagania techniczne dla zasobów w formacie CSV".

1.2. Format JSON

JSON to prosty format udostępniania ustrukturyzowanych danych. Zastosowanie dodatkowych standardów, takich jak JSON-LD[5], pozwala na właściwe wyrażenie metadanych również na najwyższych poziomach otwartości. W związku z tym **format JSON jest zalecany do udostępniania otwartych danych na wszystkich poziomach otwartości**.

Szczegółowe wymagania techniczne związane z formatem JSON znajdują się w **punkcie 4** „Szczegółowe wymagania dla zasobów w formacie JSON".

1.3. Format XML

Format XML jest dopuszczalny do udostępniania otwartych danych na wszystkich poziomach otwartości. **Format ten także pozwala na właściwe wyrażenie metadanych również na najwyższych poziomach otwartości**.

Szczegółowe wymagania techniczne związane z formatem XML znajdują się w **punkcie 5** „Szczegółowe wymagania dla zasobów w formacie XML".

1.4. Format HTML

Format HTML jest otwartym formatem, pozwalającym na udostępnianie ustrukturyzowanych danych. Możliwe jest stosowanie formatu HTML na wszystkich

[4] https://www.w3.org/TR/tabular-data-model/

[5] https://json-ld.org/spec/latest/json-ld/

poziomach otwartości, przy wykorzystaniu narzędzi takich jak mikroformaty, RDFa, czy poprzez uzupełnienie go formatem JSON-LD.

Nie zaleca się stosowania formatu HTML na poziomach otwartości powyżej 3★★★.

Przetwarzanie formatu HTML umożliwia wiele narzędzi, takich jak biblioteka Beautiful Soup[6].

1.5. Format OpenDocument Spreadsheet (ODS)

Format ODS jest otwartym formatem udostępniania ustrukturyzowanych danych i w związku z tym **jest dopuszczalny do poziomu otwartości 3★★★**. Na wyższych poziomach zaleca się zastąpienie go formatem CSV lub JSON.

Możliwe jest przetworzenie zasobów do postaci zgodnej z wyższymi poziomami otwartości np. przy pomocy pakietu LibreOffice[7].

1.6. Formaty z rodziny Office Open XML (XLSX, DOCX)

Formaty Office Open XML to otwarte formaty udostępniania dokumentów (DOCX) i ustrukturyzowanych danych (XLSX) i w związku z tym przyjmuje się, że udostępnione dane w tym formacie spełniają poziom otwartości 3★★★. Ze względu na praktyczne problemy związane z przetwarzaniem plików w tych formatach, **format Office Open XML nie jest zalecany jako format udostępniania otwartych danych powyżej poziomu 2★★**.

Możliwe jest przetworzenie zasobów do postaci zgodnej z wyższymi poziomami otwartości np. przy pomocy pakietu LibreOffice.

1.7. Formaty MS Office (DOC, XLS)

Są to własnościowe formaty dokumentów, **dopuszczalne na poziomie otwartości do 2★★**.

Do arkusza danych w formacie XLS wymaga się stosowania szczegółowych wymagań dla zasobów w formacie CSV na poziomie otwartości 3★★★ określonych w punktach 4.1 i 4.2.

[6] https://www.crummy.com/software/BeautifulSoup

[7] https://pl.libreoffice.org/

Wobec tego, zaleca się zastąpienie formatu XLS formatem CSV dla uzyskania trzeciego poziomu otwartości.

Możliwe jest przetworzenie zasobów do postaci zgodnej z wyższymi poziomami otwartości np. przy pomocy pakietu LibreOffice.

1.8. Format PDF

Format PDF jest dopuszczalny wyłącznie na poziomie otwartości 1★, ponieważ co do zasady nie nadaje się do automatycznego przetwarzania.

Możliwe jest przetworzenie zasobów do postaci zgodnej z wyższymi poziomami otwartości np. przy pomocy: pakietu LibreOffice, narzędzi OCR takich jak Tesseract czy GOCR, bądź pakietów oprogramowania takich jak biblioteka pdf2text.

1.9. Formaty JPEG, PNG

Formaty graficzne są dopuszczalne jako formaty udostępniania otwartych danych wyłącznie na poziomie otwartości 1★, ponieważ co do zasady nie nadają się do automatycznego przetwarzania zamieszczonych w nich danych. Wyjątkiem jest sytuacja, w której same obrazy (a nie przedstawiona na nich zawartość, np. tekst) są udostępnianymi danymi. Wówczas poziom otwartości zależy od poziomu otwartości metadanych, którymi opatrzony jest obraz, przygotowanymi zgodnie z zaleceniami niniejszego dokumentu.

Możliwe jest przetworzenie zasobów do postaci zgodnej z wyższymi poziomami otwartości np. przy pomocy narzędzi OCR takich jak Tesseract czy GOCR.

1.10. Format dBase

Format bazy danych dBase jest dopuszczalny do poziomu otwartości 3★★★.

Możliwe jest przetworzenie zasobów do postaci zgodnej z wyższymi poziomami otwartości np. przy pomocy bibliotek do przetwarzania danych w formacie DBF, takich jak pakiet dbfpy.

1.11. Format TXT

Format TXT, pozbawiony struktury, **jest dopuszczalny na poziomie otwartości 1★**. Zależnie od faktycznej struktury danych, powinny one być udostępniane z oznaczeniem innego, właściwego typu pliku (np. CSV).

1.12. Pliki archiwów

Dopuszczalne jest udostępnianie danych w postaci skompresowanej.

Na wszystkich poziomach otwartości dopuszcza się otwarte formaty kompresji, takie jak ZIP (bez własnościowych rozszerzeń), 7z, Gzip czy Bzip2. Na poziomach otwartości do 2★★ dopuszcza się także popularne formaty własnościowe: RAR, rozszerzony ZIP.

1.13. Inne otwarte formaty

Dopuszczalne jest stosowanie innych otwartych formatów danych, jeśli jest to uzasadnione kontekstem, np. dla danych mapowych GPX, Shapefile (SHP), GML, WMS, WMTS itp.

W szczególności:

- mapy rastrowe bez georeferencji **są dopuszczalne na poziomie otwartości 1★**
- mapy rastrowe z georeferencją **są dopuszczalne na poziomie otwartości 3★★★**
- mapy wektorowe i dane przestrzenne niezaopatrzone w kontekst pozwalający na automatyczne przetworzenie do postaci zgodnej z RDF **są dopuszczalne na poziomie otwartości 3★★★**
- dane przestrzenne zaopatrzone w kontekst pozwalający na automatyczne przetworzenie do postaci zgodnej z RDF **są dopuszczalne na wszystkich poziomach otwartości**. Udostępniając takie dane, należy zachować dobre praktyki opisane w dokumencie Spatial Data on the Web Best Practices[8].

[8] https://www.w3.org/TR/sdw-bp/

2.

Formatowanie danych

Niezależnie od wyboru formatu pliku danych wymaga się stosowanie właściwego formatowania w szczególności dla danych typu: liczba, data, godzina, wartość logiczna.

Na poziomach otwartości 4★★★★ i 5★★★★★ każda właściwość ma określony przez URI[9] typ, który wyznacza oczekiwany format wartości (zob. szczegółowe opisy poniżej w punktach dot. CSV, JSON, XML). Przykładowo, jeśli mamy do czynienia z właściwością schema:datePublished, to jednoznacznie ustalony zostaje typ wartości na schema:Date, który z kolei jest datą w formacie zgodnym z normą ISO 8601[10].

Na tych poziomach wymagane jest określenie typu danych w ramach definicji właściwości.

Na poziomach otwartości do 3★★★ zaleca się również stosowanie zapisu daty i czasu w formacie zgodnym z normą ISO 8601, oraz odpowiednio podstawowych typów danych ze standardu XML Schema[11].

Oznacza to w szczególności:

- stosowanie literalnych wartości *true*, *false* jako wartości logicznych,
- stosowanie kropki dziesiętnej (a nie przecinka) w zapisie ułamków dziesiętnych, bez żadnych dodatkowych separatorów (np. oddzielających tysiące); dopuszczalny jest tzw. zapis naukowy,
- zapis dat w postaci *yyyy-mm-dd*, a łącznie dat i godzin w postaci *yyyy-mm-ddThh:mm* lub *yyyy-mm-dd hh:mm:ss* (spacja między datą a czasem).

Dane zaleca się udostępniać w możliwie najwyższym stopniu granulacji (rozdrobnienia), tzn. nie łączyć kilku danych w jednym polu.

2.1. Przykłady formatowania danych

REGON – wymaga się zapisywania identyfikatora podmiotu jako ciągu 9 lub 14 cyfr, bez spacji lub łączników.

NIP – wymaga się zapisywania numeru identyfikacji podatkowej jako ciągu 10 cyfr, bez spacji lub łączników.

Numer telefonu – w polskiej strefie numeracyjnej zaleca się zapisywanie numeru telefonu jako ciągu 9 cyfr, bez wyróżniania tzw. numeru kierunkowego miejscowości

[9] https://tools.ietf.org/html/rfc3986

[10] https://www.iso.org/iso-8601-date-and-time-format.html

[11] https://www.w3.org/TR/xmlschema/

i bez spacji lub łączników. Dopuszczalne jest stosowanie prefiksu międzynarodowego poprzedzonego znakiem plus „+”.

Tabela 2. Prawidłowe formatowanie danych

Data	Liczba	Wartość logiczna
1970-01-01	42.01	true
2018-12-31	1.05e3	false

Tabela 3. Nieprawidłowe formatowanie danych

Data	Liczba	Wartość logiczna
1/1/1970	45,01	T
1.1.1970	1,234.56	0
1 I 1970	1 234,5	NIE

2.2. Dane adresowe

Wymaga się, aby adres pocztowy był udostępniany jako szereg danych opisowych: nazwy ulicy, numeru budynku, kodu pocztowego, miejscowości – umieszczonych w odrębnych polach, a nie jako jedno pole tekstowe, zawierające wszystkie wymienione dane.

Przykład:

Nazwa	Kod PNA	Miejscowość	Cecha	Nazwa ulicy	Nr budynku	Nr lokalu	Nr telefonu
CSIOZ	00-184	Warszawa	ul.	Stanisława Dubois	5A		225970927
KPRM	00-583	Warszawa	al.	Aleje Ujazdowskie	1/3		226946000
Biuro	51-152	Wrocław	pl.	Marsz. Józefa Piłsudskiego	4	75	712558765

Przykład negatywny:

Nazwa	Adres	Nr telefonu
CSIOZ	00-184 Warszawa, ul. Dubois 5A	+48 22 597-09-27
KPRM	00-583 Warszawa, Aleje Ujazdowskie 1/3	(22) 694-60-00 (cent.)
Biuro	51-152 Wrocław, J. Piłsudskiego 4/75	71-25-58-765

Wymaga się, aby dane opisowe (tekstowe) składające się na adres były literalnie zgodne z ich zapisem w źródle ich wytworzenia, a sposób udostępniania nie może wpływać na ich wartość, tzn. nie mogą one w procesie przekazywania do miejsca publikacji ulegać zmianom.

Zbiorowym miejscem publikacji danych, takich jak nazwy ulic, po zrealizowaniu procesu uchwalenia (nazwy ulic nadawane są przez Radę Gminy) oraz ogłoszeniu przez wojewodę w wojewódzkim dzienniku urzędowym, jest rejestr TERYT. W tym rejestrze do konkretnych nazw przypisane są kody – identyfikatory ze zbiorów ULIC[12] oraz SIMC. W nim też znajdują się kody terytorialne TERC. Kody terytorialne mogą się zmieniać w zależności od aktualnego podziału terytorialnego kraju, natomiast identyfikatory miejscowości, a nawet ich części, nadawane są ostatecznie. Z uwagi na potrzeby związane z poprawną adresacją – w lokalizacją punktów adresowych – utworzono pojęcie **adresu uniwersalnego**.

Adres uniwersalny to adres zapisany jako ciąg kodów – kodu pocztowego (tzw. PNA) bez myślnika (5 cyfr), kodu terytorialnego składającego się z kodów województwa, powiatu, miejscowości (6cyfr), identyfikatora miejscowości podstawowej (7 cyfr), identyfikatora miejscowości (7 cyfr), współrzędnych geodezyjnych x, y danego punktu adresowego oraz numeru budynku – oddzielonych separatorem „|”.

Zaleca się stosowanie adresu uniwersalnego. Konstrukcja adresu uniwersalnego umożliwia jego maszynowe (automatyczne) rozkodowanie pozwalające na uzyskanie nazwy województwa, powiatu, gminy, miejscowości, nazwy ulicy oraz wyświetlenie mapy z lokalizacją danego punktu adresowego. Adres uniwersalny nie służy do odczytu przez człowieka, lecz przez system informatyczny, którego zadaniem jest wizualizacja danego punktu adresowego na mapie.

[12] http://eteryt.stat.gov.pl/eTeryt/.../pliki_pelne_struktury.aspx

Aplikacje służące do wprowadzania danych adresowych do różnego rodzaju rejestrów czy też systemów teleinformatycznych, powinny posiadać zabezpieczenia przed wprowadzaniem adresów nieistniejących. Zatem systemy te powinny umożliwiać dokonywanie wyboru adresu ze słowników rzeczywiście istniejących obiektów. Przykładem może być aplikacja służąca do weryfikacji adresów, znajdująca się na portalu danepubliczne.gov.pl[13]. Aplikacja ta wykorzystuje rozwijane listy utworzone w oparciu o słowniki TERYT oraz dane zawarte w Państwowym Rejestrze Granic. Posiada również konwerter zakodowanych adresów uniwersalnych na adresy pocztowe punktów adresowych.

Przykład: adres uniwersalny Centrum Systemów Informacyjnych Ochrony Zdrowia przy ul. Stanisława Dubois 5A w Warszawie wygląda następująco:

```
00184|146501|0918123|0918123|04337|489147.9218|636045.6562|5A|
```

Poprawne adresy uniwersalne znajdują się w kolumnie AdresCSIOZ w plikach CSV dostępnych na stronie **http://integracja.gugik.gov.pl/daneadresowe/**. W tych samych plikach, a także plikach GML[14], znajdują się szczegółowe dane adresowe, tworzone bezpośrednio w gminach. Są to dane źródłowe i najbardziej wiarygodne. Jednocześnie, fakt otwarcia danych adresowych oraz ściśle określone zasady tworzenia zapewniają interoperacyjność we wszelkich zastosowaniach.

[13] **https://danepubliczne.gov.pl/application/from_users_310esh618qtmrvnf**

[14] „Ustawa z dnia 17 maja 1989 r. - Prawo geodezyjne i kartograficzne (Dz. U. z 2010 r. Nr 193, poz. 1287) i rozporządzenie Ministra Administracji i Cyfryzacji z dnia 9 stycznia 2012 r. w sprawie ewidencji miejscowości, ulic i adresów (Dz.U. z 2012 r. poz. 125)”

3.

Szczegółowe wymagania dla zasobów w formacie CSV

3.1. Format pliku CSV

W praktyce występują pliki określane jako „CSV”, które różnią się szczegółami technicznymi. Różne są znaki nowej linii, kodowania znaków, używane są różne separatory pól: przecinek, średnik, tabulator.

Na poziomie otwartości 3 ★★★ zaleca się, natomiast na poziomie 4 ★★★★ wymaga się ścisłego stosowania formatu CSV określonego w RFC 4180[15]. W szczególności oznacza to:

- stosowanie znaków końca wiersza w formacie CRLF;
- używanie przecinka, jako separatora pól;
- używanie cudzysłowów do otaczania znaków specjalnych (takich jak przecinki czy znaki nowej linii wewnątrz pól);
- stosowanie domyślnego kodowania znaków UTF-8, bądź prawidłowe oznaczenie kodowania w nagłówku Content-type jeśli stosowane jest inne kodowanie znaków;

Przykład:

```
Content-type: text/csv

title,datePublished
„Do gościa (Gościu, tak jakoś począł, już do końca
czytaj...)”,2007-09-07
```

Przykład negatywny:

```
Content-type: text/plain

title;datePublished
Do gościa (Gościu, tak jakoś począł, już do końca
czytaj...);2007-09-07
```

W powyższym przykładzie: nagłówek Content-type jest nieprawidłowy, pola są rozdzielone średnikiem zamiast przecinkiem, z kolei tekst zawierający przecinek nie jest ujęty w cudzysłów.

[15] https://tools.ietf.org/html/rfc4180

3.2. CSV na poziomie otwartości 3★★★

Dla każdego pliku w formacie CSV wymagany jest dokładnie jeden wiersz nagłówka, opisujący zawartość tabeli w sposób możliwie najbardziej zrozumiały dla użytkownika. Jak zostało powiedziane wcześniej: na tym poziomie kluczowa jest rola człowieka, który musi nadać sens danym przeznaczonym do dalszego automatycznego przetwarzania.

Wymaga się, aby wiersz nagłówka zawierał wyłącznie nagłówki poszczególnych kolumn, a wszystkie pozostałe wiersze zawierały wyłącznie dane kolejnych rekordów. Niedopuszczalne jest, przykładowo, wyodrębnianie fragmentu arkusza jako „legenda".

Niedopuszczalna jest obecność kolumn bez nagłówka lub z pustym (składającym się tylko z białych znaków) nagłówkiem. Niedopuszczalne są także nagłówki powtarzające się w różnych kolumnach.

Wymaga się, aby wszystkie wiersze tabeli, łącznie z nagłówkiem, miały taką samą liczbę komórek.

Niedopuszczalna jest obecność całkowicie pustych kolumn bądź wierszy.

Wymaga się, aby wszystkie wiersze (poza wierszem nagłówka) były traktowane jako niezależne od siebie. Niedopuszczalne jest w szczególności pozostawianie w tabeli wolnych miejsc w celu uniknięcia powtórzeń danej wartości w kolejnych wierszach. Jeśli dana wartość jest identyczna dla kolejnych rekordów, to nadal musi być osobno określona dla każdego rekordu.

Przykład:

```
datePublished,views,change
2018-01-01,1234,15
```

Przykład negatywny:

```
datePublished,views,,change
2018-01-01,1234,20%,15,10%
```

W powyższym przykładzie pojawiła się dodatkowa kolumna bez nagłówka, oraz dodatkowa wartość pozbawiona nagłówka.

3.3. CSV na poziomie otwartości 4 ★★★★

Na tym poziomie otwartości kluczowe jest zapewnienie możliwości maszynowego odczytu wszystkich danych z uwzględnieniem ich kontekstu semantycznego. W tym celu wymagane jest stosowanie standardu Model for Tabular Data and Metadata on the Web, pozwalającego dołączyć do plików CSV kontekst zgodny z JSON-LD.

Na poziomie 4★★★★ wymagane jest zapewnienie kontekstu obejmującego wszystkie kolumny tabeli CSV.

Przykład:

```
Link: <metadata.json>; rel="describedBy";
type="application/csvm+json"

datePublished,views,change
2018-01-01,1234,15
```

Gdzie plik metadata.json to prawidłowy plik CSVM[16].

Przykład negatywny:

```
datePublished,views,change
2018-01-01,1234,15
```

W powyższym przykładzie brak kontekstu nie pozwala zinterpretować danych.

3.4. CSV na poziomie otwartości 5 ★★★★★

Dodanie wymaganych na tym poziomie otwartości hiperlinków do innych zasobów odbywa się w sposób analogiczny do udostępniania danych w formacie JSON-LD: poprzez zdefiniowanie właściwych kolumn, reprezentujących linki, jako elementów typu hydra:Link.

[16] zob. przykłady w **https://www.w3.org/TR/tabular-data-primer/**.

Przykład:

Poniższe dane w CSV stają się zrozumiałe jako link:

```
kolejka
/wizyty
```

jeśli w kontekście odpowiednio zdefiniujemy pole „kolejka” jako hydra:Link.

```
...
{
   „titles”: „kolejka”,
   „propertyUrl”: „http://danepubliczne.gov.pl/kolejka”,
},
...
{
   „@id”: „http://danepubliczne.gov.pl/kolejka”
   „link”: „hydra:Link”,
}
...
```

Przykład negatywny:

```
kolejka
/wizyty
```

Bez kontekstu definiującego kolumnę jako link, nie można zidentyfikować wartości jako link.

4.

Szczegółowe wymagania dla zasobów w formacie JSON

Format JSON jest zalecany do udostępniania otwartych danych na wszystkich poziomach otwartości. Jest to prosty format, nadający się do przetwarzania ogólnie dostępnymi narzędziami znajdującymi się w standardowych bibliotekach wielu języków programowania.

Doprecyzowania wymaga udostępnianie w tym formacie dużych kolekcji (stronicowanie), określanie semantyki na wyższych poziomach otwartości, a także określanie dodatkowych funkcji związanych z udostępnianiem danych w API, takich jak sortowanie, filtrowanie, wyszukiwanie i wybieranie podzbiorów zwracanych pól z wyników zapytania.

4.1. JSON na poziomie otwartości 3 ★★★

Na tym poziomie wymagane jest stosowanie prawidłowego, tj. zgodnego z RFC 4627[17], formatu JSON.

Ponadto, najważniejsze jest zorganizowanie danych, nazw właściwości, pól służących do nawigacji między stronami kolekcji, a także mechanizmów do wyszukiwania i filtrowania, w sposób jak najbardziej zrozumiały dla użytkownika – programisty, który będzie miał za zadanie skorzystać z udostępnionych danych.

Zaleca się przy tym organizowanie danych w strukturę, która pozwoli w łatwy sposób rozszerzyć ją o semantyczny kontekst, wymagany na wyższych poziomach otwartości danych.

Podobnie, zaleca się używanie pól do stronicowania kolekcji w sposób pozwalający na rozszerzenie ich do postaci zgodnej z hydra:PartialCollectionView[18], wymaganej na kolejnym poziomie otwartości.

Dopuszczalne jest także użycie formatu linków do nawigacji ze standardu JSON API 1.0[19] lub późniejszym, ewentualnie hiperlinków zgodnych ze standardem HAL[20].

Dodatkowe funkcje nawigacyjne, takie jak wyszukiwanie, filtrowanie, sortowanie i wybieranie podzbiorów pól z wyników zaleca się wprowadzać w sposób określony w standardzie JSON API. Wymaga się, aby funkcje takie, jeśli są obecne, były dokładnie opisane w dokumentacji API, określonej w Standardzie API.

[17] https://tools.ietf.org/html/rfc4627

[18] http://www.hydra-cg.com/spec/latest/core/#collections

[19] http://jsonapi.org/format/

[20] https://tools.ietf.org/html/draft-kelly-json-hal-08

Przykład:

```
Content-type: application/json

{
   „totalItems”: „100”,
   „member”: [
      {
         „name”: „Punkt Obsługi”,
         „address”: {
            „postalCode”: „00-001”,
            „addressLocality”: „Warszawa”,
            „streetAddress”: „Marszałkowska 1”
         },
         „url”: „https://punkt-obslugi.example.com”,
         „wizyty”: „/wizyty”
      }
   ],
   „view”: {
      „first”: „/punkty?page=1”,
      „previous”: „/punkty?page=2”,
      „next”: „/punkty?page=4”,
      „last”: „/punkty?page=20”
   }
}
```

Przykład negatywny:

```
Content-type: text/plain

{
   „properties”: {
      // „Dataset”: {
      // ...
      // }
   }
}
```

W powyższym przykładzie: dane są udostępnione z nieprawidłowym nagłówkiem *Content-type*, zaś struktura pliku jest naruszona przez pozostawione przez programistę nieprawidłowe znaki komentarza.

4.2. JSON na poziomie otwartości 4 ★★★★

Na tym poziomie otwartości kluczowe jest zapewnienie możliwości maszynowego odczytu wszystkich danych, bez potrzeby ingerencji człowieka, który wstępnie nadałby znaczenie poszczególnym właściwościom danych bądź linkom nawigacyjnym. Komputer, na podstawie odczytanego *Content-type*, powinien mieć odpowiednią ilość informacji, by móc dotrzeć do wszystkich zasobów udostępnionych w API.

Na poziomie 4★★★★ wymagane jest stosowanie standardu JSON-LD i opatrzenie danych pełnym kontekstem semantycznym. Co oznacza mapowanie wszystkich właściwości udostępnianych danych odpowiednim URI w modelu RDF.

Do udostępniania kolekcji obiektów wymagane jest stosowanie typu Collection[21] ze standardu Hydra. Jako mechanizm stronicowania tych kolekcji wymagane jest stosowanie mechanizmu PartialCollectionView z tego samego standardu.

Wymagane jest również stosowanie typu hydra:Link[22] do opisywania linków pomiędzy punktami końcowymi (*endpoints*) API.

W szczególności wymaga się, aby pojedynczy punkt startowy API umożliwiał dotarcie do wszystkich punktów końcowych (*endpoints*) API i do wszystkich danych udostępnionych w API poprzez automatyczne nawigowanie po kolekcjach hydra:Collection i linkach hydra:Link.

Mechanizmy służące do sortowania, filtrowania, wyszukiwania oraz wybierania pól mogą być udokumentowane za pomocą klasy hydra:IriTemplate oraz hydra:Operation[23]. Dopuszcza się pozostawienie opisu tych mechanizmów na poziomie dokumentacji API, podobnie jak na poziomie 3★★★ korzystanie z tych mechanizmów jest dopuszczalne, ale nie wpływa na dostęp do danych udostępnionych w API.

[21] http://www.hydra-cg.com/spec/latest/core/#collections.

[22] http://www.hydra-cg.com/spec/latest/core/#adding-affordances-to-representations.

[23] http://www.hydra-cg.com/spec/latest/core/#templated-links i http://www.hydra-cg.com/spec/latest/core/#documenting-a-web-api.

Aby ułatwić poruszanie się po API oraz usprawnić walidację danych, zalecane jest określenie struktury udostępnianych danych za pomocą API poprzez dedykowany schemat JSON Schema.

Przykład:

```
{
   „@context”: [
      „http://www.w3.org/ns/hydra/context.jsonld”,
      „http://schema.org”
   ],

   „@id”: „https://api.gov.pl/punkty”,
   „@type”: „Collection”,
   „totalItems”: „100”,
   „member”: [
      {
         „@id”: „/punkty/punkt-obslugi”,
         „name”: „Punkt Obsługi”,
         „address”: {
            „postalCode”: „00-001”,
            „addressLocality”: „Warszawa”,
            „streetAddress”: „Marszałkowska 1”
         }
      }
   ],
   „view”: {
         „@id”: „http://api.example.com/an-issue/
            comments?page=3”,
         „@type”: „PartialCollectionView”,
         „first”: „/punkty?page=1”,
         „previous”: „/punkty?page=2”,
         „next”: „/punkty?page=4”,
         „last”: „/punkty?page=20”
   }
}
```

Przykład negatywny:

```
{
   „@context”: „http://www.w3.org/ns/hydra/context.jsonld”,

   „@id”: „https://api.gov.pl/punkty”,
   „member”: [
      {
         „@id”: „/punkty/punkt-obslugi”,
         „name”: „Punkt Obsługi”,
         „address”: {
            „postalCode”: „00-001”,
            „addressLocality”: „Warszawa”,
            „streetAddress”: „Marszałkowska 1”
          }
      }
   ],
   „first”: „/punkty?page=1”,
   „previous”: „/punkty?page=2”,
   „next”: „/punkty?page=4”,
   „last”: „/punkty?page=20”
}
```

W powyższym przykładzie: część danych jest pozbawiona kontekstu nadającego im znaczenie, zaś linki do nawigacji nie są prawidłowo oznaczone jako obiekt hydra:PartialCollectionView.

4.3. JSON na poziomie otwartości 5 ★★★★★

Ten poziom otwartości danych publicznych wymaga uzupełnienia danych hiperlinkami do powiązanych zasobów znajdujących się w zbiorach danych w innych lokalizacjach.

Aby dodać takie linki, wymaga się dodania elementów typu hydra:Link innych niż tylko linki służące do nawigacji.

Przykład poniżej zawiera kilka elementów będących linkami.

Linkami są pola obiektu *view: first*, *previous*, *next* i *last*, ponieważ *context* związany ze standardem Hydra przypisuje je do typu hydra:Link. Są to linki służące do nawigacji po kolekcji.

Linkiem jest pole *wizyty*, ponieważ jest tak wprost zdefiniowane w przykładzie. Kontekst przypisuje polu *wizyty* URI *„http://danepubliczne.gov.pl/kolejka”* oraz typ wskazujący, że wartością tego pola jest również URI. Drugi element grafu (na dole przykładu) definiuje, że *http://danepubliczne.gov.pl/kolejka* jest typu hydra:Link.

Przykład:

```
[
   {
      „@context”: [
         „http://www.w3.org/ns/hydra/context.jsonld”,
         „http://schema.org”,
         {
            „wizyty”: {
               „@id”: „http://danepubliczne.gov.pl/kolejka”,
               „@type”: „@id”
            }
         }
      ],

      „@id”: „https://api.gov.pl/punkty”,
      „@type”: „Collection”,
      „totalItems”: „100”,
      „member”: [
         {
            „@id”: „/punkty/punkt-obslugi”,
            „name”: „Punkt Obsługi”,
            „address”: {
               „postalCode”: „00-001”,
               „addressLocality”: „Warszawa”,
               „streetAddress”: „Marszałkowska 1”
            },
            „url”: „https://punkt-obslugi.example.com”,
            „wizyty”: „/wizyty”
         }
      ],
```

```
      „view”: {
         „@id”: „http://api.example.com/an-issue/
         comments?page=3”,
         „@type”: „PartialCollectionView”,
         „first”: „/punkty?page=1”,
         „previous”: „/punkty?page=2”,
         „next”: „/punkty?page=4”,
         „last”: „/punkty?page=20”
      }
   },

   {
      „@context”: „http://www.w3.org/ns/hydra/context.
jsonld”,
      „@id”: „http://danepubliczne.gov.pl/kolejka”,
      „@type”: „Link”
   }
]
```

Przykład negatywny

```
{
   „@context”: [
      „http://www.w3.org/ns/hydra/context.jsonld”,
      „http://schema.org”,
      {
         „wizyty”: „http://danepubliczne.gov.pl/kolejka”
      }
   ],

   „@id”: „https://api.gov.pl/punkty”,
   „@type”: „Collection”,
   „totalItems”: „100”,
   „member”: [
```

```
{
        „@id”: „/punkty/punkt-obslugi”,
        „name”: „Punkt Obsługi”,
        „address”: {
           „postalCode”: „00-001”,
           „addressLocality”: „Warszawa”,
           „streetAddress”: „Marszałkowska 1”
        },
        „wizyty”: „/wizyty”
     }
  ],
  „view”: {
     „@id”: „http://api.example.com/an-issue/
comments?page=3”,
     „@type”: „PartialCollectionView”,
     „first”: „/punkty?page=1”,
     „previous”: „/punkty?page=2”,
     „next”: „/punkty?page=4”,
     „last”: „/punkty?page=20”
  }
}
```

W powyższym przykładzie element *wizyty* jest przyporządkowany właściwości zidentyfikowanej przez URI, ale jego wartość nie jest określona jako URI, zaś sama właściwość nie ma nadanego typu hydra:Link, co uniemożliwia automatyczne wykrycie powiązania między zasobami.

5.

Szczegółowe wymagania dla zasobów w formacie XML

XML to uniwersalny język znaczników, pozwalający na reprezentowanie i przechowywanie różnych danych w strukturalizowany sposób. XML pozwala na identyfikowanie znaczników jako URI, co umożliwia stosowanie go do udostępniania danych na wszystkich poziomach otwartości. Obecnie format XML w wielu kontekstach zastępowany jest prostszymi formatami. Dopuszcza się stosowanie formatu XML, jednocześnie zalecając stosowanie formatów JSON lub CSV.

Dane w formacie XML udostępnia się w sposób analogiczny do udostępniania w formacie JSON-LD.

5.1. XML na poziomie otwartości 3 ★★★

Na tym poziomie wymagane jest stosowanie dowolnej poprawnej struktury XML, w której właściwości obiektów reprezentowane są przez elementy XML.

Przykład:

```
Content-type: application/xml

<punkt>
   <adres>Warszawa, Marszałkowska 1</adres>
   <url>http://example.com</url>
</punkt>
```

Przykład negatywny

```
Content-type: text/plain

<punkt>
   <adres>Warszawa<br>Marszałkowska 1</adres>
   <url>http://example.com</url>
</punkt>
```

W powyższym przykładzie: XML ma błędny nagłówek *Content-type* oraz nieprawidłową strukturę.

5.2. XML na poziomie otwartości 4 ★★★ i 5 ★★★★

Wymaga się, aby na tych poziomach była określona semantyka (identyfikacja przez URI) wszystkich elementów za pomocą mechanizmu XML Namespaces[24] i dedykowanych dla tych danych schem. Poszczególne punkty API muszą być ze sobą połączone hiperlinkami utworzonymi przy użyciu standardu XLink.

Podobnie jak w przypadku formatu JSON – wymagane jest zapewnienie, że do wszystkich danych udostępnionych za pośrednictwem danego API można dotrzeć, korzystając z pojedynczego punktu startowego i hiperlinków łączących ze sobą poszczególne punkty API.

Przykład:

```
Content-type: application/xml

<punkt xmlns=”http://danepubliczne.gov.pl”
xmlns:schema=”https://schema.org/”>
   <schema:address>
      <schema:postalCode>00-001</schema:postalCode>
      <schema:addressLocality>Warszawa</schema:postalCode>
      <schema:streetAddress>Marszałkowska 1</
      schema:postalCode>
   </schema:address>
</punkt>
```

Przykład negatywny:

```
Content-type: application/xml

<punkt>
   <address>
      <postalCode>00-001</postalCode>
      <addressLocality>Warszawa</postalCode>
      <streetAddress>Marszałkowska 1</postalCode>
   </address>
</punkt>
```

W powyższym przykładzie: brak przestrzeni nazw dla użytych elementów.

[24] https://www.w3.org/TR/xml-names/

6.

Podsumowanie

Poniższa tabela zawiera zalecenia dotyczące udostępniania danych na określonych poziomach otwartości. Jak wskazano we Wstępie, nie zaleca się publikowania danych na pierwszym poziomie otwartości.

Tabela 4. Zalecenia

Rozwiązanie/ standard	**Poziom 2** ★★	**Poziom 3** ★★★	**Poziom 4** ★★★★	**Poziom 5** ★★★★★
Udostępnianie przez API	–	zalecane	zalecane	zalecane
Format CSV	zalecany	zalecany	zalecany	zalecany
Format JSON	zalecany	zalecany	zalecany	zalecany
Format XML	dopuszczalny	dopuszczalny	dopuszczalny	dopuszczalny
Format HTML	dopuszczalny	dopuszczalny	niezalecany	niezalecany
Format ODS	dopuszczalne	dopuszczalne	niedopuszczalne	niedopuszczalne
Formaty Office Open XML (XLSX, DOCX)	dopuszczalne	niedopuszczalne	niedopuszczalne	niedopuszczalne
Formaty DOC, XLS, RTF	dopuszczalne	niedopuszczalne	niedopuszczalne	niedopuszczalne
Format PDF	niedopuszczalny	niedopuszczalny	niedopuszczalny	niedopuszczalny
Formaty JPEG, PNG	niedopuszczalne	niedopuszczalne	niedopuszczalne	niedopuszczalne
Format dBase	dopuszczalny	dopuszczalny	niedopuszczalny	niedopuszczalny
Format TXT	niedopuszczalny	niedopuszczalny	niedopuszczalny	niedopuszczalny
Formaty ZIP, 7z, Gzip, Bzip2	dopuszczalne	dopuszczalne	dopuszczalne	dopuszczalne
Własnościowe rozszerzenia ZIP	dopuszczalne	niedopuszczalne	niedopuszczalne	niedopuszczalne
Format RAR	dopuszczalny	niedopuszczalny	niedopuszczalny	niedopuszczalny
Inne otwarte formaty, np. mapowe	dopuszczalne	dopuszczalne	dopuszczalne	dopuszczalne

Rozwiązanie/ standard	Poziom 2 ★★	Poziom 3 ★★★	Poziom 4 ★★★★	Poziom 5 ★★★★★
Zapis dat w formacie ISO 8601	zalecany	zalecany	–	–
Typy danych z XML Schema	zalecane	zalecane	–	–
Typy danych określone przez URI	–	–	wymagane	wymagane
Stronicowanie przez hydra:Partial CollectionView	zalecane	zalecane	wymagane	wymagane
Stronicowanie przez linki JSON API	dopuszczalne	dopuszczalne	niedopuszczalne	niedopuszczalne
Stronicowanie przez linki HAL	dopuszczalne	dopuszczalne	niedopuszczalne	niedopuszczalne
Filtrowanie, wyszukiwanie zgodnie z JSON API	zalecane	zalecane	zalecane	zalecane
Filtrowanie, wyszukiwanie przez hydra:IriTemplate oraz hydra:Operation	dopuszczalne	dopuszczalne	dopuszczalne	dopuszczalne
JSON Schema	–	–	zalecane	zalecane
Hyperlinki hydra:Link	–	–	–	wymagane
Format CSV zgodny z RFC 4180	zalecane	zalecane	wymagane	wymagane

Rozwiązanie/ standard	Poziom 2 ★★	Poziom 3 ★★★	Poziom 4 ★★★★	Poziom 5 ★★★★★
Nagłówek w pliku CSV	zalecany	zalecany	wymagany	wymagany
Model for Tabular Data and Metadata on the Web	–	–	wymagane	wymagane
XML Namespaces	–	–	wymagane	wymagane
XLink	–	–	wymagane	wymagane