Data Catalog Vocabulary (DCAT)

OER4SDI

Open Educational Resources for Spatial Data Infrastructures

In this tutorial you will learn about the Data Catalog Vocabulary (DCAT), the concepts on which it is based and how it is used in practice. You will create and validate DCAT metadata yourself and use SPARQL to run queries on the Rhein-Kreis Neuss Open Data Catalog.

James Ondieki, Albert Remke
Institute for Geoinformatics, University of Münster

August 2024

ein Kooperationsprojekt, empfohlen durch:

DIGITALE HOCHSCHULE NRW

gefördert durch:

Ministry of Culture and Science of the State of North Rhine-Westphalia

1. Overview

In this tutorial you will learn about the Data Catalog Vocabulary (DCAT), the concepts on which it is based and how it is used in practice. You will create and validate DCAT metadata yourself and use SPARQL to run queries on the Rhein-Kreis Neuss Open Data Catalog.

In our practical exercises we will use:

- RDForms, an online RDF tool for creating DCAT metadata
- Apache Jena Fuseki Server, a triple store that enables the management and querying of DCAT metadata using the SPARQL query language
- Docker to manage our Apache Fuseki runtime environment

The tutorial is structured as follows:

This tutorial is designed for students and professionals who want to improve their understanding of Spatial Information Infrastructures, and in particular, how DCAT promotes the reusability and interoperability of spatial data. We assume that you have basic knowledge about spatial data and spatial data services (for example Web Map Services (WMS) and Web Feature Services (WFS)). If you don't, you can still proceed with the module as we will not be going into the technical details or usage of the spatial data and data services, but just how they are described using DCAT. You will need about 90 minutes to work through this tutorial.

This tutorial has been developed in the context of the OER4SDI Project at the Institute for Geoinformatics, University of Münster. Authors are James Ondieki and Albert Remke. The latest version of this tutorial is always available on GitHub. We hope you will use GitHub issues to provide feedback and suggest improvements.

The OER4SDI project has been recommended by the Digital University NRW and is funded by the Ministry of Culture and Science NRW.

2. Background

2.1 Data Sharing and Metadata

Sharing of data and data services between individuals, organizations, research institutions or governments requires the provision of metadata. Metadata refers to any information that describes an item or an information resource. Just like a library catalog card provides data about a book or a label on a can of fruit juice provides metadata about the fruit juice, a metadata description provides metadata about a dataset.

A book’s metadata is useful when a reader wants to find out more about the book. When there are many books available, the reader can use a set of library catalog cards, to get an overview of the books available in the library, and search them by category or author. The set of library cards can be seen as aggregated metadata about the books.

The same concept applies to open data or information resources in general. Providers of data and data services create, manage and publish the metadata that accompanies the information resource or service they provide. The metadata can be aggregated so that interested users can get a good overview of existing data and services that can be used for various purposes.

Metadata of information resources can be categorized according to their purpose:

1. **Supporting discovery and identification**

   This is metadata that supports the discovery and identification of information resources. It includes information such as title, abstract, author, keywords, publication date, language, ISBN, DOI, etc.

2. **Supporting data understanding**

   This is metadata that supports a deeper understanding of the data and thus helps to decide on the usability of the data for a specific purpose. It includes information such as the conceptual data model, data elements and relations, reference systems, information on the data quality, links to comprehensive documentation and contacts for further questions.etc.

3. **Supporting data access and use**

   This is metadata that supports data access and use. It includes information such as access points, terms for access and use (license), API definition, encoding, etc.

Library catalog cards have a standardized content structure. For example, they might contain the book title, the author's name, the category and the year in which the book was published. In the same way, descriptions of datasets and services from various providers must also be standardized in order to be published in a data portal. The use of standardized conceptual data models supports the interoperability of metadata.

To address the need for metadata standardization, a number of conceptual data models and encodings have been developed to promote metadata interoperability. These include:

1. **ISO 19115**

   This is an international standard for describing metadata of geographic information. It defines a comprehensive set of metadata elements covering various aspects of geographic information including the identification, the geographical extent, the quality, the spatial and temporal aspects, the content, the spatial reference, distribution, and other properties of digital geographic data and services.

2. **Dublin Core**

   It provides a simple and standardized set of metadata elements for describing digital resources. It includes elements such as title, creator, subject, description, publisher, date, format, and identifier.

3. **Data Catalog Vocabulary (DCAT)**

   DCAT is a W3C recommendation for describing datasets on the web. DCAT can be used to describe datasets and data services in a catalog using standard terms that facilitate the consumption and aggregation of metadata from multiple catalogs.

These conceptual models play a role in facilitating the interoperability of metadata between different systems and applications. The interoperability of metadata helps to ensure that data can be discovered, shared, and understood across different domains.

The focus of this tutorial is on DCAT, a metadata standard that is based on RDF (Resource Description Framework). But before diving into DCAT, let us have a short overview of RDF and how it relates to DCAT.

2.2 Resource Description Framework - RDF

The Resource Description Framework (RDF) is a standard of the World Wide Web Consortium (W3C) for representing information about resources on the web (i.e. metadata).

RDF represents data as triples, consisting of a subject, a predicate, and an object, which together form a statement about resources:

- Subject is a resource being described by the triple.
- Predicate describes the relationship between the subject and the object.
- Object is a resource or a property value that is related to the Subject.

For example:

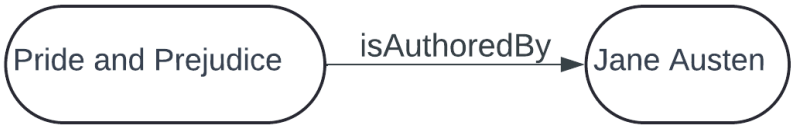

Fig. 1: An RDF graph containing one triple

The book “Pride and Prejudice” is a famous book authored by “Jane Austen”. As depicted in Figure 1, the book itself is the *subject* of the triple statement while the author is the *object*. The *predicate* is “isAuthoredBy” which shows the relationship between the subject and the object.

In this example,

- the subject may be represented by the URI ‘http://example.com/books/PrideAndPrejudice’,
- the predicate may be represented by ‘http://purl.org/dc/terms/creator’ and the
- object may be represented by ‘http://example.com/authors/JaneAusten’.

Sets of RDF triples form directed graphs. In our book example, you can imagine further triples describing further properties and resources related to the book and to Jane Austen.

RDF uses URIs (Uniform Resource Identifiers) to unambiguously represent the semantics of subjects, predicates and objects. Only for objects it is allowed to use literals, e.g. a text string

that represents a title or an abstract. Using URIs ensures that data can be precisely referenced and interlinked across diverse datasets and applications.

RDF triples can be stored in so-called triple stores such as Apache Jena or Virtuoso. These are databases providing native support for managing and querying RDF data using the query language SPARQL.

The World Wide Web Consortium (W3C) maintains the standards for RDF, including specifications for the SPARQL query language and various RDF encodings such Turtle, RDF-XML, and JSON-LD.

RDF can be used to create so-called vocabularies, defining classes of objects, their properties and relationships as well as a set of terms and their meaning. The friends-of-a-friend vocabulary for example is about describing people, their relationships and activities. Dublin Core and DCAT are managed vocabularies that are used to describe information resources.

2.3 Data Catalog Vocabulary (DCAT)

DCAT was created to enable data publishers to describe datasets and services in a standardized manner. It is an RDF based vocabulary designed to support the interoperability between various data catalogs on the web. DCAT is a W3C recommendation and is more general-purpose. It is used for describing any type of dataset, not just geospatial ones.

The namespace for DCAT is http://www.w3.org/ns/dcat#. DCAT defines just a minimal set of classes and properties of its own and extensively re-uses terms from existing vocabularies, particularly Dublin Core.

Figure 2 shows an overview on the classes and properties of the DCAT vocabulary.

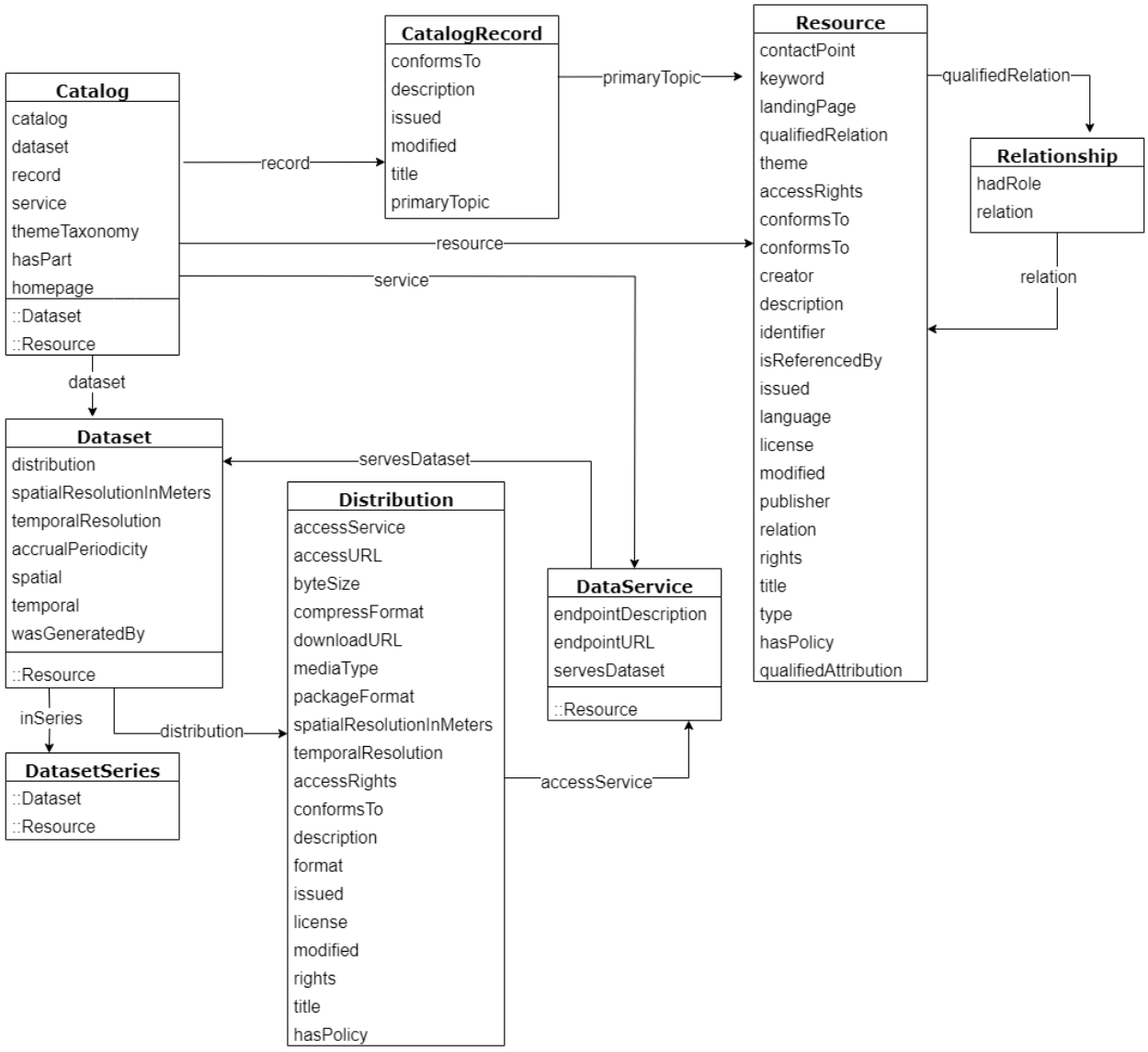

Fig. 2: Overview of DCAT model, showing the classes of resources that can be members of a Catalog, and the relationships between them.

The DCAT vocabulary defines the following classes:

- **dcat:Resource** - This abstract class represents a dataset, a data service or any other resource that may be described by a metadata record in a catalog. It is the parent class of dcat:Dataset, dcat:DataService and dcat:Catalog.
- **dcat:Dataset** - This is the core class in DCAT, representing a dataset. It typically describes a collection of data that is published or curated by an organization or individual, and is available for download or access in one or more representations.

- **dcat:DatasetSeries** - This is a dataset that represents a collection of datasets that are published separately, but share some characteristics that group them.
- **dcat:Distribution** - Distribution refers to a specific representation of a dataset. This class represents an accessible form of a dataset, such as a downloadable file, an API endpoint, or a web service through which the dataset can be accessed.
- **dcat:DataService** - This class represents a service that provides access to data, such as a web service that allows querying or retrieving data from a dataset. A data service typically provides selection, extraction, combination, processing or transformation operations over datasets that might be hosted locally or remote to the service.
- **dcat:Catalog** - This class represents a catalog, which is a collection of metadata describing various resources such as datasets and services. It serves as a container for organizing and describing multiple datasets within a single catalog.
- **dcat:CatalogRecord** - This represents a record or entry in a catalog, describing the registration of a single 'dcat:Resource'. This class is optional and it exists for catalogs where a distinction is made between metadata about a dataset or service and metadata about the entry in the catalog about the dataset or service. For example, the publication date property of the dataset reflects the date when the information was originally made available by the publishing agency, while the publication date of the catalog record is the date when the dataset was added to the catalog.

The properties shown in each class are both the unique properties of that particular class and those properties inherited or acquired from the "::super-class". The classes "dcat:Dataset", "dcat:DataService", and "dcat:Catalog" all inherit from the "dcat:Resource" class. The class "dcat:DatasetSeries" inherits properties from both the "dcat:Resource" and "dcat:Dataset" classes.

2.4 DCAT Profiles

A number of user communities have developed DCAT Profiles to adapt DCAT to their specific needs. This is useful as soon as additional classes and properties are required to describe the resources of the community. A DCAT Profile is a specification of a data catalog that adds additional constraints to DCAT. A data catalog that conforms to the profile also conforms to DCAT.

Additional constraints in a profile may include classes and properties for additional metadata fields not covered in the DCAT vocabulary specification, sub-classes and sub-properties of the standard DCAT classes and properties or controlled vocabularies or IRI sets as acceptable values for the various properties. In some cases, a profile extends one of the DCAT profiles themselves, by adding classes and properties for metadata fields not covered in the referenced DCAT profile.

DCAT Profiles maintain compatibility with DCAT by:

a) adhering to mandatory DCAT elements
b) adopting selected optional DCAT elements
c) adding complementary elements

In the following, some DCAT profiles are listed that are particularly relevant in Europe and Germany:

- **DCAT-AP**: This is a profile developed by the European Commission for data portals in Europe. It is the de facto standard of metadata interchange across European data portals.
- **GeoDCAT-AP**: Developed by the European Commission, this profile is a geospatial extension to the DCAT-AP profile. It closes the gaps in ISO19115. It defines a standard way of describing geospatial datasets, data series and services across data portals in Europe. The profile does not intend to replace the INSPIRE regulations on Metadata nor the technical guidelines but intends to support the data exchange between INSPIRE metadata catalogs in Europe
- **DCAT-AP.de** : This is the German National Profile of DCAT-AP. It is the official metadata exchange format for open data from public authorities in Germany. It adds some elements and constraints relevant to metadata exchange between the national GovData catalog and catalogs of various administrative levels. Other countries including Italy, Norway, Sweden, and Belgium have their own national profiles of the DCAT-AP profile.

3. Practical Exercises

In the following chapters we will work with well established open data portals and we will present technologies that can be used to create, validate and analyze DCAT metadata. Each section provides you with practical tasks that you can use to deepen and test your knowledge of DCAT.

3.1 Practical Example of DCAT Metadata

In our first exercise, we will take a closer look at the DCAT metadata of a publicly accessible data set.

The Rhine district of Neuss maintains a dataset that shows the locations of various wind turbines in the region. The dataset is published by the State Office for Nature, Environment and Consumer Protection of North Rhine-Westphalia and is available in various formats.

The dataset can be found at the Open-Data Portal for Rhein-Kreis Neuss district, https://opendata.rhein-kreis-neuss.de/explore/dataset/rhein-kreis-neuss-windenergieanlagen/information/. Figure 3 shows the page of the open data portal for the Rhein-Kreis Neuss with metadata for the wind turbine dataset.

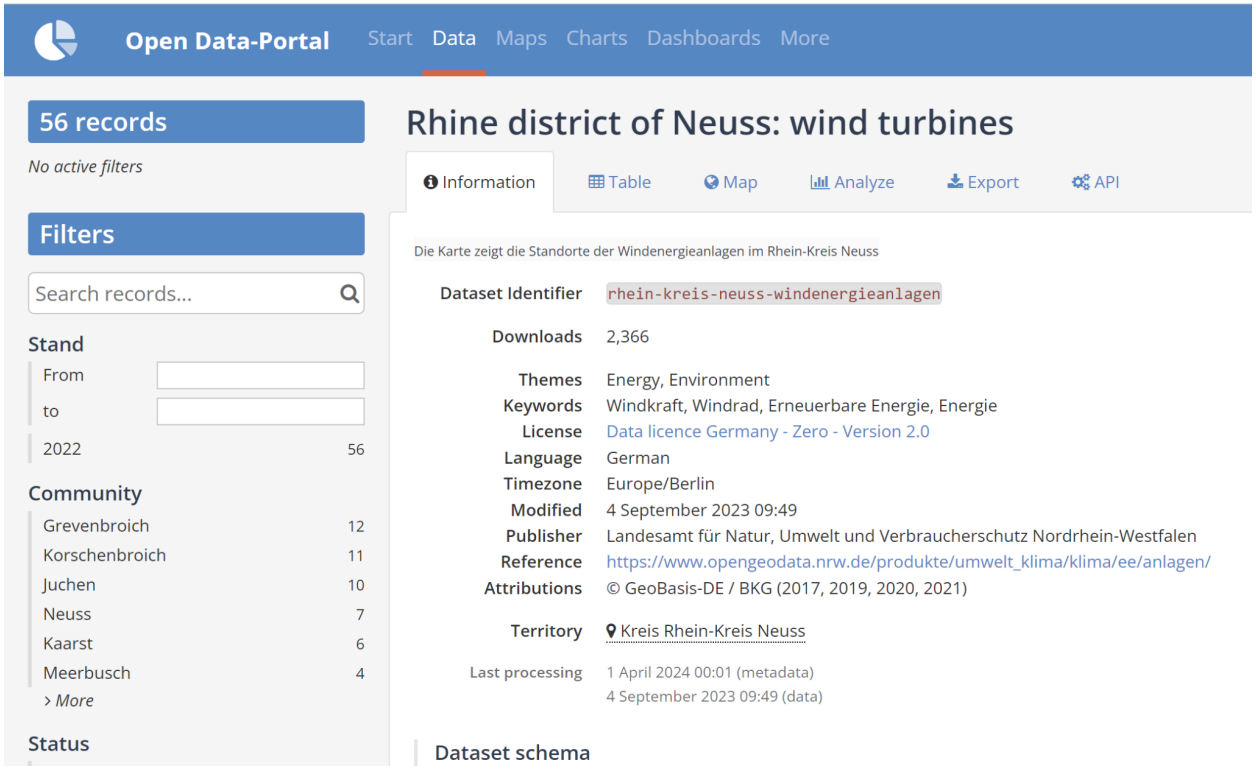

Fig. 3: Open data portal page for the Rhein-Kreis Neuss District showing the wind turbines dataset

The various formats or distributions for the dataset can be viewed by clicking the “Export” tab at the top of the web page.

The state government of North Rhine-Westphalia (NRW) operates and maintains another open data portal that provides users with access to open data from the public sector, https://open.nrw/. The open data portal aggregates and publishes metadata from the catalogs of lower administrative regions in NRW, making them visible and available on the open data portal of the state government. This includes the catalog from Rhein-Kreis Neuss district and its metadata on the wind turbines dataset (see Figure 4).

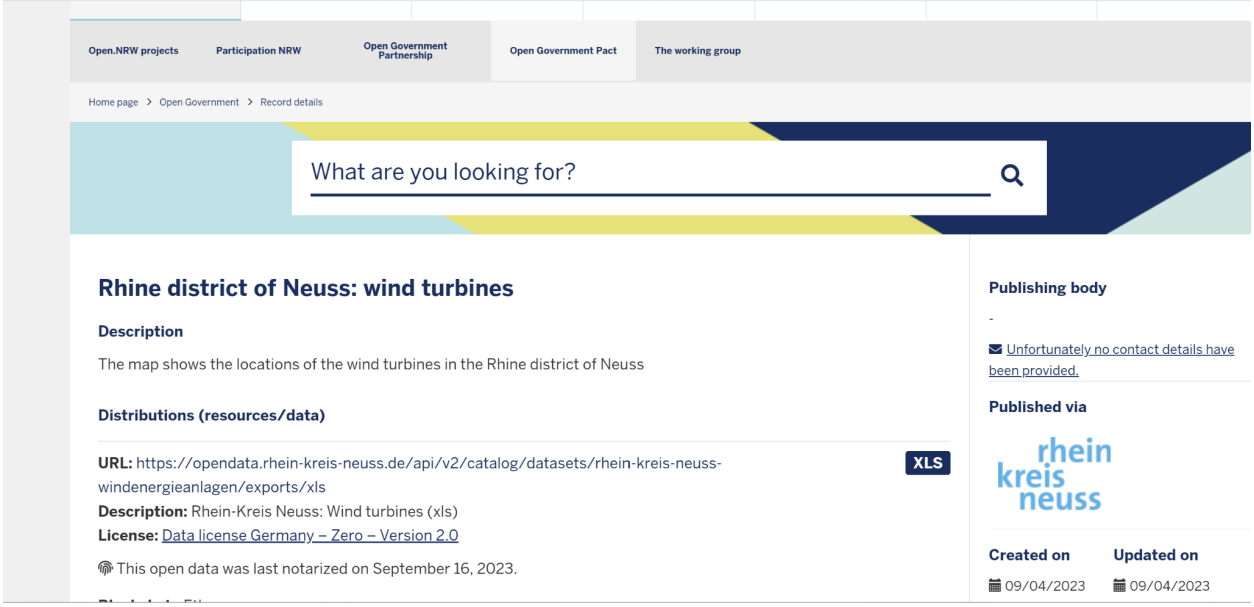

Fig. 4: Open data portal page for the state government of North Rhine-Westphalia showing the wind turbines dataset from the Rhein-Kreis Neuss District.

At the next administrative level, the federal government of Germany operates and maintains an open data portal, https://www.govdata.de/. In addition to publishing the federal government's data, it aggregates and publishes metadata from the catalogs of the various federal states. As shown in Figure 5, the metadata of the Rhein-Kreis Neuss wind turbine dataset can also be found via the federal government's Open Data portal.

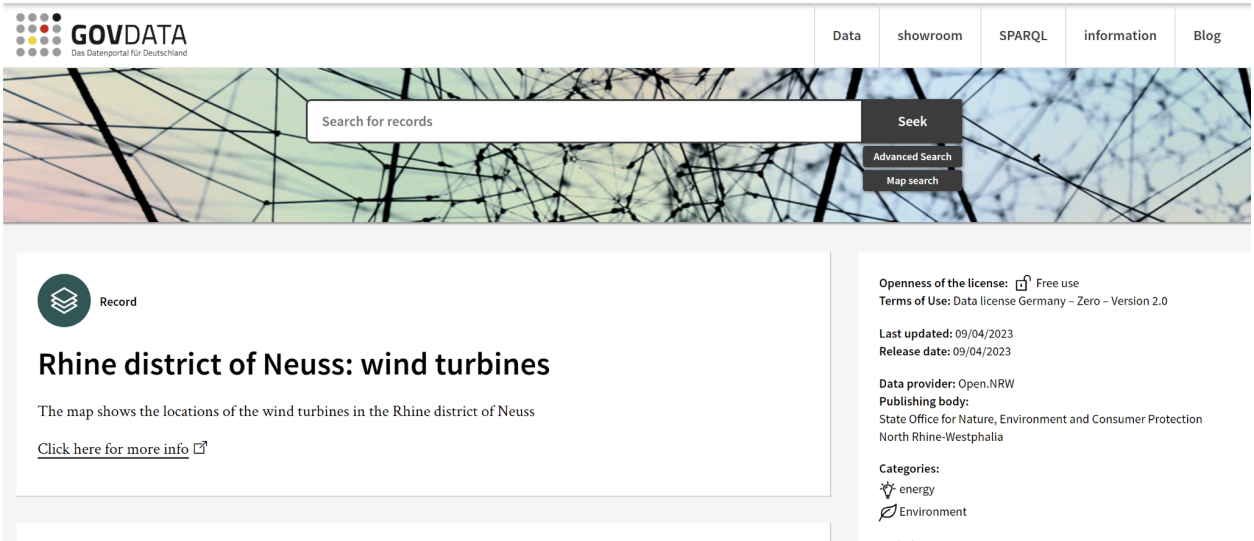

Fig. 5: National open data portal page showing the wind turbines dataset from the Rhein-Kreis Neuss district

Task 3.1-A

Download the DCAT metadata file (format RDF/XML) for the wind turbines dataset using the link https://open.nrw/dcatap/dataset/rhein-kreis-neuss-windenergieanlagen-ne.rdf. In case of problems you'll find a copy of the dataset in the /data folder of this tutorial's GitHub repository.
Use a text editor such as Notepad++, VS Code or Notepad to open the file and familiarize yourself with its contents.

Figure 6 shows an excerpt from the DCAT metadata set.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
  xmlns:dct="http://purl.org/dc/terms/"
  xmlns:dcat="http://www.w3.org/ns/dcat#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dcatap="http://data.europa.eu/r5r/"
  xmlns:dcatde="http://dcat-ap.de/def/dcatde/"
>
  <dcat:Dataset rdf:about="https://opendata.rhein-kreis-neuss.de/api/v2/catalog/datasets/rhein-kreis-neuss-windenergieanlagen">
    <dcat:keyword>Energie</dcat:keyword>
    <dcat:distribution>
      <dcat:Distribution rdf:about="https://opendata.rhein-kreis-neuss.de/api/v2/catalog/datasets/rhein-kreis-neuss-windenergieanlagen-json">
        <dct:title>json</dct:title>
        <dct:modified rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2023-09-04T07:49:36.309000+00:00</dct:modified>
        <dct:description>Rhein-Kreis Neuss: Windenergieanlagen (json)</dct:description>
        <dct:language rdf:resource="http://publications.europa.eu/resource/authority/language/DEU"/>
        <dct:format rdf:resource="http://publications.europa.eu/resource/authority/file-type/JSON"/>
        <dcatap:availability rdf:resource="http://publications.europa.eu/resource/authority/planned-availability/AVAILABLE"/>
        <dcat:accessURL rdf:resource=
        "https://opendata.rhein-kreis-neuss.de/api/v2/catalog/datasets/rhein-kreis-neuss-windenergieanlagen/exports/json"/>
        <dct:license rdf:resource="http://dcat-ap.de/def/licenses/dl-zero-de/2.0"/>
        <dct:issued rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2023-09-04T07:49:36.309000+00:00</dct:issued>
      </dcat:Distribution>
    </dcat:distribution>
    <dcat:distribution>
      <dcat:Distribution rdf:about="https://opendata.rhein-kreis-neuss.de/api/v2/catalog/datasets/rhein-kreis-neuss-windenergieanlagen-ov2">
        <dct:title>ov2</dct:title>
        <dct:issued rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2023-09-04T07:49:36.309000+00:00</dct:issued>
        <dcatap:availability rdf:resource="http://publications.europa.eu/resource/authority/planned-availability/AVAILABLE"/>
        <dct:language rdf:resource="http://publications.europa.eu/resource/authority/language/DEU"/>
        <dct:license rdf:resource="http://dcat-ap.de/def/licenses/dl-zero-de/2.0"/>
        <dct:format rdf:resource="http://publications.europa.eu/resource/authority/file-type/OV2"/>
        <dct:modified rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2023-09-04T07:49:36.309000+00:00</dct:modified>
        <dct:description>Rhein-Kreis Neuss: Windenergieanlagen (ov2)</dct:description>
        <dcat:accessURL rdf:resource=
        "https://opendata.rhein-kreis-neuss.de/api/v2/catalog/datasets/rhein-kreis-neuss-windenergieanlagen/exports/ov2"/>
      </dcat:Distribution>
    </dcat:distribution>
```

Fig. 6: DCAT metadata elements for the wind turbines dataset

At the beginning of the RDF file, you will find the prefix definitions of the namespaces of various vocabularies used within the file. As already mentioned, DCAT makes extensive use of terms from existing vocabularies and defines further classes and properties of its own. In this case, the metadata uses terms from:

a) **xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"** : This defines the prefix of the RDF namespace (Resource Description Framework).

b) **xmlns:foaf="http://xmlns.com/foaf/0.1/"** : This specifies the namespace for the FOAF (Friend of a Friend) vocabulary, which is used for describing people, their relationships, and their activities.

c) **xmlns:dct="http://purl.org/dc/terms/"** : This specifies the namespace for the Dublin Core Terms (DCT) vocabulary, a widely-used standard for describing metadata elements such as title, description, creator, and publisher.

d) **xmlns:dcat="http://www.w3.org/ns/dcat#"** : This specifies the namespace for the DCAT (Data Catalog Vocabulary) standard, which is used for describing data catalogs, datasets, and distributions.

e) **xmlns:dcatap="http://data.europa.eu/r5r/"** : This specifies the namespace for DCAT-AP (Application Profile), a European Union profile of the DCAT standard designed to facilitate interoperability and data sharing across EU member states.

f) **xmlns:dcatde="http://dcat-ap.de/def/dcatde/"** : This specifies the namespace for DCAT-AP.DE, a German application profile of the DCAT standard designed to facilitate interoperability and data sharing within Germany.

The **dcat:Dataset** element contains metadata properties and other elements that describe the wind turbines dataset. It includes various metadata such as title, description, landing page URL, identifier, keywords, issuing and modification dates, language, and details of the publisher.
The **dcat:distribution** element outlines the various distribution formats that are available for the dataset. Each distribution format has properties that describe its title, description, license, access URL, language, format, issuing date, modification dates and availability. The available distribution formats for this dataset are:

a) KML (Keyhole Markup Language)
b) JSON-LD (JavaScript Object Notation for Linked Data)
c) GEOJSON (Geographic JSON)
d) OV2 (TomTom POI format)
e) GPX (GPS Exchange format)
f) Parquet (columnar storage file format)
g) XLS (Microsoft Excel Spreadsheet)
h) SHP (Shapefile format)
i) RDF/XML (Resource Description Framework in XML format)
j) JSONL (JSON Lines format)
k) N3 (Notation3 format)

All of these distribution formats are available for download through the provided URLs, and the dataset is released under a license permitting open data use. As we can see, The dataset is easily accessible through various URLs hosted on the Open-Data Portal for the Rhein-Kreis Neuss District. The dataset's metadata plays a crucial role, as it is aggregated and shared with larger data portals such as open.nrw. This federated approach to metadata search significantly enhances discoverability, enabling efficient and comprehensive data access across different

platforms. Users can search for wind turbine data on the larger data portals, filter the results to the Rhein-Kreis Neuss District dataset and use the available access urls to access the data.

Task 3.1-B

Look at the DCAT dataset in Figure 6, which is encoded as RDF/XML. Give one example of a subject-predicate-object statement that describes a resource.

Subject:	
Predicate:	
Object:	

Task 3.1-C

According to the Dublin Core vocabulary, what is the definition of the terms “dct:issued” and “dct:modified”?

dct:issued	
dct:modified	

Task 3.1-D

Explore the metadata of another resource:

- Select a dataset on a topic of your choice in the open Data Catalog of the Rhein-Kreis Neuss,
- identify the same dataset in the open data catalog open.nrw,
- download the DCAT metadata of this resource from open.nrw,
- open the DCAT metadata file with a text editor and identify the various distribution formats of this dataset.

3.2 Creating, Editing and Visualizing DCAT metadata with RDForms

There are many software tools that enable the creation, management and processing of metadata according to various metadata standards. These include ArcGIS Pro, ArcGIS Online, QGIS, CKAN, RDForms and many specialized tools.

For this practical exercise, we will use RDForms as a metadata editor because it is easy to use and offers DCAT support. RDForms is an online tool that provides DCAT-compliant metadata forms for editing metadata on datasets, data catalogs and other data-related resources. The tool has an editor tab where the metadata can be entered, a presentation tab where the results are displayed, and an RDF tab that displays the resulting metadata in both RDF and JSON-LD format. The editor is controlled by the RDForms library, which uses templates to generate HTML forms for editing RDF. The RDForms DCAT editor (see Figure 7) can be accessed via the link https://rdforms.com/editors/dcat/#editor .

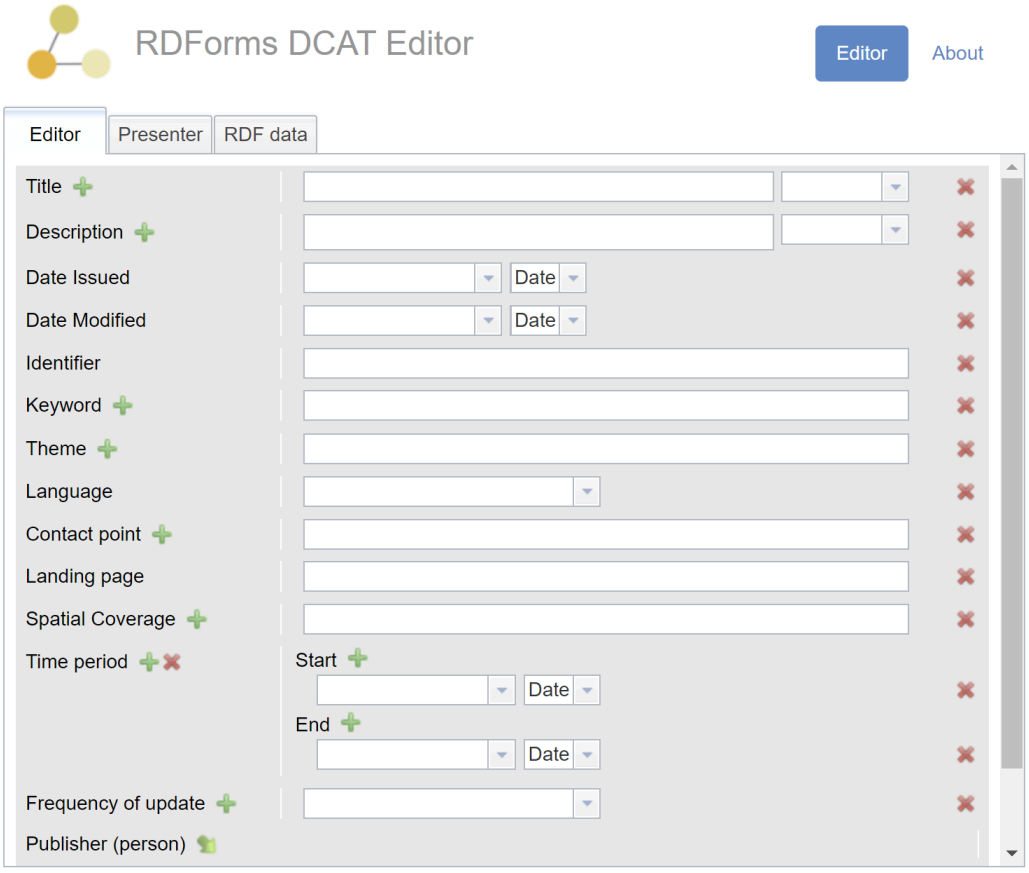

Fig. 7: RDForms landing page.

Task 3.2-A - Creating a DCAT metadata record

Imagine that you work at the planning department of the city of Muenster. The department has collected data on buildings in the city and wants to make that data accessible and findable on Muenster's open data portal. The dataset is available in different distribution forms, including shapefile download, GeoJSON and as a web map service (WMS). Your task is to create metadata for the buildings dataset and its distributions.

Use RDForms and the explanations below to create a meaningful metadata set.

1. **Title** - This refers to the name given to the resource. In our case, the title assigned to the buildings dataset. The dropdown field next to the title offers a selection for the title's language. The title can be repeated in a different language by clicking the green "+" sign next to the "Title" field, and selecting the appropriate language. This is helpful for multi-language metadata catalogs that integrate our metadata.
   Choose an appropriate Title and corresponding Language selection for the Building dataset.
2. **Description** - This is text that offers more information about the resource. The dropdown field next to the description offers a selection for the language used in the description. The description can also be repeated in a different language by clicking the green "+" sign next to the "Description" and selecting the appropriate language. Enter a short description that describes the building, and choose the appropriate language.
3. **Date Issued** - This refers to the date when the resource was formally issued or made available, for example, through publication. The value of the date should be encoded using the ISO 8601 date and time format. However, RDForms provides a simple date selection input control, and implicitly converts the selected date to the required format. Select an arbitrary date that represents when the dataset was issued.
4. **Date Modified** - Datasets that are made available may undergo some changes, modifications and updates. The date modified refers to the most recent date when the dataset was updated or modified. The value may be null if the dataset has never been updated or if the last date of updating the database is not known. Just like the date issued property, RDForms provides a simple date selection input control, and implicitly converts the selected date to the required ISO 8601 date and time format. Select an arbitrary date when the dataset might have been modified. Ideally, the date modified should be more recent than the date issued.
5. **Identifier** - This refers to a text string that is assigned to the resource, that uniquely identifies the resource or dataset. The identifier is a text string that is assigned to the resource to provide an unambiguous reference within a particular context.
6. **Keyword** - A keyword is a specific term or phrase that describes a particular aspect of the dataset, such as a concept, entity, or attribute. Keywords are often used to facilitate the search and filtering of datasets Multiple keywords can be assigned to the same resource, by clicking the green "+" sign. For our buildings dataset, we could add the keywords "Building", "Property", "Real Estate", "Urban

Planning" etc. You can also consider adding keywords that are related to the attributes of the buildings dataset, for example: building type (residential, commercial, industrial), architectural style, accessibility features etc.

7. **Theme** - A theme is a broader category or topic that a dataset belongs to, which encompasses multiple related keywords. Themes provide a higher-level context for understanding the dataset's content and relevance. The theme property refers to a concept from a controlled vocabulary such as a classification system or taxonomy that categorizes datasets. DCAT often uses themes from established vocabularies like the INSPIRE theme codes[1]. Unlike the keyword property, the theme is expected to have URIs rather than simple strings. This ensures that themes are standardized, making it easier to interlink datasets across different catalogs and improve interoperability. Multiple themes can be assigned to the resource being described by clicking the "+" sign.
8. **Language** - The language property is used to specify the language(s) used in the metadata, including the dataset's title, description, keywords, and other metadata elements. It also identifies the language used in the textual values of the dataset's distribution. This will help users search for datasets in their preferred language. RDForms provides a dropdown selection control for selecting the language used in the dataset. Choose an appropriate language that matches the language used in the dataset's metadata.
9. **Contact Point** - This property is used to provide contact information for the dataset's maintainer, curator, or owner. This information is essential for users who want to learn more about the dataset, request access, or report issues. Let us assume that you will be the contact point for the dataset. Input your name in the text box provided by RDForms.
10. **Landing Page** -This property is used to provide a URL that links to a page that provides more information about the dataset, such as a dataset description, documentation, or access instructions. Ideally, the web page should provide a brief summary of the dataset including its scope, purpose, content, information on how to access the dataset, including any necessary authentication or authorization procedures. The provided value should have the structure of a valid URL. Input a sample URL that will lead users to the landing page. You can use a dummy website URL such as 'http://example.com/muenster-buildings/'
11. **Spatial Coverage** - This property is used to describe the geographic boundaries or spatial coverage of the dataset. This information is essential for users who want to understand the dataset's spatial scope and relevance or find datasets that cover a particular region of interest. The spatial coverage can be provided as

[1] https://inspire.ec.europa.eu/codelist. Use this INSPIRE vocabulary to find/filter out relevant theme(s) to our buildings dataset. Click on the theme and copy its URI.

the Bounding Box (BBOX) coordinates that define the spatial extent of the dataset. It may also be expressed as a URI that identifies a place or region from a standard geospatial vocabulary or service like GeoNames[2]. For this task, we are going to provide a sample URI link.

12. **Time Period** - The time period property is the temporal period that the dataset covers and is represented by the start date and end date values. The start date represents the beginning of the time period that is covered by the dataset while the end date represents the end of the period that is covered by the dataset. This information can help users understand the dataset's content and relevance. RDForms provides simple date selection input controls for the start date and end date, and implicitly converts the selected dates to the required ISO 8601 date and time format. Choose arbitrary start and end dates for the Buildings dataset.
13. **Frequency of Update** - This represents the time interval in which the dataset is updated as a whole. RDForms provides a selection dropdown with various values for selection, for example daily, monthly, quarterly, annual, half-yearly etc. This information is important to users of the dataset, especially if the phenomenon represented by the dataset is expected to change frequently. Choose an appropriate frequency of update from the dropdown.
14. **Publisher (Person)** - The publisher property is used to identify the entity responsible for publishing the dataset. This is typically used to provide attribution and credit to the entity responsible for creating and publishing the dataset. The entity in this scenario is a person. RDForms provides separate text boxes for inputting the details of the person, including the first name, family name, and personal mailbox details. These details can be used by users of the dataset to contact the publisher, incase of inquiries regarding the dataset. We are going to skip this section and instead provide the Publisher (Organization) details described in the next step.
15. **Publisher (Organization)** - This is used to identify the organization that is responsible for creating, maintaining and publishing the dataset. Such organizations include government agencies, research institutions, universities, or other organizations that produce and publish datasets. This information can be used to provide attribution and credit to the organization and it enhances findability of datasets published by the same organization. RDForms provide separate text boxes for inputting the name of the organization, a link to the homepage of the organization and mailbox details of the organization. These details can be used by users to contact the organization that is responsible for creating and publishing the dataset, incase of inquiries. Lets provide some arbitrary details of the planning department that made the dataset available.

[2] https://www.geonames.org/ Using this tool, search for Muenster or the city of interest. Click on the city and copy ist URI from the address bar of the web browser. For example, the URI for Munich is https://www.geonames.org/2867714/

16. **Distribution** - The distribution property is used to provide information about how to access and retrieve the dataset, making it easier for users to find and use the dataset. RDForms provides various text boxes for inputting the distribution information.
    These include:
    - Title of the distribution
    - Description of the distribution
    - The date that the distribution was issued
    - The date that the distribution was modified
    - The license information
    - The rights statement
    - Access URL of the dataset distribution
    - Download URL of the dataset distribution
    - The format of that distribution
    - The size in bytes of the distribution.

A dataset can be available in multiple distribution formats, including CSV, shapefile, GeoJSON etc. Lets assume that our buildings dataset is available for download in shapefile and GeoJSON format, and can also be accessed as a Web Map Service (WMS). Input the details of the various distributions by clicking the “+” sign next to the “Distribution” property name.
Note that the WMS distribution has no download URL, as it is a web service that is meant to be accessed and used in other applications using the provided Access URL.

The RDForms Editor should now contain input values for all metadata elements of the building’s dataset. The resulting DCAT metadata set can be viewed by selecting the *RDF Data* tab. You can switch between XML/RDF and JSON-LD representation. We are going to use the JSON-LD format to validate our metadata.

3.3 Validating DCAT Metadata

Having created some DCAT metadata of the Buildings dataset, we are now going to validate it. This is to ensure that our metadata conforms to DCAT, and that it can be seamlessly integrated in other data portals. We are going to use the **DCAT-AP.de validator**, which is a tool developed to validate the DCAT metadata of the German Profile. Remember, metadata that conforms to a DCAT Profile, also conforms to the DCAT standard. The validator tool can be found at https://www.itb.ec.europa.eu/shacl/dcat-ap.de/upload. Figure 8 shows the landing page of the validator tool.

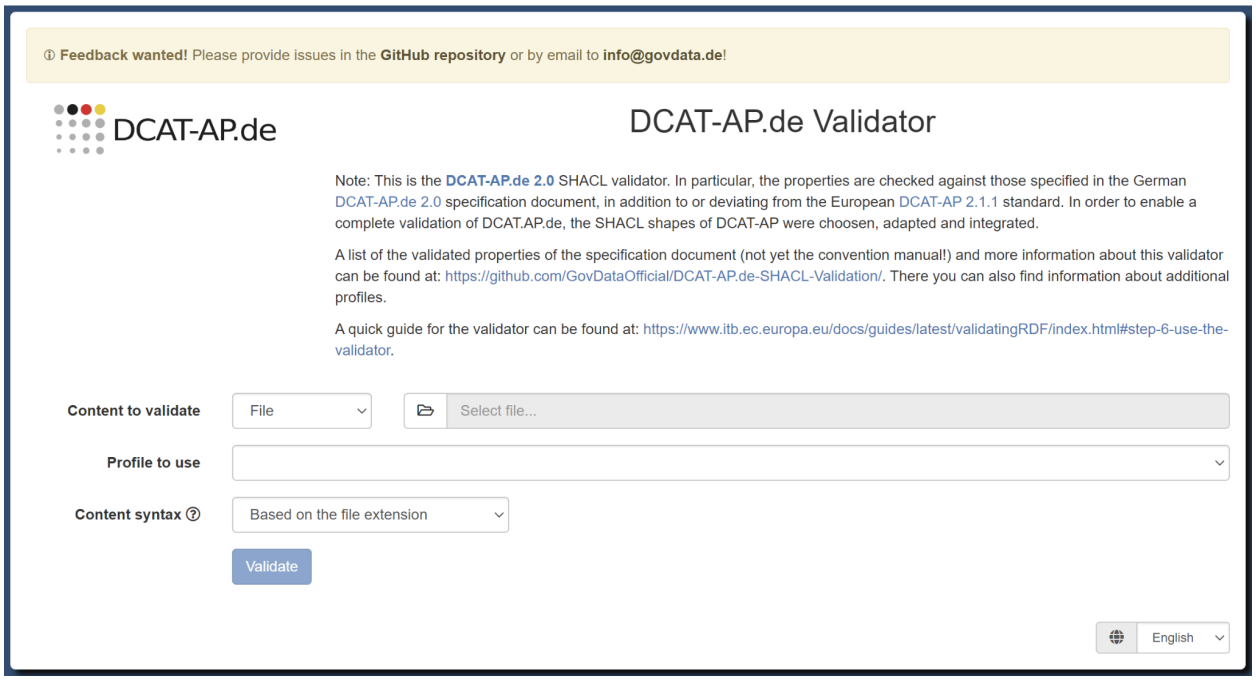

Fig. 8: Landing page of the DCAT-AP.de validator tool.

The language has been switched to English using the button on the bottom right corner.

Task 3.3-A

Open theDCAT-AP.de validator tool and change the *Content to validate* option to “Direct Input” which means that you can paste the generated metadata from RDForms to the validator tool.

The tool can validate different formats of DCAT metadata. Select “JSON-LD” as *Content syntax* option and “DCAT-AP.de Specification (2023-01-23)” as *Profile to use*.

Copy and paste the RDForms JSON-LD metadata into the *Content to validate* section and activate the validation.

The data from RDForms should be error-free. Check the validation by making some changes to the metadata in the *Content for validation* section and reactivating the validation.

Figure 9 shows the results of validating the metadata using the DCAT-AP.de validator tool.

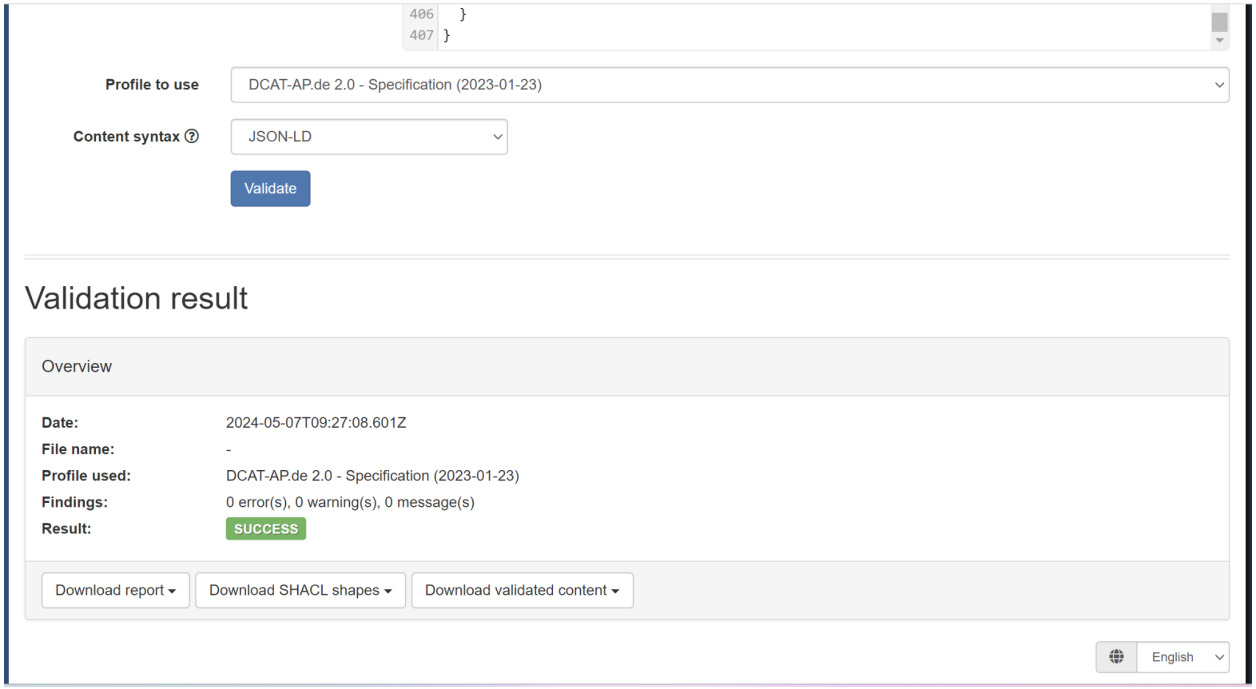

Fig. 9: Results of validating the RDForms metadata using the DCAT-AP.de validator tool.

3.4 Running SPARQL Queries on DCAT metadata.

In this section, you will learn how to:

1. Set up an RDF store using Apache Jena Fuseki with Docker.
2. Load DCAT metadata into the RDF store.
3. Execute SPARQL queries to retrieve and manipulate metadata.

By the end of this tutorial, you should have a functional understanding of DCAT, RDF, and SPARQL, and be able to apply these concepts to manage and query data catalogs.

3.4.1 Overview on the relevant Technologies

As you have learned already, RDF is a standard model for data interchange on the web. It allows data to be expressed as triples (subject, predicate, object) and is the underlying format used for representing metadata in DCAT. Built on RDF, DCAT employs RDF schema to define classes and properties that describe datasets and their related entities. DCAT extends RDF to provide a structured vocabulary for cataloging datasets, making them discoverable and interoperable.

- **SPARQL (SPARQL Protocol and RDF Query Language)**

  SPARQL is a powerful query language and protocol used to retrieve and manipulate data stored in RDF format. It allows users to write complex queries to access specific data points, filter data, and combine results from multiple datasets. SPARQL supports data querying and updating, making it a versatile tool for working with RDF data.
  As a query language for RDF data, SPARQL allows users to retrieve and manipulate the metadata expressed in DCAT. It enables querying across RDF datasets to extract meaningful information, filter results, and perform various analytical operations.

- **Apache Jena Fuseki**

  Apache Jena Fuseki is an open-source triple store for handling RDF data, providing support for reading, writing, and querying RDF using SPARQL. It offers a robust and scalable solution for managing semantic web data, supporting various RDF serialization formats and providing a convenient web-based interface for running SPARQL queries.
  Fuseki will be used as the RDF store and SPARQL endpoint where we will load DCAT metadata and run their SPARQL queries. It serves as the backend for storing and retrieving data, making it a central component of this tutorial.

- **Docker**

  Docker is a platform that allows you to package applications and their dependencies into lightweight, portable containers that can run consistently across different computing environments. Docker simplifies the deployment process, ensuring that applications work seamlessly across various infrastructures.
  Docker helps in setting up Apache Jena Fuseki in a standardized and reproducible manner. Using Docker makes the setup process straightforward, ensuring a consistent working environment regardless of the operating system or infrastructure being used.

3.4.2 Installing the Software and Metadata

For installing Apache Jena Fuseki with Docker and uploading the metadata catalog follow this workflow step by step:

1. **Install Docker**
   If you're completely new to Docker, don't worry! Follow these steps to get everything up and running. Visit the Docker website and download the Docker Desktop installer for your operating system. Run the installer which will guide you through the installation

process. Once the installation is complete, please start up the Docker Desktop application and make sure that it is running.

2. **Pull and Run the Apache Jena Fuseki Docker Image**
   We are going to pull a docker image for the Apache Jena Fuseki Server. This will be used to instantiate a running container with the server.

   Open a terminal or command prompt on your computer. On Windows, you can use PowerShell or Command Prompt. On Mac or Linux, open the Terminal application. In the terminal, run the following command to download and run the Apache Jena Fuseki Docker image:

```
docker run -p 3030:3030 -e ADMIN_PASSWORD=admin123 stain/jena-fuseki:5.1.0
```

   Fig. 10: Docker command for instantiating a docker container based on an image of Apache Jena Fuseki

   Running the above command in the terminal or powershell will make docker pull or download the stain/jena-fuseki docker image if it is not locally available, before running it. The specific docker image is version 5.1.0 . We set the password for the admin interface to be ‘admin123’. Running or executing a docker image results in docker container which can be accessed and used. After the process is complete, the docker container with Apache Jena Fuseki server will be available and accessible through http://localhost:3030/. Figure 11 shows the results of running the docker command on the terminal.

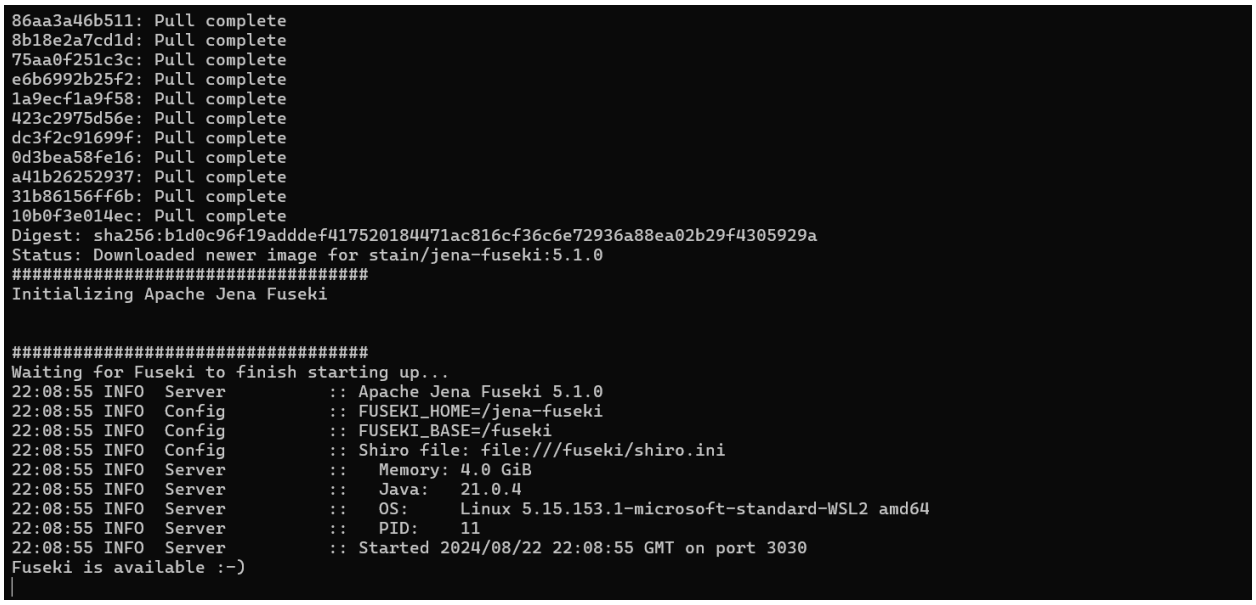

Fig. 11: Running the docker command on the terminal.

3. Access the Fuseki Admin UI

Open any browser of your choice and go to http://localhost:3030/. This opens the Apache Jena Fuseki management interface. Log into the server using 'admin' as the username and 'admin123' as the admin password. Figure 12 shows the Apache Jena Fuseki interface.

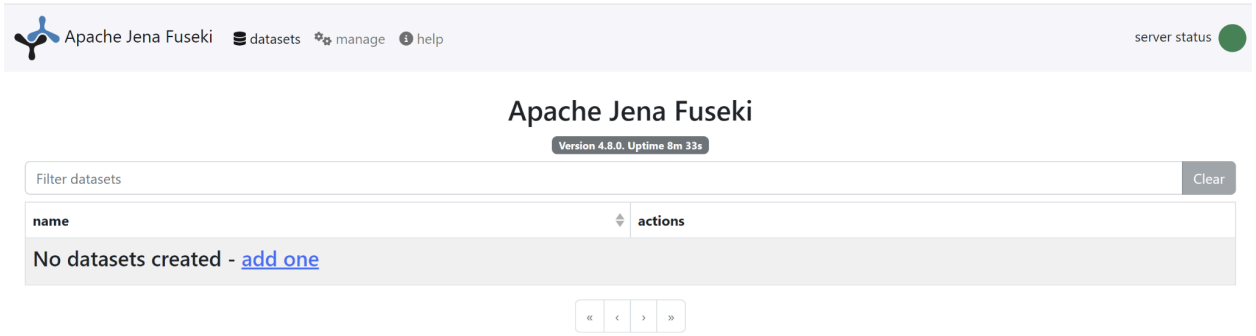

Fig. 12: Apache Jena Fuseki interface.

At the moment, our server is empty and has no datasets or metadata that can be queried. We are going to load some metadata in the next step.

4. Load DCAT Metadata into Fuseki

We are going to use the metadata for the open data catalog of the Rhein-Kreis Neuss district for this section. The catalog contains the description of various datasets that are

published or managed by the district. For simplicity, the metadata for the whole catalog can be found at https://github.com/oer4sdi/OER-DCAT/blob/main/data/Rhein-Kreis-Neuss_OpenDataCatalog.rdf . The link will open the Github repository that contains the catalog metadata file. Click the download icon to download the file, which is in *.rdf* format.

To load DCAT metadata into our server, we need to create a dataset. Click the **'add one'** option to open the page for adding new datasets. Enter 'Rhein-Kreis-Neuss_OpenDataCatalog' as the name of the dataset. Select **'Persistent'** as the dataset type. This ensures that the dataset that we create is not lost when we stop or restart our docker container.

Click the **'Create Dataset'** button to save the changes. Figure 13 is a visualization of this step.

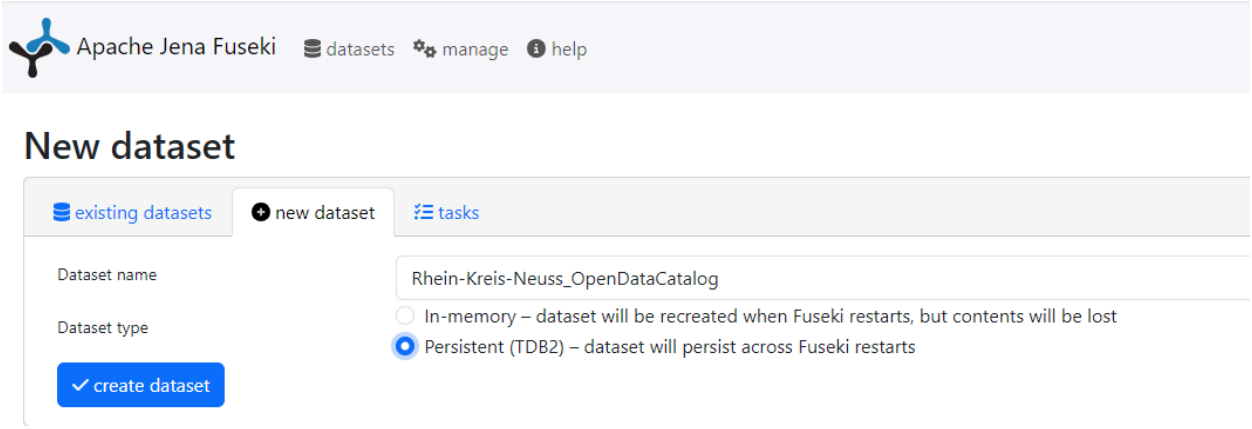

Fig. 13: Adding the open data catalog of the Rhein-Kreis Neuss district to the server

We can now add the DCAT metadata to the dataset that we created. In the Fuseki admin UI, choose the Rhein-Kreis-Neuss_OpenDataCatalog dataset that we have just created. Navigate to the **"Add data"** tab. Click the **'Select files'** option and choose the metadata file that we downloaded.
Click the **'upload now'** button under the **'actions'** tab to finish the process. Figure 14 shows the server interface after uploading the metadata. The status button turns to Green, which indicates that our metadata has been uploaded successfully. As can be seen, 92015 triples have been uploaded.

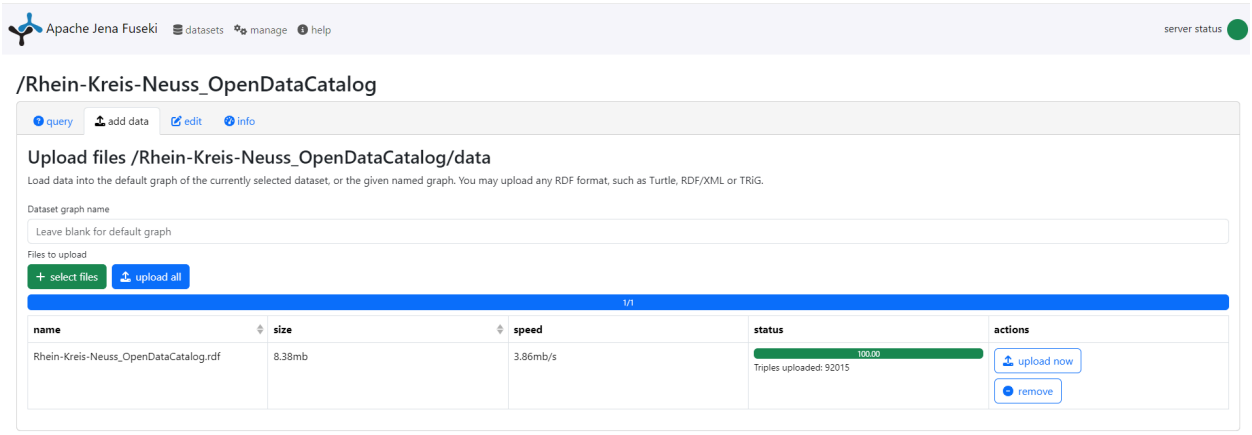

Fig. 14: Uploading the metadata to the server.

3.4.3 Query the DCAT Metadata using SPARQL

We are now going to see some sample SPARQL queries that we can use to interact with and extract some useful information from the DCAT metadata that we just uploaded. You don't need any knowledge of SPARQL as the code will be provided to you together with its explanation.

We will start with a basic query to retrieve all datasets and their titles from the metadata.

```
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>

SELECT ?dataset ?title
WHERE {
  ?dataset a dcat:Dataset ;
          dct:title ?title .
}
```

Fig. 15: SparQL query, retrieving all datasets

Copy the highlighted text above and paste it into the Query text area of Apache Jena Fuseki server as shown in Figure 16.

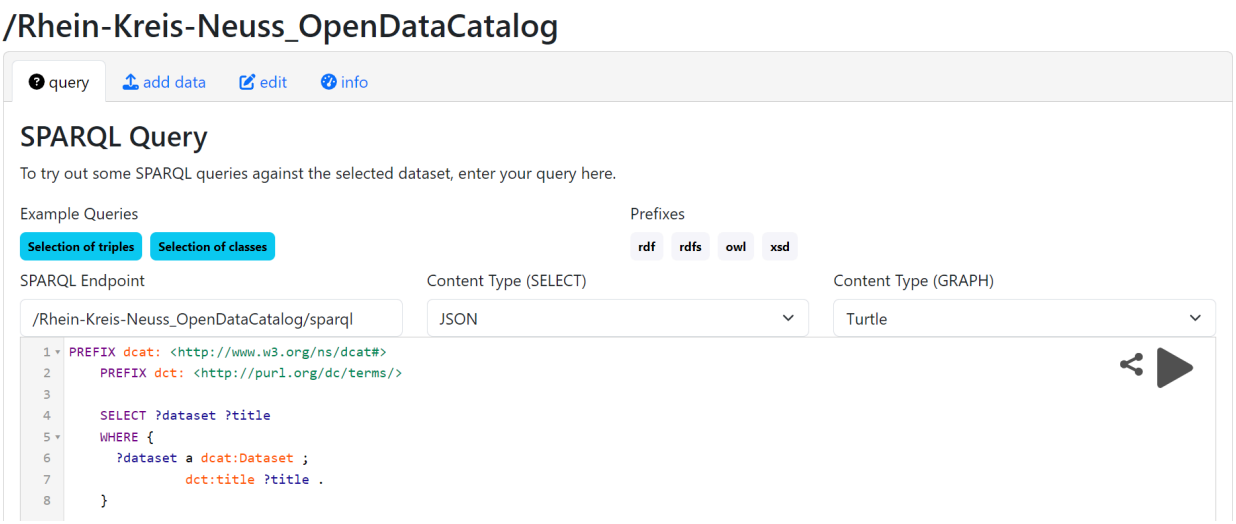

Fig. 16: Adding a SPARQL query to the Query text area of the Fuseki server.

This basic query retrieves all datasets and their titles from the DCAT metadata. The query uses the *dcat:Dataset* class to identify datasets and the *dct:title* property to retrieve the titles.

PREFIX: These lines define the namespaces for the DCAT and Dublin Core terms vocabularies, allowing us to use short forms like dcat: and dct:.

SELECT: This specifies the variables we want to retrieve, in this case, the dataset (?dataset) and its title (?title).

WHERE: This part of the query specifies the pattern to match in the RDF data. Here, we're looking for triples where the subject is a dcat:Dataset and has a dct:title property.

Figure 17 shows some results of the query.

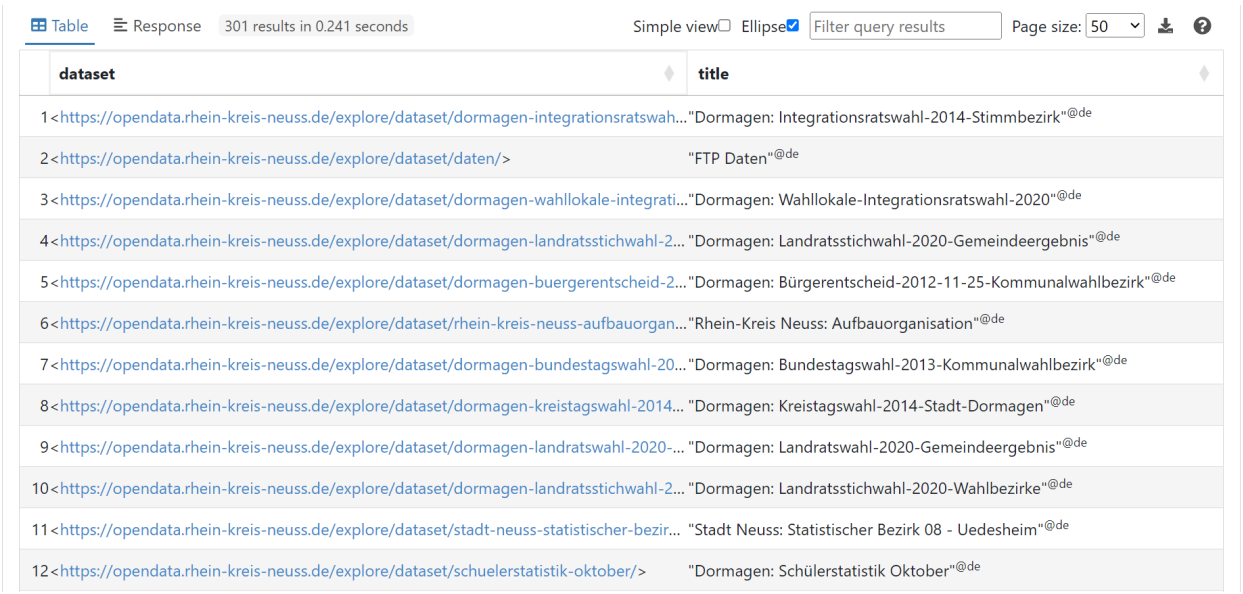

Table | Response | 301 results in 0.241 seconds | Simple view | Ellipse | Filter query results | Page size: 50

	dataset	title
1	<https://opendata.rhein-kreis-neuss.de/explore/dataset/dormagen-integrationsratswah...	"Dormagen: Integrationsratswahl-2014-Stimmbezirk"@de
2	<https://opendata.rhein-kreis-neuss.de/explore/dataset/daten/>	"FTP Daten"@de
3	<https://opendata.rhein-kreis-neuss.de/explore/dataset/dormagen-wahllokale-integrati...	"Dormagen: Wahllokale-Integrationsratswahl-2020"@de
4	<https://opendata.rhein-kreis-neuss.de/explore/dataset/dormagen-landratsstichwahl-2...	"Dormagen: Landratsstichwahl-2020-Gemeindeergebnis"@de
5	<https://opendata.rhein-kreis-neuss.de/explore/dataset/dormagen-buergerentscheid-2...	"Dormagen: Bürgerentscheid-2012-11-25-Kommunalwahlbezirk"@de
6	<https://opendata.rhein-kreis-neuss.de/explore/dataset/rhein-kreis-neuss-aufbauorgan...	"Rhein-Kreis Neuss: Aufbauorganisation"@de
7	<https://opendata.rhein-kreis-neuss.de/explore/dataset/dormagen-bundestagswahl-20...	"Dormagen: Bundestagswahl-2013-Kommunalwahlbezirk"@de
8	<https://opendata.rhein-kreis-neuss.de/explore/dataset/dormagen-kreistagswahl-2014...	"Dormagen: Kreistagswahl-2014-Stadt-Dormagen"@de
9	<https://opendata.rhein-kreis-neuss.de/explore/dataset/dormagen-landratswahl-2020-...	"Dormagen: Landratswahl-2020-Gemeindeergebnis"@de
10	<https://opendata.rhein-kreis-neuss.de/explore/dataset/dormagen-landratsstichwahl-2...	"Dormagen: Landratsstichwahl-2020-Wahlbezirke"@de
11	<https://opendata.rhein-kreis-neuss.de/explore/dataset/stadt-neuss-statistischer-bezir...	"Stadt Neuss: Statistischer Bezirk 08 - Uedesheim"@de
12	<https://opendata.rhein-kreis-neuss.de/explore/dataset/schuelerstatistik-oktober/>	"Dormagen: Schülerstatistik Oktober"@de

Fig. 17: Results from the SPARQL query showing all datasets in the uploaded metadata.

Executing the query returns 301 datasets. Since the datasets are too many to fit in one view, the results table is paginated. These datasets represent or describe the various actual datasets that are available through the open data portal of Rhein-Kreis-Neuss district. We are now going to look deeper for some individual datasets or datasets that meet a particular criteria.

In the second query, we are going to retrieve dataset(s) with a specific keyword, in this case, datasets with the “Windkraft” keyword. Remember, a keyword is a specific term or phrase that describes a particular aspect of the dataset and is used to facilitate searching and filtering datasets.

```
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>

SELECT ?dataset ?title ?keyword
WHERE {
  ?dataset a dcat:Dataset ;
          dct:title ?title ;
          dcat:keyword ?keyword .
  FILTER (CONTAINS(?keyword, "Windkraft"))
}
```

Fig. 18: SparQL query, retrieving all datasets that relate to a keyword

This query retrieves datasets along with their titles and keywords, but it filters the results to only show datasets where the keyword contains the term "Windkraft" (German for "wind energy").

FILTER: The FILTER function is used to apply a condition to the results. CONTAINS checks if the keyword includes the term "Windkraft".

Executing the above query produces the result shown in Figure 19.

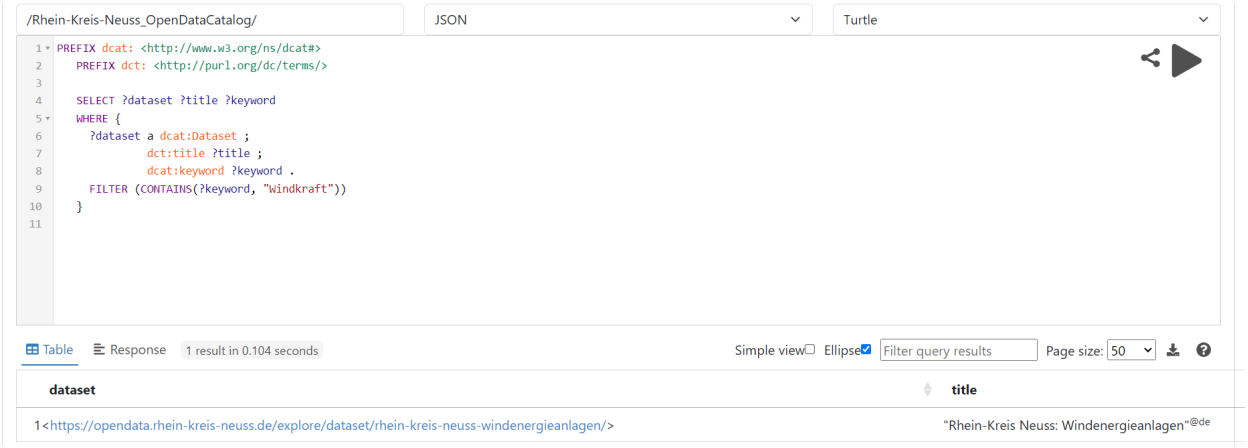

Fig. 19: Results from the SPARQL query showing all datasets with the keyword 'Windkraft'

The results show that *Rhein-Kreis Neuss: Windenergieanlagen* is the only dataset that meets the criteria. Such a query is important for searching through metadata that has multiple datasets, for example, the metadata catalog of an entire data portal.

Task 3.4.3-A

> Open the *Rhein-Kreis-Neuss_OpenDataCatalog* metadata that we uploaded to the server on a text editor, for example Notepad. Copy one of the keywords in the wind turbines metadata and replace it in the SPARQL query above. Execute the query and view the results.

In the third query, we are going to extract dataset(s) and their distributions. Remember, distribution refers to a specific representation of a dataset.

```
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>

SELECT ?dataset ?title ?distribution
WHERE {
```

```
  ?dataset a dcat:Dataset ;
        dct:title ?title ;
        dcat:distribution ?distribution .
}
```

Fig. 20: SparQL query, extracting datasets with their titles and distributions

dcat:distribution: This property links datasets to their distributions. The query retrieves the URI of the distribution, which can provide further details like the format and access URL.

Executing the query produces the results shown in Figure 21.

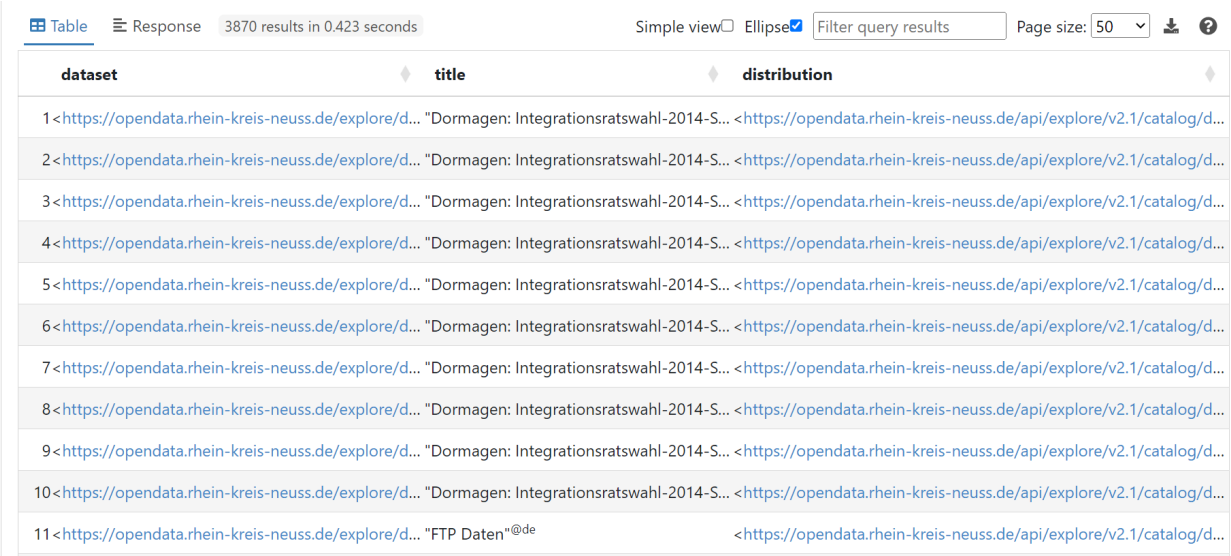

Fig. 21: Results from the SPARQL query that shows all distributions of all datasets

From the results, there are 3870 distributions belonging to the various datasets that are managed by the open data catalog. However, these results are not very useful when we are interested in a particular dataset. Let's assume we are interested in the *Rhein-Kreis Neuss: Windenergieanlagen* dataset that we initially filtered using the keyword. We want to know what distributions are available for that dataset. We can do this by filtering the available distributions by the ‘Windkraft’ keyword:

```
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>

SELECT ?dataset ?title ?distribution
WHERE {
```

```
  ?dataset a dcat:Dataset ;
          dct:title ?title ;
          dcat:keyword ?keyword ;
          dcat:distribution ?distribution .
  FILTER (CONTAINS(?keyword, "Windkraft"))
}
```

Fig. 22: SparQL query, retrieving all distributions of the *Windenergieanlagen* dataset

dcat:distribution property is used to get the distribution details of the filtered dataset(s).

Running the query produces the results shown in figure 23.

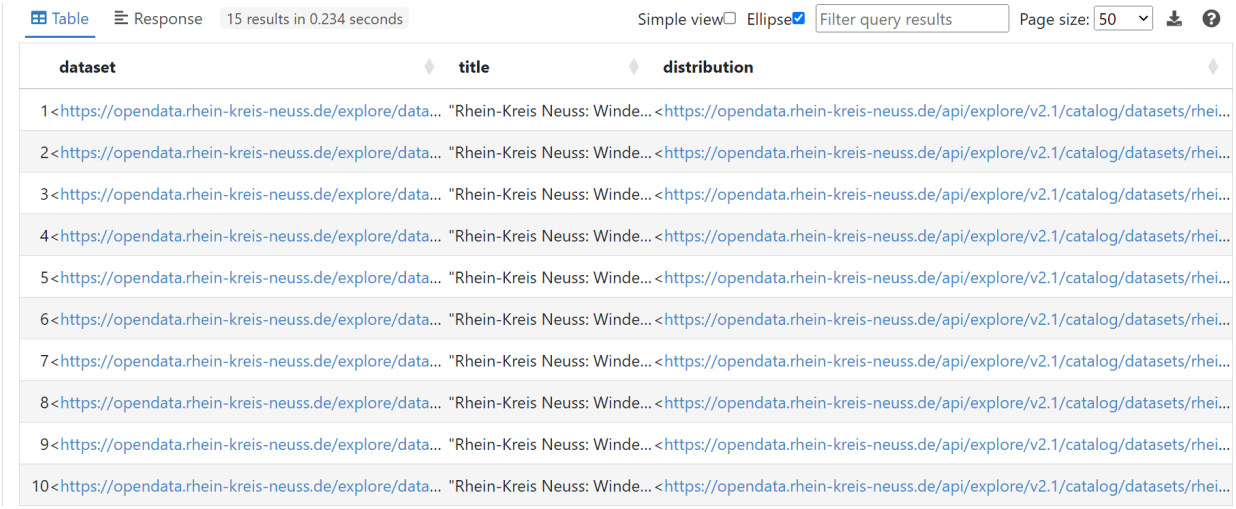

Fig. 23: Results from the SPARQL query that shows all distributions from the *Rhein-Kreis Neuss: Windenergieanlagen* dataset

The *Rhein-Kreis Neuss: Windenergieanlagen* dataset has 15 different distributions. However, this information is not very useful to a user who wants to use the dataset, as there is no mention of the distribution names and the associated information.
We can go further and get more information about the distribution formats, including their names and access URLs.

```
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>

SELECT ?dataset ?datasetTitle ?distribution ?distributionTitle ?accessURL
```

```
WHERE {
  ?dataset a dcat:Dataset ;
          dct:title ?datasetTitle ;
          dcat:distribution ?distribution ;
          dcat:keyword ?keyword .

  OPTIONAL { ?distribution dct:title ?distributionTitle . }
  ?distribution dcat:accessURL ?accessURL .

  FILTER (CONTAINS(?keyword, "Windkraft"))
}
```

Fig. 24: SparQL query, retrieving details of the *Windenergieanlagen* dataset's distributions

This advanced query retrieves dataset(s) of interest, their titles, the titles of their distributions (if available), and the access URLs for those distributions. It first gets the details of all the distributions for all the datasets. It then filters them down to only retain the distributions of those datasets that have the keyword *Windkraft*.

OPTIONAL: This clause is used to include optional information. In this case, it retrieves the title of the distribution if it exists. If a distribution does not have a title, it will still be included in the results without discarding the entire result.

dcat:accessURL: This property gives the URL where the distribution can be accessed or downloaded.

Executing the query produces the results shown in Figure 25.

Table | Response | 15 results in 0.248 seconds | Simple view | Ellipse | Filter query results | Page size: 50

	dataset	datasetTitle	distribution	distributionTitle	accessURL
1	<https://opendata.rhein-kr...	"Rhein-Kreis Neuss: ...	<https://opendata.rhein-kreis-neuss.de/..."rhein-kreis-neuss-winden...	<https://opendata.rhein-kreis-neuss.d...
2	<https://opendata.rhein-kr...	"Rhein-Kreis Neuss: ...	<https://opendata.rhein-kreis-neuss.de/..."rhein-kreis-neuss-winden...	<https://opendata.rhein-kreis-neuss.d...
3	<https://opendata.rhein-kr...	"Rhein-Kreis Neuss: ...	<https://opendata.rhein-kreis-neuss.de/..."rhein-kreis-neuss-winden...	<https://opendata.rhein-kreis-neuss.d...
4	<https://opendata.rhein-kr...	"Rhein-Kreis Neuss: ...	<https://opendata.rhein-kreis-neuss.de/..."rhein-kreis-neuss-winden...	<https://opendata.rhein-kreis-neuss.d...
5	<https://opendata.rhein-kr...	"Rhein-Kreis Neuss: ...	<https://opendata.rhein-kreis-neuss.de/..."rhein-kreis-neuss-winden...	<https://opendata.rhein-kreis-neuss.d...
6	<https://opendata.rhein-kr...	"Rhein-Kreis Neuss: ...	<https://opendata.rhein-kreis-neuss.de/..."rhein-kreis-neuss-winden...	<https://opendata.rhein-kreis-neuss.d...
7	<https://opendata.rhein-kr...	"Rhein-Kreis Neuss: ...	<https://opendata.rhein-kreis-neuss.de/..."rhein-kreis-neuss-winden...	<https://opendata.rhein-kreis-neuss.d...
8	<https://opendata.rhein-kr...	"Rhein-Kreis Neuss: ...	<https://opendata.rhein-kreis-neuss.de/..."rhein-kreis-neuss-winden...	<https://opendata.rhein-kreis-neuss.d...
9	<https://opendata.rhein-kr...	"Rhein-Kreis Neuss: ...	<https://opendata.rhein-kreis-neuss.de/..."rhein-kreis-neuss-winden...	<https://opendata.rhein-kreis-neuss.d...
10	<https://opendata.rhein-kr...	"Rhein-Kreis Neuss: ...	<https://opendata.rhein-kreis-neuss.de/..."rhein-kreis-neuss-winden...	<https://opendata.rhein-kreis-neuss.d...
11	<https://opendata.rhein-kr...	"Rhein-Kreis Neuss: ...	<https://opendata.rhein-kreis-neuss.de/..."rhein-kreis-neuss-winden...	<https://opendata.rhein-kreis-neuss.d...

Fig. 25: Results from the SPARQL query that shows all distributions of the dataset, including their titles and access urls.

We now see the titles of the various distributions and their access URLs.

Task 3.4.3-B SPARQL query - retrieving datasets related to the keyword “Ladesäulen”

> Imagine that you are part of a team responsible for planning new electric vehicle (E-charging) stations across your state. To make informed decisions, you need to identify relevant datasets that will help in understanding the infrastructure needs, current energy use, and traffic patterns.
>
> Your task is to search the catalog for datasets that might be valuable for this planning process. Specifically, you are looking for datasets that contain the keyword "Ladesäulen" to ensure they are related to electric vehicles, energy, or charging infrastructure.
>
> Write a SPARQL query that retrieves datasets with the keyword "Ladesäulen".
> Retrieve the title and description of these datasets to understand what information they provide.
> Analyze the results and identify which datasets are most relevant for making decisions about where to place E-charging stations.

Task 3.4.3-C SPARQL query - retrieving details on the distributions of a certain dataset

> Select one dataset of choice from the available options, and extract more information about its distributions, including the distribution names and access URLs.
>
> **Note**: To extract more information about an individual dataset, you should filter the results by the title, instead of filtering by a keyword, since multiple datasets can contain the same keyword. In this case, we can modify the filter parameters to:
> *FILTER (CONTAINS(?datasetTitle, "dataset_title"))*
>
> Replace the *dataset_title* with the actual title of the dataset.

4. Summary and Discussion

In this tutorial, we examined DCAT's role in improving the discoverability, accessibility, and interoperability of datasets by standardizing metadata descriptions. DCAT provides a uniform way of cataloging and describing datasets, making it easier for users to find and use data across different platforms. This was showcased in the wind turbines’ dataset example. The

dataset's metadata has been integrated into various catalogs hence making the dataset discoverable from different data portals.

The practical exercises, involving tools like RDForms and SPARQL, gave us hands-on experience in creating, validating, and querying DCAT metadata. RDForms simplifies metadata creation, while validation tools like DCAT-AP.de help maintain consistency with established standards. SPARQL enables complex queries on DCAT metadata, and Apache Jena Fuseki provides a powerful platform for executing these queries.

As the demand for reusable data grows, the use of DCAT is expected to increase significantly. Its ability to standardize and promote interoperability of metadata will play a huge role in future open data portals and initiatives. Geospatial software and tools like ArcGIS Hub[3] have also started to support DCAT. The support for DCAT by other geospatial software is bound to increase as Linked Data principles become more widespread on the web.

The following resources may help you to further dive into the topic and deepen your understanding:

1) W3C official documentation for DCAT.
   https://www.w3.org/TR/vocab-dcat-3/

2) Linked data
   https://www.w3.org/wiki/LinkedData

3) W3Schools RDF/XML
   https://www.w3schools.com/xml/xml_rdf.asp

4) W3C official documentation for RDF
   https://www.w3.org/TR/rdf12-concepts/

5) Getting started with Docker
   https://docker-curriculum.com/

[3] https://doc.arcgis.com/en/hub/content/federate-data-with-external-catalogs.htm