

Berufsakademie Sachsen  
Staatliche Studienakademie Leipzig

# **Multimodale Datenakquisition, Veredelung und Visualisierung nationaler Wissenschaftsinstitutionen**

Bachelorthesis  
zur Erlangung der staatlichen Abschlussbezeichnung eines  
Bachelor of Science (B. Sc.)  
in der Studienrichtung Informatik

**Eingereicht von:** Moustafa Kasab Banqusli  
Seminargruppe: cs17-2  
Matrikelnummer: 5001280  
Anschrift: Reichpitschstraße 43, 04317 Leipzig

**Betreuer:** Martin Röbert  
Firma: InfAI Managment  
Anschrift: Hainstraße 11, 04109 Leipzig

Leipzig, 29. April 2020

## Inhaltsverzeichnis

<i>Danksagung</i> .....	4
<i>1 Einleitung</i> .....	5
<i>1.1 Motivation</i> .....	5
<i>1.2 Zielstellung</i> .....	6
<i>1.3 Struktur der Arbeit</i> .....	7
<i>2 Grundlagen</i> .....	8
<i>2.1 Datenformate</i> .....	8
<i>2.1.1 Comma-Separated Values - CSV</i> .....	8
<i>2.1.2 Resource Description Framework - RDF</i> .....	9
<i>2.1.3 JavaScript Object Notation- JSON</i> .....	10
<i>2.2 Datensatz und Datenelemente</i> .....	11
<i>2.2.1 CORDIS Datensatz</i> .....	11
<i>2.2.2 Metadaten</i> .....	13
<i>2.2.3 Data Catalog Vocabulary- DCAT</i> .....	13
<i>2.2.4 SPARQL</i> .....	14
<i>2.3 REST API</i> .....	18
<i>2.4 Wissensgraph</i> .....	19
<i>3 Analyse des CORDIS Datensatz</i> .....	21
<i>4 Anforderungen</i> .....	22
<i>4.1 Abfragen mit SPARQL</i> .....	24
<i>4.2 RDFLIB</i> .....	26
<i>4.3 JSON-LD</i> .....	26
<i>4.4 Visualisierung</i> .....	29
<i>4.4.1 Vega</i> .....	29
<i>4.4.2 Leaflet</i> .....	31
<i>4.4.3 Data Tables</i> .....	32

---

5 Implementierung.....	35
5.1 Datentransformation.....	35
5.2 Wissensgraph mit RDFLIB.....	38
5.3 Visualisierung.....	39
5.3.1 Kompakte Daten mit JSON-LD.....	40
5.3.2 Visualisierung JSON-LD Data.....	42
5.3.3 Weltkarte mit Leaflet.....	42
6 Evaluation.....	43
6.1 Daten anfordern.....	43
6.2 Erfolgreiche Datenkonvertierung.....	43
6.3 RDFLIB Speicher.....	43
6.4 Von Wissensgraph-Speicher zu JSON-LD.....	44
6.5 Auswertung der Visualisierung.....	44
6.6 Visualisierung mit Leaflet.....	44
7 Zusammenfassung.....	46
8 Verzeichnisse.....	48
8.1 Abkürzungsverzeichnis.....	48
8.2 Tabellenverzeichnis.....	49
8.3 Abbildungsverzeichnis.....	49
8.4 Quellenverzeichnis.....	50
A Anhang.....	51
I Quellcode.....	51
II Github-Repository.....	52

## Danksagung

Für die Möglichkeit, mein Studium abzuschließen, danke ich meinem Projektbetreuer Martin Röbert, der mich immer unterstützt hat und Michael Martin, der mir bei der Arbeit geholfen hat, indem er mir wichtige Informationen über die Arbeit gegeben hat. Ich danke Prof.in Dr.in Susanne Schneider, dass Sie mir Ideen gegeben hat, wie diese Arbeit geschrieben werden soll und wie man sich auf die grammatikalische Struktur konzentriert. Abschließend möchte ich allen danken, die mich während der Bachelorarbeit unterstützt haben.

# 1 Einleitung

Vor dem Einstieg in das Thema der Arbeit sollte man über die Quelle der Informationen sprechen, die zum Aufbau benötigt werden. CORDIS<sup>1</sup> (Community Research and Development Information Service) ist die Community-Informationsservice für Forschung und Entwicklung. CORDIS ist der Europäischen Kommission und die wichtigste Ergebnisquelle für das Projekt. CORDIS verfügt über ein umfassendes und strukturiertes öffentliches Register, das alle Projektinformationen der Europäischen Kommission enthält, z. B. Projektdatenblätter, Teilnehmer, Berichte, Leistungen und Links zu frei zugänglichen Veröffentlichungen. Der erste Teil der Arbeit widmet sich den Grundlagen. Er gibt eine Übersicht über die Datentypen, die in dieser Arbeit verwendet werden. Es werden außerdem weitere Informationen zu den Daten und Erläuterungen zum Auffinden der erforderlichen Daten gegeben. Bereits der Titel wirft die Frage auf, was eigentlich Multimodalen Daten sind. Multimodalen Daten sind eine Sammlung miteinander verbundener Daten, die Text, Grafiken, Bilder, Animationen, Videos, Audiodaten usw. umfassen. Jedes nationale Wissenschaftsinstitut sammelt Daten zu seinen Projekten und dabei sind diese Daten sehr unterschiedlich und vielfältig. Diese Arbeit skizziert eine Visualisierungsmöglichkeit für diese Daten, um die enthaltenen Informationen zu sortieren und in grafischer Form anzuzeigen. Ziel ist es, im CORDIS Datenportal nach den richtigen Datensätzen<sup>2</sup> zu suchen, diese in ein geeignetes Format zu konvertieren um sie dann zu visualisieren. Die angeforderten Daten sollen im Resource Description Framework (RDF) - Format angelegt sein und dann in einem Wissensgraph gespeichert werden.

## 1.1 Motivation

Die breite Öffentlichkeit ist zunehmend an einem Zugang zu offenen Informationen interessiert, insbesondere in Bereichen wie Gesundheitswesen, Politik und Klimawandel. Diese Informationen sind auch für Wissenschaftler und Programmierer von Bedeutung, um beispielsweise Annahmen zu überprüfen oder Verknüpfungen zwischen Datensätzen aufzudecken. Die große Herausforderung besteht dabei darin, den richtigen Datensatz für einen zu untersuchenden Sachverhalt zu finden. In unserem Fall sollen Datensätze gefunden werden, die Projekte und verwandte Organisationen enthalten, die von der Europäischen Union finanziert werden. Der Datensatz einer Institution enthält alle Projekte und verwandte Organisationen. Er kann in verschiedene Dateien aufgeteilt werden, sodass es einfacher ist, mit ihnen umzugehen. Die Dateien können in unterschiedlichsten Datenformat vorliegen. Daraus ergibt sich die Herausforderung, die Informationen von ihrem ursprünglichen Format in ein anderes Format zu

---

1 CORDIS : Europäische Kommission: [https://ec.europa.eu/info/index\\_en](https://ec.europa.eu/info/index_en)

2 Gruppe in bestimmter Hinsicht zusammengehöriger Daten einer Datei

konvertieren. Die Konvertierung ist wichtig, um die Daten in einem Wissensgraph zu speichern.

## 1.2 Zielstellung

Für die Visualisierung der Daten aus einem Wissensgraph muss man die Daten entsprechend anpassen. CORDIS kann Daten auch in Form von comma-separated values (CSV) bereitstellen, welche sich dann einfach in das Format des Resource Description Framework (RDF) konvertieren lassen um daraus einen Wissensgraph erstellen zu können. Zur Visualisierung müssen die Daten im nächsten Schritt aus dem Wissensgraph extrahiert werden. Die Daten werden als Netz von Informationen visualisiert, die über Beziehungen miteinander verbunden sind. Dadurch entsteht ein besseres Verständnis der Daten. Ein Beispiel für einen Wissensgraph wird in Abbildung 1 gezeigt.

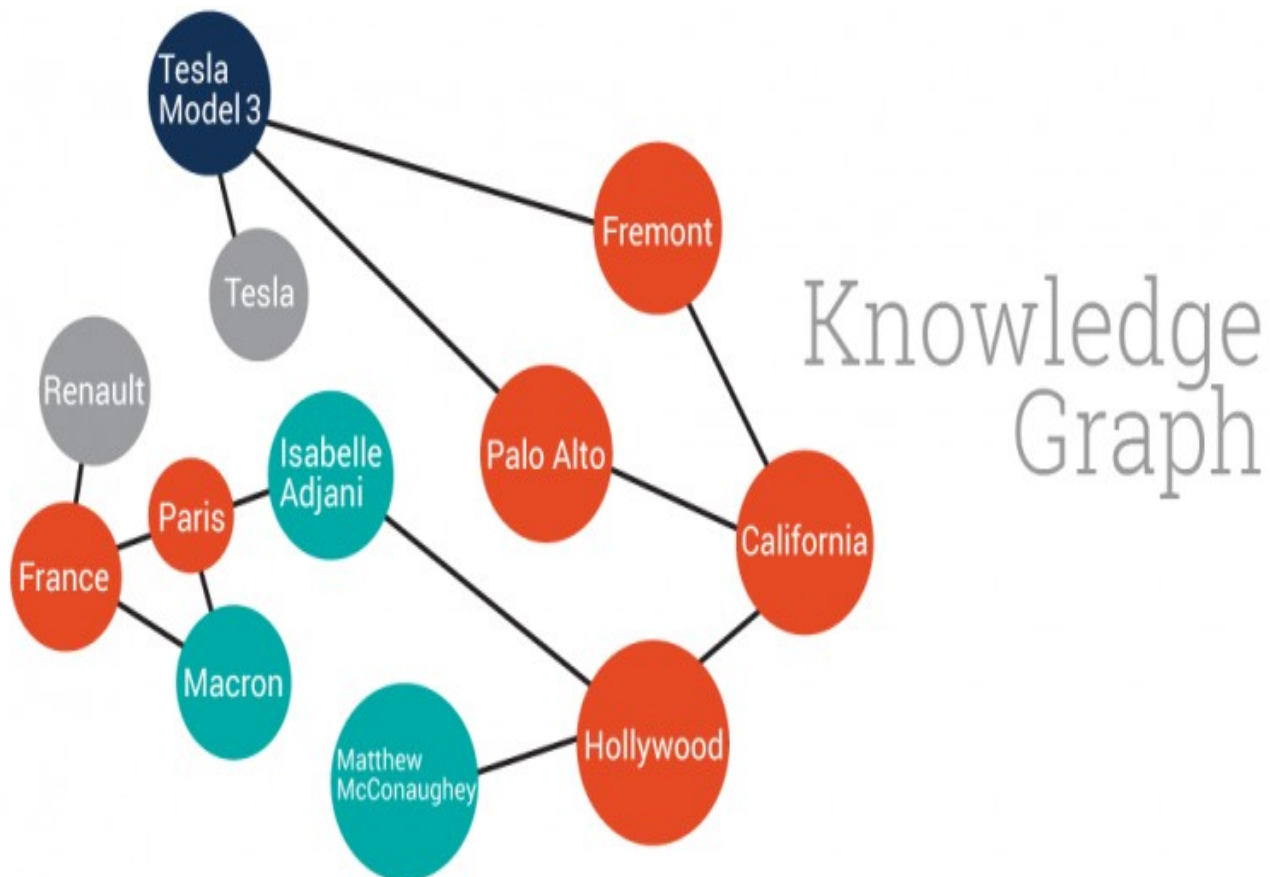


Abbildung 1: Wissensgraph als Datenvisualisierung<sup>3</sup> [1]

Die Daten können in vielen verschiedenen Varianten dargestellt werden und können so z.B. deutlich machen, welches Land die meisten Projekte in Europa hat.

<sup>3</sup> Quelle: <https://www.ontotext.com/knowledgehub/fundamentals/what-is-a-knowledge-graph/>

## 1.3 Struktur der Arbeit

Um das Ziel dieser Arbeit zu erreichen, müssen die erforderlichen Daten über das EU ODP<sup>4</sup> gefunden werden. Die Daten werden dann untersucht, deren Type ermittelt um dann konvertiert und gespeichert zu werden. Die konvertierten Daten dienen im nächsten Schritt als Grundlage für die Erstellung eines Wissensgraph, welcher in unterschiedlichen Varianten der Daten-Visualisierung dient. Zu Beginn werden Grundlagen dieser Arbeit erläutert und die Anforderungen dargelegt. Einige Bibliotheken werden in den Anforderungen erwähnt und die sind wichtig für die Implementierung der Arbeit. Danach werden alle Implementierungsschritte beschrieben, welche für die Zielerreichung nötig sind. Schließlich werden die Ergebnisse im Detail untersucht und beurteilt.

---

<sup>4</sup> European Union Open Data Portal, <https://data.europa.eu/euodp/en/home>

## 2 Grundlagen

Die nächsten Seiten gehen auf grundlegendes Wissen für diese Arbeit ein. Es werden Datenformate erklärt sowie auf den Aufbau eines Datensatzes eingegangen. Weiter wird die REST API, SPARQL<sup>5</sup> und der Wissensgraph erläutert.

### 2.1 Datenformate

Diese Arbeit beschäftigt sich mit den Datentypen CSV, RDF und JSON. Das CSV-Format ist das Haupt Format der Daten, die man von CORDIS erhält. Das RDF-Data ist das Zielformat, welches nach der Konvertierung die Grundlage der Visualisierung bildet. JSON wird in dieser Arbeit als unterstützende Bibliothek genutzt.

#### 2.1.1 Comma-Separated Values - CSV

Diese Art von Daten kann man direkt verstehen, da die Werte einfach durch ein Komma voneinander getrennt sind. Zum Beispiel hat man eine Excel-Tabelle zum Gehalt von Mitarbeitern. Die Tabelle enthält die Spalten Mitarbeiter-ID, Name des Mitarbeiters und Gehalt.

employee-id	employee-name	salary
1	Tom	2500
2	Toni	2300
3	Jack	2600

Tabelle 1: Beispiel über das Gehalt der Mitarbeiter

Um diese Daten im CSV-Format darzustellen, werden die Werte einfach durch ein Komma getrennt:

employee-id,employee-name,salary
1,Tom,2500
2,Toni,2300
3,Jack,2600

Tabelle 2: Daten im CSV-Format

Die erste Zeile enthält den Spaltennamen, gefolgt von den Werten.

<sup>5</sup> SPARQL (Sparql Protocol And Rdf Query Language) ist eine Semantische Abfragesprache für Datenbanken, mit der im RDF-Format gespeicherte Daten abgerufen und bearbeitet werden können.



## 2.1.2 Resource Description Framework - RDF

RDF steht für Resource Description Framework und ist eine wichtige Grundlage für die Unterstützung des Semantic Web<sup>6</sup>. RDF wurde 1999 als W3C-Empfehlung angenommen. Die RDF 1.0-Spezifikation wurde im Jahr 2004 vom World Wide Web Consortium (W3C)<sup>7</sup> veröffentlicht. Sie ist eine einfach zu verwendende Metadaten-Sprache. RDF erweitert die Funktionen der XML-Sprache und ist eine homogene und einheitliche Methode zur Beschreibung der Internetressourcen und zum Anfordern von Informationen von Text- und Grafikseiten bis hin zu Audio- und Videodateien. Das Resource Description Framework (RDF) eignet sich zum Veröffentlichen von Datenbanken im Web. Allen Elementen in der Datenbank wird durch das RDF-Format auch ihr Ursprung hinzugefügt. Mit dieser Verknüpfung der Daten-Elemente durch RDF können außerdem andere Programme Daten miteinander abgleichen. Die Sprache des Resource Description Framework (RDF) ist damit eine einfache Sprache zur Beschreibung von Datenbank-Modellen, der Quellen und ihren Beziehungen - auch zu anderen Datenbanken. Zur Visualisierung des RDF-Formates im Web wird häufig HTML bzw. CSS genutzt. Mit RDF können der Inhalt, die Verknüpfungen und die Strukturebene zwischen den Elementen in Form eines Datenmodells beschrieben werden.[3]

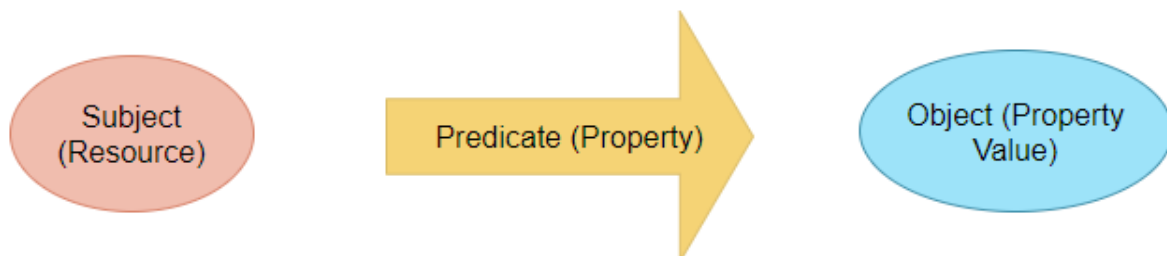


Abbildung 2: Ein Bild zeigt, das Subjekt, Prädikat und das Objekt

RDF-Dokumente werden in XML geschrieben. Extensible Markup Language (XML) ist HTML sehr ähnlich. Es werden Tags verwendet. HTML wird jedoch zum Anzeigen der Daten verwendet, XML allerdings zum Speichern und Transportieren von Daten verwendet wird.[2]

---

<sup>6</sup> Das Semantic Web ist eine Erweiterung des World Wide Web durch Standards, die vom World Wide Web Consortium (W3C) festgelegt wurden. Ziel des Semantic Web ist es, Internetdaten maschinenlesbar zu machen.

[https://en.wikipedia.org/wiki/Semantic\\_Web](https://en.wikipedia.org/wiki/Semantic_Web)

<sup>7</sup> Das World Wide Web Consortium (W3C) ist eine internationale Gemeinschaft, die offene Standards entwickelt, um das langfristige Wachstum des Web sicherzustellen. <https://www.w3.org/>

Die RDF-Daten sehen demnach XML-Daten sehr ähnlich, allerdings mit zusätzlichen Namespaces<sup>8</sup> und Präfixes, welche über den Daten hinzugefügt werden. Dabei ist ein Präfix eine Abkürzung für den vollständigen XML-Namespace. Dies wird in Abschnitt 2.2.4 näher erläutert.

```
<?xml version="1.0"?>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:si="https://www.w3schools.com/rdf/">

  <rdf:Description rdf:about="https://www.w3schools.com">
    <si:title>W3Schools</si:title>
    <si:author>Jan Egil Refsnes</si:author>
  </rdf:Description>

</rdf:RDF>
```

Tabelle 3: Beispiel für ein RDF-Dokument<sup>9</sup>

```
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

Tabelle 4: Präfix und Namespace

### 2.1.3 JavaScript Object Notation- JSON

JSON eine Abkürzung für den Begriff JavaScript-Object Notation. Es ist eine Methode zum Darstellen und Austauschen von Daten zwischen verschiedenen Programmiersprachen, damit Daten von Menschen gelesen und verstanden werden können. Außerdem ist es einfach, diese Daten in verschiedenen Programmiersprachen zu analysieren und zu verwenden. In der Regel wird diese Sprache (JSON) verwendet, um die Übertragung von Daten vom Gerät des Benutzers zum Server zu erleichtern und umgekehrt, wo der Mensch die im JSON-Format dargestellten Daten als temporäre Datenbank betrachten kann. JSON ist keine Programmiersprache, sondern eine Formel oder Methode zur Darstellung von Daten und wird in den meisten Programmiersprachen unterstützt, in denen im JSON-Format geschriebener Text in Object konvertiert wird - und umgekehrt. Aufgrund der einfachen Erstellung des Objekts in der JavaScript-Sprache, in der die geschweiften Klammern geschrieben werden, In ihnen werden die Daten einzeln geschrieben und durch ein Komma getrennt. JSON ist eine Möglichkeit,

<sup>8</sup> Namespaces werden verwendet, um eindeutig benannte Elemente und Attribute in einem XML-Dokument bereitzustellen

<sup>9</sup> Quelle: [https://www.w3schools.com/xml/xml\\_rdf.asp](https://www.w3schools.com/xml/xml_rdf.asp)

Daten auszudrücken, und es ist sowohl für Menschen lesbar als auch maschinenlesbar. Es ist nur eine Art wirklich einfaches Format für den Eigenschafts Schlüsselwert. Wie in Tabelle 5 haben wir Name, Alter und Auto sind Schlüssel und wir haben John, 30 und null sind die Werte.

```
{ "name":"John", "age":30, "car":null }
```

Tabelle 5: JSON Objekt<sup>10</sup>[3]

## 2.2 Datensatz und Datenelemente

In diesem Abschnitt wird der Datensatz näher beschrieben, Metadaten erklärt und erläutert, wie man Daten mittels SPARQL finden kann.

### 2.2.1 CORDIS Datensatz

Die CORDIS-Datenbank enthält alle Datensätze zu Forschungsprojekten sucht. Die Datensätzen beziehen sich auf Projekte und Organisationen in unterschiedlichen Zeiträumen. Der für diese Arbeit erforderliche Datensatz findet sich im Rahmenprogramm „Horizont 2020“ für Forschung und Innovation von 2014 bis 2020. Dieser Datensatz enthält viele unterschiedliche Dateien. Die Datei 'H2020-Projekte' enthält die Informationen zu öffentlichen Zuschüssen für jedes Projekt, einschließlich der folgenden Informationen:

Spaltenname	Beschreibung
id	Um jedes Projekt mit einer eindeutigen Nummer anzugeben, die jedem Projekt zugewiesen wird .Diese Spalte kann mit der Spalte projectID der Organisationen verglichen werden, um festzustellen, welches Projekt zu welcher Organisation gehört
frameworkProgramme	Dies zeigt, zu welcher frameworkProgramme das Förderprogramm gehört
programme	Die Spalte zeigt, welche finanzierten Programme jedem Projekt folgen, z. B. "H2020-EU.1.3.2"
startdate	Diese Spalte zeigt das Startdatum eines Projekts
enddate	Diese Spalte zeigt das Enddatum eines Projekts
title	Projekttitel

<sup>10</sup> [https://www.w3schools.com/js/js\\_json\\_objects.asp](https://www.w3schools.com/js/js_json_objects.asp)

acronym	enthält die Abkürzung für Projekttitel
rcn	Datensatz Kontrollnummer
status	Aus dieser Spalte kann man den Projektstatus ermitteln. Zum Beispiel, wenn das Projekt "CLOSED" oder "SIGNED" wurde
totalCost	Die Gesamtkosten des Projekts
ecMaxContribution	Der EU-Kontribution (Beitrag)

Tabelle 6: H2020-Projekte Data

Die Datei 'H2020 Organisationen', die enthält die Organisationen, die von der Europäischen Union im Rahmen des Rahmenprogramms „Horizont 2020“ für Forschung und Innovation von 2014 bis 2020 finanziert werden. Die Datei enthält wichtige Informationen über die Organisationen wie der nächste Tabelle:

Spaltenname	Beschreibung
id	Um jedes Organisation mit einer eindeutigen Nummer anzugeben, die jedem Organisation zugewiesen wird
name	Name der Organisation
shortName	Kurzname der Organisation
country	Diese Spalte zeigt, in welchem Land sich die Organisation befindet
city	Diese Spalte zeigt, in welchem Stadt sich die Organisation befindet
street	Der Name der Straße
postCode	Postleitzahl der Organisation
ecContribution	Der EU-Kontribution
projectID	Um jedes Projekt mit einer eindeutigen Nummer anzugeben, die jedem Projekt zugewiesen wird
organisationUrl	Die Organisationshomepage

Tabelle 7: H2020-Organisationen Data

Es gibt auch die Datei 'H2020 Report Summaries', in der Zusammenfassungen der Projekte seit September 2018 enthalten sind. Diese Datei ist für die aktuelle Arbeit nicht relevant.

## 2.2.2 Metadaten

Der Begriff Metadaten wird auch als Daten über Daten bezeichnet. Metadaten erklären die Inhalte anderer Daten. Sie werden verwendet, um grundlegende Informationen zu Daten zusammenzufassen. Metadaten helfen den Benutzern, relevante Informationen zu finden. Ein Beispiel: Ein privates Telefongespräch. Das Telefon zeichnet die Nummer, die Uhrzeit und die Länge des Anrufs sowie den Ort auf, an dem dieser Anruf getätigt wurde.[14]

Es gibt zwei verschiedene Arten von Metadaten:

- **Descriptive metadata:** Hier werden Metadaten verwendet, um einem eindeutigen Datenelement zusätzliche Details hinzuzufügen. > z.B. Telefonnummer
- **Structural metadata:** wobei Metadaten die Struktur der Anzahl verwandter Daten definieren. > z.B. Dauer des Anrufs

Es wurde deutlich, dass Daten aus vielen Datensätzen bestehen und die Metadaten dabei helfen, diese Daten zu verstehen, z.B. wann diese Daten veröffentlicht wurden, wie viele Einträge sie enthalten, wann diese Daten das letzte Mal geändert wurden.

## 2.2.3 Data Catalog Vocabulary- DCAT

In den letzten Jahren hat DCAT geholfen, Informationen zu Organisationen zu katalogisieren und Unternehmen zur Verfügung zu stellen. Der Datenkatalog enthält Informationen zu Datenelementen in der Organisation, z.B. was ist die Datenquelle, wofür wird sie verwendet, Datenkatalog gibt es schon seit langer Zeit, sie boten den Menschen eine Möglichkeit zu finden, Wo waren die Daten? und wie wurde es verwendet? Data Catalog Vocabulary (DCAT) ist ein RDF-Vokabular, das die Interoperabilität zwischen im Web veröffentlichten Daten Katalogen erleichtert. Durch die Verwendung von DCAT zur Beschreibung von Datasets in Katalogen erhöhen Publisher die Erkennbarkeit und ermöglichen es Anwendungen, Metadaten aus mehreren Katalogen zu verwenden. Es ermöglicht die dezentrale Veröffentlichung von Katalogen und die Suche nach Verbund Datensätzen über Kataloge hinweg. Aggregierte DCAT-Metadaten können als Manifestdatei dienen, um die digitale Aufbewahrung zu erleichtern. DCAT<sup>11</sup> "bietet RDF-Klassen und -Eigenschaften, mit denen Datensätze und Datendienste beschrieben und in einem Katalog aufgenommen werden können".[5]

Der Datenkatalog bietet Namespaces und Präfixe zu verwenden.

---

<sup>11</sup>Data Catalog Vocabulary, URL: <https://www.w3.org/TR/vocab-dcat-2/>

Präfixe	Namespaces
dcat	<a href="http://www.w3.org/ns/dcat#">http://www.w3.org/ns/dcat#</a>
dctype	<a href="http://purl.org/dc/dcmitype/">http://purl.org/dc/dcmitype/</a>
rdf	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a>
locn	<a href="http://www.w3.org/ns/locn#">http://www.w3.org/ns/locn#</a>
rdfs	<a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a>
vcard	<a href="http://www.w3.org/2006/vcard/ns#">http://www.w3.org/2006/vcard/ns#</a>
dc	<a href="http://purl.org/dc/elements/1.1/">http://purl.org/dc/elements/1.1/</a>
owl	<a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a>

Tabelle 8: Einige DCAT Namespaces und Präfixe[5]

## 2.2.4 SPARQL

Sparql Protocol And Rdf Query Language (SPARQL) ist Abfragesprache für RDF. Um nach dem richtigen Datensatz zu suchen, werden spezielle Techniken wie SPARQL<sup>12</sup> verwendet. Das Wichtigste ist, den konkreten Datensatz und die Datensatz-ID zu kennen. Mit ihnen kann man die REST-API verwenden, die die EU ODP (European Union Open Data Portal) anbietet, um die Daten anzufordern. RDF beschreibt Daten mit 3-teiligen Aussagen, welche auch als Triple bezeichnet werden. Beispiel: Der Mitarbeiter gehört zum Bereich Webentwicklung.

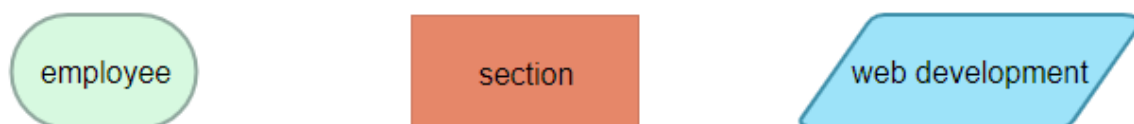


Abbildung 3: Triple beispiel

Ein Triple ist eine Menge von drei Entitäten. Das Subjekt und das Prädikat werden als URI (Uniform Resource Identifier) dargestellt. URIs sind wie URLs (Uniform Resource Locator) aufgebaut, aber sie sind keine Adressen, sondern Bezeichner. Zum Beispiel in Abbildung 4: Diese URIs zeigen, dass ein bestimmter Mitarbeiter

<sup>12</sup> SPARQL Query Language for RDF: <https://www.w3.org/TR/rdf-sparql-query/>

(employee) eines bestimmten Unternehmens (infai.org) in einem bestimmten Bereich arbeitet (web development). Das Objekt oder der dritte Teil des Triple kann bei Bedarf auch eine URI sein.[4, Kap 2, S 39]



Abbildung 4: Triple: 2 x URI und 1 x Objekt

Mit dieser Einteilung der Daten (Triple) können Ressourcen (z.B. Subjekt) gleichzeitig Teil (z.B. Objekt) eines anderen Triples sein. Diese daraus resultierenden Datennetze werden als Graph bezeichnet, siehe auch Abbildung 5.

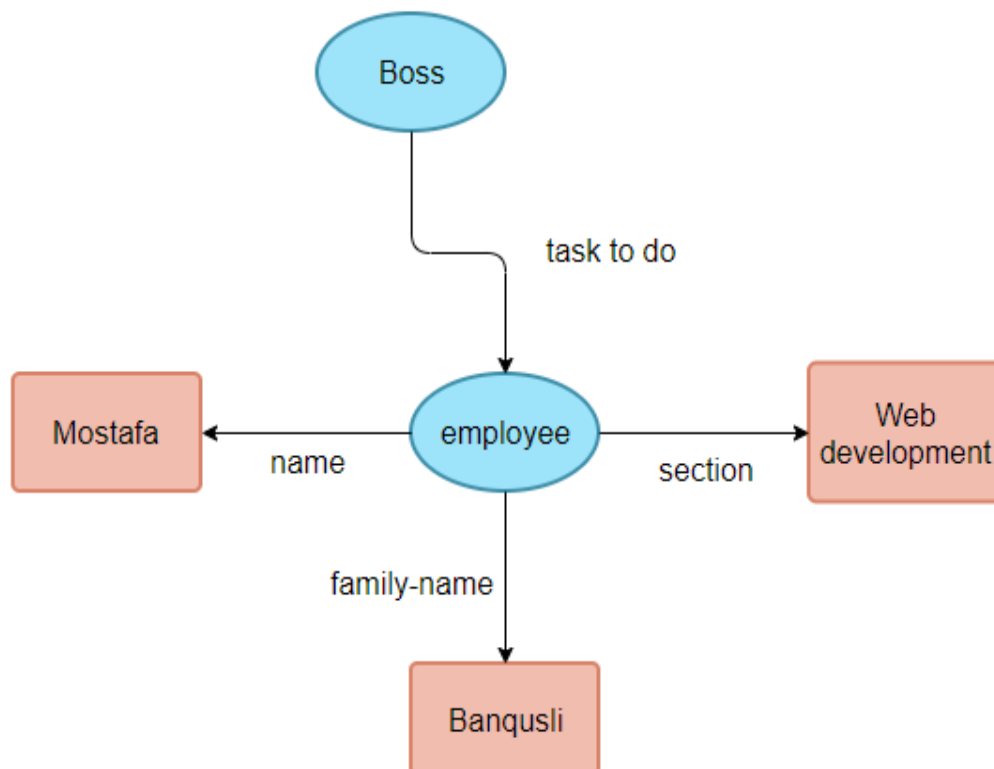


Abbildung 5: Mehrere Triples ergeben ein Datennetz

Zusätzlich verkürzt die beliebte RDF-Turtle-Syntax<sup>13</sup> die URIs häufig weiter, indem vor dem letzten Teil ein abgekürztes Präfix.

@prefix vcard: <http://www.w3.org/2006/vcard/ns#>

@prefix infai: <http://www.infai.org/hr/>

infai:employee vcard:section "Web development"

Tabelle 9: Prefix beispiel

Nahezu alle Daten können als Sammlung von Triples dargestellt werden. Im folgenden Beispiel kann jeder Wert der Tabelle als Triple dargestellt werden: die Tabellenzeilen-ID ist das Subjekt, der Spaltenname das Prädikat und der Wert das Objekt.

id	name	family-name	section (Spaltenname)
1	Tom	Smith	Web development
2 (Tabellenzeilen- ID)	Toni	Jackson	Web development (Objekt)
3	Jack	Kasab	Web development

Tabelle 10: Jeden Eintrag ist ein Triple

Als Beispiel für eine einfache SPARQL-Abfrage, bei der einige dieser Daten abgerufen werden, möchte man eine Liste aller Mitarbeiter, deren Nachname Smith ist. Genau wie die weiter oben erwähnte Turtle-RDF-Abfrage lässt SPARQL Präfixe definieren, damit man nicht die vollständigen URIs in seine Abfragen schreiben muss. Für die meisten SPARQL-Abfragen ist es am besten, mit dem where-Befehl zu beginnen um die gewünschten Triples zu finden. Der where-Befehl kann dabei mit einzelnen oder mehreren Triple-Bestandteilen (Subjekt, Prädikat oder Objekt), auch Triple-Muster genannt, durchgeführt werden. Im nächsten Beispiel hat die Abfrage nur einen Triple-Bestandteil (aus Daten der Tabelle 10).

<sup>13</sup> Ermöglicht das vollständige Schreiben eines RDF-Diagramms in einer kompakten und natürlichen Textform mit Abkürzungen für gängige Verwendungsmuster und Datentypen.  
<https://www.w3.org/TR/turtle/>



```
Where{
  ?person vcard: family-name "Smith".
}
```

Tabelle 11: Triple-Muster/Bestandteil

Das Triple-Muster wird mit Tripeln verglichen, deren Prädikat die Familiennamen-Eigenschaft aus dem vcard-Vokabular ist und deren Objekt der String "Smith" ist. Dabei ist das Subjekt unbekannt. Die unbekannte Variable ist in diesem Beispiel "?person". Der select-Befehl gibt an, welche Variablen aufgelistet werden sollen, nachdem die Abfrage ausgeführt wurde. Diese Abfrage enthält im nächsten Beispiel nur eine Variable. Wenn diese Abfrage ausgeführt wird, wird ein Triple gefunden, die dem spezifischen Muster entspricht, in unserem Fall "infai:employee" (Tabelle 13).

```
PREFIX vcard: <http://www.w3.org/2006/vacrd/ns#>

SELECT ?person {
  ?person vcard:family-name "Smith".
}
```

Tabelle 12: SPARQL-Abfrage

<code>?person</code>
<code>infai:employee</code>

Tabelle 13: SPARQL-Abfrage Ausgabe

Die Ausgabe sagt aktuell noch nicht, wer der Mitarbeiter ist. Im nächsten Schritt wird ein weiteres Triple-Muster hinzugefügt, das mit dem Vornamen des Mitarbeiters übereinstimmt. Das zweite Triple-Muster, das man hinzugefügt hat, um den Vornamen des Mitarbeiters herauszufinden, der mit dem ersten Triple-Muster übereinstimmt, heißt "?name". Zusätzlich muss auch der select-Befehl angepasst werden.

```

PREFIX vcard: <http://www.w3.org/2006/vacrd/ns#>

SELECT ?person ?name {
  ?person vcard:family-name "Smith".
  ?person vcard:name ?name.
}

```

Tabelle 14: SPARQL-Abfrage mit zwei Triple-Muster

?person	?name
infai:employee	Tom

Tabelle 15: Die Ausgabe von SPARQL-Abfrage mit zwei Triple-Muster

SPARQL kann noch mehr leisten, als hier aufgeführt wurde. an kann damit außerdem:

- Abfrageergebnisse sortieren, filtern und aggregieren.
- Daten hinzufügen, löschen, aktualisieren[4]

In der vorliegenden Arbeit dient SPARQL die notwendigen Daten zu finden. Im Abschnitt Anforderungen wird der konkrete Einsatz von SPARQL erläutert.

## 2.3 REST API

In dieser Arbeit wurde die REST-API lediglich verwendet, um Metadaten zu erfassen, die Ihre Hauptdatensatz Beschreiben. Diese Beschreibung wird auch zur Visualisierung verwendet, um die Hauptdaten besser zu verstehen und einen Blick auf die Funktionsweise der REST-API zu werfen. Das EU Open Data Portal <sup>14</sup> (EU ODP) bietet den Programmierern die Möglichkeit, mithilfe der REST-API<sup>15</sup> auf den gewünschten Datensatz zuzugreifen. Alle Kernfunktionen des Portals sind über die application programming interface (API) verfügbar. Die abgerufenen Informationen können dann von einem externen Code verwendet werden, um zu transformieren, zu aktualisieren oder zu referenzieren und neue Eingaben für weitere Aufrufe der API bereitzustellen. Das EU Open Data Portal (EU ODP) gibt Entwicklern eine Basis-URL, um die benötigten Daten abzurufen oder zu aktualisieren. Diese Basis-URL ist nicht vollständig, um von den Entwicklern verwendet zu werden, da sie besonders wichtige Informationen wie Dataset und Dataset-ID benötigt.

<sup>14</sup>Das EU Open Data Portal (EU ODP), Zugang zu offenen Daten der Europäischen Union: <https://data.europa.eu/euodp/en/home>

<sup>15</sup>REST API, URL: <https://data.europa.eu/euodp/en/developerscorner>

Anfragetyp	Basis-URL	Dataset-info
GET	https://data.europa.eu/euodp/	/dataset/{datasetId}.rdf

Tabelle 16: API Dokumentation

Das Nachrichtenformat für den Body der Request und die Antwort ist JavaScript Object Notation (JSON) für alle Anrufe. Dienste, die Datensatz Details bereitstellen, bieten jedoch auch eine Antwort in RDF/XML, aber normalerweise eingebettet in die JSON-Antwort.[9]

Anfragetyp	Basis-URL	Dataset-info
GET	https://data.europa.eu/euodp/	en/data/dataset/ cordisH2020projects.rdf

Tabelle 17: GET-Anfrage für Projekte

Man weiß bereits, dass der Datensatz aus zwei Hauptdateien besteht, es handelt sich um Projekte und Organisationen. EU ODP stellt die Hauptdaten jedoch nicht über die REST-API bereit. Wenn man GET-Request wie in Tabelle 17 verwendet, werden die Metadaten in der Form RDF / XML zurückgegeben. Es erklärt nur das Enthalten der Daten. Zum Beispiel, wann Daten erstellt wurden und von wem, wie der Titel der Daten lautet.

## 2.4 Wissensgraph

Der Wissensgraph ist eine Sammlung miteinander verbundener Beschreibungen von Entitäten, Objekten oder anderen Dingen. Wenn Jemand im Netz sucht, tippt er nicht nur einfache Wörter. Diese Worte beziehen sich auf eine reale Sache, auf ein Objekt in dieser Welt. zum Beispiel, jemand sucht nach einem Buch, erhält er relevante Informationen über das Buch, z. B. das Veröffentlichungsdatum oder Anzahl der Seiten und auch Informationen zum Autor. Diese Informationen sind nicht nur zufällige Dinge von Charakteren, sie können auch nur den richtigen Inhalt erhalten, nach dem Jemand im Web sucht. In der Wissensgraph geht es darum, Informationen über Objekte in dieser Welt zu sammeln, das Objekt könnte es sein eine Person, ein Buch oder irgendetwas auf dieser Welt. Es ist nur eine Möglichkeit, Informationen über reale Verbindungen aufzubauen. Es ist nur eine Möglichkeit, Informationen über reale Welt Verbindungen aufzubauen. Durch die Suche im Web kann der Wissensgraph ein Bereich direkt neben den Suchergebnissen sein. Der Wissensgraph kann so reich an Informationen sein, dass es die Möglichkeit bietet, mehr über die Abfragen des Benutzers zu verstehen und um mehr über die Informationen im Web zu verstehen und die Informationen intelligent miteinander zu verbinden.[7]

Wissensgraph "Es bietet eine Struktur und eine gemeinsame Schnittstelle für alle Ihre Daten und ermöglicht die Erstellung intelligenter multilateraler Beziehungen in Ihren Datenbanken. Das als zusätzliche virtuelle Datenschicht strukturierte Wissensgraph liegt auf Ihren vorhandenen Datenbanken oder Datensätzen, um alle Ihre Daten maßstabsgetreu miteinander zu verknüpfen - sei es strukturiert oder unstrukturiert." [8]

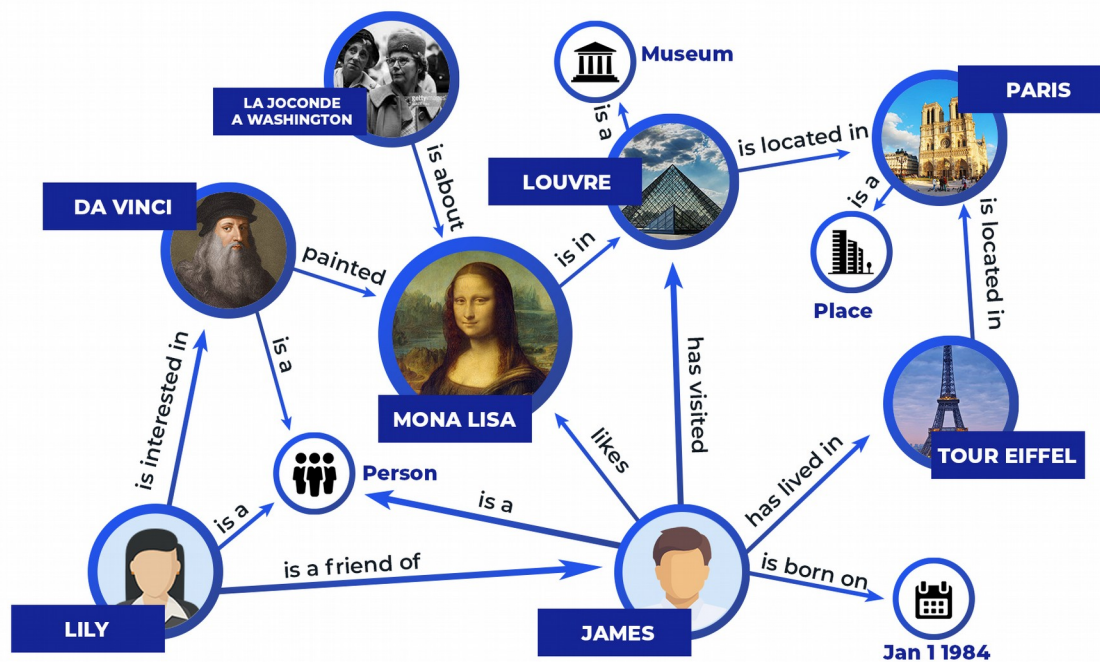


Abbildung 6: Beispiel über Wissensgraph[6]

### 3 Analyse des CORDIS Datensatz

Kern der Arbeit ist der schon bereits Abschnitt 2.2.1 beschriebene CORDIS Datensatz. Die beiden wichtigsten Dateien, die benötigt werden, sind Projekte und Organisationen. Die Datei zu den Projekten enthält ungefähr 131.191 Einträge. Die Datei zu den Organisationen umfasst etwa 115.672 Einträge. Jeder Eintrag aus beiden Dateien muss vom CSV-Format hin zum RDF-Format konvertiert werden um später einen Wissensgraph zu generieren. Nach der Konvertierung liegt jeder Wert als Triple vor: Für jedes Subjekt und Prädikat wird eine URI zugewiesen. Beispiel: Die Werte der nachstehender Projekt-Tabelle (Tabelle 18) ergeben nach der RDF-Konvertierung die in Abbildung 7 dargestellten Triples. Die Projekt-ID liegt als Subjekt, das Startdatum als Prädikat und das Datum "3-5-2020" als Objekt vor.

project-id	start-date
453	3-5-2020

Tabelle 18: Projekte Tabelle

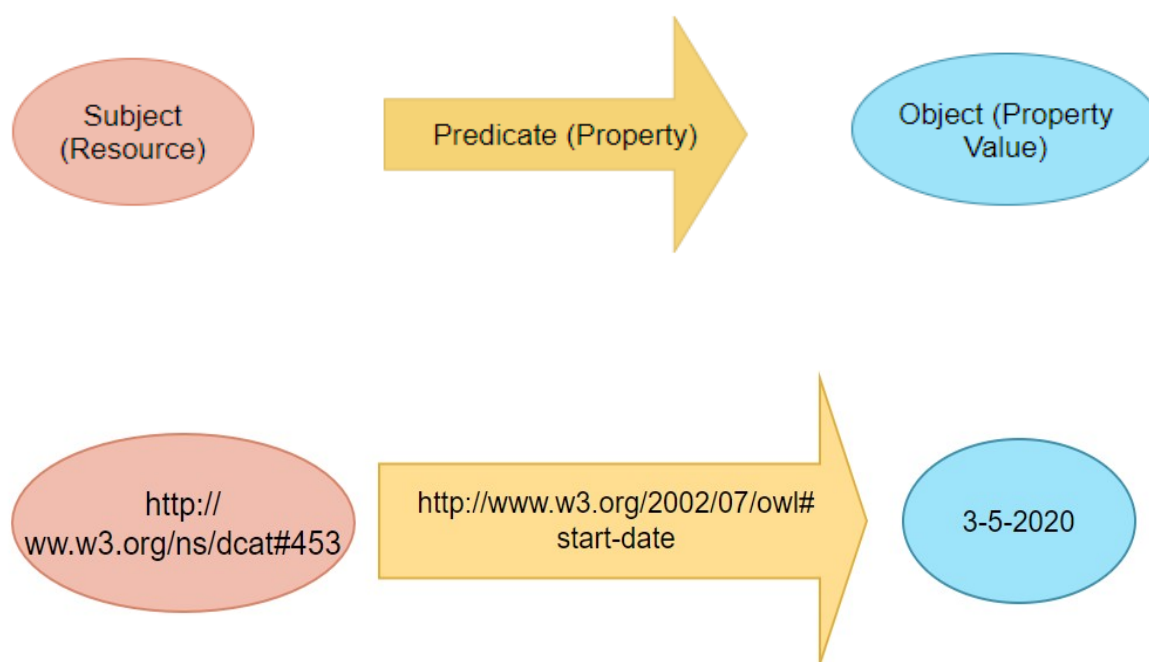


Abbildung 7: Ein projekt Eintrag in RDF

## 4 Anforderungen

Um das Ziel der Arbeit zu erreichen, kommen unterschiedliche Techniken zum Einsatz. Zuerst müssen die erforderlichen Daten gefunden werden, bevor sie in das RDF-Format konvertiert werden können. Für die Suche und die Konvertierung müssen jeweils eigene Methoden geschrieben werden.

Danach ist es einfach, die Daten mit Hilfe von RDFLIB<sup>16</sup> in einem Wissensgraph zu speichern. Der letzte Schritt besteht darin, die Daten aus dem Wissensgraph als Statements (Triples) zu extrahieren und im Frontend zu visualisieren.

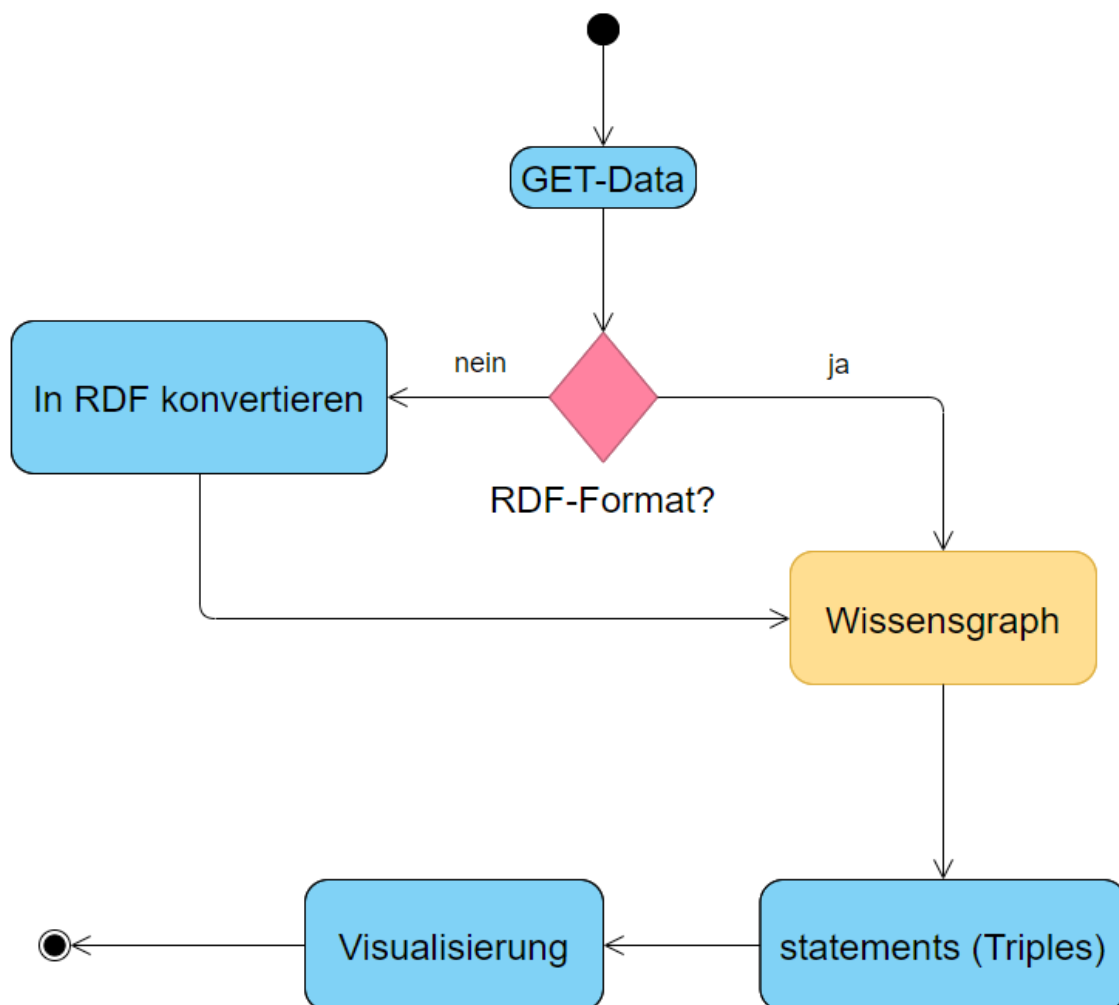


Abbildung 8: Arbeitsprozess

<sup>16</sup>RDFLIB ist ein RDF-JavaScript-Bibliothek, kann auch mit anderen Sprachen wie Python verwendet werden.

Die Bibliotheken, die in dieser Arbeit benötigt werden, sind:

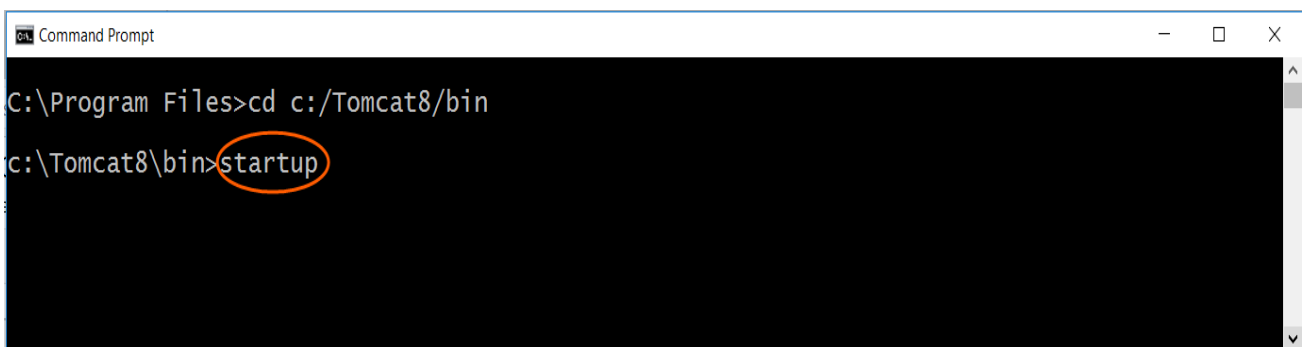
Bibliothek	Version	Zweck
Tomcat	9	Es wird verwendet, um eine Verbindung zu einem Server auf dem lokalen Computer herzustellen, der auf Port 8080 ausgeführt wird
RDFLIB	–	Umgang mit RDF-Daten im Backend
JSON-LD	1.0.0	Es ist in der Lage, Schlüssel-Wert-Paare aus RDF-Daten auszudrücken
Vega	4.4	zum Erstellen interaktiver Visualisierung Designs in Browsern
Leaflet	1.5.1	Ist eine Open-Source-JavaScript-Bibliothek für interaktive Karten
Bootstrap	4.5.0	CSS Framework for developing responsive and mobile-first websites

Tabelle 19: Bibliotheken Tabelle

**Der Apache Tomcat-Server:** ist ein Java-basierter Open Source-Webanwendungscontainer<sup>17</sup>, der zum Ausführen von Servlet- und JavaServer Pages-Webanwendungen<sup>18</sup> (JSP) erstellt wurde. Apache Tomcat ist sehr stabil und verfügt über alle Funktionen Webanwendungscontainers. Tomcat bietet außerdem zusätzliche Funktionen, die es zu einer hervorragenden Wahl für die Entwicklung einer vollständigen Webanwendungslösung machen. Um eine Verbindung zu einem Server auf dem lokalen Computer zu erstellen, kann der Benutzer nur die startup Datei starten. Genau wie Abbildung 9

<sup>17</sup> URL: <http://tomcat.apache.org/>

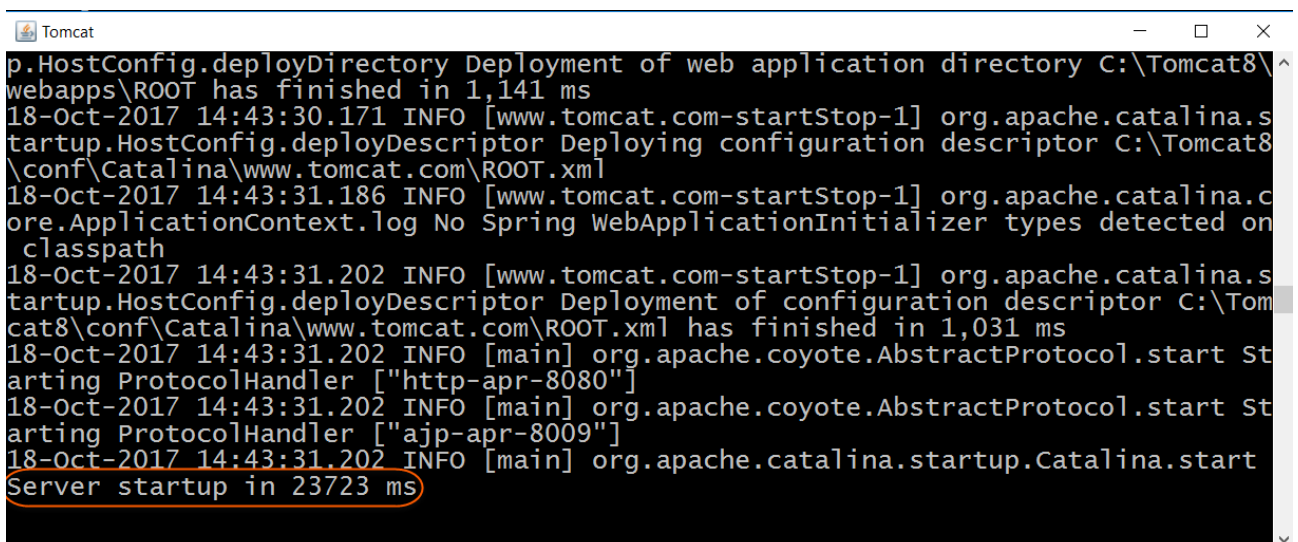
<sup>18</sup> Ein Servlet ist ein kleines Java-Programm, das auf einem Webserver ausgeführt wird.



```
Command Prompt
C:\Program Files>cd c:/Tomcat8/bin
c:\Tomcat8\bin>startup
```

Abbildung 9: Starten des Tomcat-Servers.

Ein separates Fenster wird geöffnet und eine Reihe von Meldungen wird angezeigt, gefolgt von der Meldung, dass der Server gestartet wurde. Sieh Abbildung 10



```
Tomcat
p.HostConfig.deployDirectory Deployment of web application directory C:\Tomcat8\
webapps\ROOT has finished in 1,141 ms
18-Oct-2017 14:43:30.171 INFO [www.tomcat.com-startStop-1] org.apache.catalina.s
tartup.HostConfig.deployDescriptor Deploying configuration descriptor C:\Tomcat8
\conf\Catalina\www.tomcat.com\ROOT.xml
18-Oct-2017 14:43:31.186 INFO [www.tomcat.com-startStop-1] org.apache.catalina.c
ore.ApplicationContext.log No Spring WebApplicationInitializer types detected on
classpath
18-Oct-2017 14:43:31.202 INFO [www.tomcat.com-startStop-1] org.apache.catalina.s
tartup.HostConfig.deployDescriptor Deployment of configuration descriptor C:\Tom
cat8\conf\Catalina\www.tomcat.com\ROOT.xml has finished in 1,031 ms
18-Oct-2017 14:43:31.202 INFO [main] org.apache.coyote.AbstractProtocol.start St
arting ProtocolHandler ["http-apr-8080"]
18-Oct-2017 14:43:31.202 INFO [main] org.apache.coyote.AbstractProtocol.start St
arting ProtocolHandler ["ajp-apr-8009"]
18-Oct-2017 14:43:31.202 INFO [main] org.apache.catalina.startup.Catalina.start
Server startup in 23723 ms
```

Abbildung 10: Tomcat Server wurde gestartet.

Die genaue Anzahl der Millisekunden hängt von der Anzahl der bereitgestellten Webanwendungen ab. Um den Tomcat-Server zu stoppen, kann der Benutzer nur shutdown eingeben.

## 4.1 Abfragen mit SPARQL

Dieser Teil widmet sich der Suche nach den erforderlichen Daten mithilfe von SPARQL. Grundlegende Informationen zu SPARQL wurden bereits im Abschnitt 2.2.4. beschrieben. Das EU Open Data Portal (EU ODP) bietet Menschen die



Möglichkeit nach Daten zu suchen, auch mittels SPARQL<sup>19</sup>. Es gibt viele Techniken, um nach Daten zu suchen. Der Autor wählt SPARQL anstelle einer zufälligen Suche in EU ODP, da SPARQL es ermöglicht, die Daten besser zu filtern. Die Ergebnisse geben dem Suchenden einen direkten Link zu den gewünschten Daten, abhängig davon, wie die Suche eingeschränkt wurde, z.B. auf Organisationen und Projekte. Mit dieser Art von Such-Einschränkung kann man jeden Datensatz-Titel im EU-Datenbank durchsuchen. Wenn der Suchende konkrete Werte abgleichen möchte, z.B. Datumsangaben oder StringText, dann verwendet man den Befehl FILTER.

In der nächsten Abfrage wird der Titel jeden Datensatzes geprüft, ob dieser bestimmte Wörter Enthält.

```
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX odp: <http://data.europa.eu/euodp/ontologies/ec-odp#>
PREFIX dc: <http://purl.org/dc/terms/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT * WHERE {
  graph ?Graph
  {
    ?DatasetURI a <http://www.w3.org/ns/dcat#Dataset>;
    dc:title ?DatasetTitle.
    FILTER(regex(?DatasetTitle, " research projects "))
  }
} LIMIT 10
```

Tabelle 20: SPARQL-Abfrage für einen bestimmten Datensatz

Die letzte Abfrage liefert die Datensätze, die die Schlüsselwörter im Datensatz-Titel enthalten. Eines der 10 Ergebnisse ist der Link zu den Daten, nach denen gesucht wurde. Im nächsten Schritt müssen die im CSV-Format vorliegenden Suchergebnisse in das RDF-Format konvertiert werden. Dazu wurde eine eigenständige Methode geschrieben, siehe Kapitel 5.

Um die Daten in einem Wissensgraph zu speichern muss der der Autor spezielle Anforderung erfüllen und verwendet dazu die Bibliothek RDFLIB, welchen nachfolgend beschrieben wird.

<sup>19</sup> SPARQL-Backend, der Abfragen der RDF-Beschreibungen von Datasets ermöglicht, URL: <https://data.europa.eu/euodp/en/linked-data>

## 4.2 RDFLIB

Zur Vereinfachung der Arbeit mit dem RDF-Format wird zur Unterstützung die Javascript-Bibliothek RDFLib genutzt. RDFLib hilft, einen Datenspeicher (Wissensgraphspeicher) zu erstellen. RDFLib dient zum Arbeiten mit RDF-Daten. RDFLib ermöglicht das schneller Abrufen, Parsen und Serialisieren von RDF-Daten. Parsen ist der Prozess der Analyse einer Folge von Symbolen, entweder in natürlicher Sprache oder in Datenstrukturen. Bei der Serialisierung wird ein Objekt in einen Bytestrom konvertiert, um das Objekt zu speichern oder in den Speicher, eine Datenbank oder eine Datei zu übertragen. Der Wissensgraphspeicher enthält RDF-Graphs. Jeder Wissensgraph ist ein Triple. Mit diesen Optionen kann das Wissensgraph-Speicher Objekt mit SPARQL abgefragt oder bestimmte Tripel hinzugefügt oder gelöscht werden. RDFLIB bietet auch die Möglichkeit, die Daten im Speicher zu verwenden. Diese Daten werden danach gefiltert und den Benutzern in Grafiken angezeigt. Es gibt auch ein RDFLib-Modul für Python. man sollte die RDFLib Python-Bibliothek nicht mit der Javascript-Bibliothek mischen. In Abschnitt 5 werden die Schritte zum Erstellen eines Speicher für den Graph erklärt, um die XML / RDF-Daten darin zum Speichern. RDFLIB sortiert jedes Graph und fügt es im Statement Objekt ein, das aus Tripeln besteht.

## 4.3 JSON-LD

Nach der Speicherung der Daten mithilfe von RDFLib wird JSON-LD (JavaScript Object Notation for Linked Data) benutzt um zur Visualisierung zu gelangen. JSON steht für Javascript-Object Notation und wurde bereits im Abschnitt 2.1.3 erklärt. Es wird verwendet, um Daten zwischen Websites und Browsern zu übertragen. Der Zusatz "Linked Data" steht für die Verknüpfung von Daten im Netz, ähnlich eines Hyperlinks. Beispielsweise befinden sich unterschiedliche HTML(Hypertext Markup Language)-Dokumente, die sich auf einem Webserver befinden. HTML-Dokumente können Bilder, Videos oder Text enthalten. Zur Verknüpfung dieser Daten können Hyperlinks<sup>20</sup> verwendet werden, um die einzelnen Element zu verknüpfen. Das hilft dem Leser, schneller relevante Daten bz. Informatione zu finden. Abbildung 11 macht diese Verknüpfung von Daten nochmal deutlich.

---

<sup>20</sup> Weitere Informationen zum Hyperlink: <https://en.wikipedia.org/wiki/Hyperlink>

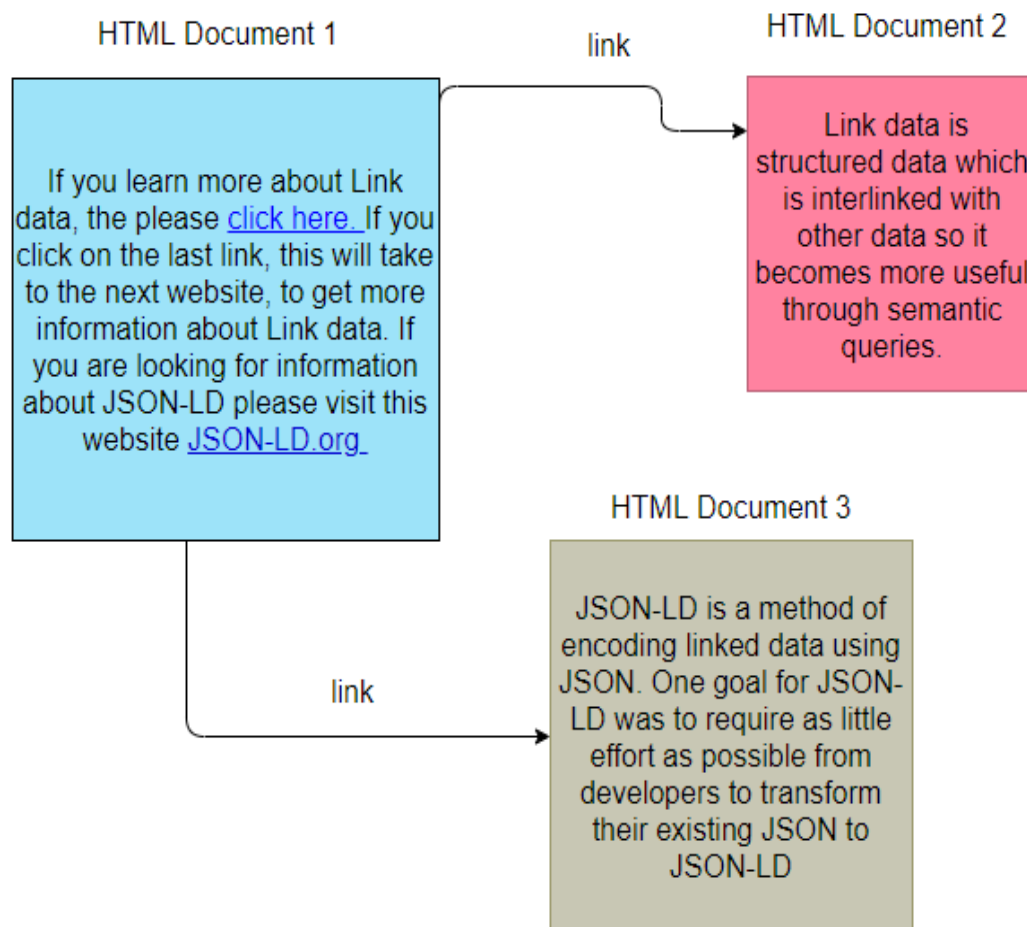


Abbildung 11: Hyperlink zwischen Webseiten

Die Verknüpfung mit Hyperlinks findet im Frontend statt. Im Backend wird JSON genutzt, um Daten bereitzustellen. Websites sind in zwei Teile unterteilt. Es gibt das Frontend und Backend<sup>21</sup>. Wie bereits erwähnt, ist es sinnvoll, Daten zwischen mehreren Websites zu verknüpfen, was als Linked Data beschrieben wird. Wie bereits in Abschnitt 2.1.2 erklärt, ist RDF wichtig für Suchmaschinen und Netzwerkseiten. Während Benutzer im Frontend die HTML-Seiten sehen, "sehen" Computer bzw. Programme die Daten, aus denen die Website besteht. Und bei genau diesen Backend-Daten kommt JSON bzw. JSON-LD ins Spiel. JSON-LD bringt Standardmethoden zum Ausdrücken von Linked Data in JSON mit. Es ist bekannt, dass Websites durch Request-Data miteinander interagieren. Nehmen wir an, dass Daten von mehreren Websites empfangen werden, siehe Abbildung 12. Es ist nicht klar, ob beide Websites das Objekt "Name" auf dieselbe

<sup>21</sup> Frontend und Backend: Frontend ist was in dem man im Grunde das HTML und Javascript im Web sieht, und es gibt eine andere Seite der Website namens Backend, die normalerweise dazu gedacht ist, Daten in Form von JSON bereitzustellen.

Weise verwenden. Wenn dafür nur JSON benutzt wird, führt das zu einem Mehrdeutigkeitsproblem. Wie in Abbildung 12 zu sehen, haben zwei unterschiedliche Objekte die gleiche URL. In diesem Fall man weiß es nicht, ob der Objektname "mostafa" der normale Name oder der Anmeldename auf dieser Website ist.

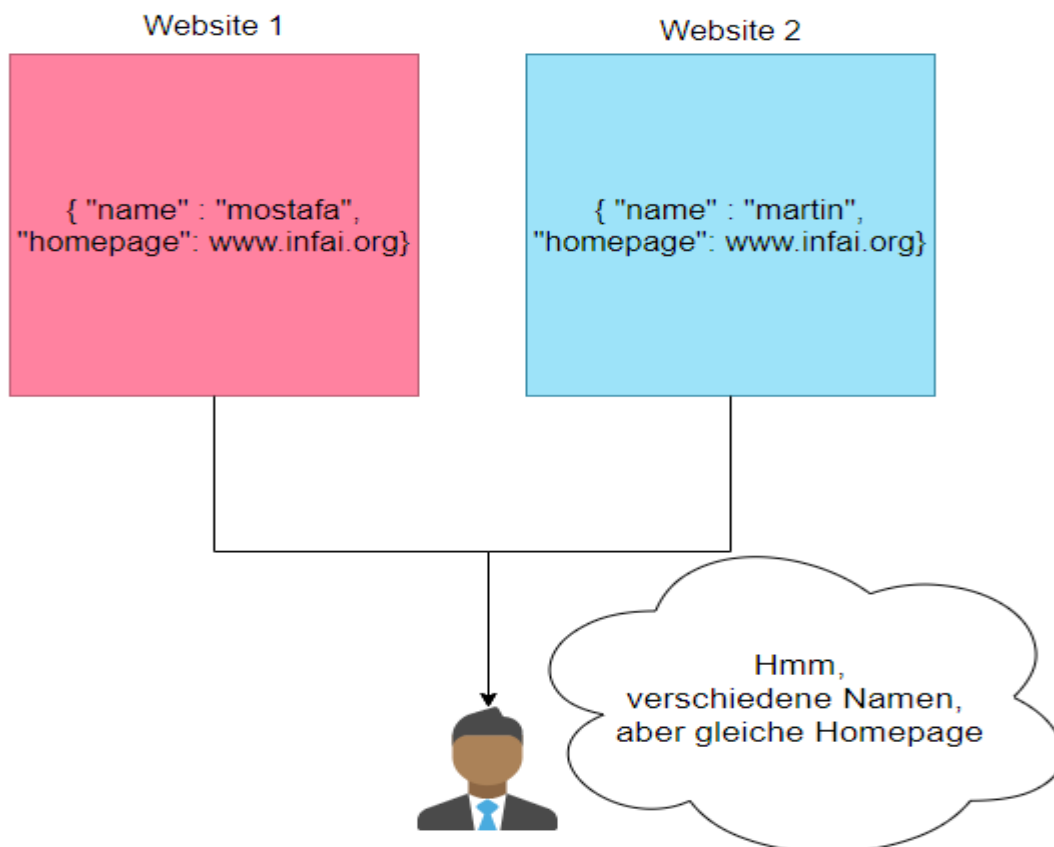


Abbildung 12: Daten von vielen Websites

Zur eindeutigen Kennzeichnung der Daten wird deshalb in diesem Fall ein URI verwendet, siehe Tabelle 21.

```
{
  "http://www.w3c.org/name": "mostafa",
  "http://www.example.org/homepage": "www.infa.org"
}
```

Tabelle 21: Eindeutige Kennzeichnung von Daten durch URI

Da die genannte eindeutige Kennzeichnung mit URIs sehr umständlich beim Programmieren ist, wird JSON-LD als unterstützende Bibliothek eingesetzt. JSON-LD führt ein sehr einfaches Konzept ein, das als `@context` bezeichnet wird. `@context` teilt der Application mit, wie der Rest der Daten im Dokument interpretiert werden soll. `@context` bezeichnet damit eine URL zu einem Dokument im Web oder lokal. Dieses verknüpfte Dokument definiert eindeutig, was "Name" und "Homepage" sind und welchen Datentyp sie angehören, z.B. String oder Integer. Mit JSON-LD kann man also Objekte eindeutig identifizieren. Neben `@context` wird auch `@id` eingesetzt um Daten mit JSON-LD zu identifizieren. Auch hier hilft `@context` bei der Interpretation von Objekten wie "Name", "Homepage". "`@id`" teilt mit, sagt uns, dass dies die universelle Kennung dieser Daten im Web ist. Mit anderen Worten, im selben Beispiel ist `@id` ein Bezeichner. Wenn also jemand über "mostafa" sprechen möchte, kann er diesen Bezeichner `@id` verwenden

```
{
  "@context": "https://json-ld.org/contexts/person.jsonld",
  "@id": "http://dbpedia.org/resource/mostafa",
  "name": "mostafa",
  "born": "1993-01-01",
  "homepage": "http://dbpedia.org/resource/mostafa"
}
```

Tabelle 22: JSON-LD Beispiel<sup>22</sup>

Schließlich können die JSON-LD-Funktionen wie folgt zusammengefasst werden :

1. Es gibt einen Dokumenten-Kontext `@context`.
2. Durch die Verwendung kurzer Begriffe (`@context` und `@id`) wird die Datenabfrage vereinfacht
3. Es können zusätzlich URIs zur Kennung von Daten genutzt werden

## 4.4 Visualisierung

Daten können auf viele verschiedene Arten visualisiert werden. Daten können in einer Tabelle visualisiert werden oder auch als Koordinaten in einer Karte, als Grafik etc. Es gibt viele Bibliotheken in Javascript um die Daten zu visualisieren. In dieser Arbeit wird die bekannte Bibliothek Vega<sup>23</sup> eingesetzt, da sie ein Vielzahl an Visualisierungsmöglichkeiten bietet.

### 4.4.1 Vega

Vega ist eine deklarative Sprache zum Erstellen, Speichern und Teilen interaktiver Visualisierung Designs. Vega ist eine Javascript-Bibliothek, die viele Visualisierung

<sup>22</sup> <https://json-ld.org/>

<sup>23</sup> Vega ist eine Javascript-Bibliothek, die interaktive Grafiken bietet. <https://vega.github.io/vega/>

Designs bietet, die JSON-Daten zur Visualisierung verwenden. Vega bietet Online-Editor für jedes Design mit Quellcode. Mit Vega ist es für Entwickler einfach, Daten online zu visualisieren und unterschiedliche Funktionen auszuprobieren. Bedingt durch die großen Datenmengen und der Beziehungen zwischen ihnen, ist es wichtig die passende Grafik für die Anzeige de Daten auszuwählen.

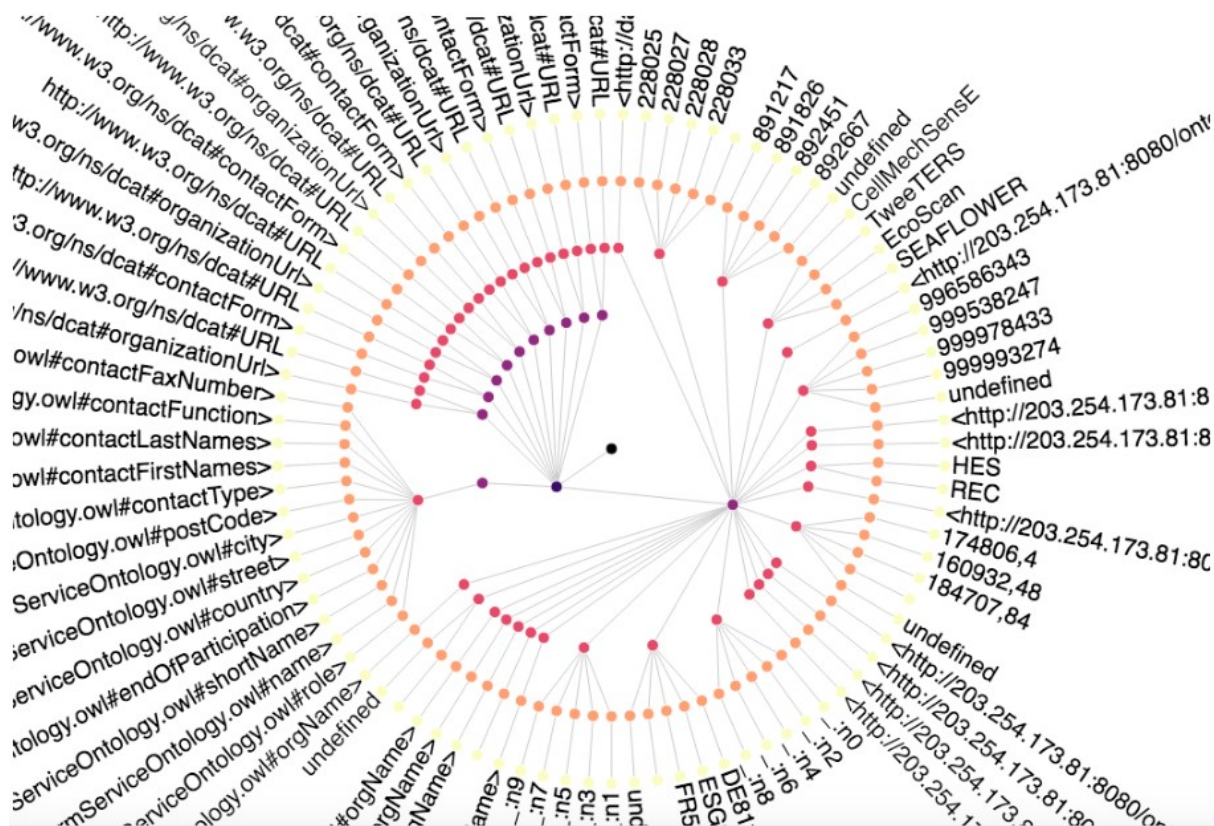


Abbildung 13: Mit Vega erstelltes Baum Design zeigt Subjekte, Prädikate und Werte.

In Abbildung 13 ist eine mit Vega erzeugte Datenvisualisierung zu sehen. Alle Teile der Daten-Triples - Subjekte, Prädikate, Objekte - sind klar zu erkennen. Die roten Kreise bilden die Subjekte ab, die orange farbigen Kreise die Prädikat und die gelben Kreise die Objekt bzw. Werte. Die Werte können auch URIs sein, wie man an dem äußeren Kreis erkennt. Wie bereits erwähnt, gibt es zahlreiche andere Grafiken, um Daten zu filtern und grafisch anzuzeigen. Im nächsten Beispiel wird visualisiert, welche Stadt in Deutschland die meisten Organisationen hat, die von der europäischen Union finanziert werden. Abbildung 14 zeigt ein Design, das von Vega bereitgestellt wird und mit den gesuchten Daten übereinstimmt.



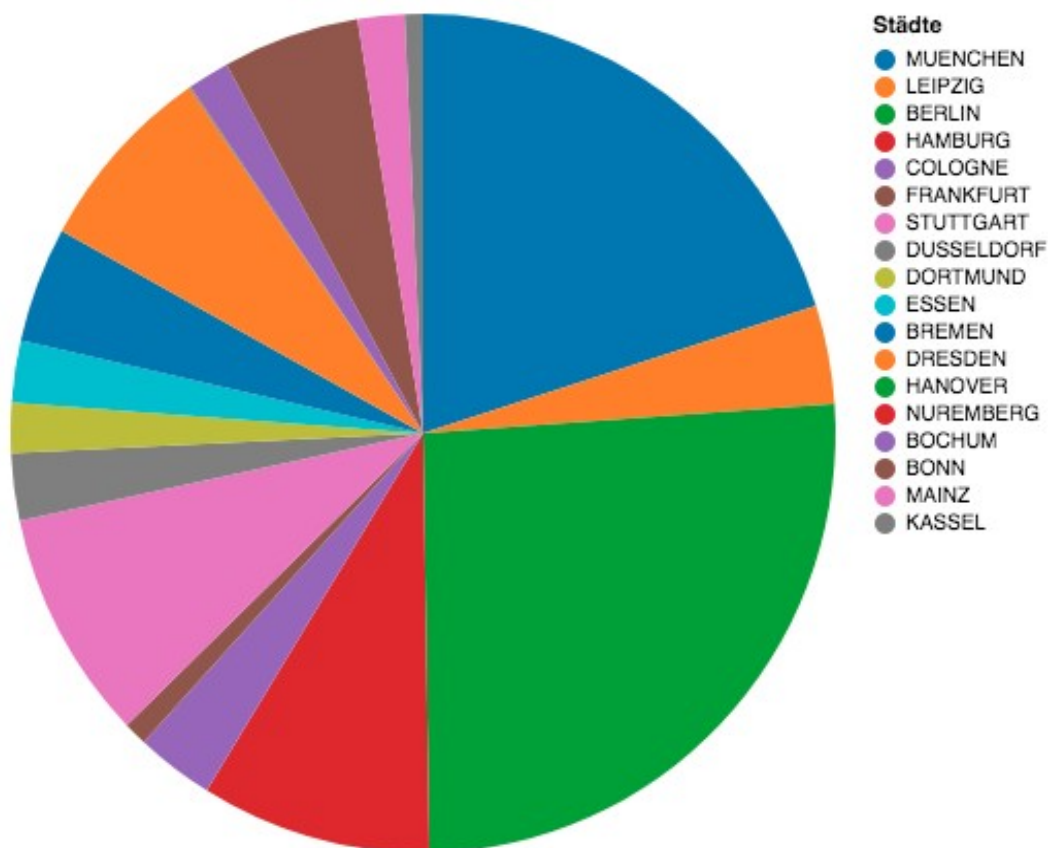


Abbildung 14: Pie Design aus Vega; zeigt die deutschen Städte, deren Organisationen von der Europäischen Union finanziert werden.

Wie es im letzten Abschnitt erklärt wurde, werden über RDFLIB alle Daten-Triples in einem Objekt abgespeichert, was als Statement bezeichnet wird. Diese Statements sind allerdings direkt im JSON-Format abgelegt und werden auch so an Vega zur Visualisierung übergeben, da Vega nur JSON-Daten akzeptiert.

#### 4.4.2 Leaflet

Leaflet<sup>24</sup> ist die führende Open-Source-JavaScript-Bibliothek für mobil freundliche interaktive Karten. Leaflet bieten Weltkarte mit vielen Funktionen können hinzugefügt werden. um eine Linie durch die Karte zu erstellen oder um Punkte miteinander zu verbinden. Es bietet auch die Möglichkeit, Punkte auf der Karte darzustellen. In unserem Fall hilft uns dies, alle Organisationen auf der Karte zu finden. Der Benutzer kann nur mit der Maus hinein- und herauszoomen und den Ort klar ablesen. Die in der vorliegenden Arbeit benutzten Daten enthalten Informationen über die Standorte von Organisationen. Diese Daten können in einer Tabelle oder auf einer Karte angezeigt werden. Durch die Verwendung der

<sup>24</sup> URL: <https://leafletjs.com/>

Weltkarte kann man den Standort der Organisationen besser anzeigen, wofür Leaflet eingesetzt wird.

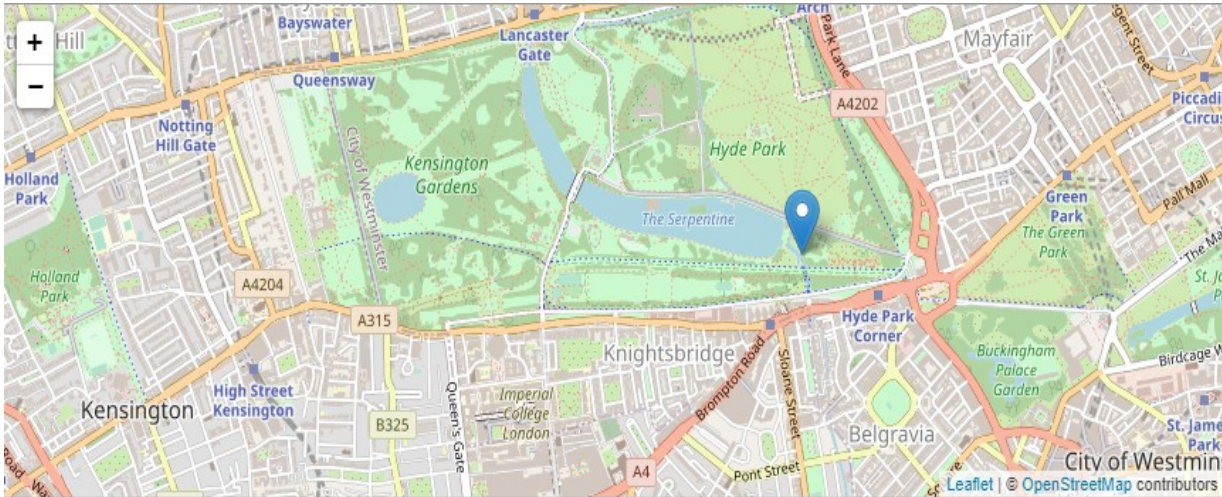


Abbildung 15: Standort punkt liegt in London.

Der Datensatz, der vom EU-ODP bereitgestellt, enthält die Adressen der Organisationen wie Postleitzahl, Straße, Stadt und Land. Diese Adressen der Organisationen reichen leider nicht aus, um den Standort zu finden, da man zwingend die Koordinaten mit X und Y benötigt.

#### 4.4.3 Data Tables

“DataTables ist ein Plug-In für Javascript. Es ist ein hochflexibles Tool, das auf den Grundlagen der progressiven Verbesserung basiert und all diese erweiterten Funktionen zu jeder HTML-Tabelle hinzufügt”<sup>25</sup> [11]. Durch die Verwendung von DataTables kann man die Daten als json angeben und mit DataTables Funktionen kann jemand die Daten nach Spaltennamen sortieren. DataTables können Knöpfe (Buttons) wie Suchen, Hinzufügen, Aktualisieren und Löschen hinzugefügt werden. Der große Vorteil von DataTables besteht darin, Daten aus vier verschiedenen grundlegenden Quellen zu erhalten. DataTables ist so einfach zu bedienen. Nachdem jemand die erforderlichen Bibliotheksdateien heruntergeladen hat, kann mit nur wenigen Codezeilen ein erstaunliches Ergebnis erzielt werden. Wie Tabelle 23.

25 URL: <https://datatables.net/>



```
$(document).ready( function () {
    $('#myTable').DataTable();
} );
```

Tabelle 23: Data Tables

Aus Tabelle 23 kann man das Ergebnis wie in Abbildung 16 ansehen. Hier wird eine Sucheingeabe bereitgestellt und noch der Index der Seiten. Die Daten können nach Bedarf eingeschränkt werden.

Show  entries

Search...

Firstname	Lastname	Position	Office	Age	Startdate	Salary	extn	email
Airi	Satou	Accountant	Tokyo	33	2008/11/28	\$162700	5407	a.satou@datatables.net
Angelica	Ramos	Chief Executive Officer (CEO)	London	47	2009/10/09	\$1200000	5797	a.ramos@datatables.net
Ashton	Cox	Junior Technical Author	San Francisco	66	2009/01/12	\$86000	1562	a.cox@datatables.net
Bradley	Greer	Software Engineer	London	41	2012/10/13	\$132000	2558	b.greer@datatables.net
Brenden	Wagner	Software Engineer	San Francisco	28	2011/06/07	\$206850	1314	b.wagner@datatables.net
Brielle	Williamson	Integration Specialist	New York	61	2012/12/02	\$372000	4804	b.williamson@datatables.net
Bruno	Nash	Software Engineer	London	38	2011/05/03	\$163500	6222	b.nash@datatables.net
Caesar	Vance	Pre-Sales Support	New York	21	2011/12/12	\$106450	8330	c.vance@datatables.net
Cara	Stevens	Sales Assistant	New York	46	2011/12/06	\$145600	3990	c.stevens@datatables.net
Cedric	Kelly	Senior Javascript Developer	Edinburgh	22	2012/03/29	\$433060	6224	c.kelly@datatables.net

Showing 1 to 10 of 57 entries

Previous **1** 2 3 4 5 6 Next

Abbildung 16: Data Tables<sup>26</sup>

Daten Tabeles kann auch weitere Funktionen wie das Drucken der Tabellendaten oder die Verwendung von Datentabellen-Knöpfe zum Speichern der Daten als PDF oder CSV bereitstellen. Sieh Abbildung 17

<sup>26</sup> URL: <https://datatables.net/>

Below are client-side buttons demo, go here to see [server-side buttons demo](#)

	<input type="checkbox"/>	ID <small>↑↓</small>	Actions	Name <small>↑↓</small>	Username	Email <small>↑↓</small>	Address1 <small>↑</small>
+	<input type="checkbox"/>	1	<input type="button" value="Edit"/> <input type="button" value="Delete"/>	Leanne Graham	Bret	Sincere@april.biz	Kulas Light, Apt. 556
+	<input type="checkbox"/>	2	<input type="button" value="Edit"/> <input type="button" value="Delete"/>	Ervin Howell	Antonette	Shanna@melissa.tv	Victor Plains, Suite 879
+	<input type="checkbox"/>	3	<input type="button" value="Edit"/> <input type="button" value="Delete"/>	Clementine Bauch	Samantha	Nathan@yesenia.net	Douglas Extension, Suite 841
+	<input type="checkbox"/>	4	<input type="button" value="Edit"/> <input type="button" value="Delete"/>	Patricia Lebsack	Karianne	Julianne.OConner@kory.org	Hoeger Mall, Apt. 692
+	<input type="checkbox"/>	5	<input type="button" value="Edit"/> <input type="button" value="Delete"/>	Chelsey Dietrich	Kamren	Lucio_Hettinger@annie.ca	Skiles Walks, Suite 351
+	<input type="checkbox"/>	6	<input type="button" value="Edit"/> <input type="button" value="Delete"/>	Mrs. Dennis Schulist	Leopoldo_Corkery	Karley_Dach@jasper.info	Norberto Crossing, Apt. 950
+	<input type="checkbox"/>	7	<input type="button" value="Edit"/> <input type="button" value="Delete"/>	Kurtis Weissnat	Elwyn.Skiles	Telly.Hoeger@billy.biz	Rex Trail, Suite 280
+	<input type="checkbox"/>	8	<input type="button" value="Edit"/> <input type="button" value="Delete"/>	Nicholas Runolfsdottir V	Maxime_Nienow	Sherwood@rosamond.me	Ellsworth Summit, Suite 729
+	<input type="checkbox"/>	9	<input type="button" value="Edit"/> <input type="button" value="Delete"/>	Glenna Reichert	Delphine	Chaim_McDermott@dana.io	Dayna Park, Suite 449
+	<input type="checkbox"/>	10	<input type="button" value="Edit"/> <input type="button" value="Delete"/>	Clementina DuBuque	Moriah.Stanton	Rey.Padberg@karina.biz	Kattie Turnpike, Suite 198

Showing 0 to 0 of 0 entries

Show  entries

Abbildung 17: Data Tables Speichern-Knöpfe<sup>27</sup>

<sup>27</sup> URL: <https://vuejsexamples.com/vue-jquery-datatables-net-wrapper-component/>

## 5 Implementierung

Dieser Abschnitt gliedert sich in drei Teile:

- **Datenanalyse und Transformation:** In diesem Abschnitt wird die Konvertierung der Daten anhand eines Beispiels erläutert.
- **Speichern der Daten im Wissensgraph:** Durch Verwendung von RDFLIB können die Daten in einem Wissensgraph gespeichert werden, um die Tripel bereitzustellen.
- **Visualisierung der Daten im Frontend:** Daten können in vielen Designs visualisiert werden, um dem Benutzer das klare Lesen der Daten zu ermöglichen.

### 5.1 Datentransformation

Dies ist der Hauptprozess dieser Arbeit. Es geht um die Methode zum Konvertieren der Daten vom CSV-Format in das RDF-Format, wie auch bereits in Abschnitt 2.1.1 zum Format CSV erläutert. Mit dem gleichen Beispiel wird nachfolgend der Prozess der Transformation beschrieben.

employee-id,employee-name,salary 1,Tom,2500 2,Toni,2300 3,Jack,2600
--

Tabelle 24: Beispiel über das Gehalt der Mitarbeiter als CSV-Data

Wie bei allen CSV-Daten steht in der ersten Zeile der Tabellenkopf mit den Spaltenbezeichnung. Jede Spalte ist mit Komma (,) voneinander getrennt. Es kann auch mit Semikolon (;) getrennt werden (wie auch die Untersuchungsdaten dieser Arbeit). Unter den Bezeichnungen der Tabellenköpfe folgen die Werte, ebenfalls getrennt durch Komma oder Semikolon. In den CSV-Daten werden zusätzlich Namespaces hinzugefügt, um später nicht qualifizierte XML-Tags unterscheiden zu können. Für die Namespaces ist es besser, wenn diese ausreichend lang und unterschiedlich sind. Sie werden den nicht qualifizierten XML-Tags vorangestellt. Um Namenskonflikte zu vermeiden, wird eine Abkürzung des Namespace als zusätzliches Präfix eingesetzt. Die Daten werden Zeile für Zeile gelesen. Es ist wichtig, zu wissen, welches die erste Zeile ist, das die erste Zeile immer das Prädikat eines Triple bestimmt. So folgt das Subjekt auf das Prädikat und danach folgen die Werte (Objekte). Mit einer normalen Mapping-Funktion kann man jede Zeile in einem eigenen Array speichern. Auf diese Weise kann gezielt auf bestimmte Datenelemente, wie etwa auf den Head-Tag zugegriffen werden. Beispielsweise werden in dieser Arbeit die Head-Tags über ein spezielles Array für Headern erreicht, das erste Daten-Element.

Die erste Operation in der Abfrage ist die Entfernung der Trennungszeichen zwischen den Elementen, in unserem Fall die Entfernung der Kommas. Damit erhält man alle Head-Tags und kann diese in einem neuen Array speichern. Das wird erreicht durch die Programmierzeile in Tabelle 25.

```
var Headings = Data[0].split(',');
```

Tabelle 25: Head-Tags Array

In unserem Beispiel enthält das Array alle Tabellenköpfe der Datenquelle (CSV).

```
["employee-id", "employee-name", "salary"]
```

Tabelle 26: Head-Tags Array Werte

Nach den Tabellenköpfen kann sich den Werten gewidmet werden. Dafür wird eine FOR-Schleife auf die übrigen Zeilen - die Zeilen mit den Werten - angewendet. Da die Tabellen in unserem Fall die Zeile 0 belegen, beginnt die FOR-Schleife bei Zeile 1. Wie im ersten Schritte werden die Kommas zwischen den Werten entfernt und ein neues Daten-Array mit dem Variablennamen Details angelegt.

```
for (var i = 1; i < Data.length; i++) {  
    var Details = Data[i].split(',');  
}
```

Tabelle 27: FOR-Schleife für die Werte

In unserem Beispiel enthält das Array Details alle Werte (Objekte).

```
["1", "Tom", "2500"]  
["2", "Toni", "2300"]  
["3", "Jack", "2600"]
```

Tabelle 28: Arrays enthalten die Werte

Jetzt hat man zwei Arrays bzw. zwei Variablen, die Headern und die Werte. Im nächsten Schritte werden die RDF-Daten erstellt. Zuerst definiert man eine weitere Variable, die die RDF/XML-Tags und die Namespaces enthält. Unter Verwendung der FOR-Schleife wurden Header und Werte hinzugefügt, damit RDF-Daten erstellt werden können. Vor jedem Header muss ein Präfix stehen. Dieses Präfix ist die Abkürzung für die Namespaces, die man zuerst definiert hat. Das Prädikat ist unser Spaltenname. Entweder fügt man dem Prädikat einen URI hinzu

oder man nutzt einfach den Prädikat-Wert. Die Abfolge der Schritte wird in Tabelle 29 deutlich.

```
<?xml version="1.0"?>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:si="https://www.w3.org/2002/07/owl#"
  xmlns:dcate="https://www.w3.org/ns/dcate#"

  <rdf:Description rdf:about="https://www.w3.org/2002/07/owl#1">
    <si:employee-id>1</si:employee-id>
    <dcate:employee-name>Tom</dcate:employee-name>
    <si:salary>2500</si:salary>
  </rdf:Description>

  <rdf:Description rdf:about="https://www.w3.org/2002/07/owl#2">
    <si:employee-id>2</si:employee-id>
    <dcate:employee-name>Toni</dcate:employee-name>
    <si:salary>2300</si:salary>
  </rdf:Description>

  <rdf:Description rdf:about="https://www.w3.org/2002/07/owl#3">
    <si:employee-id>3</si:employee-id>
    <dcate:employee-name>Jack</dcate:employee-name>
    <si:salary>2600</si:salary>
  </rdf:Description>

</rdf:RDF>
```

Tabelle 29: Transformation zu RDF-Daten

Damit ist die Konvertierung der CSV-Daten zu den benötigten RDF-Daten abgeschlossen. Jetzt kann der Wissensgraph-Speicher für die RDF-Daten erstellt werden.

## 5.2 Wissensgraph mit RDFLIB

Nach der Datentransformation soll ein Wissensgraph-Speicher für die RDF-Daten erstellt werden. Die RFLIB-Bibliothek bietet die Möglichkeit, RDF-Daten in einem Wissensgraph zu speichern. Wissensgraph-Speicher nimmt Daten auf und sortiert sie in viele Objekte. Eines dieser Objekte sind die verwendeten Namespaces. Ein anderes Objekt können zum Beispiel andere Statements sein, wie etwa die Triple. Für die Erstellung eines neuen, leeren Wissensgraph-Speicher wird folgende Operation genutzt:

```
var store = $rdf.graph();
```

Tabelle 30: Wissensgraph-Speicher erstellen

Um die RDF-Daten zum Wissensgraph-Speicher hinzuzufügen, muss man die Daten zunächst analysieren. Die Bibliothek RFLIB gibt dazu die Möglichkeit, Daten zu parsen. RFLIB wird mitgeteilt, welche Daten analysiert werden müssen, um welche Art von Daten es sich handelt und in welchem Speicher die Daten nach dem Parsen abgelegt werden sollen. Dies alles kann in einer Programmierzeile geschrieben werden, Tabelle 31 zeigt es alles.

```
$rdf.parse(data, store, baseURI, 'application/rdf+xml')
```

Tabelle 31: Parse Data im Wissensgraph-Speicher

Der erste Parameter "data" bezieht sich auf unsere RDF-Daten. Der Parameter "store" ist der Wissensgraph-Speicher, der erstellt wurde (siehe Tabelle 30). Der Parameter "baseURI" ist der Namespace der RDF-Ressourcen. Der letzte Parameter "application/rdf+xml" ist der Datentyp, in welchem die Daten am Ende im Wissensgraph-Speicher abgelegt werden sollen. Jetzt ist der Wissensgraph-Speicher mit den Daten gefüllt. Diese werden im Statement-Objekt als Triples gespeichert. Die Daten im Statement-Objekt sind bereits für jeden Triple sortiert und bestehen aus Subjekt, Prädikat und Objekt. Man kann auf die Daten aus dem Wissensgraph-Speicher zugreifen, indem man eine FOR-Schleife für alle Triples im Statement-Objekt verwendet. Jetzt kann man die Daten verwenden, um sie im Frontend zu visualisieren.

## 5.3 Visualisierung

Die Daten, die man aus dem Statement-Objekt visualisieren möchte, bestehen aus vielen Triples. Nachdem man die Daten in RDF konvertiert hat - wie in Tabelle 29 gezeigt - werden diese Daten jetzt für die Visualisierung genutzt. Jeder XML-Tag aus unseren RDF-Daten ist aufgebaut wie in Abbildung 18:

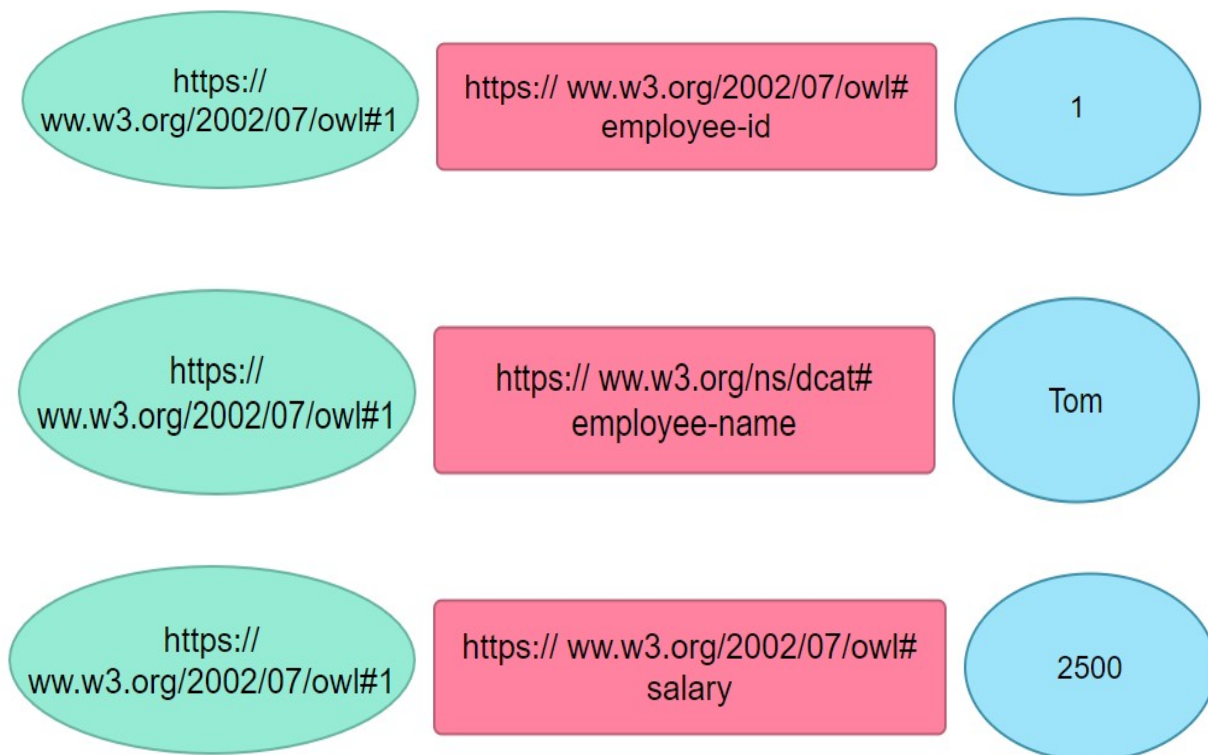


Abbildung 18: Visualisierung eines XML-Tag 1

Die grünen Entitäten sind die Subjekte, die roten Entitäten sind die Prädikate und die blauen Entitäten sind die Werte (Objekte). Die Beziehungen zwischen diesen Entitäten wird in Abbildung 19 dargestellt.

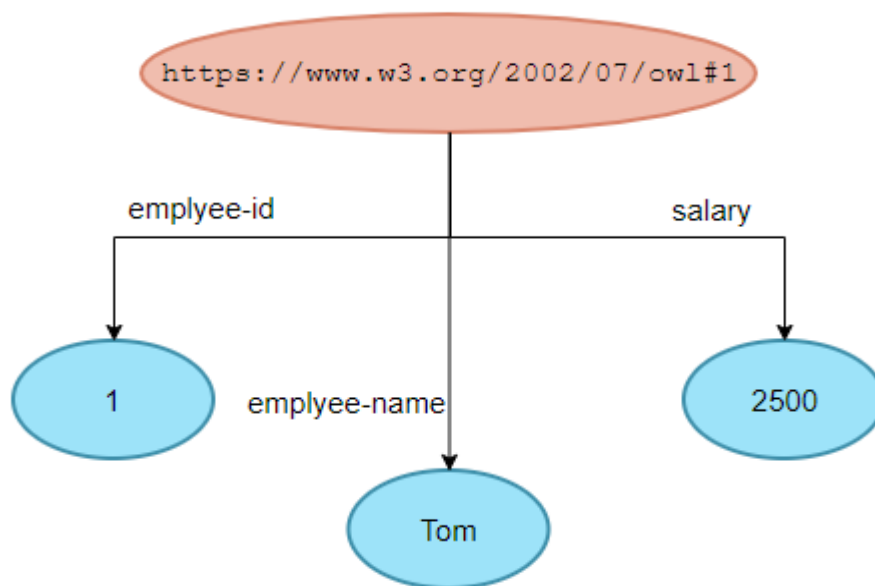


Abbildung 19: Visualisierung ein RDF/XML Tag 1 als Baum

Nach dem gleichen Prinzip können nun auch alle anderen Daten visualisiert werden. Diese Visualisierung der Beziehung zwischen den Daten bietet eine einfache Erklärung der Daten im Netz. Ein Beispiel für diese Vereinfachung ist die Suche im Internet nach einem Unternehmen in einer bestimmten Stadt. Auf diese Weise kann eine Visualisierung helfen, zusätzlichen Informationen zu diesem Unternehmen und dieser Stadt zur Verfügung zu stellen. In im speziellen Fall der vorliegenden Arbeit kann eine solche Visualisierung deutlich machen, welche Unternehmen es gibt, die von der Europäischen Union in einer bestimmten Stadt oder einem bestimmten Land finanziert werden.

### 5.3.1 Kompakte Daten mit JSON-LD

Es wurde bereits erwähnt, dass die Daten in zwei Datenquellen aufgeteilt sind: Organisationen und Projekte. Man kann entweder beide Datenquellen getrennt visualisieren oder die beiden Datenquellen können zusammen komprimiert visualisiert werden - was das Ziel dieser Arbeit ist. Für diesen Fall bietet die Bibliothek JSON-LD eine Funktion, die beide Datenquellen verwendet und eine Kombination beider Quellen erstellt. Für die Kombination ist es wichtig, die unterschiedlichen Datenquellen genau zu kennen. Beide Datenquellen, sowohl Organisationen als auch Projekte, verwenden eine identische Spalte, die Projekt-ID. Diese gemeinsame Projekt-ID wird zu einer Quelle zusammengefasst. Diese



Kombination erweitert den Wissensgraphen und gibt jeder Entität mehr Informationen, wie Abbildung 20 und Abbildung 21 deutlich machen.

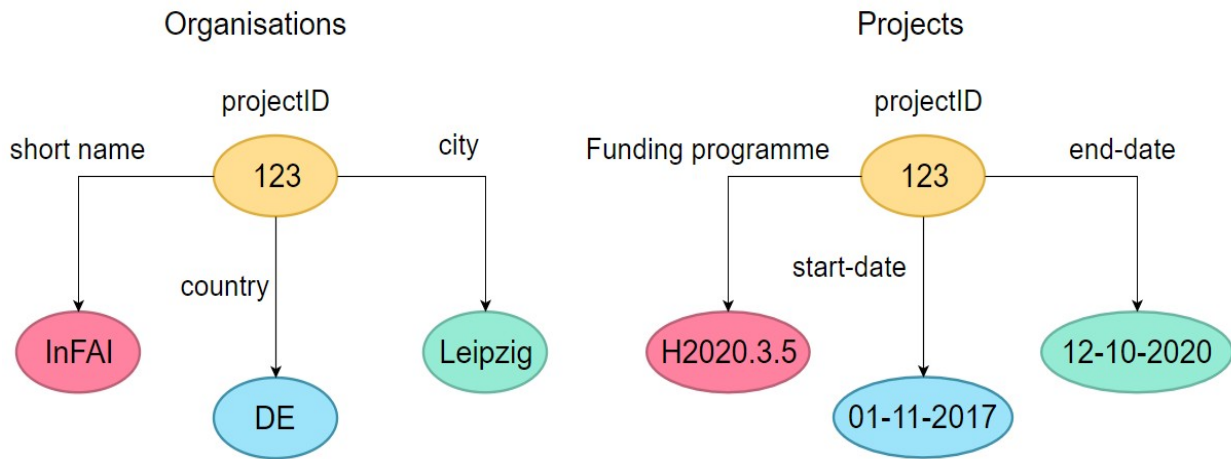


Abbildung 20: Datenquellen vor dem Komprimieren

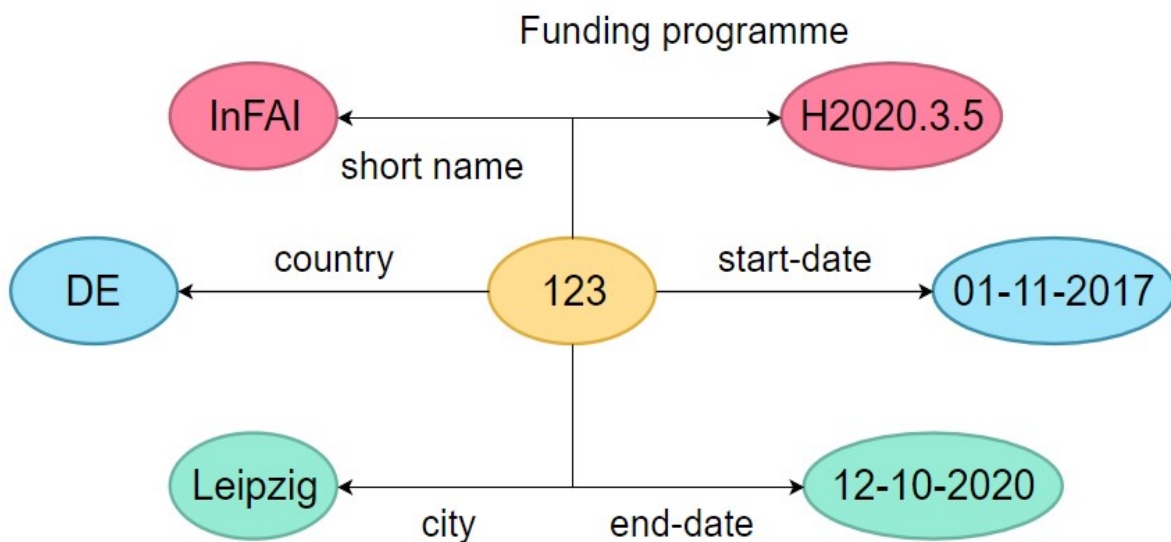


Abbildung 21: Datenquellen nach dem Komprimieren

### 5.3.2 Visualisierung JSON-LD Data

JSON-LD arbeitet nahtlos mit RDF-Bibliotheken zusammen. Dies ermöglicht mehr Optionen für den Umgang mit dem RDF-Datensatz. Der RDF-Datensatz ist eine Sammlung von RDF-Wissensgraphen und ein Wissensgraph besteht aus einer Reihe von RDF-Tripeln. JSON-LD verwendet den Wissensgraph-Speicher, der von RDFLIB oder andere RDF-Bibliotheken erstellt wurde und kann mit hilfreicher Funktion die Daten in N-Quads<sup>28</sup> konvertieren. N-Quads ist das Format zum Codieren eines RDF-Datensatzes. Dann wird es durch Übergeben der N-Quads-Daten in JSON-LD konvertiert. Wie in Abschnitt 4.3 erläutert, ist das Ergebnis ein JSON-LD-Objekt, das `@context` und `@id` enthält, um alle Triplets zu identifizieren. Mit einigen FOR-Schleifen können die Daten gefiltert und sortiert und dann visualisiert werden.

### 5.3.3 Weltkarte mit Leaflet

Es ist von Vorteil, eine Karte zu verwenden, um die Standorte der Organisationen darauf zu lokalisieren. So können Interessenten das gesamte Gebiet und auch die nahe gelegenen Organisationen sehen. Bei Verwendung von Leaflet werden allerdings die konkreten Koordinaten eines Ortes benötigt, um die Organisationen auf der Karte zu lokalisieren. Die Koordinaten sollte die horizontale Achse haben und wird als x-Achse bezeichnet. Die vertikale Achse wird als y-Achse bezeichnet. Der Punkt, an dem sich die x-Achse und die y-Achse schneiden, ist der Ort der Organisation. Die in der vorliegenden Arbeit genutzten Daten enthalten jedoch keine Koordinaten, sondern Straßename, Postleitzahl, Stadt und Land. In dieser Situation können allerdings Leaflet-Plugins verwendet werden. Diese helfen, die gesamte Adresse in Koordinaten umzuwandeln. Dazu wird ein Array erstellt, das in jedem Index die gesamte Adresse enthält, und diese an das Leaflet-Plugin übergibt. Das Plugin generiert dann aus dieser Adresse die genauen Koordinaten des Ortes. Danach kann der Standort der Organisationen leicht auf der Karte gefunden werden.

---

28 <https://www.w3.org/TR/n-quads/>

## 6 Evaluation

In diesem Kapitel wird das Ergebnis der Implementierung vorgestellt. Das zu erwartende Ergebnis und das tatsächliche Ergebnis der Implementierung werden diskutiert. Es wird geprüft, wie schnell Daten angefordert werden, ob die Konvertierung zu RDF Fehler verursachen kann und ob wirklich alle Daten visualisiert werden können. Mit diesen Fragestellungen werden die einzelnen Schritte des Prozesses nachfolgend beleuchtet.

### 6.1 Daten anfordern

Die Anforderung der Daten von CORDIS konnte erfolgreich durchgeführt werden. Organisationsdaten mit einer Größe ca. 40 MB und die Projektdaten mit einer Größe von ca. 67 MB wurden angefordert. Durch die genannte Größe der Daten wurde mehr Zeit für die Anforderung der Daten benötigt, als initial gedacht. Das Lesen der gesamten Organisationsdaten mit dem Google Chrome-Browser dauerte ca. 540 ms und für die Projektdaten ca. 910 ms. Es kam trotz der hohen Anfragezeit zu keinen timeouts.

### 6.2 Erfolgreiche Datenkonvertierung

Die Konvertierung der Organisations-Daten und Projekt-Daten, welchem im CSV-Format angefordert wurden, wurde erfolgreich durchgeführt. Als Ergebnis lagen RDF-Daten vor, wobei die Namespaces und das Präfix in den ersten Zeilen der Daten definiert wurden. Einige Werte wurden aus den Projekt-Daten nicht definiert. Das bedeutet, dass diese Werte mit "NULL" definiert wurden, was grundsätzlich vermieden werden sollte.

DCAT-Vokabular-Namespaces wurden verwendet und vor jedem Header ein Präfix hinzugefügt, um die Daten zu beschreiben. Wenn man die gesamten Organisations-Daten in das RDF-Format konvertiert, erhöht sich die initiale Größe von 67 MB auf 335 MB erhöht. Die Konvertierung selbst dauerte ca. 50 Sekunden. Der Konvertierungsprozess der Projekt-Daten vom CSV-Format in das RDF-Format dauerte ca. 30 Sekunden.

### 6.3 RDFLIB Speicher

Wie in den letzten Abschnitten erläutert wurde, müssen die Daten nach der Umwandlung in das RDF-Format in einen Wissensgraph-Speicher abgelegt werden. Nach der Konvertierung wurden die Daten erfolgreich in einen neu generierten Wissensgraph-Speicher abgelegt. Es wurden sowohl die Organisations-Daten als auch die Projekt-Daten abgespeichert. Der Prozess des Speicherns der Organisations-Daten dauerte ca. 7 Minuten. Damit lag dieser

Prozess zeitlich auch stark über der initial geplanten Zeit für diesen Prozess (anfordern, konvertieren, speichern). Es ist zu beachten, dass hier eine relativ große Datenmenge verarbeitet wird (335 MB) und der Leistungsgrad des genutzten Computers ebenfalls eine Rolle bei der Be- und Verarbeitungszeit der Daten spielt. Als Ergebnis des Speicherns der Organisations-Daten in dem Wissensgraph-Speicher wurden 763296 Statements (Triples) erstellt. Jedes dieser 763296 Statements (Triples) enthält Subjekt, Prädikat und Objekt, was für die folgende Visualisierung von Bedeutung ist.

## 6.4 Von Wissensgraph-Speicher zu JSON-LD

Die Verarbeitung der gesamten konvertierten Daten würde mit der aktuellen Größe von 335 MB sehr lange dauern und im Zweifel sogar zu timeouts führen. Um das zu vermeiden, werden nur Teile der Daten verarbeitet. Mit dieser erfolgreichen Bearbeitung von Teil-Daten der gesamten Datenmenge können die erhaltenen JSON-LD-Daten optimal für die Visualisierung genutzt werden, z.B. für die Vega-Baumstruktur. Die Teil-Daten können direkt an Vega gesendet werden, für unterschiedliche Designs. Wenn an dieser aber zu viele FOR-Schleifen genutzt werde, um die Daten an Vega zu geben, kann es ebenfalls zu Problemen kommen (Browser-Crash, Timeout).

## 6.5 Auswertung der Visualisierung

Wie in Kapitel 5.2 dargelegt, werden FOR-Schleifen für alle Triples im Statement-Objekt verwendet, um die Daten im Wissensgraph abzufragen. Obwohl der Prozess der Erstellung des Wissensgraph-Speichers nur 7 Minuten dauerte, kam es bei der Verwendung mehrerer FOR-Schleifen zu Abstürzen des Prozesses. Deshalb wurde daraufhin nur ein Teil der Daten in den Prozess aufgenommen. Diese Teil-Daten konnten ohne Komplikationen verarbeitet werden. Die Visualisierung wurde erfolgreich in einer Baumstruktur durchgeführt, siehe Abbildung 13. Die Visualisierung kann auch mit anderen Vega-Designs wie etwa in Abbildung 14 erfolgen. Für alternative Visualisierungen sind genauere Filter notwendig, was durch den Einsatz weiterer FOR-Schleifen problematisch sein kann (Timeouts).

## 6.6 Visualisierung mit Leaflet

Bei der Durchführung der in Abschnitt 5.3.3 beschriebenen Schritte ist die Verwendung eines Arrays notwendig, welches alle Adressen aller Organisationen enthält. Das Array wurde erfolgreich erstellt. Bei der Konvertierung der Daten in Koordinaten kam es zu zwei Komplikationen. Das erste Problem war, dass einige der Adressen nicht vollständig waren oder der Server sie nicht in Koordinaten konvertieren konnte. Das zweite Problem war, dass bei der Abfrage der Adressen von allen Organisationen zu viele Requests gesendet werden müssen, was dazu

führte, dass die Server weitere Requests blockiert wurden. An dieser Stelle war es ebenfalls hilfreich nur einen Teil der Daten zu verarbeiten. Wenn man bei der Abfrage nur einen Teil der Daten verwendet, können diese erfolgreich visualisiert werden. Es können natürlich auch bei diesen Teilmengen fehlerhafte bzw. unvollständige Adressdaten enthalten sein.

## 7 Zusammenfassung

Zusammenfassend lässt sich sagen, dass der Einsatz von Wissensgraph eine gute Methode ist, um Informationen bzw. Daten über die realen Welt zu visualisieren und deren Verbindungen untereinander deutlich zu machen.

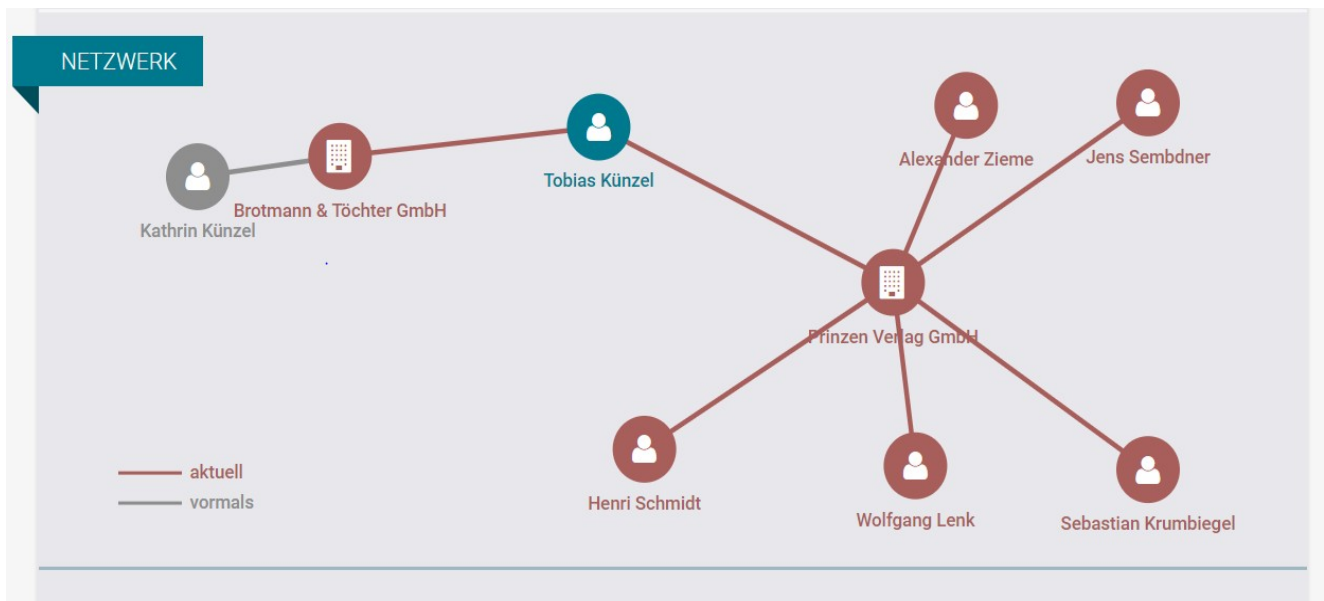
Wie die Auswertung dieser Arbeit zeigt, lassen sich Daten zu europäischen Organisationen und Projekten gut verarbeiten um diese am Ende visuell abzubilden und Interessenten deren Verbindungen untereinander deutlich zu machen.

Beispielsweise haben die Organisation unterschiedliche Standorte, nach denen gesucht werden kann. Diese Arbeit kann noch erweitert werden, indem nicht nur der Standort der Organisation auf der Karte angezeigt wird, sondern auch die Höhe der Finanzierung in einer bestimmten Stadt. Um Probleme zu vermeiden, die in Abschnitt 6.6 über Leaflet aufgetreten sind, kann eine alternative Visualisierung verwendet werden. Die alternative Visualisierung anstelle der Weltkarte (Leaflet) ist DataTables. Wenn jemand eine Tabelle zur Visualisierung der Daten verwenden möchte, ist DataTables die beste Wahl. DataTables wurden bereits in Abschnitt 4.4.3 erläutert. Im Datensatz befinden sich Spalten für Postleitzahl, Straße, Stadt und Land. Mit dataTables kann jemand nach Organisationen in einer bestimmten Stadt oder einem bestimmten Land suchen. Um die lange Wartezeit zu vermeiden, bis der Browser die Daten visualisiert. Die Daten können mithilfe von North Data<sup>29</sup> auf mehreren Ebenen visualisiert werden. Wie Abbildung 22. "North Data analysiert Handelsregisterbekanntmachungen und andere Pflichtveröffentlichungen deutscher Firmen, um Wirtschaftsinformationen zu gewinnen, insbesondere zu finanziellen Kennzahlen und zu Zusammenhängen zwischen Firmen untereinander sowie zu Personen. Dazu werden Methoden der Big-Data-Verarbeitung und der Künstlichen Intelligenz (KI) verwendet." [15]

Wissensgraph entspricht der Beschreibung der Ontology, den Triples sind miteinander verwandt und der Graph kann konstruiert werden. In unserer Wissensgraph Darstellung verwenden wir anstelle von URIs zur Bezeichnung von Knoten Symbole und Beschriftungen (Labels). Auf diese Weise kann der Benutzer die angeforderten Informationen einfacher erkennen als alle URIs, die normalerweise weniger anschaulich sind. Der Benutzer kann auf die Beschriftungen (Labels) oder Symbole klicken, um die zweite Ebene zu öffnen. In diesem Fall lädt der Browser die erforderlichen Daten separat auf jede Ebene. Dadurch ist der Visualisierungsprozess viel schneller als das gleichzeitige Laden der gesamten Daten.

---

<sup>29</sup> URL: <https://www.northdata.de/>

Abbildung 22: North Data<sup>30</sup>

EU ODP bieten viele Datensätze, diese Daten sind ein Schatz für Programmierer, die sich darauf freuen, die Daten zu verfeinern und zu visualisieren, um für die Gesellschaft nützlicher zu sein. In Zukunft können diese Daten der Vorsprung sein, um ein großes Wissensgraph über alle Wissenschaftsorganisationen auf der ganzen Welt zu erstellen. Diese Informationen aus dem EU ODP können Menschen dabei helfen, herauszufinden, wonach sie suchen, z. B. Projekte, Berichte, Organisationen, Finanzierungsprogramme usw., und diese Daten können in Grafiken angezeigt werden, die das Filtern und Sortieren der Daten vereinfachen. Genauso wie ist diese Arbeit darüber geht.

30 URL: <https://www.northdata.de/K%C3%BCnzel,+Tobias,+P%C3%B6nitz/1aat>

## 8 Verzeichnisse

### 8.1 Abkürzungsverzeichnis

<b>CORDIS</b>	Community Research and Development Information Service
<b>RDF</b>	Resource Description Framework
<b>XML</b>	Extensible Markup Language
<b>CSV</b>	Comma Separated Values
<b>JSON</b>	JavaScript Object Notation
<b>EU ODP</b>	European Union Open Data Portal
<b>URL</b>	Uniform Resource Locator
<b>URI</b>	Uniform Resource Identifier
<b>SPARQL</b>	Sparql Protocol And Rdf Query Language
<b>W3C</b>	World Wide Web Consortium
<b>KG</b>	Knowledge Graph
<b>REST</b>	Representational State Transfer
<b>API</b>	Application Programming Interface
<b>JSON-LD</b>	JavaScript Object Notation for Linked Data
<b>DCAT</b>	Data Catalog Vocabulary
<b>JSP</b>	JavaServer Pages



## 8.2 Tabellenverzeichnis

Tabelle 1: Beispiel über das Gehalt der Mitarbeiter.....	8
Tabelle 2: Daten im CSV-Format.....	8
Tabelle 3: Beispiel über das Gehalt der Mitarbeiter.....	10
Tabelle 4: Namespace.....	10
Tabelle 5: JSON Objekt.....	11
Tabelle 6: H2020-Projekte Data.....	12
Tabelle 7: H2020-Organisationen Data.....	12
Tabelle 8: Einige DCAT Namespaces und Präfixe.....	14
Tabelle 9: Prefix beispiel.....	16
Tabelle 10: Jeden Eintrag ist ein Triple.....	16
Tabelle 11: Triple-Muster/Bestandteil.....	17
Tabelle 12: SPARQL-Abfrage.....	17
Tabelle 13: SPARQL-Abfrage Ausgabe.....	17
Tabelle 14: SPARQL-Abfrage mit zwei Triple-Muster.....	18
Tabelle 15: Die Ausgabe von SPARQL-Abfrage mit zwei Triple-Muster.....	18
Tabelle 16: API Dokumentation.....	19
Tabelle 17: GET-Anfrage für Projekte.....	19
Tabelle 18: Projekte Tabelle.....	21
Tabelle 19: Bibliotheken Tabelle.....	23
Tabelle 20: SPARQL-Abfrage für einen bestimmten Datensatz.....	25
Tabelle 21: Eindeutige Kennzeichnung von Daten durch URI.....	28
Tabelle 22: JSON-LD Beispiel.....	29
Tabelle 23: Data Tables.....	33
Tabelle 24: Beispiel über das Gehalt der Mitarbeiter als CSV-Data.....	35
Tabelle 25: Head-Tags Array.....	36
Tabelle 26: Head-Tags Array Werte.....	36
Tabelle 27: FOR-Schleife für die Werte .....	36
Tabelle 28: Arrays enthalten die Werte.....	36
Tabelle 29: RDF/XML Daten.....	37
Tabelle 30: Wissensgraph-Speicher erstellen.....	38
Tabelle 31: Parse Data im Wissensgraph-Speicher.....	38

## 8.3 Abbildungsverzeichnis

Abbildung 1: Data Visualisierung.....	6
Abbildung 2: Ein Bild zeigt, das Subjekt, prädikat und den Wert.....	9
Abbildung 3: Triple beispiel.....	14
Abbildung 4: URI, um eine Entität zu definieren.....	15
Abbildung 5: Mehr Triples.....	15
Abbildung 6: Beispiel über Wissensgraph.....	20
Abbildung 7: Ein projekt Eintrag in RDF.....	21
Abbildung 8: Arbeitsprozess.....	22
Abbildung 9: Starten des Tomcat-Servers.....	24

Abbildung 10: Tomcat Server wurde gestartet.....	24
Abbildung 11: Hyperlink zwischen Webseiten.....	27
Abbildung 12: Daten von vielen Websites.....	28
Abbildung 13: Mit Vega erstelltes Baum Design zeigt Subjekte, Prädikate und Werte.....	30
Abbildung 14: Pie Design aus Vega; zeigt die deutschen Städte, deren Organisationen von der Europäischen Union finanziert werden.....	31
Abbildung 15: Standort punkt liegt in London.....	32
Abbildung 16: Data Tables.....	33
Abbildung 17: Data Tables Speichern-Knöpfe.....	34
Abbildung 18: Visualisierung ein RDF/XML Tag 1.....	39
Abbildung 19: Visualisierung ein RDF/XML Tag 1 als Baum.....	40
Abbildung 20: Datenquellen vor dem Komprimieren.....	41
Abbildung 21: Datenquellen nach dem Komprimieren.....	41
Abbildung 22: North Data.....	47

## 8.4 Quellenverzeichnis

- [1] **ontotext**, what is a knowledge graph?, Wissensgraph Foto URL: <https://www.ontotext.com/knowledgehub/fundamentals/what-is-a-knowledge-graph/>, (besucht am 19. 04. 2020)
- [2] **W3.org**, Extensible Markup Language (XML), (besucht am 02. 05. 2020), URL: <https://www.w3.org/XML/>
- [3] **W3schools**, Beispiel über JSON Objekt. (besucht am 23. 04. 2020) URL: [https://www.w3schools.com/js/js\\_json\\_objects.asp](https://www.w3schools.com/js/js_json_objects.asp)
- [4] **Bob DuCharme**. Learning SPARQL. Publisher: O'Reilly Media, Kap. 1-2, S. 1-60, ISBN: 1449371434,9781449371432
- [5] **W3.org**, Data Catalog Vocabulary, (besucht am 12. 04. 2020), URL: <https://www.w3.org/TR/vocab-dcat-2/>
- [6] **yashueth.blog**, (besucht am 14. 04. 2020), Foto über Wissensgraph URL: <https://yashueth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/>
- [7] **Mayank Kejriwal**, Domain-Specific Knowledge Graph Construction (2019), Publisher: Springer International Publishing, S. 1-10, ISBN: 978-3-030-12374-1,978-3-030-12375-8
- [8] **poolparty.biz**, What is a Knowledge Graph? (besucht am 24. 04. 2020), URL: <https://www.poolparty.biz/what-is-a-knowledge-graph/>
- [9] **data.europa.eu**, REST API, (besucht am 29. 03. 2020) ,URL: <https://data.europa.eu/euodp/en/developerscorner>
- [10] **Gang Wu**, Semantic Web, In-Tech intechweb.org, Published by In-Tech 2010, ISBN 978-953-7619-54-1
- [11] **Datatables** URL: <https://datatables.net/> , (besucht am 3. 06. 2020)
- [12] **Miguel-Angel Sicilia, Miltiadis D. Lytras**, Metadata and semantics,(2008), Publisher: Springer, ISBN: 038777744X,9780387777443

- [13] **Jeffrey Pomerantz**, Metadata, The MIT Press Essential Knowledge series. 2015, ISBN 978-0-262-52851-1
- [14] **NISO**, Understanding Metadata, (besucht am 5. 06. 2020), URL: <https://www.niso.org/publications/understanding-metadata-2017>
- [15] **North Data**, URL: [https://www.northdata.de/\\_about](https://www.northdata.de/_about), (besucht am 24. 07. 2020)

## A Anhang

### I Quellcode

Methode, die CSV-Daten in RDF / XML-Daten konvertiert

```
function convert_to_rdf(csvData){

  let Data = csvData.split('\n').map(row => row.trim());

  let headings = Data[0].split(';');

  let rdf = `<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:org="http://www.onem2m.org/ontology/Base_Ontology#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:dcatapop="http://data.europa.eu/88u/ontology/dcatapop#"

xmlns:cfso="http://203.254.173.81:8080/ontologies/ConnectedFarmService
Ontology.owl#"
xmlns:dcat="http://www.w3.org/ns/dcat#">`;

  for(let i = 1; i < Data.length; i++) {

    let details = Data[i].split(';');
    rdf += `<dcatapop:organisation rdf:about="http://203.254.173.81:8080/
ontologies/ConnectedFarmServiceOntology.owl#org">`;

    for(let j = 0; j < headings.length; j++) {

      if(headings[j] == "id" || headings[j] == "ecContribution" || headings[j]
== "vatNumber" || headings[j] == "projectRcn" || headings[j] == "projectID" ||
headings[j] == "contactTelephoneNumber" || headings[j] ==
"contactFaxNumber "){
        rdf += `<owl:${headings[j]}>${details[j]}</owl:${headings[j]}>`;
      }else if(headings[j] == "contactForm" || headings[j] ==
"organizationUrl"){
        rdf += `<dcat:${headings[j]}><dcat:${headings[j]}
```

```

rdf:resource="http://www.w3.org/ns/dcat#URL"/></dcat:${headings[j]}>`;
    }else if(headings[j] == "activityType"){
        if(details[j] != null){
            rdf += `<rdf:${headings[j]}>${details[j]}</rdf:${headings[j]}>`;
        }
    }else if(headings[j] == "projectAcronym"){
        if(details[j] != null){
            rdf += `<dcatapop:${headings[j]} xml:lang="en">${
{details[j]}</dcatapop:${headings[j]}>`;
        }
    }else if(headings[j] == "firstNames" || headings[j] == "lastNames" ||
headings[j] == "contactTitle"){

        if(details[j] != null){
            rdf += `<dcat:${headings[j]}>${details[j]}</dcat:${headings[j]}>`;
        }
    }else{
        if(details[j] != null){
            rdf += `<cfso:${headings[j]}><cfso:${headings[j]}
rdf:about="http://203.254.173.81:8080/ontologies/ConnectedFarmServiceOn
tology.owl#orgName"/></cfso:${headings[j]}>`;
        }
    }
}

rdf += "</dcatapop:organisation>"
}

rdf += "</rdf:RDF>";

graph(rdf);
};

```

## II Github-Repository

Link zum Repository: [https://github.com/banqusli/Knowledge\\_graph](https://github.com/banqusli/Knowledge_graph)

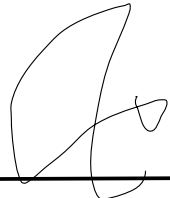
## Selbstständigkeitserklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Leipzig, 07.07.2020

---

Ort und Datum



---

Unterschrift