Datenmodelle verschiedener Leipziger Plattformen im Vergleich

Hans-Gert Gräbe

Version vom 7. August 2021

Inhaltsverzeichnis

1 Vorbemerkungen

Innerhalb einer Stadt wie Leipzig gibt es viele Akteure mit eigenen Portalen. Eine Zusammenführung von Informationen erfolgt teilweise auf der Ebene von Multiplikatoren wie den „Leipziger Ecken“ im Leipziger Osten, die das Zusammenspiel von Akteuren innerhalb einzelner Stadtteile koordinieren und eine gemeinsame Stadtteilplattform[1] betreiben. Weiterhin gibt es stadtweite Plattformen mit speziellen „Sammelgebieten“[2].

Eines der Ziele des *Leipzig Data Projekts*[3] besteht darin, einen **Leipziger Open Data Raum** voranzubringen, indem verschiedene dieser Leipziger Portalanbieter miteinander vernetzt werden. Ein erfolgversprechendes Vorgehen setzt auf dies folgenden Prämissen:

1) Die Partner betreiben eigene Portale zur Datenerfassung und organisieren den dazu erforderlichen techno-sozialen Prozess einschließlich einer gewissen Qualitätssicherung (insbesondere Spamvermeidung) in ihrer jeweiligen Zielgruppe.
2) Über eine Schnittstelle (derzeit präferiert im JSON-Format) wird lesender Zugriff auf einen definierten Teil der Daten gewährt.
3) Diese Schnittstellen werden im Kontext von *Leipzig Data* integriert, entsprechende Metadaten und Dienste entwickelt, um einen einheitlichen *verteilten* Datenraum zu konstituieren, der über diese integrierte Schnittstelle angemessen exploriert werden kann.

Die Schnittstellen der einzelnen Partner greifen üblicherweise auf eine lokale Datenbank zu, welche die lokal verwalteten Zustände (also genaue Informationen über einzelne Akteure, Orte, Events, ... auf Instanzebene) einmal für die eigene Webdarstellung verwaltet und zum anderen für den Austausch vorhält. Jeder solchen Datenbank liegt ein eigenes *Datenmodell* zu Grunde, das im Rahmen der Integrationsbemühungen zunächst zu explizieren und mit den Datenmodellen der anderen Partner zu vergleichen ist.

Mit dieser Zusammenstellung wird ein erster Schritt in dieser Richtung zur Integration verschiedener Leipziger Plattformen gegangen. Die Darstellungen und Vergleiche orientieren sich dabei am Datenmodell von Leipzig Data, da dieses von Anfang an unter dem Blickwinkel der Interoperabilität entwickelt wurde. Das Leipzig Data Datenmodell (LDD) wird im Abschnitt 2 genauer dargestellt. Für die anderen Portale werden die Datenmodelle in Relation zum LDD beschrieben.

Im Weiteren werden neben allgemein gebräuchlichen Namensräumen (rdfs, gsp, foaf, dct, org, ical, owl) die lokalen Namenspräfixe

```
ld: <http://leipzig-data.de/Data/Model/>
le: <http://leipziger-ecken.de/Data/Model#>
nl: <http://nachhaltiges-leipzig.de/Data/Model#>
```

verwendet.

[1] `http://leipziger-ecken.de`.
[2] Etwa `https://nachhaltiges-leipzig.de` oder `https://leipzig.afeefa.de`.
[3] `http://leipzig-data.de/`.

2 Leipzig Data

2.1 Hintergrund

Die *Leipziger Initiative für Offene Daten* ist angetreten, um die Bemühungen zur Etablierung Offener Daten als wesentlichen Teil einer sich entfaltenden Weblandschaft in der Leipziger Region voranzubringen. Kern der Bemühungen ist die Etablierung von *Leipzig Data* als einer signifikanten Menge von Beschreibungen des „Leipziger Lebens", die unter einer freien Lizenz in digital adressierbarer Form als Teil der *Linked Open Data Cloud*[4] öffentlich verfügbar sind.

Im Fokus steht allerdings nicht so sehr *Open Data* als vielmehr *Free Speech*, da wir offene Daten nicht als Selbstzweck begreifen, sondern als Voraussetzung der Selbstermächtigung mündiger Bürger, Freie Rede über die sie betreffenden Angelegenheiten zu führen.

Die Initiative setzt die Aktivitäten von API Leipzig (2008-2012) mit veränderter Schwerpunktsetzung fort und wurde von der Stadt Leipzig in einem Kurzzeitprojekt (Nov. 2012 bis April 2013) im Rahmen ihrer „Open Innovation" Ausschreibung unterstützt.

Leipzig Data betreibt folgende Infrastruktur:

- Webseiten `http://leipzig-data.de` auf Wordpress-Basis[5],
- Einen github Organisationsaccount `https://github.com/LeipzigData` mit den Projekten
  - `RDFData` – RDF-Wissensbasen (als Primärdaten)
  - `Tools` – eine Reihe von Werkzeugen, im Wesentlichen zum Anschauen, wie es gehen könnte (Blick über die Schulter auf die Werkbank),
  - `web` – die Webseiten des Projekts. Der Code kann studiert werden, wenn es um die Einbindung von RDF-Quellen in Websites geht.
  - sowie einer Zahl von weiteren Repos für Teilprojekte.
- Dazu wird eine *Leipzig Ontology* `http://leipzig-data.de/ontology/` entwickelt.
- Unter `http://leipzig-data.de:8890/sparql` wird ein Sparql Endpunkt für Queries auf den Daten angeboten, die in einem Virtuoso basierten RDF-Store (als Sekundärdaten) gehostet sind.
- Unter `http://leipzig-data.de/info` und `http://leipzig-data.de/demo` werden zwei Infoseiten mit Beispielen betrieben, in denen demonstriert wird, wie sich Webseiten aus RDF-Quellen erzeugen lassen. Die Quellen dieser Anwendungen sind im Repo `https://github.com/LeipzigData/web` zu finden.

2.2 Datenmodell

Das Datenmodell ist hier nur bis zu einer groben Granularitätsebene beschrieben. Für weitere Informationen wird auf `http://leipzig-data.de/ontology` verwiesen.

[4] `http://lod-cloud.net/`.

[5] Dies soll in absehbarer Zeit durch github pages abgelöst werden.

Allgemeine Übersicht über den Datenbestand

Kern dieses Datenbestands sind aktuell *Akteure*, *Orte*, *Adressen*, *Treffpunkte* und *Events*, die zusammen ein System von *White Pages* bilden, mit denen Geolokalität auf einheitliche Weise referenzierbar (und damit auch aufeinander beziehbar) wird. Basis dieses Systems sind die aus dem API-Leipzig Projekt und damit letztlich von der Stadt Leipzig übernommenen und weiter aktualisierten über 65 000 Datensätze Adressdaten, von denen über 63 000 mit Geokoordinaten (Übernahme aus Nominatim mit anschließender weiterer Konsolidierung und Qualitätssicherung) versehen sind. Damit lassen sich auch ohne Dienste wie Google Maps Leipziger Orte an Geodaten binden und so auf Karten lokalisieren. Die höhere Qualität gegenüber einer reinen Referenzierung der Geokoordinaten etwa über die Google-API ergibt sich aus der vorgenommenen Disambiguierung, die etwa dem Unterschied zwischen der Verwendung von Rechneradressen und Rechnernamen entspricht, sowie der Möglichkeit, an diese Adress- oder Orts-URIs weitere Information zu binden. Diese White Pages werden vor allem im Leipzig Data Event Projekt fortgeschrieben.

Daneben gibt es Einzelprojekte, mit denen Daten aus verschiedenen Quellen aufbereitet worden sind wie MINT-Orte, Schulen, Polizeidirektion, Seniorenbüros.

RDF-Graphen und Klassen

Adressen und Geodaten der Stadt Leipzig. Eine *Leipziger Adresse* als Instanz der RDF-Klasse `ld:LeipzigerAdresse` ist ein geolokaler Punkt in der Stadt Leipzig mit einer Hausnummer, wo zum Beispiel eine Postzustellung möglich ist. Die Daten (über 65 000 Datensätze) wurden 2012 im Rahmen des API-Leipzig Projekts von der Stadtverwaltung (einschließlich Referenzen auf das Straßenverzeichnis) übernommen, im Rahmen des Leipzig Data Projekts unter Verwendung eines einheitlichen Namensschemas für URIs in das RDF-Format transformiert und in `Adressen.ttl` zusammengefasst.

Hausnummern sind *Grundstücken* zugewiesen, diese können aus mehreren *Flurstücken* bestehen. Verschiedene Gebäude auf einem solchen Grundstück könnten später durch URIs bezeichnet werden, die die Grundstücksadresse als Namenspräfix haben. Damit folgt unsere Bezeichnung der im Kataster der Stadt Leipzig[6].

Im Rahmen des *Linked Geodata Projekts*[7] wurden diese Daten mit Geodaten angereichert. Die Daten wurden initial von Claus Stadler über `nominatim`[8] aus Open Streetmap extrahiert, danach weiter ergänzt und aktualisiert.

Ausgewählte Adressen außerhalb von Leipzig sind in `WeitereAdressen.ttl` nach demselben URI-Schema in einer verkürzten Ontologie erfasst.

Es gibt geolokale Punkte, die nicht durch eine solche Adresse referenziert werden können, wie Treffpunkte, Kinderspielplätze u.ä. Dafür wurde das Konzept *Treffpunkt* eingeführt, das aus einem Bezeichner und weiterer geolokaler Information besteht.

In Anwendungen (etwa den Events) sind die standardisierten Adressen bis auf die Hausnummer aufgelöst. Weitere Informationen sind als Adresszusatz (`ld:hasAddressAddendum`) einzutragen.

[6] `https://www.leipzig.de/bauen-und-wohnen/bauen/liegenschaftskataster`.
[7] `http://aksw.org/Projects/LinkedGeoData.html`.
[8] `https://wiki.openstreetmap.org/wiki/Nominatim`.

Die URI einer solchen Adresse wird nach einheitlichen Prinzipien aus den Informationen (PLZ, Stadt, Straße, Hausnummer) aufgebaut, so dass auch aus anderen Adresssystemen diese URIs generierbar sind, insofern diese vier Datenbestandteile separiert werden können.

Prädikate:

- ld:hasPostCode Literal – Postleitzahl
- ld:hasCity Literal – Stadt
- ld:hasStreet Literal – Straße, in welcher sich die Adresse befindet
- rdfs:label Literal – Bezeichner, etwa "Leipzig, Messering 6"
- gsp:asWKT Literal – Geokoordinaten im WKT-Format "Point(long lat)"
- ld:inOrtsteil ld:Ortsteil – Ortsteil, in welchem sich die Adresse befindet
- ld:inStreetId ld:Strasse – Straße, in welcher sich die Adresse befindet

Orte. Ein *Leipziger Ort* ist ein Ort mit einer *Adresse* und einem *Träger*, der ein genaueres *Profil* hat und wo *Events* veranstaltet werden.

Prädikate:

- dct:modified xsd:date – letzte Modifikation des Datensatzes
- Orga-Literale foaf:mbox, foaf:phone, foaf:homepage
- ld:erreichbar Literal – Erreichbarkeit mit öffentlichem Nahverkehr
- ld:hasAddress ld:LeipzigerAdresse – Adresse
- ld:hasAddressAddendum Literal – Adresszusatz
- ld:hasAnschrift Literal – Anschrift, Postfach oder so, wenn der Ort keine ld:Adresse hat (Beispiel: LSGM)
- ld:hasSupplier org:Organization oder Unterklasse – Träger
- ld:hasTag ld:Tag – Klassifizierung (Konsolidierung der literalen Werte von ld:Art und ld:Bereich)
- ld:contactPerson, ld:engagedPerson ld:Person – am Ort engagierte Personen
- rdfs:label Literal – Bezeichnung
- Inhalts-Literale (alle mehrfach möglich) ld:Arbeitsformen, ld:Art, ld:Auszeichnungen ld:Finanzierung, ld:Hintergrund, ld:Kosten, ld:Kurzinformation, ld:Leistungsangebot, ld:Oeffnungszeiten, ld:Teilnahmebedingungen, ld:Zielgruppe, ld:Zielstellung
- Einordnungsliterale zu einer der städtischen Übersichten: ld:Categories, ld:hasStadtId

Neben dieser Art von Orten existieren auch Stellen in Leipzig, die einen gebräuchlichen Namen haben, aber nur durch Geokoordinaten referenzierbar sind, also zur Kategorie *Treffpunkt* gehören. Dazu gehören *Spielplätze*, die aus Stadtdaten übernommen wurden, die im Rahmen der OK-Lab Initiative `codefor.de/leipzig` (Stand 2014) extrahiert hat, sowie *Haltestellen* des Nahverkehrs, die aus Daten im Open Data Portal der Stadt Leipzig extrahiert wurden.

Akteure und Personen. Synonym wird auch der Begriff *Träger* verwendet. *Akteure* sind natürliche oder juristische Personen mit verschiedenen Aktivitäten. Diese wurden zunächst in einem RDF-Graphen `Traeger.ttl` zusammengefasst, sind aber nun nach verschiedenen

Kriterien in einzelne RDF-Graphen aufgesplittet. Die Ontologie orientiert sich an der `org`-Ontologie, die für die verschiedenen Unterklassen um einzelne Felder erweitert wird.

Juristische Personen sind rechtsfähige Träger verschiedener Aktivitäten und Betreiber von Orten in Leipzig. Eine juristische Person ist eine Unterklasse von `org:Organization` und kann weiteren Klassen wie `ld:Verein`, `ld:Unternehmen` zugeordnet werden, soll aber immer auch `org:Organization` sein, um Inferenzen längs Vererbungshierarchien zu vermeiden. URIs juristischer Personen haben die Gestalt `Data/<OrgForm>/<name>`, wobei `<OrgForm>` auf die Organisationsform hinweist. Damit soll diese Information perspektivisch verfeinert werden.

Personen sind im RDF-Graphen `Personen.ttl` als `foaf:Person` erfasst und in den anderen RDF-Graphen referenziert, etwa über das Prädikat `org:hasMember`.

Über das Prädikat `owl:sameAs` werden Verweise auf dieselben Akteure oder Personen in den Datenbanken von Partnern verwaltet.

Events. Ziel dieses Teilprojekts ist es, eine Infrastruktur aufzubauen, in die Event-Daten in einheitlichem Format aus verschiedenen Quellen und von verschiedenen Akteuren eingespeist werden und der Allgemeinheit zum Gebrauch zur Verfügung stehen.

Die Infrastruktur bietet keinen elaborierten eigenen Service zur Präsentation dieser Event-Daten, sondern überlässt die Zusammenführung mit weiteren Event-Daten, Filterung und Präsentation den Anbietern, die auf diese Infrastruktur zugreifen möchten. Die prinzipiellen Möglichkeiten eines solchen Events Frameworks wurden in mehreren Beispielprojekten demonstriert.

Der primäre Zugriff erfolgt über Sparql-Anfragen auf einen Sparql-Endpunkt, in dem die Event-Daten mit weiteren Daten über Veranstalter und Veranstaltungsorte angereichert und zusammengeführt sind. Das derzeit vorliegende einheitliche Format (aka Protokoll) ist das Ergebnis eines längeren Entwicklungsprozesses. Fragen der Weiterentwicklung des Protokolls sind zwischen interessierten Partnern noch genauer abzustimmen.

Prädikate:

- ld:contactPerson foaf:Person – Ansprechpartner für das Event
- ical:contact Literal – Kontaktinformation als String
- ical:description Literal – Beschreibung des Events
- ld:hatVeranstaltungsort ld:Ort – Ort, an dem das Event stattfindet
- ld:hatTreffpunkt ld:Treffpunkt – Treffpunkt für das Event
- ld:hasAddressAddendum Literal – genauere Bezeichnung innerhalb von ical:location
- ical:summary Literal – kurze Beschreibung (max. 100 Zeichen) des Events
- ical:dtstart, ical:dtend Literale (xsd:date oder xsd:datetime)
- ld:hatVeranstalter org:Organization – Veranstalter des Events
- ical:sentBy foaf:Agent – Quelle der Eventinformation
- ld:hasTag ld:Tag – Tags für das Event
- ld:zurReihe ld:Projekt – Zuordnung zu einer Veranstaltungsreihe
- Weitere Orga-Literale wie ld:Kosten

3 Leipziger Ecken

3.1 Hintergrund

Leipziger Ecken ist eine Stadtteilinitiative, welche die Plattform `https://leipziger-ecken.de` betreibt. Die Plattform basiert auf Drupal und wird im Wesentlichen von Felix Albroscheit gewartet und weiterentwickelt, mit dem zunächst eine RDF-Schnittstelle auf einem Dump der Datenbank entwickelt wurde, die aktuell unter `https://leipziger-ecken.de/Data/` auf der Produktivplattform ausgerollt ist.

Am 19.06.2021 schreibt Felix Albroscheit über eine neu implementierte API:

> Meine Ambitionen gingen bisher dahin, eine vollständig durch das Drupal-Team unterstützte API anzubieten (samt unterstützter Paginierung, Filterung und Sortierung). Aktuell ist dies mit unserer JSON-API der Fall, welche nun auch halbwegs dokumentiert ist:
>
> `https://github.com/Leipziger-Ecken/drupal/blob/master/docu/API.md`
>
> Es gäbe also die Möglichkeit, auf Basis dieser API die Daten nach RDF umzuwandeln und als weiteren Endpunkt anzubieten.

3.2 Schnittstelle

Hier ist eine Übersicht über die einzelnen Teil-APIs, die aus der oben genannten Quelle übernommen wurde

Index	`https://leipziger-ecken.de/jsonapi/`
Akteure/Actors	`https://leipziger-ecken.de/jsonapi/akteure`
Veranstaltungen/Events	`https://leipziger-ecken.de/jsonapi/events`
Bezirke/Districts	`https://leipziger-ecken.de/jsonapi/districts`
Akteurtypen/Actor types	`https://leipziger-ecken.de/jsonapi/akteur_types`
Kategorien/Categories	`https://leipziger-ecken.de/jsonapi/categories`
Schlagwörter/Tags	`https://leipziger-ecken.de/jsonapi/tags`
Zielgruppen/Target groups	`https://leipziger-ecken.de/jsonapi/target_groups`

3.3 Datenmodell (alt)

Im Modell werden die vier Klassen *Akteure*, *Adressen*, *Events* und *Sparten* unterschieden. Die folgenden Ausführungen nehmen Bezug auf die in den Leipzig Data Tools verfügbaren Transformationsskripte[9].

Adressen. Instanzen dieser Klasse enthalten die Textprädikate `plz` (Stadt ist immer Leipzig), `strasse`, `nr`, `adresszusatz`, `gps` (Geokoordinaten) sowie einen Verweis auf den Stadtbezirk (präziser: auf einen der 63 Leipziger Ortsteile). Mit dem Adresszusatz sind die Instanzen

[9] Siehe `https://github.com/LeipzigData/Tools/tree/master/leipziger-ecken/le-rdf`.

also nicht ganz identisch mit `ld:LeipzigerAdresse`. Im Transformationsskript werden daraus (syntaktisch korrekte Vorschläge für) Instanzen von `ld:LeipzigerAdresse` generiert. Der Verweis auf den Ortsteil ist redundant, da aus einer LDD Leipziger Adresse der zugehörige Ortsteil ermittelt werden kann.

Akteure. In dieser Klasse sind Informationen zur juristischen Person (`ld:Akteur`), zum betriebenen Ort (`ld:Ort`) sowie zu einigen beteiligten Personen (`foaf:Person`) zusammengefasst. Im Transformationsskript wird versucht, diese drei Bestandteile voneinander zu trennen, wobei die Personen-Referenzen nur bis zu `org:Membership` aufgelöst werden können, da Personen über einen Fremdschlüssel in eine Tabelle referenziert werden, die aus naheliegenden Gründen nicht mit exportiert wird[10]. Hier wäre es sinnvoll, diese Angaben zusätzlich als `foaf:Person` zu extrahieren und damit die exportierten RDF-Daten zu ergänzen.

Instanzen dieser Klasse enthalten die Textprädikate `name`, `email`, `telefon`, `url`, `ansprechpartner`, `funktion`, `bild`, `beschreibung`, `oeffnungszeiten` sowie die Verweise auf `adresse` und `ersteller`.

Event. Die Modellierung folgt der von `ld:Event`. In der neuen LE-Version sind für Events nur noch Start- und Endzeit gegeben, die komplexeren Möglichkeiten von regelmäßig stattfindenden Events wird aktuell – wie in `ld:Event` – nicht unterstützt.

Instanzen dieser Klasse enthalten die Textprädikate `name`, `kurzbeschreibung`, `bild`, `url`, die Verweise auf `ort` und `ersteller` sowie die `datetime`-Prädikate `start_ts` und `ende_ts`.

Sparten. Instanzen dieser Klasse spannen eine nicht konsolidierte Tagwolke auf, um die Events zu kategorisieren.

Unterschiede zu `ld:Event`.

- `ical:location` verweist nicht auf einen `ld:Ort`, sondern auf eine `le:Adresse`.
- `ical:creator` verweist auf eine Person in der nicht zugänglichen Personentabelle.
- Über `le:hatAkteur` ist einem Event teilweise ein Akteur zugeordnet.
- Über `le:zurSparte` sind einem Event Schlagworte zugeordnet.

[10] Im Gegensatz zu den anderen Tabellen, die eine Erweiterung des Drupal-Datenmodells darstellen, verweisen diese Referenzen auf die Drupal-Usertabelle.

4 Nachhaltiges Sachsen

4.1 Hintergrund

Nachhaltiges Sachen[11] ist ein sachsenweites Projekt, um Anbietern in den Bereichen Nachhaltigkeit und MINT eine gemeinsame Plattform zu bieten, über die standardisierte Informationen zu den *Anbietern* sowie deren *Aktivitäten* verbreitet werden. Das Projekt entstand aus dem Vorgängerprojekt *Nachhaltiges Leipzig* durch eine Werweiterung um Regionen. Das Vorgängerprojekt ist in diesem Projekt aufgegangen.

Die Anbieter nutzen eine webbasierte Erfassungsschnittstelle[12], um die entsprechenden Informationen bereitzustellen. Die Plattform stellt eine REST-Schnittstelle zur Verfügung, über welche Informationen strukturiert ausgelesen werden können.

4.2 Schnittstelle

Die Schnittstelle ist unter `https://daten.nachhaltiges-sachsen.de/api/doc` genauer beschrieben.

- `https://daten.nachhaltiges-sachsen.de/api/v1/activities.json`
  Returns an array of all activity objects. They are the core concept of this application.
- `https://daten.nachhaltiges-sachsen.de/api/v1/api/v1/categories.json`
  Returns an array of all category objects. Activities may belong to one or more categories and subcategories. Each activity can have one or more goals, think of them as „tags“. The count for each `goal_cloud` object shows how many activities from each category use each goal. Use this to build a tag cloud.
- `https://daten.nachhaltiges-sachsen.de/api/v1/api/v1/locations.json`
  Returns an array of all location objects. Locations can be referenced by users and activities.
- `https://daten.nachhaltiges-sachsen.de/api/v1/api/v1/regions.json`
  Returns an array of all region objects.
- `https://daten.nachhaltiges-sachsen.de/api/v1/api/v1/products.json`
  Returns an array of all product objects. They are specific to the activity type *Filiale*.
- `https://daten.nachhaltiges-sachsen.de/api/v1/api/v1/trade_categories.json`
  Returns an array of all `trade_category` objects. They are specific to the activity type *Filiale*.
- `https://daten.nachhaltiges-sachsen.de/api/v1/api/v1/trade_types.json`
  Returns an array of all `trade_type` objects. They are specific to the activity type *Filiale*.
- `https://daten.nachhaltiges-sachsen.de/api/v1/api/v1/users.json`
  Returns an array of all user objects.

4.3 Datenmodell

Im Modell werden die zwei Klassen *Akteure* (`users`) und *Aktivitäten* (`activities`) unterschieden, welche über die REST-API ausgelesen werden können. Im Modell sind weitere Da-

[11] `https://nachhaltiges-sachsen.de/`.
[12] `https://daten.nachhaltiges-sachsen.de/`.

tenstrukturen implizit als Konzepte vorhanden.

Das ist zum einen eine genauere Aufteilung eines *Akteurs* in

- die Beschreibung des Akteurs,
- die Adresse,
- der Ansprechpartner und
- die Accountdaten des Ansprechpartners,

die alle in einer gemeinsamen Datenbanktabelle *users* zusammengefasst sind.

Aktivitäten sind in verschiedene Aktivitätstypen unterteilt, welche durch den Wert eines speziellen Attributs `type` unterschieden werden. Neben generischen Attributen haben die einzelnen Aktivitätstypen weitere spezielle Attribute, was man in einer zweistufigen (erweiterbaren) Vererbungshierarchie oder in einem RDF-Modell bzw. als abstrakten Datentyp (im Sinne etwa von Java Interfaces) als Mengen von Signaturen beschreiben kann. Im Weiteren wird die letztere Darstellungsform verwendet.

Die folgenden Ausführungen nehmen außerdem Bezug auf die in den Leipzig Data Tools verfügbaren Transformationsskripte[13].

Die Klasse *Akteure* umfasst wie im LE-Projekt ein Sublimat aus Informationen zu `ld:Akteur`, zu dessen Adresse sowie zu Personen, die für den Akteur als Ansprechpartner tätig sind. Es lässt sich nicht unterscheiden, ob zum Beispiel eine Telefonnummer oder eine Adresse zum Vereinsbüro gehört oder zum Ansprechpartner. Mit dem Transformationsskript wird versucht, diese Informationen in die Teile `org:Organization` (Akteure), `foaf:Person` (Personen) und `org:Membership` (Rolle dieser Person beim Akteur) aufzuteilen. An anderer Stelle wird aber sehr wohl der Datentyp *Kontakt* („Kontakt zur Abholung“ einer Ressource, „Ansprechpartner“ eines Bildungsangebots) als Teilinterface mit [Name, Email-Adresse, Telefon] verwendet.

Die Klasse *Aktivitäten* zerfällt in die Unterklassen („Datensatztypen“ in der NL-Sprache) `Event`, `Action`, `Project`, `Service` und `Store`, die über das Feld `type` unterschieden werden. Im Zuge der Überarbeitung Ende 2018 wurden die „Klassen“ `Ressource`, `Bildungsangebot` und `Beratungsangebot` als Unterklassen von `Service` eingeführt, wobei hier die zusätzliche Unterscheidung über das Feld `service_type` ausgeführt wird.

In der aktuellen (Febr 2019) Version werden für einen „einfachen Akteur“ Erfassungsmasken für die Datensatztypen *Project*, *Event*, *Action* sowie die drei Servicetypen *Ressourcen*, *Bildungsangebot* und *Beratungsangebot* angeboten.

Zur Unterscheidung der Instanzen werden ID's als numerische Primärschlüssel der Datenbank verwendet. In den Transformationsskripten werden lokale URIs direkt aus diesen Primärschlüsseln erzeugt. Diese haben (bis auf Personen, deren Daten dem Namensschema von Leipzig Data folgen, die Informationen werden aus Datenschutzgründen nur intern verwendet) grundsätzlich die Struktur `<Präfix>/<Typ>.<Id>` mit `<Typ>`$\in$ {`Person`, `Akteur`, `Activity`}.

Adressen. In der Anforderungsbeschreibung für eine Erweiterung (Ende 2017) soll eine weitere Klasse *Orte* ergänzt werden, die eine akteursübergreifende Verwaltung von Adressdaten umsetzt. Eine *Adresse* ist dabei eine implizite Datenstruktur, die über die API als Menge von Attributen

[13] Siehe `https://github.com/LeipzigData/Tools/tree/master/nachhaltiges-leipzig/nl-rdf`.

- full_address – String
- district – String
- latlng – Array

ausgeliefert wird. `full_address` ist dabei bereits ein Aggregat, das aus den intern separierten Bestandteilen *Straße und Hausnummer*, *PLZ* und *Ort* kombiniert wurde.

Adressen[14] werden grundsätzlich als Datenaggregate verwaltet, nicht als Datenobjekte. Wenn zu einer Aktivität keine Adresse angegeben ist, wird die Adresse aus den Profildaten des Akteurs übernommen. Spätere Änderungen dieser Profildaten haben aber aktuell keine Auswirkung auf diesen Klon.

Es ist eine akteursübergreifende Datenbank zu Orten im Aufbau, aus der häufig verwendete Adressdaten übernommen werden können. Auch dies geschieht durch einfache Übernahme der entsprechenden Attributwerte. Geplant ist, Änderungen in dieser Datenbasis in die entsprechenden ADressfelder von Aktivitäten (und auch Akteuren?) zu propagieren, wozu ein System von Backlinks aufgebaut werden müsste. Unklar bleibt, wie mit zwischenzeitlichen Änderungen umgegangen wird, die vom Besitzer der Aktivität an den früher übernommenen Adressdaten vorgenommen worden sind.

Akteure. In der Collection `users` (Akteure) sind Informationen über Akteure zusammengefasst, wobei nicht zwischen den juristischen Personen und den für diese agierenden Personen unterschieden wird.

Prädikate in users.json:

- Akteur
  - id – String
  - name – String, Name der Organisation
  - organization_type – String, Art der Organisation (gewerbliches Unternehmen, gemeinnütziger Verein, Stiftung, Genossenschaft, Initiative, Freiberufler, Bildungseinrichtung, Sonstige Organisation)
  - organization_url – String, Homepage
  - organization_logo_url – String, Logo
- ?? – Checkbox, Handels- oder Gastronomieeinrichtung (nicht mit ausgeliefert)
- ?? – Checkbox, veröffentlicht (nicht mit ausgeliefert)
- ?? – Checkbox, aktiv (nicht mit ausgeliefert)
- Adresse (des Akteurs oder des Ansprechpartners?)
- ?? – Checkbox, Anschrift öffentlich sichtbar (nicht mit ausgeliefert)
- Ansprechpartner
  - first_name – String
  - last_name – String
  - organization_position – String
- Account- und Erreichbarkeitsdaten
  - email – String

[14] In der NL-Terminologie „Orte“, allerdings werden zu Orten in der aktuellen Konzeption keine Daten jenseits der Adressdaten und eines (verbindlichen) Namens gespeichert.

- phone_primary – String
- phone_secondary – String
- Passwort (nicht mit ausgeliefert)

Aktivitäten. `activities` ist ein Obertyp zu verschiedenen Arten von Aktivitäten (Aktionen, Events, Projekte, Services, Stores, ...), die mit dem Prädikat `nl:hasType` näher spezifiziert werden. In der Collection `activities` sind Informationen über die verschiedenen Typen von Aktivitäten zusammengefasst, wobei nicht alle Prädikate bei allen Untertypen verwendet werden. Leere Prädikate werden bei den RDF-Dumps nicht berücksichtigt.

Generische Prädikate:

- id – String
- type – String (Typ der Aktivität), Auswahl
- user_id – String (Id des beteiligten Akteurs), Auswahl
- name – (Titel, Name) String
- description – (Beschreibung) String
- Adresse (an der die Aktivität stattfindet) – Ortsauswahl
- is_fallback_address – String (Boolescher Wert, Bedeutung unklar)
- info_url – String
- video_url – String
- image_url – String
- categories – Array, weitere Kategorien (siehe Kategorienkonzept)
- first_root_category – String, Hauptkategorie

Weitere Prädikate für Actions:

- start_at – String, Zeitraum/Termine (als Freitextfeld)

Weitere Prädikate für Events:

- start_at – String (mit Auswahl Datum/Zeit hinterlegt)
- end_at – String (mit Auswahl Datum/Zeit hinterlegt)
- target_group – (Zielgruppe) String, Freitextfeld
- costs – (Kosten) String, Freitextfeld
- requirements – (Bedingungen) String, Freitext-Area
- speaker – (Referenten) String, Freitextfeld
- ?? – Checkboxen, kostenfrei, kinderfreundlich, barrierefrei (nicht mit ausgeliefert)
- goals – (Ziele) Array, Mehrfachauswahl, wird aktuell nicht mit ausgeliefert.

Weitere Prädikate für Projects:

- short_description – (Kurzbeschreibung) String, Freitext-Area
- goals – (Ziele) Array, Mehrfachauswahl
- property_list – Array, Liste mit speziellen Merkmalen, als Freitext-Area, die zeilenweise ausgelesen wird.

Weitere Prädikate für Services:

- target_group – String, Zielgruppenbeschreibung
- costs – String, Kosten
- requirements – String, Bedingungen
- short_description – String, Kurzbeschreibung
- goals – Array
- service_type – String, Angebotsart (Workshop, Exkursion, Vortrag, GTA, Unterrichtseinheit, Beratungsangebot)
- target_group_selection – String, Auswahl (Kindergarten, Grundschule, Sekundarstufe, Erwachsenenbildung)
- duration – String

Neu gibt es Beratungsangebote und Bidungsangebote, jeweils ohne Feld „Angebotsart“

Weitere Prädikate für Store:

- short_description – String
- property_list – Array
- products – Array
- trade_categories – Array
- trade_types – Array

Auf dieser Basis sind folgende Transformationen nach LDD möglich:

- `full_address` kann als `ld:proposedAddress` in eine syntaktisch korrekte URI einer `ld:LeipzigerAdresse` transformiert werden.
- `latlng` kann in eine `gsp:asWKT` Geo-Adresse transformiert werden.

Weitere Teile der Modellierung. Diese sind noch wenig ausgearbeitet und enthalten oft nur wenige Instanzen pro Klasse.

- `categories` repräsentiert eine baumartige Struktur verschiedener Tags, die einzelnen Aktivitäten zugewiesen sind. Diese Struktur wird zur Menüführung verwendet.
- `goals` repräsentiert eine geordnete Liste verschiedener Tags, die einzelnen Aktivitäten zugewiesen sind. Das ist als Tagwolke modelliert, kann aber – mit Blick auf die Datenqualität – nur von einem Administrator erweitert werden.
- `products` repräsentiert eine Liste verschiedener Produktkategorien, die einzelnen Stores zugewiesen sind.
- `trade_types` und `trade_categories` repräsentieren zwei geordnete Listen verschiedener Tags, die einzelnen Akteuren über Crossreferenz-Tabellen zugewiesen sind.

5 Gebäudenavigator

5.1 Hintergrund

Im Rahmen der Zusammenarbeit mit dem Behindertenverband Leipzig e.V. hat Konrad Abcht vom AKSW-Team einen Gebäudenavigator[15] entwickelt, in welchem die behindertengerechte Ausstattung von etwa 1800 Leipziger Orten hinterlegt ist.

Datenbasis ist eine vom BVL gepflegte Datenbank mit Informationen über 2600 öffentliche Gebäude und Plätze in Leipzig. Jedes Gebäude und jeder Platz wurde nach der Eignung für Menschen mit Mobilitätseinschränkungen klassifiziert. Es wurden jedoch nicht nur Angaben zur Mobilität gemacht, sondern auch Informationen über viele andere Zugangshilfen hinterlegt. Für Menschen mit Sinnesbeeinträchtigungen (blind, gehörlos) gibt es, falls vorhanden, genaue Beschreibungen der Angebote vor Ort.

Die Datenbasis (Instanzdaten) ist im Open Data Portal der Stadt Leipzig im RDF Graphen `https://opendata.leipzig.de/bvlplaces/` zugänglich. Es wird der Namensraum `https://places.behindertenverband-leipzig.de/` verwendet und die Instanzdaten unter der jeweiligen Adresse RDF-konform menschenlesbar dargestellt.

5.2 Schnittstelle

Die Daten können über den SPARQL Endpunkt `https://opendata.leipzig.de/virt-sparql` des Open Data Portals der Stadt Leipzig ausgelesen werden.

Die folgende Anfrage (der Prefix ist hier aus drucktechnischen Gründen eingekürzt) etwa listet alle 814 Orte auf

```
PREFIX bvlo: <https://github.com/AKSW/.../ontologies/place/ontology.ttl#>
SELECT *
FROM <https://opendata.leipzig.de/bvlplaces/>
WHERE {
  ?s a bvlo:Place; dc:title ?t .
}
```

5.3 Datenmodell

Alle Instanzen gehören zur Klasse `bvlo:Place`. Neben den bekannten Ontologien (rdf, dc, dcterm, schema, skos, dbpedia-owl, geo) sowie der weniger bekannten „Wheelchair Accessibility“ Ontologie

```
swa: <http://semweb.mmlab.be/ns/wa#>
```

werden zwei eigene Ontologien `bvlo` und `bvla` entwickelt, um die Zugänglichkeit der Places genauer zu beschreiben.

Aus den drei Prädikaten `schema:addressLocality` (Ort), `schema:postalCode` (PLZ) und `schema:streetAddress` (Straße und Hausnummer) lässt sich für viele Instanzen ein Mapping auf `ld:LeipzigerAdresse` konstruieren.

[15] `http://leipzig-data.de/anwendungen/gebaudenavigator/`

Die Prädikate sind oft selbsterklärend, da „sprechende Namen“ verwendet werden:

- rdf:type, dc:title (rdfs:label), dcterms:identifier (ID des Datensatzes)
- dbpedia-owl:category – Zuordnung zu einer Dbpedia Kategorie, hier als Literal
- skos:note – genauere Aufzeichnungen zur Historie des Datensatzes
- Adressinformationen: schema:addressLocality, schema:postalCode, schema:streetAddress, geo:lat, geo:long
- Weitere Informationen zum Betreiber: schema:email, schema:WebSite, schema:faxNumber, schema:telephone

Jeder solche `bvlo:Place` wird dann umfassend bzgl. der Zugangsparameter auf der Basis der drei Ontologien swa, bvla und bvlo beschrieben.

6 MINT-Orte

6.1 Hintergrund

Während die Grundausrichtung des Leipzig Data Projekts auf den Aufbau von White Pages bzw. Yellow Pages zu Akteuren, Orten und Adressen orientiert ist, wurde zu verschiedenen Zeiten auch mit Möglichkeiten einer genaueren Darstellung von Akteuren und deren Aktivitäten experimentiert.

Im Zuge eines dieser Experimente wurde der 2014 in Printform von der Stadt Leipzig herausgegebene „Katalog der MINT- und Umweltangebote“ in eine digitale Form übertragen. In diesem Katalog sind entsprechende Akteure und außerschulische Lernorte zusammen mit einer Auswahl von Angeboten gelistet.

Die Arbeiten wurden im Rahmen eines studentischen Projekts ausgeführt[16], in welchem die Daten nicht nur als RDF digitalisiert, sondern diese auch noch mit weiteren Informationen aus eigener Recherche (Fotografien der einzelnen Objekte) angereichert wurden. Die Daten sind in der Turtle-Datei `http://leipzig-data.de/RDFData/MINTBroschuere2014.ttl` zusammengefasst.

6.2 Schnittstelle

SPARQL Endpunkt `http://leipzig-data.de:8890/sparql` des LD-Projekts.

Beispielanfrage:

```
PREFIX mint2014: <http://leipzig-data.de/Data/MINT-2014/>
PREFIX ld: <http://leipzig-data.de/Data/Model/>
SELECT distinct ?a ?l ?k
from <http://leipzig-data.de/Data/MINTBroschuere2014/>
WHERE {
  ?a a mint2014:Ortsbeschreibung; rdfs:label ?l .
  optional { ?a mint2014:Kurzinformation ?k . }
} order by ?a
```

6.3 Datenmodell

Das Datenmodell umfasst drei Klassen `mint2014:Ortsbeschreibung`, `mint2014:Angebot` und `mint2014:Schwerpunkt`, mit denen eine genauere Beschreibung der einzelnen Orte, der dort vorgehaltenen Angebote sowie die Zuordnung dieser zu Schwerpunkten entsprechend der Methodik der vorliegenden städtischen Broschüre umgesetzt wird. Die Zuordnung zu Instanzen vom Typ `ld:Ort` erfolgt durch ein Prädikat `mint2014:describes`

mint2014:Ortsbeschreibung – Beschreibung eines MINT-Orts

- rdfs:label Literal – Bezeichnung
- mint2014:Kurzinformation Literal – Kurzinformation

[16] `http://leipzig-data.de/ontology/mintbroschuere/`

- mint2014:Leistungsangebot Literal – Leistungsangebot
- mint2014:Oeffnungszeiten Literal – Öffnungszeiten
- mint2014:OePNV-Anbindung Literal – ÖPNV-Anbindung
- mint2014:Telefon Literal – Telefon
- mint2014:Fax Literal – Fay
- mint2014:Mail Literal – Email
- mint2014:Internet Literal Webseite
- mint2014:hasTag ld:Tag – verschiedene Tags
- mint2014:hasImage Literal – Dateiname eines Fotos
- mint2014:hasLogo Literal – Dateiname des Logos
- mint2014:describes ld:Ort – MINT-Ort als Leipzig Data Ort

mint2014:Angebot – MINT-Angebot eines MINT-Orts

- ld:Lernziele Literal – Lernziele des Angebots
- ld:Zielgruppen Literal – Zielgruppen
- ld:Kosten Literal – Kosten
- ld:Veranstaltungsort Literal – Adresse – in LD.Adresse zu verwandeln
- ld:Hinweise Literal – Hinweise zum Angebot
- ld:Laufzeit Literal – Laufzeit des Angebots
- ld:relatedBundle mint2014:Ortsbeschreibung – MINT-Ort, welcher das Angebot verantwortet
- mint2014:hasTag ld:Tag – verschiedene Tags
- rdfs:label Literal – Bezeichnung des Angebots

mint2014:Schwerpunkt – MINT-Schwerpunkt

- ld:Zielgruppen Literal – Zielgruppe des Schwerpunkts
- ld:Kosten Literal – Kosten
- ld:GTA Literal – als Ganztagsangebot nutzbar?
- ld:relatedBundle mint2014:Ortsbeschreibung – Schwerpunkt welches MINT-Orts
- rdfs:label Literal – Bezeichnung des Schwerpunkts

7 Jugendstadtplan

7.1 Hintergrund

Mit ähnlicher Motivation wie im Teilprojekt „MINT-Orte“ wurde bereits 2013 im Vorfeld der in Leipzig stattfindenden „World Skills“ an einem Jugendtsdtplan gearbeitet.

Die Arbeiten wurden im Rahmen eines studentischen Projekts ausgeführt[17], in welchem eine entsprechende Datenbasis aufgebaut wurde. Die Daten wurden 2013 von einer studentischen Projektgruppe gesammelt. Die Informationen sind in vielen Fällen in Englisch und Deutsch vorhanden. Die Daten sind in der Turtle-Datei `http://leipzig-data.de/RDFData/Jugendstadtplan.ttl` zusammengefasst.

7.2 Schnittstelle

SPARQL Endpunkt `http://leipzig-data.de:8890/sparql` des LD-Projekts.

Beispielanfrage:

```
PREFIX jsp: <http://leipzig-data.de/Data/Jugendstadtplan/>
PREFIX ld: <http://leipzig-data.de/Data/Model/>
SELECT distinct ?a ?l ?amb
from <http://leipzig-data.de/Data/Jugendstadtplan/>
WHERE {
  ?a a jsp:Ort; rdfs:label ?l .
  optional { ?a jsp:Ambiente ?amb . }
} order by ?a
```

7.3 Datenmodell

In diesem Projekt wurde eine größere Hierarchie von Klassen aufgebaut, um so Zusammenhänge zu modellieren, die in späteren Ansätzen als Tags oder Eigenschaften modelliert wurden.

Zentrale Klasse ist `jsp:Ort`, die 142 Instanzen von Einrichtungen in Leipzig beschreibt, die für Jugendliche interessant sind. Dies entspricht einer Ortsbeschreibung, allerdings mit einer eigenen Ontologie. Über das Prädikat `jsp:describes` sind die Orte zu Instanzen von Typ `ld:Ort` relatiert und damit in die restliche Welt des Leipzig Data Projekt eingebunden. Als Namenspräfix wird (bis auf aktuell zwei Ausnahmen – das ist zu fixen)

`http://leipzig-data.de/Data/Jugendstadtplan/Ort/`

verwendet.

Prädikate von jsp:Ort

- rdfs:label, foaf:homepage, jsp:describes

[17] `http://leipzig-data.de/jugendstadtplan/`

- jsp:AccessibilityComment
- jsp:Altersbeschraenkung
- jsp:Ambiente
- jsp:Angebot
- jsp:Anmeldung
- jsp:Kostentyp
- jsp:Musikrichtung
- jsp:Rauchmoeglichkeit
- jsp:Religioeser_Ort
- jsp:Type
- jsp:Unterpunkt
- jsp:hasAccessibility
- jsp:hasArt
- jsp:hasCost
- jsp:hasEquipment
- jsp:hasFreeTag
- jsp:hasInstitutionType
- jsp:hasIrregularOpeningHours
- jsp:hasOpeningHours
- jsp:hasUnterpunkt
- jsp:hascategory
- jsp:hasdescription
- jsp:isType
- jsp:suppliedBy
- jsp:Untergruppe

Die weitere Struktur ist noch genauer zu beschreiben.

8 Afeefa

8.1 Hintergrund

2017 startete die Dresdner Afeefa-Gruppe[18] Aktivitäten, um ein ähnliches Projekt in verschiedenen anderen Städten, so auch in Leipzig[19], auszurollen. Das Leipziger Projekt wird – nach Aussage auf der Webseite – von „Interaction Leipzig e.V“ betrieben, hat sich also organisatorisch von der Dresdner „Mutter“ abgespalten. Der Verein hat sich besonders im Bereich der Flüchtlingsarbeit positioniert und profiliert.

8.2 Schnittstelle

Wo ist die zu finden?

8.3 Datenmodell

Ein erster Pitch einer Datentransformation, die über eine uns kommunizierte REST-API ausgeführt wurde, ist unter `Transform/Data/Afeefa-Leipzig.ttl` im LD „Tools“ Repo zu finden.

Das Ganze muss noch weiter analysiert werden.

9 bne-sachsen.de

9.1 Hintergrund

Zu ergänzen.

9.2 Schnittstelle

Anfang Juni 2020 wurde die folgende API veröffentlicht[20], über die einzene Teile der Datenbasis ausgelesen werden können. Die Daten werden als JSON-Array zurückgegeben. Die folgenden Informationen sind aus der Datei übernommen, die uns am 4.6.2020 geschickt wurde.

- Angebote: `https://bne-sachsen.de/wp-json/content/offers`
- Veranstaltungen: `https://bne-sachsen.de/wp-json/content/events`
- Materialien: `https://bne-sachsen.de/wp-json/content/materials`
- News: `https://bne-sachsen.de/wp-json/content/posts`

Mit Hilfe von Parametern können die zurückgegebenen Beiträge eingegrenzt werden. Parameter werden immer mit der Syntax `?{PARAMETER_NAME}={WERT}` übergeben und können auf die übliche Weise kombiniert werden.

[18] `https://afeefa.de/`

[19] `https://leipzig.afeefa.de/`

[20] Siehe dazu die Mail von Mara Kayser an Matthias Schirmer vom 4.6.2020.

Bsp.: `https://bne-sachsen.de/wp-json/content/offers?p=8234`

- `p` – einzelnen Eintrag ausgeben. ID des Beitrags muss übergeben werden.
  Beispiel: `https://bne-sachsen.de/wp-json/content/offers?p=8234`
- `limit` – Maximum der ausgegebenen Beiträgen angeben.
  Beispiel: `https://bne-sachsen.de/wp-json/content/offers?limit=20`
- `author` – Gibt alle Beiträge eines Autors zurück.
  Beispiel: `https://bne-sachsen.de/wp-json/content/offers?author=115`
- `author_name` – Gibt alle Beiträge eines Autors zurück. Der „Slug“ des Autors muss übergeben werden. Der Slug kann aus dem Link zum Autorenprofil übernommen werden.
- `district` - Gibt alle Beiträge aus dem Landkreis zurück. Nur für Veranstaltungen und Angebote verfügbar.
  Beispiel: `https://bne-sachsen.de/wp-json/content/offers?district=dresden`

9.3 Analyse HGG vom 7. August 2021

Die API von `bne-sachsen.de` wurde inzwischen komplett geändert. Im Gegensatz zu Juni 2020 gibt es keine districts mehr und wenn ich nach events frage, bekomme ich nur posts geliefert. Die vier Aufrufe

- `https://bne-sachsen.de/wp-json/content/posts?p=9025`
- `https://bne-sachsen.de/wp-json/content/events?p=9025`
- `https://bne-sachsen.de/wp-json/content/materials?p=9025`
- `https://bne-sachsen.de/wp-json/content/offers?p=9025`

liefern alle denselben post-Eintrag.