Informationspraxis Bd. 4, Nr. 1 (2018)

DOI: https://doi.org/10.11588/ip.2018.1.52445

Adrian POHL, Fabian STEEG & Pascal CHRISTOPH

lobid – Dateninfrastruktur für Bibliotheken

Zusammenfassung
lobid ist der zentrale Anlaufpunkt für die Linked-Open-Data-Dienste des Hochschulbibliothekszentrums des Landes Nordrhein-Westfalen (hbz). Das Akronym „lobid" steht für „Linking Open Bibliographic Data". lobid umfasst Rechercheoberflächen und Web-APIs.

Die lobid-Dienste bieten Zugriff auf die Titeldaten des hbz-Verbundkatalogs, Beschreibungen von bibliothekarischen Organisationen und anderen Gedächtnisinstitutionen aus dem Sigelverzeichnis und der Deutschen Bibliotheksstatistik (DBS) sowie auf die Gemeinsame Normdatei (GND). Die Datensets können in verschiedenen Kontexten einheitlich (JSON-LD über HTTP) genutzt und eingebunden werden. Vielfältige Möglichkeiten der Datenabfrage werden unterstützt.

Der Artikel beschreibt zunächst die technischen Hintergründe von lobid und die Erfahrungen, die bei der Transformation verschiedener Datensets nach JSON-LD gemacht wurden. Vorgestellt wird auch der Entwicklungsprozess und die Art und Weise der Dokumentation der Dienste.

Schlüsselwörter
Linked Open Data, JSON-LD, Web APIs, GND, Verbundkatalog, ISIL-Verzeichnis, lobid

lobid – data infrastructure for libraries

Abstract
lobid is the central hub for the linked open data services provided by the North Rhine-Westphalian Library Service Centre (hbz). lobid stands for "linking open bibliographic data". It offers search interfaces for users and web APIs for developers.

The lobid services provide access to the title data of the hbz union catalog, to descriptions of library organisations and other memory institutions from the German ISIL registry and the German Library Statistics (DBS), and to the Integrated Authority File (GND). Through the lobid services, these datasets are offered in a consistent form (JSON-LD over HTTP), ready to be used and integrated in various contexts. We provide versatile mechanisms to query the data.

This article outlines the technical background for implementing lobid and our experiences in transforming various data sets to JSON-LD. We also describe our development process and our approach to documenting the services.

Keywords
Linked Open Data, JSON-LD, Web APIs, GND, union catalogue, ISIL registry, lobid

Dieser Text steht unter der CC BY 4.0-Lizenz

Inhaltsverzeichnis

lobid: Schnittstellen für Recherche und Entwicklung

Lobid stellt offene Programmierschnittstellen (APIs) und Rechercheoberflächen zur Verfügung, die auf Linked Open Data (LOD) basieren. lobid wird vom Hochschulbibliothekszentrum des Landes NRW betrieben und umfasst derzeit drei Dienste: lobid-resources bietet Zugriff auf den hbz-Verbundkatalog, lobid-organisations bietet Informationen zu Gedächtnisinstitutionen im deutschsprachigen Raum, lobid-gnd bietet Zugriff auf die Gemeinsame Normdatei (GND).

lobid richtet sich primär an Mitarbeiter*innen in bibliothekarischen und wissenschaftlichen Einrichtungen im gesamten deutschsprachigen Raum. Zum einen sind dies Bibliothekar*innen und Wissenschaftler*innen, die etwa eine Recherche im hbz-Verbundkatalog, der GND oder dem Sigelverzeichnis vornehmen wollen. Zum anderen richtet sich lobid mit der Bereitstellung zuverlässiger und leicht nutzbarer Web-APIs an Entwickler*innen, die am Aufbau oder der Verbesserung von Anwendungen in ihren Einrichtungen arbeiten.

Ursprung

lobid wurde von Anfang an um die bereitzustellenden *Daten* herum konzipiert, so lautet die Auflösung des Akronyms "linking open bibliographic data". Zunächst war es über mehrere Jahre hinweg notwendig, die offene Lizenzierung der Quelldaten zu propagieren (vgl. Pohl 2010). Daneben war deren Transformation und die Modellierung der Zieldaten sowie die Auswahl der RDF-Properties und -Klassen über lange Zeit Kern der Arbeit. 2010 begann lobid mit der Bereitstellung der transformierten Daten über einen Triple Store, also einer Graphdatenbank. Der Triple Store war aber für performance-kritische Anwendungsfälle (wie einen Entity-Lookup via Textstring) nicht optimiert. Zudem gab es hohe Einstiegshürden bei der Nutzung der Daten. Um die Performanz zu optimieren und die Nutzbarkeit zu erleichtern, wurde das lobid-Backend 2013 auf Elasticsearch mit JSON-Daten, also einen Dokumentenspeicher, umgestellt. Auf der Basis unserer Erfahrungen mit dieser Version der lobid-API veröffentlichten wir 2017 (lobid-resources und lobid-organisations) bzw. 2018 (lobid-gnd) die aktuellen Versionen der Dienste, die wir im Folgenden beschreiben.[1]

1 Dieser Text basiert auf Blogbeiträgen vom lobid-Blog unter http://blog.lobid.org. Die Blogbeiträge wurden für diesen Beitrag erweitert und verbessert sowie teilweise aus dem Englischen übersetzt.

Technik

Warum APIs?

Die lobid-API bietet einheitlichen Zugriff auf bibliothekarische Daten über eine webbasierte Programmierschnittstelle ("application programming interface", API). Sie liefert JSON für Linked Data (JSON-LD):

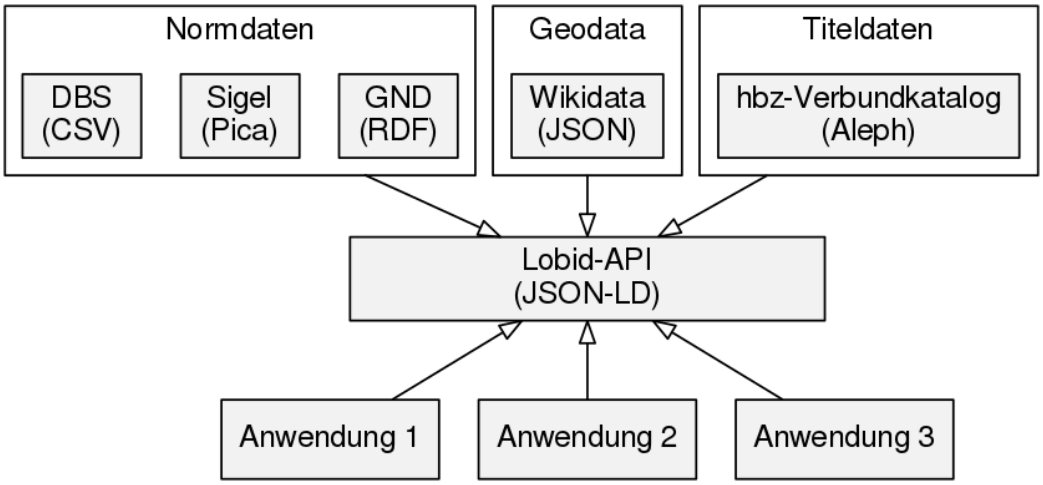

Abbildung 1: lobid-API: Datenquellen und Anwendungen

Die Grundidee ist dabei eine Entkopplung der Anwendungen von spezifischen Datenquellen, Formaten und Systemen. So können sich die Datenquellen und Systeme ändern, ohne dass Änderungen in den Anwendungen nötig werden, die auf die Daten über die API zugreifen. Dies ermöglicht die Entwicklung von system- und anbieterunabhängigen, nachhaltigen Anwendungen auf Basis bibliothekarischer Daten (siehe auch Steeg 2015a).

Die stabile Bereitstellung und Weiterentwicklung einer API bringt Herausforderungen mit sich. Ziel muss dabei grundsätzlich die API-Stabilität sein. Wenn Anwendungen ständig wegen API-Änderungen angepasst werden müssen, verliert die API ihren oben beschriebenen Sinn. Zugleich muss die API aber einen Mehrwert bieten und tatsächlich einheitlich und einfach nutzbar sein. Bei grundsätzlichem Verbesserungsbedarf muss sich eine API daher auch grundsätzlich weiterentwickeln können.

Architektur: Von horizontalen Schichten zu vertikalen Schnitten

Das lobid 1.x-System basierte auf einer klassischen monolithischen Schichtenarchitektur: Wir hatten ein Git-Repository, das die Implementierung für das Backend enthielt, mit der Logik aller Datentransformationen und der Indexschicht für alle Daten. Ein weiteres Git-Repository implementierte die API und ein gemeinsames Frontend für sämtliche Datensets, die so alle innerhalb eines Prozesses ausgeliefert wurden.

Dies führte insgesamt zu einer Verquickung der verschiedenen Datensets: Um etwa auf eine neuere Version unserer Suchmaschine (Elasticsearch) umzustellen, die wir für eines der Datensets benötigten, mussten alle Datensets umgestellt werden. Gleichzeitig gab es inhaltlich eigentlich unnötige Abhängigkeitskonflikte zwischen Software-Bibliotheken, die jeweils nur von den APIs einzelner Datensets benötigt wurden.

Daher spalteten wir lobid für die 2.0-Version in vertikale, in sich abgeschlossene Systeme für jedes Datenset (resources, organisations, gnd) auf (Steeg 2015b):

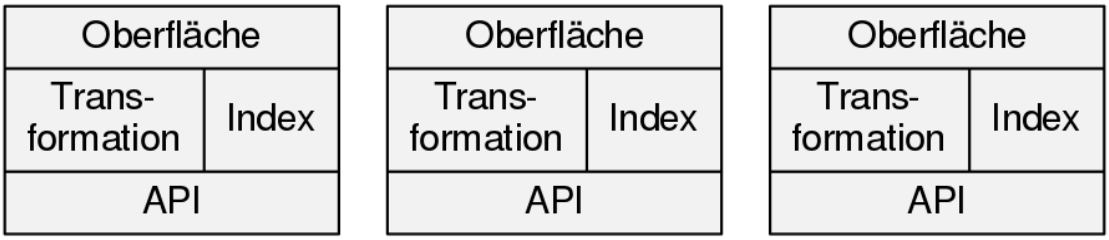

Abbildung 2: Architektur abgeschlossener Softwaresysteme

Durch die horizontale Kombination dieser Module haben wir nach wie vor APIs und Oberflächen für alle Dienste, doch Teile dieser APIs und Oberflächen sind in Module gekapselt, die je ein Datenset behandeln. Diese Module enthalten den für das jeweilige Datenset spezifischen Code und die spezifischen Abhängigkeiten und können unabhängig analysiert, verändert und installiert werden.

Linked Open Usable Data (LOUD) mittels JSON-LD

lobid hat nicht nur den Anspruch, leicht verwendbare und nützliche APIs für Entwickler*innen anzubieten, sondern war immer auch ein Linked-Data-Dienst. Auf dieser Basis wurde lobid ein Beispiel für die Bereitstellung von *Linked Open Usable Data (LOUD)*.

Robert Sanderson prägte diesen Begriff, um eine Form der Linked-Open-Data-Publikation voranzutreiben, die den Aspekt der Nutzung durch Software-Entwickler*innen und deren Konventionen und Bedürfnisse in den Vordergrund stellt (vgl. Sanderson 2016 sowie Sanderson 2018). Die Anforderungen an LOUD fasst Sanderson (2018), Folie 22 wie folgt zusammen (Übersetzung/Paraphrase von uns):

- der Zielgruppe angemessene Abstraktion
- niedrige Einstiegshürden
- unmittelbar verständliche Daten
- Dokumentation mit funktionierenden Beispielen
- wenig Ausnahmen, möglichst einheitliche Struktur

In dieser – zugegebenermaßen eher ungenauen – Begriffsbestimmung findet sich das lobid-Team wieder: Die LOUD-Prinzipien weisen eine große Überschneidung mit Konzepten der lobid-Datenpublikation auf. Beispielsweise setzt lobid seit 2013 auf JSON-LD, das bei der Erfüllung der LOUD-Anforderungen eine zentrale Rolle spielt, sollen doch alle Daten konsistent mit Blick auf JSON-LD modelliert werden (Sanderson 2018, Folien 32 und 37ff).

JSON-LD ist eine W3C-Empfehlung für eine JSON-basierte Linked-Data-Serialisierung. Man kann JSON-LD aus zwei Perspektiven betrachten: einerseits als RDF-Serialisierung (wie N-Triples, Turtle oder RDF/XML), andererseits als eine Möglichkeit, JSON zum Verlinken von Daten zu verwenden. Diese doppelte Perspektive spiegelt sich auch in der JSON-LD-Spezifikation wider, die beschreibt dass JSON-LD "als RDF verwendet werden kann", aber auch "direkt als JSON, ohne Kenntnis von RDF" (Sporny 2014, Übersetzung von uns). Reguläres JSON wird durch das Beifügen eines JSON-LD-Kontexts zu JSON-LD und damit als RDF serialisierbar.

Im Folgenden wird dargestellt, wie die aktuellen lobid-Daten gegenüber dem 1.x-System verbessert wurden und damit die Anforderungen an Linked Open Usable Data umfassender erfüllt werden.

Generisches JSON-LD im lobid 1.x-System

Da lobid von 2010 bis 2013 die Daten in einer Graphdatenbank speicherte, erzeugten die vorhandenen Datentransformationsprogramme N-Triples. In der ersten Version der lobid-APIs verwendeten wir diese Datentransformationsprogramme wieder und konvertierten die N-Triples automatisch mit einem JSON-LD-Prozessor (Abbildung X). Hier betrachteten wir JSON-LD vollständig als RDF-Serialisierung.

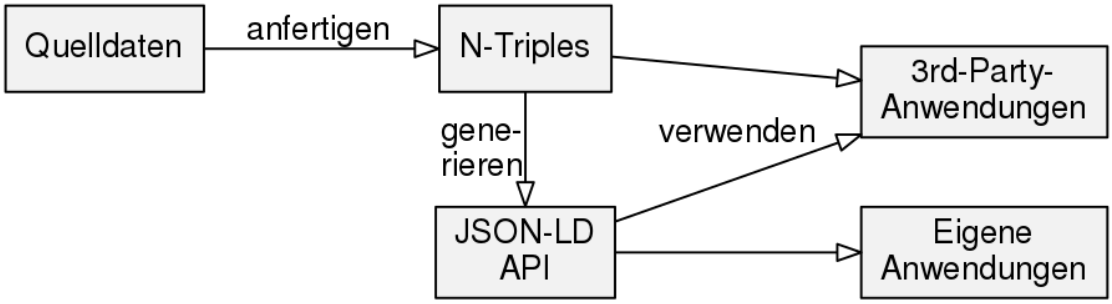

Abbildung 3: JSON-LD-Generierung in lobid, Version 1.x

Die Property-URIs der Triple wurden im JSON-LD zu JSON-Schlüsselwörtern. Diese Daten indexierten wir als expandiertes JSON-LD in Elasticsearch (Beispiel gekürzt):

```
{
 "@graph" : [{
  "@id" : "http://d-nb.info/gnd/11850553X",
  "http://d-nb.info/standards/elementset/gnd#preferredNameForThePerson" : [{
   "@value" : "Bach, Johann Sebastian"
  }],
  "http://d-nb.info/standards/elementset/gnd#biographicalOrHistoricalInformation" : [{
   "@value" : "Sohn: Bach, Friedemann",
   "@language" : "de"
  },{
   "@language" : "de",
   "@value" : "Dt. Komponist u. Musiker"
  }],
  "http://d-nb.info/standards/elementset/gnd#placeOfBirth" : [{
   "@id" : "http://d-nb.info/gnd/4014013-1"
  }]
 }]
}
```

Elasticsearch erfordert konsistente Daten für ein gegebenes Feld, z. B. muss etwa der Wert des Feldes `alternateName` entweder immer ein String oder immer ein Array sein. Wenn die Werte mal ein String, mal ein Array sind, führt dies bei der Indexierung in Elasticsearch zu einem Fehler. In der kompakten JSON-LD-Serialisierung werden einzelne Werte direkt serialisiert (z. B. als String). Wenn jedoch in einem anderen Dokument für das gleiche Feld mehrere Werte angegeben sind, wird ein Array verwendet. Expandiertes JSON-LD verwendet hingegen immer Arrays. Eine JSON-LD-Form, bei der kompakte Keys (Schlüsselwörter) mit expandierten Werten kombiniert sind gibt es derzeit nicht (siehe https://github.com/json-ld/json-ld.org/issues/338).

Beim Ausliefern der Daten über die API konvertierten wir die Daten dann in kompaktes JSON-LD, um anstelle der URIs kurze, benutzerfreundliche JSON-Keys zu erhalten (Beispiel gekürzt):

```
{
  "@graph" : [ {
    "@id" : "http://d-nb.info/gnd/11850553X",
    "preferredNameForThePerson" : "Bach, Johann Sebastian",
    "biographicalOrHistoricalInformation" : [ {
      "@language" : "de",
      "@value" : "Sohn: Bach, Friedemann"
    }, {
      "@language" : "de",
      "@value" : "Dt. Komponist u. Musiker"
    } ],
    "placeOfBirth" : "http://d-nb.info/gnd/4014013-1"
  }]
}
```

Damit erzeugten wir im Grunde zwei verschiedene Formate: das interne Indexformat und das extern sichtbare API-Format.

Maßgeschneidertes JSON-LD in den neuen Systemen

Maßgeschneidertes JSON mit Kontext für JSON-LD: lobid-organisations

Bei lobid-organisations (Pohl et al. 2018), dem ersten Datenset, das wir auf den neuen Ansatz umgezogen haben, haben wir das Vorgehen umgedreht. Statt manuell N-Triples anzufertigen und diese automatisch in JSON-LD zu konvertieren, erzeugen wir das JSON mit genau der Struktur, die wir benötigen. Auf dieser Grundlage generieren wir dann RDF-Serialisierungen wie N-Triples:

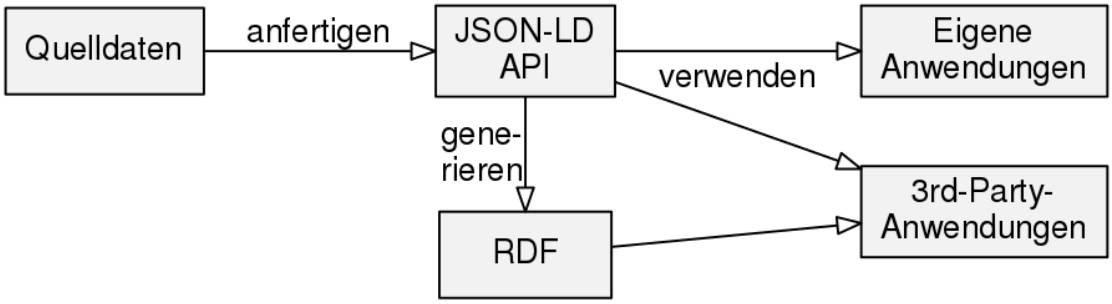

Abbildung 4: JSON-LD-Generierung in lobid, Version 2

Der zentrale Vorteil dieses Ansatzes ist, dass der konkrete Anwendungsfall ins Zentrum rückt: Wir bauen unsere API so, wie sie für unsere Anwendungen Sinn macht, anstatt zuerst eine Abstraktion zu erzeugen, aus der wir dann konkrete Darstellungen für die Anwendungen generieren.

Im Vergleich zum Ansatz im ersten lobid-System befinden wir uns hier am anderen Ende des Spektrums der Perspektiven auf JSON-LD, wie wir es oben beschrieben haben: Wir behandeln JSON-LD als JSON, ohne dass bei der Produktion der Daten oder bei ihrer Verwendung Kenntnisse von RDF erforderlich sind.

Maßgeschneidertes JSON-LD nach RDF-Serialisierung: lobid-resources

Für die neue Version von lobid-resources (Christoph et al. 2018) wählten wir einen Mittelweg. Wir entschlossen uns, auf die bestehende Transformation der Katalogdaten in N-Triples aufzubauen. Wir verwenden Code, der von Jan Schnasse im Etikett-Projekt (Schnasse & Christoph 2018) entwickelt wurde, um maßgeschneidertes JSON-LD aus den N-Triples zu erzeugen. Wie in lobid-organisations (und im Gegensatz zur ersten Version von lobid-resources), ist das maßgeschneiderte JSON-LD zugleich das Index- wie auch das von der API gelieferte Format.

Maßgeschneiderte RDF-Serialisierung für JSON-LD: lobid-gnd

Auch in lobid-gnd (Steeg et al., 2018a) erzeugen wir für Index und API dasselbe Format. In diesem Fall liegen unsere Ausgangsdaten bereits als RDF vor, und werden mithilfe eines detaillierten JSON-LD-Kontexts und JSON-LD-Framing, sowie einigen Anpassungen am RDF-Modell, in das gewünschte Format umgewandelt. Details dazu finden sich in Steeg et al. (2018b).

Vorteile des maßgeschneiderten JSON-LD

Alle drei Ansätze erzeugen also maßgeschneidertes JSON-LD, sei es auf spezifische Weise aus RDF generiertes oder manuell erzeugtes JSON. Dieses maßgeschneiderte JSON-LD hat mehrere Vorteile.

Was du siehst, ist, was du abfragen kannst

Ein zentraler Aspekt der neuen Systeme ist, dass wir nun das gleiche Format liefern, das auch im Index gespeichert ist. Dies ermöglicht beliebige Abfragen der Daten über generische Mechanismen, ohne dass wir spezifische Anfragen implementieren müssen. Beim Betrachten eines bestimmten Datensatzes, z. B. http://lobid.org/organisations/DE-605.json, sehen wir folgendes:


```
"classification" : {
  "id" : "http://purl.org/lobid/libtype#n96",
  "type" : "Concept",
  "label" : {
    "de" : "Verbundsystem/ -kataloge",
    "en" : "Union Catalogue"
  }
}
```


Auf Basis der Daten, die wir hier sehen, können wir ein beliebiges Feld nehmen, z. B. `classification.label.en` (die Punkte bilden die Schachtelung der Felder ab) und eine Abfrage wie die folgende formulieren: `http://lobid.org/organisations/search?q=classification.label.en:Union`. Im alten System, bei dem im Index expandiertes JSON-LD gespeichert war, die API aber kompaktes JSON-LD lieferte (s. Beispiele oben), brauchten wir spezifische Parameter, um Feldsuchen, etwa für Titel, Autoren oder Schlagwörter, in praxistauglicher Art umzusetzen (ohne URIs als Feldnamen, die wiederum komplexes Maskieren von Sonderzeichen erfordern), z. B.: `http://lobid.org/resource?name=Ehrenfeld`. Diese können nun stattdessen über einen generischen `q`-Parameter und die tatsächlichen Feldnamen aus den Daten formuliert werden: `http://lobid.org/resources/search?q=title:Ehrenfeld`. Auf diese Weise haben wir eine Suchsyntax, die sich an den Feldnamen orientiert, vermeiden zugleich eine Beschränkung auf die von uns antizipierten Arten von Abfragen und öffnen so die kompletten Daten für den API-Zugriff.

Hierarchisch strukturierte Daten

Das generierte JSON-LD des alten Systems war eine flache Struktur mit JSON-Objekten in einem Array unter dem `@graph`-Schlüsselwort, z. B. in `http://lobid.org/organisation?id=DE-605&format=full`:

```
"@graph": [
  {
    "@id": "http://purl.org/lobid/fundertype#n02",
    "prefLabel": [{
          "@language": "de",
          "@value": "Land"
       },{
          "@language": "en",
          "@value": "Federal State"
    }]
  },{
    "@id": "http://purl.org/lobid/stocksize#n11",
    "prefLabel": [{
          "@language": "en",
          "@value": "Institution without a collection"
       },{
          "@language": "de",
          "@value": "Einrichtung ohne Bestand"
    }]
  }
]
```

Diese Struktur war nicht sehr praktisch und entsprach nicht dem pragmatischen Geist von JSON-LD (vgl. Steeg 2014). Wenn man beispielsweise automatisch die englische Bezeichnung des Unterhaltsträgers einer Einrichtung verwenden will, muss man hier über alle `@graph`-Objekte iterieren und jeweils prüfen, ob die `@id` die Unterhaltsträger-ID ist, dann über alle `prefLabel`-Objekte iterieren und jenes mit dem passenden `@language`-Feld suchen, das dann als `@value` den gesuchten Wert enthält.

In den neuen Systemen bieten wir die Daten in einem geschachtelten, JSON-typischen Format an:

```
"fundertype": {
  "id": "http://purl.org/lobid/fundertype#n02",
  "type": "Concept",
  "label": {
    "de": "Land",
    "en": "Federal State"
  }
},
"collects": {
  "type": "Collection",
  "extent": {
    "id": "http://purl.org/lobid/stocksize#n11",
    "type": "Concept",
    "label": {
      "de": "Einrichtung ohne Bestand",
      "en": "Institution without holdings"
```

```
    }
  }
}
```

Dies ermöglicht einen einfacheren, direkten Zugriff auf die Daten. Das gesuchte Datum aus dem obigen Beispiel ist hier statt über mehrere Schleifen und Zeichenkettenvergleiche per Direktzugriff auf das geschachtelte Feld `fundertype.label.en` verfügbar.

Labels für IDs

Ein typisches Nutzungsszenario bei der Verwendung der lobid-APIs ist die Anzeige von Labels für die URIs, die zur Identifikation von Ressourcen, Autoren, Schlagwörtern etc. verwendet werden. Für Anwendungen, die auf dem alten System basierten, implementierten wir das Nachschlagen dieser Labels in unterschiedlichen Formen. Um diesen Anwendungsfall zu vereinfachen, liefern die neuen APIs die Labels mit den IDs zusammen aus, soweit dies möglich und sinnvoll ist.

In den alten Daten hatten wir etwa zur Identifikation des Mediums einer Publikation nur einen URI:

```
"medium" : "http://rdvocab.info/termList/RDAproductionMethod/1010"
```

Um nun ein Label für einen solchen URI anzuzeigen, mussten wir in den Client-Anwendungen, die die lobid-APIs nutzten, Zuordnungen etwa in Form von Mapping-Tabellen verwalten. In den neuen APIs liefern wir die Labels mit den IDs zusammen aus (aus Konsistenzgründen wird auch hier auf oberster Ebene ein einzelner Wert als Array geliefert, s.o.):

```
"medium": [{
  "id": "http://rdaregistry.info/termList/RDAproductionMethod/1010",
  "label": "Print"
}]
```

Wie die Erstellung des JSON-LD allgemein unterscheidet sich auch die Implementierung dieser Labels in den oben beschriebenen Umsetzungen. In lobid-organisations ist die Ergänzung der Labels (wie alle Aspekte der JSON-Erzeugung) Teil der Datentransformation. In lobid-resources wird eine labels.json-Datei während der Konvertierung von N-Triples in JSON-LD verwendet. lobid-gnd schließlich benutzt einen Bootstrapping-Ansatz, bei dem die vorige Version des Dienstes als Quelle für die Labels verwendet wird, ergänzt um weitere Quellen wie die GND-Ontologie. Details zur Datentransformation in lobid-gnd finden sich in Steeg et al. (2018b).

Zwischenfazit: JSON-LD ist nicht gleich JSON-LD

Eine zentrale Schlussfolgerung unserer Erfahrung mit JSON-LD ist, dass JSON-LD sehr unterschiedlich erzeugt und verwendet werden kann. Wie es erzeugt wird, hat dabei große Auswirkungen darauf, wie es verarbeitet werden kann und wie nützlich es je nach fachlichem Hintergrund erscheint. Eine reine RDF-Serialisierung wie in unserem alten System kann etwa perfekt passen, wenn sowieso mit einem RDF-Modell gearbeitet wird, während sie Web-Entwickler*innen, die mit JSON vertraut sind, absurd und schwer verwendbar erscheinen wird. Diese sehr unterschiedlichen Strukturierungsmöglichkeiten von JSON-LD stellen eine Herausforderung für die Kommunikation über die Nützlichkeit von JSON-LD dar. Hierin liegt aber zugleich die Stärke von JSON-LD, das mit seiner Doppelnatur – als RDF-Serialisierung und als einfacher Weg, JSON-Daten zu vernetzen – unterschiedliche Nutzungsszenarien abdeckt.

Vokabulare

Ein zentraler Aspekt jeder Linked-Data-Anwendung sind die genutzten RDF-Vokabulare und Ontologien. In lobid-organisations verwenden wir schema.org als Basisvokabular. lobid-resources basiert auf DC Terms, Bibframe, der Bibliographic Ontology (Bibo), schema.org und anderen, siehe für Details Ewertowski & Pohl (2017). Grundlage der Daten in lobid-gnd ist die GND-Ontologie (Haffner 2018).

Benutzerschnittstellen

Über die hier skizzierten APIs und Datenstrukturen hinaus bietet lobid in der neuen Version (im Gegensatz zur rudimentären Darstellung der alten Dienste) Suchoberflächen für Endnutzer*innen mit erweiterten Funktionen wie facettierter Suche und Kartenvisualisierungen. Eine ausführliche Darstellung der Funktionalitäten am Fallbeispiel lobid-gnd findet sich in Steeg et al. (2018b).

Entwicklungsprozess

Das lobid-Kernteam besteht seit 2012 aus den drei Autoren dieses Artikels, d. h. aus einer bibiothekarischen Fachkraft und zwei Entwicklern. Simon Ritter trug 2014–2016 als Entwickler maßgeblich zur Umsetzung von lobid-organisations bei. Christoph Ewertowski war 2017–2018 Teil des Teams und dabei eine große Unterstützung im bibliothekarischen Bereich, insbesondere bei der Verbesserung der Datentransformation. Insgesamt werden für Entwicklung, Pflege und Betrieb von lobid je nach anstehenden Aufgaben und sonstigen laufenden Projekten etwa 1,5 bis 2,5 Vollzeitäquivalente eingesetzt. Im folgenden wird skizziert, wie die Teammitglieder die Arbeit an der Entwicklung von lobid organisieren.

Open Source

Wir entwickeln die lobid-Dienste als Open Source Software auf GitHub. Wir veröffentlichen nicht nur Ergebnisse auf GitHub, sondern der gesamte Prozess findet dort statt, d.h. Planung, Issue Tracking & Diskussion, Pull Requests, Implementierung sowie Testen der Software. GitHub hat einen integrierten Issue Tracker, dessen primäres Organisationsmittel beliebige farbige Labels sind, die sich vielseitig anwenden lassen (s.u.). Dieser Issue Tracker ermöglicht es auf einfache und funktionale Weise, andere Prozesse in GitHub zu referenzieren, beispielsweise können so Links zu Code, Commits und Benutzern erstellt werden.

Visualisierung

GitHub Issues sind immer mit einem GitHub Code Repository assoziiert. Für die Bereitstellung von lobid.org werden derzeit neun Repositories auf GitHub verwendet, dazu kommen weitere Repositories etwa für das lobid Blog. Für einen einheitlichen Blick auf alle vom Team bearbeiteten Issues in sämtlichen Repositories verwenden wir zur Visualisierung des Workflows Waffle. Hierbei handelt es sich um ein Kanban Board mit GitHub-Integration, bei dem jedes GitHub Issue einer Karte entspricht und die Spalten des Boards Labels der GitHub-Issues entsprechen.

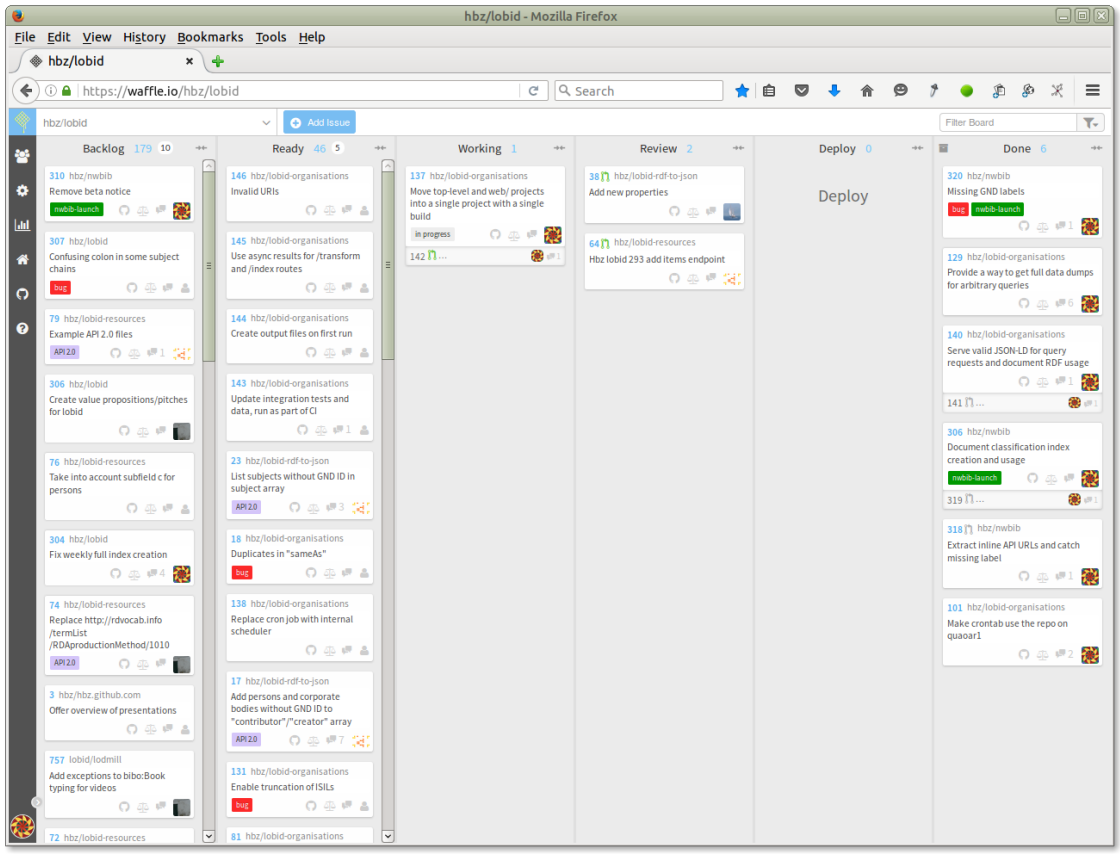

Abbildung 5: Das lobid Kanban Board auf Basis von waffle.io

In unserem Prozess durchläuft eine Karte das Board von links nach rechts. Priorisierte Karten schieben wir in der jeweiligen Spalte nach oben. Karten, die Fehler (Bugs) beschreiben, werden generell priorisiert.

Backlog	Ready	Working	Review	Deploy	Done
Neue Issues ohne Label	Bereit, d.h. Anforderungen und Abhängigkeiten sind klar	In Bearbeitung	In Überprüfung	Bereit für Produktion	In Produktion

Begutachtung

Ein Kernelement unseres Entwicklungsprozesses, durch das bibliothekarische Anforderungen und Entwicklung miteinander verzahnt werden, sind die Begutachtungen bzw. Reviews. Hier unterscheiden wir zwischen funktionalem Review, einer fachlich-inhaltlichen Begutachtung aus bibliothekarischer Sicht, und Code Review, einer implementationstechnischen Begutachtung aus Entwicklungssicht.
Zur Einleitung des funktionalen Reviews stellt einer unserer Entwickler neue oder reparierte Funktionalität auf dem Testsystem bereit und beschreibt im korrespondierenden Issue, wie getestet werden kann (z. B. durch Angabe von Links auf die betreffenden Seiten im Testsystem). Dann weist er das Issue einem Team-Mitglied zur Begutachtung zu. Dieses testet, gibt Feedback (bei Bedarf aktualisiert der Entwickler den Code und die Version auf dem Testsystem mehrfach), und schließt die Begutachtung mit einem "+1" Kommentar ab.

Nach Abschluss des funktionalen Reviews weist der Begutachter die zum Issue gehörigen Code-Anpassungen (in Form eines Pull Request) einem anderen Entwickler zur Begutachtung zu (Code Review). Je nach Fall inspiziert dieser nur den Diff im Pull Request oder er testet den Branch lokal. Die Ausführung des Builds und der Tests erfolgt automatisch im Pull Request durch Travis CI, ein in GitHub integrierter Continuous-Integration-Dienst. Auch hier wird die Begutachtung mit einem "+1" Kommentar abgeschlossen, der Begutachter weist das Issue wieder dem Entwickler zu, und verschiebt es in die Spalte "Deploy".

Nach Abschluss beider Begutachtungsschritte wird die neue bzw. reparierte Funktionalität auf dem Produktivsystem installiert. Details zu unserem Entwicklungsprozess finden sich in unserer Dokumentation und in Steeg (2016).

Dokumentation

Bei der Dokumentation einer API gibt es unterschiedliche Aspekte: das Datenset als Ganzes, die Struktur von API-Anfragen und -Antworten, die verwendeten RDF-Properties und -Klassen, sowie Provenienzinformationen. Im Folgenden beschreiben wir unsere Herangehensweise an die Dokumentation von lobid-resources und lobid-organisations, mit dem Schwerpunkt auf dem zuletzt genannten Dienst.

Dokumentation des Datensets

Um für die menschliche und maschinelle Nutzung der Daten einen Überblick zu geben, folgen wir im Wesentlichen der W3C-Empfehlung für Daten im Web (Lóscio 2017). Das Ergebnis ist eine JSON-LD-Datei und eine daraus generierte HTML-Version. Im Gegensatz zu den Beispielen der W3C-Empfehlung verwenden wir so weit wie möglich Vokabular von schema.org anstelle von DC Terms und des DCAT-Vokabulars. Die folgende Abbildung zeigt die HTML-Version der lobid-organisations Datenset-Beschreibung.

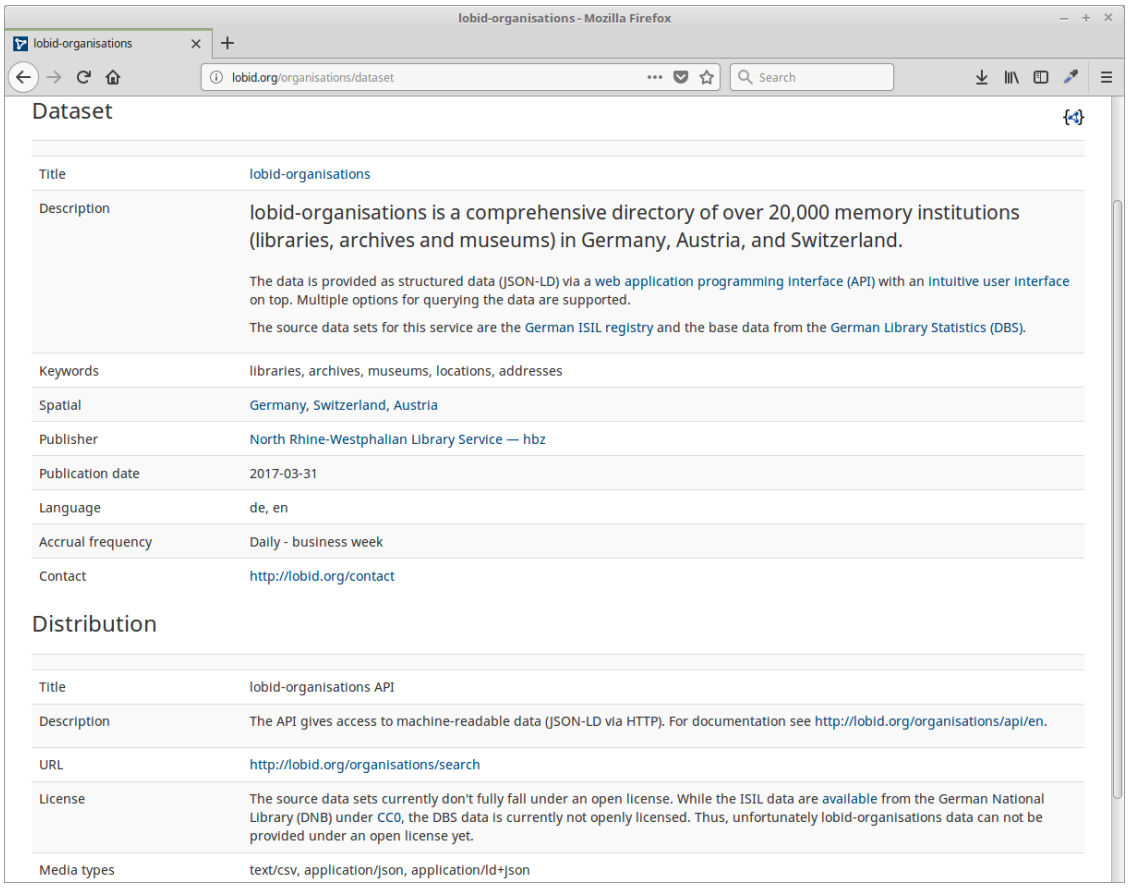

Abbildung 6: Beschreibung des lobid-organisations Datensets

Dokumentation der API

Die API-Dokumentation (lobid-organisations, lobid-resources, lobid-gnd) führt zunächst grundlegende Konzepte der API anhand von Beispielen ein und zeigt darauf aufbauend komplexere Anfragen und spezielle Fälle wie die Suche nach URLs (in denen bestimmt Zeichen maskiert werden müssen). Im Hinblick auf eine vollständige Referenz zu den Suchmöglichkeiten verweisen wir auf die Lucene-Dokumentation (Elasticsearch basiert auf Lucene und verwendet eine kompatible Abfragesyntax).

Im weiteren Verlauf beschreibt die Dokumentation die unterstützten Inhaltstypen mit Beispielanfragen. Wir beschreiben den Einsatz der API zur Umsetzung einer Autosuggest-Funktionalität mit einem in die Dokumentationseite eingebetteten, funktionstüchtigen Beispiel. Schließlich beschreiben wir spezifische Funktionen der einzelnen Dienste wie ortsbezogene Abfragen, CSV-Export, und die Integration der APIs in OpenRefine (siehe dazu auch Steeg et al. 2018b sowie Steeg & Pohl 2018).

Dokumentation mit Beispielen

Um einen leichten Zugang zu einem Schema und seiner Verwendung zu bekommen, wird häufig nach Beispielen gesucht. Leider sind Beispiele oft nur zweitrangige Elemente einer Dokumentation, wenn sie überhaupt gegeben werden. Sehr verbreitet ist dagegen ein beschreibender Ansatz zur Dokumentation von Vokabularen oder Applikationsprofilen, bei dem Elemente in einer Tabelle (häufig in einem PDF) aufgelistet werden, mit Beschreibungen verschiedener Aspekte in mehreren Spalten.

Eine Ausnahme bildet schema.org, das viele Beispiele bietet. Aber selbst hier sind die Beispiele, wenn vorhanden, ein Anhängsel der Beschreibung (z. B. ist es schwierig zu erfahren, wie die Property `publication` oder die Klasse `PublicationEvent` verwendet werden). Wir sind der Ansicht, dass Beispiele Kernelemente der Dokumentation sein sollten. Seitenlange Tabellen, die Elemente eines Metadatenschemas auflisten, halten wir dagegen weder für sehr hilfreich noch für praktisch. Aus diesem Grund experimentierten wir damit, wie Beispiele ins Zentrum der Dokumentation gerückt werden können.

Web-Annotationen für API-Dokumentation

Beispiele alleine sind nicht hinreichend (Gruenbaum 2009), aber benötigen wir wirklich eine Tabelle, in der jedes Element unserer Daten beschrieben wird? Wenn wir das Beispiel in den Mittelpunkt stellen, können wir die strukturierten beschreibenden Daten (Name, Beschreibung, etc.) direkt dem Beispiel beifügen?

Hier kommen Werkzeuge zur Web-Annotation ins Spiel. Wir verwenden hypothes.is zum Annotieren von Beispielen unserer produktiven JSON-LD-Daten. Unser erster Ansatz war, direkt die JSON-Darstellungen zu annotieren (z. B. http://lobid.org/organisations/DE-38M.json), doch dann würden die Annotationen nur bei Verwendung des hypothes.is Chrome-Plugins sichtbar. Eine weitere Option ist die Verwendung des hypothes.is via service, doch dieser unterstützt keine Annotation von Textdateien. Wir entschlossen uns daher, die JSON-Beispiele in die HTML-Dokumentationsseite einzubetten und hypothes.is über JavaScript einzubinden.

In lobid-organisations reicht aufgrund der homogenen Daten die Annotation eines einzigen Beispiels aus. Um die wesentlichen Felder in den lobid-resources-Daten abzudecken, mussten wir jeweils ein Beispiel verschiedener Ressourcentypen (Buch, Periodikum, Artikel, Serienband) annotieren. Für lobid-gnd haben wir den Annotationsansatz noch nicht umgesetzt.

Zum Zwecke der Dokumentation wird jedes JSON-Schlüsselwort mit den folgenden Informationen annotiert:

- *Name*: eine menschenlesbare Bezeichnung für das Feld
- *Beschreibung*: eine kurze Beschreibung der Information in diesem Feld
- *Abdeckung*: die Anzahl der Ressourcen im Datenset mit Informationen in diesem Feld.
- *Anwendungsfälle*: Beispiele zur Verwendung der Information in diesem Feld, häufig mit Beispielanfragen
- *URI*: die RDF-Property, die diesem Feld entspricht (d.h. der auf den JSON-Key im JSON-LD-Kontext gemappte URI)
- *Provenienz*: Informationen über die Felder in den Quelldaten, aus denen die Information in diesem Feld erzeugt wurde

Die ersten beiden Punkte (Name und Beschreibung) sowie der URI werden bei allen Keys angegeben, die anderen Werte sind (noch) nicht überall verfügbar. Wir versuchen durch Beispielanfragen in den Annotationen, ein Gefühl für Nutzungsmöglichkeiten der API zu vermitteln, insbesondere in den Abschnitten 'Abdeckung' und 'Anwendungsfälle'.

Unter http://lobid.org/organisations/api kann man die annotationsbasierte Dokumentation in Aktion sehen (für lobid-resources siehe http://lobid.org/resources/api). Im Abschnitt zu JSON-LD öffnet sich durch einen Klick auf die hervorgehobenen JSON-Keys die hypothes.is-Seitenleiste mit Informationen über das entsprechende Element. Hier zum Beispiel die Annotation für das "rs" Feld:

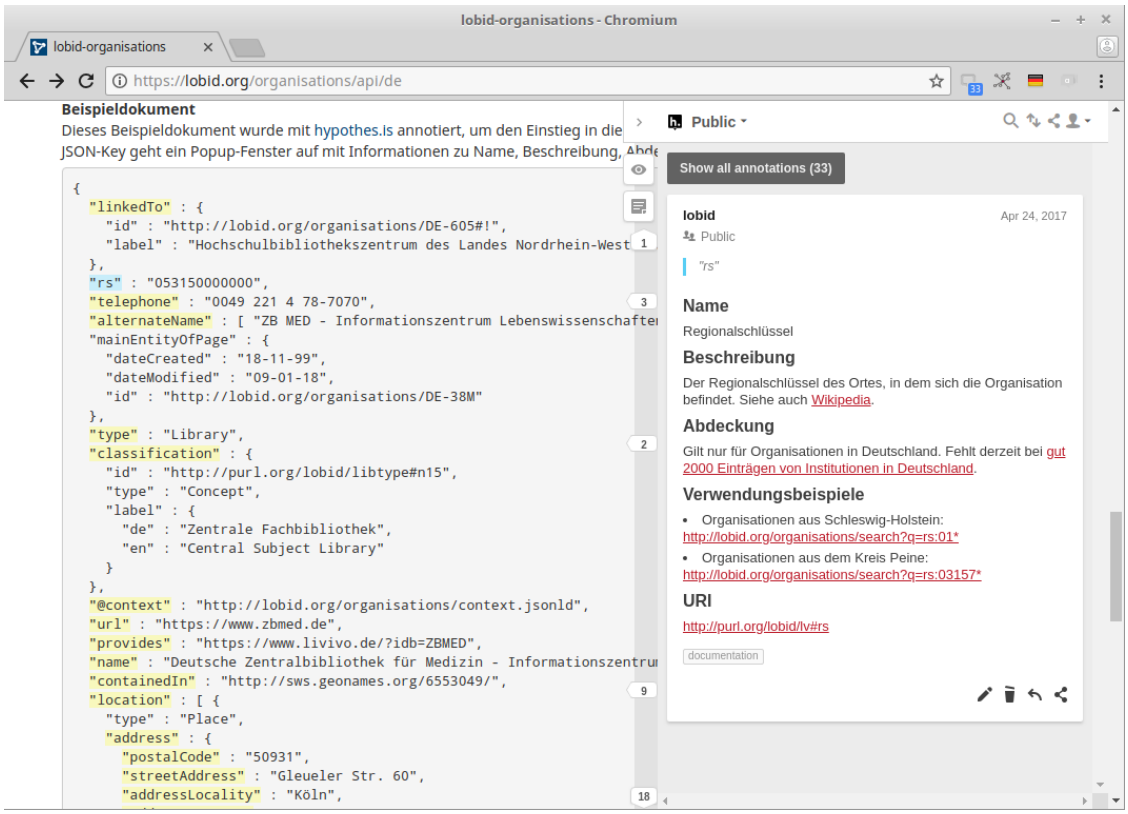

Abbildung 7: Beispielannotation des "rs"-Schüssels in lobid-organisations"

Vorteile

Die Beispiele, die zur Dokumentation annotiert werden, sollten im besten Fall Live-Daten aus dem Produktivsystem sein. Dies gewährleistet, dass die Beispiele und damit die Dokumentation im Fall von Änderungen automatisch aktuell bleiben. Ein manueller Synchronisationsaufwand zwischen Daten und Dokumentation wird dadurch hinfällig.

Wir sind der Meinung, dass dieser Dokumentationsansatz hilfreicher ist als der traditionelle beschreibende Ansatz. Er bietet Nutzenden eine intuitive und interaktive Schnittstelle um die Daten der lobid-APIs zu erkunden und zu verstehen. Bei Fragen oder Unklarheiten kann innerhalb der hypothes.is-Seitenleiste auf die Annotationen geantwortet werden. So können spezifische Teile der Dokumentation in ihrem unmittelbaren Zusammenhang diskutiert werden.

Ausblick

Auf Basis der seit einigen Jahren gewonnenen Erfahrungen und Entwicklungen im Bereich Linked Open Data und APIs bietet lobid mit den aktuellen Web-APIs eine übersichtliche, performante und vielseitige Infrastruktur für bibliothekarische Daten an. Die lobid-Quelldaten werden in Systemen gepflegt, die vermutlich in den kommenden Jahren abgelöst werden. Diese Entwicklung wird grundlegende Anpassungen der lobid-Importstrecke erfordern.

Ob bei der Entwicklung und dem Einsatz neuer Syteme Ansätze verfolgt werden, die ein Angebot von Web-APIs enger an das Mastersystem koppeln wird sich zeigen. Wir würden es jedenfalls sehr begrüßen, wenn das Angebot von Web-APIs zur Nutzung durch Entwickler*innen in Zukunft bereits bei der Datenproduktion und -haltung im Fokus stünde.

Kontakt

Wir freuen uns über Rückmeldungen und Fragen zur Nutzung der lobid-Dienste. Möglichkeiten, mit dem lobid-Team Kontakt aufzunehmen, finden sich unter http://lobid.org/team.

Referenzen

Christoph, Pascal, Fabian Steeg, Christoph Ewertowski, Adrian Pohl & Jan Schnasse (2018): *lobid-resources* [Software]. Zugriff am 2018-09-21. Verfügbar unter https://doi.org/10.5281/zenodo.1423194

Ewertowski, Christoph & Pohl, Adrian (2017): *Which vocabularies to use for bibliographic descriptions?* [online]. Zugriff am 2018-09-12. Verfügbar unter: http://blog.lobid.org/2017/04/19/vocabulary-choices.html

Gruenbaum, Peter (2010): *Web API Documentation Best Practices* [online]. Zugriff am 2018-09-12. Verfügbar unter: https://www.programmableweb.com/news/web-api-documentation-best-practices/2010/08/12

Haffner, Alexander (2018): *GND Ontology* [online]. Zugriff am 2018-09-12. Verfügbar unter: http://d-nb.info/standards/elementset/gnd

Longley, Dave, Manu Sporny, Gregg Kellogg, Markus Lanthaler & Niklas Lindström (2018): *JSON-LD 1.1 Framing – An Application Programming Interface for the JSON-LD Syntax* [online]. Zugriff am 2018-09-12. Verfügbar unter: https://json-ld.org/spec/latest/json-ld-framing/

Lóscio, Bernadette Farias, Caroline Burle & Newton Calegari (2017): *Data on the Web Best Practices* [online]. Zugriff am 2018-09-12. Verfügbar unter: https://www.w3.org/TR/dwbp/, hier insbesondere Abschnitt "8.2 Metadata": https://www.w3.org/TR/dwbp/#metadata

Pohl, Adrian (2010): *Open Data im hbz-Verbund*. In: ProLibris 2010 (3), S. 109-113. Preprint frei zugänglich unter: http://hdl.handle.net/10760/14838

Pohl, Adrian, Fabian Steeg, Simon Ritter, Philipp v. Böselager, Pascal Christoph & Jakob Voß (2018): *lobid-organisations* [Software]. Zugriff am 2018-09-21. Verfügbar unter https://doi.org/10.5281/zenodo.1423192

Sanderson, Rob (2016): *Community Challenges For Practical Linked Open Data* [online]. Vortragsfolien zur Linked Pasts 2 Keynote vom 15.12.2016. Zugriff am 2018-09-12. Verfügbar unter: https://de.slideshare.net/azaroth42/community-challenges-for-practical-linked-open-data-linked-pasts-keynote

Sanderson, Rob (2018): *Shout it Out: LOUD* [online]. Vortragfolien zur EuropeanaTech Keynote am 15.05.2018. Zugriff am 2018-09-12. Verfügbar unter https://de.slideshare.net/azaroth42/europeanatech-keynote-shout-it-out-loud

Schnasse, Jan & Christoph, Pascal (2018): *etikett: Version 0.1.0* [Software]. Zugriff am 2018-09-12. Verfügbar unter http://doi.org/10.5281/zenodo.1406968

Sporny, Manu, Dave Longley, Gregg Kellogg, Markus Lanthaler & Niklas Lindström (2014): *JSON-LD 1.0. A JSON-based Serialization for Linked Data* [online]. Zugriff am 2018-09-12. Verfügbar unter https://www.w3.org/TR/2014/REC-json-ld-20140116/

Steeg, Fabian (2014): *One issue with JSON-LD that seems not so pragmatic* [online]. Zugriff am 2018-09-12. Verfügbar unter http://fsteeg.com/notes/one-issue-with-json-ld-that-seems-not-so-pragmatic

Steeg, Fabian (2015a): *Why LOD needs applications, and libraries need APIs* [online]. Zugriff am 2018-09-12. Verfügbar unter http://fsteeg.com/notes/why-lod-needs-applications-and-libraries-need-apis

Steeg, Fabian (2015b): *More self-containedness, less code sharing* [online]. Zugriff am 2018-09-12. Verfügbar unter http://fsteeg.com/notes/more-self-containedness-less-code-sharing.

Steeg, Fabian (2016): *Der Lobid-Entwicklungsprozess* [online]. Zugriff am 2018-09-12. Verfügbar unter http://hbz.github.io/slides/lobid-entwicklungsprozess/.

Steeg, Fabian & Pohl, Adrian (2018): *GND reconciliation for OpenRefine* [online]. Zugriff am 2018-09-21. Verfügbar unter http://blog.lobid.org/2018/08/27/openrefine.html

Steeg, Fabian, Adrian Pohl, Christoph Ewertowski & Pascal Christoph (2018a): *lobid-gnd* [Software]. Zugriff am 2018-09-21. Verfügbar unter https://doi.org/10.5281/zenodo.1423188

Steeg, Fabian, Adrian Pohl & Pascal Christoph (2018b): *lobid-gnd – Eine Schnittstelle zur Gemeinsamen Normdatei für Mensch und Maschine* [online]. Zugriff am 2018-09-21. Verfügbar unter https://github.com/hbz/lobid/blob/infoprax/doc/lobid-gnd.md

Autoren

Adrian Pohl, Hochschulbibliothekszentrum des Landes NRW, Jülicher Str.6, D-50674 Köln
http://lobid.org/team/ap
pohl@hbz-nrw.de

Fabian Steeg, Hochschulbibliothekszentrum des Landes NRW, Jülicher Str.6, D-50674 Köln
http://lobid.org/team/fs
steeg@hbz-nrw.de

Pascal Christoph, Hochschulbibliothekszentrum des Landes NRW, Jülicher Str.6, D-50674 Köln
http://lobid.org/team/pc
christoph@hbz-nrw.de