swissbib

Komponenten moderner Datenplattformen

MAS BIW M3, 14.2.2020

Günter Hipler
Sebastian Schüpbach
Silvia Witzig

Universität Basel

Agenda

- Einführung swissbib
- Architektur der Datenplattform

  Diskussion: Was bedeutet technische Innovation für Bibliotheken?

- linked.swissbib
- Entwicklung, Deployment, Betrieb

  Diskussion: Notwendige Bedingungen zur aktiven Gestaltung digitaler Transformation?

- Datenanalyse

  Diskussion: Datenanalyse - ein Service von Bibliotheken?

Einführung swissbib

Auslöser/Rahmenbedingungen für das Projekt

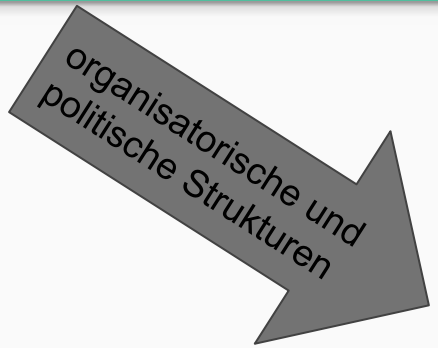
Zielsetzungen und Projektplan

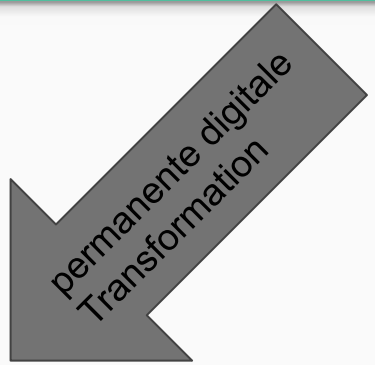
Grundsätze Systemarchitektur

Entwicklungsstationen 2010 - 2018

Standortbestimmung 2018

Auslöser/Rahmenbedingungen für das Projekt





https://www.degruyter.com/view/j/bfup.1999.23.issue-2/bfup.1999.23.2.133/bfup.1999.23.2.133.xml
(Alice Keller / Wolfram Neubauer: Hochschulbibliotheken der Schweiz Position und Ausrichtung, 1999)
https://www.degruyter.com/view/books/9783110553796/9783110553796-002/9783110553796-002.xml
https://www.degruyter.com/view/books/9783110553796/9783110553796-006/9783110553796-006.xml
(Bibliotheken der Schweiz: Innovation durch Kooperation, Festschrift für Susanna Bliggenstorfer, 2018)

* sehr heterogene Strukturen
* unterschiedliche Geldgeber
* Vielzahl unterschiedlicher Bibliotheks (-verwaltungs) Systeme
* zwei unterschiedliche Katalogisierungsformate
* kein Gesamtdiscovery auf schweizerische Ressourcen

Erwartungen von (2008):
* Google-like Suchen
* moderne Oberflächen (Web 2.0)
* schneller Zugriff auf content

* Thema 2004 - 2008: Search Engine Technology and Digital Libraries
http://www.dlib.org/dlib/june04/lossau/06lossau.html
http://www.dlib.org/dlib/september04/lossau/09lossau.html

* “Krise des Webopacs”
* technische Architekturen integrierter Bibliothekssysteme erreichen “End of Life” (u.a Aleph) - technische Umsetzung neuer Anforderungen sehr schwierig (Im IDS 1998 erst eingeführt…)

Zielsetzung und Umsetzungsplan

weltweite Ausschreibung nach Gatt-Kriterien (Juni 2008):

http://www.swissbib.org/doc/mas_zb/ausschreibung_swissbib_2008.pdf

mögliche Lieferformen spiegeln die Hauptanforderung wider:
Modularität sowie Flexibilität in der Zukunft

a) Lieferung der Gesamtlösung
(Komponenten für: Datenaufbereitung, Suchmaschine, Userdiscovery und Maschinen-API)
b) Lieferung einzelner Komponenten durch unterschiedliche Hersteller
explizit nicht ausgeschlossen

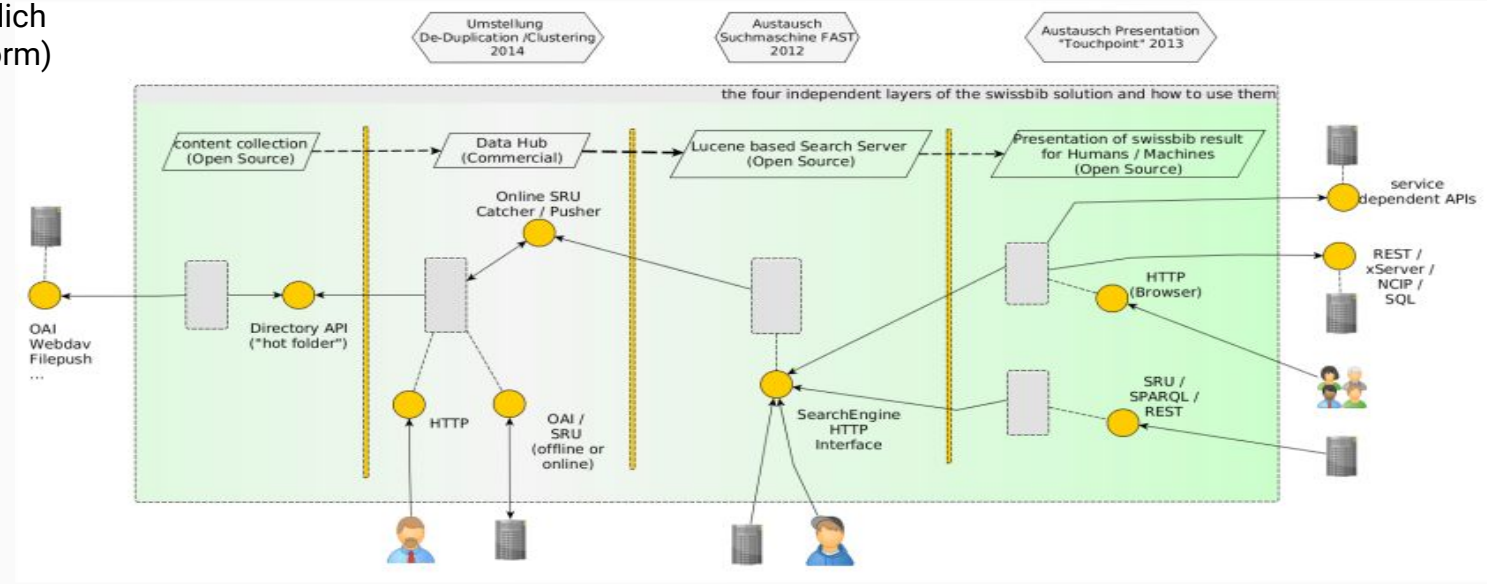
Grundsätze der Lösungs- (System) architektur

Für unsere Anforderungen und Ziele suchten wir nach einer Lösung - nicht nach einem Produkt

→ “gelayerte” (modulare) Architektur
→ voneinander unabhängige Softwarekomponenten
→ offene Schnittstellen zur Kommunikation nach innen und aussen
→ Kombination von Open Source und kommerziellen Komponenten sollte explizit gegeben sein (“make and buy”)

November 2008: Wir hatten die Wahl!

* austauschbare Komponenten
* Schnittstellen nach innen und aussen
* “make and buy” möglich
(OCLC, Lösungsplattform)



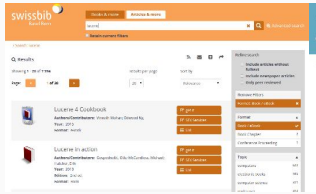
* „Publishing Platform“ (Daten)
* Lucene als Suchmaschine
* Oberflächenkomponente
(ExLibris, Produkt Primo)

Entwicklungsstationen 2010 - 2013
(Umbau / Erweiterungen / Integrationen)

- Umbau Clustering Datenhub
  (flexiblere Möglichkeiten Daten zu clustern/de-duplizieren)

- die swissbib Plattform wird zum regionalen Datenhub zwischen Schweizer Institutionen und worldcat
- OAI als weitere Schnittstelle zum Bereitstellen der aufbereiteten swissbib Daten (z.B. früheres Webportal ETH, Kartenportal, HGK Basel …)
- Austausch der kommerziellen Präsentationskomponente durch die Open Source Lösung Vufind
- Austausch der kommerziellen Suchmaschine FAST durch die Open Source Lösung SOLR / Lucene
- Migration der Server-Infrastruktur von einer kommerziellen Cloud (Holland) zur privaten Cloud (ITS Services Uni Basel)

- zusätzliche lokale views



Entwicklungsstationen 2014 - 2018

(Verbesserung End-User Services / Durchführung “linked - swissbib” / Entwicklung hin zur Dienstleistungsplattform für externe Services und Projekte)

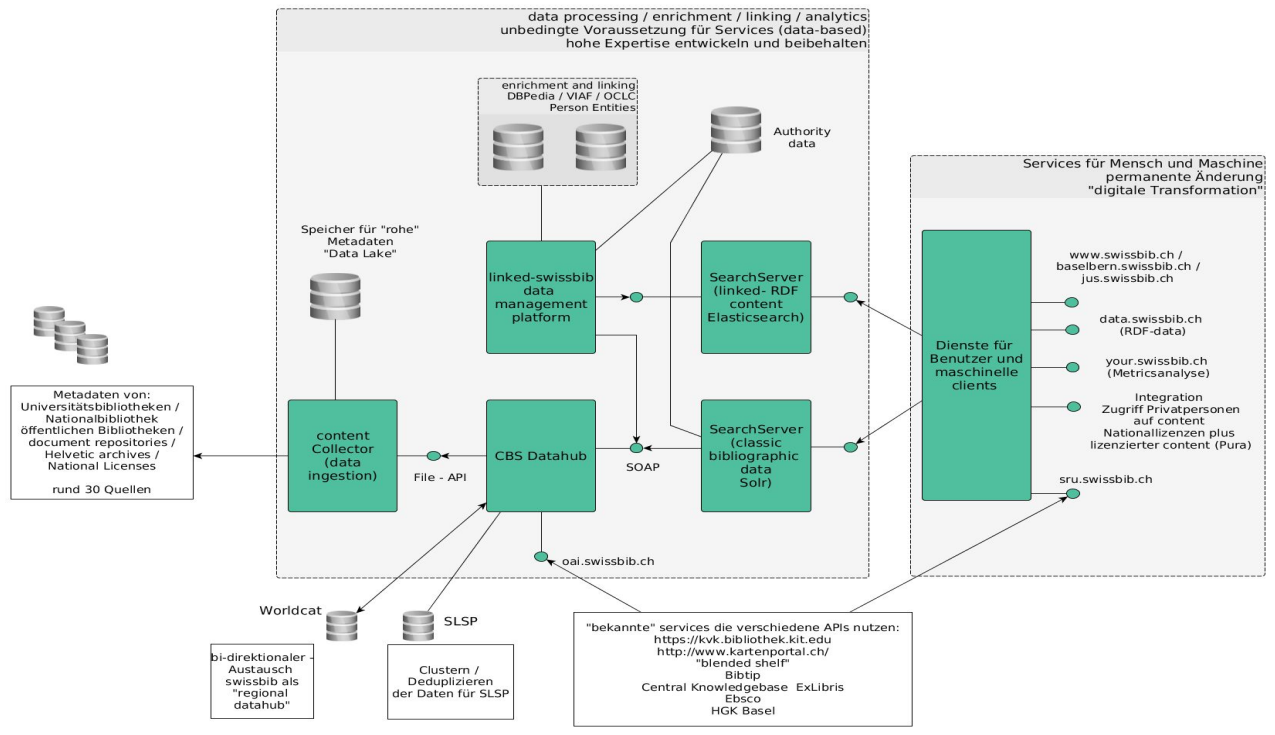
- “Responsive Design” Präsentationskomponente
  (ursprüngliches Design 2010, entwickelt von Fa. Nose, Zürich:
  http://www.swissbib.org/doc/mas_zb/nose/suchergebnis.html)

- Integration zusätzlicher Datenquellen (Boris, Serval, KBs Schaffhausen und Thurgau, IOC, HEMU, ...)
- Nachvollzug des Wechsels hin zu RDA
- Start des Projekts linked.swissbib.ch zusammen mit FH Graubünden und HES Genève

- Integration content P5 Projekt Nationallizenzen und Zugriffsworkflows für Privatpersonen
- Entwicklung Pura service (2018, zusammen mit ZB Zürich)
  http://www.swissbib.org/wiki/index.php?title=Private_User_Remote_Access_(Pura)

- Clustering (De-Duplizierung) der Daten für SLSP Projekt (2018 - 2020)

Standortbestimmung 2018



- swissbib hat sich von einem “Schweizer Metadatenkatalog” (2008) zu einer Datenplattform entwickelt
- eine moderne Datenplattform ist Voraussetzung für kompetitive innovative Services von Bibliotheken in der Zukunft.

digitale Transformation im Jahre 2020

Big Data

mehr unstrukturierte Daten
(auch ForschungsDaten)

Open Source
treibend in jedem
Bereich "data technolgy"

strukturiertes
(semantic) web

Streaming data

Microservices

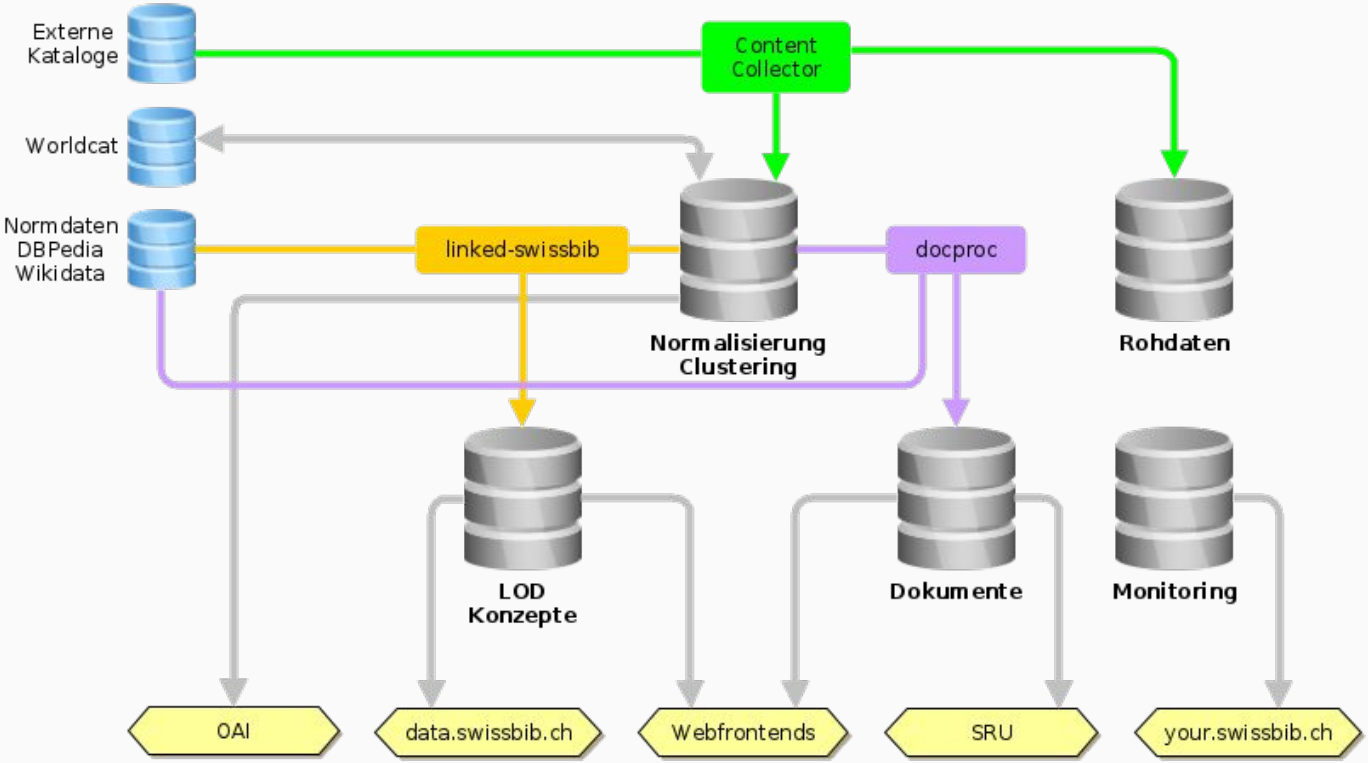
Bereitstellung von Möglichkeiten
zur interaktiven Datenanalyse

Containertechnik
(Orchestrierung)

Zusammenspiel
Public / Private Cloud

Architektur der Datenplattform

Schematische Systemarchitektur Swissbib



Serviceorientierte Architektur: Definition

“Service-oriented architecture (SOA) is a style of software design where **services** are provided to the other components by **application components**, through a communication protocol over a network.”

(https://en.wikipedia.org/wiki/Service-oriented_architecture)

Was sind Komponenten?

- Strukturieren Applikation
- Umfassen Funktionen / Daten einer bestimmten Domäne (hohe Kohäsion)
- Modular:
  - Entkoppelt von anderen Komponenten
  - Stellen Schnittstellen zur Verfügung
  - Implementierung ist abgeschirmt (“Blackbox”)

Was sind Services?

- “Dienstleistungen”
  - von Softwarekomponenten zur Verfügung gestellt und
  - durch andere Komponenten / Endanwender genutzt
- Bausteine der Geschäftslogik einer Applikation
- Verfügbar via (standardisierten) Kommunikationsprotokollen (bspw. HTTP)

SOA: Vorteile, Nachteile

Komponenten sind gut

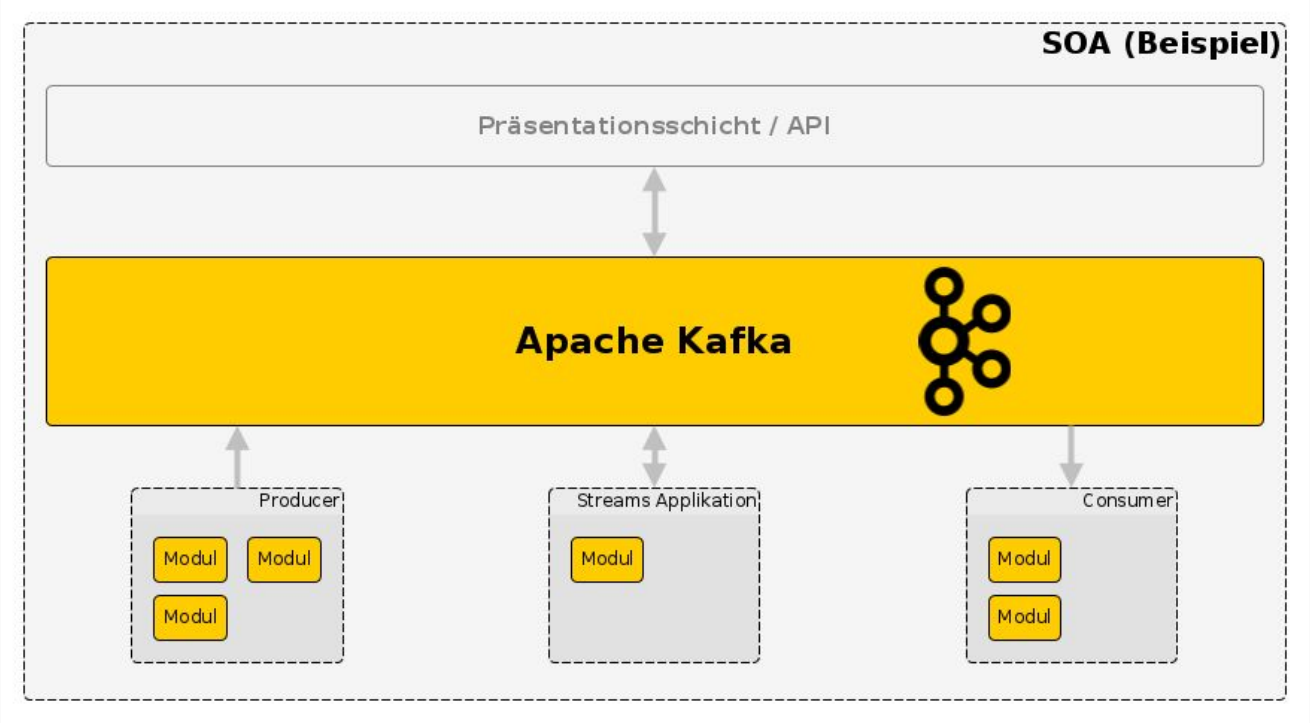
- wiederverwendbar
- ersetzbar
- ausbaubar
- pflegbar
- individuell ausrollbar

=> Flexible Architektur

=> Agile Entwicklung

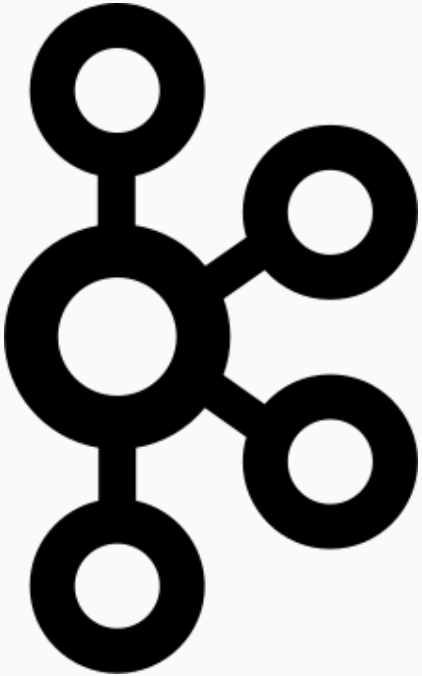
- Komplexe Architektur
- Schlechtere Performanz
- Bestimmung der Granularität der Komponenten schwierig
- Standardisierte und dokumentierte Schnittstellen unabdingbar

SOA am Beispiel Kafka

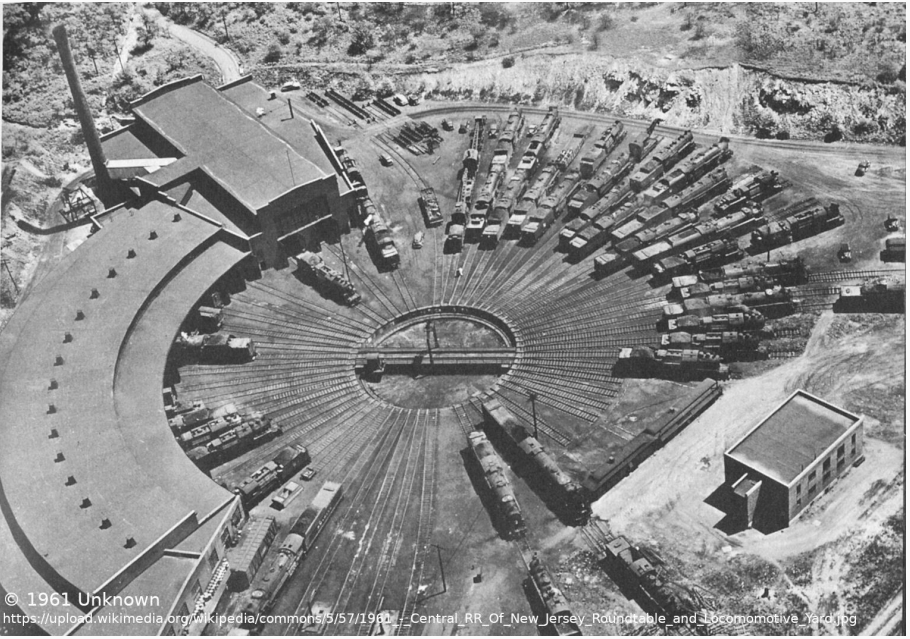


Apache Kafka: Eckdaten

- Projekt der Apache Software Foundation
- Open Source
- Aktuelle Version 2.4.0
- Framework für Erstellen von
  - **Datenpipelines** und
  - **Streaming-basierten Applikationen**



“Datendrehscheibe”



- Dateninput, Datenoutput
- Datentransformation
- Standardisierte Schnittstellen

Schnittstellen

API (Application Programming Interface)
Hier: Softwarebibliothek, welche in Anwendungen eingebunden werden kann

Producer API: Einspielen von Datensätzen nach Kafka

Consumer API: Exportieren von Datensätzen aus Kafka

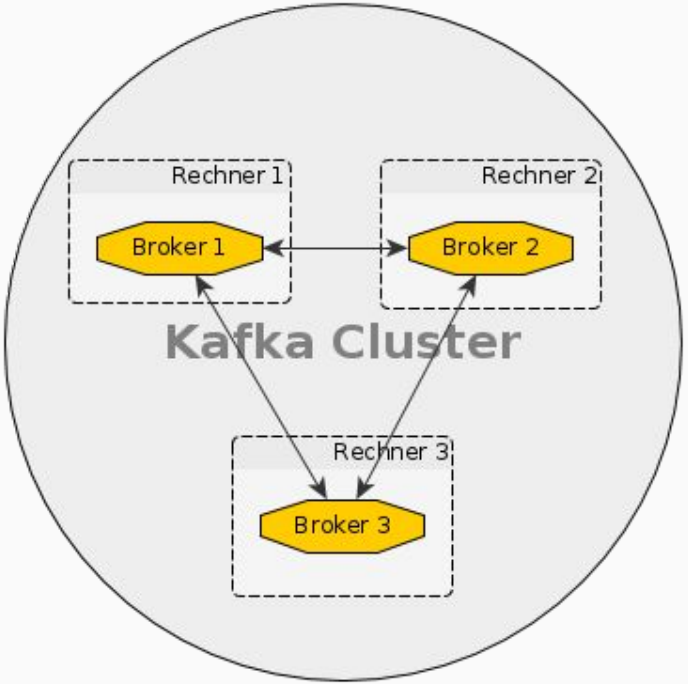
Streams API: Transformation von Datensätzen in Kafka

Brokers

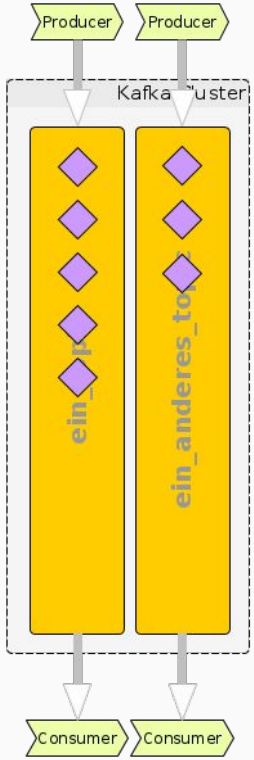
- **Broker**: Kafka-Instanz
- Cluster: Verbund von Brokers (verteilt auf mehrere Rechner)

Brokers *skalieren* Cluster *horizontal*

↗️ Durchsatzrate
↗️ Verfügbarkeit
✓ Parallelisierung
✓ Replikation



Topics + Messages



Topics

- Datenkanäle in Kafka-Cluster
- Producers schreiben in Topics
- Consumers “abonnieren” Topics

Messages

- Einzelne Nachrichten in Topic
- Immer in Form `(Schlüssel, Wert)`

Partitionen

- Topics werden in *Partitionen* unterteilt
- Partitionen: *Message Queues*
- Queue basiert auf FIFO-Prinzip (First in, first out):

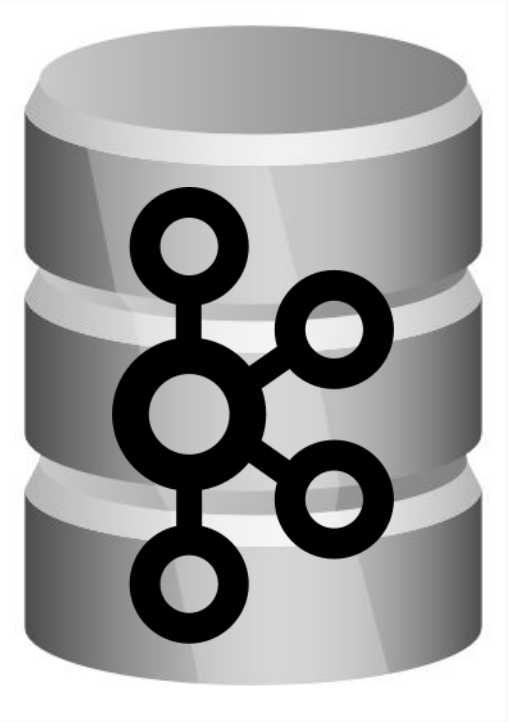
- Schlüssel entscheidet über Partitionszuteilung
  - Gleicher Schlüssel = gleiche Partition
- Kafka verteilt Partitionen auf Brokers
- Replizierung möglich (Ausfallsicherheit)

Persistenz

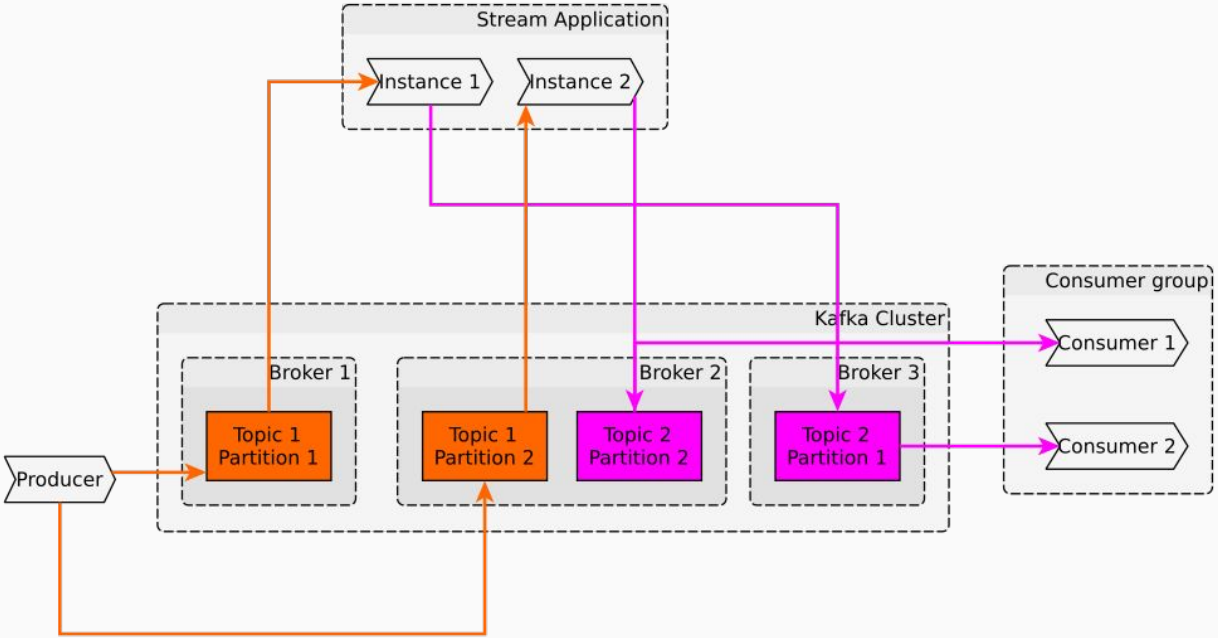
- Partition = Transaktionslog
  - Messages sind nicht modifizierbar
  - Reihenfolge garantiert in Partition (aber nicht in Topic!)
- Daten auf Festplatte persistiert
  - Zeitlich (un-)beschränkte Speicherung
  - Schlüsselbasierte Speicherung (*log compaction*)

Kafka = Datenbanksystem

- Lesender Zugriff sehr performant
- Datenreplikation
- Garantien



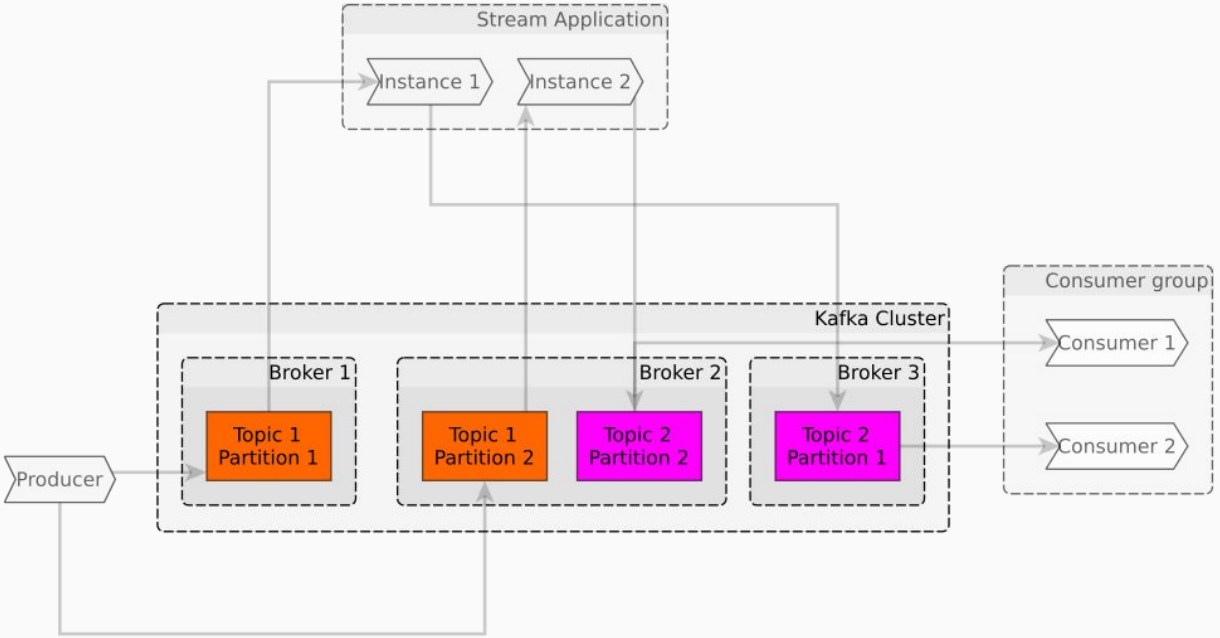
Datenpipeline: Übersicht



Basale Datenpipeline:

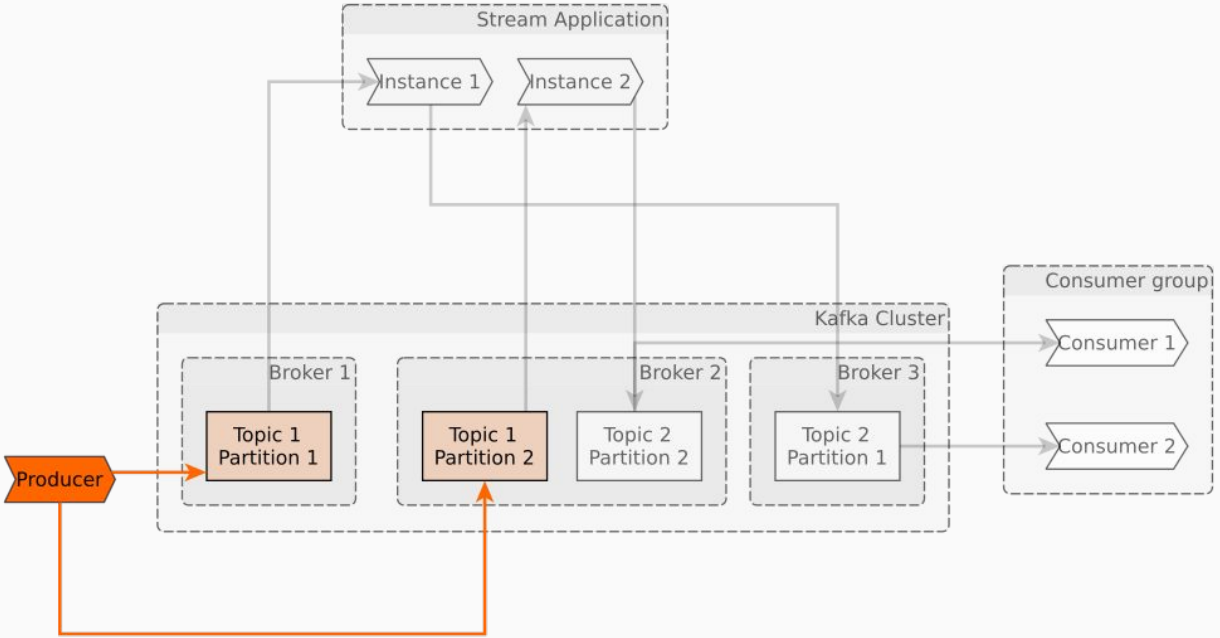
1. Daten werden nach Kafka importiert
2. Daten werden bearbeitet
3. Daten werden aus Kafka exportiert

Datenpipeline: Infrastruktur



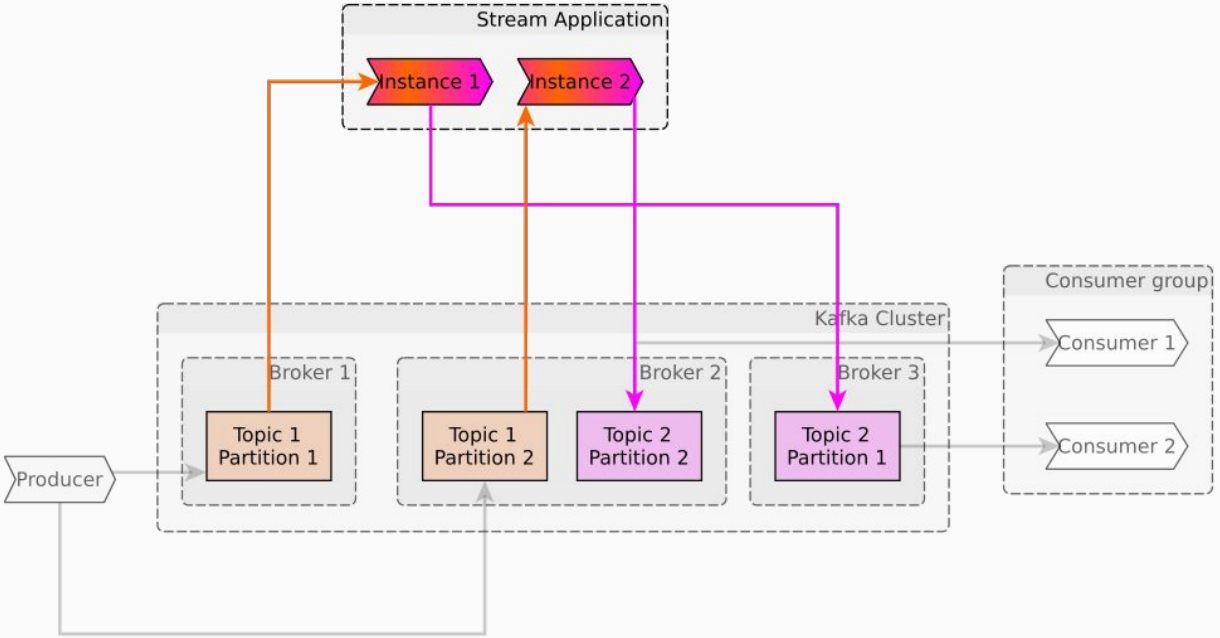
- Kafka Cluster hat drei Brokers
- Zwei Topics (Topic 1, Topic 2)
- Topics jeweils auf zwei Partitionen verteilt (Partition 1, Partition 2)
- Partitionen werden +/- ausgeglichen auf Brokers verteilt

Datenpipeline: Datenimport



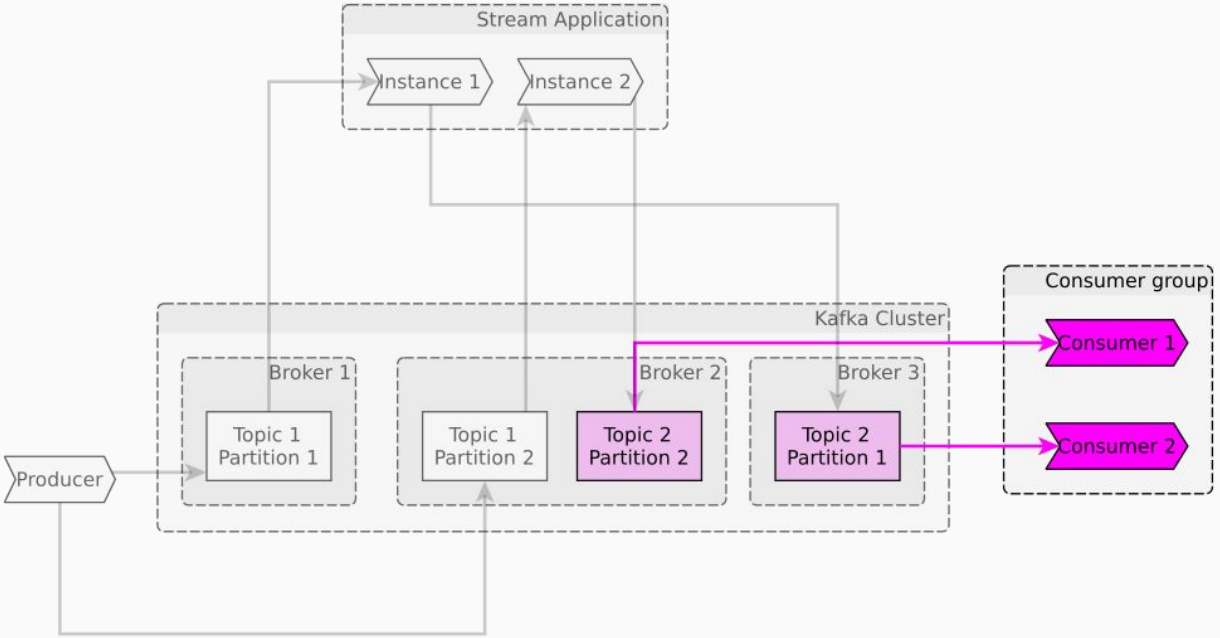
- Producer erstellt Messages
- Messages werden in Topic 1 geschrieben
- Messages werden auf die beiden Partitionen verteilt.
- Schlüssel ist für Zuteilung ausschlaggebend

Datenpipeline: Transformation



- Streams App wird in zwei Instanzen gestartet
- Kafka “bemerkt” Vorhandensein mehrerer Instanzen derselben Applikation durch Verwendung gleicher ID
- Durch mehrfache Instanziierung Parallelisierung möglich

Datenpipeline: Datenexport



- Consumer wird in zwei Instanzen gestartet
- Consumer Group (=Instanzen haben gleiche Group ID) teilt Extraktionsprozess unter sich auf
- Dadurch Parallelisierung analog Streams App möglich

Events und Streams

Event: Eine Aktivität, optional mit Inhalt (*Payload*). Beispiele:

- Nutzerin bewegt Maus (Payload: neue Position)
- USB-Stick wird eingesteckt (Payload: Seriennummer)
- Bibliografische Aufnahme wird erstellt (Payload: Datensatz)

Eventstream

- Strom an diskreten Ereignissen
- Prinzipiell von unendlicher Dauer (*unbounded*)

Events in Kafka

Event = Kafka-Message
Eventstream fliesst durch Topic

Producer: Bildet / transferiert Events als Messages und schreibt diese in Topic
Beispiele:

- Datei wird zeilenweise ausgelesen -> `Message(id, <Zeileninhalt>)`
- Knopf wird gedrückt -> `Message(id, <Zeichen>)`

Consumer: Konsumiert Messages (aus Topic)
Beispiel: Payload wird sequentiell in Datei geschrieben

Streams: Transformiert Events
Beispiel: Neue bibliografische Aufnahme -> Inkrementierung Anzahl der Aufnahmen

Event processing: Kafka Streams

- Anwendung liest aus Topic A (Rolle als Consumer)
- Transformation von Schlüssel und / oder Wert, bspw.
  - Filtern
  - Mapping (`WertA -> WertB`)
  - Branching (`Stream -> StreamA | StreamB`)
- Aggregation von Werten per Schüssel
- Joins: Vereinigung von Events aus Streams via Schlüssel
- Anwendung schreibt in Topic B (Rolle als Producer)

Beispiel

Ziel: Personen mit Lebensdaten und Verweisen auf Autoritätsdateien

Eingabe: Komprimierte Liste von NTriples in der Form:

<http://viaf.org/viaf/101500089> <http://schema.org/sameAs> <http://id.loc.gov/authorities/names/no2009161437> .
<http://viaf.org/viaf/101500155> <http://schema.org/birthDate> "1953" .

Probleme:

- Liste ungeordnet
- Liste enthält irrelevante Properties
- Nicht benötigte Ressourcentypen vorhanden (z.B. Orte)

Beispiel

Producer

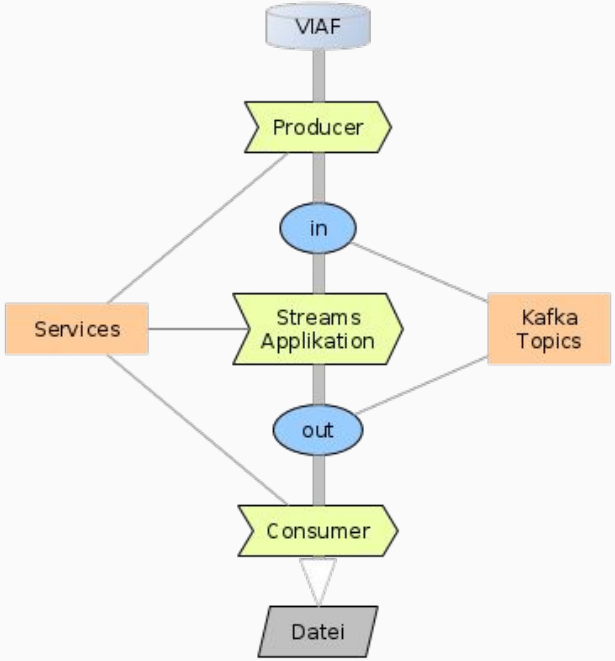
- Download Dump
- Erstellen von Message: `(Subject, (Property, Object))`

Streams-Applikation

- Sortieren nach Ressourcen
- Filtern unnötiger Properties
- Filtern irrelevanter Ressourcen (≠ Person)

Consumer

- Ausgabe in Datei



Streams: Topologie

```
val builder: StreamsBuilder = new StreamsBuilder  // DSL-Builder
val source = builder.stream[String, String]("in")  // Auslesen aus Topic in
val sortedAndFilteredTriples = source  // Deklaration Prozessierungsschritte
 .filter((_, v) => propertyIsInWhitelist(v))
 .groupByKey
 .reduce((agg, v) => if (agg == “”) v else agg + "##" + v)
 .toStream
 .filter((_, v) => resourceIsPerson(v))
sortedAndFilteredTriples.to("out")  // Ausgabe in Topic out (compacted)
val topology = builder.build()  // Erstellen der Topologie
val streams = new KafkaStreams(topology, props)  // Erstellen Applikation
streams.start()  // Start der Applikation
```

Fazit: Welche Probleme löst Kafka?

- Einheitliche Schnittstellen zur Bildung von Datenpipelines
- Horizontale Skalierbarkeit (höherer Durchsatz)
- Ausfallsicherheit (Resilienz, Fehlertoleranz)
- Datenbank-Ersatz
- Abstraktion über Streamprozessierung

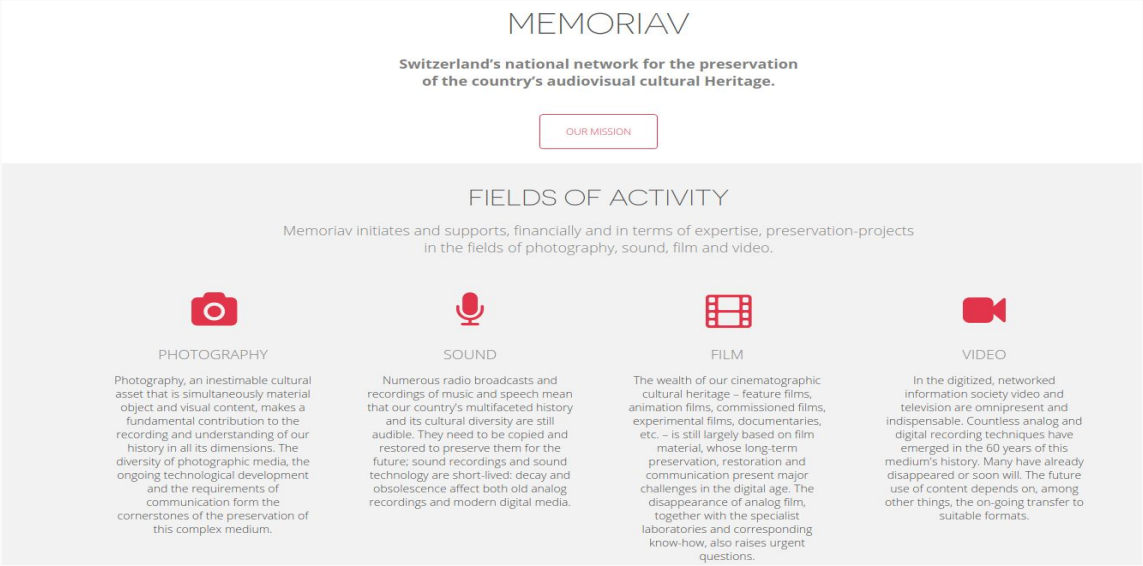
Was bedeutet technologische Innovation für Bibliotheken?

- Diskussion -

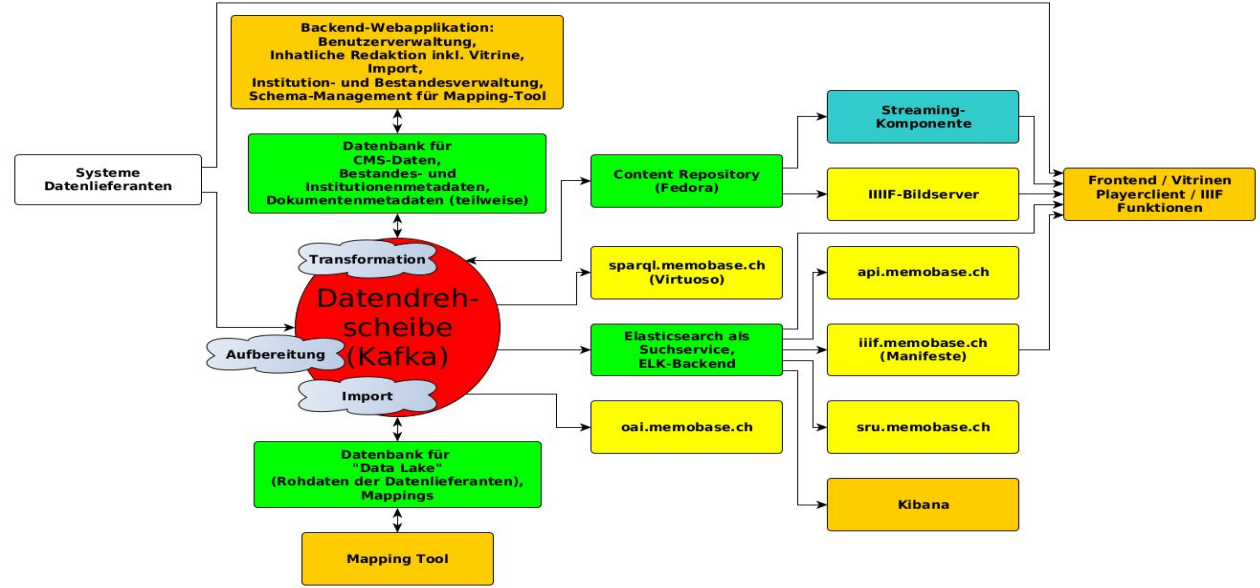
zum Einstieg in die Diskussion

Können die Bausteine der swissbib Datenplattform auch in anderen (datenbasierten) Kontexten verwendet werden?

Lösungsvorschlag der UB Basel in Kooperation mit zwei externen Firmen zur Migration des www.memobase.ch Service der Stiftung Memoriav, Bern (September 2019)



Architektur des Memobase Proposals (angenommen 30.1.2020)



docuteam

Universität Basel
Universitätsbibliothek

OUTER MEDIA

Thesen zur Diskussion

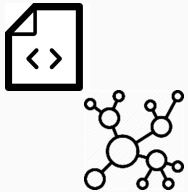
Die (Software-)Bausteine für den Umgang mit Daten unterliegen einem sehr hohen Innovationsdruck (Stichworte “Big Data”, “Artificial Intelligence”, “Echtzeitprozessierung” etc.).
-> Sollen sich die Bibliotheken diesem Innovationsdruck beugen? Oder ist hier Innovation fehl am Platz?
-> Muss die technologische Innovation auch von den Bibliotheken selbst ausgehen?

Für Bibliotheken und Archive ist die Rolle als Vermittler von (Open) Data für eigene und Zwecke Dritter zentral.
-> Wie können sie dieser Rolle gerecht werden?
-> Was brauchen sie auf strategischer, auf technischer Ebene dafür?
-> Von welcher Beschaffenheit müssen die Daten sein, um diese Rolle zu erfüllen?
-> Ist diese Rolle im digitalen Bereich überhaupt noch zeitgemäss, wenn andere (Google, Amazon etc.) dies effizienter und in unvergleichbar grösserem Massstab leisten können?

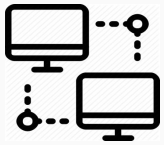
Sind die zukünftigen Bibliotheken Arbeitgeber von SoftwareentwicklerInnen oder ausschliesslich Einkäufer von Software?
-> Was für Konsequenzen können sich aus jeder Position ergeben?
-> Gibt es im Zeitalter, wo kommerzielle Anbieter von Bibliothekssystemen zunehmend in die Cloud gehen, noch einen mittleren Weg?

linked.swissbib

Ziele Projekt linked.swissbib



Konversion der swissbib-Daten in ein RDF-Datenmodell



Das neue Datenmodell anderen (auch über Schnittstellen) zur Nachnutzung anbieten

Die Daten zur Verbesserung von www.swissbib.ch nutzen

Entwickelte Elemente

- Datenmodell
- Datentransformation
- Verlinkung
- Anreicherung
- Workflows (auch für tägliche Verarbeitung)
- Oberflächenelemente
- Schnittstelle

Datenmodell

Benutzergesteuerte Entwicklung des Modells:

Was wollen wir auf der Oberfläche anbieten?

- Aggregationsseiten (eigener und angereicherter Inhalt)
- Knowledge Cards (Inspiration: Google Knowledge Graph)

→ jeweils zu Autoren, Werken und Themen

Datenmodell

6 Bibliografische Konzepte

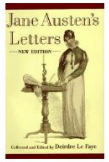
- Bibliographic Resource
- Document
- Item
- (Work)
- Person
- Organisation

https://linked-swissbib.github.io/datamodel/

Vokabularien in linked.swissbib

- Dublin Core (dc/dct)
- Bibliographic Ontology (bibo)
- Bibframe (bf)
- RDA – unconstrained properties (rdau)
- Friend of a friend (foaf)
- DBpedia (dbo)
- Schema (sch)
- GND (gnd)
- Wikidata (wdt)
- Web Ontology Language (owl)
- RDF Schema (rdfs)
- Simple Knowledge Organisation System (skos)

Beispiel Transformation



Jane Austen's letters

Verfasser / Beitragende:	collected and ed. by Deirdre Le Faye
Ort, Verlag, Jahr:	Oxford [etc.] : Oxford Univ. Press, 1995
Beschreibung:	XXVIII, 643 S. ; 23 cm
Format:	Buch (Brief)
Ausgabe:	3rd ed.

Weitere Ausgaben

Thema

Austen, Jane > 1775-1817

Briefsammlung

Briefsammlung > 1796-1817

Verfasser / Beitragende

Austen, Jane

Le Faye, Deirdre

Standorte & bestellen | Beschreibung | Ähnliche Einträge | Felder

- IDS Basel Bern
- IDS St. Gallen
- NEBIS
- RERO - Réseau romand

https://www.swissbib.ch/Record/260865931

Input:

MARC-Aufnahme von swissib

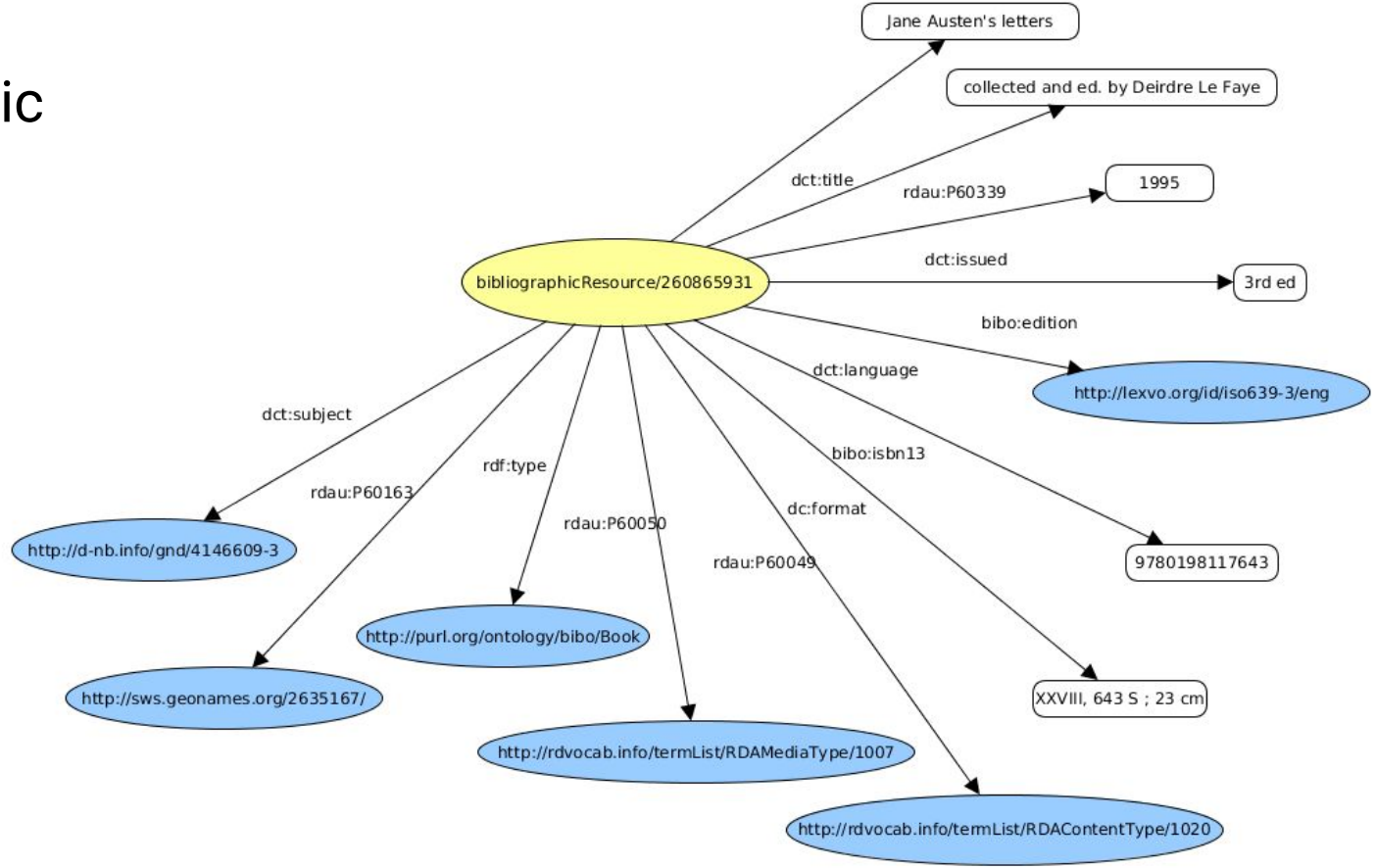
(d.h. gemergt, in CBS verarbeitet)

```
LEADER cam a22 u 4500
001 260865931
003 CHVBK
005 20200106190040.0
008 130816s1995 xxk 00 i eng d
020   |a 0-19-811764-7
0247  |a http://catalogue.bnf.fr/ark:/12148/cb37458092g  |2 url
035   |a (IDSBB)001232874
035   |a (RERO)2146824
035   |a (IDSSG)000098294
035   |a (NEBIS)001491103
040   |a SzZuIDS BS/BE B400  |b ger  |e kids
044   |a xxk  |a xxu
072  7|a s1an  |2 rero
08204|a 823/.7  |2 20
084   |a HL 1682  |2 rvk
1001  |a Austen  |D Jane  |d 1775-1817  |0 (DE-588)118505173
24510|a Jane Austen's letters  |c collected and ed. by Deirdre Le Faye
250   |a 3rd ed.
264  1|a Oxford [etc.]  |b Oxford Univ. Press  |c 1995
300   |a XXVIII, 643 S.  |c 23 cm
504   |a Includes bibliographical references (p. 473-482) and indexes
60017|a Austen, Jane  |0 (IDREF)02669719X  |2 idref
60017|a Austen, Jane  |d 1775-1817  |0 (DE-588)118505173  |2 gnd
60017|a Austen, Jane  |2 idszbz
60010|a Austen, Jane  |d 1775-1817
60010|a Austen, Jane  |d 1775-1817  |x Correspondence
650  7|a Briefsammlung  |0 (DE-588)4146609-3  |2 gnd
650  0|a Authors, English  |y 19th century  |x Correspondence
650  0|a Novelists, English  |y 19th century  |x Correspondence
650  0|a Women novelists, English  |y 19th century  |x Correspondence
650  0|a Young women  |z England  |v Fiction
651  0|a England  |x Social life and customs  |y 19th century  |v Fiction
655  7|a Correspondance  |0 (RERO)A021099251  |2 rero
655  7|a Briefsammlung  |y 1796-1817  |2 gnd-content
7001  |a Le Faye  |D Deirdre
898   |a BK020000  |b XK020000  |c XK020000
898   |a BK020100  |b XK020100  |c XK020100
908   |D 1  |a Briefe = Correspondance
912  7|a le  |2 SzZuIDS BS/BE
912  7|a M367  |2 Z01
949   |B RERO  |F RE01001  |b RE01001  |c RE010010001  |j NA 96.519  |z [3rd impr.]
949   |B RERO  |F RE61001  |b RE61001  |c RE610010021  |j BGE Taa 113
949   |B RERO  |F RE61036  |b RE61036  |c RE610360008  |j A AUST,J 3*Jan al  |s BFLA 97759  |z [3rd impr.]
949   |B IDSSG  |F HSG  |b SGSBI  |c BURON  |j HL 1682 .995(3)
949   |B NEBIS  |F Z19  |b Z19  |c Z19ES  |j Ol Aus 24
949   |B NEBIS  |F Z01  |b Z01  |c 03  |j GL 40607
949   |B IDSBB  |F A100  |b A100  |c 100FM  |j UBH AOe 16754  |x Akz: ba/uub-/k/274350
949   |B IDSBB  |F B400  |b B400  |c 400J4  |j BeM RAA 816  |x Akz: be/stub/k/96/102085
949   |B NEBIS  |F UKOMP  |b UKOMP  |c ULKBU  |j A Aust 1
950   |B IDSBB  |P 100  |E 1-  |a Austen  |D Jane  |d 1775-1817  |0 (DE-588)118505173
950   |B IDSBB  |P 700  |E 1-  |a Le Faye  |D Deirdre
950   |B RERO  |P 100  |E 1-  |a Austen  |D Jane  |d 1775-1817  |0 (IDREF)02669719X  |4 cre
950   |B RERO  |P 700  |E 1-  |a Le Faye  |D Deirdre  |0 (RERO)A003503340
950   |B IDSSG  |P 100  |E 1-  |a Austen  |D Jane
950   |B IDSSG  |P 700  |E 1-  |a Le Faye  |D Deirdre
950   |B NEBIS  |P 100  |E 1-  |a Austen  |D Jane  |d 1775-1817  |0 (DE-588)118505173
950   |B NEBIS  |P 700  |E 1-  |a Austen  |D Jane  |d 1775-1817  |0 (DE-588)118505173
986   |a SWISSBIB  |b 024504246
986   |a ORANGE  |b 260865931
```

MARC zu RDF: Bibliographic Resource

- Die am häufigsten vorkommenden MARC Felder werden transformiert
- Für Oberfläche notwendige MARC Felder werden transformiert
- Bibliographic Resource enthält Informationen aus 22 MARC-Feldern

Bibliographic Resource



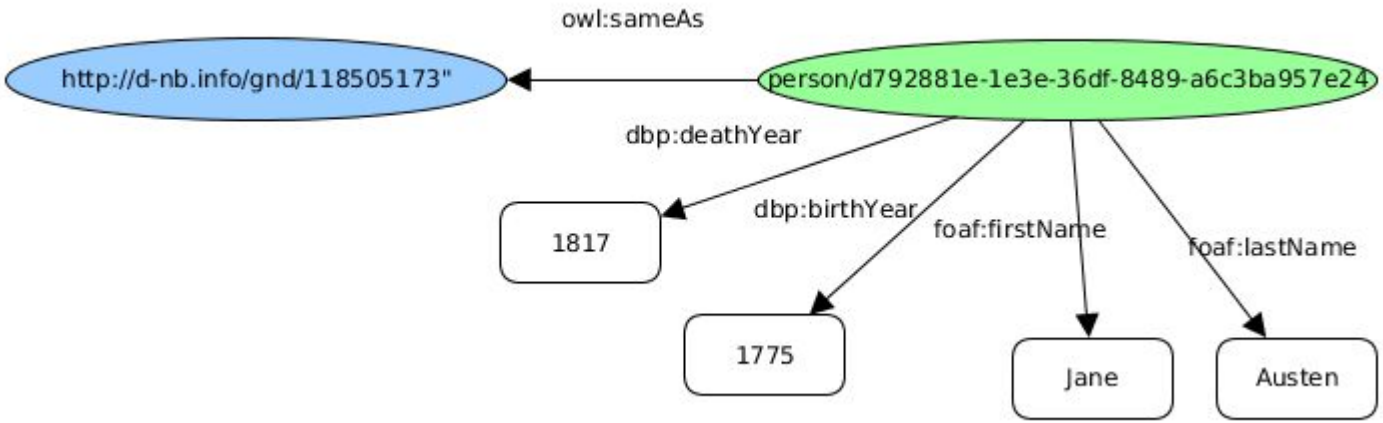
MARC zu RDF: Personen und Organisationen

- Personen wenn möglich als identisch erkennen
- Gleichen Identifier (URI) für eine Person vergeben

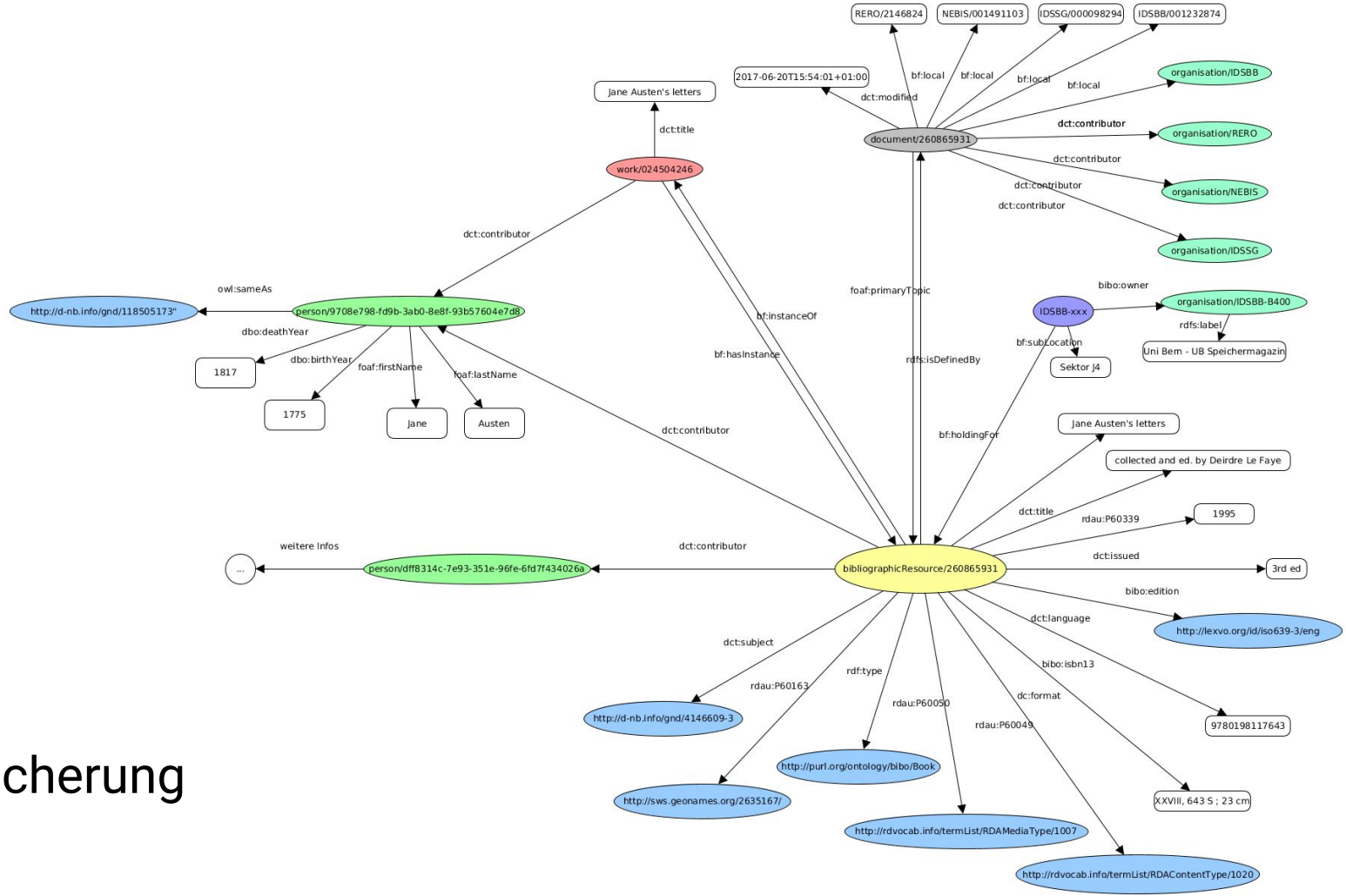
Identifier

- GND oder RERO ID
- Name, Titel der Person und Lebensdaten
- Name, Titel der Person und Titel der Publikation
- Organisation: Name, Abteilung, Datum, Ort

Person

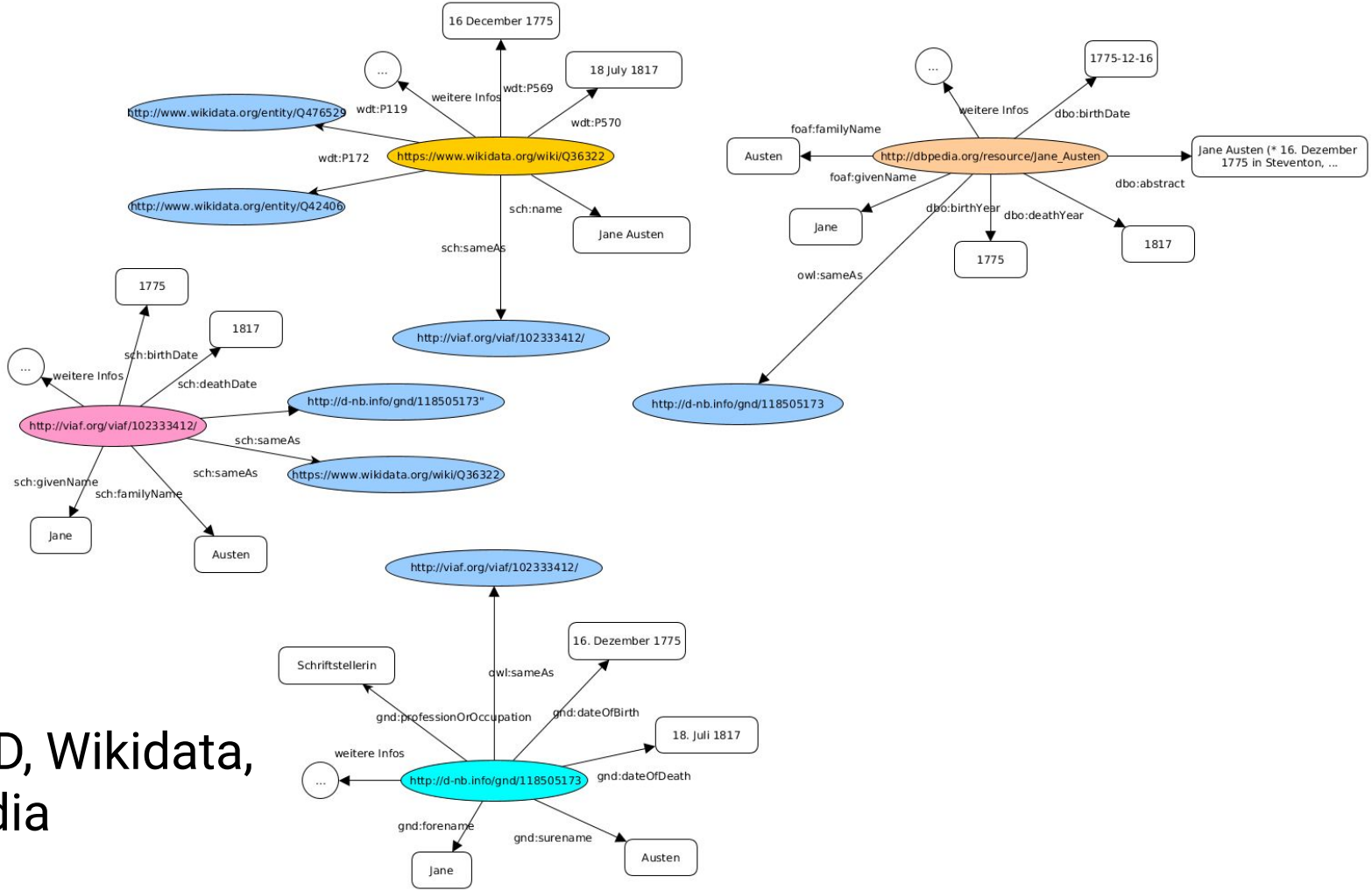

owl:sameAs
http://d-nb.info/gnd/118505173"
person/d792881e-1e3e-36df-8489-a6c3ba957e24
dbp:deathYear
dbp:birthYear
foaf:firstName
foaf:lastName
1817
1775
Jane
Austen

Resultat
vor Anreicherung

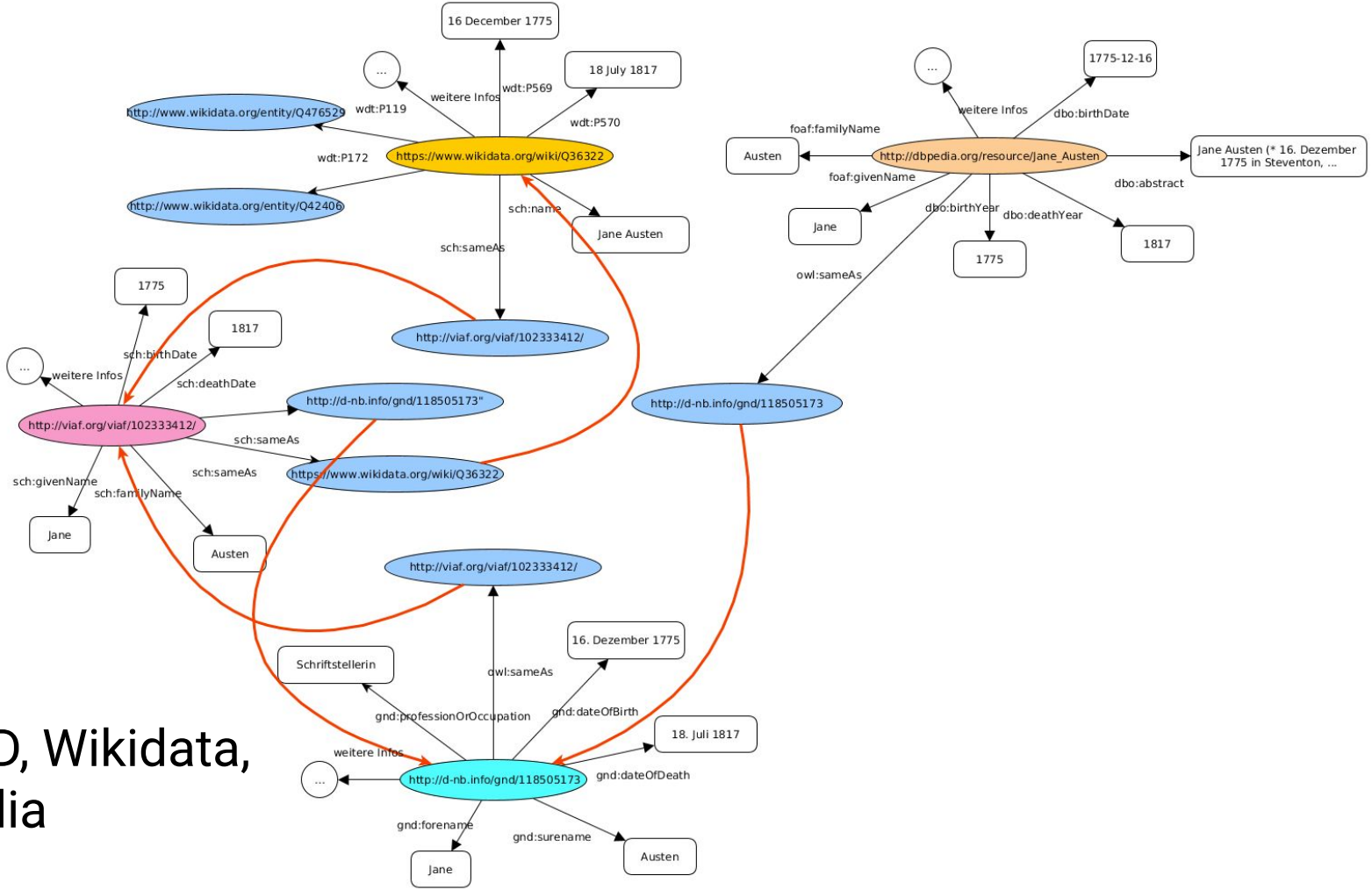


Verlinkung und Anreicherung

- Personen und Organisationen werden verlinkt
- Verlinkung mit GND, Wikidata, VIAF und DBpedia
- Matching über GND- oder RERO-ID
- swissbib-Person/Organisation wird angereichert mit den Informationen aus GND, Wikidata, VIAF und DBpedia
- Ab 2020 auf neuer Basis, mit GND und Wikidata als zusätzlichen Quellen



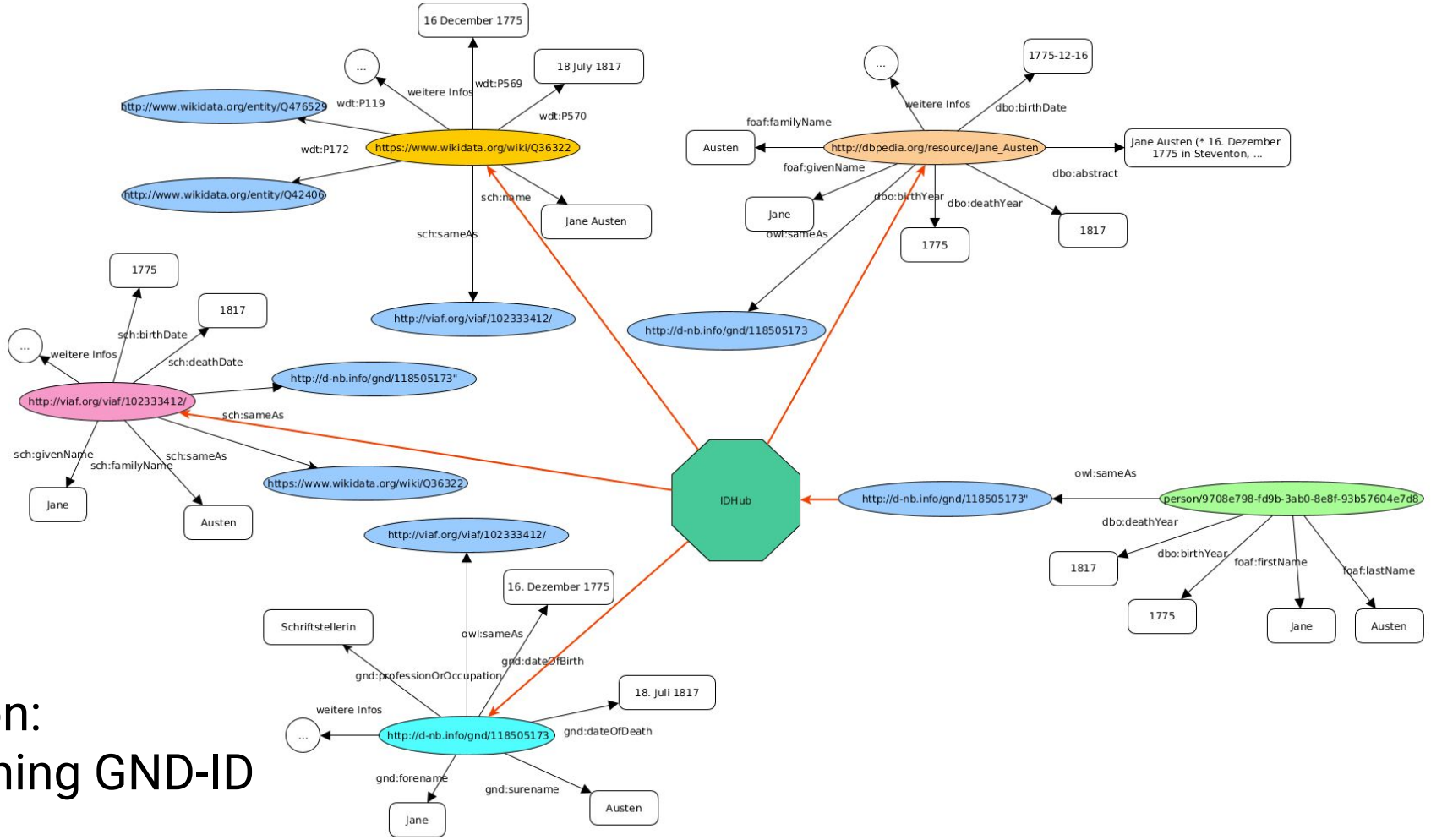
Person: GND, Wikidata, VIAF, DBpedia



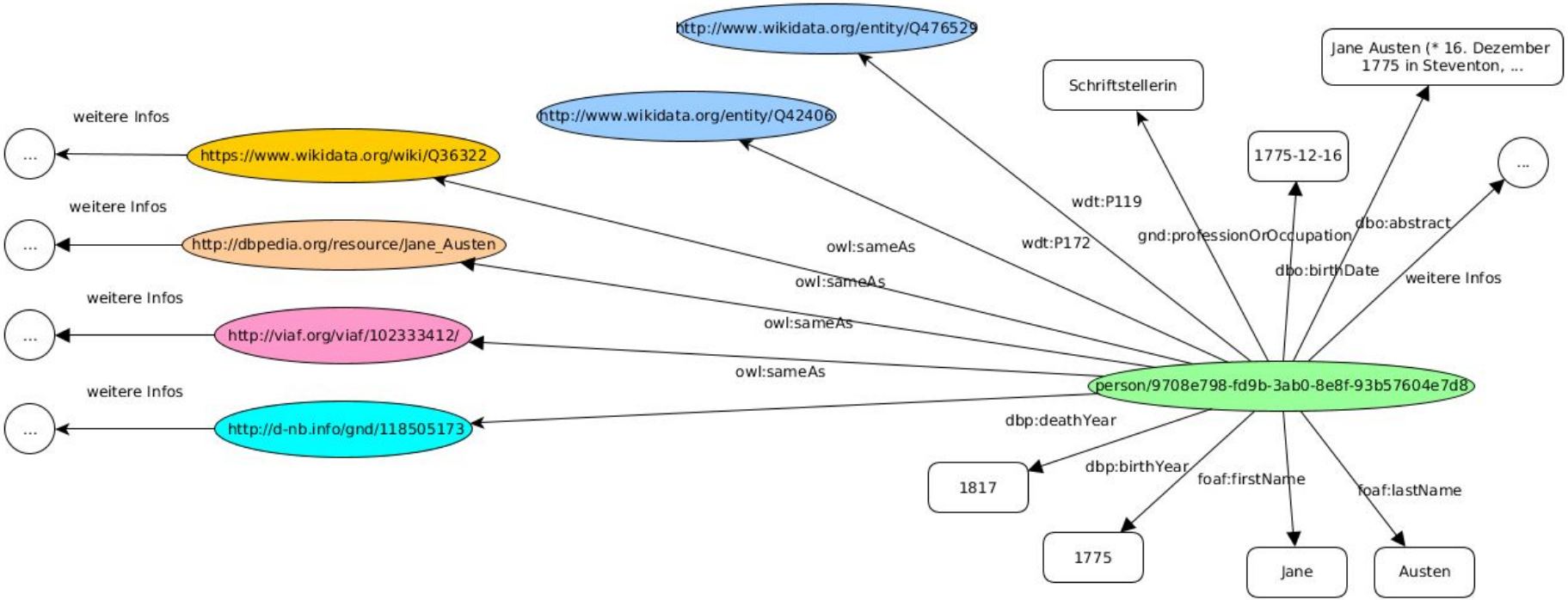
Person: GND, Wikidata, VIAF, DBpedia

IDHub https://d-nb.info/gnd/1016813244	
identifiers.GND	118505173
identifiers.VIAF	102333412
identifiers.DBPEDIA	Jane_Austen
identifiers.WIKIDATA	Q36322

Person: IDHub mit Identifier aus allen Quellen

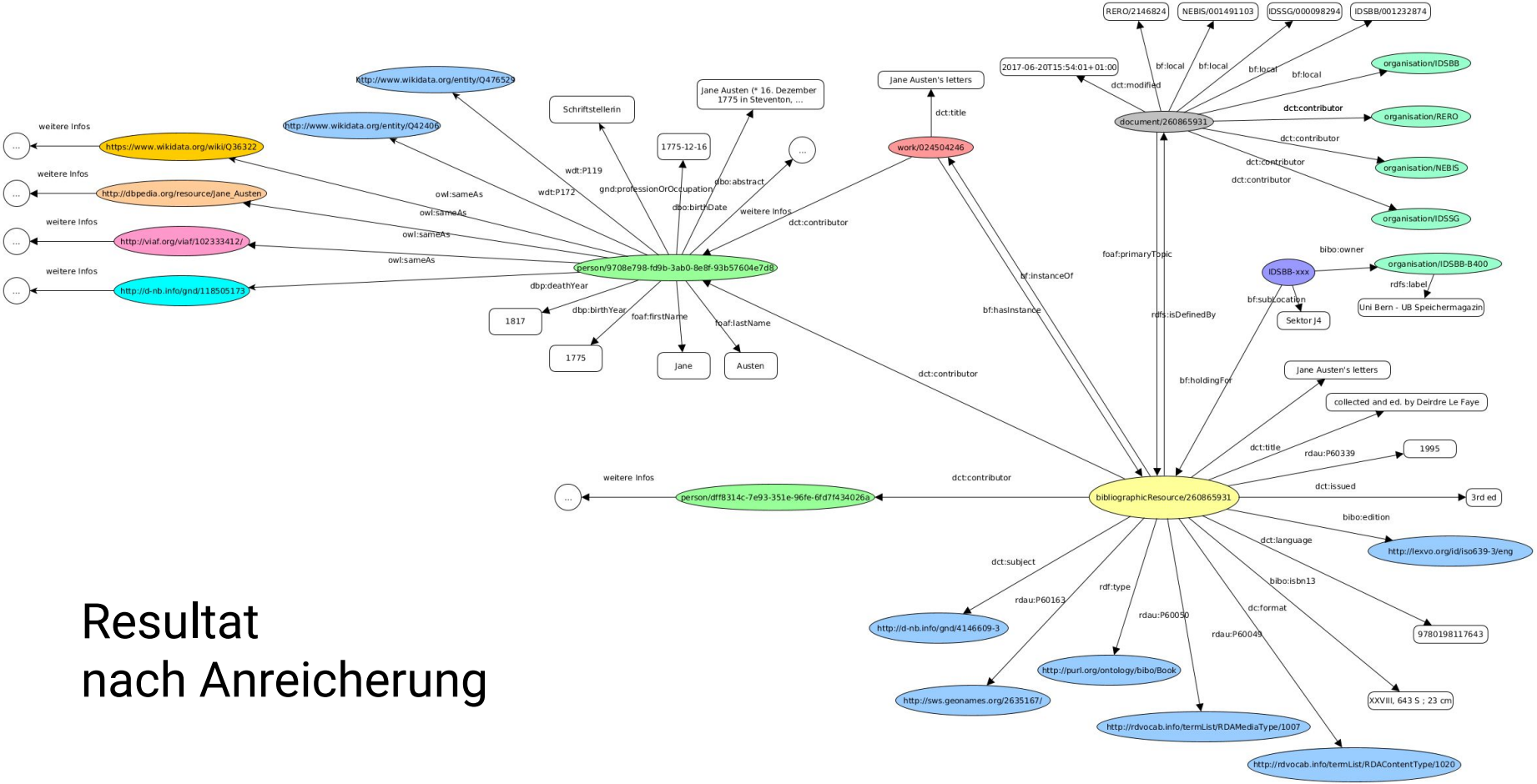


Person: Matching GND-ID



Person: Anreicherung

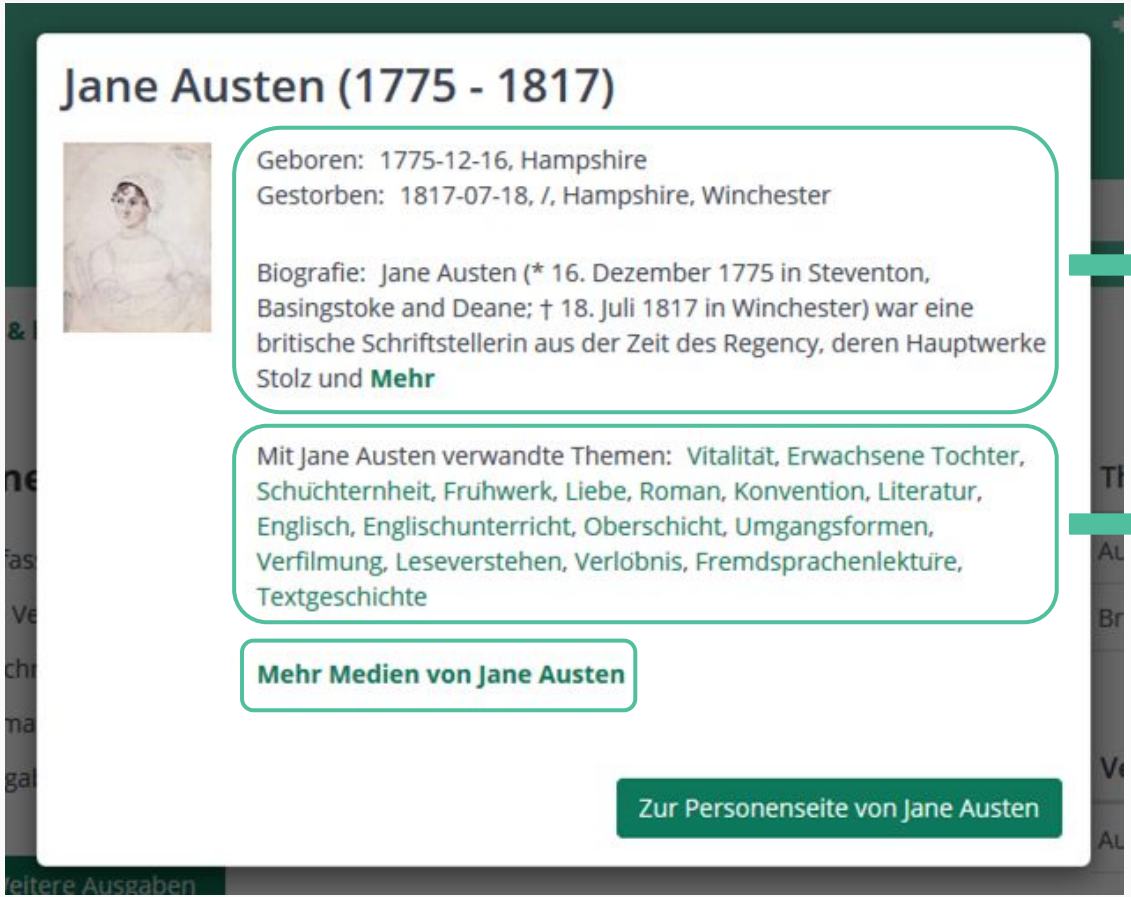
https://linked-swissbib.github.io/datamodel/person.html



Resultat
nach Anreicherung

https://data.swissbib.ch/person/8ef1ded7-8a3c-3b26-9a1c-344102ba6365

Knowledge Card für Personen

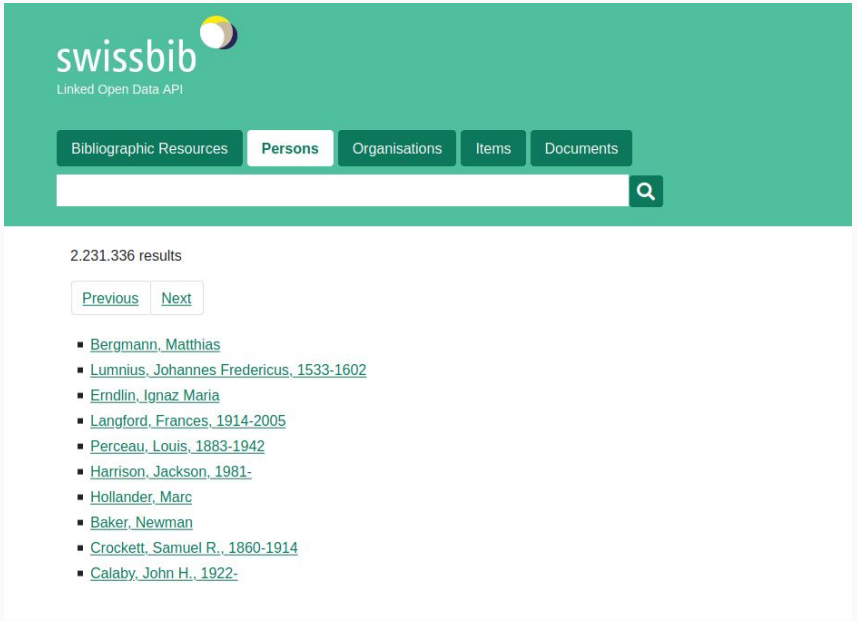


Informationen aus GND, Wikidata, VIAF, DBpedia

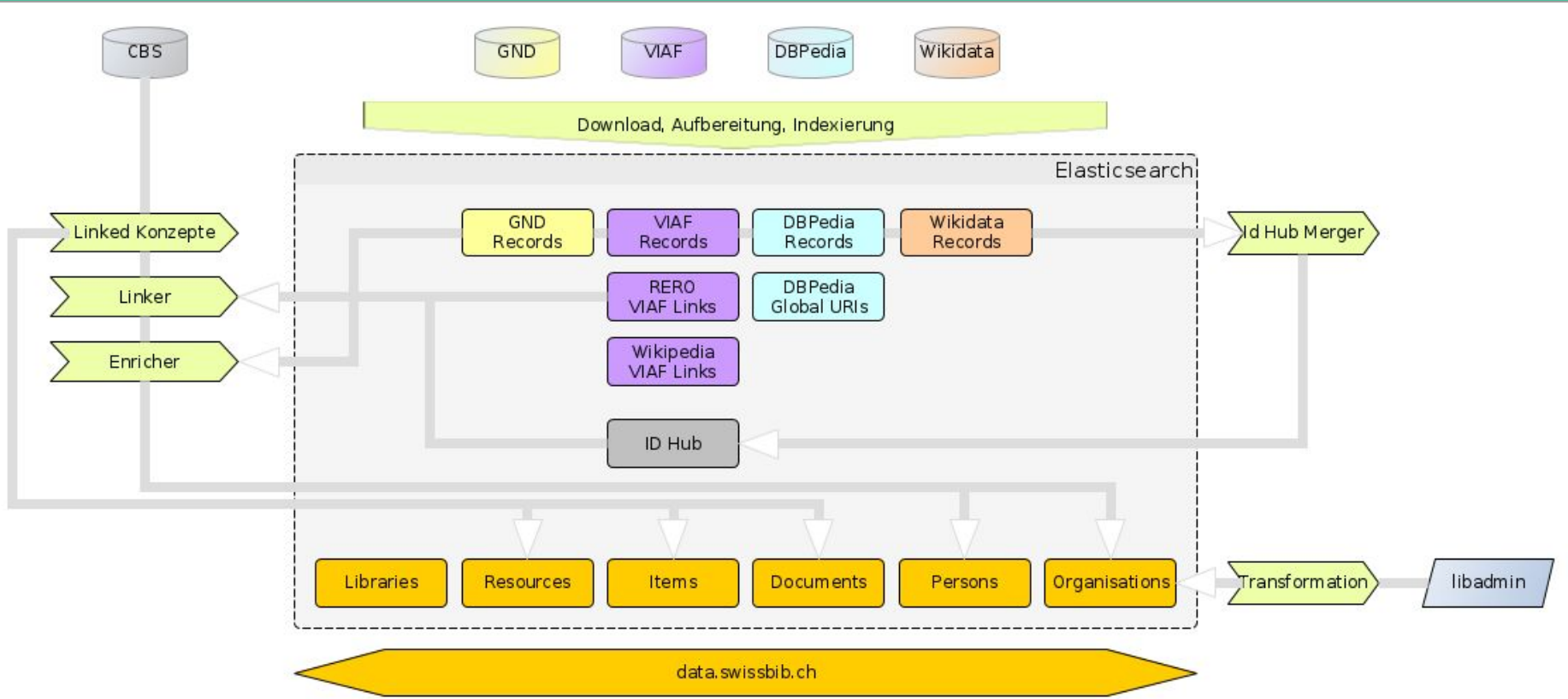
Themen (Feld 650 GND), die in Aufnahmen erfasst sind, wo die Person in 100/700 vorkommt

data.swissbib.ch

https://test.data.swissbib.ch/wikidata/P166/Q37922



linked: Workflows

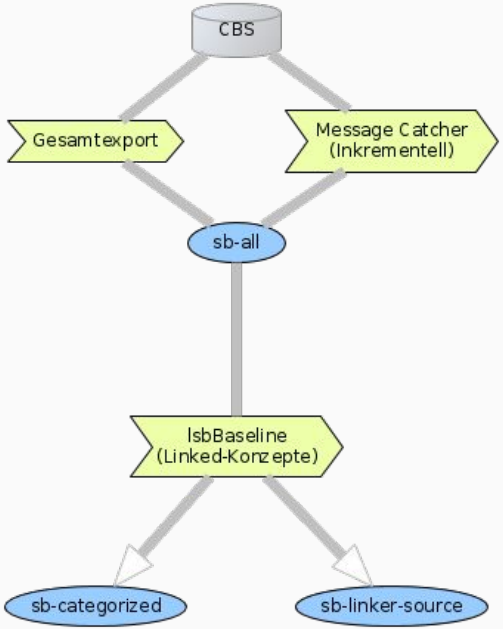


Erstellung LOD-Konzepte

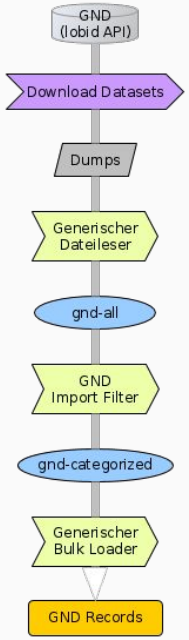
Ziel: Erstellung der LOD-Konzepte

Input: Zusammengeführte bibliografische Daten im MARC-XML-Format

Output: Konzepte im JSON-LD-Format



Import: GND



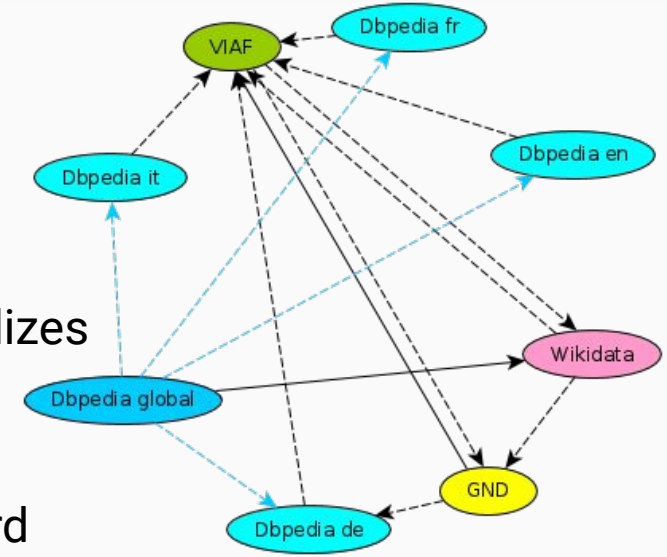
Ziel: Indexierung relevanter GND-Datensätze

Input: Dynamisch generierte GND-Dumps im JSON-LD-Format

Output: Elasticsearch-Index

ID Hub

Ziel: Cluster (Dokument) mit allen IDs identischer Ressourcen in Autoritätsdateien zwecks effizienterer Anreicherung von swissbib-Daten



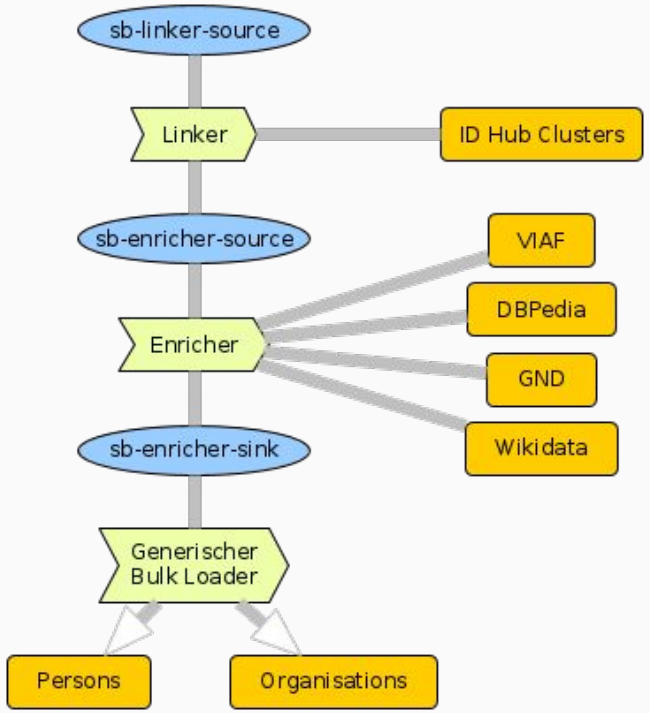
1. Auslesen *aller* owl:sameAs-Beziehungen aus Indizes
2. Normalisierung URIs
3. Lookup für jede URI in Dokument in ID Hub:
    - ID vorhanden => Vorliegender Datensatz wird mit ID Hub Cluster gemergt
    - Keine ID im Cluster vorhanden => Neuer Cluster im ID Hub wird erstellt

Verlinkung + Anreicherung

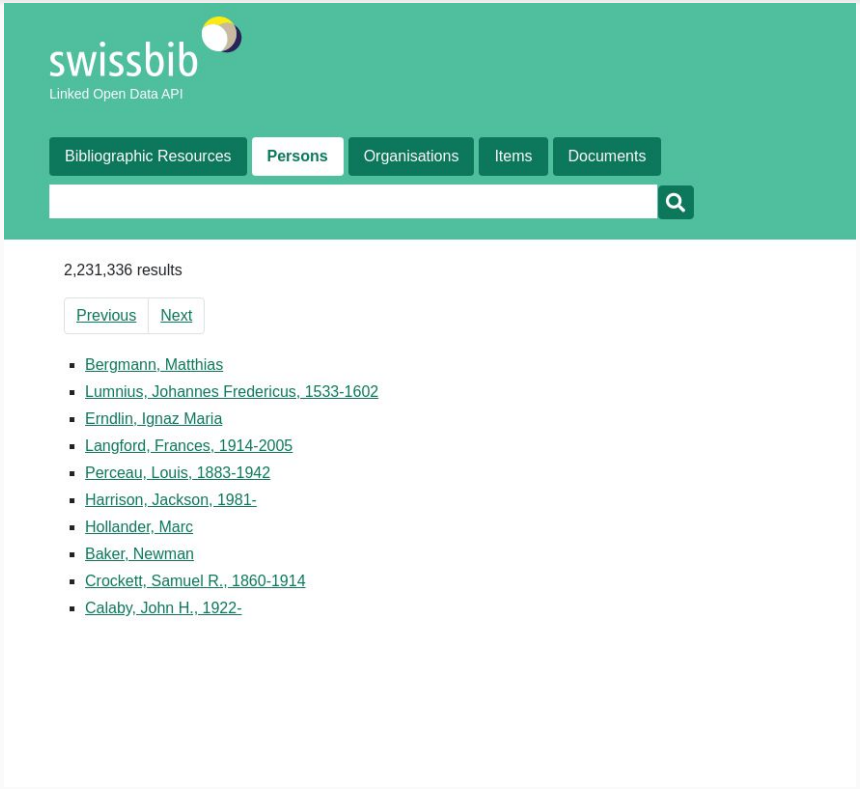
Ziel: Verlinkung und Anreicherung der swissbib-Daten

Input:
Personen-/Organisationen-Konzepte; ID-Clusters, Normdaten u.a.

Output: Angereicherte Personen-/Organisationen-Konzepte



data.swissbib.ch

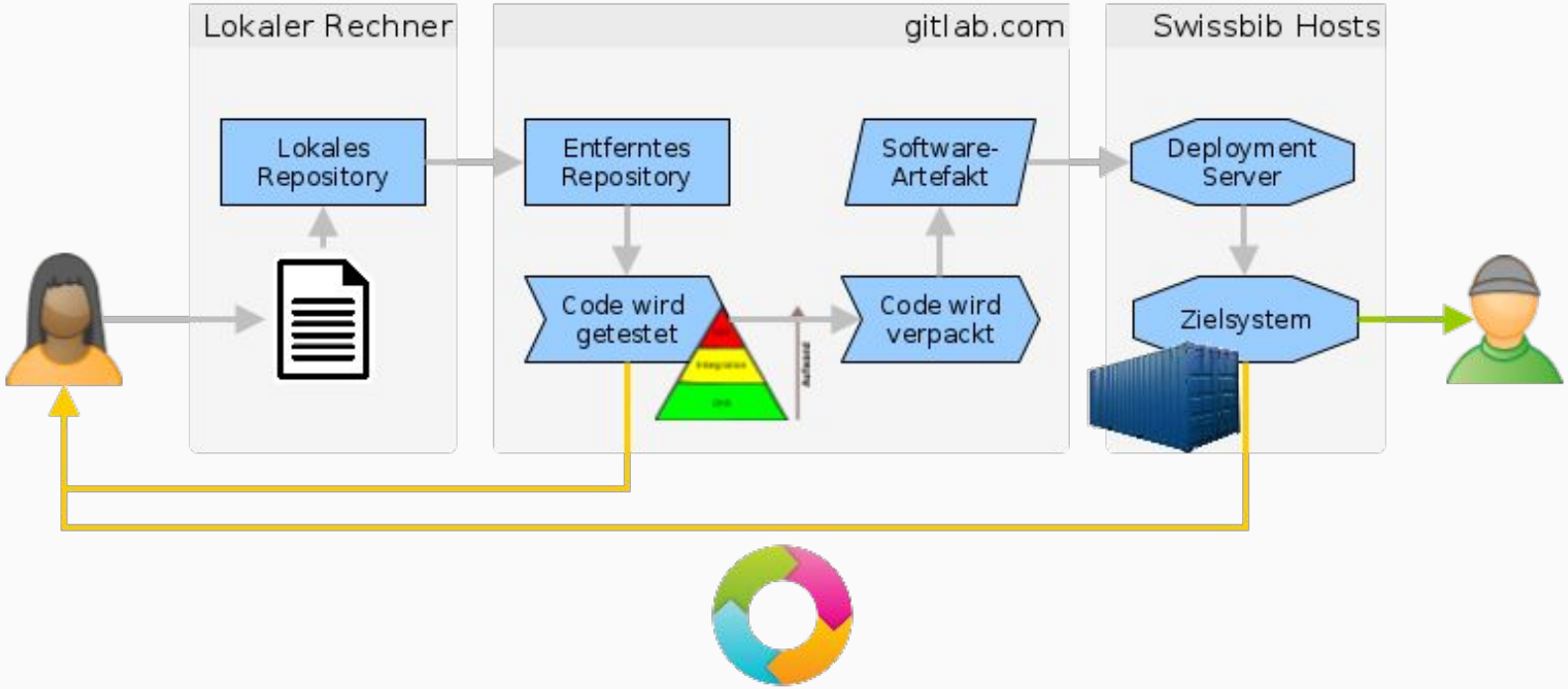


https://data.swissbib.ch

- Frontend
- API für maschinellen Zugriff auf Ressourcen
- “SPARQL light”
- Unterstützt Hydra-Vokabular (WIP)

Entwicklung, Deployment, Betrieb

Entwicklungsprozess (idealisiert)



Prinzipien agiler Entwicklung

Agil oder Wasserfall?

- ★ **Individuals and interactions** over processes and tools
- ★ **Working software** over comprehensive documentation
- ★ **Customer collaboration** over contract negotiation
- ★ **Responding to change** over following a plan

(Auszug aus dem Agile Manifesto)

Methoden und Merkmale agiler Entwicklung

- Selbstorganisierte Teams
  - Enge Zusammenarbeit von Fachexperten und Entwicklerinnen
  - Gemeinsame Erarbeitung von Architekturen, Anforderungen, Designs; keine Rollen
  - Paarprogrammierung
- Schnelle Entwicklungszyklen
- Häufige Codereviews unter Einbeziehung der Kunden
- Tests als Dokumentation => Test-Driven Development

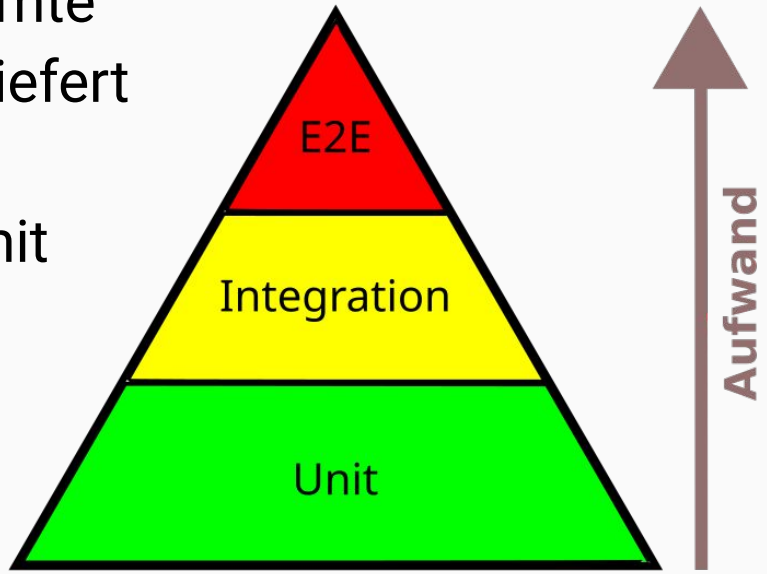
Weshalb Tests?

Prüfen, ob Anforderungen erfüllt werden unter anderem

- hinsichtlich ihrer Vollständigkeit
- hinsichtlich Korrektheit des Codes *unter bestimmten Bedingungen*
- hinsichtlich ihrer Lauffähigkeit in der Zielinfrastruktur
- hinsichtlich Sicherheit
- hinsichtlich der Laufzeit

Stufen von Softwaretests

- **Unit Tests**: Prüfen, ob eine bestimmte Codeeinheit korrekte Ergebnisse liefert
- **Integration Tests**: Prüfen, ob eine bestimmte Komponente korrekt mit anderen Komponenten interagiert
- **System / E2E Tests**: Prüfen die Funktionsweise des gesamten Systems



Containerisierung

Virtualisierung: Softwarebasierte Nachbildung (Abstraktion) von Geräten (Hardware) und/oder Diensten (bspw. Betriebssystemkern). Applikationen interagieren so nur indirekt (über die Abstraktionsschicht) mit dem System.

- *Virtualisierung* auf Betriebssystemebene
  - Prozesse laufen in abgekapselten Instanzen (“Containers”)
  - Mehrere Containers in einem System möglich
  - Leichtgewichtig, da geteilten Betriebssystemkern
- Vorteile:
  - Explizite Zuweisung von Hardwareressourcen
  - Isolierung

Docker

- Beispiel einer Containerlösung
- Abstrahiert vom zugrundeliegenden Betriebssystem
- Docker Engine:
  - Läuft als Hintergrundprozess (`dockerd`)
  - Stellt API zur Verfügung: `docker ...`
  - Stellt Kommunikation zwischen Containers sicher

Genese eines Containers

Dockerfile → Image → Container

- Dockerfile: Bauanleitung für Image
- Image: Schablone für Container
- Container: Laufende Instanz

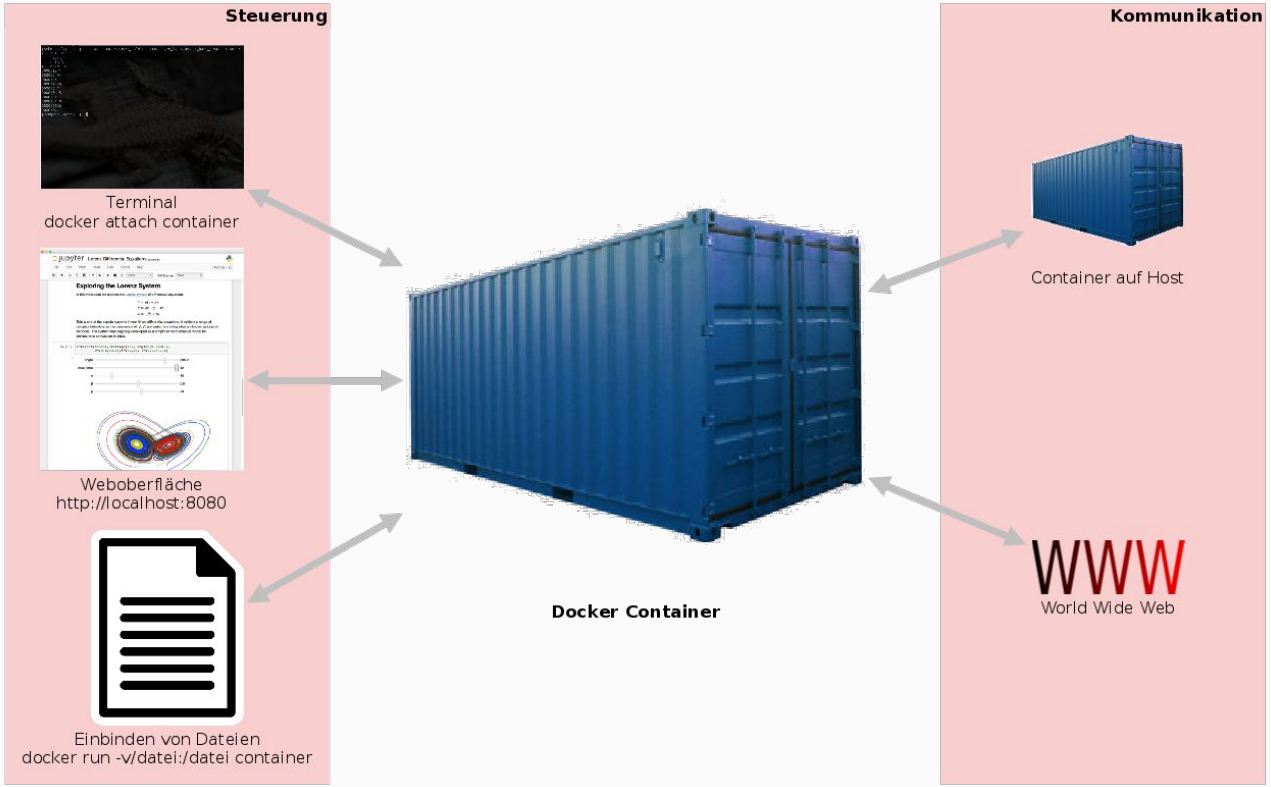
```
FROM ubuntu:18.04

COPY . /app

RUN make /app

CMD python /app/app.py
```

Interaktion mit Container



Docker: Beispiel

1. Voraussetzung: Docker Engine ist installiert und läuft
2. In Terminal: `docker run hello-world`

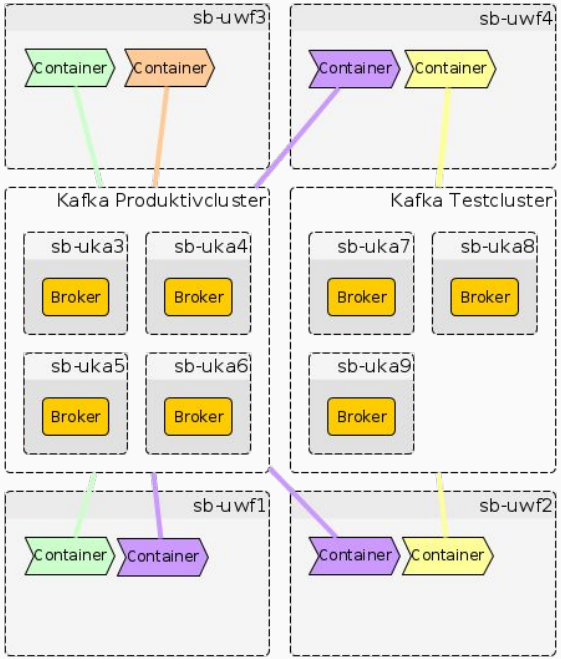
Was im Hintergrund passiert:

- Docker Engine kontaktiert Standard-Registry
- Image mit Namen `hello-world` wird heruntergeladen
- Container wird aus Image erstellt und gestartet
- Ausgabe erfolgt via Terminal

Orchestrierung: Docker Swarm

- Verbund von Docker Engines auf verschiedenen Hosts
- Kommunikation über Netzwerk (Overlay)
- Orchestriert werden *Services* (Gruppen von Containers)
- Vorteile (u.a.):
  - Ausfälle von Hosts können ausgeglichen werden
  - Lastverteilung (bspw. Webserver)

Orchestrierung in Swissbib



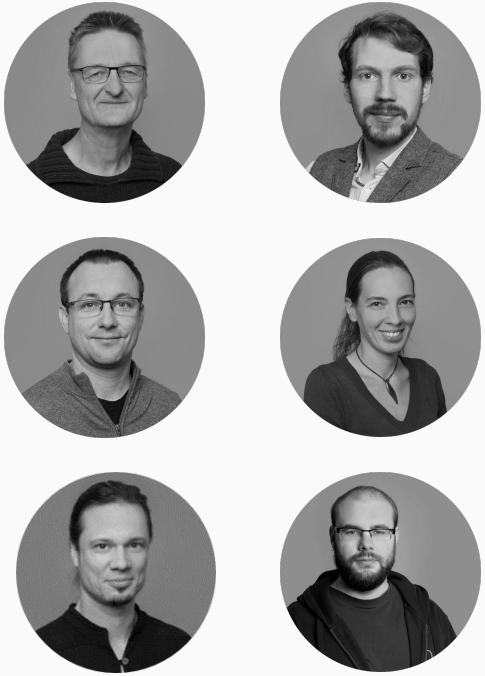
- Zwei Kafka-Clusters für produktiven und Testbetrieb
- Docker Swarm auf vier Hosts
- Deployment von Docker Images via Docker Hub oder Gitlab Registry
- Zukünftig: Webhook-Lösung

Notwendige Bedingungen zur aktiven Gestaltung digitaler Transformation?

- Diskussion über Organisationsformen / Anforderungsprofile / Zusammenarbeitsmodelle -

swissbib Team

- Personen mit IT- und Bibliothekshintergrund
- Selbstorganisiertes Team
- Agile Entwicklung
- Integriert in IT-Abteilung UB Basel
- Zusammenarbeit mit FHs, Firmen, ...



Diskussion

- Welchen Platz hat IT im Berufsbild der zukünftigen Bibliothekarin oder Informationswissenschaftlers?
- Was braucht es damit sich interessierte Personen in diese Richtung entwickeln können?
- Wie sehen Sie das für sich persönlich?

Datenanalyse

Monitoring

Programmlogs
geben Auskunft über

- Fehler
- Warnungen
- relevante Ereignisse, z.B.
  - Programmzustandsänderung
  - Logins
  - Seitenzugriffe

System- und Programmmetriken
geben Auskunft über

- Systemzustand, z.B.
  - Up- / Downtime
  - Latenz (Verzögerung)
  - Throughput (Durchsatzrate)
- Ressourcenverbrauch, z.B.
  - RAM
  - CPU
  - Festplatten

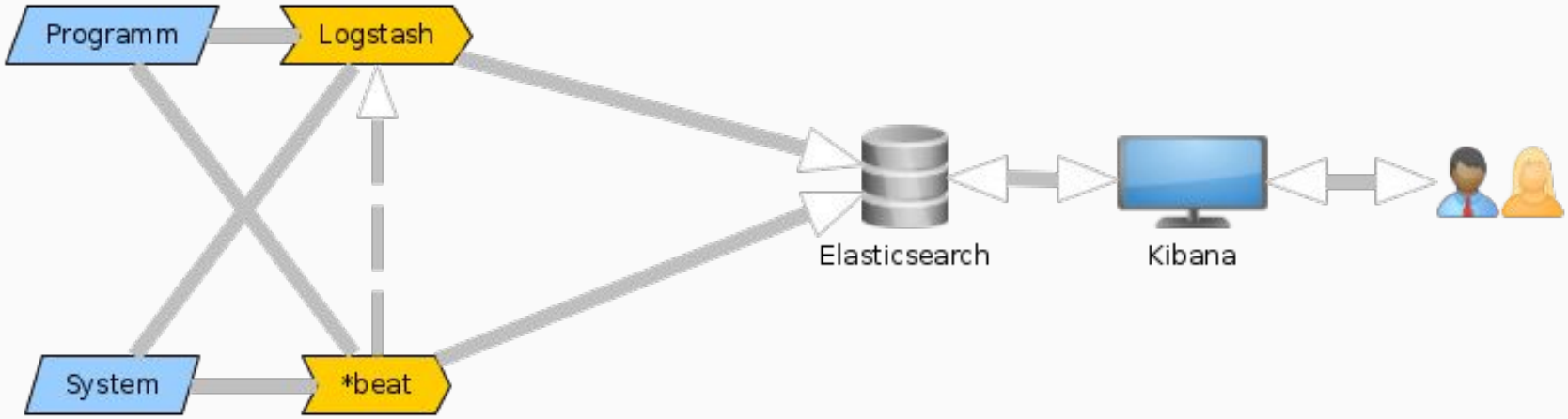
Weshalb Monitoring?

- Defekte Hardware
- Unerwarteter Abbruch eines Programms
- (Zu) hoher Ressourcenverbrauch
- Zombieprozesse
- Sicherheitsrelevante Ereignisse
- Nutzeranalyse (v.a. Webanalytik)
- unbedingt erforderlich, um SLAs (Service Level Agreements) einhalten zu können

ELK-Stack

- Analyseplattform
- Visualisierung aufbereiteter und gespeicherter Logs/Metriken
- Komponenten:
  - **E**lasticsearch: Datenpersistierung
  - **L**ogstash: Datenaufbereitung
  - **K**ibana: Datenvisualisierung
  - Beats: Datenaufbereitung (anwendungsfallspezifisch)

ELK: Workflow



Kibana - Einsatz bei swissbib

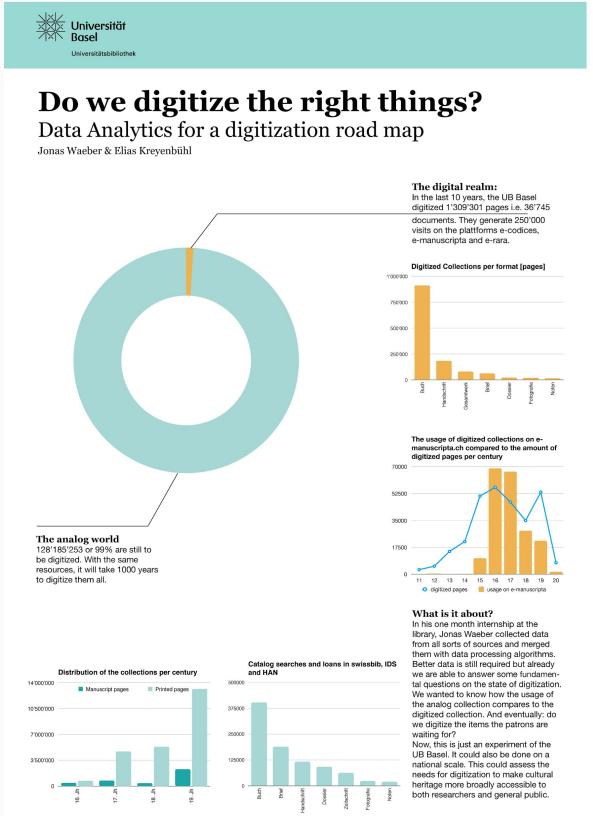
- Monitoring
- Nutzungsstatistik Oberflächen
- Analyse Metadaten
- Zusammenführung Nutzungs- und Metadaten (Prototyp)

Beispiel: linked.swissbib

- Einblick in Daten
- Aufbau Datenmodell: Mapping der Felder aus den unterschiedlichen Quellen
- Anzeige auf swissbib.ch: Welche Felder anzeigen?

Beispiel: Digitalisierung UB Basel

- Wie viele Seiten aus dem digitalisierbaren Bestand der UB Basel sind digitalisiert?
- Werden die Bestände digitalisiert, die auf e-rara, e-manuscripta genutzt werden?
- Wie verteilt sich der Bestand auf Format, Publikationsjahr, etc.?



Universität Basel
Universitätsbibliothek

Do we digitize the right things?
Data Analytics for a digitization road map
Jonas Waeber & Elias Kreyenbühl

The digital realm:
In the last 10 years, the UB Basel digitized 1'309'301 pages i.e. 36'745 documents. They generate 250'000 visits on the plattforms e-codices, e-manuscripta and e-rara.

Digitized Collections per format [pages]

The analog world
128'185'253 or 99% are still to be digitized. With the same resources, it will take 1000 years to digitize them all.

The usage of digitized collections on e-manuscripta.ch compared to the amount of digitized pages per century

Distribution of the collections per century

Catalog searches and loans in swissbib, IDS and HAN

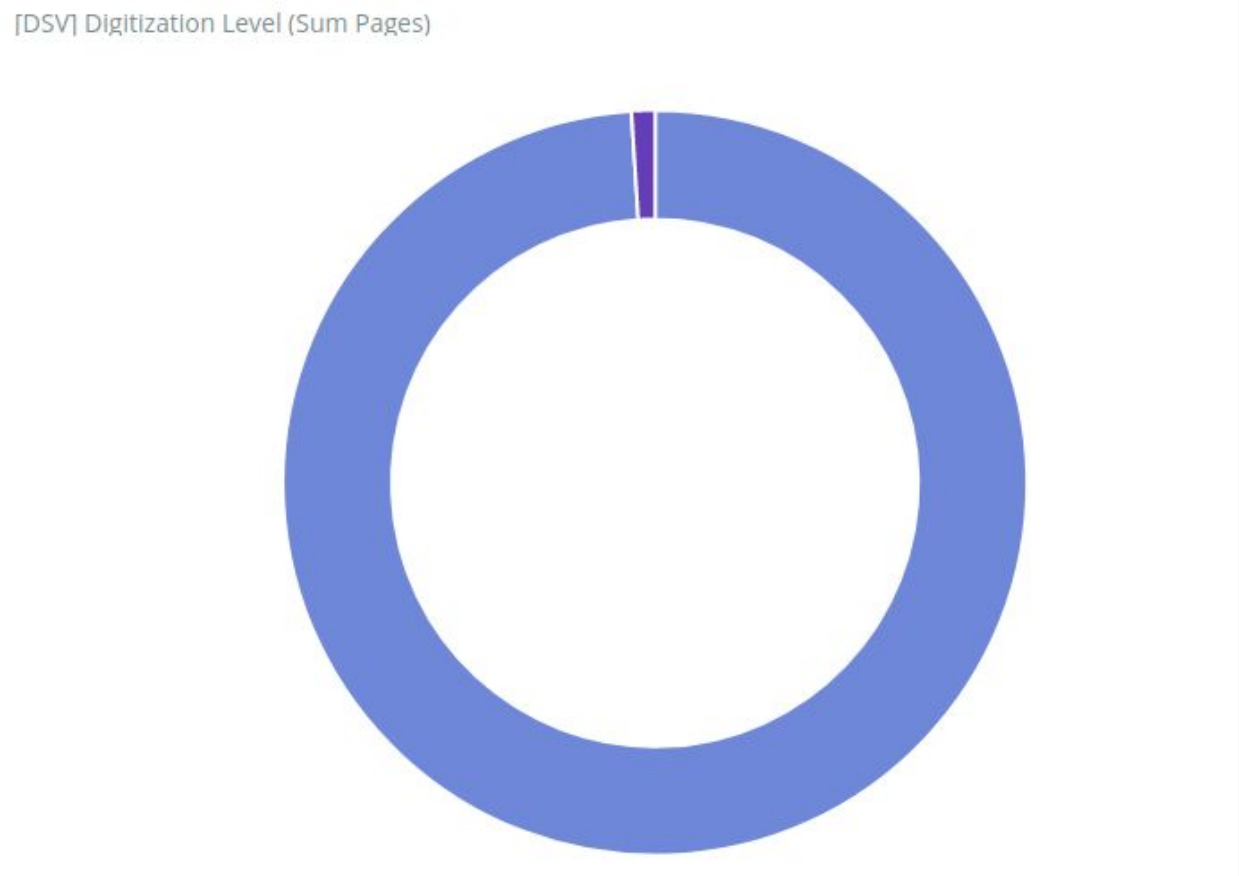
What is it about?
In his one month internship at the library, Jonas Waeber collected data from all sorts of sources and merged them with data processing algorithms. Better data is still required but already we are able to answer some fundamental questions on the state of digitization. We wanted to know how the usage of the analog collection compares to the digitized collection. And eventually: do we digitize the items the patrons are waiting for?
Now, this is just an experiment of the UB Basel. It could also be done on a national scale. This could assess the needs for digitization to make cultural heritage more broadly accessible to both researchers and general public.

Beispiel: Digitalisierung UB Basel

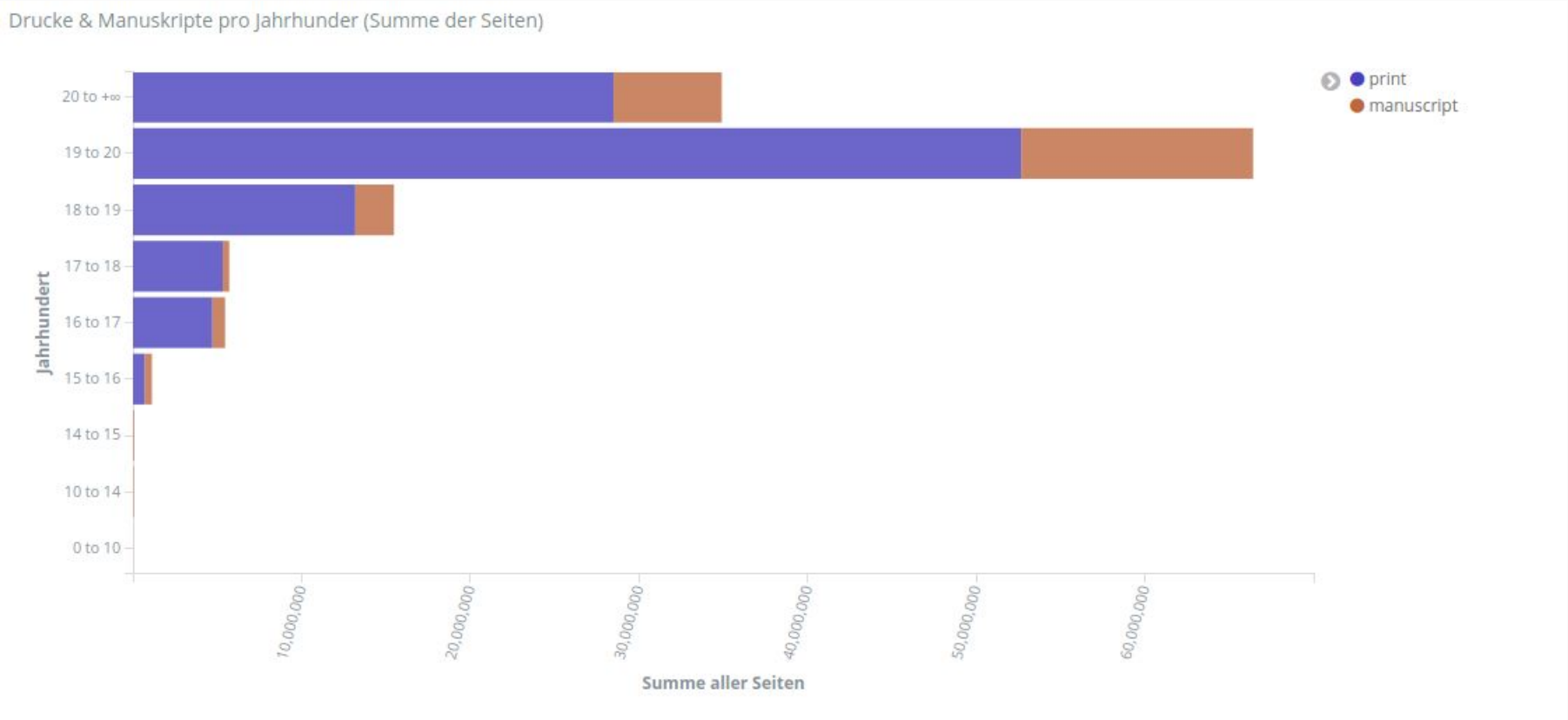
Daten:

- Bibliographische Daten (inkl. Aufbereitung für Seitenzahlen)
- Nutzungsdaten mehrerer Oberflächen (e-rara, e-manuscripta, e-codices, swissbib, HAN OPAC)
- Zusammengeführt über Systemnummer der Aufnahme

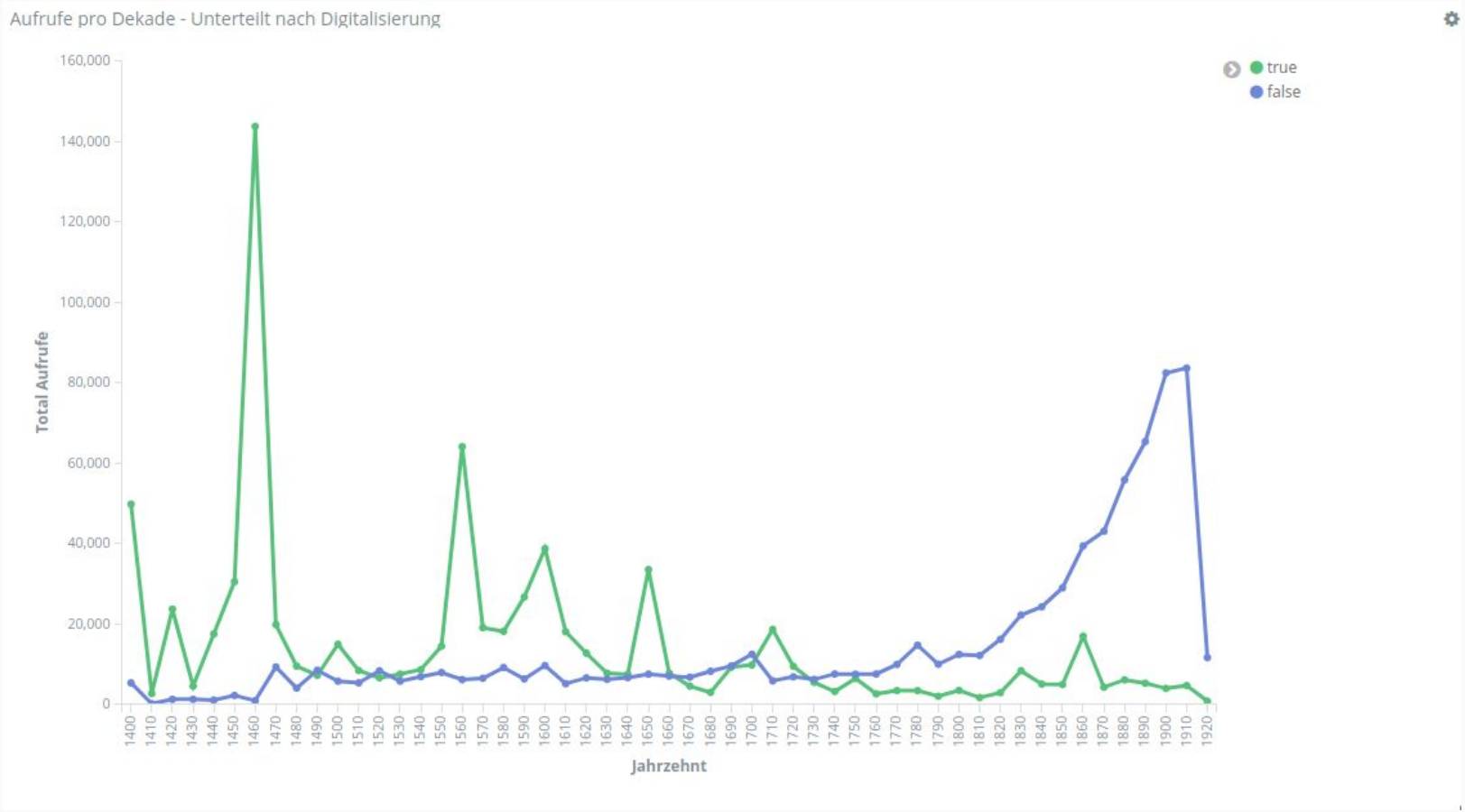
Anteil digitalisierter Seiten



Drucke & Manuskripte pro Jahrhundert (Summe der Seiten)

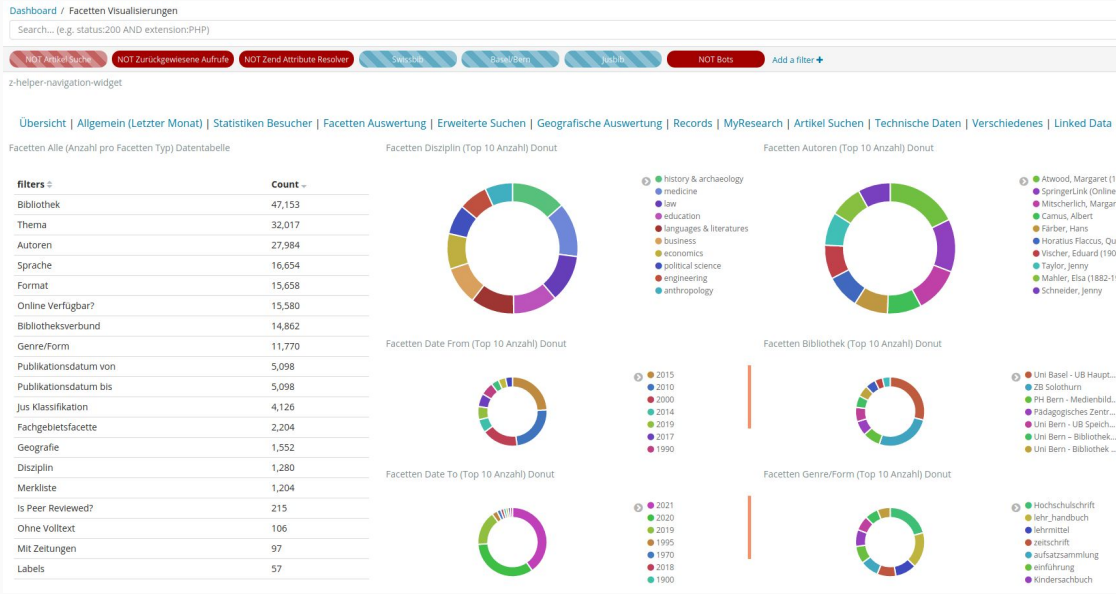


Aufrufe pro Dekade - Unterteilt nach Digitalisierung



Beispiel: Nutzungsdaten swissbib

- Access-logs
- Anreicherung der Daten
- Nutzungsstatistik
- Detaillierte Auswertung bestimmter Elemente



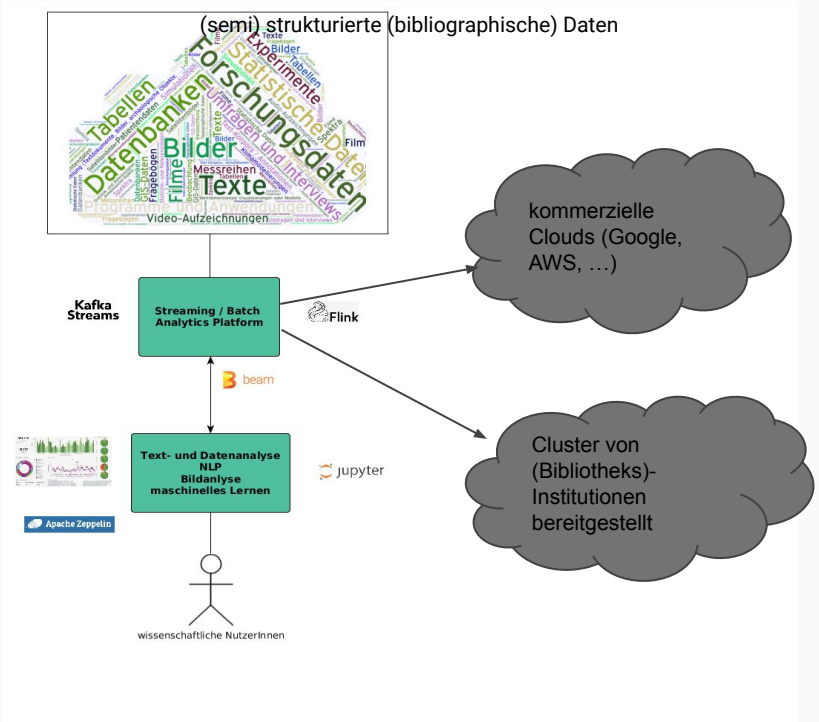
lab.swissbib.ch (neue Form der “data-based services” ?)

Grundidee:

- Nutzerinnen erhalten die Möglichkeit, über niederschwellige webbasierte Schnittstellen direkt mit ihrem dataset zu arbeiten.
- “Arbeiten” kann heissen:
  Daten transformieren, anreichern, analysieren und verknüpfen
  maschinelles Lernen, ...
- Es sollen heute verbreitete Programmier- und Analysesprachen verwendet werden können (z.B. Python)
- Es sollen nicht nur kleine sondern auch grosse datasets verwendet werden können.

lab.swissbib.ch (Fortsetzung)

- Plattform für endnutzerorientierte Analyseservices, bspw.
  - Jupyter, Apache Zeppelin
  - Kibana
  - OpenRefine
- Institutionen (Bibliothek) unterstützen Nutzerinnen bei ihren Aufgaben/Ideen. Dafür braucht es Kenntnisse im Umgang mit den eingesetzten Werkzeugen.
- Für den Umgang mit grossen Daten-mengen werden “Big Data”-Frameworks benötigt (OpenSource oder kommerziell)



Jupyter lab (https://jupyter.org/)

- interaktive Entwicklungsumgebung für sogenannte Web Notebooks
- kann mit verschiedenen Programmiersprachen verwendet werden (häufig Python)
- häufig für Bereiche eingesetzt, die mit den Schlagworten:
  → data science
  → scientific computing
  → machine learning
  umschrieben werden
- sollte im Teil-Modul “Daten Workshop” kurz vorgestellt worden sein

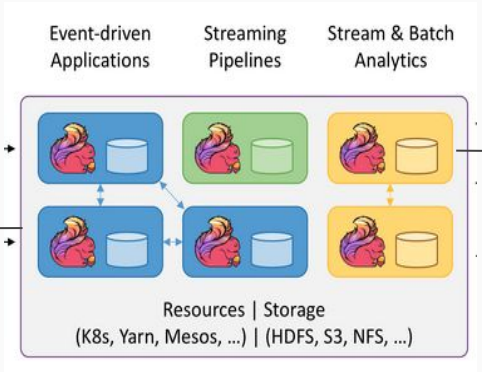
Apache Flink (https://flink.apache.org/)

- ermöglicht die verteilte Verarbeitung auch (sehr) großer Datenmengen in einem Rechnernetzwerk
- Verarbeitung sowohl eines permanenten Datenstroms (“Streaming”) als auch endlicher datasets (“bounded streams”)
- eines der aktivsten Apache Projekte, ursprünglich an Berliner Hochschulen entwickelt
- wird auch von sehr grossen Firmen eingesetzt
  (Alibaba, zalando, UBER, aws)
- von swissbib bereits für einen Teilaspekt des SLSP De-Duplizierungsprojekts produktiv eingesetzt (https://gitlab.com/swissbib/slsp/series-transformation/volumes-series-enrichment-flink)

Problemstellung:

Python

Java



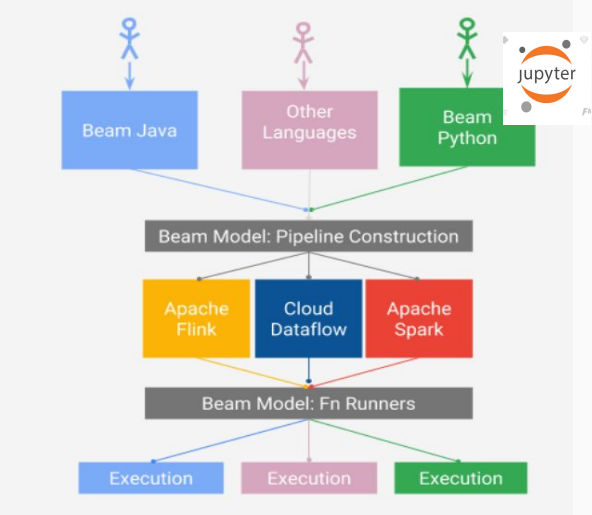
strukturierte Metadaten

Wie können Nutzerinnen ihre Problemstellung in einfachen Webfrontends (Jupyter) mit niederschwelligen Programmiersprachen wie Python beschreiben und gleichzeitig Frameworks (z.B. Apache Flink) zur verteilten Verarbeitung grosser Datenmengen nutzen?

Diese “BigData” Frameworks werden in der Regel mit der Programmiersprache Java benutzt.

eine mögliche Lösung: Apache Beam
(https://beam.apache.org/)

- wie Flink ein sehr aktives Apache Projekt
- ursprünglich von Google entwickelt



Prinzip:
die Beschreibung der Problemlösung für meine Datenanalyse in den Mitteln, die mir Beam bereitstellt, wird auf sogenannte “Runners” übertragen, die auf dem verteilten Framework ausgeführt werden.

https://opensource.com/article/18/5/apache-beam

Demo: Datenanalyse als Nutzerservice

Aufbau der Demo:

- File mit bibliographischen Beschreibungen aus swissbib (~120.000)
- Ziel: Extraktion der Autoren (Feld 100) , Anzahl der Häufigkeit eines jeden Autors
- Erstellen einer kleinen Datenpipeline mit Hilfe von Apache Beam und Python
- Ausführen der Pipeline auf einem lokalen Apache Flink cluster (könnte alternativ auch der cluster eines Cloud Anbieters sein)
- Perspektive: die Demo dient als Beispiel dafür, wie interaktive Datenanalysen aufgebaut sein können. Erweiterbar mit jeder Python library für verschiedenste Bereiche (z.B. maschinelles Lernen)
- Sourcecode: https://gitlab.com/swissbib/lab/services/jupyter-beam-flink

Datenanalyse - ein Service von Bibliotheken?

- Diskussion -

Diskussion

- Ist die Bereitstellung von Tools und Daten für (interaktive) Datenanalysen ein zukünftiger Service von Bibliotheken?
  - für interne Zwecke (MitarbeiterInnen der Institutionen)
  - durch NutzerInnen / für wissenschaftliche Zwecke ausschliesslich auf eigenen datasets oder Verknüpfungen mit (strukturierten) Daten aus dem GLAM Bereich
- Welche Anwendungsfälle dafür gibt es für Bibliotheken selbst? Welche für die Forschung?
- Welche Voraussetzungen braucht es, damit auch weniger IT-affine Personen Daten nutzen können?