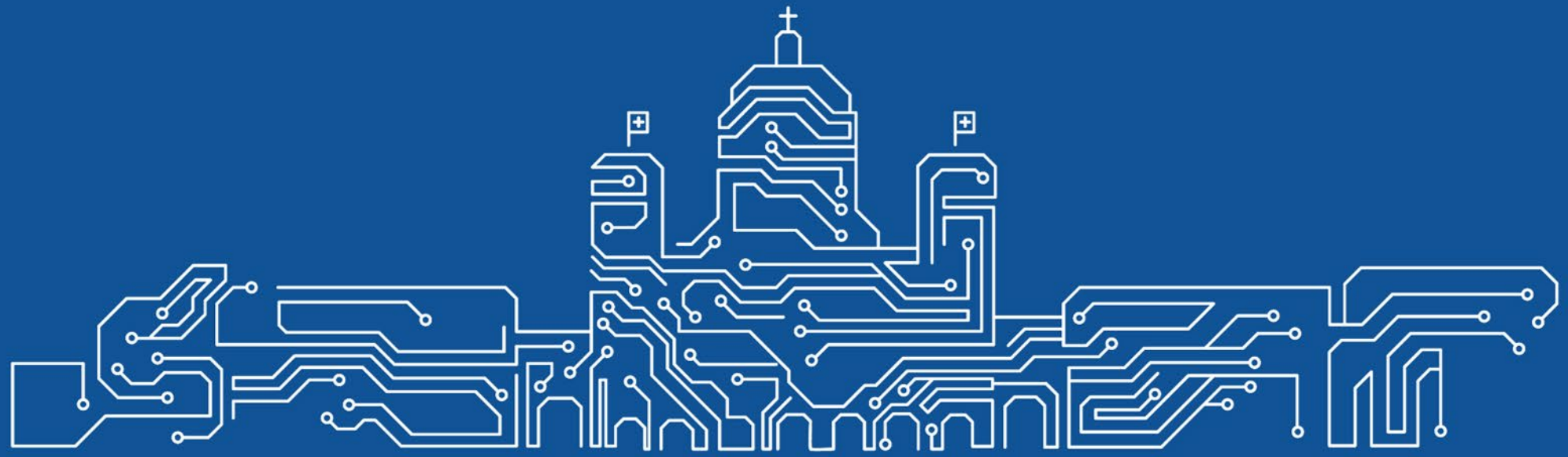
Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Bundeskanzlei BK
Digitale Transformation und IKT-Lenkung (DTI)

Sounding Board Prototyp Linked Data

im Rahmen des Projekts Datenökosystem der BK-DTI im Auftrag von DVS

Juni 2024



Sounding Board «Prototyp Linked Data»

- 15. Februar 2024
- 22. März 2024
- 25. April 2024
- **13. Juni 2024**

Agenda

- Rückblick
  - Résumé
- Datenaustausch
- Herausforderung / Fragestellung
  - Fragen und Schwierigkeiten
- Nächstes Sounding Board

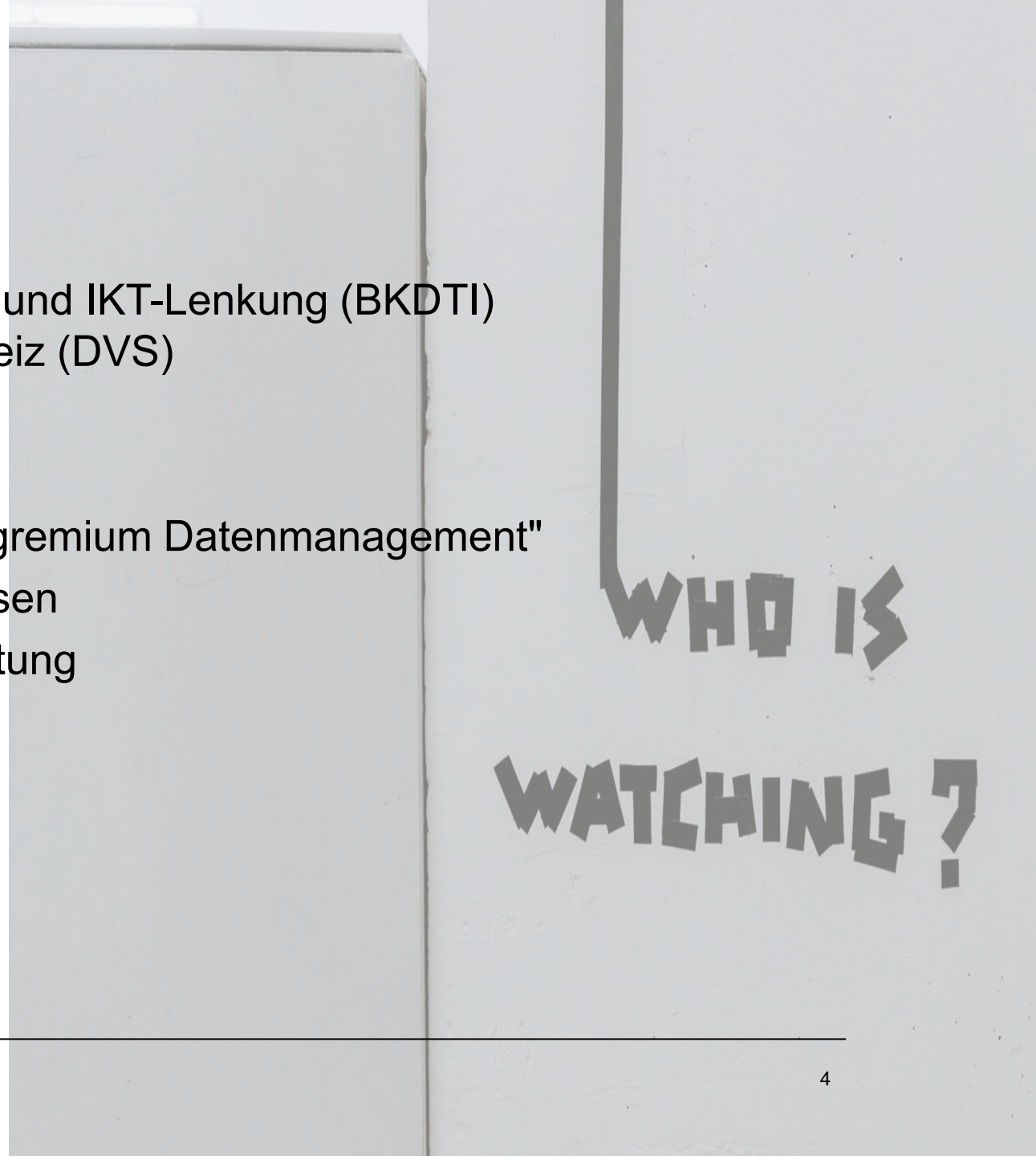
Begrüssung & Vorstellung

Wir:

- Bundeskanzlei, Digitale Transformation und IKT-Lenkung (BKDTI) im Auftrag von Digitale Verwaltung Schweiz (DVS)

Ihr:

- Vertreter aus "Nationalem Koordinationsgremium Datenmanagement"
- Vertreter aus eCH Fachgruppe Meldewesen
- Linked Data Interessierte aus der Verwaltung
- Von Anderen eingeladene Teilnehmer



Sounding Board: Ziele und Organisatorisches

Ziele:

- Teilen von unserer Erfahrung und dem gelernten
- Teilnehmer mit auf eine mögliche Reise künftiger Datenverwaltung nehmen.
- Fragen beantworten und Herausforderungen aufnehmen (nicht lösen)
- Sammelstelle von Feedbacks, um daraus Erkenntnisse zu gewinnen (jede Frage ist wichtig und ein möglicher Hinweis für Unbekanntes).

Organisatorisches:

- Unbeantwortete Fragen werden wir auf einem Parkplatz sammeln
- Das Sounding Board trifft keine Entscheidungen.

Hausregeln

- **Chatham House Rules**: Alles gesagte darf weiterverwendet werden. Wer sich geäussert hat, bleibt unter uns.
- Sprache: **Jeder** darf **in seiner Sprache** sprechen (Deutsch, Französisch und Englisch haben gute Chancen verstanden zu werden)
- Form: Grundsätzlich ungezwungen "**per Du**" - sollte sich jemand damit nicht wohl fühlen, darf er dies melden.
- **Fragen sind willkommen**, wenn wir sie nicht beantworten können, ist es ein Hinweis auf einen blinden Fleck – bitte stellt diese!
- **Andere Meinungen** sind Hinweise auf Spannungsfelder – bitte bringt diese ein!

Rückblick

Webseite zum Prototyp mit Folien der letzten Sounding Boards

- deutsch
- français
- italiano

Zwei GIT-Repositories wurden erstellt, worin die Arbeiten direkt geteilt werden.

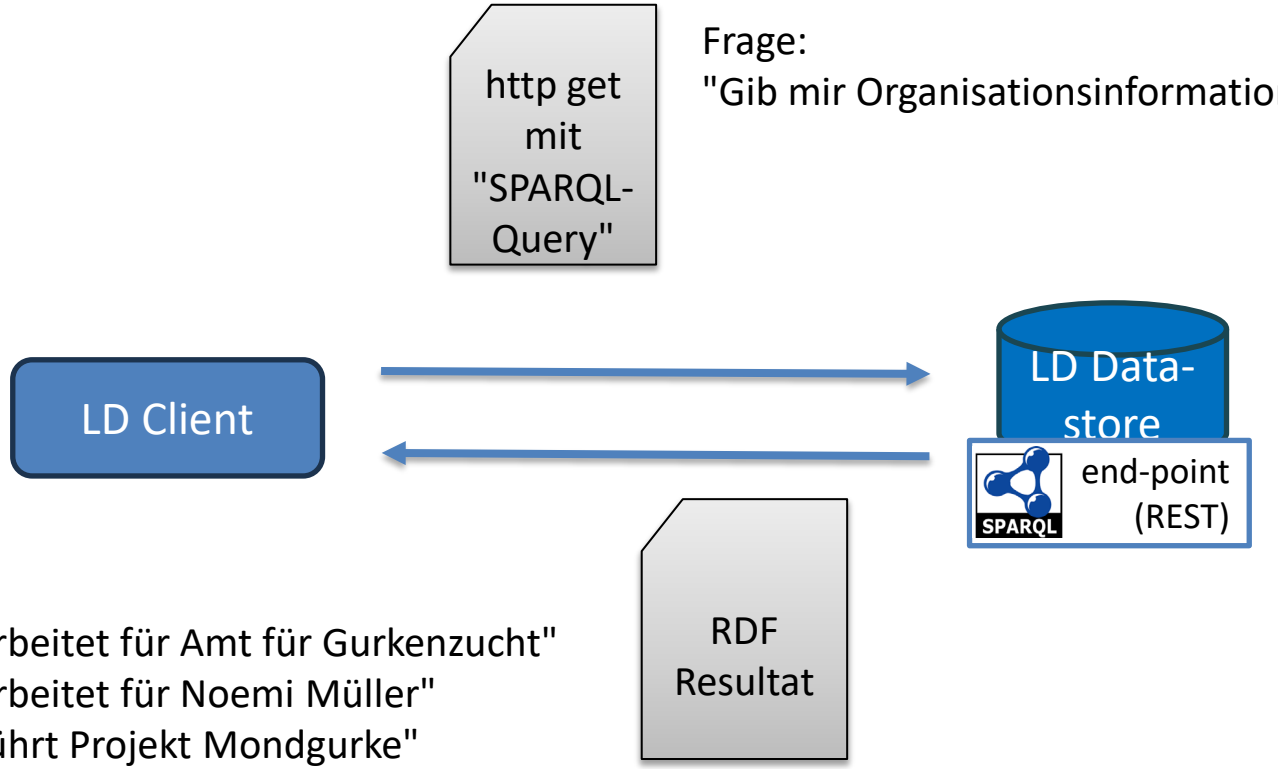
- Repository https://github.com/swiss/ld-prototype-data für die Daten und einfache Schnittstelle
- Repository https://github.com/swiss/ld-prototype-proxy für die Entwicklung des Security-Layer mit dem Proxy

Datenaustausch

Bearbeitete Fragestellungen

- Wie werden verteilte Informationen abgefragt
- Wie wird sichergestellt, dass nur die Information ankommt, wofür jemand berechtigt ist.

Einfache Abfrage



Frage:
"Gib mir Organisationsinformation zu Walter Muster"

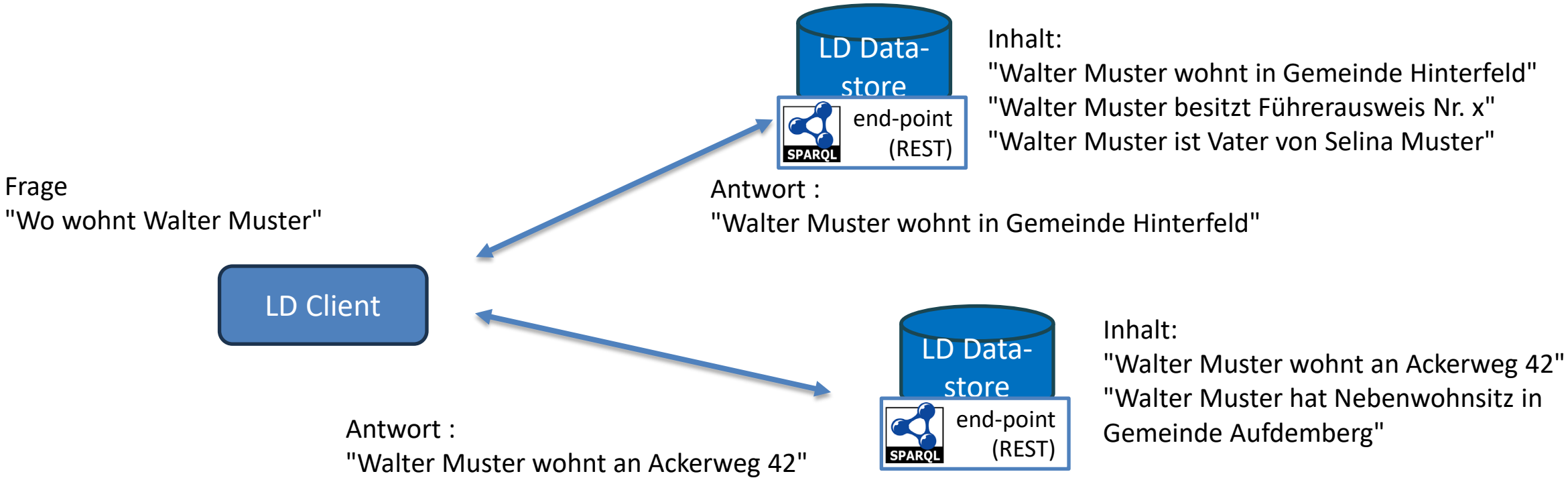
Inhalt:
"Walter Muster arbeitet für Amt für Gurkenzucht"
"Walter Muster arbeitet für Noemi Müller"
"Walter Muster führt Projekt Mondgurke"
" Walter Muster bezieht Kinderzulagen"

Antwort:
"Walter Muster arbeitet für Amt für Gurkenzucht"
"Walter Muster arbeitet für Noemi Müller"
"Walter Muster führt Projekt Mondgurke"

HTTP Protokoll

Resultat der Abfrage kann von einem Datenspeicher beantwortet (verarbeitet) werden
Resultat enthält nur angefragte Information

Abfrage bei Verteilter Information



Abfrage muss durch mehrere Datenspeicher beantwortet werden
Resultatenthält nur

Einfache Abfrage mit Dereferenzieren

Frage:
"Gib mir Information zu Walter Muster"

Http get Request zu Ressourc e

LD Client

LD Data-store

end-point (REST)

Inhalt:
"Walter Muster arbeitet für Amt für Gurkenzucht"
"Walter Muster arbeitet für Noemi Müller"
"Walter Muster führt Projekt Mondgurke"
" Walter Muster bezieht Kinderzulagen"

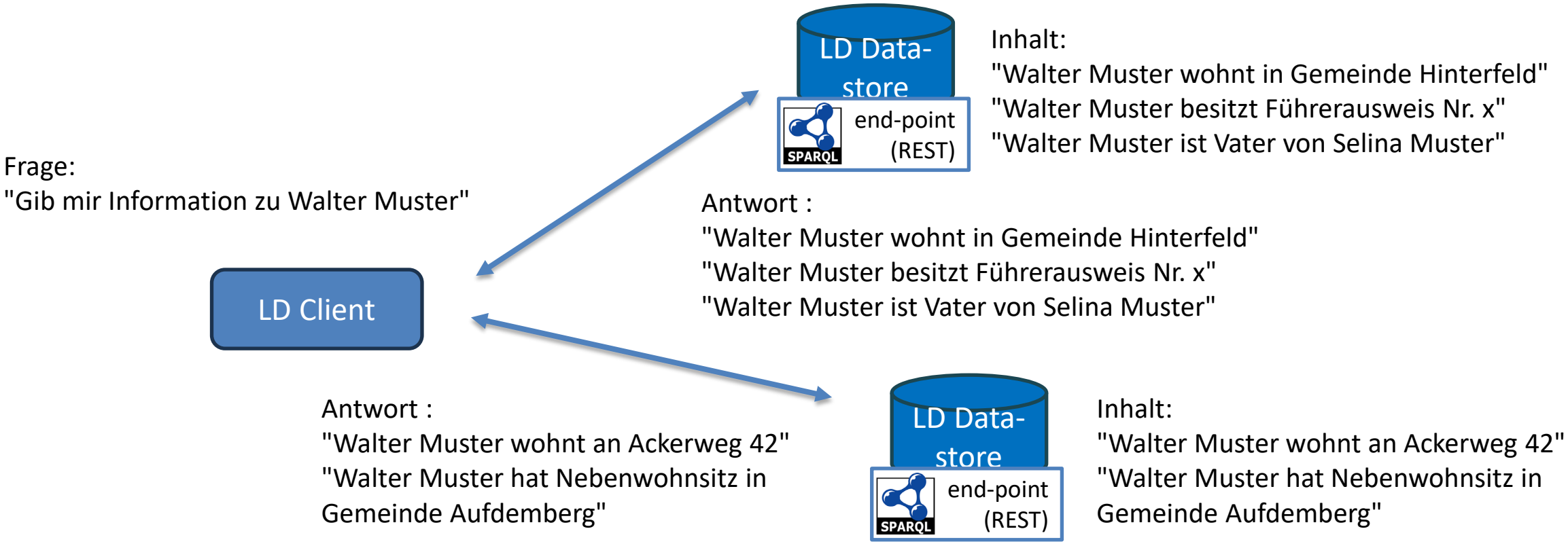
RDF Resultat

Inhalt:
"Walter Muster arbeitet für Amt für Gurkenzucht"
"Walter Muster arbeitet für Noemi Müller"
"Walter Muster führt Projekt Mondgurke"
" Walter Muster bezieht Kinderzulagen"

HTTP Protokoll

Resultat der Abfrage kann von einem Datenspeicher beantwortet werden
Resultat enthält alle Information zu angefragter Ressource (Antwortformat könnte auch xml oder json sein.)

Abfrage bei Verteilter Information mit Dereferenzieren



Abfrage muss durch mehrere Datenspeicher beantwortet werden
Antworten enthalten alle Angaben über Ressource

Gegenüberstellung

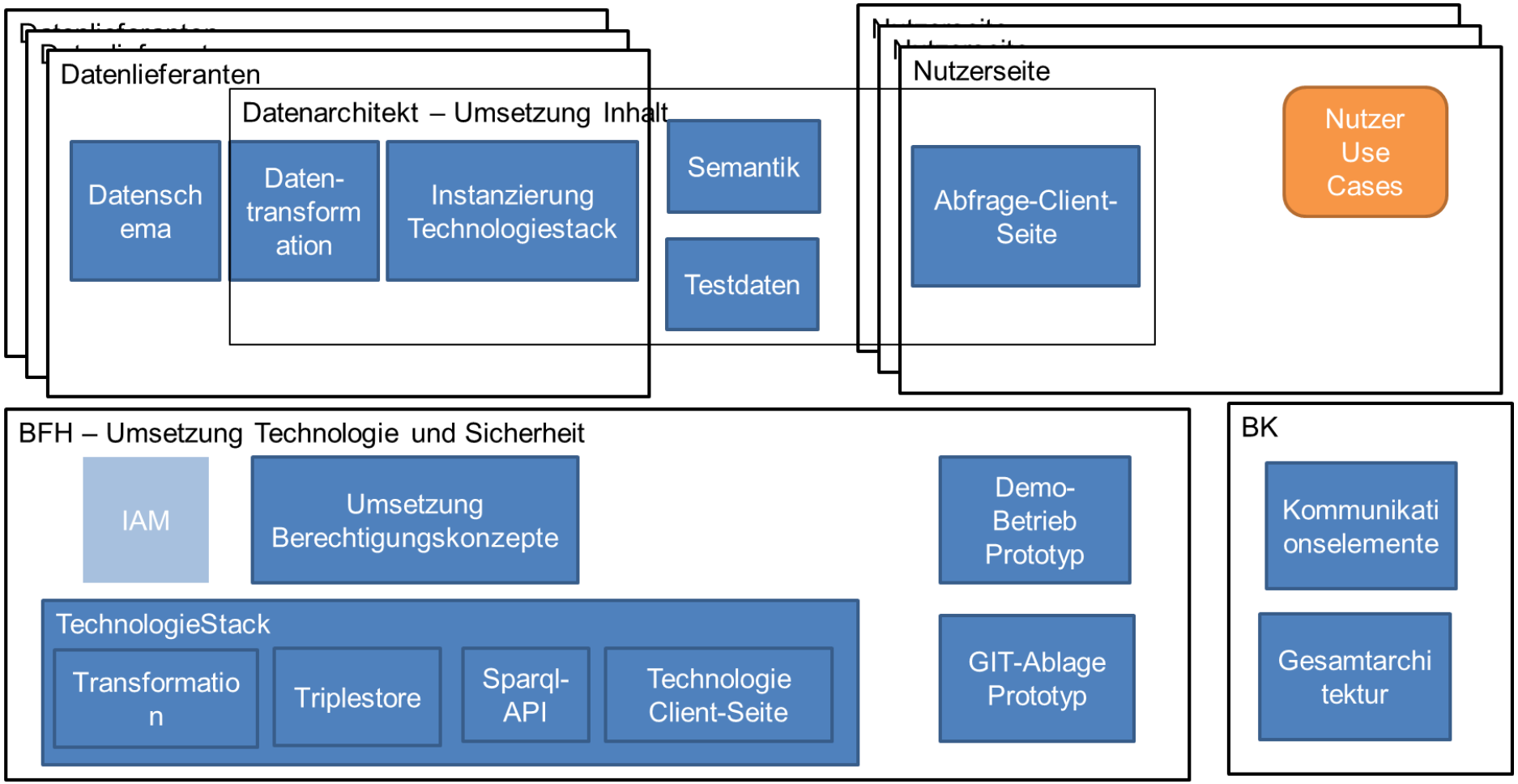
Abfragen mit SPARQL

- Frage kann präziser Formuliert werden
- Antworten enthalten nur angefragte Information
- Inhalt der Information muss bereits bekannt sein, damit Frage präzise formuliert werden kann
- Komplexer in Abfrage - Serververarbeitung, Einfacher im Antworthandling auf Client-Seite
- Wenig Bandbreite, da Antworten schlanker und auch mehrere Ressourcen in einer Antwort möglich

Dereferenzieren

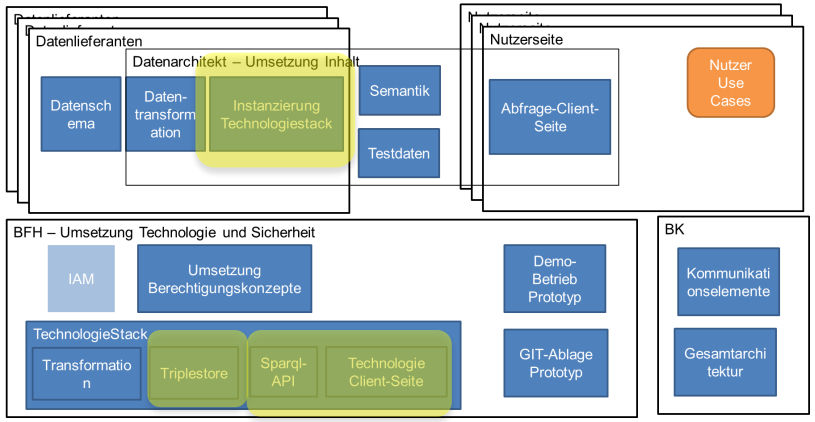
- Frage nach einer Ressource, welche beschrieben ist
- Antwort enthält Ressourcenbeschreibung
- Inhalt muss nicht bekannt sein
- Einfach in Abfrage, Antwort muss verarbeitet werden (wegfiltern von nicht benötigtem) - Alternativ werden Ressourcen in verschiedenen Formaten angeboten.
- Nicht geeignet für Massenabfragen (jede Ressource eine Abfrage).

Arbeitspakete



Herausforderungen und Erfahrungen

- Wahl des Triple-Stores:
  Der Triple-Store wird für den PoC so einfach wie möglich gestaltet. Die einfachste Möglichkeit ist ab einem Turtle-File-Format (.ttl) als Datensatz mit rdflib-endpoint ein Triple Store im Memory zu initialisieren.
- Wahl des Sparql-Endpunkt:
  mit rdflib-endpoint gibt es eine Bibliothek, welche als SPARQL-Endpoint die in-Memory Datenbank zugänglich macht.
- Wahl der Abfrage:
  mit einem Python-Skript können mit rdflib die Abfragen durchgeführt werden.



Herausforderungen und Erfahrungen (Semantik)

Tests pour l‘interrogation de deux sources de données:

- 2 petits jeux de données actuels: EWR (résidences principales), UPI (personnes)
- 2 jeux de données publiés sur 2 SPARQL end-point
- Lien entre les deux: identifiant de la personne (URL) basé sur le numéro AVS
- Requête EWR pour obtenir toutes les personnes (leur identifiant) et leur résidence principale, requête(s) UPI pour obtenir les informations sur ces personnes (nom, prénom).

PoC de comparaison des possibilités de requêtage:

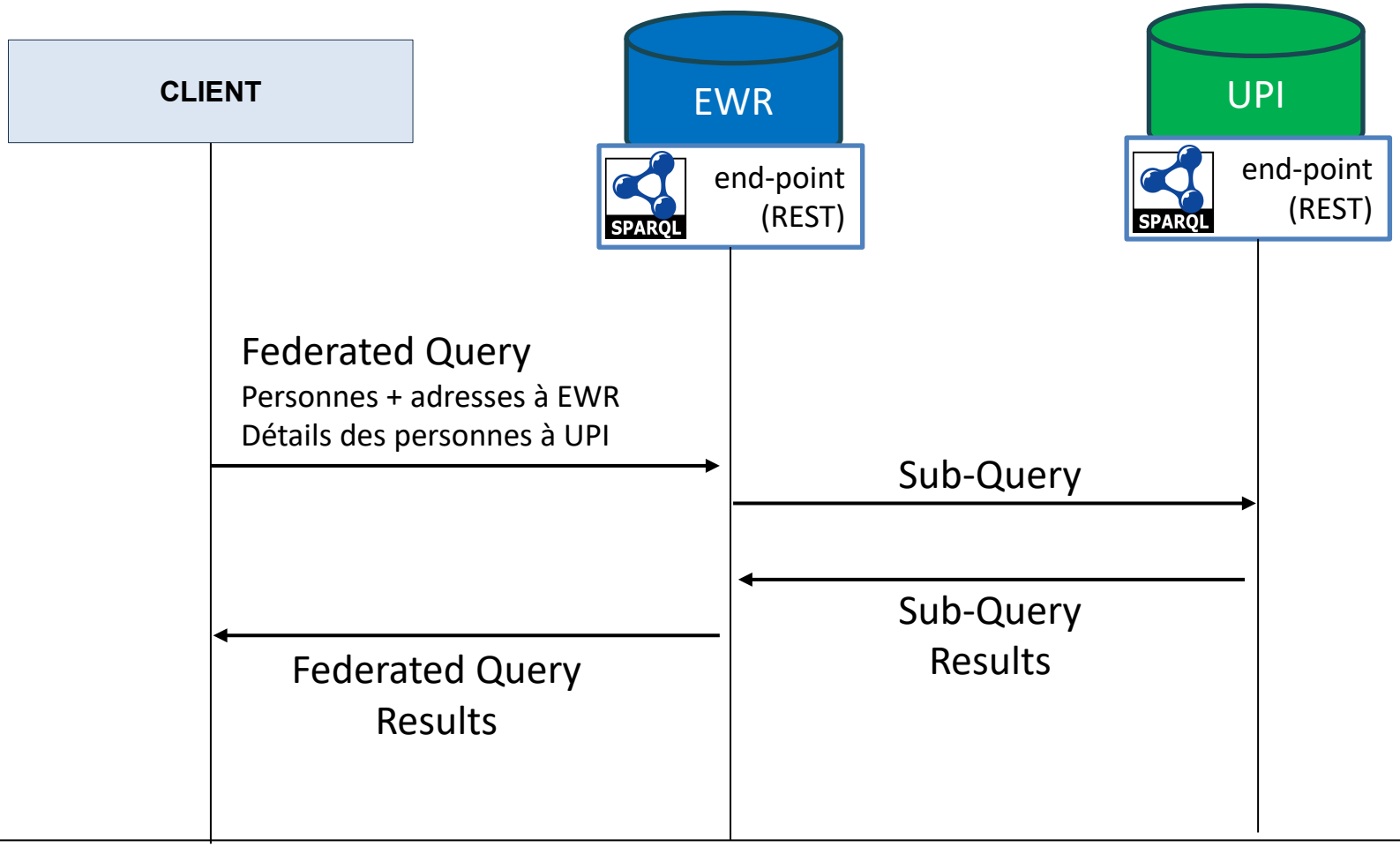
- Une seule requête SPARQL: Federated SPARQL
- Une requête EWR puis une seule requête UPI
- Une requête EWR puis une requête UPI par personne

PoC du déréférencement:

- Une requête EWR pour obtenir les personnes et leur URL de Wikidata
- Puis un HTTP GET pour chaque URL Wikidata

Herausforderungen und Erfahrungen (Abfrage Client)

Variante 1: Une seule requête SPARQL: Federated SPARQL



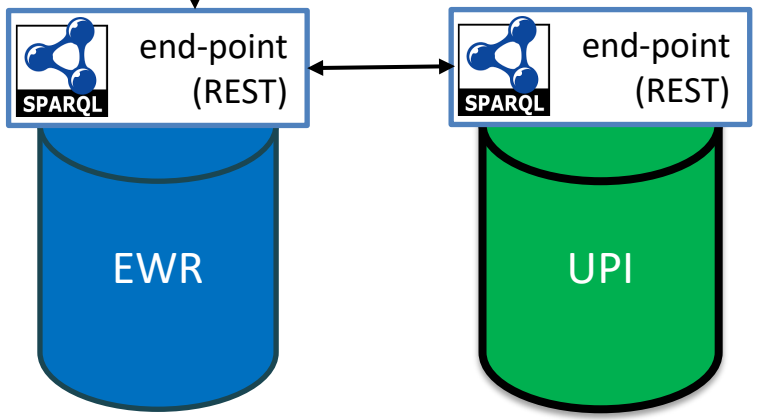
Herausforderungen und Erfahrungen (Abfrage Client-Seite)

Variante 1: Une seule requête SPARQL: Federated SPARQL

Mot-clé SERVICE pour spécifier sur quel end-point trouver certaines données

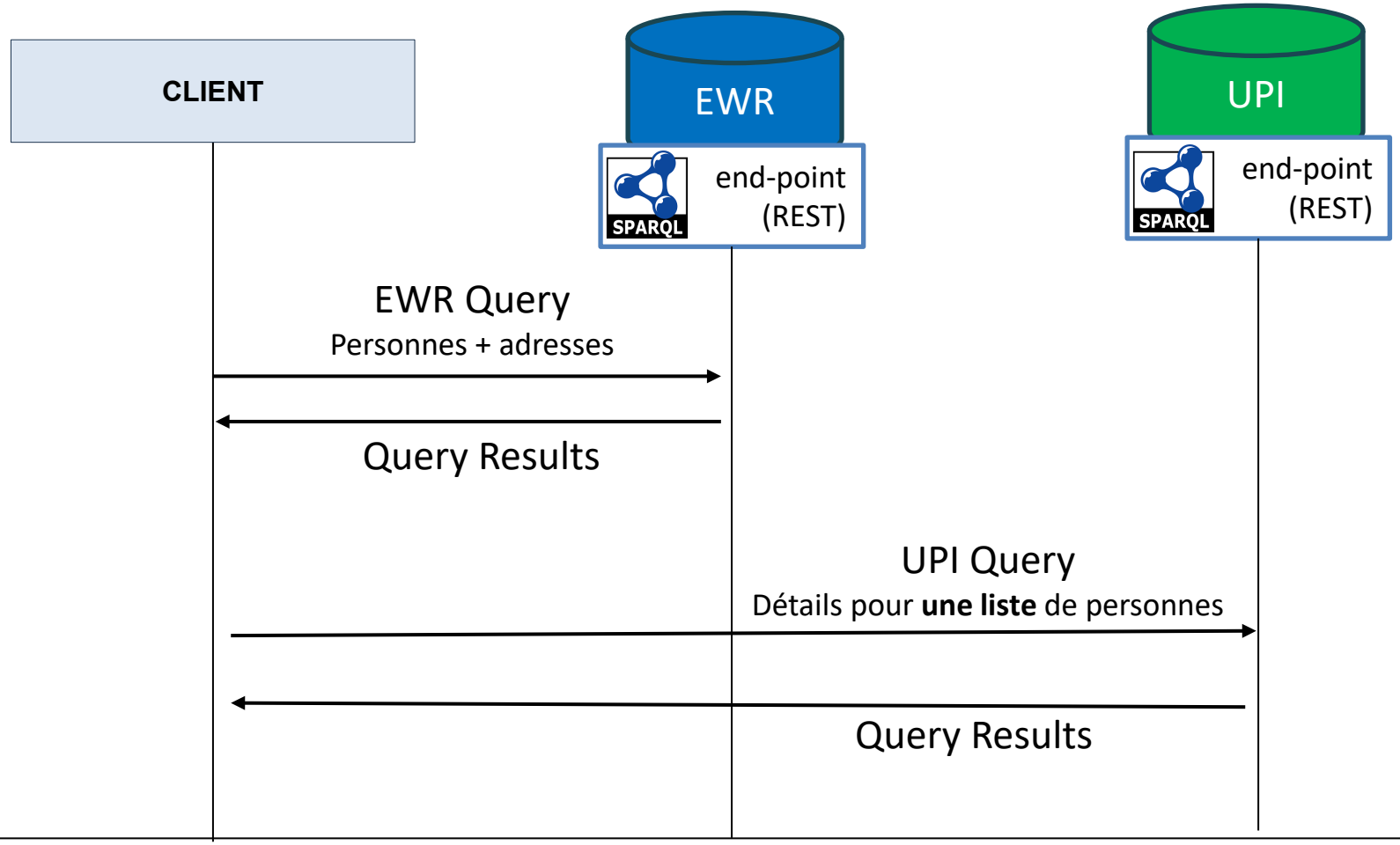
```
SELECT ?person ?city ?name WHERE {
?evidence dct:conformsTo ex:evidenceTypePrincipalResidence ;
  dct:subject ?person ;
  cv:registeredAddress/locn:postName ?city .

SERVICE <http://upi-endpoint.ch> {
   ?person foaf:familyName ?name .
   }
}
```



Herausforderungen und Erfahrungen (Abfrage Client-Seite)

Variante 2: Une requête EWR puis une seule requête UPI

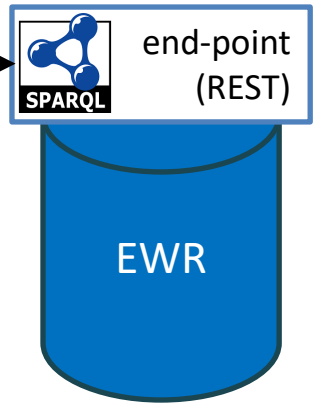


Herausforderungen und Erfahrungen (Abfrage Client-Seite)

Variante 2: Une requête EWR puis une seule requête UPI

Requête EWR pour obtenir la liste de personnes et leur adresse

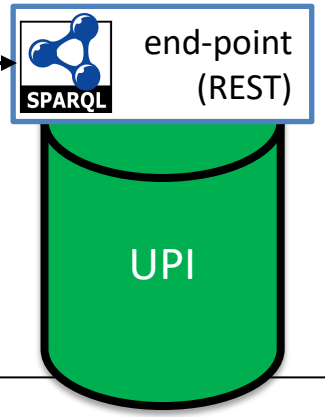
```
SELECT ?person ?city WHERE {
?evidence dct:conformsTo ex:evidenceTypePrincipalResidence ;
  dct:subject ?person ;
  cv:registeredAddress/locn:postName ?city .
```



➔ Selon le résultat: création de la liste de personnes

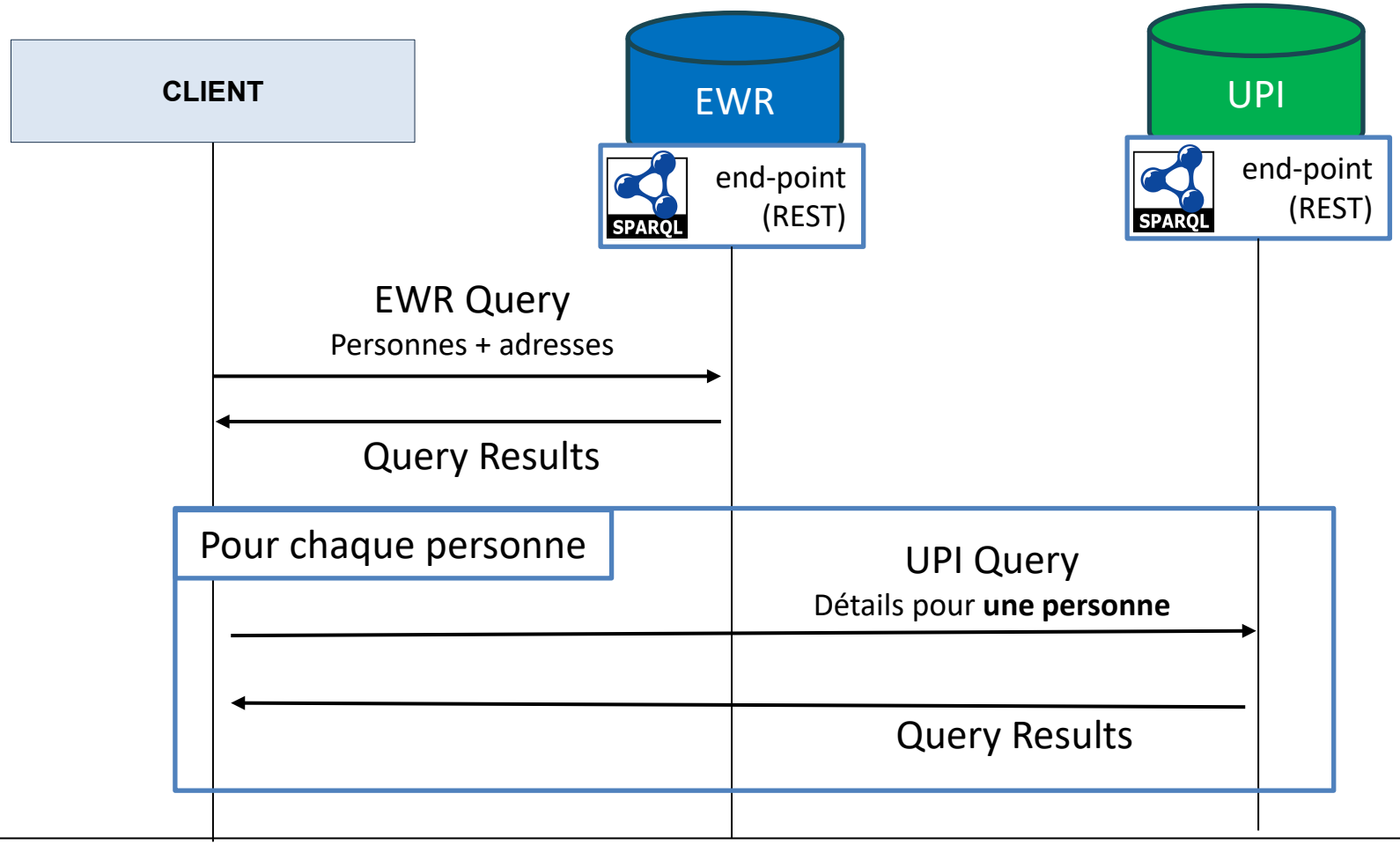
➔ Utilisation de cette liste pour le mot-clé VALUES: limiter les résultats possibles de la deuxième requête

```
SELECT ?person ?name WHERE {
VALUES ?person { PERSON_VALUES_TO_REPLACE }
?person foaf:familyName ?name .
}
```



Herausforderungen und Erfahrungen (Abfrage Client-Seite)

Variante 3: Une requête EWR puis une requête UPI par personne

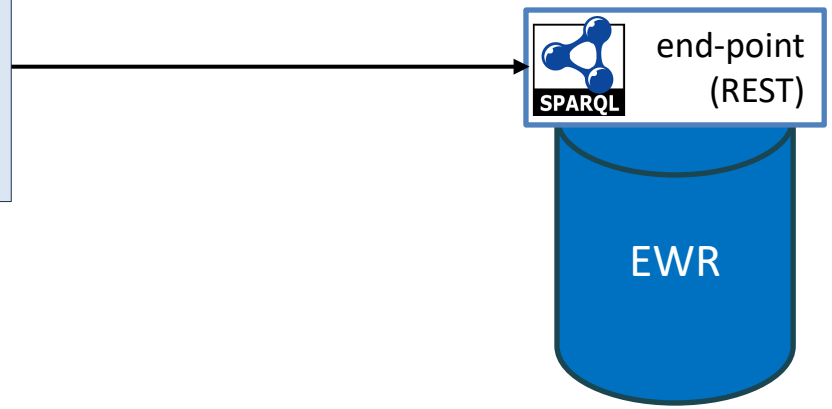


Herausforderungen und Erfahrungen (Abfrage Client-Seite)

Variante 3: Une requête EWR puis une requête UPI par personne

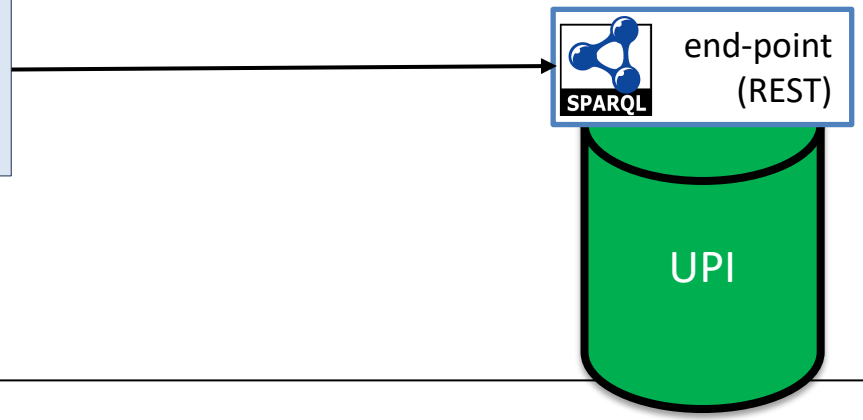
Requête EWR pour obtenir la liste de personnes et leur adresse

```
SELECT ?person ?city WHERE {
?evidence dct:conformsTo ex:evidenceTypePrincipalResidence ;
  dct:subject ?person ;
  cv:registeredAddress/locn:postName ?city .
```



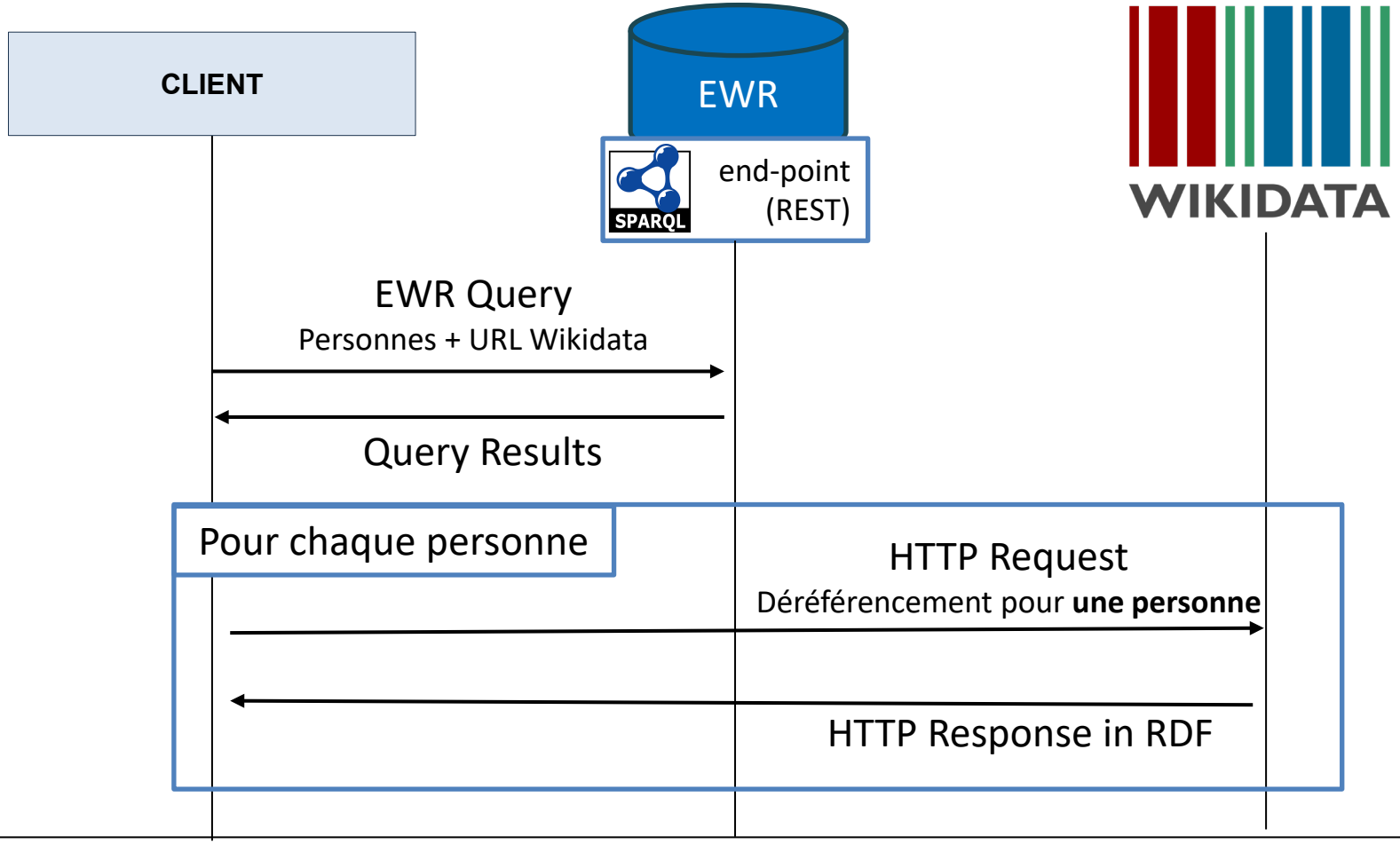
➔ Boucler sur les personnes du résultat et interroger UPI personne par personne

```
SELECT ?name WHERE {
<ONE_SPECIFIC_PERSON> foaf:familyName ?name .
}
```



Herausforderungen und Erfahrungen (Abfrage Client-Seite)

Variante 4: Une requête EWR puis déréférencement des URLs des personnes sur Wikidata

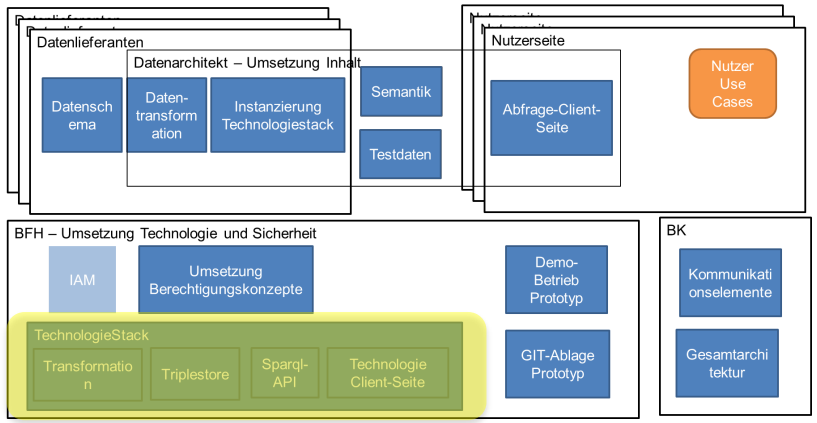


Fragen zur Wahl des Abfrage-Stils

- Wo wird welche Last ausgelöst
  - Wenn viele Abfragen durch einen verarbeitet werden müssen, kann dieser zum Flaschenhals werden. - Diese Last eher dort ansiedeln, wo es viele Teilnehmer gibt.
  - Wenn viele Einzelabfragen durchgeführt werden, kann die Netzwerklast ein Problem werden
- Variante 1: Vertrauen in Sicherheitsprüfung von anderem Amt oder Sicherheit-Informationen föderieren. Nur eine Endpunkt-Integration für Client
- Variante 2: alle Endpunkte müssen angebunden werden.
- Variante 3: eher nicht wählen, wenn oft grössere Mengen angefragt werden, da Last für Netzwerk und Clientseite hoch wird.
- Variante 4: Vertrauen darin, das Clientseite die "zu viel" gelieferte Information verwirft. Einschränkung von V3 noch verstärkt, da Datenpakete noch grösser sind und auf Clientseite mehr verarbeitet werden muss.

Herausforderungen und Erfahrungen (Technologie-Stack)

- **Technologie-Evaluation**
- **Technische Umsetzung Prototyp (Konzept)**



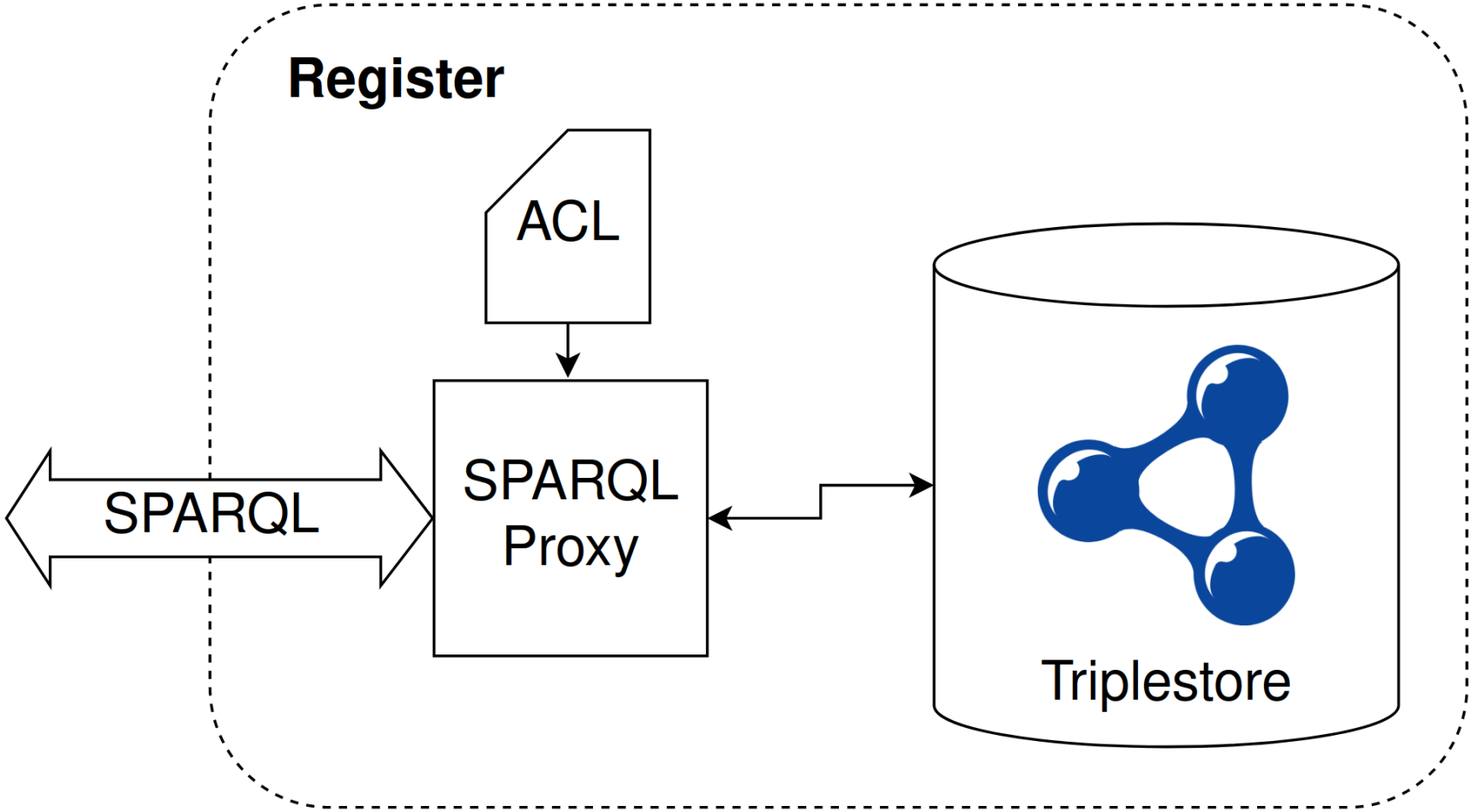
Technologie-Evaluation

- **Marktanalyse Triplestores**
  - Status: Abgeschlossen
  - Fokus: angebotene Mechanismen zur Zugriffskontrolle
  - Untersuchte Triplestores:
    Fuseki2, Telicent, Oxigraph, Stardog, Oracle, GraphDB, Parliament, Virtuoso
- **Erkenntnisse**
  - Umsetzung Zugriffskontrolle sehr unterschiedlich...
  - Technologie Triplestore grundsätzlich reif, austauschbar
  - Zum aktuellen Zeitpunkt keine Empfehlung für bestimmte(n) Triplestore(s)
    Begründung: Reale Umsetzung erfolgt erst in Zukunft, Lösungen ändern sich

Technische Umsetzung Prototyp (Konzept)

- **Entscheid Architektur**
  - Architekturvariante Proxy
- **Begründung**
  - Unabhängig von konkretem "Backend-Triplestore"
  - Lässt sich vor jeden SPARQL-Endpunkt schalten
  - Ermöglicht Abbildung der Use Cases
  - Ermöglicht jedoch auch freie Tests gegen beliebige Datenquellen (bspw. Wikidata)
  - Potential für weitere Funktionalität, bspw. Debugging/Logging/Accounting

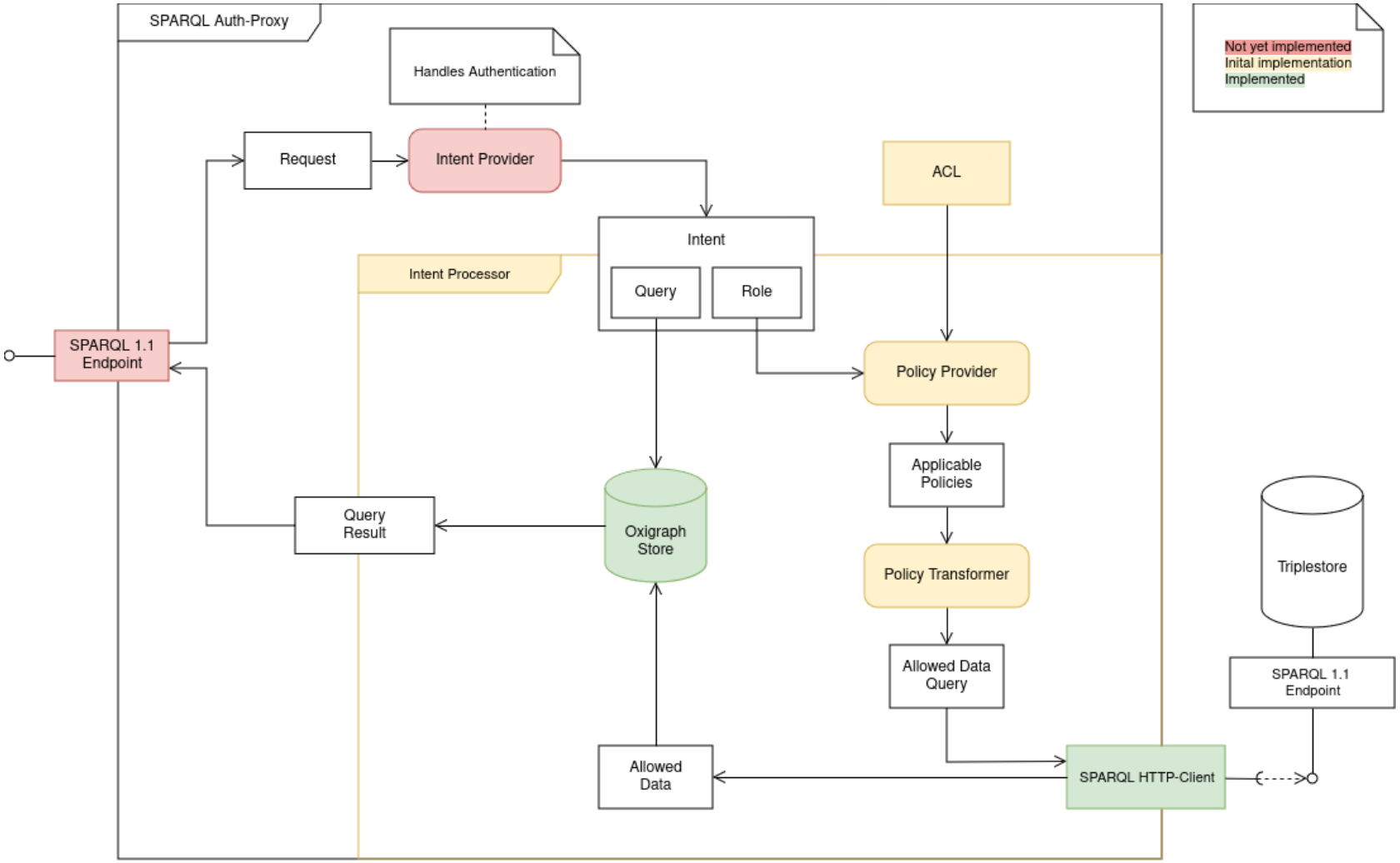
Technische Umsetzung Prototyp (Konzept)



Technische Umsetzung Prototyp (Konzept)

- **Idee**
  - *"Authorization Proxy for SPARQL Endpoints"*, R. Stojanov + M. Jovanovik, 2017
  - *"Linked Data Authorization Platform"*, R. Stojanov, S. Gramatikov, I. Mishkovski + D. Trajanov, 2017 / 2018
  - Implementation vorhanden, jedoch veraltet (bspw. Java 8)
  - Kann evtl. als Vergleichsvariante für Tests in Betrieb genommen werden
- **sparql-auth-proxy (BFH)**
  - Implementation in Javascript
  - Im Wesentlichen SPARQL-Endpoint + SPARQL-Client
    + IAM-Anbindung und ACL-Engine
    + In-memory Triplestore
  - Enforcement der ACLs durch modifizierte Queries an Backend
  - Client-Query (zum Proxy) auf reduzierten Graph im internen Triplestore

Architekturübersicht Proxy



Beispiel für eine ACL mit zwei Policies

```
{
    "role": null,
    "permission": "allow",
    "subject": [ "wd:Q556", "wd:Q560" ],
    "predicate": "*",
    "object": "*",
    "priority": 1
},
{
    "role": "alice",
    "permission": "deny",
    "subject": "*",
    "predicate": "rdfs:label",
    "object": "*",
    "priority": 2
}
```

Proxy-Demo!

Termine

Nächste Sounding Boards

- **22. August 2024 – 09:00 – 10:00 Uhr (kurzes Update)**
- **24. Okt. 2024 09:00 – 11:00 Uhr (Prototyp und gelerntes)**

Weitere Termine ...

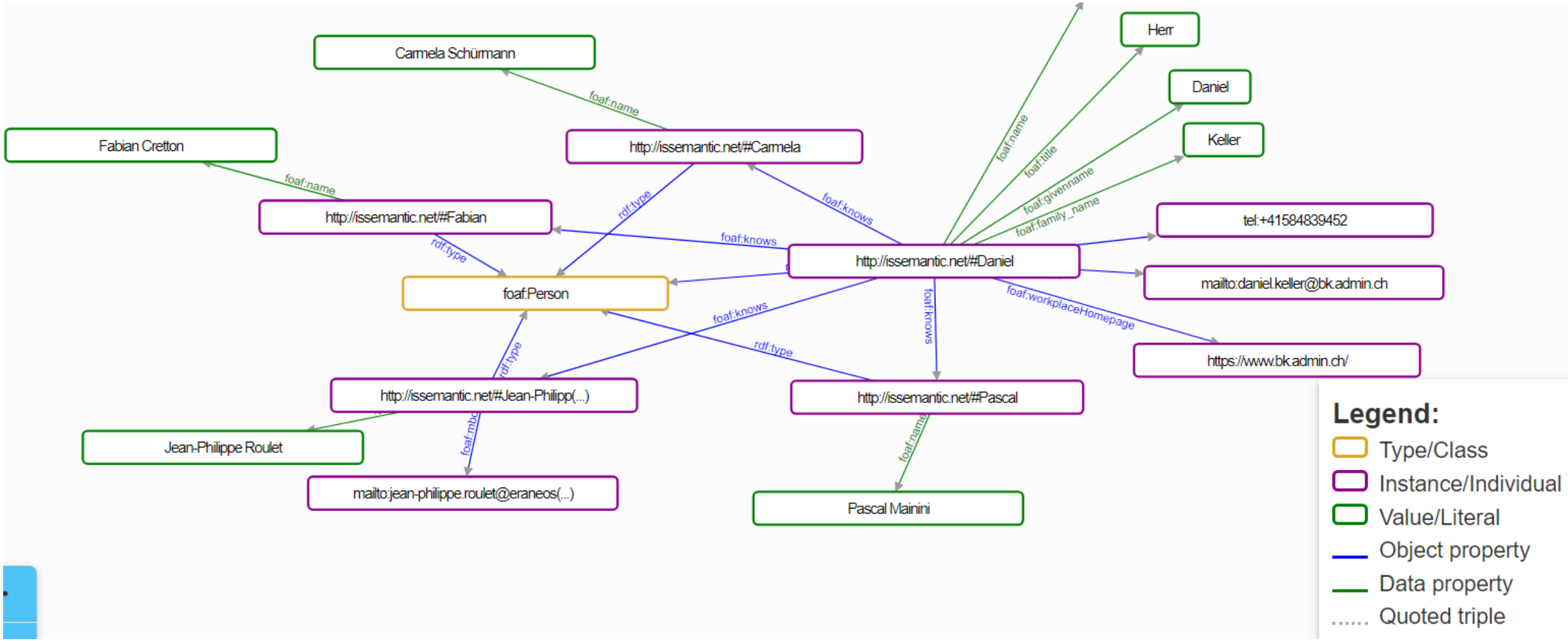
- **05. Nov. 2024 17:00 – 19:00 Uhr**

  Präsentation am Linked Data Meetup

Schulungen (Bundesverwaltung)

- **Linked Data – Einführung: DE 14.08.2024, Fr. 19.11.2024**
- **Linked Data – Lindas Tools: DE 13.06.2024, Fr. 19.12.2024**

Kontakt



Made by: FOAF-a-matic -- Sich selbst beschreiben mit RDF (Idodds.com) & Online RDF graph visualizer: Turtle, JSON-LD, TriG, RDF-star (issemantic

Backup Folien "Proxy-Demo"

Wikidata: Query ohne Rolle

Proxy: Query ohne Rolle

```
npm run repl

PS ~\Documents\Git\raa3\sparql-auth-proxy master> npm run repl

> sparql-auth-proxy@1.0.0 repl
> node ./etc/repl.js ./settings/

running setup...
loading settings\config.json
loading settings\prefixes.json
loading settings\policies.json
> await query("SELECT ?l WHERE { ?s rdfs:label ?l. }")
processing intent...
{
  role: null,
  queryString: 'PREFIX wd: <http://www.wikidata.org/entity/>\n' +
    'PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n' +
    'SELECT ?l WHERE { ?s rdfs:label ?l. }'
}
transforming policies into construct query...
{
  constructQuery: 'PREFIX wd: <http://www.wikidata.org/entity/>\n' +
    'PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n' +
    'CONSTRUCT { ?s ?p ?o. } WHERE\n' +
    '{\n' +
    'VALUES ?s { wd:Q556 wd:Q560 }\n' +
    '?s ?p ?o.\n' +
    '}'
}
loading temporal dataset...
{ datasetSize: 1651 }
executing intent query against temporal dataset...
{ resultSize: 444 }
undefined
>
```

Wikidata: Simulierte Berechtigungen Rolle "alice"

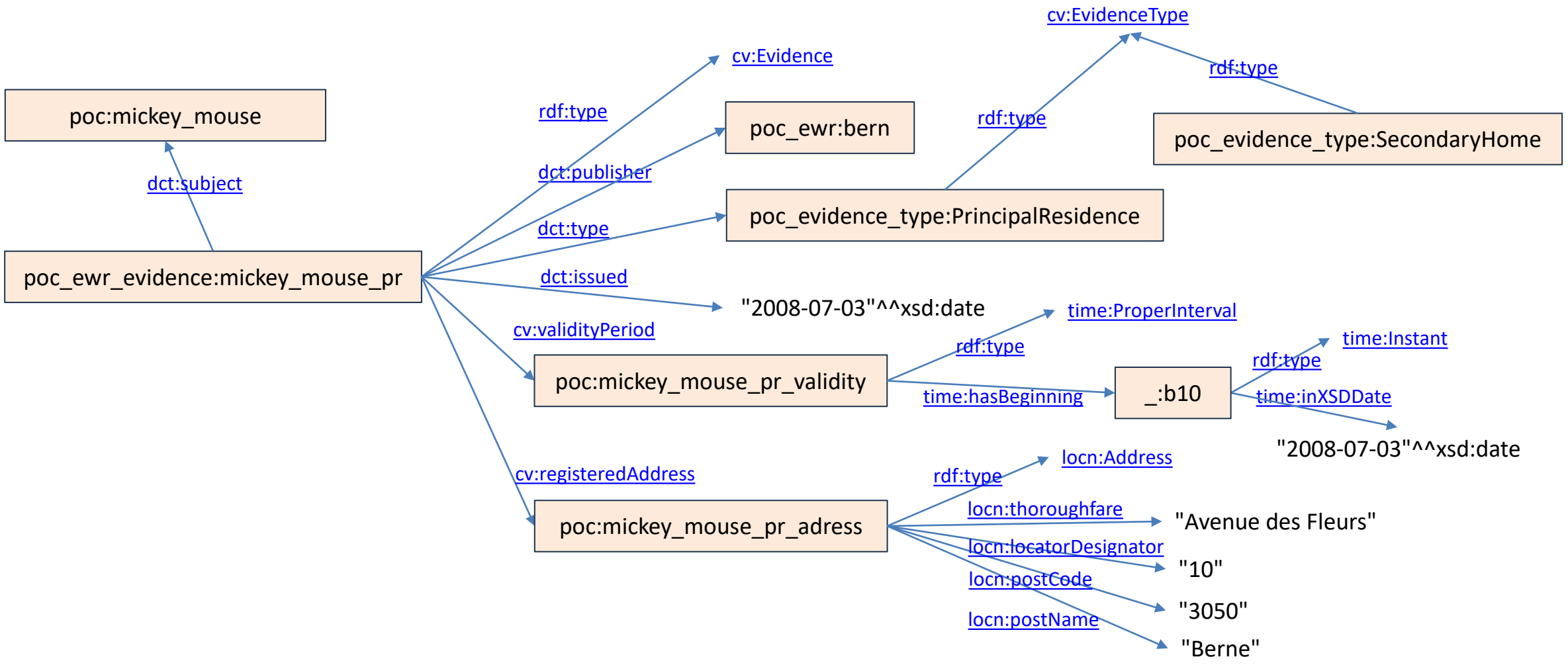
Proxy: Query mit Rolle "alice"

```
npm run repl

PS ~\Documents\Git\raa3\sparql-auth-proxy master> npm run repl

> sparql-auth-proxy@1.0.0 repl
> node ./etc/repl.js ./settings/

running setup...
loading settings\config.json
loading settings\prefixes.json
loading settings\policies.json
> login("alice")
logged in as (alice)!
undefined
> await query("SELECT ?l WHERE { ?s rdfs:label ?l. }")
processing intent...
{
  role: 'alice',
  queryString: 'PREFIX wd: <http://www.wikidata.org/entity/>\n' +
    'PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n' +
    'SELECT ?l WHERE { ?s rdfs:label ?l. }'
}
transforming policies into construct query...
{
  constructQuery: 'PREFIX wd: <http://www.wikidata.org/entity/>\n' +
    'PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n' +
    'CONSTRUCT { ?s ?p ?o. } WHERE\n' +
    '{\n' +
    '{\n' +
    'VALUES ?s { wd:Q556 wd:Q560 }\n' +
    '?s ?p ?o.\n' +
    '}\n' +
    'MINUS\n' +
    '{\n' +
    'VALUES ?p { rdfs:label }\n' +
    '?s ?p ?o.\n' +
    '}\n' +
    '}'
}
loading temporal dataset...
{ datasetSize: 1207 }
executing intent query against temporal dataset...
{ resultSize: 0 }
undefined
>
```

EWR Meldeverhältnis

pour un personnage fictif UPI – Résidence principale: