

JSON & SKOS

MALIS25: Modul IT2

5. Präsenzphase, Adrian Pohl, Petra Maier

2025-10-08

Dokumentation & Fragen in HedgeDoc

-> <https://pad.lobid.org/malis25-it2-maier-pohl>

Begrüßung

Werdegang Adrian

- Magister Kommunikationswissenschaft und Philosophie an der RWTH Aachen
- MALIS09-Absolvent, zum 5. Mal MALIS-Lehre
- Seit 2008 im hbz, seit 2019 Leitung der Gruppe Metadateninfrastruktur
- Arbeitsschwerpunkte:
 - Management eines Software-Entwicklungsteams
 - Datenmodellierung
 - Qualitätsmanagement
 - Standardisierung

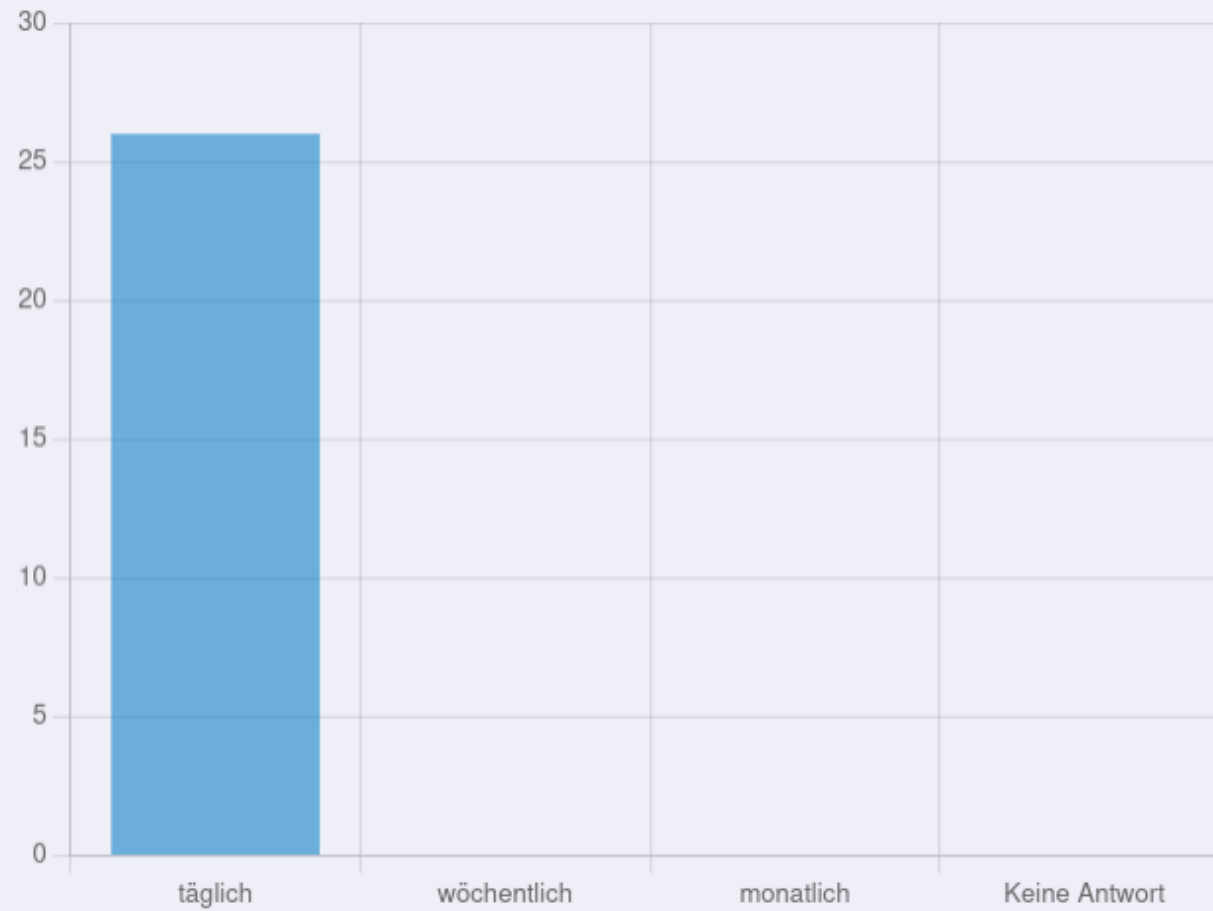
Werdegang Petra

- Bibliotheks- und Informationsmanagement BA (HdM)
- MALIS13-Absolventin
- ZBIW-Zertifikat “Data Librarian”
- aktuell: Datenkonvertierung an der VZG
- Arbeitsschwerpunkte:
 - Datentransformation (überwiegend mit Python, teils Metafacture)
 - Datenqualitätskontrollen/-analysen
 - Standardisierungsarbeit, konzeptionelle Datenmappings

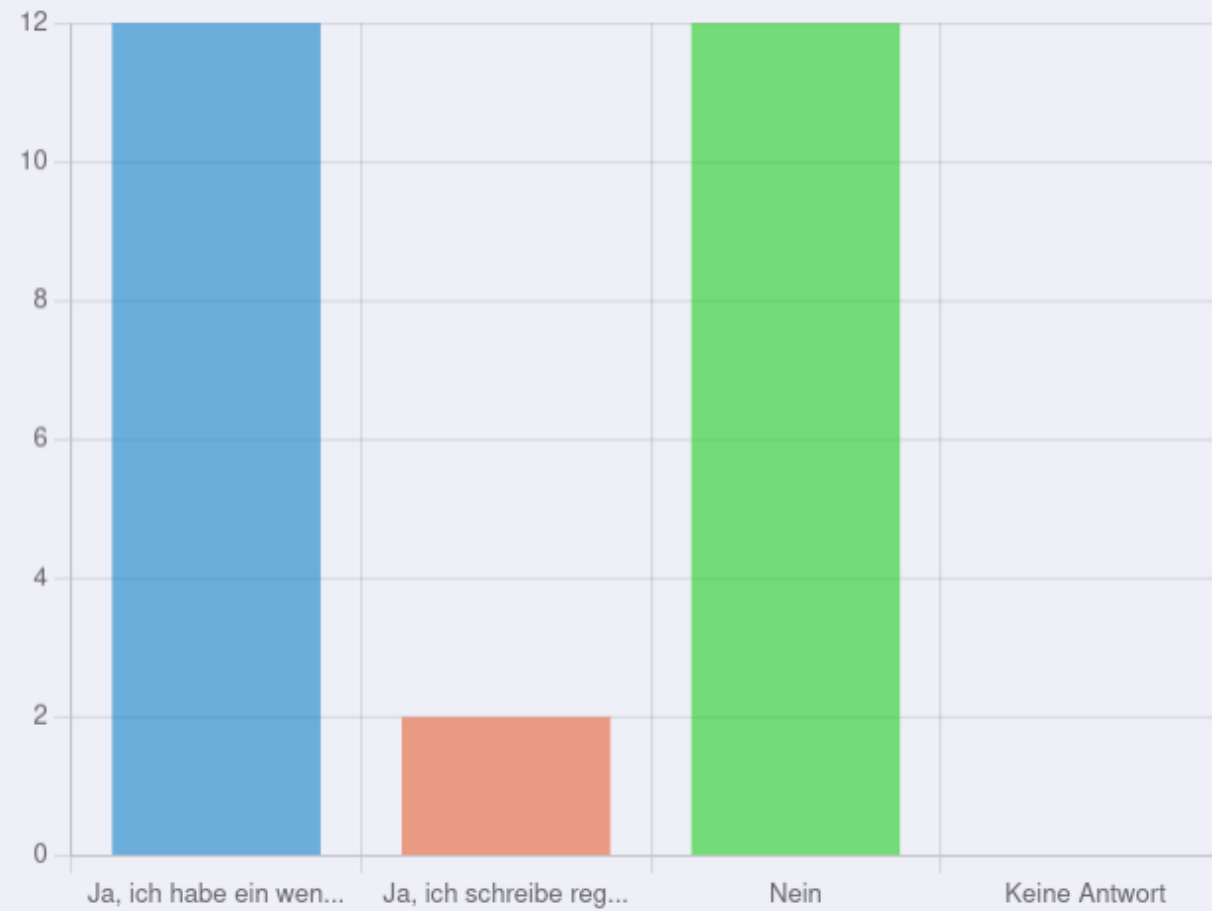
Und Sie?

- Moodle-Profile haben wir uns angeschaut.
- Nun eine kurze Auswertung der Vorabumfrage

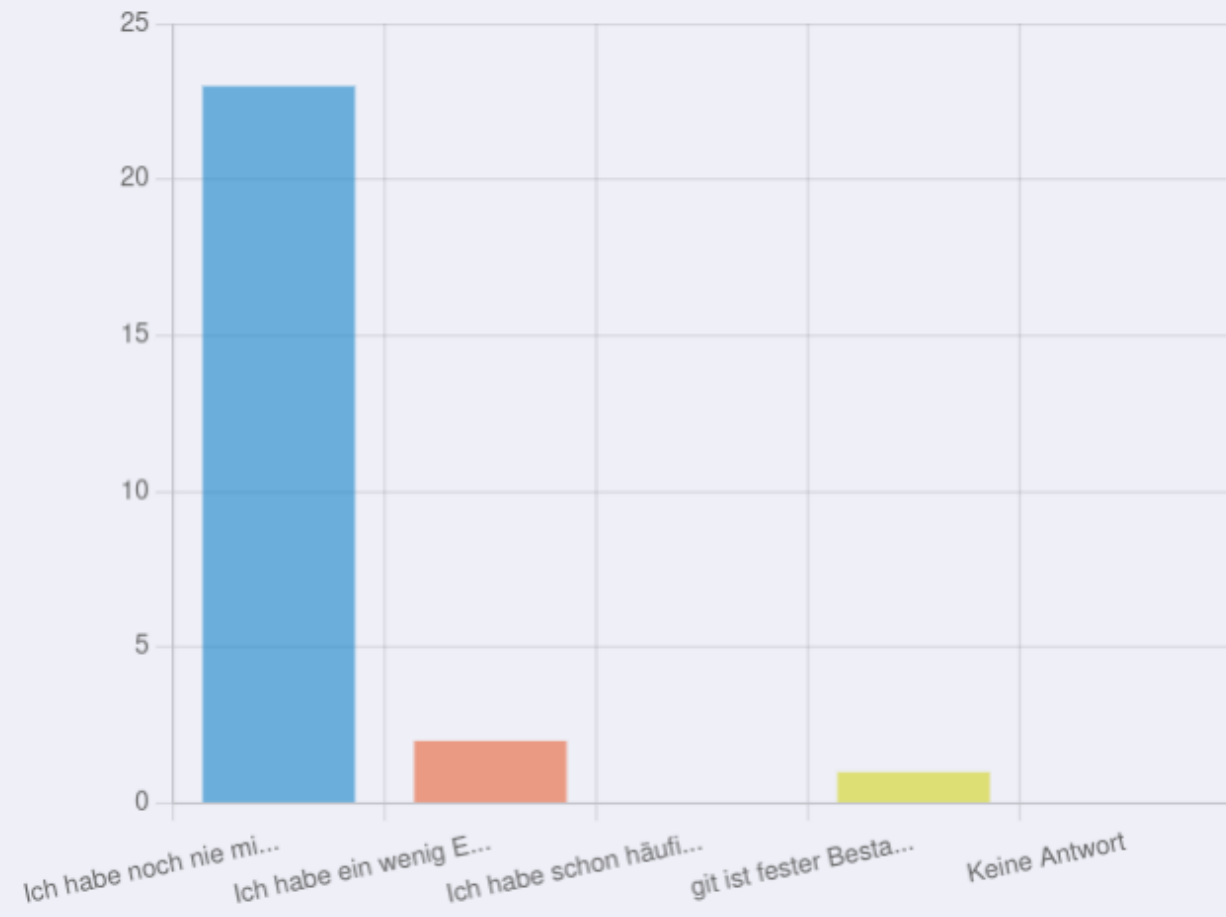
Wie häufig benutzen Sie einen Webbrowser?



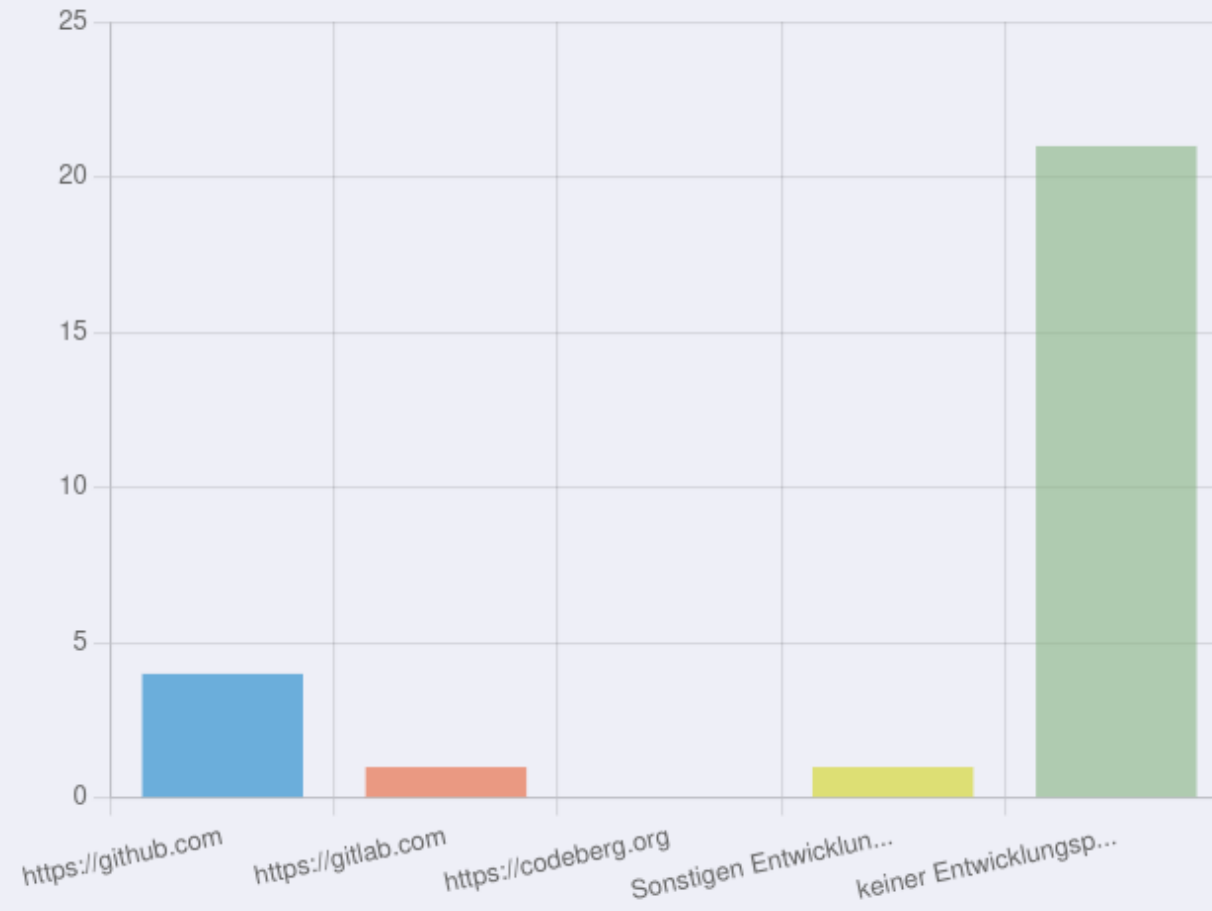
Kennen Sie Markdown?



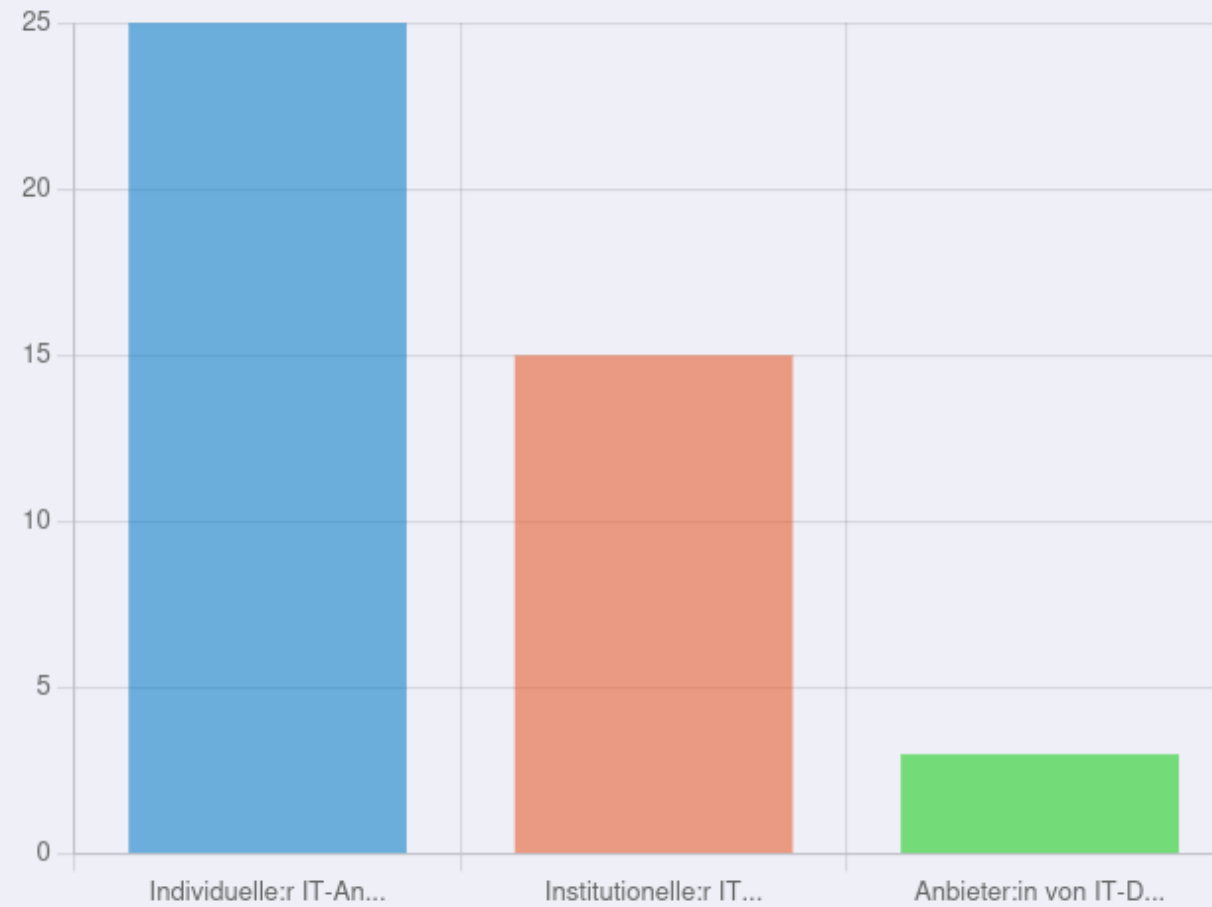
Wie sind Ihre git-Kenntnisse?



Ich habe ein Konto bei...

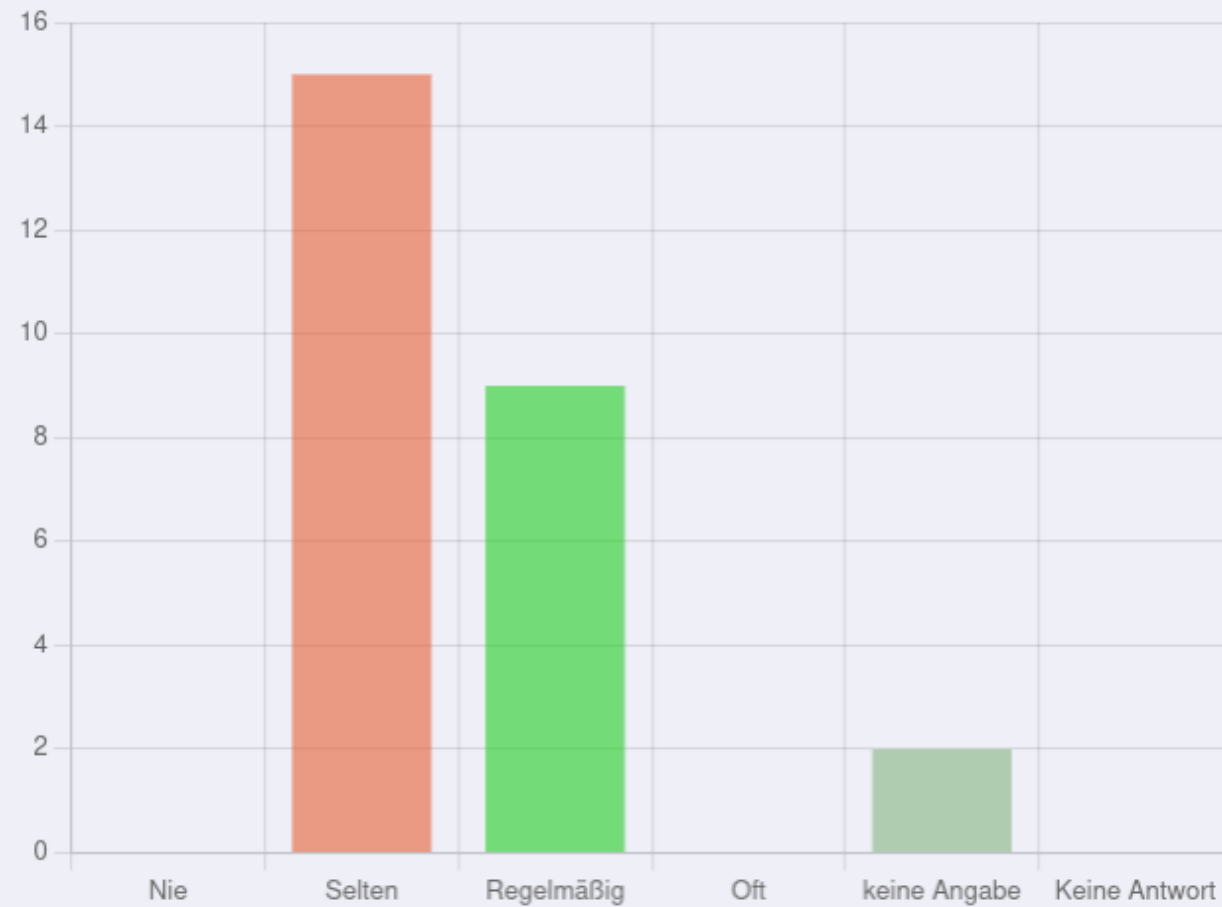


Welche Rollen nehmen Sie bei der IT-Nutzung ein?



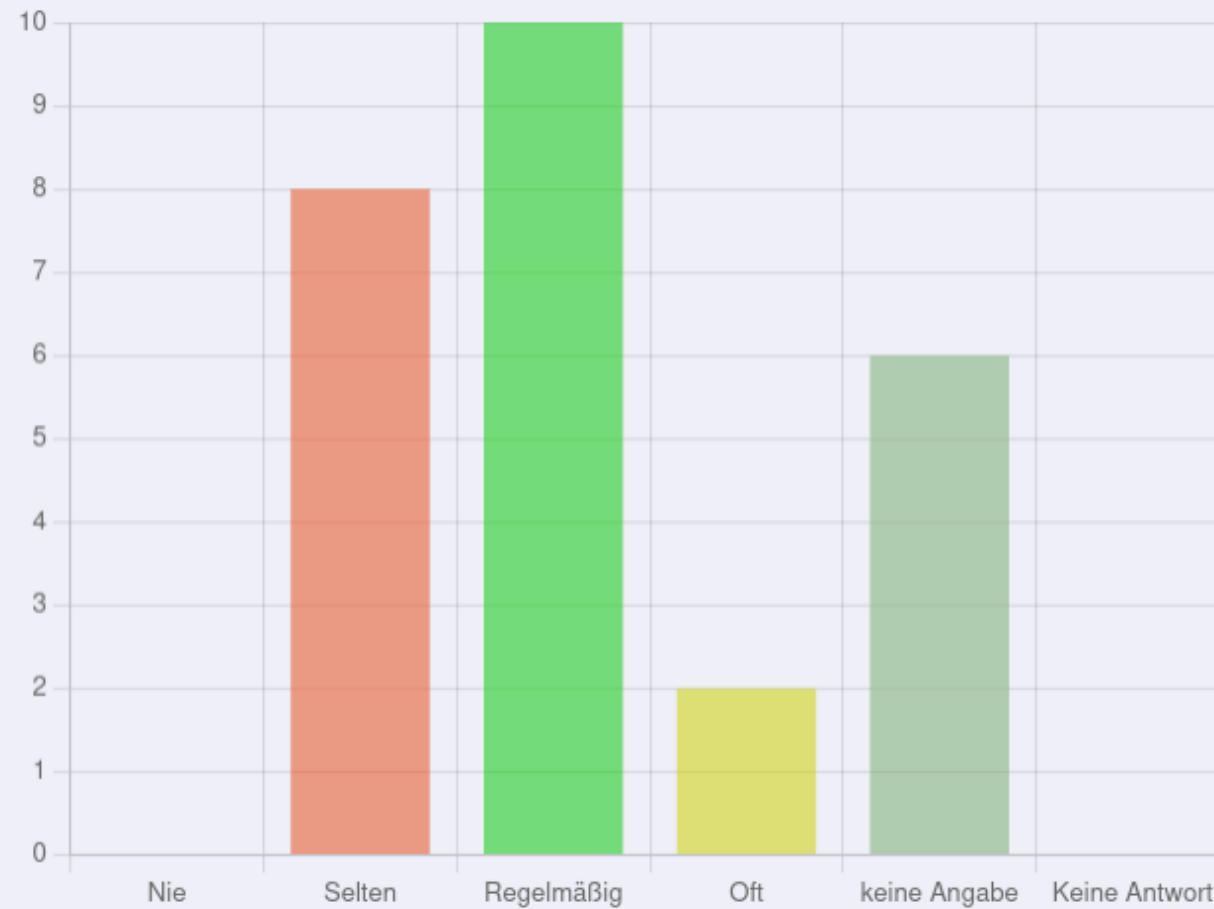
Denken Sie an im Arbeitsalltag auftretende IT-Probleme. In welcher Rolle treten die Probleme auf?

[als individuelle:r IT-Anwender:in]

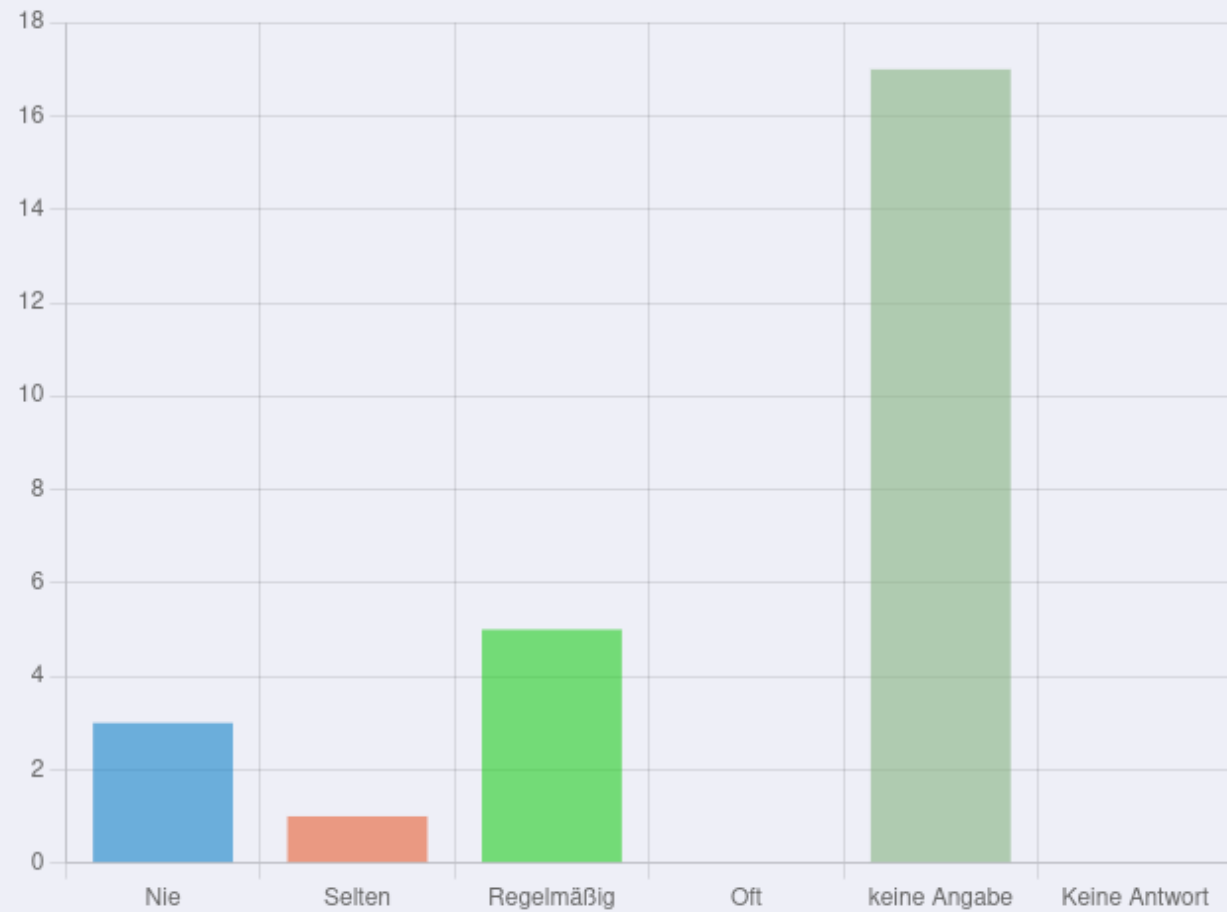


Denken Sie an im Arbeitsalltag auftretende IT-Probleme. In welcher Rolle treten die Probleme auf?

[als institutionelle:r IT-Anwender:in]



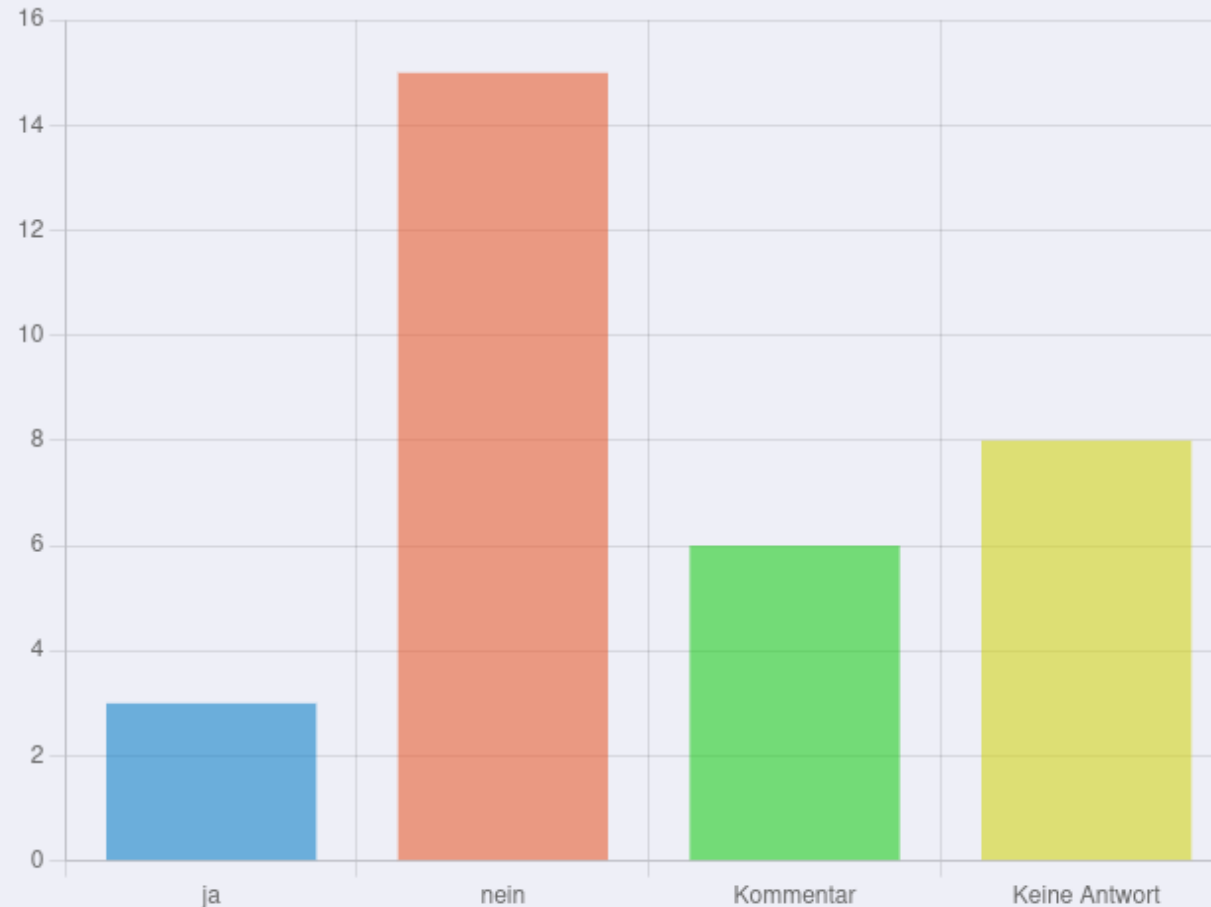
**Denken Sie an im Arbeitsalltag auftretende IT-Probleme. In welcher Rolle treten die Probleme auf?
[als Anbieter:in von IT-Diensten]**



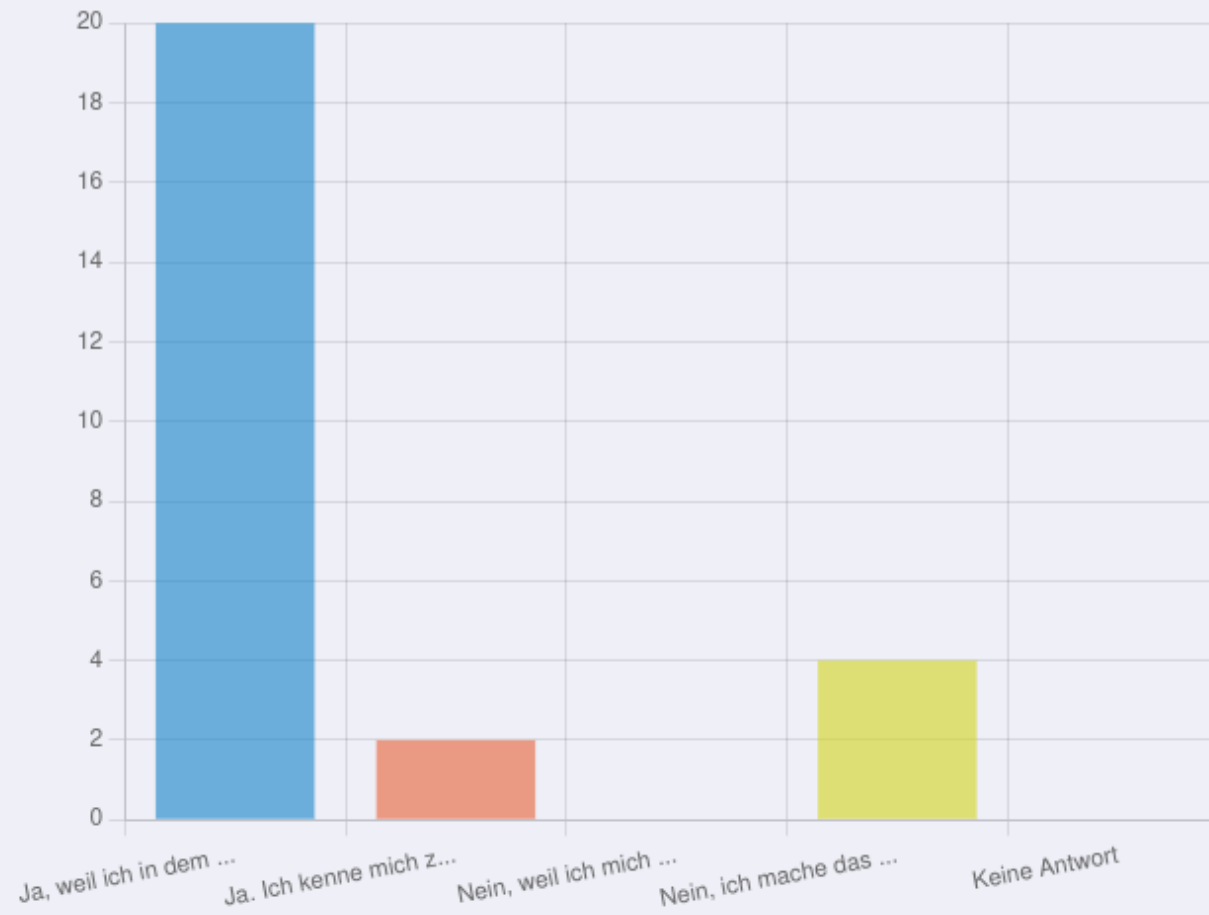
**Schildern Sie ein aktuelles – bestenfalls noch
ungelöstes – IT-Problem aus Ihrer Arbeitspraxis**



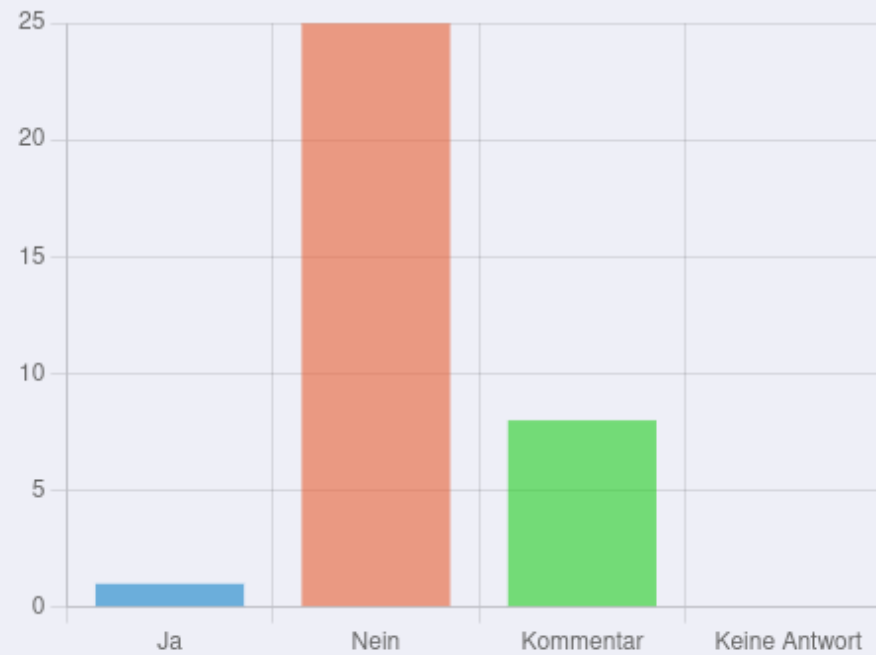
Wären Sie damit einverstanden, das geschilderte Problem als Input für eine moderierte Gruppendiskussion während der nächsten Präsenzphase zu verwenden? Falls ja, geben Sie bitte Ihre E-Mail-Adresse an.



Wir werden uns in IT2 mit JSON und Web-APIs, KI, Linked Open Data und dem Umgang mit IT-Problemen aus Ihrer Praxis befassen. Sehen Sie den Inhalten mit Interesse entgegen?



Für eine Wahloption der Aufgabe IT2.1a müssen zwei npm-Module installiert werden und ggf. ein Bash-Skript ausführbar gemacht werden. Wären Sie ggf. bereit, eher unerfahrene Kommiliton:innen bei der Einrichtung zu unterstützen?



Organisatorisches

Modul-Etikette

- Fragen sind stets willkommen.
- Helfen Sie anderen, wo Sie können.
- Seien Sie offen für unterschiedliche Fachkenntnisse & Erfahrungen.
- Seien Sie freundlich.

Aufgabe

- Einzelarbeit
- 25% der IT2-Note, wenn Praxisprojekt in IT2
- 50% der IT2-Note, wenn kein Praxisprojekt in IT2
- Abgabefrist: 15.12.2025
- Auswahl von Aufgabe 2.1a “Erstellung eines JSON-Schemas zur Validierung vorgegebener Beispieldaten” ODER Aufgabe 2.1b “Konzeption einer Prozessoptimierung”
- Dazu eine Aufgabe zu LZA von Frau Piesche

Kleines Praxisprojekt (optional)

- 50% der Gesamtnote
- 2-3er Gruppen
- Workload: 50 Stunden/Person
- Erstellung eines SKOS-Vokabulars und Publikation mit SkoHub Vocab

Abgaben

- Ideal: Einreichen via E-Mail an adrian.pohl@th-koeln.de mit Link auf
 - Aufgabe 2.1a: git-Repo (GitHub, GitLab, Codeberg)
 - Aufgabe 2.1b: HedgeDoc-Dokument
- Natürlich ist auch Datei-Upload in Moodle möglich.

Fragen?

Zeit	Thema
13:15-13:45	Begrüßung & Organisatorisches
13:45-14:15	Web-APIs
14:15-14:30	 Pause
14:30-15:30	{ “JSON”: “und JSON Schema” }
15:30-16:15	SKOS
16:15-16:30	 Pause
16:30-17:15	IT-Probleme: Gruppenarbeit
17:15-17:30	Feedbackrunde / Ende

Web APIs

Warum das Ganze?

- praktischer Einstieg zum Umgang mit Daten im Web
- Ermächtigung zum Inspizieren von APIs und zur Evaluierung durch sie bereitgestellter Daten
- Kennenlernen der Standardmethode zur Modellierung und Publikation kontrollierter Vokabulare (SKOS)
- Vorbereitung auf die Aufgabe und das kleine Praxisprojekt

Bibliotheken sind Software

Our collections and services are delivered primarily via software. [...]

The choices we make in the development, selection, and implementation of this software [...] define the limits of our content and services. We can only be as good as our software.

— Cody Hanson, *Libraries are Software*

Zum Einstieg

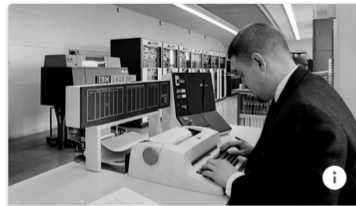
- Was sagt Ihnen der Begriff “API”?
- Sind Sie schon einmal APIs begegnet? Wo und welche?

APIs in der Bibliothekswelt

Europeana

EXPLORE OUR APIS

Our API documentation has moved from Europeana Pro to the Europeana Knowledge Base. Select the specific API that you want to use below to go directly to its documentation page, or [visit the API landing page](#) where you can find all our API documentation, FAQs, and other supporting information.



[Accessing the APIs](#)

Learn how to access each of our APIs once you have an API key.



[Search API](#)

This API provides a way to search for metadata records and media on the Europeana repository.



[Record API](#)

This API takes a Record ID as input and returns all of its metadata and object information modelled in the EDM.



[Entity API](#)

This API allows you to retrieve and search for LOD entities from Europeana's Entity Collection, comprising people, places and mor...



[Annotations API](#)

The Annotations API allows you to create and retrieve user- and machine-created annotations on Europeana objects or media.



[IIIF APIs](#)

These APIs offer an alternative means to group, search for and access information about cultural heritage images.

DDB



Deutsche Digitale Bibliothek REST API develop OAS3

<https://api.deutsche-digitale-bibliothek.de/OpenAPI>

Authors: Patrick Dinger and [Michael Büchner](#)

Description

The API is an [application programming interface](#) that allows users to access data via methods provided by the [Deutsche Digitale Bibliothek \(DDB\)](#). It allows the development of a range of applications that uses material in the DDB, displaying it as they see fit and embedding it in a variety of contexts. The API is available to all users who have registered with the DDB portal, as soon as they have set up an access code ('API key') in their "My DDB" area of the portal.

Users must verify their identity using an API key in order to use the API of the Deutsche Digitale Bibliothek. The key is a unique series of characters that has to be transmitted each time the user sends an enquiry to the API. All registered users of the Deutsche Digitale Bibliothek can obtain their personal authentication key for using the API. Keys can be issued once users are logged into their [DDB account \("My DDB"\)](#) at our portal and used in this interactive API documentation ("Authorize") as well.

Licenses

In general all metadata provided via the API is licensed [CC0](#). For binary data like images, videos etc. DDB has introduced a licensing model that allows participating institutions to select from a bundle of licenses and license notices. If a reuse of this data is allowed must be checked in every case according to the information provided by the metadata.

[Terms of service](#)

[Service Point of the German Digital Library - Website](#)

[Apache 2.0](#)

Servers

<https://api.deutsche-digitale-bibliothek.de/> - Production

[Authorize](#) 

lobid

 **lobid** [resources](#) [organisations](#) [gnd](#) Publikationen ▾ English 👤 ▾

lobid — Dateninfrastruktur für Bibliotheken



lobid stellt Rechercheoberflächen und offene Programmierschnittstellen (APIs) zur Verfügung, die auf Linked Open Data (LOD) basieren. lobid wird vom Hochschulbibliothekszentrum des Landes NRW betrieben.

[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 | ein Dienst des hbz [Gewährleistung](#) | [Impressum](#) | [Datenschutz](#) | [Barrierefreiheit](#) | [Mastodon](#) | [GitHub](#) | [Blog](#)



API Specifications - International Image Interoperability Framework™ — IIIF | International Image Interoperability Framework — Mozilla Firefox

API Specifications - Internal: X +

https://iiif.io/api/index.html

International Image Interoperability Framework

Home Demos News & Events Get Involved Get Started

Get Started, APIs & Documentation

API Specifications – International Image Interoperability Framework™

Current specifications

Feedback requested

Approved extensions

Community translations

Community cookbook

Validators

Older versions

Notes

Process

Current specifications

API	Current Version	Description
Image API	3.0.0	The IIIF Image API specifies a web service that returns an image in response to a standard HTTP(S) request. The URI can specify the region, size, rotation, quality characteristics and format of the requested image.
Presentation API	3.0.0	The IIIF Presentation API provides the information necessary to allow a rich, online viewing environment for compound digital objects to be presented to a human user, often in conjunction with the IIIF Image API.
Authorization Flow API	2.0.0	The IIIF Authorization Flow specification describes a set of workflows for guiding the user through an existing access control system.
Change Discovery API	1.0.0	The IIIF Change Discovery API provides the information needed to discover and make use of IIIF resources.
Content Search API	2.0.0	The IIIF Content Search API specification lays out the interoperability mechanism for performing searches of text annotations associated with an object within the IIIF context.
Content State API	1.0.0	The IIIF Content State API provides a way to refer to a IIIF Presentation API resource, or a part of a resource, in a compact format that can be used to

Reconciliation Service API



W3C Community Group
Draft Report

TABLE OF CONTENTS

- Abstract**
- Status of This Document**
- 1. Introduction**
 - 1.1 Data Matching on the Web
 - 1.2 History of the Reconciliation API
 - 1.3 External Resources
 - 1.4 Versions
 - 1.4.1 0.1
 - 1.4.2 0.2
 - 1.4.3 This draft: 1.0-draft
- 2. Core Concepts**
 - 2.1 Terms Used
 - 2.2 Entities
 - 2.3 Types
 - 2.4 Properties
 - 2.5 Property Values
 - 2.6 Conformance
- 3. Service Definition**
 - 3.1 Service Manifest
 - 3.2 Overview of Possible Routes
 - 3.3 HTTP(S) Access
 - 3.4 Cross-Origin Access
 - 3.5 Error Handling and Rate-limiting
 - 3.6 Authentication
- 4. Match Service**
 - 4.1 Structure of a Reconciliation Query
 - 4.2 Reconciliation Query Responses

Reconciliation Service API

A protocol for data matching on the Web

Draft Community Group Report 11 September 2025

Latest published version:
<https://www.w3.org/community/reports/reconciliation/CG-FINAL-specs-0.2-20230410/>

Latest editor's draft:
<https://reconciliation-api.github.io/specs/1.0-draft/>

Editors:
[Antonin Delpuch](#) 
[Adrian Pohl](#)  (Hochschulbibliothekszentrum NRW)
[Fabian Steeg](#)  (Hochschulbibliothekszentrum NRW)
[Thad Guidry Sr.](#) 
[Osma Suominen](#)  (National Library of Finland)
[Gregory Saumier-Finch](#)  (Culture Creates)
Add Yourself Here!

Feedback:
[GitHub reconciliation-api/specs](#) (pull requests, new issue, open issues)
public-reconciliation@w3.org with subject line [specs] ... message topic ... (archives)

Copyright © 2025 the Contributors to the Reconciliation Service API Specification, published by the [Entity Reconciliation Community Group](#) under the [W3C Community Contributor License Agreement \(CLA\)](#). A human-readable [summary](#) is available.

Abstract

This document describes the reconciliation service API, a protocol edited by the [W3C Entity Reconciliation Community Group](#). It is intended as a comprehensive and definitive specification of this API in its given state. Various aspects of this API need to be improved, as hinted by notes throughout this document.

Status of This Document

This specification was published by the [Entity Reconciliation Community Group](#). It is not a W3C Standard nor is it on the W3C Standards Track. Please note that under the [W3C Community Contributor License Agreement \(CLA\)](#) there is a limited opt-out and other conditions apply. Learn

ReSpec

UNOFFICIAL

35

APIs

Application Programming Interface;
deutsch: Programmierschnittstellen

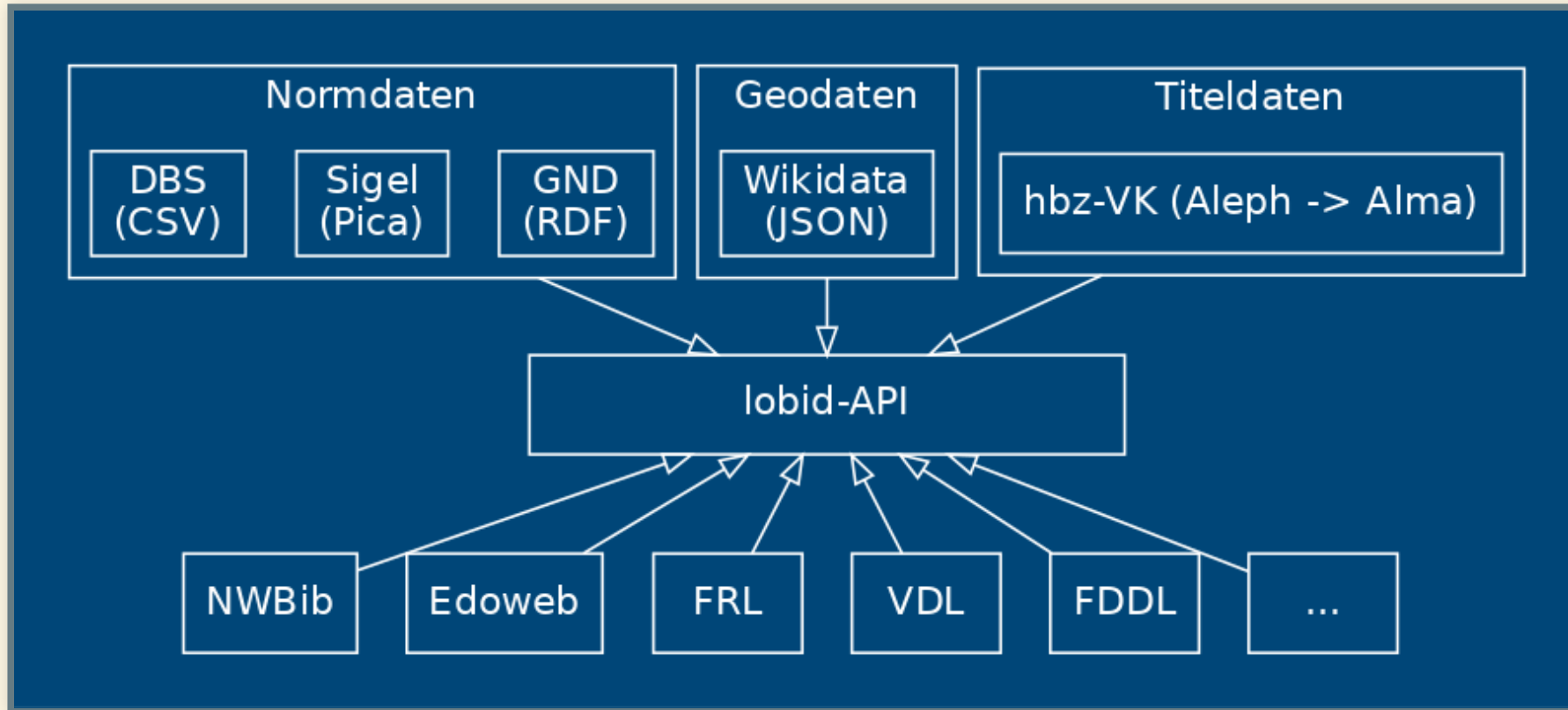
“ein Programmteil, der von einem Softwaresystem
anderen Programmen zur Anbindung an das System
zur Verfügung gestellt wird”

([Wikipedia](#))

APIs

- Software baut auf APIs auf
- APIs machen Softwareentwicklung handhabbar
- APIs ermöglichen Nutzung und Integration
- APIs entkoppeln Anwendungen von Datenquellen, Formaten und Systemen
- Sie ermöglichen modulare, zukunftsfähige Anwendungen

Zum Beispiel lobid-Formate und -Anwendungen



Und wie APIs im Web bereitstellen?

- heutzutage meist **JSON über HTTP**
- De-facto Standard für Daten im Web, siehe z.B. Target, Sinclair (2017): The Rise and Rise of JSON, <https://twobithistory.org/2017/09/21/the-rise-and-rise-of-json.html>

Anwendungsspezifisch oder standardisiert

- APIs können anwendungsspezifisch sein (vgl. [lobid](#))
- oder auf geteilten Standards beruhen (z.B. [IIIF](#), [DAIA-API](#), [Reconciliation Service API](#))
- Standardisierte APIs ermöglichen Nutzung von 3rd-Party-Software (siehe z.B. [Liste von IIIF Viewers, libraries etc.](#), [Liste von Reconciliation Clients](#))

bloße **Lese-APIs** (lassen sich gut mit Suchmaschinentechnologie umsetzen)

vs.

vollständige **Unterstützung von CRUD-**
Operationen: Create, Read, Update, Delete

URIs

```
foo://example.com:8042/over/there?name=ferret#nose
\_/  \_____/\_____/\_____/\_/
 |      |           |           |           |
scheme authority   path       query    fragment
 |
/ \ /_____|\ \
urn:example:animal:ferret:nose
```

Basis von Web-APIs: URIs & HTTP

- Path: Welche API-Endpoints gibt es?
- Parameter: z.B. `q`, `from` und `size`
- Bsp.: <https://lobid.org/gnd/search?q=Köln&filter=type:PlaceOrGeographicName&size=50&from>

Website + API vs. Website = API

- Manchmal ist die API integraler Bestandteil einer Website, manchmal ist sie separat
- Leichtes Umschalten zwischen HTML und JSON kann sehr nützlich sein, sowohl für das Entwicklungsteam als auch für Nutzer*innen
- Je nach Komplexität des Systems und den angebotenen Funktionen können separate APIs sinnvoll oder nötig sein

Beispiel: lobid

<https://lobid.org/resources/search?q=MALIS>

vs.

```
$ curl
```

```
'https://lobid.org/resources/search?  
q=MALIS'
```

oder

[https://lobid.org/resources/search?
q=MALIS&format=json](https://lobid.org/resources/search?q=MALIS&format=json)

Nutzung von Web APIs

API Keys

- Für viele APIs ist eine Authentifizierung nötig
- Diese geschieht meist mit einem API Key
- Die API-Dokumentation sollte beschreiben, wie ein API Key bezogen & angewendet werden kann

API-Dokumentation

Für die Nutzung einer API ist die Dokumentation essentiell, um die Funktionalitäten der API kennenzulernen und zu verstehen, wie sie genutzt werden können.

OpenAPI 1/2

- Standard für die API-Dokumentation
- JSON Schema-kompatibel, kann auch YAML
- Beantwortet Fragen wie:
 - Wie sehen valide Input- oder Output-Daten aus?
 - Welche Pfade unterstützen welche HTTP-Methoden?
 - Welche Parameter werden unterstützt?

OpenAPI 2/2

- OpenAPI unterstützt auch die Entwicklung, die Visualisierung und die direkte softwareseitige Nutzung einer API
- Beispiele: [DDB](#) oder [OERSI-API](#)

Demo

APIs nutzen mit curl & jq

- `curl` ist nützlich für HTTP Requests
- `jq` ist eine Programmiersprache und ein Kommandozeilentool zur Verarbeitung von JSON
- Die Kombination von curl und jq eignet sich gut, um APIs und die darüber angebotenen Daten zu erforschen
- Ermöglicht wird die Kombination der beiden Tools in der Kommandozeile durch “Pipes”

Pipes |

- Pipes (engl. “Rohrleitungen”) leiten den Datenstrom eines Programms an ein anderes weiter
- `programm1 (...) | programm2 (...)`
- `$ curl <URL> | jq -...`
- **Unix-Philosophie**: “Mache nur eine Sache und mache sie gut.”
- curl ist die beste Software zum Übertragen von Daten in Rechnernetzen und jq ist das beliebteste Kommandozeilentool für die Verarbeitung von JSON

Beispiele Open Textbook Library

- die Titel der ersten zehn Ressourcen:

```
$ curl -H "accept: application/json"  
"https://open.umn.edu/opentextbooks/textbooks" | jq  
.data[].title
```

- die Subjects:

```
$ curl -H "accept: application/json"  
"https://open.umn.edu/opentextbooks/textbooks" | jq  
.data[].subjects[].name
```

Übung: Open Textbook API abfragen mit curl & jq

- **Frage:** Wie aktuell sind die neuesten offenen Textbücher?
- **Aufgabe:**
 1. in der API Doku ([hier](#) verlinkt) die URL für die Liste der neuesten Bücher herausfinden
 2. mit curl die JSON-Daten holen und mit jq das Feld mit dem Copyright-Jahr herausfiltern

lobid-resources

```
$ curl "https://lobid.org/resources/search?  
q=MALIS+AND+Köln&size=50" | jq -r  
.member[].contribution[0].agent.label
```

-> Name der 1. beitragenden Person von Titeln mit
“MALIS” und “Köln”

Sortierung und Zählung von Dubletten

- `$ ... | sort | uniq -c | sort -nr`
- `sort`: sortieren nach Alphabet
- `uniq -c`: entferne dublette Strings und zähle (-c, count) das Gesamtvorkommen
- `sort -nr`: numerische Sortierung (n), umgedreht (r, reverse)

PAUSE





Einladung

- Falls Sie regelmäßig Metadaten softwaregestützt verarbeiten, laden wir Sie ein, sich bei metadaten.community zu registrieren und zu beteiligen.
- Nutzt das Forum für eure Zwecke und Themen
 - stellt Fragen, gebt Antworten
 - tragt Veranstaltungen ein
 - nutzt es für eure Arbeitsgruppen
- Siehe auch diese [BiblioCon25-Folien](#)

Zeit	Thema
13:15-13:45	Begrüßung & Organisatorisches
13:45-14:15	Web-APIs
14:15-14:30	 Pause
14:30-15:30	{ “JSON”: “und JSON Schema”}
15:30-16:15	SKOS
16:15-16:30	 Pause
16:30-17:15	IT-Probleme: Gruppenarbeit
17:15-17:30	Feedbackrunde / Ende

JSON

JavaScript Object
Notation

JSON

- Ein einfaches Key-Value-Format für die Speicherung und den Austausch strukturierter Daten
- Key ist immer ein String
- Value ist String, Boolean (true/false), Number, Null, Array, Object
- `{ "key": "value" }`
- Quelle: [RFC 8259](#)

Ein JSON-Dokument

```
{
  "degreeProgramme": "MALIS",
  "module": "IT2",
  "topics": [
    "LZV",
    "Web APIs",
    "Linked Open Data"
  ],
  "instructor": [
    {
      "id": 0,
      "name": "Adrian Pohl",
      "affiliation": "hbz"
    },
    {
```

Das gleiche Dokument

```
{"degreeProgramme": "MALIS", "module": "IT2", "topics": ["LZV", "Web A
```

Einrückungen und Zeilenumbrüche (sog. “pretty print”) sind nur zur besseren Lesbarkeit und werden beim Verarbeiten durch Maschinen weggekürzt.

Übung: JSON-Fehler erkennen

Fehler 1

```
{  
  "Hello": "world"
```

Fehler 2

```
{  
  "degreeProgramme": "MALIS"  
  "module": "IT2"  
}
```

Fehler 3

```
{  
  "degreeProgramme": "MALIS",  
  "module": "IT2"  
}
```

Fehler 4

```
{
  "degreeProgramme": "MALIS",
  "module": "IT2",
  "instructor": [
    {
      "name": "Adrian Pohl",
      "affiliation": "hbz"
    },
    {
      "name": "Petra Maier",
      "affiliation": "VZG"
    }
  ]
}
```

Übung: JSON schreiben

- Öffnen Sie den JSON Editor unter <https://jsonformatter.org/json-editor>
- Schreiben Sie ein JSON-Dokument, das einen String, ein Objekt und ein Array von Strings enthält
- Klicken Sie auf “Format JSON”, um zu sehen, ob das JSON valide ist

JSON Schema

JSON ist nur eine Datenstruktur

- die Verwendung von JSON als Datenformat legt nur die Syntax fest, ansonsten können mit JSON beliebige Informationen repräsentiert werden
- Feldnamen und -inhalte sowie die hierarchische Schachtelung eines JSON-Dokuments können beliebig gewählt werden
- zur Beschreibung eines konkreten JSON-Datenmodells gibt es JSON Schema

JSON Schema und seine Funktionen

- <https://json-schema.org/>
- Datenmodellierung
- Dokumentation
- Validierung von Instanzdaten
- Generierung von Webformularen

Was definiert JSON Schema?

- Feldnamen
- Typen von Feldwerten (`string`, `array`, `object`, `number`, `boolean`, `null`)
- Pflichtfeld: `ja/nein` (`required`)
- Formatierung eines Strings (z.B. `uri`, `date`; oder `pattern` plus Regular Expression)

Ein Beispielschema

URL:

<https://malis.acka47.net/data/beispielschema.json>

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "https://malis.acka47.net/data/beispielschema.json",
  "title": "Beispielressource",
  "description": "Dies ist ein generisches Schema zur Beschreibung",
  "type": "object",
  "properties": {
    "@context": {
      "type": "string",
      "const": "https://schema.org"
    },
    "id": {
      "title": "URL",
      "type": "string",
      "format": "uri"
    }
  }
}
```

Beginnen wir am Anfang

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "https://malis.acka47.net/data/beispielschema.json",
  "title": "Beispielressource",
  "description": "Dies ist ein generisches Schema zur Beschreibung",
  "type": "object",
  "properties": {
    ...
  }
}
```

Beschreibung des Wurzelschemas

- Angabe der JSON-Schema-Version mit `"$schema"`
- Angabe der `$id` (muss nicht notwendigerweise eine URL sein)
- Titel und Beschreibung des Schemas: `title`, `description`
- Jedes JSON-Dokument ist ein Objekt (`{ "key": "value" }`), deshalb `"type": "object"`
- Auflistung der genutzten Feldnamen mit `properties`

Feld mit nicht spezifiziertem String

```
"name": {  
  "title": "Titel",  
  "type": "string"  
}
```

Feld mit URI-formatiertem String

```
"id": {  
  "title": "URL",  
  "type": "string",  
  "format": "uri"  
}
```

Aufzählung möglicher Strings

```
"type": {  
  "title": "Type",  
  "type": "string",  
  "enum": [  
    "3DModel",  
    "AmpStory",  
    "Article",  
    ...  
  ]  
}
```

Objekt als Wert

```
"publisher": {  
  "title": "Publisher",  
  "type": "object",  
  "properties": {  
    ...  
  }  
}
```

Array als Wert

```
"creator": {  
  "title": "Urheber",  
  "type": "array",  
  "items": {  
    "type": "object",  
    "properties": {  
      ...  
    }  
  }  
}
```

Lust auf mehr JSON Schema?

-> [Aufgabe 2.1a](#)

Zu technisch?

-> [Aufgabe 2.1b](#)

Flowchart: <https://pad.gwdg.de/malis-it2-aufgaben1>

SKOS

Basis für kleines Praxisprojekt

- Aufgabe: Publikation eines kontrollierten Vokabulars mit S
- URL:
<https://codeberg.org/acka47/malis/src/branch/main/pages/>



- SKOS: **S**imple **K**nowledge **O**rganization **S**ystem
- Datenmodell für die maschinenlesbare und webbasierte Publikation kontrollierter Vokabulare als Linked Open Data 2009 als **Empfehlung des World Wide Web Consortiums (W3C)** veröffentlicht
- Unterstützt Mehrsprachigkeit, Abbildung von Beziehungen & einiges mehr

Kontrollierte Vokabulare als LOD

- **auflösbare IDs:** ein HTTP URI (Uniform Resource Identifier) für ein Vokabular und jeden Wert
- **Verlinkung:** Werte sind innerhalb eines Vokabulars und zwischen Vokabularen verlinkt
- **Erweiterbarkeit:** weitere Aussagen oder Funktionen auf Basis anderer Webstandards können ergänzt werden

Vorteile

- Webintegration
- Maschinenlesbarkeit
- Interoperabilität
- Nachnutzbarkeit
- Entkoppelung von Datenmodellierung und Softwareentwicklung

Bauen wir uns
gemeinsam ein
kontrolliertes Vokabular
mit **SkoHub Pages**

Beispiel: Hochschulfächersystematik

<https://w3id.org/kim/hochschulfaechersystematik/scheme>

Zwei Typen: Vokabular vs. Deskriptor

- ConceptScheme:
 - das Vokabular selbst, wird genau **1 Mal** pro Vokabular definiert
 - generelle Informationen wie Lizenz, Titel
- Concept:
 - ein Wert des Vokabulars
 - Informationen wie Label, Beziehungen

Beispiel: Concept Scheme

```
<scheme> a skos:ConceptScheme ;  
  dct:title "Destatis-Systematik der Fächergruppen, Studienbereiche  
  dct:alternative "Hochschulfächersystematik"@de ;  
  dct:description "Diese SKOS-Klassifikation basiert auf der Destatis-Systematik  
  dct:issued "2019-12-11" ;  
  dct:publisher <https://oerworldmap.org/resource/urn:uuid:fd062  
  vann:preferredNamespacePrefix "hfs" ;  
  vann:preferredNamespaceUri "https://w3id.org/kim/hochschulfaecher  
  schema:isBasedOn <http://bartoc.org/node/18919> ;  
  skos:hasTopConcept <n1>, <n2>, <n3>, <n4>, <n5>, <n7>, <n8>,
```

RDF auf GitHub

Beispiel: Concepts

```
<n9> a skos:Concept ;  
    skos:prefLabel "Kunst, Kunstwissenschaft"@de, "Art, Art Theory"  
    skos:narrower <n74>, <n75>, <n76>, <n77>, <n78>;  
    skos:notation "9" ;  
    skos:topConceptOf <scheme> .  
  
<n01> a skos:Concept ;  
    skos:prefLabel "Geisteswissenschaften allgemein"@de, "Humanities"  
    skos:narrower <n004>, <n090>, <n302>;  
    skos:broader <n1> ;  
    skos:notation "01" ;  
    skos:inScheme <scheme> .
```

RDF auf GitHub

Wichtige SKOS-Properties

- Verbindung zwischen Vokabular und seinen Werten: `hasTopConcept`, `topConceptOf`, `inScheme`
- Vorzugsbezeichnung: `prefLabel`
- Alternativbezeichnung: `altLabel`
- Informationen zur Hierarchie: `narrower/broader`

Beispielvokabulare

- Kompetenzzentrum Interoperable Daten (KIM):
 - Hochschulfächersystematik
 - Resource Types
 - Schulfächer
- Kerndatensatz Forschung: Interdisziplinäre Forschungsfeldklassifikation
- Wir lernen online:
<https://vocabs.openeduhub.de/>
- ZPID Vocabularies (läuft auf Basis von Skosmos)

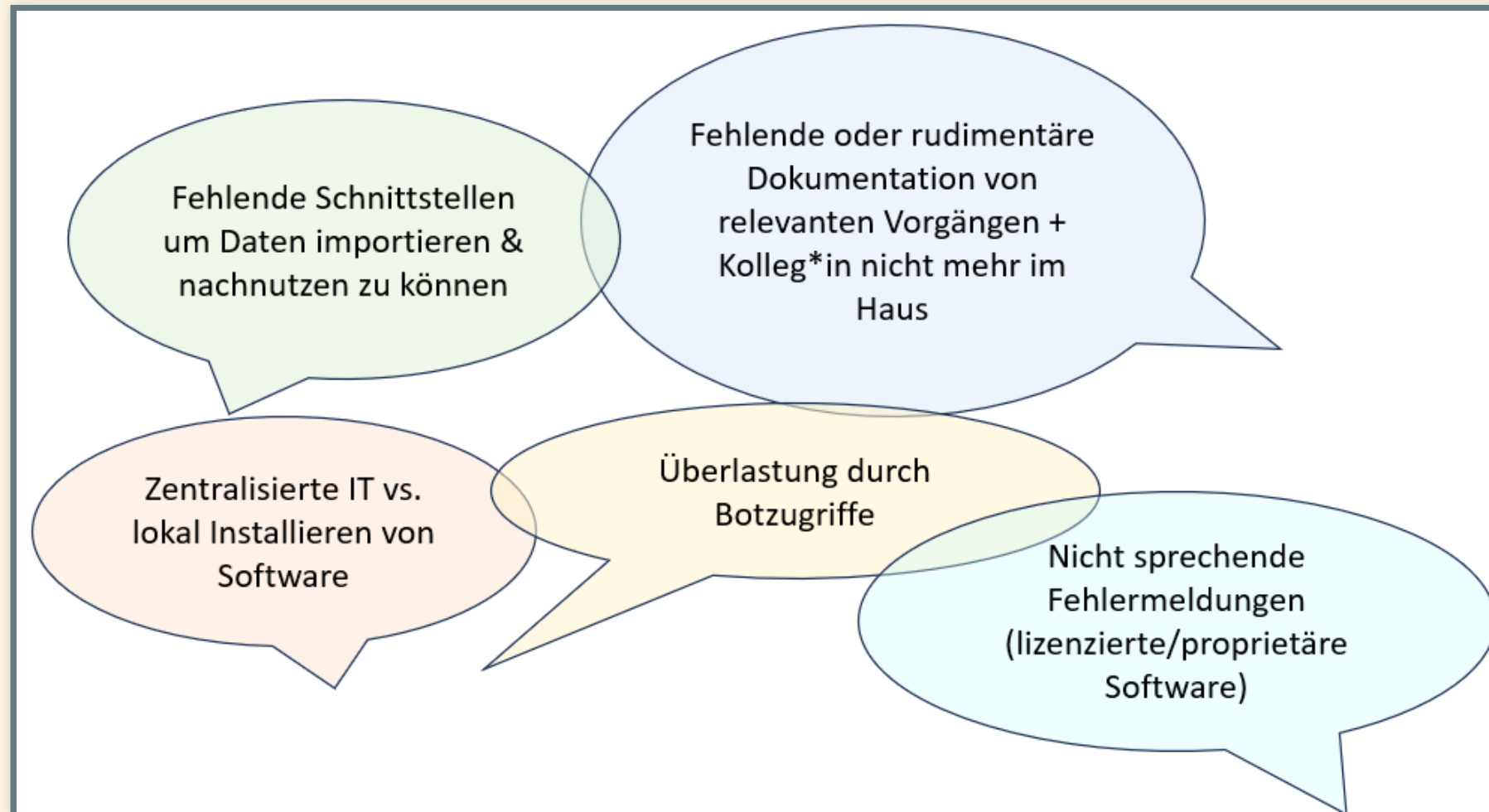
PAUSE



Zeit	Thema
13:15-13:45	Begrüßung & Organisatorisches
13:45-14:15	Web-APIs
14:15-14:30	 Pause
14:30-15:30	{ "JSON": "und JSON Schema" }
15:30-16:15	SKOS
16:15-16:30	 Pause
16:30-17:15	IT-Probleme: Gruppenarbeit
17:15-17:30	Feedbackrunde / Ende

Diskussion: Was
verstehen Sie unter
“Bibliotheks-IT”?
(15 min.)

IT-Herausforderungen



Danke für Ihre Aufmerksamkeit!

Zum Abschluss: One up, one down

→ im **Pad**