Bachelor at Institut für Informatik der Freien Universität Berlin

AG Corporate Semantic Web

Developing a service endpoint to integrate semantic collection data from botanical databases and other information systems

Marcus Ernst

First Reviewer: Prof. Dr. A. Paschke

Second Reviewer: Prof. Dr. C. Benzmüller

Berlin, 03.09.2021

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, den February 27, 2023

Marcus Ernst

Abstract

The digitization of botanical collections has increasingly brought biodiversity research activities online. In order to make these data usable in the most efficient way, various obstacles have to be overcome. One such obstacle is a lack of ability to integrate information from other sources. While agreed upon, machine-understandable data standards such as ABCD have resulted in concepts that can already be described semantically, yet they are often transmitted as free-text information. The utilization of identifiers for collectors has created opportunities for the integration of data from external information systems. However, since the identifiers used are not standardized and vary from institution to institution, this work aims to develop a web service demonstrating that this problem can be overcome by applying appropriate Linked Data methods on centralized knowledge bases such as Wikidata. After eliciting requirements from participating CETAF institutions, an API was designed and implemented on this basis that can integrate biographic, bibliographic, and collection data into a single semantic file format by leveraging multiple endpoints. Thus, the work shows that diverse identifiers used in collection databases do not have to be a problem. Moreover, missing IDs for important information sources such as Wikidata can be found and used. Heterogeneous data from different sources can be merged using previously defined mappings, although such data may not be available in semantic formats. Further sources of information could thus be added in the future. Furthermore, a future focus on annotated geographic identifiers is also conceivable to additionally integrate semantic data on collection object found locations.

Zusammenfassung

Die Digitalisierung von botanischen Sammlungen hat Forschungsaktivitäten zur Biodiversität immer mehr ins Netz geführt. Um diese Daten möglichst effizient nutzbar zu machen, müssen verschiedene Hindernisse überwunden werden. Eines dieser Hindernisse sind fehlende Integrationsmöglichkeit von Informationen aus anderen Quellen. Während vereinbarte, maschinenverständliche Datenstandards wie ABCD dazu geführt haben, dass Konzepte bereits semantisch beschrieben werden können, werden diese jedoch häufig als Freitextinformationen übertragen. Die Verwendung von Identifikatoren für Sammler*innen hat die Möglichkeiten für die Integration von Daten aus externen Informationssystemen geschaffen. Da aber die verwendeten Identifikatoren nicht standardisiert sind und sich von Institution zu Institution unterscheiden, soll der in dieser Arbeit entwickelte Web Service zeigen, dass dieses Problem überwunden werden kann, indem geeignete Linked-Data-Methoden auf zentralen Wissensdatenbanken wie Wikidata angewendet werden. Nach der Erhebung von Anforderungen mit beteiligten CETAF-Institutionen, ist auf dieser Basis eine API entworfen und implementiert wurden, die sowohl biografische und bibliografische als auch Sammlungsdaten über die Verwendung diverser Endpunkte in eine einzige Datei eines semantischen Formates integrieren kann. Die Arbeit zeigt, dass die in den Sammlungsdatenbanken verwendeten unterschiedlichen Identifikatoren kein Problem darstellen müssen und auch fehlende IDs für wichtige Informationsquellen wie Wikidata gefunden und verwendet werden können. Heterogene Daten aus verschiedenen Quellen können über zuvor definierte Mappings zusammengeführt werden, auch wenn diese nicht in semantischen Formaten verfügbar sind. Weitere Informationsquellen könnten somit in Zukunft hinzugefügt werden. Außerdem ist auch denkbar, annotierte geographische Identifikatoren zu verwenden, um zusätzlich semantische Daten über Fundorte von Sammlungsobjekten zu integrieren.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Outline 2

2 Background **3**

2.1 Biodiversity Informatics 3

2.2 Semantic Web 6

2.2.1 Semantic Web 6

2.2.2 Ontology 7

2.2.3 XML 8

2.2.4 RDF 8

2.2.5 SPARQL 9

2.2.6 JSON-LD 10

2.3 RESTful Web Services 12

3 Requirements **15**

3.1 Use cases 15

3.2 Functional requirements 16

3.3 Non-functional requirements 17

4 Design **19**

4.1 Information systems 19

4.2 REST and alternatives 20

4.3 Identifiers 21

4.4 Resources and endpoints 22

4.5 Architectural scheme 23

5 Implementation **25**

5.1 Technologies 25

5.1.1 Slim Framework 25

5.1.2 JsonLD 25

5.1.3 EasyRDF . . . 25
5.2 Components . . . 26
5.3 SPARQL Queries . . . 27
5.4 Mappings . . . 28
5.4.1 Wikidata properties . . . 28
5.4.2 BHL responses . . . 29
5.4.3 Europeana responses . . . 31
5.5 Pagination . . . 31
5.6 Versioning . . . 31
5.7 Documentation . . . 32

6 Evaluation **33**
6.1 Functional Requirements . . . 33
6.2 Non-functional requirements . . . 33
6.3 Performance testing . . . 34

7 Summary **35**
7.1 Discussion . . . 35
7.2 Conclusion . . . 37

Bibliography **39**

List of Figures

2.1 Example of a specimen representation in ABCD 4

4.1 External and internal structure of the API 24

5.1 Operation example: Bibliography call by using Wikidata ID . . 26
5.2 API Documentation using Swagger 32

6.1 Test diagram 34

List of Tables

3.1 Functional Requirements 16
3.2 Non-functional Requirements 17

4.1 Endpoints and methods 23

6.1 Evaluation of functional requirements 33
6.2 Evaluation of non-functional requirements 33

1 Introduction

1.1 Motivation

In recent years, institutions, such as museums holding botanical collection databases have started publishing their collection object data as Linked Open Data (LOD) and enriching it with links to semantic resources.[1][2] For instance, botanical collection records in herbaria contain links to the Wikidata page of the collector who gathered that plant in the field. Likewise, the institution itself is linked to this collection object via an institution predicate, as well as to other collection objects. Together, these objects and links are forming a graph containing statements about resources.
An example of such a graph exists in a triple store at the Botanic Garden and Botanical Museum Berlin (BGBM) of the Freie Universität.[3] The store holds data as subject-predicate-object statements and can be queried via a SPARQL endpoint. The web-based demonstrator "CETAF Botany Pilot" was co-ordinate as a demonstrator by BGBM, combining semantic collection data from the triple store as well as information from other distributed resources such as Wikidata or virtual library systems. This information is displayed in a dynamic HTML page which is linked to the collection information system of the BGBM.[4]
However, for integrating this data at other institutes, it has been necessary to specify and develop a generic service that makes it possible to give access to the data mentioned above in the Linked Data format JSON-LD. Designed as close as possible to the REST architectural style, this API transforms client requests for collector data into SPARQL queries to the mentioned triple store and Wikidata and into various requests to other information systems such as Biodiversity Heritage Library (BHL) or Europeana. Since different identifiers for the collector might be used in the requests, the API must find the correct collector IDs for each information source via Linked Data techniques. All data obtained are then integrated into a JSON-LD file and returned to the client. If the data exceeds a predefined threshold, it can also be returned paginated.

However, the creation of new resources or the deletion of existing resources, e.g., in the triple store, has not been implemented in this project. The specification of the service is based on a requirements analysis, created in cooperation with selected European collections (Plantentuin Meise[5] and Botanical Garden Berlin).

The source code is available in Gitlab[1], the API can be tested in its Swagger documentation[2].

1.2 Outline

First, the subject-specific background from biodiversity informatics will be described. Additionally, the evolution of the biodiversity research field will be outlined in order to understand how the web service can provide support for research activities (chapter 2.1). As the web service uses technologies from the Semantic Web world, the idea, the fundamental concepts, and the standards of Semantic Web will be described in chapter 2.2. Some technical basics about the REST architectural style will be provided and, additionally, reasons stated for choosing this style by comparing it to its alternatives (chapter 2.3). Subsequently, the functional and non-functional requirements for the tool are extracted by describing several use cases that emerged from the discussions with the collection institutions mentioned above (chapter 3). Coming from that the design of the API will be sketched and the rationale behind some design decisions elucidated (chapter 4). Following this, light will be shed on the implementation of the API by briefly describing the technologies used and explaining how information from different sources are brought into a knowledge graph (chapter 5).
An Evaluation based on the previously identified requirements will be carried out to see whether the expectations could be met (chapter 6). The last step is to draw a conclusion. Problems encountered will be discussed and an outlook will be given as to the possible direction of further development of the web service (chapter 7).

[1] `https://git.bgbm.org/mernst/bgbmapi_bsc`
[2] `https://api.bgbm.org/swagger/?url=https://api.bgbm.org/collector/v1/swagger.json`

2 Background

2.1 Biodiversity Informatics

> "Over the last two decades, curators in Natural History Museums as well as those responsible for ecological mapping or monitoring projects started databasing their collection and observation information. [...] In view of accelerating environmental changes, loss of biodiversity, and a pressing need for fast decisions in environmental politics, this information base must be utilized to the greatest possible extent."[6, p.5]

For decades, changes in biodiversity have been discussed and studied by various scientific disciplines. Among them there are traditional disciplines such as botany as well as comparatively new disciplines such as biodiversity informatics. But what does biodiversity mean?
The United Nations Biodiversity Convention defines biodiversity as "the variability among living organisms from all sources [...] and the ecological complexes of which they are part[...]".[7] Numerous facets of this diversity can be measured, e.g., to describe the quantity of different species worldwide or at a specific place.[8] In addition to this spatial dimension, the temporal dimension of biodiversity can be studied as well since changes in biodiversity can be observed starting from historical records. Biodiversity informatics utilizes specimens as a link between past and present. Thus, it is possible to develop comprehensive hypotheses, e.g., about entire ecosystems.[9] Moreover, a single specimen record can be represented as a vector in a coordinate system spanned by of a spatial and a time dimension.
Based on the scientific findings on biodiversity, political decisions on the use and management of resources and ecosystems can emerge. Since accurate information has often not been available in the past, sustainability criteria have been neglected in policy-making for a long time. The availability of comprehensive data on the globe's species and ecosystems not only supports researchers in conducting further research activities, but also decision-makers in imple-

menting more sustainable policies.[10]
Approximately 3 billion specimens are held in natural history and botanical collections.[11] To answer contemporary research questions about biodiversity, it has been recognized that the digitization of these collections can be of vital importance. As of August 01, 2021, there are more than 200 million digitized specimens from numerous herbaria in the data of Global Biodiversity Information Facility (GBIF), an international data infrastructure for biodiversity research.[12] This and other networks, as well as organizations provide various portals for research activities to access these data. However, the lack of services for analysis and publication, and especially a lack of integration of data from different collections, can be major obstacles in scientific workflows.
In a first step common data standards had to be agreed upon before sharing data from different herbaria. Formats like "Access to Biological Collection Data" (ABCD)[13] and "DarwinCore" (DwC)[14] are among such standards that should facilitate the exchange of collection data. For example, ABCD describes a physical specimen by representing its data elements from collection and observation data such as taxa, geographic information or the collector.[15]
In order to identify a specimen, the members of the Consortium of European

Figure 2.1: Example of a specimen representation in ABCD

Taxonomic Facilities (CETAF)[16] have agreed on the introduction of a sys-

tem of HTTP URI-based stable identifiers that allows the identification of the specimen as a data body fully independent of name changes. The system in its entirety follows Linked Open Data (LOD) principles and aims at introducing redirection mechanisms to both human-readable and machine-processable representations. As a result, collection entries can be used much more easily for data-driven studies, as linking data can be simplified, time is reduced, and workflows can be automated. In addition, this system provides the basis for semantic inference.[1]

Although standards such as ABCD can describe the concepts semantically, the data is often still transferred in the form of free-text information. Due to the different naming of a collector, assignments of specimens to the person are complicated. The ambiguity of such free text information can become a major problem: For example, there may be multiple spellings for a particular collector name, or a collector may change his/her name during his/her lifetime. Moreover, it is possible that two collectors have the same name.

In fact, behind this collector, a great treasure of further information might be hidden, since this collector can be considered as a separate entity with its meta-information, similar to a specimen. Unlike changes to taxa, which can be split, for example, the metadata about a collector is not affected by name changes - thus, it can be considered as the most stable data objects and can be easily annotated accordingly.[17] This has already been done by numerous CETAF institutions, providing the impetus for the development of the "Botany Pilot" demonstrator at BGBM to integrate both cross-collection data and data from external information systems into one representation.[4]

According to Berendsohn, one of the key tasks of biodiversity informatics is the development of web services that are able to exchange and integrate data in order to achieve an increasing interoperability of data across collections.[15] This allows following up on further interesting research questions: In which regions did the collector focus his work? What specimens did the collector publish about? Which collections contain a considerable number of the collector's objects?

The web service developed in this project will contribute to the investigation of these research questions by providing collector-centered linked data.

2.2 Semantic Web

> "The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation." [18, p.37]

The World Wide Web largely consists of unstructured information presented in a human-understandable form, e.g., HTML web pages. The fact that people can nevertheless access the information is due to optimized search engines, that display search results sorted according to various criteria such as relevance. These serve as an entry point to the information people are looking for. However, search engines suffer from various problems that cannot easily be resolved without structuring the information itself. For example, they return a large number of results quickly, but these can be quite inaccurate.[19]
Another problem is the lack of recognition of semantic similarities, i.e., a search for literature on a certain plant species does not yield rich results, although this species has been published on in the last decades. This might happen if that species was searched under an obsolete name - the semantic "synonym" relationship between new and old name is unknown to the machine.
Thus, in recent years, methods have been used to extract the meaning of terms in statements. One approach is Natural Language Processing, which uses techniques from computational linguistics as well as artificial intelligence to separate statements written in natural language in order to understand the meaning of the individual parts.[20] Another approach is the Semantic Web, which will be described below.

2.2.1 Semantic Web

The Semantic Web forms an extension of the existing web. The data in existing documents is to be represented in a machine-readable way from the beginning and annotated by meta information (information about data). Tim Berners-Lee can be considered as the central figure of the Semantic Web. His goal was to make the semantics of information more heavily weighted. According to Berners-Lee, the Semantic Web consists of three basic components:

1. structuring of information (e.g., by XML)
2. expression of meaning (e.g., by RDF)

3. ontologies (e.g., by RDFS or OWL)

Additional building blocks include cross-ontology logics and semantic agents.[18] The W3C tries to promote this idea by introducing and maintaining standards such as RDF. Such standards are intended to be as open and flexible as possible to ensure interoperability in the heterogeneous information space of the Web.[21] Interoperability is reinforced by Linked Data: Linked Data helps to achieve the vision of the Web of Data by linking the different data sources to each other. Whenever a URI is available as information, it shall be used as a name rather than a literal. Whenever an HTTP URI is available, it shall be used since such addresses are more easily accessible to users. Whenever machine-readable standards, such as RDF or SPARQL, are available, they should be used. Whenever data is related to other sources, they should be connected by integrating links to them and making their semantic relationship explicit.[22] In this way, numerous knowledge graphs and ontologies can be linked and build a global knowledge graph.

2.2.2 Ontology

Ontology comes from the Greek “ontologia”, which means ”essence of being”. In computer science, the term was then adopted to describe a model that defines a system of concepts and objects in a particular domain through relationships.[23]
An ontology consists of a list of concepts, which represents classes of objects, and their interrelationships. Relationships specify, for example, whether one term is a subclass of another one or how one class relates to another as a property. Restriction rules specify the ranges and domains of properties. Additionally, there are further specifications for classes and rules that allow logical reasoning within the ontology.
To organize the knowledge of a domain, ontologies specify rules that the WWW lacks. Thus, a whole range of problems can be solved, and two examples shall be mentioned here:

- Terminological differences of synonyms can be resolved if these two terms themselves have been explicated as synonyms by an attribute or a relation.

- The contextual meaning of a homonym, i.e., a term that can contain multiple meanings, can be resolved into a shared vocabulary by appropriate

mapping.

In the last two decades, several standards have been introduced that can be used to model and process ontologies. These include RDF, RDFS and OWL. Also, various serializations have been developed for the representation of RDF. In addition, SPARQL, a language for pattern search in ontologies, has been developed.

2.2.3 XML

The extensible markup language (XML) is a meta-language that, strictly speaking, is not a specification of the Semantic Web, but was developed primarily to represent syntactically structured information.[24] Thus, it forms the basis of the RDF/XML specification. XML is intended to separate the content of a document from its formatting, but similar to HTML it works with tags to denote elements.
Furthermore, nesting is used to represent hierarchical relationships among elements, creating a machine-processable tree structure. Unlike HTML, XML does not have fixed tags or constraints. These must first be defined in XML schemas or vocabularies. There are several query languages for XML documents, e.g., XPath, which make it possible to search the XML tree and extract information.
As mentioned above, XML is originally not designed to make the meaning of the information machine-understandable - the meaning of the tags depends on the application that interprets the document. This results from the universal interchangeability of the XML format. However, the goal of the Semantic Web is universal validity of the meaning of statements. Therefore, the syntactic interoperability of XML in RDF/XML has been extended by semantic interoperability.[19]

2.2.4 RDF

Resource Description Framework (RDF) is a standard for describing structured information that was originally intended to encode metadata about individual web resources. It allows semantic statements to be constructed that are uniformly valid and thus interchangeable without meaning change. In addition, this standard provides a data model that represents metadata and allows semantic processing of this data. In RDF, a statement basically consists of an

object-attribute-value triple or, more generalized, an object-predicate-subject triple. Databases that store a set of these triples are usually referred to as triple stores. These triples can be represented in XML format (RDF/XML), but also in other serializations such as Turtle or JSON-LD. An ontology can be modeled in RDF Schema (RDFS) for lightweight variants and/or in Ontology Web Language (OWL). The latter allows modeling of more expressive ontologies with more complex relationships between classes. Besides classes, properties are also described, their relationship to each other defined, and ranges and domains specified. Therefore, relationships get an application-independent meaning.
Objects of a statement are resources that are denoted with a Uniform Resource Identifier (URI). This is often a URL but can also be any other identifier. URIs also identify properties, which themselves define the relationship between resources. In turn, subjects can be a resource or a literal. The latter are atomic values such as strings, numbers, or dates.
URIs allow unique, unambiguous identifiers of the things of the Semantic Web. Relationships between things can also be non-hierarchical, whereas in XML they always have a hierarchical character due to embeddings. In general, a set of triples can be described as a graph with directed edges, unlike XML which spans a tree. This makes it easier to unify the knowledge graphs of two resources.

2.2.5 SPARQL

The W3C Recommendation SPARQL Protocol And RDF Query Language (SPARQL) is a query language that searches RDF graphs for specific patterns. Essentially, its syntax is based on Turtle. SPARQL additionally provides the feature to make queries and results more readable by using namespaces. As in SQL, SELECT, FROM, and WHERE are components of the queries. SELECT performs the projection function of the query to specify return variables, FROM is an optional source specification and in WHERE the constraints and variable names are determined. Such constraints can be graph patterns or FILTER constraints with various operators (comparative, special, arithmetic, boolean). Variable names can be used for objects and subjects as well as for predicates.
The output format is determined by the keyword that starts the query: If

SELECT is given, then the result is tabular. If the keyword CONSTRUCT is used, then the result is an RDF result graph. Besides that, queries can also start with two other keywords: ASK returns a Boolean value depending on whether the pattern is matched. In cases where the properties of resources are unknown, DESCRIBE is a suitable keyword. Hence, only the resource to be described must be specified in the query, making the result graph contain all statements found about the resource.

2.2.6 JSON-LD

> ”JSON has become the lingua franca of data on the web. It’s a simple way to represent data that works well with client-side code in web browsers.”[25, p.3]

Javascript Object Notation - Linked Data (JSON-LD, current version: 1.1) is an RDF serialization that semantically extends the JSON format, making it suitable for web services and data exchange due to its wide use. JSON uses key-value pairs to represent data, separated by a colon and each enclosed in quotes. Objects and their associated properties are each enclosed in curly braces. Apart from literals, resource URIs are used in JSON-LD to denote keys and values instead of simple string labels. A resource is described by summarizing its properties and the objects of the statements separated by commas. Since JSON-LD is an instance of RDF syntax, any RDF graph can be serialized into a JSON-LD document. Conversely, however, not all JSON-LD documents can be interpreted as RDF, since JSON-LD-specific differences exist such as the use of blank nodes for properties. A special feature of JSON-LD is the denotation of a vocabulary ”@context”. A vocabulary can either be described within the same file or it is referenced to a web resource that provides an ontology. More syntactic structures can be looked up in the W3C specification of the JSON-LD standard.[26]

```
{
  "@graph" : [ {
    "@id" : "http://w.jacq.org/object/W19810012214",
    "description" : "A herbarium specimen of Potamogeton pectinatus L.
        collected by Chase,A.",
    "publisher" : "W",
    "title" : "Potamogeton pectinatus L.",
    "type" : "PreservedSpecimen",
```

```
    "dwciri:recordedBy" : {
      "@id" : "http://www.wikidata.org/entity/Q3822242"
    },
    "country" : "USA",
    "eventDate" : "1900-09-03",
    "family" : "Potamogetonaceae",
    "genus" : "Potamogeton",
    "institutionID" : "https://ror.org/01tv5y993",
    "locality" : "S.W. end of Wolflake. Indiana",
    "recordedBy" : "Chase,A."
  } ],
  "@context" : {
    "recordedBy" : {
      "@id" : "http://rs.tdwg.org/dwc/terms/recordedBy"
    },
    "description" : {
      "@id" : "http://purl.org/dc/terms/description"
    },
    "country" : {
      "@id" : "http://rs.tdwg.org/dwc/terms/country"
    },
    "type" : {
      "@id" : "http://purl.org/dc/terms/type"
    },
    "family" : {
      "@id" : "http://rs.tdwg.org/dwc/terms/family"
    },
    "publisher" : {
      "@id" : "http://purl.org/dc/terms/publisher"
    },
    "title" : {
      "@id" : "http://purl.org/dc/terms/title"
    },
    "locality" : {
      "@id" : "http://rs.tdwg.org/dwc/terms/locality"
    },
    "genus" : {
      "@id" : "http://rs.tdwg.org/dwc/terms/genus"
    },
    "eventDate" : {
      "@id" : "http://rs.tdwg.org/dwc/terms/eventDate"
    },
    "institutionID" : {
```

```
51        "@id" : "http://rs.tdwg.org/dwc/terms/institutionID",
52        "@type" : "@id"
53      },
54      "dwciri" : "http://rs.tdwg.org/dwc/iri/",
55      "owl" : "http://www.w3.org/2002/07/owl#"
```

Listing 2.1: JSON-LD: Beispiel eines (gekürzten) Specimen-Eintrages aus dem CETAF Triple Store

2.3 RESTful Web Services

> "A Web service is a software system designed to support interoperable machine-to-machine interaction over a network."[27]

This W3C definition describes the general function of a Web service. Depending on the type of a Web service, the specific definition may vary to some extent. Further, Web services can be divided into REST-compliant services, which provides a fixed set of stateless operations, and other services with arbitrary additional operations. Important representatives of Web services are Remote Procedure Calls (RPC), Simple Object Access Protocol (SOAP) and Representation State Transfer (REST).[27] Previously involved in the development of Hypertext Transfer Protocol standards, Fielding developed REST in 2000 in his dissertation.[28]
REST is an architectural style defining only principles and constraints for designing an interface - the implementation of the design, however, is up to the developer. In his dissertation, Fielding derives this style from an arbitrary set of architectures that are unconstrained ("null style"). Then, he defines further architectural constraints that determine the design of the components and the conditions between them, further restricting the set of possible architectures until they satisfy the predefined REST constraints. Below, these constraints will be described in more detail.[28]

1. Server-Client: This constraint is based on the separation of concerns principle. While the client only sends requests to the server, the server decides whether the request is accepted and executed or whether it is rejected. Thus, both components are functionally separated and can be maintained and developed independently.

2. Statelessness: The server or the API respectively does not store any context information of the requests, i.e., each request must contain all information necessary for processing. On the one hand, this leads to redundant requests with complete request information increasing the network load, but on the other hand, it also increases reliability if there is incorrect information about past requests on the server side.

3. Cacheability: In the REST context, cacheability means that all responses from the server to the client are implicitly or explicitly marked as cacheable or non-cacheable. Therefore, the client is made able to cache data with the information it receives. Comparing this constraint with the statelessness constraint, the possible advantages and disadvantages are mirrored here: If the client first checks whether the data to be queried is already available in the local cache, this can save network traffic. On the other hand, however, this can affect reliability, since the cached data can be an outdated image of the current server-side data.

4. Layered System: A system of hierarchical layers means that each layer only knows the respective neighboring layers. Since each layer is assigned its own area of responsibility, this layer architecture also fulfills the principle of separation of concerns. The advantage of this is easier maintainability as the consequences of modifying a single layer remain manageable. An example of separate tasks is “authentication - caching - data access”.

5. Code on Demand: This constraint is intended to allow the download and client-side execution of code to extend its functionality. However, Fielding points out that this constraint is optional.

6. Uniform Interface: This constraint is emphasized by Fielding as the key difference between interfaces according to REST and those of other architectural styles. In his thesis, it is noted that the introduction of uniform interfaces reduces functionality, since data can only be transferred in a standardized way. However, since big data should be sent as efficiently as possible, specialization is not of great relevance. Strictly speaking, this constraint consists of four sub-constraints: Identification of resources, manipulation of resources by representations, self-descriptive messages, and hypermedia.

3 Requirements

Prior to this project, partner institutions such as Plantentuin Meise were contacted to elicit requirements. This information was first used to develop use cases. Subsequently, the requirements were specified, for which it was necessary to consider the different perspectives, e.g., developers and users.

3.1 Use cases

1. A web portal (Meise) provides a web page on occurrences of specimens. A portal's user wishes to learn more about the collector and wants biographical information to be displayed in a dedicated profile box. Once activated by the user, the web page sends a request to the web service containing the collector ID associated with the specimen. The web page presents the biographical information on the fly in an info box in a human-readable form.

2. A researcher uses a collector's VIAF ID to search for both the specimens associated with that person and the related literature to find out which collected species or taxa that collector has published on. Using this ID, the API finds all specimens from the herbaria of various institutions, as well as bibliographic information from BHL and Europeana and provides it in a JSON-LD file.

3. As part of the Synthesis+ Specimen Data Refinery project, it is planned to enrich image object data with identifiers. The images are scanned via OCR and the recognized names of the collectors are linked to person IDs. The IDs are then linked to the associated person metadata by requesting the web service developed in this work. The API is expected to give access to the data easily and without using dedicated SPARQL queries. The client can choose between JSON-LD or RDF/XML as output format.

3.2 Functional requirements

Requirement ID	Title	Description	Use Case
FR1	Providing collection data from a requested collector	The API provides a document containing all specimens and associated metadata from the triple store	(2)
FR2	Providing biographical data on a searched collector	The API provides a document containing all biographical data about the person	(1)
FR3	Providing bibliographic data of a collector	The API provides a document that contains all the bibliographic data about the person.	(2)
FR4	Provide an overall summary of a collector	The information resulting from the first three requirements in a compacted JSON-LD file.	(3)
FR5	Finding all alternative URIs of a collector	The API must be able to use Linked Data methods to find all existing alternative URIs.	(1), (2), (3)
FR6	Pagination	The number of collection objects can be very high, slowing down the API significantly.	(2)
FR7	REST constraint compliance	An easy-to-access API without the necessity of sending SPARQL queries is required	(1), (2), (3)

Table 3.1: Functional Requirements

3.3 Non-functional requirements

Requirement ID	Title	Description	Use Case
NFR1	Versioning	To ensure a stable API as well as easy further development of the API, versioning is used.	-
NFR2	Documentation	Swagger is to be used to create the documentation. All endpoints and available methods are to be publicly documented here.	-

Table 3.2: Non-functional Requirements

4 Design

A design of the Web service is to be developed from the requirements defined in Chapter 3. Choosing REST as the applied architectural style emerged from the elicited requirements which is outlined in section 4.2. The design serves as a blueprint for the implementation of the API. The API is located between the clients and the data sources and consists of three components which is shown in Section 4.5. A specification of the client side is not done in this thesis. The API provides an endpoint to access various sub-resources. As each sub-resource satisfies one of the requirements (1-3), they differ in how they need to access specific information systems. However, before the endpoint and resources are described in more detail in this chapter, the requested information systems are to be described first.

4.1 Information systems

1. CETAF Triple Store: The CETAF Triple Store, hosted at the BGBM, integrates collection data from many partner institutions into a unique database. The focus is on the collected specimens which exceed 10 million. Each of these specimens represents a unique resource containing numerous meta-information such as collector, scientific name, institution or locality in the form of RDF triples. Collector IDs are resources and can be linked to another collector ID via a "sameAs" property. The data is provided via an open-access SPARQL endpoint that provides requested data in non-semantic formats such as JSON or CSV, or in semantic formats such as RDF/XML or JSON-LD.

2. Wikidata: Wikidata is a free knowledge database containing information about almost 95 million data objects. Statements about these resources are stored as RDF statements that can be edited by anyone in the user interface. Since Wikidata is an international project, multilingual information is also provided, e.g., about the description of the object or the name spelling. A very important feature regarding this project is the fact

that Wikidata contains further identifiers and links to other information systems, including the ones described below. Its data is made available via a publicly available SPARQL endpoint providing the data as XML or JSON.[29]

3. Biodiversity Heritage Library (BHL): BHL is a digital library that provides free access to biodiversity literature from the 15th to 21st centuries from around the world. The collection totals over 59 million pages and is operated by the Smithsonian Libraries and Archives, which aims to digitize the natural history literature held by its members to promote research activities in the biodiversity community. The BHL API provides its literature data as XML or JSON files via various REST-like web services, requiring possession of an API key for access.[30]

4. Virtual Authority File (VIAF): VIAF is a service that merges authority files from various globally distributed libraries into a single virtual authority file and is operated by the Online Computer Library Center (OCLC). Authority files are directories of standardized terms such as persons and works. As entities, these terms have their individual URIs and are linked to each other. In this project, the information about the collector is of interest, e.g. international names and identifiers of other information systems. The norm files on the collectors can be accessed openly as RDF/XML or as JSON-LD documents.[31]

5. Europeana Europeana is an EU project to create a virtual library aiming to make cultural and scientific works freely accessible for educational and research purposes. It contains more than 50 million objects. The data has been structured as an ontological data model, the European Data Model (EDM), following the principle of the Semantic Web. Access to the information is given via various APIs (REST, Search, SPARQL, etc.) and is accordingly also offered as a JSON-LD file. An API key is required to use the API.[32]

4.2 REST and alternatives

Apart from the REST architectural style, there is also the Simple Object Access Protocol (SOAP) as a protocol for message transmission. The W3C recommendation was developed from the RPC variant XML-RPC and uses XML

as the format for its messages. SOAP provides features for authenticated transmission and restricted access to remote system methods.[33] The latter aspect makes SOAP a good choice for transmissions where operations beyond CRUD are needed. However, this increases the scope of the implementation.
The requirements elicitation revealed the necessity of merely a small set of available methods. However, SOAP is a protocol designed to implement more complex methods. Retrieval methods implemented with REST based on HTTP GET methods are sufficient to meet the requirements. Thus, the validation overhead for the XML messages used in SOAP is avoided as well. Likewise, authentication mechanisms are not required since the API is available to any client and no sensitive data is used. As the REST constraints allow a flexible implementation, the API can be easily extended following this prototype.

4.3 Identifiers

Since collections use different URIs to annotate collectors, e.g., Meise mainly annotates their data with VIAF IDs in use case 1 (see 3.1), the API must be able to provide collector data although the Wikidata ID is unknown. Besides Wikidata IDs, the following identifiers may occur:

1. VIAF-IDs, e.g., 44697854
2. ORCiD-IDs, e.g., 0000-0001-9945-7606
3. Harvard-IDs, e.g., 379

Regular expressions can be used to recognize these identifiers. An equivalent Wikidata ID can then be found using the following two mechanisms:

1. In the triple store, alternative identifiers for a collector are queried via SPARQL and referenced with the http://www.w3.org/2002/07/owl#sameAs property. If a Wiki ID has been found, the API continues. If not, then the API is supposed to use the next method.
2. In Wikidata, an entity containing the specified ID is searched for via its SPARQL endpoint. Beforehand, the given ID must be associated with the corresponding provider and thus with the matching Wikidata property. If a person with this ID exists, then the API continues. If no person with this ID was found, then, depending on the requested

resource, the API returns either all found specimens or an file containing an empty graph.

At this point, it becomes clear how important Wikidata is as a central source for person-related data. All identifiers to the previously described information systems in section 4.1 are obtained from the SPARQL queries to the Wikidata endpoint.

4.4 Resources and endpoints

The design of the resources is based on the requirements developed from the use cases (see 3.1). Crucial is the REST-typical separation of the representations from resources, i.e., file extensions are not necessary to be used in the URL, since the requested content format is negotiated via the information from the request header.[34] Self-descriptive nouns are used for both the collector resource and the listing of sub-resources such as the collection data or the literature resources. Strictly speaking, verbs should be used between resources in the URL since resources are connected by predicates in statements. As the integration of verbs could imply a method-oriented SOAP service, nouns are to be used as a compromise.
In this API, results must be created from a collector-centric view, i.e., even collection objects containing collectors as statement subjects must be found using collector IDs. Therefore, a resource ”collector” is used. Three sub-resources result from the use cases described above.

1. Collector: The document as a resource contains all information from the required information systems (Requirement 4). Finally, in the result document, the collector obtains an RDFS type (@type). Since the collector can be identified as a ”person” and the shared vocabulary schema.org derived from RDFS was created in order to support search engines, the type ”schema:Person” is anntotated to the collector. Moreover, search engine supportive vocabularies may facilitate collector searches via the Web Portal mentioned in Use Case 1 (3).

2. Biography: This sub-resource returns all biographical data about the person from Wikidata and VIAF. This includes, for example, the various name spellings, birth and death dates, and descriptions. Since Wikidata uses its own vocabulary of properties with alphanumeric variants

such as ”P18”, these properties can be mapped into semantically similar predicates from the schema.org vocabulary.

3. Collection: This sub-resource provides a list of all collected specimens from the Triple Store directly and indirectly associated with the given collector.

4. Bibliography: This sub-resource provides a list of all works published by the collector from BHL and Europeana.

The endpoint can be accessed as described in Table 4.1. In this work, only GET methods are provided, as they are sufficient to meet the requirements.

Method	**URL: host/collector/v1**	**Description**
GET	/{collectorId}/	Get data from of {collectorId}
GET	/{collectorId }/biography	Get biography resources of {collectorId}
GET	/{collectorId }/collection	Get collected specimen resources of {collectorId}
GET	/{collectorId }/bibliography	Get literature resources of {collectorId}

Table 4.1: Endpoints and methods

4.5 Architectural scheme

The API consists of three components: The query component, the logic component and the routing component. The REST-oriented separation of tasks is implemented by this architecture. In this scheme, only the adjacent components communicate with each other, which leads to easier maintenance and development of the API.

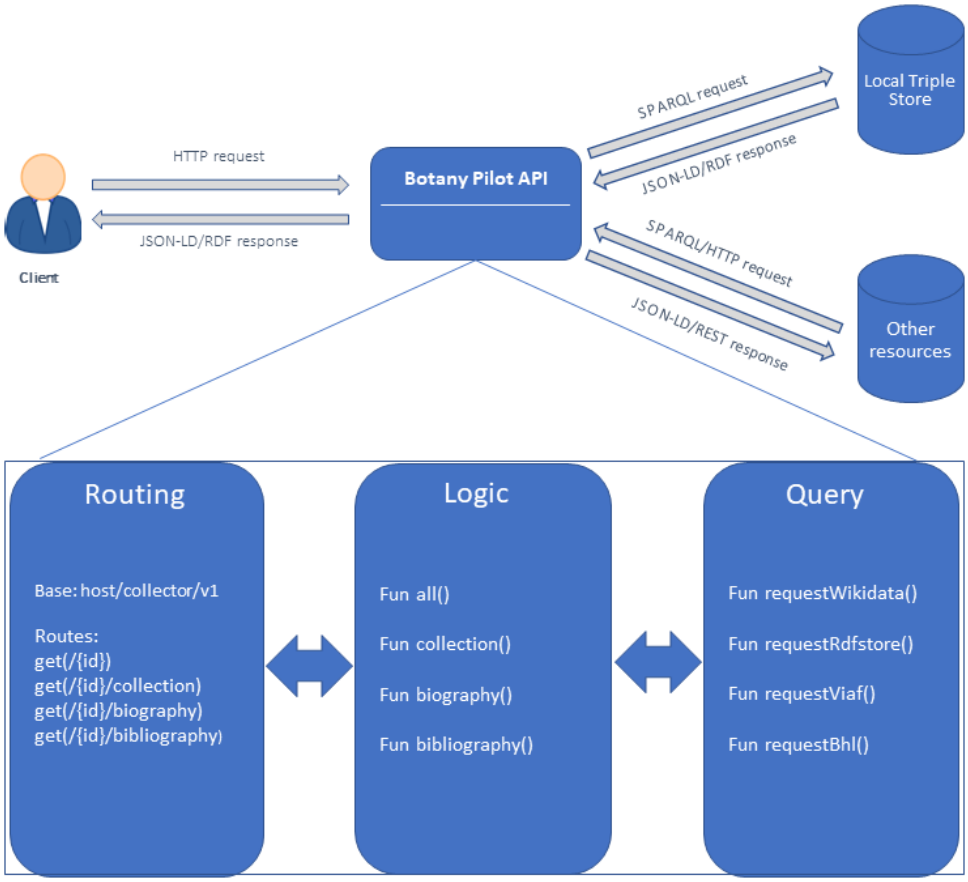

Figure 4.1: External and internal structure of the API

5 Implementation

5.1 Technologies

5.1.1 Slim Framework

The Slim Framework[35] is a micro-framework for implementing APIs. Slim provides core API functionalities such as routing, caching, logging or redirecting. Basically, it is very well suited to develop fast prototypes with a small amount of time due to its small functional scope. A micro-framework is appropriate in this project as an API was developed that essentially pulls linked data together and integrates it into a response file. Moreover, this framework is already used in the existing API infrastructure at BGBM, where the API will be integrated later. An alternative would be Laravel.[36] This full-stack framework is well suited for more complex applications, e.g., by providing out of the box functions for authentication or security. However, features like these are not needed in this project.

5.1.2 JsonLD

JsonLD is a JSON-LD processor for PHP developed by Markus Lanthaler.[37] It complies with the official requirements of the JSON-LD test suite. All JSON-LD transformation algorithms such as flattening or expanding are provided. Since the developer was already involved in the official specification of the standard, this processor was chosen.

5.1.3 EasyRDF

EasyRDF is a PHP library by Nicholas Humfrey for processing RDF.[38] The library provides a wide range of classes and methods to, for instance, create graphs from resources, query SPARQL endpoints, and parse or serialize formats into other formats. The supported formats RDF/XML and JSON-LD are particularly relevant for this work. Currently, PHP libraries with this extent of functionality do not exist. This library also utilizes this processor but does

not access all functionalities in a suitable manner. This especially applies if specification-compliant identifiers such as @id or @value are used, which can be lost in the transformation algorithms called by EasyRDF.

5.2 Components

The routing component instantiates the logic component and determines which method of the logic component is called depending on the requested route. Basically, it serves as the entrypoint to the endpoint. Additionally, it determines the representation of the response based on the request's header parameter Accept:application/format. In this work, both RDF/XML and JSON-LD are provided. The logic component operates depending on the selected route. On

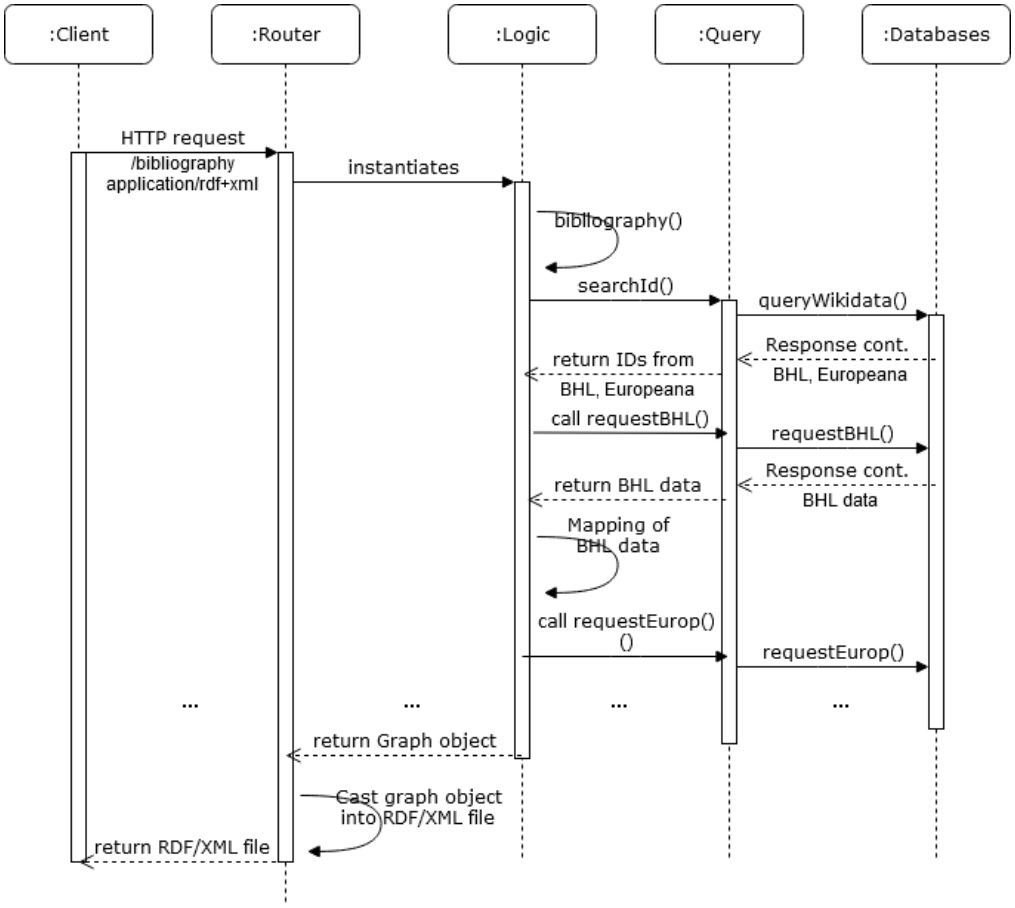

Figure 5.1: Operation example: Bibliography call by using Wikidata ID

the one hand, it calls the corresponding request in the query component and, on the other hand, it integrates the data from the endpoints into a graph that

is returned to the routing component after processing.
The query component provides basic query functions to the various database endpoints. For the queries, cURL or EasyRDF is used. The latter is used for the Wikidata SPARQL endpoint, as the use of cURL did not work, which could be caused by missing header options in the request. However, their introduction in the request led to errors in the queries to other endpoints.

5.3 SPARQL Queries

The API makes various SPARQL queries to the Triple Store and Wikidata respectively. For example, in Wikidata, both the identifiers of the external information systems and biographical information are queried if the Wikidata identifier is known, whereas one other query looks for the collector's Wikidata entity and its ID by using identifiers of other systems. However, as an example, only the Specimen query to the RDF store will be described in this section.
Basically, this SPARQL query identifies all specimen resources that are linked to the searched collector ID via a predicate and constructs a result graph that in turn contains all statements about each specimen. In more detail, the WHERE specification combines three possible cases via UNION. In the first case, all triples are queried that contain a subject that is linked to the searched collector ID in a statement by the property dwciri:recordedBy.

```
PREFIX dwciri: <http://rs.tdwg.org/dwc/iri/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

CONSTRUCT
{ ?subject ?predicate ?object. }
WHERE {
  {
    ?subject ?predicate ?object .
    ?subject dwciri:recordedBy ?a .
    VALUES ?a { <http://www.wikidata.org/entity/Q3822242> }
  }UNION{
    ?subject ?predicate ?object .
    ?subject dwciri:recordedBy ?a .
    ?b owl:sameAs ?a .
```

```
15    VALUES ?b { <http://www.wikidata.org/entity/Q3822242> }
16  }UNION{
17    ?subject ?predicate ?object .
18    ?subject dwciri:recordedBy ?a .
19    ?a owl:sameAs ?b .
20    VALUES ?b { <http://www.wikidata.org/entity/Q3822242> }
21  }}
```

Listing 5.1: Specimen SPARQL query to the triple store

In the second case all triples are queried whose subject is part of a recordedBy statement with a collector object, which itself is additionally the object of a statement with the predicate owl:sameAs linked to the searched collector ID. The third case is basically constructed similar to the second case. However, the searched collector ID is here the object of the "sameAs" relation. The reason for the distinction between case two and three is the unidirectionality of the property in a single statement, i.e., a subject of a statement cannot simultaneously be the object. These pattern cases can also be written in abbreviated form in SELECT queries. In CONSTRUCT clauses, however, these relations must be written out and are not allowed in an abbreviated form. The reason for not using a SELECT clause here is the available format: JSON-LD documents are only offered by the endpoint of the triple store if CONSTRUCT queries are used.

5.4 Mappings

As described in the Resources section 4.4, some responses require proper mappings, e.g., if the source does not return a semantic format. In this API, mappings to different vocabularies are performed in two cases. In another case, the data is condensed to the core relevant to the content according to the identified requirements.

5.4.1 Wikidata properties

As an additional requirement a human friendly representation of the information from Wikidata was given. Therefore, the following properties with alphanumeric abbreviations are mapped.

1. Statements about BHL (P4081), VIAF (P214), and Europeana (P7704): The properties indicating that an entity contains statements about the respective IDs are mapped to the schema:sameAs property in the result graph. Since Wikidata does not return the IDs as URLs, their respective domain addresses are added.

2. Statements about image links (P18): The property of this statement is mapped to the predicate schema:image.

3. Statements about date of birth and date of death (P569/P570): The properties of these statements are mapped to the predicates schema:birthDate and schema:deathDate, respectively.

5.4.2 BHL responses

The BHL endpoint returns the data in a JSON file without Linked Data extension. For literature information, other ontologies and vocabularies are available, such as BIBO.[39] However, for consistency, the following information is mapped into the schema.org vocabulary.

1. Linking to the person entity: Combined with the @reverse keyword, the schema:author property is used to represent the set containing works written by the given collector.

2. BHL entries as book entity: BHL entries with the ”BHLType” value ”Title” are mapped to an entity of the type schema:Book containing the BHL title URL as identifier (@id). In addition, the following mappings are applied in this case:

   - Title -> schema:name
   - PublicationDate -> schema:datePublished
   - PublisherName -> schema:publisher
   - PublisherPlace -> schema:location

   For the publisher information, a corresponding entity of type schema:Organization is created in the form of a blank node (without ID), which is directly linked to the book entity.

```
else {
    if ($value["Genre"] == "Article") {
```

```
3         $currentWork["@id"] = $value["PartUrl"];
4         $currentWork["@type"] = "http://schema.org/Article";
5         $currentWork["http://schema.org/name"] = $value["Title"];
6         if (isset($value["PageRange"])) {
7             $currentWork["http://schema.org/pageStart"] = explode("
                  --", $value["PageRange"])[0];
8             if (count(explode("--", $value["PageRange"])) == 2) {
9                 $currentWork["http://schema.org/pageEnd"] = explode(
                      "--", $value["PageRange"])[1];
10            }
11        }
12        if (isset($value['ContainerTitle'])) {
13            $currentWork["http://schema.org/isPartOf"] = array('
                  @type' => 'http://schema.org/PublicationIssue', '
                  http://schema.org/datePublished' => $value["Date"],
                  'http://schema.org/volumeNumber' => $value["Volume"
                  ], 'http://schema.org/isPartOf'=> array('@type' => '
                  http://schema.org/Periodical', 'http://schema.org/
                  name' => $value["ContainerTitle"]));
14        }
15        $result['@reverse']['http://schema.org/author'][] =
              $currentWork;
16    }
17 }
```

Listing 5.2: Logic component: Mapping of an BHL 'Article' entity into a schema entity

3. BHL Entries as Article Entity: Since botanists often publish in journals, articles frequently occur in the BHL data whose "Genre" data field contains the value "Article". This case must be handled separately, i.e., the data must be transformed to entities of the semantically appropriate type schema:Article. The BHL "PartURL" is used as the identifier of the entity. The pages of the journal where the article is located are mapped this way:

   - PageRange -> schema:pageStart + schema:pageEnd

   According to the schema.org documentation, the journal must be mapped into a "PublicationIssue" type entity, which is part of a "Periodical" type entity. While the latter then contains the name of the journal, the "Pub-

licationIssue" entity is linked to information such as the publication year as well as to the issue number using the corresponding properties of the vocabulary.

5.4.3 Europeana responses

Two of the available Europeana endpoints are used to search for media entities and their respective metadata: The search API and the record API. While the search endpoint queries all item IDs for a given Europeana collector ID, the record API queries single items. Theoretically, it would also be possible to use the results of the search API for the result graph. However, since these are not in a Linked Data format and further do not comply with EDM, the Record API is additionally queried for every single record. The latter returns the data in a format that is compliant with the EDM. The items in the responses are represented as sub-units, of which the following are filtered out by the logic component: Item, Provider and Europeana Proxy, and Provider and Europeana Aggregation (see [40, p.22]). These sub-items contain the information relevant for the API, e.g., the link to the collector's Europeana ID via dc:creator.

5.5 Pagination

Since the number of specimens queried in the collection endpoint might be extensive, pagination is necessary for data minimization. Therefore, an offset-based pagination has been implemented. The query is limited to the value 5,000 to limit the response time. The client sets the offset using the ?page parameter. If additional pages are not available, the following statement is omitted from the result document:

```
<collector-url?page={current-page = x}> as:next <collector-url?page={
    current-page = x+1}>
```

5.6 Versioning

The URL contains a version number to make the API easily extensible without losing backward compatibility. Thus, all enhancements can be implemented

separately in a new version. The URL will be structured according to this scheme: host/version/collector/...

5.7 Documentation

Swagger is to be used for the documentation of the interface. This tool is suitable for describing the used endpoints of a REST API in a machine and human readable format like YAML or JSON. The OpenAPI specification is used as the specification. It documents the endpoints responses and describe what the request parameters and request objects are. This is done independently of the programming language used to implement the API. Further, the SwaggerUI tool can be used to create an interactive console that provides easy access for the user to understand the API. The OpenAPI specifications provide documentation options for both the ”code first” and ”design first” approaches. The first approach is used in this work.

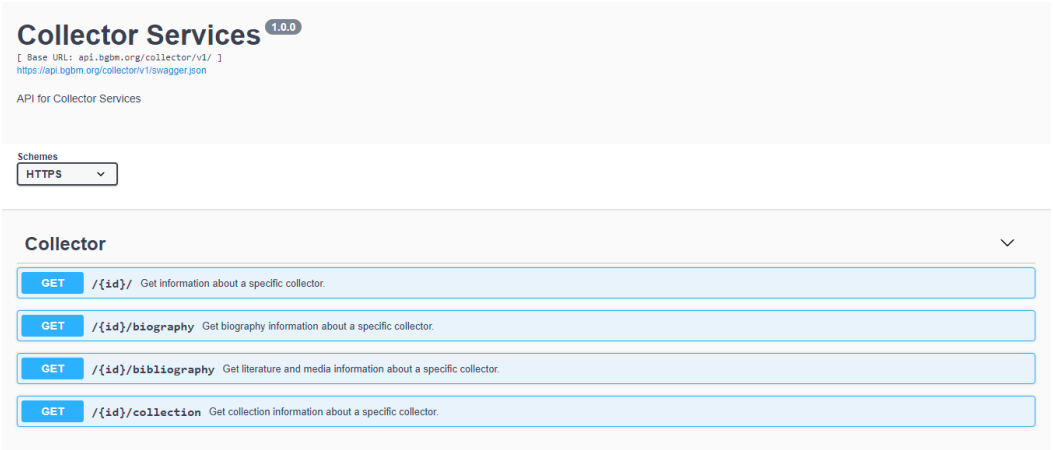

Figure 5.2: API Documentation using Swagger

6 Evaluation

6.1 Functional Requirements

Requirement ID	Title	Use Case	Y / N / Partially
FR1	Providing collection data from a requested collector	(2)	Y
FR2	Providing biographical data on a searched collector	(1)	Y
FR3	Providing bibliographic data of a collector	(2)	Y
FR4	Provide an overall summary of a collector	(3)	Y
FR5	Finding all alternative URIs of a collector	(1), (2), (3)	Y
FR6	Pagination	(2)	Y
FR7	REST constraint compliance	(1), (2), (3)	Partially

Table 6.1: Evaluation of functional requirements

6.2 Non-functional requirements

Requirement ID	Title	Use Case	Y / N / Partially
NFR1	Versioning	-	Y
NFR2	Documentation	-	Y

Table 6.2: Evaluation of non-functional requirements

6.3 Performance testing

During the development stage, performance tests were carried out to determine efficient LIMIT values in the specimen queries. According to Miller[41], 1,000ms interruption is perceived as noticeable, but not an interruption. The range of about 1,500-2,000ms was set as a tradeoff to not fall below a threshold of 5,000 triples in the response. This reduces the number of paginated follow-up requests. Furthermore, by tagging the response data as cacheable, a large part of the second requests can be intercepted via the client cache. The latter is important in web portals like that described in Use Case 1. Jmeter[42] was used as the test software and different loads were tested. The following diagram shows the response times of three test groups with 10 threads and 50 requests in 10 seconds. The peaks are often caused by delays in the various endpoints, such as Wikidata.

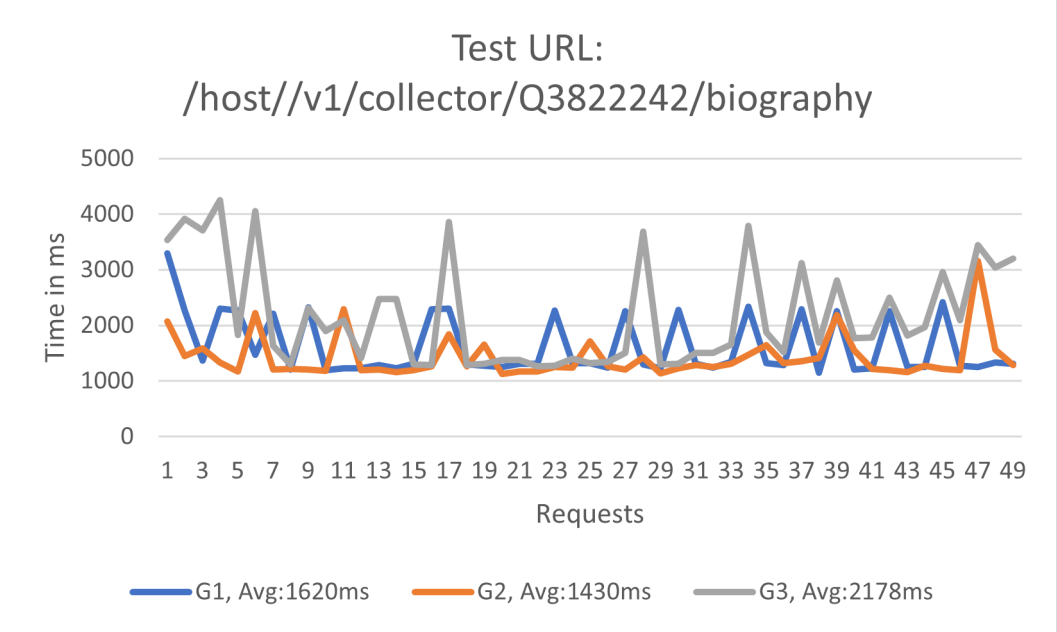

Figure 6.1: Test diagram

7 Summary

7.1 Discussion

The objective of this work was to develop an API capable of integrating distributed botanical collection data, publications, and biographical information about a collector through the use of a person ID. The basic requirements to achieve this are botanical collections enriching their data with identifiers and making them available so that they can be stored in a central triple store. Furthermore, it could be shown that the use of different collector IDs in the different institutions does not have to be an obstacle for the integration of the data. By using central knowledge bases such as Wikidata, alternative identifiers for the person can be found and thus data from different collections can be assigned to a single person. The inclusion of appropriate sameAs predicates in queries has also been helpful in addressing this issue.
However, to meet the requirements that emerged from discussions with the BGBM's partner institutions, this was not the only problem to be solved. Other problems, such as the availability of non-semantic data through the Biodiversity Heritage Library (BHL), could only be solved by mapping this information into a shared vocabulary, schema.org. This vocabulary was chosen primarily for consistency, as the collector itself is an object of type schema:Person. Nevertheless, it should be mentioned here that there are a number of other vocabularies for this purpose.[39]
Another issue in the project was the choice of a suitable API design. Based on the requirements, it was clear that the API had to be easy to access and that it should not require a great amount of prior knowledge of SPARQL for its use. From sketching the necessary implementations in alternatives such as SOAP and from the knowledge gained from related work[43], the choice fell on the REST architectural style. Moreover, this style should facilitate future enhancements of functionality and the integration of data from other information systems. Also, REST strongly influenced the layout of the individual API components - a distinct separation of the functions is intended to make the

maintenance of the API as convenient as possible.
However, some problems arose during the project that could not be solved within the scope of this work. One of them concerns the REST constraint HATEOAS. Hypermedia is intended to facilitate navigation through the API by allowing the use of paths to deeper resources in the URL. In this API, this means that starting from the collector as the central entity, it is possible to navigate to the lists of publication objects, Europeana objects, and collection objects (level 1), and subsequently to each of the single objects (level 2). Level 1 can be accessed, level 2 however not, which is due to the following considerations.
For instance, the collection objects from the distributed collections are denoted with local object identifiers whose domains differ between institutions.[1] In HATEOAS, local IDs in the API URL could be used to navigate to the object. However, since these objects do not have centrally assigned identifiers, it cannot be ruled out that two objects with different URIs from different collections share the same local identifier. Although this is rather unlikely, distinguishability cannot be guaranteed. One approach in this regard is to assign local identifiers via a sameAs statement in the RDF store, but this would require the stability of these identifiers to be guaranteed. The identifiers of objects from Europeana, for example, are queried on the fly - here the API itself would have to make ID assignments accordingly. Another option is to use a modified object ID in the navigation URL, e.g. replacing the slashes with underscores and including the domain.
Owing to this problem, the REST constraints could be met only partially. Hence, the API is not RESTful. According to the Richardson Maturity Model, an API is only RESTful if it also reaches the highest level, Hypermedia Controls. [44] That is, the API could currently rather be classified as RESTlike. Accordingly, a contradiction in the elicited requirements can be seen here: a REST API is to be developed that does not have to offer hypermedia up to the level 2 mentioned before. As other institutions may be interested in hypermedia functions in the API, a possible compromise could be to use the URI conventions already discussed.
In the context of this work, component tests have not been carried out at this point. In consultation with the partners, these will be conducted during the integration into their services. This will require the development of a test model in which, e.g., the different responses of the external information systems are

considered, creating a recognisable distinction between an unsent data situation and an unavailable resource situation.
Along with annotating collector identifiers, CETAF institutions have already started annotating collection objects with geographic identifiers.[45] Previously available as free text only, the use of URIs (e.g. of GeoNames[46]) allows to integrate geographic entity information. Thus, all locations where the collector was active might be retrieved via an additional API endpoint. Different founds at different times are linked to a location identifier - another step in the integration of Linked Data in biodiversity research.

7.2 Conclusion

This thesis covers the implementation of a generic service for the integration of semantic data and it was shown that the enrichment of collection data with identifiers of different domains can be overcome. For this, it was necessary to develop methods that can associate different identifiers to a single collector. Linked data mechanisms searching central knowledge bases such as Wikidata for biographical data and identifiers, as well as using links in the collection data held in an RDF store. Integration of information from external information systems such as BHL could be provided via mappings into shared vocabularies.
Moreover, it was evaluated and discussed whether the requirements elicited with the partners could be fully met. While the key functionalities could be implemented, the HATEOAS constraint of the REST architectural style could not be fully implemented. In the discussion, the basic problem of identifiers in the URL causing this issue was discussed. Additionally, basic requirements for the future implementation of this functionality were outlined. To conclude, the implementation of HATEOAS was not yet necessary in the context of this work, but could become important in future enhancements if new use cases emerge due to new partners or the integration of additional information systems.
An API was developed, documented via Swagger, which can be easily integrated into partner institutions' systems through open access requirements. Machine-understandable collection data, as well as biographical and bibliographical information about a collector, can also be provided when an arbitrary ID from previously defined domains is used. The API developed in this

thesis is also the starting point for data integration via the use of additional identifiers. Semantic annotations to collection object occurrence locations have already been started. Therefore, it is considered to design another endpoint to geographic information that provides collection objects to a given geographic identifier.

Bibliography

[1] A. Güntsch, R. Hyam, G. Hagedorn, S. Chagnoux, D. Röpert, A. Casino, G. Droege, F. Glöckler, K. Gödderz, Q. Groom, J. Hoffmann, A. Holleman, M. Kempa, H. Koivula, K. Marhold, N. Nicolson, V. S. Smith, and D. Triebel, "Actionable, long-term stable and semantic web compatible identifiers for access to biological collection objects," *Database : the journal of biological databases and curation*, vol. 2017, no. 1, 2017.

[2] Q. Groom, R. Hyam, and A. Güntsch, "Data management: Stable identifiers for collection specimens," *Nature*, vol. 546, no. 7656, p. 33, 2017.

[3] "Botanischer Garten und Botanisches Museum Berlin," 01.09.2021. [Online]. Available: https://bo.berlin/

[4] "Botany Pilot Search," 20.08.2021. [Online]. Available: https://services.bgbm.org/botanypilot/

[5] "Plantentuin meise," 01.09.2021. [Online]. Available: https://www.plantentuinmeise.be/fr/

[6] W. G. Berendsohn, Ed., *Resource Identification for a biological collection information service in Europe (BioCISE): Results of the Concerted Action "BioCISE Resource Identification" funded by the European Commission, DG XII, within the EU Fourth Framework's Biotechnology Programme, August 1, 1997 to december 31, 1999.* Berlin-Dahlem: Botanical Garden and Botanical Museum, 2000.

[7] *Convention on biological diversity.* s.l.: Environmental Law and Institutions Programme Activity Centre, 1992.

[8] A. Purvis and A. Hector, "Getting the measure of biodiversity," *Nature*, vol. 405, no. 6783, pp. 212–219, 2000. [Online]. Available: https://www.nature.com/articles/35012221

[9] I. N. Sarkar, "Biodiversity informatics: organizing and linking information across the spectrum of life," *Briefings in bioinformatics*, vol. 8, no. 5, pp. 347–357, 2007.

[10] J. L. Edwards, M. A. Lane, and E. S. Nielsen, "Interoperability of biodiversity databases: Biodiversity information on every desktop," *Science*, vol. 289, no. 5488, pp. 2312–2314, 2000. [Online]. Available: http://www.jstor.org/stable/3077957

[11] B. P. Hedrick, J. M. Heberling, E. K. Meineke, K. G. Turner, C. J. Grassa, D. S. Park, J. Kennedy, J. A. Clarke, J. A. Cook, D. C. Blackburn, S. V. Edwards, and C. C. Davis, "Digitization and the future of natural history collections," *BioScience*, vol. 70, no. 3, pp. 243–251, 2020.

[12] "Gbif," 01.08.2021. [Online]. Available: https://www.gbif.org/occurrence/charts

[13] "Abcd - access to biological collection data," 05.08.2019. [Online]. Available: https://abcd.tdwg.org/

[14] "Darwin core - darwin core," 30.07.2021. [Online]. Available: https://dwc.tdwg.org/

[15] W. G. Berendsohn, A. Güntsch, N. Hoffmann, A. Kohlbecker, K. Luther, and A. Müller, "Biodiversity information platforms: From standards to interoperability," *ZooKeys*, no. 150, pp. 71–87, 2011.

[16] Consortium of European Taxonomic Facilities - CETAF, "Cetaf - homepage v2 - consortium of european taxonomic facilities - cetaf," 19.07.2021. [Online]. Available: https://cetaf.org/

[17] Q. Groom, A. Güntsch, P. Huybrechts, N. Kearney, S. Leachman, N. Nicolson, R. D. M. Page, D. P. Shorthouse, A. E. Thessen, and E. Haston, "People are essential to linking biodiversity data," *Database : the journal of biological databases and curation*, vol. 2020, 2020.

[18] T. I. BERNERS-LEE, J. HENDLER, and O. R. LASSILA, "The semantic web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001. [Online]. Available: http://www.jstor.org/stable/26059207

[19] G. Antoniou and F. van Harmelen, *A semantic Web primer*, 2nd ed., ser. Cooperative information systems. Cambridge, Mass.: MIT Press, 2008.

[20] *Natural language processing*, 2001. [Online]. Available: http://surface.syr.edu/cgi/viewcontent.cgi?article=1019&context=cnlp

[21] "Resource description framework (rdf): Concepts and abstract syntax," 09.10.2018. [Online]. Available: https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[22] "Linked data - design issues," 06.11.2017. [Online]. Available: https://www.w3.org/DesignIssues/LinkedData.html

[23] "ontology - wiktionary," 20.08.2021. [Online]. Available: https://en.wiktionary.org/wiki/ontology

[24] "Xml essentials - w3c," 19.06.2019. [Online]. Available: https://www.w3.org/standards/xml/core

[25] R. Page, "Towards a biodiversity knowledge graph," *Research Ideas and Outcomes*, vol. 2, p. e8767, 2016.

[26] "Json-ld 1.1," 08.07.2020. [Online]. Available: https://www.w3.org/TR/json-ld11/

[27] "Web services architecture," 09.10.2018. [Online]. Available: https://www.w3.org/TR/ws-arch/#introduction

[28] Roy T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, 2000. [Online]. Available: https://www.researchgate.net/publication/216797523_Architectural_Styles_and_the_Design_of_Network-based_Software_Architectures

[29] "Wikidata," 30.08.2021. [Online]. Available: https://www.wikidata.org/wiki/Wikidata:Main_Page

[30] "Biodiversity heritage library," 30.08.2021. [Online]. Available: https://www.biodiversitylibrary.org/

[31] "Viaf," 01.02.2021. [Online]. Available: https://viaf.org/

[32] Europeana Pro, "Europeana data model | europeana pro," 30.08.2021. [Online]. Available: https://pro.europeana.eu/page/edm-documentation

[33] "Soap version 1.2 part 1: Messaging framework (second edition)," 02.10.2017. [Online]. Available: https://www.w3.org/TR/soap12-part1/

[34] Pascal Giessler, Michael Gebhart, Dmitrij Sarancin, Roland Steinegger, and Sebastian Abeck, "Best practices for the design of restful web services," *Proceedings - International Conference on Software Engineering*, vol. 10, pp. 392–397, 2015. [Online]. Available: https://www.researchgate.net/publication/301694429_Best_Practices_for_the_Design_of_RESTful_Web_Services

[35] Slim Framework, "Slim framework," 23.08.2021. [Online]. Available: https://www.slimframework.com/

[36] "Laravel - the php framework for web artisans," 03.09.2021. [Online]. Available: https://laravel.com/

[37] GitHub, "Github - lanthaler/jsonld: Json-ld processor for php," 31.08.2021. [Online]. Available: https://github.com/lanthaler/JsonLD

[38] N. Humfrey, "Easyrdf - rdf library for php," 31.08.2021. [Online]. Available: https://www.easyrdf.org/

[39] "Linked open vocabularies (lov)," 31.08.2021.

[40] "Europeana data model primer," 14.07.2013. [Online]. Available: https://pro.europeana.eu/files/Europeana_Professional/Share_your_data/Technical_requirements/EDM_Documentation/EDM_Primer_130714.pdf

[41] R. B. Miller, "Response time in man-computer conversational transactions," in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I)*. New York, New York, USA: ACM Press, 1968.

[42] "Apache jmeter - apache jmeter™," 22.01.2021. [Online]. Available: https://jmeter.apache.org/

[43] Markus Schröder, Jörn Hees, Ansgar Bernardi, Daniel Ewert, and Steffen Stadtmüller, *Simplified SPARQL REST API - CRUD*

on JSON Object Graphs via URI Paths, 2018. [Online]. Available: https://www.researchgate.net/publication/324982439_Simplified_SPARQL_REST_API_-_CRUD_on_JSON_Object_Graphs_via_URI_Paths

[44] Ivan Salvadori and Frank Siqueira, "A maturity model for semantic restful web apis," 2015. [Online]. Available: https://www.researchgate.net/publication/281287283_A_Maturity_Model_for_Semantic_RESTful_Web_APIs

[45] D. Röpert, F. Reimeier, J. Holetschek, and A. Güntsch, "Semantic annotation of botanical collection data," *Biodiversity Information Science and Standards*, vol. 3, 2019. [Online]. Available: https://www.researchgate.net/publication/333756870_Semantic_Annotation_of_Botanical_Collection_Data

[46] "Geonames," 28.08.2021. [Online]. Available: https://www.geonames.org/