A Generic Web Cache Infrastructure for the Provision of Multifarious Environmental Data

Thorsten Schlachter, Eric Braun, Clemens Düpmeier, Christian Schmitt, Wolfgang Schillinger

A Generic Web Cache Infrastructure for the Provision of Multifarious Environmental Data

Thorsten Schlachter[1], Eric Braun[1], Clemens Düpmeier[1],
Christian Schmitt[1], Wolfgang Schillinger[2]

[1]Karlsruhe Institute of Technology, Karlsruhe, Germany
thorsten.schlachter@kit.edu | clemens.duepmeier@kit.edu |
eric.braun2@kit.edu | christian.schmitt@kit.edu
[2]Baden-Wuerttemberg State Institute for Environment, Measurements,
and Nature Conservation, Karlsruhe, Germany
wolfgang.schillinger@lubw.bwl.de

Abstract. As a basis for the efficient data supply for web portals, web-based and mobile applications of several German environmental authorities, a microservice-based infrastructure is being used. It consists of a generic data model and a series of corresponding generic services, e.g. for the provision of master data, metrics, spatial data, digital assets, metadata, and links between them. The main objectives are the efficient provision of data as well as the use of the same data by a wide range of applications. In addition, the used technologies and services should enable data supply as open (government) data or as linked data in the sense of the Semantic Web. In a first version, these services are used exclusively for read access to the data. For this purpose, the data are usually extracted from their original systems, possibly processed and then stored redundantly in powerful backend systems ("Web Cache"). Generic microservices provide uniform REST interfaces to access the data. Each service can use different backend systems connected via adapters. In this way, consuming components such as frontend modules in a Web portal can transparently access various backend systems via stable interfaces, which can therefore be selected optimally for each application. A number of tools and workflows ensure the updating and consistency of the data in the Web Cache. Microservices and backend systems are operated on the basis of container virtualization using flexible cloud infrastructures.

Keywords: Environmental information systems, Generic data model, Master data, Time series, Spatial data, Semantics, Schema, Microservices, REST, Web portals, Mobile apps, Container-based virtualization, Cloud computing, Open government data, Linked data, Semantic Web.

1 Introduction

A direct consequence of the Aarhus Convention was the adoption of the EU directive 2003/4/EG [1]. This directive, respectively its implementation into national law, e.g. the Environmental Information Act in Germany [2], regulates access to environmental

information for the public. Authorities are obliged to the active dissemination of environmental information [2, §7]. The Internet provides a perfect platform for this purpose. Therefore, a lot of environmental information, as well as (raw) data in the sense of open (government) data [3], are already made available using Internet based applications, e.g. websites, portals, mobile applications [4].

Nevertheless, even today, 15 years after the Aarhus Convention entered into force, much environmental information are not yet or only partially available online. The reasons are manifold and correspond with the challenges of providing open government data [4, pp. 30-39]: political, technical, legal, organizational, cultural, and economic reasons are hindering the free dissemination of environmental information. The majority of these barriers cannot be broken down technically, but modern technologies can help in reducing them.

In some cases, however, long-term processes must lead to a rethinking and rerouting of people and institutions in politics and administration. With the concept of a Web Cache presented in this paper, we want to give an impulse how more environmental information can be made accessible to a wider public by addressing at least some of the inhibitory causes.

2 Idea and Basic Concepts

Starting point of our considerations is that environmental information is made available through central entry points such as Web portals and mobile applications. Most of these applications ultimately use data that have been arisen in the daily work of environmental agencies. Mostly the primary purpose of this (original) data is not information for the public, which leads to the problems and challenges listed above. For example, environmental information include personal data, are subject to licenses, consist of large amounts of information, require appropriate user rights, are stored in special data formats, are not accessible via the Internet, are incomprehensible to lay people, aren't available around the clock, etc.

Our basic idea is to provide "Internet-enabled" copies[1] of the original data on a "Web Cache" (Figure 1). The information is being extracted automatically from the original systems (data sources), e.g. professional databases and specialist applications, then being processed (data ingestion) and provided in redundant systems (data management). The avoidance of direct access to original data sources allows for better availability and usage-based scaling of services (data services and data management), and offers security benefits by strict separation of internal and external/public requests. Data flow is designed unidirectional from data sources to the Web Cache. So the Web Cache represents a read-only copy of the data. Consistency or coherence conditions are set for each data type and any data source influencing the nature and frequency of synchronization between data source and Web Cache.

Limited to mainly unidirectional data flows, the Web Cache application is ideally suited to be implemented as a horizontally scalable microservice-based framework. In

[1] An excerpt of the original data meeting all the conditions for publication on the Internet.

general, however, the framework provides the full range of functionality for data management, i.e. functions for adding, updating and deleting data are also available (known as CRUD for create, read, update, delete), also including mechanisms for authentication and authorization, which do not apply for the Web Cache, which exclusively contains public data and does not have any access restrictions for reading.

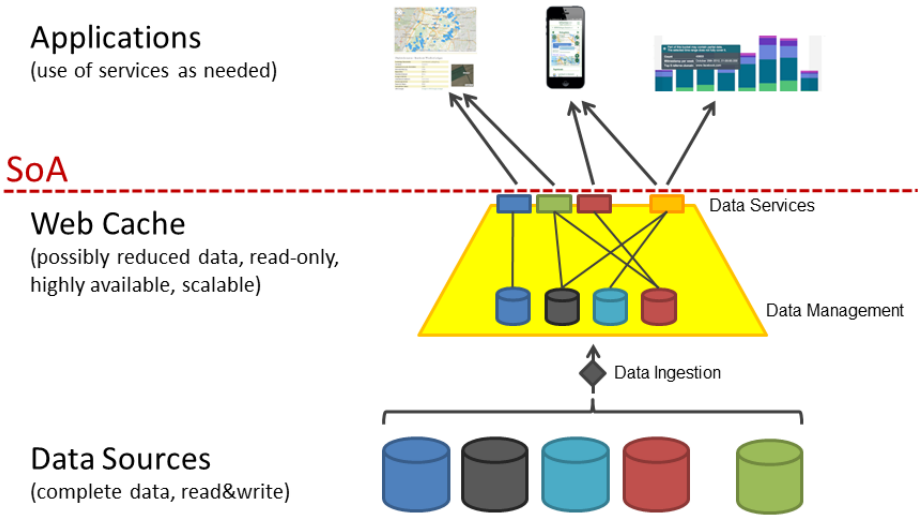

Fig. 1. Web Cache architecture at a glance

3 Architecture and Components

In order to keep efforts of setting up the Web Cache on an acceptable level, it is an essential objective of the project to provide the entire information by means of a limited number of generic services which have to be able to handle a large variety of data semantics. For this purpose, a small set of generic data services and their functionality have been defined allowing the storage of multifarious kinds of data as well as additional semantic metadata in order to provide applications with strongly typed data where necessary. Depending on the application, in addition to newly developed services the use of cloud services off the shelf is eligible[2].
For the implementation of environmental portals, e.g. the LUPO environmental portal family [6], a total of 8 generic services has been identified:

- Master Data Service
- Schema Service
- Time Series Service
- (Media and) Digital Asset Service
- (Full Text) Search Service

[2] Since they may not meet all future requirements, such standard cloud services are rather regarded as interim solutions.

- Geo Data Service
- Metadata Service
- Link Service

These 8 core services are supplemented by two additional services supporting configuration management of (and therefore rather belonging to) consuming applications:

- Application Configuration Service
- Data Discovery Service

These services are described in more detail in section 5.

In a microservice-oriented architecture all services should be independently deployable and usable, and only being coupled loosely. This requirement corresponds to the term “functional decomposition” being used for microservices [7]. Therefore, each data service provides functionality for managing one single generic type of data.

Packed in runtime containers such as Docker[3], the services can be operated without any additional effort on a variety of possible infrastructures, like dedicated servers, clusters, or in the cloud [8]. Using runtime infrastructures like Kubernetes[4], operational aspects such as (rolling) updates, monitoring, horizontal scalability and load balancing are just a matter of configuration – assumed an appropriate computing infrastructure and software design.

All services use suitable backend systems, which in particular ensure the persistence of the data. Here, again, the architecture is abstracted from concrete systems, so that backend systems can easily be replaced by others, or different backend systems can be used simultaneously. The selection of suitable backend systems, e.g. various NoSQL technologies, also ensures dynamic properties such as load balancing, scalability, etc. at this level. All services provide their functionality through versioned RESTful interfaces via content negotiation [9]. This facilitates the development, maintenance, and replacement of individual services.

The postulated independence of services must not lead to a loss of possible functionality, for example by a lack of inter-service interaction. For this purpose, a microservice-based architecture provides a messaging infrastructure (channels) for loose asynchronous coupling of services. However, unlike Gartner's microservice architecture [10], in applications similar to the Web Cache the functionality of the messaging layer "below" the microservices may be delegated to the data ingestion phase and/or to consuming applications using an event bus [11], a simplification, which is sufficient for a wide range of given use cases with unidirectional data flow.

4 Generic Data Model

The services mentioned in section 3 are the basic elements of a well-considered generic data model resulting from a use case analysis in different application domains

[3] https://www.docker.com
[4] http://kubernetes.io

beyond the environmental field. However, we do not claim it to be completely universal. Although the model implements generic data types, it compensates the loss of a strong (relational) schema by using additional semantic services which add missing semantics back into the service-oriented data management infrastructure (like it is done for the Semantic Web). One main advantage of the external provision of semantic information by means of dedicated services is that schema information is not hard-coded anymore, so it can easily be shared between applications.

The core data type of the generic data model is the master data object. A master data object may be a digital model of any entity (of the application-relevant part) of the real world, e.g. a nature protection area, a measuring station, a wind turbine, or a legal document. Each master data object is described by a set of structured properties identifiable by a certain key attribute. This structure can be formalized in a data schema. Objects of the same type use the same schema and belong to the same class of objects (master data). This classification step is of great importance as it directly assigns particular semantics to classes and the respective objects. Relationships between master data objects (or between master data types) can be expressed in various ways, e.g. a composition within a schema (an object consisting of sub-objects), or by the explicit provision of a certain relation between two objects. In general, relations can be typed, directed or undirected, and may have properties as well.

A number of master data types need special consideration. Environmental monitoring often consists of measured values, e.g. time series describing the concentration of ozone on a certain location or the performance of a wind turbine over time. In addition, most environmental objects do have a spatial reference, i.e. (at a certain time) they are located at a specific place in the world and may have specific geometry. The generic data model takes this into account and therefore provides generic data types (and respective services) for time series and spatial data. Digital assets can also be viewed as a special case. Just like other "real world objects", they are usually assigned properties (metadata). In addition, however, the concrete digital object may be provided as well, e.g. as image file, HTML snippet, PDF document or audio/video stream. The generic data types mentioned thus form the basis for specialized services, which offer specific access (service interfaces) to the respective data type.

5 Data Services

The services are divided into data services, which form the core of the framework, and supplementary services, which focus on the support of consuming applications. For the Web Cache, the following sections only describe the core data services. Further services, e.g. Link Service and Data Discovery Service, are described in the conference paper by Eric Braun et al. [12].

5.1 Master Data Service and Schema Service

Almost all real-world objects have properties which can be expressed in a corresponding data model. In general, this is not only static or structured data, but there are also

dynamic or unstructured parts, e.g. objects can contain components which can be regarded as independent objects, too (subobjects or compositions).
The Master Data Service considered here represents a simplification in comparison to such a general master data model since it essentially stores static and structured data. Other properties, in particular dynamic parts, compositions and relationships, are stored by means of references which can refer to both master data and various data types from other services.

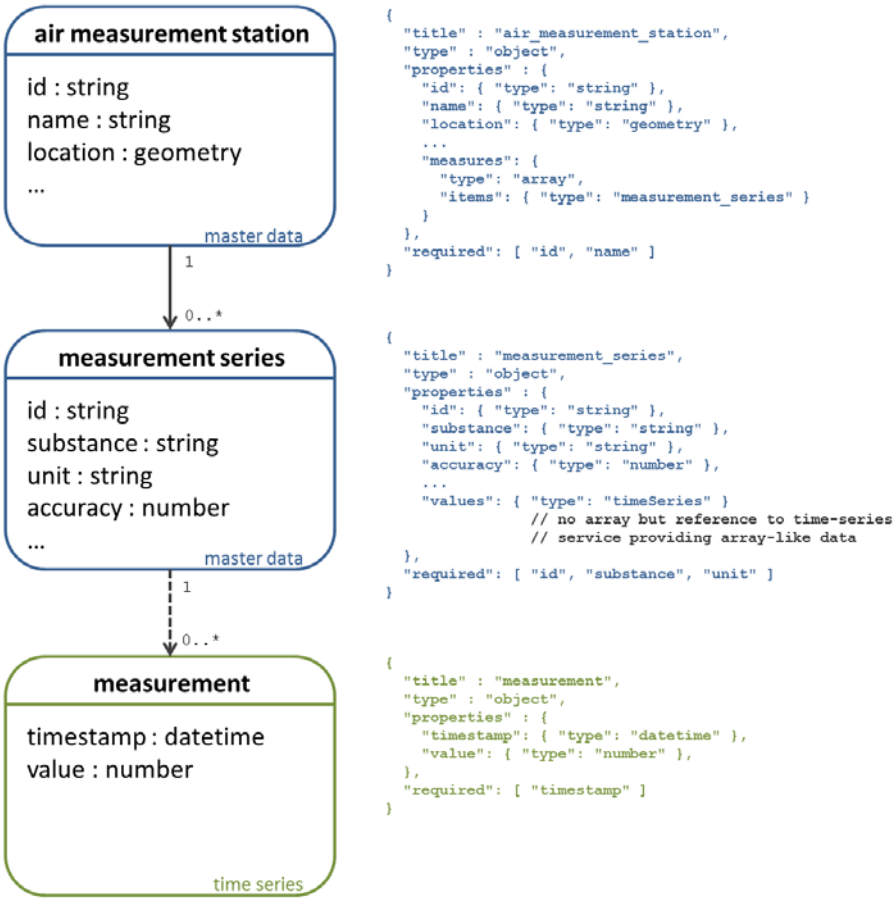

Fig. 2. Example of master data using a composition (measurement series as sub-objects of air measurement station) and references to actual measurement values in the time series service shown as UML-like diagram and JSON schema

For example, the master data of an air measurement station (id, name, location, references to multiple measurement series, etc.) are stored in the Master Data Service (Fig. 2). A measurement series is also stored in the Master Data Service (id, substance, unit, accuracy, references to the measured values, etc.), while the actual meas-

ured values are stored in the Time Series Service. The reference type “timeSeries” defines a custom data type not shown here (timeSeries ~ array of measurement).

The Master Data Service primarily provides a service facade, which guarantees a uniform and stable interface to applications. The actual persistent storage of the data takes place in different backend systems, which are each connected to the Master Data Service via adapters. This allows the connection and exchange of various backend systems (e.g. relational or NoSQL databases, search engines) depending on the specific requirements or applications.

In order to store and provide different data types using a generic service, a structural as well as a semantic description of the data is required. The descriptions of all data types are provided via a Schema Service. Since data types can change over time, schemas have to be versioned. Schemas do not only refer to the data types used in the Master Data Service, but also to the contents of all data services within the framework. Services and consuming applications may use these schemas, e.g. to validate incoming data or to use the structure when visualizing the data. The Schema Service and its implementation based on JSON schema[5] are described in more detail in [12].

Using schemas, the Master Data Service and other services may resolve references, i.e. replace references by the corresponding data, or provide references as URLs/HTTP URIs[6]. Using server-side communication via message channels, the former can lead to a considerable performance gain and possibly reduce complexity in the client or the consuming application.

On the code base of the Master Data Service, specialized content-specific variants can be set up, e.g. a service for storing and providing metadata. Another possible use is described as "Application Configuration Service" below.

5.2 Time Series Service

Since the corresponding master data is already stored in the Master Data Service, the Time Series Service simply stores the actual measured values (time stamp and value per measurement). This again corresponds to the paradigm of the single responsibility per microservice[7]. In the sense of this single responsibility, the Time Series Service, with the aid of the underlying specialized time series databases, is capable of performing specific, time series-related tasks, e.g. filtering of data, data aggregation or unit transformations.

5.3 Geo Data Service

The Geo Data Service is used to store and provide spatial data. In many cases, these data correspond to master data, whereby one or more attributes describe the position and/or the geometry of the objects. Therefore, depending on the application, e.g. if no

[5] http://json-schema.org

[6] The use of HTTP URIs as references between objects applies to the core ideas of Linked Data.

[7] As known from Unix as “Do one thing and do it well.”

complex spatial operations are required, it is possible to provide spatial data exclusively via the Master Data Service, e.g. using the GeoJSON format.

However, if specific spatial operations or special data formats are required, the provision of the data on the spatial data service is useful and necessary.

Since the requirements for the Geo Data Service are still fully supported by a (Cloud-based) solution off the shelf (CARTO[8]), the implementation of this service is currently postponed. For some applications, data is stored redundantly in both the Geo Data Service and the Master Data Service and synchronized automatically.

5.4 Digital Asset Service

Digital assets have the special characteristic that in addition to the descriptive properties (depending on the context called master data or metadata), the object itself can be accessed as binary data stream. Depending on the application, sometimes a link (HTTP URI) on the original asset may be sufficient, in other cases the provision of a copy may be useful or necessary.

Although the range of types of digital assets, their formats, and use cases is significant, the service just considers them as binary data with a certain format (MIME type) and different properties. In other words, the Digital Assets Service generally does not look into the digital assets. If necessary e.g. the Search Service can be used for that.

A digital asset may exist in several variants and/or formats, e.g. images may be available in different resolutions, or a text document either in MS Word or PDF format. According to the mechanisms of content negotiation the client application usually determines required format.

The Digital Asset Service can be connected to different backend systems via adapters, e.g. document management systems providing a CMIS[9] interface. Using suitable backend systems, streaming services (e.g. for videos) can be applied as digital assets, too.

5.5 Search Service

The last actual data service presented here is the Search Service. It provides index information related to unstructured data, often in the form of text documents. To be more specific, it provides access to full-text indexes, which usually provide relevant excerpts (snippets) from as well as references on the entire document.

The Search Service serves as a uniform interface for connecting various full-text search engines, again connected using adapters. The Search Service can also be used in conjunction with the Digital Asset Service, e.g. if an application needs to search the actual content of a document in addition to its metadata.

However, the Search Service can also provide structured data, e.g. when structured and unstructured search results have to be presented in a single view, or when the search engine provides structured information for a facetted search.

---

[8] https://carto.com

[9] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=cmis

The use of the Search Service also simplifies the replacement of a search engine product, usually by creation of an additional adapter. Many search engines can be connected via the existing OpenSearch [10] adapter, and web catalogues using the CSW [11] interface.

5.6 Application Configuration Service

The Application Configuration Service allows applications to store structured information. The service focusses on the reuse of information beyond application boundaries, e.g. complex visualization configurations, map configurations (a map as a compilation of several specific layers), or cross-system settings for individual users.

Technically the Application Configuration Service shares most of its code with the Master Data Service, but there exist some specific extensions, e.g. relaxations with regard to the use of schemas, extended multi-tenant capabilities, vastly limited access, and the use of special service-accounts for authentication and authorization.

6 Technologies

The diversity of services directly implies the use of different technologies for data management, e.g. geographic information systems, different types of NoSQL databases, time series databases, document management systems, full-text search engines, structured search engines, etc. to match the specific requirements of each case. Services can implement facades accessing underlying (Cloud) services, such as Google Cloud SQL, Bigtable, DataStore, storage, etc.

In addition, tools for data ingestion are required, for example Apache Flume [12], Logstash [13] or FME [14]. These tools are essential to ensure the necessary degree of automation for the management of large and diverse data sets, and to be able to control workflows easily and transparently. In addition, they already offer many prefabricated interfaces for the processing of standard data formats, or for adding further interfaces by configuration or by programming of small additional modules.

All services are implemented as microservices based on Java using the "Spring Boot" framework [15], are packaged in Docker containers, and operated on a Kubernetes infrastructure in the Cloud (Google Container Engine). Development instances run on dedicated servers and on a (local) computer cluster, also based on Kubernetes.

Table 1 gives a brief overview of the data services, used frameworks for implementation and runtime environment, and a selection of connectable backend systems.

---

[10] http://www.opensearch.org/Home
[11] http://www.opengeospatial.org/standards/cat
[12] https://flume.apache.org
[13] https://www.elastic.co/products/logstash
[14] http://www.safe.com
[15] http://projects.spring.io/spring-boot/

Service	Implementation and Runtime Environment	Persistence Layer
Master Data Service	Spring Boot, Docker, Kubernetes	Elasticsearch[16] MongoDB[17] Google Cloud SQL[18]
Master Data Service (Interim version)	Google App Engine	Google Cloud SQL
Schema Service	Spring Boot, Docker, Kubernetes	Elasticsearch, MongoDB
Time Series Service	Spring Boot, Docker, Kubernetes	OpenTSDB[19] InfluxDB[20] Elasticsearch
Geo Data Service	CARTO (Cloud)	CARTO
Digital Asset Service *experimental*	Spring Boot Docker, Kubernetes	Alfresco[21] (CMIS-Interface)
Search Service	Spring Boot, Docker, Kubernetes	Google Search Appli-ance[22] Elasticsearch OpenSearch (Atom)
Application Configuration Service *experimental*	Spring Boot, Docker, Kubernetes	Elasticsearch MongoDB
Link Service *experimental*	Spring Boot, Docker, Kubernetes	neo4j[23]

Table 1. Services, their underlying frameworks, and backend systems

7 Experiences

The Web Cache has gradually grown and the presented architecture is in operation since the beginning of 2016. Some precursors of individual services were based on other technologies (Servlets, Google App Engine). With the general idea of the microservice-based architecture in mind during their development, those services could be refactored to "real" microservices.

Experiences in development and operation are very positive. Because of the independence and loose coupling of services a gradual start-up was possible. The devel-

[16] https://www.elastic.co/de/products/elasticsearch
[17] https://www.mongodb.com
[18] https://cloud.google.com/sql/
[19] http://opentsdb.net
[20] https://www.influxdata.com
[21] https://www.alfresco.com
[22] https://enterprise.google.com/search/products/gsa.html
[23] https://neo4j.com

opment of individual services is straightforward and requires relatively short periods of time.

Nowadays, significantly more data are available for more applications than ever before. Data that previously haven't been available, or have been hidden in business applications, now can be used in many ways, for example, by other special applications, websites, portals and mobile apps. Rising requirements in operation, for example a growing number of accesses, can easily be scaled out on the fly using the horizontal scaling capabilities of the container virtualization infrastructure. The use of container technologies allows a greater independence in the selection of infrastructure operators. Also, the relocation of single or several services is easily possible.

Within consuming applications the use of a generic family of services enables the implementation and reuse of generic, highly configurable frontend components. This additionally simplifies the work of online editors and increases the recognition value for users.

Existing generic services are suitable for many new applications and use cases, usually implemented just by configuration.

By abstracting the (versioned) interfaces (REST APIs) from the underlying internal modules, components or even whole services can be exchanged transparently for consuming applications, depending on the infrastructure even without interrupting operations.

For the synchronization of data sources and services, a high level of automation is possible.

The Web Cache has some positive side effects: With the help of its generic services, it is possible to combine data from different sources, or to create comprehensive views on (disjoint) databases, e.g. from different federal states.

Drawbacks can be seen as opportunities, depending on the approach. The fundamental problem of redundancy (dual operation and redundant data storage with the corresponding additional expenses) generates operational flexibility, and facilitates the provision of "Internet-enabled" data. The necessity of concepts for legal issues, operations, consistency, data schemas and formats, provides opportunities to clarify issues, responsibilities and the (re-)definition of (operational) processes.

Although the provision of data for a wide range of users creates transparency, it also reveals poor data quality in some cases.

8 Conclusion and Outlook

The Web Cache concept presented in this paper defines a complete architecture for the dissemination of environmental information. The keystone of this architecture is the provision of multifarious types of data by a limited number of generic microservices. Consistent implementation of these services with (versioned) RESTful APIs, and use of container virtualization offer the greatest possible degree of flexibility in the (further) development and operation. In contrast to the development of monolithic applications, individual services or data containers can be provided very quickly. This

leads to a gradual improvement of the consuming applications, allowing the "release early, release often" philosophy in the development of modern (mobile) applications.

The development of individual services is not yet completed. Existing APIs have to be partly replaced by new, unified, and more powerful versions. This entails a better automatic processing of data, with the aim of being able to provide information according to the ideas of the Semantic Web [12], e.g. using relevant standard formats such as RDF. This also includes, for example, information on provenance of data as well as usage and exploitation rights.

Currently, only freely accessible data is stored in the Web Cache. In order to provide data with limited access via the Web Cache the existing mechanisms for authentication and authorization have to be instrumented.

References

1. European Union: “Directive 2003/4/EG” (2003) http://eur-lex.europa.eu/legal-content/DE/ALL/?uri=CELEX:32003L0004
2. Bundesrepublik Deutschland: “Umweltinformationsgesetz” (2004) https://www.bgbl.de/xaver/bgbl/start.xav?jumpTo=bgbl104s3704.pdf
3. Ubaldi, Barbara: “Open Government Data - Towards Empirical Analysis of Open Government Data Initiatives”; OECD Working Papers on Public Governance, No. 22, OECD Publishing 2013; ISSN 1993-435; DOI: 10.1787/5k46bj4f03s7-en; http://www.oecd-ilibrary.org/governance/open-government-data_5k46bj4f03s7-en
4. Schlachter, Thorsten et al.: “My Environment - A Dashboard for Environmental Information on Mobile Devices”; Environmental Software Systems. Fostering Information Sharing - 10th IFIP WG 5.11 International Symposium, ISESS 2013, Neusiedl am See, Austria, October 9-11, 2013; pp.196-203; DOI: 10.1007/978-3-642-41151-9_19
5. Wikipedia: “Web Cache” https://en.wikipedia.org/wiki/Web_cache; visited August 29th, 2016
6. Schlachter, Thorsten et al.: “LUPO Umsetzung einer (micro-)serviceorientierten Architektur (SOA) für Landesumweltportale”; in: Weissenbach, K.; Schillinger, W.; Weidemann, R. (Edts.) “F+E-Vorhaben INOVUM - Innovative Umweltinformationssysteme - Phase I 2014/2016“; KIT Scientific Reports 7715; 2016; pp. 25-38
7. Fowler, Martin; Lewis, James: “Microservices – definition of this new architectural term”; http://martinfowler.com/articles/microservices.html; visited August, 31st 2016
8. Cohen, Uri: “Containers, microservices, and orchestrating the whole symphony”; opensource.com; https://opensource.com/business/14/12/containers-microservices-and-orchestrating-whole-symphony; visited August, 31st 2016
9. Seemann, Mark: “REST implies Content Negotiation”; http://blog.ploeh.dk/2015/06/22/rest-implies-content-negotiation/; visited August, 31st 2016
10. Olliffe, Gary: “Microservices : Building Services with the Guts on the Outside”; Gartner Blog Network; http://blogs.gartner.com/gary-olliffe/2015/01/30/microservices-guts-on-the-outside/; visited August, 31st 2016
11. Schlachter, Thorsten et al.: “LUPO Umsetzung einer (micro-)serviceorientierten Architektur (SOA) für Landesumweltportale”; in: Weissenbach, K.; Schillinger, W.; Weidemann, R. (Edts.) „F+E-Vorhaben INOVUM - Innovative Umweltinformationssysteme - Phase I 2014/2016“; KIT Scientific Reports 7715; 2016; pp. 25-38o

12. Braun, Eric at al.: “A Generic Microservice Architecture for Environmental Data Management”; submitted to ISESS 2017