openlaws

**Deliverable 3.2.d1**

# Initial architecture and data model specification

Co-financed by the European Commission (DG Justice)
under action grant JUST/2013/JCIV/AG

(April 2014 – March 2016)

*JUST/2013/ACTION GRANTS*

*Grant Agreement Number 4562*

| | |
|---|---|
| Project Start date: | 01.04.2014 |
| Project End date: | 31.03.2016 |

# Report No. D3.2.d1 - Initial architecture and data model specification

| | |
|---|---|
| Version – | 1.5 |

Document prepared by: SALZBURG UNIVERSITY OF APPLIED SCIENCES (SUAS)
Thomas J. Lampoltshammer, Christian Sageder,
Thomas Heistracher

Contributors: Clemens Wass, Matteo Zanioli, Radboud Winkels

Scope: Internal

Deliverable due date: 31.10.2014
Deliverable actual date: 31.10.214
Living Document current date: 06.02.2015

Co-funded by the Civil Justice Programme of the European Union

## Document History

| Date | Revision | Comments |
|---|---|---|
| 12.8.2014 | V0.1 | First draft |
| 15.10.2014 | V0.2 | Second draft – pre final |
| 31.10.2014 | V1.0 | Final |
| 21.11.2014 | V1.3 | Replaced Vaadin by AngularJS<br>Extended Data Model<br>Adjusted Data Mapping<br>Community Integration Layer adapted |
| 06.02.2015 | V1.5 | Update to Data Model and Data Mapping |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## Document Authors

Dipl.-Ing. (FH) Thomas J. Lampoltshammer, MSc M.A.
DI Christian Sageder
Prof. (FH) DI Dr. Thomas Heistracher

*Disclaimer*
This publication has been produced with the financial support of the Civil Justice Programme of the European Union. The contents of this publication are the sole responsibility of the Salzburg University of Applied Sciences and can in no way be taken to reflect the views of the European Commission.

## Participant List

| | **Short Name** | **Organisation Name** | **Country** |
|---|---|---|---|
| 1 | UVA | Universiteit van Amsterdam | NL |
| 2 | SUSS | University of Sussex | GB |
| 3 | LSE | London School of Economics and Political Science | GB |
| 4 | ALP | Alpenite srl | IT |
| 5 | SUAS | Fachhochschule Salzburg GmbH | AT |
| 6 | BYW | BY WASS GmbH | AT |

## Executive Summary

This document comprises all necessary information concerning the architecture of the openlaws.eu platform, the associated data model and mapping as well as information regarding deployment and infrastructure.

## Table of Contents

## List of Figures

## List of Tables

# 1 Introduction

Openlaws.eu aims to initiate a platform and develop a vision for Big Open Legal Data (BOLD): an open framework for legislation, case law, and legal literature from across Europe. Based on open data, open source software and open innovation principles we are adding a *social layer* to existing *legal information* systems (Wass et al., 2013). This document is introduces the first initial architectural draft of the *openlaws* platform, together with a prototypical data model of legal data within the platform.

This document starts with a short description about the interdependencies and influences from other work streams. After that, in section 2, a description of the architecture development methodology is given, followed by a detailed discussion of each individual step and its outcome. Section 3 is then discussion the various data sources, which are potential candidates for the *openlaws* platform. Subsequently, the first initial draft of our data model is presented and discussed in detail.

## 1.1 Correlation from related work streams

This section describes dependencies, introduced by other work streams within the *openlaws* project. Figure 1 depicts the two influencing work streams, namely **WS1 (Mapping Open Law: Resources and Institutional Partners)**, **WS2 (Socio-economic and technical analysis)**, and **WS4 (Dissemination and User Community Engagement).**

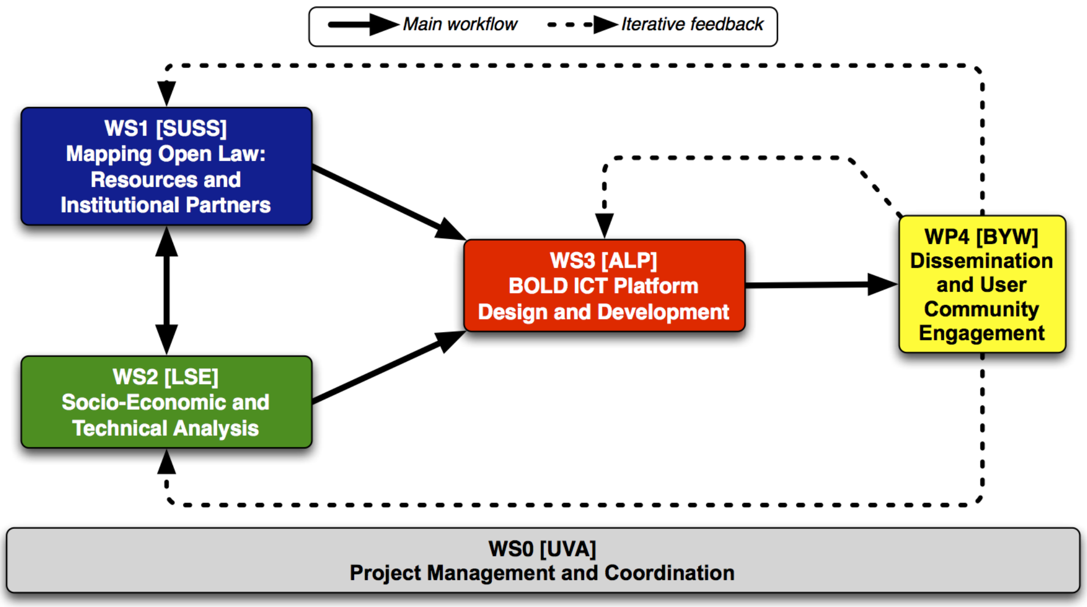

Main workflow
Iterative feedback
WS1 [SUSS] Mapping Open Law: Resources and Institutional Partners
WS2 [LSE] Socio-Economic and Technical Analysis
WS3 [ALP] BOLD ICT Platform Design and Development
WP4 [BYW] Dissemination and User Community Engagement
WS0 [UVA] Project Management and Coordination

*Figure 1 - Dependencies from related work streams*

**WS1**: This work stream contains a comprehensive comparative analysis of law, society and institutional developments towards the open access model, past developments and prognosis for the future, based on the regional case study of European Union law, and country case studies of UK, Austria, Netherlands, compared to other advanced legal systems. The analysis will explain how and whether the environment (institutions, policies and the legal community) is finally developing in which open access models such as OpenLaws.eu can take root and flourish.

**WS2**: This particular work stream focuses on the interaction between (i) the network of big open legal data (BOLD), and (ii) the network of stakeholders (citizens, legal professionals, public bodies, national governments, publishing houses, legal informatics SMEs, and the EC) that use these data. This WS is concerned with how the interaction within and between the following two networks can be optimised: (i) the network of big open legal data (BOLD), and (ii) the network of stakeholders (citizens, legal professionals, public bodies, national governments, publishing houses, legal informatics SMEs, and the EC) that use these data. The overall objectives that influence WS 3 are: (1) to characterize and describe both networks based on the mapping effected in WS1, and to determine their strengths and weaknesses; (2) to develop and partly employ (semi-) automatic ways to interlink existing data and enrich them by adding metadata, in order to facilitate the access and use of legal data; (3) to describe and partly introduce new functionalities to enable users to interlink and enrich the legal data; (4) to develop a socio-economic and governance framework, and methods to build and strengthen the social networks of legal professionals, in support of the evolution and growth of the stakeholder network as a BOLD open innovation community; (5) To develop business models and services tailored to maximising the synergies and sustainability of this heterogeneous user community.

**WS4:** This work stream will ensure that the results of OpenLaws.eu are published and disseminated amongst our target groups and other stakeholders like software developers and commercial publishers to show them how each of them can benefit from the BOLD ICT Platform and to receive broad input for our BOLD Vision 2020. A strong focus lies not only on traditional dissemination like publication in scientific papers and presentations at conferences, but also heavily on the information and engagement and management of the user community by using social media. As the development of the platform through WS 3 is an iterative process, the feedback of the community actively influences the direction of the development.

## 1.2 Target audience

The primary target audience of this deliverable is comprised by the work force of the leader of WS 3, namely ALPENITE, to support them during the first implementation phase as well as for potential extensions throughout the course of the project. In addition, this document provides the legal and developer community a feedback on how their requirements towards the *openlaws* platform are reflected within this initial architectural proposal. Furthermore, the deliverable at hand serves as comprehensive working result documentation for the EU as the funding agency.

# 2 Path Towards A Preliminary Architecture

This section of the deliverable discusses i) the strategy for the development of the *openlaws* platform architecture, ii) risk management within the architectural development, and iii) existing approaches and relevant architectural facets that can offer support during the course of the development.

## 2.1 Strategy for architectural transition from initial to final state

The development and realization of a system architecture is no trivial task. Therefore, it is most important to base its continuous development and deployment on a solid basis. To achieve this task, the *openlaws* project is embedded in a tangible cooperation network, involving all necessary stakeholders to guarantee sustainable results. Figure 2 describes the strategic development path pursued within the project.

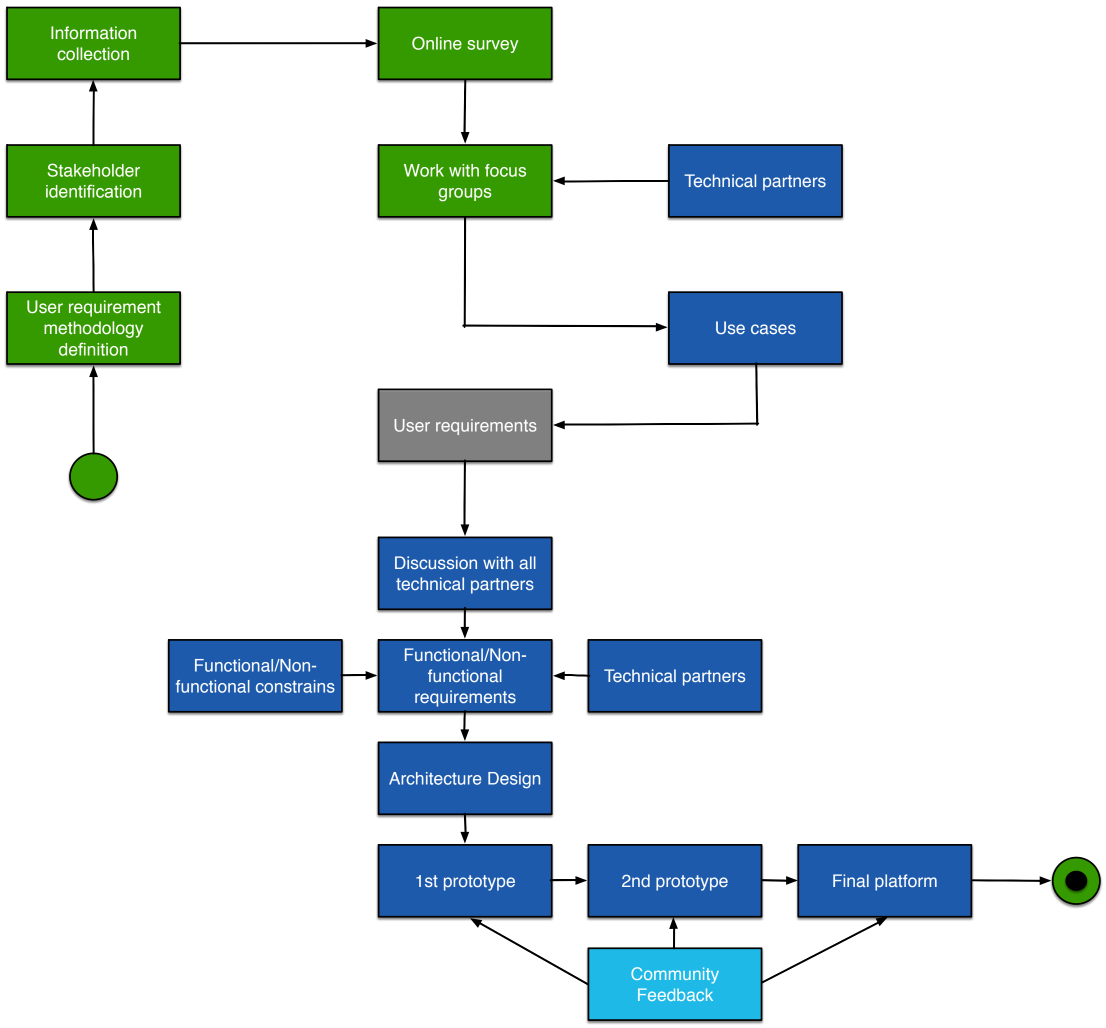


*Figure 2 – Strategy path to define architecture*

After the initial setup of the entire user requirement methodology, the stakeholders are identified and associated detail information is collected. For a comprehensive description of the stakeholders, please refer to **Del. 4.1.d3 – Handbook for Stakeholders.**

Afterwards, an online survey is conducted for identification of user needs from the legal community. These results are presented and extended by the feedback from focus groups, in cooperation with all technical partners. Out of this pool, use cases are drafted, which present the basis for the first set of user requirements. These requirements are transformed into functional and non-functional requirements for the discussion and decision process with the technical partners. After a set for the first initial architecture is found, the actual design process is started, which leads eventually to the first prototypical implementation. In the second step, additional functionalities are added to the prototype until the final state for this project has been reached. During the entire development and implementation phase, the legal community is involved within the process. The feedback associated to the intermediate versions is evaluated and incorporated into the next succeeding version of the *openlaws* platform.

## 2.2 Risk management

An early and well-planned risk management is essential throughout the entire development process. Risk management within the *openlaws* project is divided into three tiers:

1. Strategic tier: all considerations of issues regarding the project and its environment on the consortium level. The entire consortium is handling the risk management
2. Tactical tier: all considerations on work streams and their contribution to the project. The work stream leaders manage this level.
3. Operational tier: individual issues within the work packages. The particular work stream leader handles arising issues.

The first and second tiers are covered within the project handbook of the *openlaws* project. The third tier dedicated to operational concerns, is treated individually within each work stream – of course under incorporation of tier one and two. This work stream is based on a fast turn around-oriented approach, based on *agile* methods (Martin, 2003). There is a continuous exchange between the technical partners already during the architectural development phase. This supports the mapping of planned technologies and available resources and know-how between partners. Regular Skype sessions between the technical partners provide the basis for a productive exchange, early risk identification and solution finding. As this deliverable will offer the basis for a “living documentation” during the prototype development and feedback phase, weekly/bi-weekly development sprints will provide the necessary fine-granular feedback to react on arising issues quickly.

## 2.3 Existing approaches and relevant architectural facets

There already exist platforms regarding the dissemination and accessibility of legal data and information. This section is intended to present a snapshot out of the environment the *openlaws* platform will be embedded in. This snapshot includes the databases to be incorporated within the first prototypical implementation of *openlaws*.

**Legislation.gov.uk** aggregates all legal information within the United Kingdom and makes it accessible to the general public. The search functionality does not only provide access to the information itself, it is also possible to perform searches via the aspect of the geographical extent, or a certain point in time.

**Wetten.overheid.nl** is a central access point to all information concerning government organizations in the Netherlands. This includes Dutch law as well. As an extension of this service, the site doc.metalex.eu provides all legal information in a MetaLex compliant format as well as expert search functions on this data set.

**www.ris.bka.gv.at** is a service offered by the Federal Chancellory of Austria providing access to Austrian laws and rulings, with appropriate search functionalities. It offers in particular access to consolidated federal law, regional laws and rulings of several Austrian tribunals. Human access is through a web based interface, while automated access for federal laws is provided with a public connector (published by the Austrian Open Data initiative www.data.gv.at). OpenLaws has obtained access to a private connector with also regional laws.

On the European level, the portal **Eur-lex.eu** presents the dissemination platform for the European Union for legal documents. The platform provides access to law, legislation, as well as case law for all 24 official EU languages. EUR-Lex supersedes the former CELEX service and features comprehensive search functionalities as well as a Web service for automated search queries.

These Web sites/services do provide access to legal information (for a detailed technical description of the external services, please refer to **Del. 2.1.d1 – Analysis of legal networks**); however, they do neglect the social/community aspect of exchanging knowledge. Web sites such as lawsociety.com, lawyers.com, or lawyers.findlaw.com do bring people together, but more like yellow pages. Still, a social interaction – maybe compare-able to Xing or LinkedIn – does not exist.

The idea of the *openlaws* platform - and therefore the core aspect of the associated architecture - is to provide an aggregated search functionality over EU and national law repositories, while at the same time, to provide an interactive community platform. This platform will then offer management capabilities to handle not only legal data from external sources, but also user-created content such as publications or comments on existing laws and regulations. In addition, the platform will provide the possibility to find users, who are active or interested in the same legal areas and will enable cooperation and communication.

# 3 Preliminary Architecture Description

This section discusses important aspects regarding the platforms architecture in details. First, the general approach towards the architectural design is described, including associated requirements from different directions.

## 3.1 Development methodology for the initial architecture

The inherent structure of the architectural description is based on the ISO/IEC/IEEE 42010:2011 standard entitled "Systems and software engineering – Architecture description" (ISO/IEC/IEEE, 2011). This document contains actions regarding the analysis, creation, and sustainability aspects of software architectures. In addition, terminology, key aspects, as well as examples comprise this standard. Figure 3 depicts the overall concept of an architectural description.

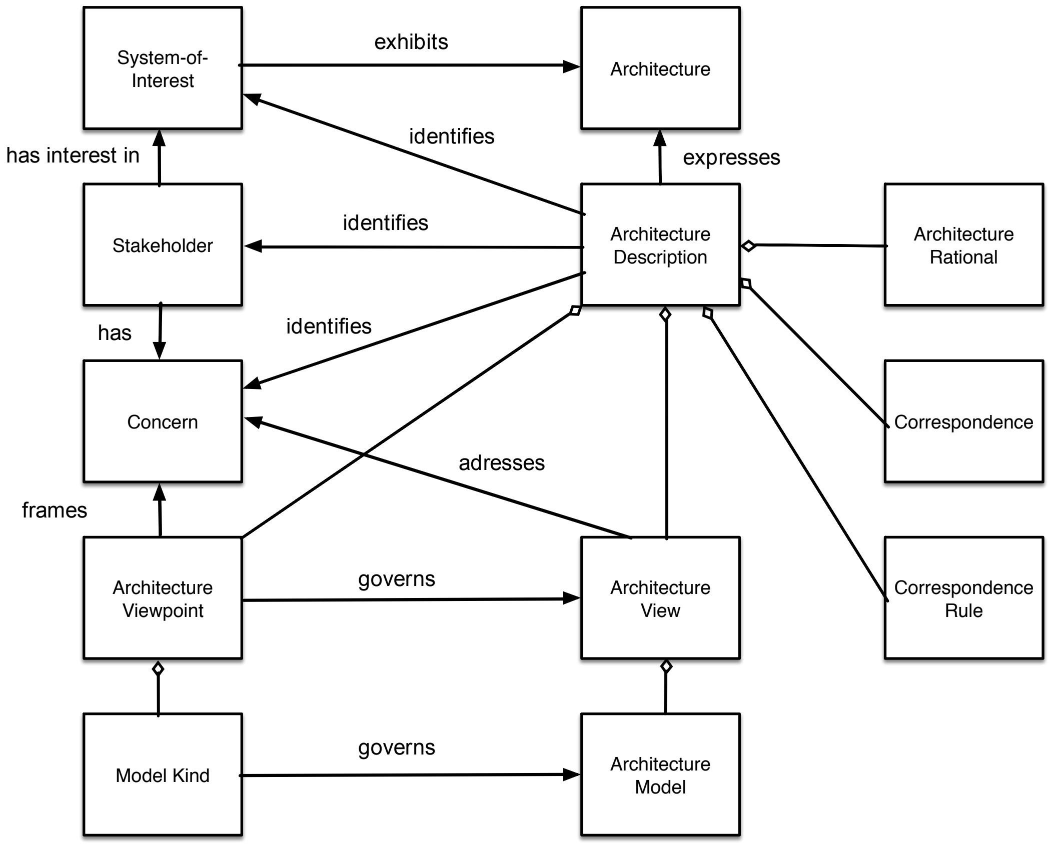


*Figure 3 – Concept of an architectural description, taken from (ISO/IEC/IEEE, 2011).*

The main difference between a system's architecture and its description is based on its meta level. While a system's architecture is an abstract concept of components, its description binds it to actual technology. This approach can be compared to the concept of a Model-driven Architecture (MDA) (Liddle, 2011). As viewpoints on the architecture strongly depend on project specific aspects such as stakeholders, their requirements and so on, the ISO/IEC/IEEE 42010:2011 standard does provide only a basic, non-specific description of these viewpoints. The set of viewpoints being considered for the *openlaws* project is based on the work of Rozanski & Woods (2011). Out of the range of viewpoints, we opted for i) the functional view, ii) the information view, iii) the development view, and iv) the deployment view to properly describe the architecture of the platform.

## 3.2 Non-functional requirements

In this section, essential non-functional requirements of the *openlaws* platform are described. Non-functional requirements per definition do not describe what functionalities the platform will deliver, but how they will be delivered (Chung et al., 2000).

I. **Connectivity and openness:** Not only the gathering of open data, but also distribution and access to these data is one of the major goals of the *openlaws* platform. To achieve the required connectivity and openness, the platform will other open, well-defined interfaces to the community and developers to access open legal data available within *openlaws*. Details about the realization of this non-functional requirement can be found in **D3.2.d2 – Open API Interface Specification.**

II. **User-friendliness:** A rich set of enrichment tools, visualizations, and open interfaces will be provided to experts and users from the law community. To support the community within their daily routine and to foster curiosity-driven exploration of open legal data, it is crucial to design and implement interfaces and system flows within the *openlaws* platform in a user-friendly and intuitive way. Details about the realization of this non-functional requirement can be found in **D3.3.d1 – User Experience Design**.

III. **Efficiency:** The *openlaws* platform will be required to handle large legal as well as user-generated data sets. Therefore, the structure of the data as well as the underlying data handling within the internal database has to be efficient and fast. Wherever possible, linking to the original data source should be preferred instead of copying all data. Furthermore, the information and data flow between each component of the platform has to be efficient and kept to a minimum.

IV. **Scalability:** As the newly introduced *openlaws* platform will provide access for the entire law community within Europe, extensive and reliable scalability becomes imperative. This applies to various aspects of data access within the platform such as creating, storing, processing, and visualization of open legal data. These facets manifest themselves in two main options:

    i. Reliable management of multiple platform instances including access rights, synchronization, and updating.
    ii. Dynamical handling of heavy load situations in real-time.

V. **High Availability:** Continuous access to official law publications as well as self-owned, created content is an essential asset of the *openlaws* platform. Therefore, the platform should feature robustness against potential hardware and software issues, implement redundancy where necessary, and offer load-balancing to provide an overall high availability.

VI. **Data Security:** The platform will provide measurements to ensure data consistency throughout the system. Backups of system components as well as of the data repositories will guarantee availability. A thorough authentication and authorisation system will protect the system and data from unauthorised access and manipulation.

VII. **Data Privacy:** The *openlaws* platform will protect the users personal data. This does not only include secure European cloud solutions but also handling of the data within the system its users and administrators.

## 3.3 User requirements/use cases

The following basic set of requirements was extracted from the outcomes of the survey within the legal community as well as from feedback regarding the focus groups:

- **Search**: The user should be able to search over all information that is present within the data model. This is not limited to legal objects, but also includes other users and their profiles.
- **Register / Invitation:** The basic search functionality should be available for everyone without registration. If a user wants to use the folder, upload, share etc. functionality, registration is mandatory. Invitation means that it is possible to send a "Peter suggest that you join *openlaws*" message. Invitations are mainly intended for legal experts to assure that annotations and are only from skilled people and not from lay persons.
- **Bookmark:** Have a list of potential interesting documents, but still these are not associated to a particular topic folder.
- **Annotation**: Legal experts can annotate legal objects so that they can attach additional textual information that enriches the meta data of the legal document.
- **Intelligent Search Folder**: it is possible to save a search configuration as a folder. This folder will automatically present all subsets fitting the related search request. If a new item is added to the knowledge base and it fits the search, the folder will update itself automatically.
- **Share to public:** Uploaded documents and annotations to these documents or to already existing documents are made searchable and addable for all users of the *openlaws* platform.
- **Share to private**: same as above, only that users decide which group of people they like to share their contents with.
- **Load Legal Content:** the search results from *openlaws* provide the possibility to reach the original document associated to the legal object.

## 3.4 Requirements for enrichment tools

These requirements are derived from **D2.2d1 – Requirements for Enrichment Tools.** As the mentioned deliverable contains requirements for both the architecture and the user experience design, this deliverable will only focus on the first one.

- **Data storage**: Simple data objects should be connectable via relationships to other simple data objects (e.g. subject-predicate-object).
- **Social network:** It should be possible to organize pieces of information for each user in private workspaces. These workspaces can be made public if the owner grants access. The quality of the information within the platform has to be maintained. A combination of user-driven and system-driven quality measurements could be employed to rank content within the *openlaws* platform.
- **Tagging and highlighting:** Users should be able to tag legal data objects with individual or system-suggested tags. Furthermore, it should be possible to highlight various amounts of text within a legal document (e.g. sentence-based or paragraph-based).
- **Bibliographic aspects and versioning:** Highlights from legal documents should have a reference to their origin. In addition, amendments to legal documents or new versions of a document should not replace the old version, but rather add a new legal object within the platform, which references to the former version. User-generated legal objects should stay the same, however, users that have access to these objects should be able to see that changes happened.
- **Languages:** While the platform itself will be available in multiple languages, documents remain in their original language but, if available, links to local language versions will be provided.
- **References:** References and relationships between legal documents should be preserved. Examples are “amended by”, “implemented in”, or “based on”.
- **MetaLex conformance:** To foster the BOLD vision and to integrate the *openlaws* platform into the legal informatics landscape, integration of existing legal standards is imperative. Therefore, one step towards this goal is the integration of *MetaLex* (Engers & Boer, 2011) within the project.
- **Replication and distribution of the data repository:** Although duplication of data during the beginning of the project may be necessary, over the long run, smart linking solutions should be integrated within the platform. In addition, non-destructive updates, fail-safe replication and backups should be implemented.

# 4 Viewpoints of the Architecture

In this section, the architecture is presented from different viewpoints to discuss essential facets imperative to the platforms realization.

## 4.1 Functional view

The functional view presents the systems functional building blocks, concerns and exchanges. It constitutes the foundations for the communication between stakeholders as well as foundations for the succeeding views of the architecture.

Figure 4 depicts a high-level abstract view of the main functional blocks of the *openlaws* platform, organized in horizontal/vertical layers and logical groups according to their functionalities:

- The first layer to be described is **the user interface layer**. This layer comprises the portal for the users to interact with our platform. The Web-based approach ensures accessibility across various operating systems and hardware platforms.
- The next layer is the **business logic layer**. This layer incorporates all components for processing data from the users coming from the layer above as well as data from the data integration layer below. Also data fusion and enrichment processes are located within this layer.
- The **data integration layer** describes the data sources of the *openlaws* platform. These data sources can be physical or virtually included within the infrastructure via an API. For a detailed description, please refer to section 4.2.2.
- The bottom layer consists of the **infrastructure layer.** This layer describes all necessary hardware and software components in terms of infrastructure to host the *openlaws* platform.
- To represent cross-cutting concerns over all layers, two vertical layers are introduced:
    - The **security layer** provides all necessary means to provide secure access to the platform as well as legal and community-related ethical compliance to user interaction as well as stored content.
    - The **community integration layer incorporates** the legal community during the development and evolution of the *openlaws* platform by providing not only regular code-based snapshots, but also features functionalities to handle the community's feedback.

This layered view is a logical representation of the functional building blocks within the architecture. This view serves the purpose of grouping the blocks together and to provide an abstract, lightweight visualisation. The separation into layer, however, does not imply that block from one layer can only communicate within their own layer or only with layer that are next to them.

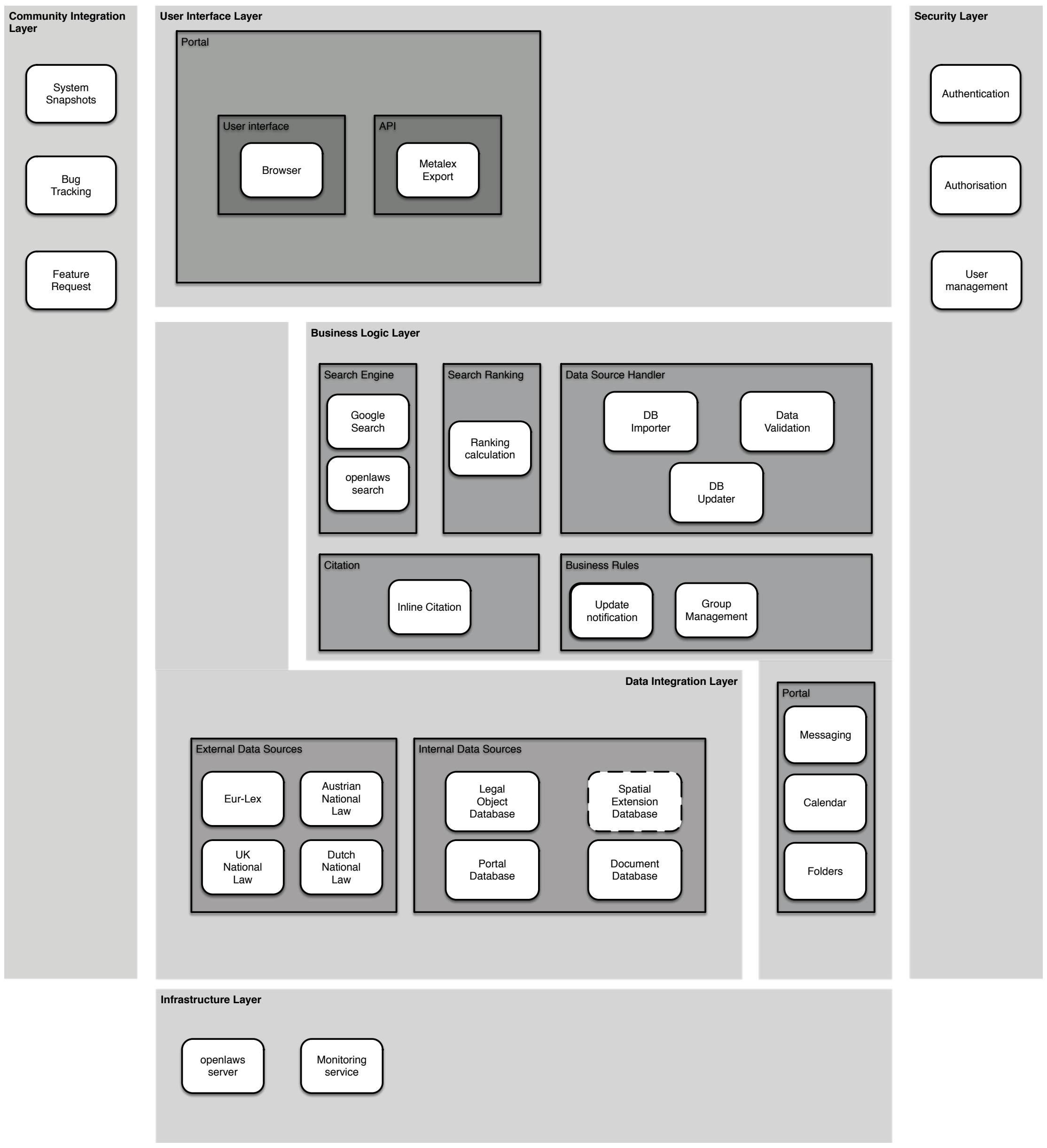


*Figure 4 – Functional viewpoint: abstract visualisation of the openlaws functional components*

The upcoming sub-sections will included a more detailed description about each functionality block.

### 4.1.1 User interface layer

The Portal represents the interaction coordination of the platform. The underlying functionalities are propagated and presented to the end-users via the user interface block. The API block enables external services to query data from the *openlaws* platform in a defined way. Tables 1a-b offer an overview of the functionalities offered from this bock.

| No. | Functionality |
|---|---|
| 1 | **Responsive design to support various access platforms** |
| 2 | **Integrates all directly offered functionalities to a single site** |

| | |
|---|---|
| 3 | **Integrates registration and access controls** |
| 4 | **Integrates collaboration functionalities** |

*Table 1a - User interface layer: offered functionalities by user interface block*

| No. | Functionality |
|---|---|
| 1 | **Integrates a well-defined API to offer data in the Metalex format** |

*Table 2b - User interface layer: offered functionalities by API block*

### 4.1.2 Business logic layer

The Search Engine block represents all search functionalities users have access to via the user interface in the user interface layer. There exist two functional components within this block. The first component includes search results from Google. The second component performs a search on data included within the internal data repositories of the *openlaws* platform. Both result sets will be combined to deliver a comprehensive result experience for the end-users. Table 3 offers an overview of the functionalities offered from this block.

| No. | Functionality |
|---|---|
| 1 | **Present Google search results on a search term** |
| 2 | **Link search result with openlaws meta data** |
| 3 | **Present openlaws search results on a search term** |
| 4 | **Visual incorporation of both result sets** |
| 5 | **Presentation of the combined visualisation to the end-users** |

*Table 3 – Business logic layer: offered functionalities by search engine block*

The **Search Ranking Block** contains functionalities to sort search results from the *openlaws* data repositories in terms of relevance by calculation via a heuristic function. Table 4 offers an overview of the functionalities offered from this block.

| No. | Functionality |
|---|---|
| 1 | **Calculate heuristic for sorting search results** |
| 2 | **Sort search results according to the associated heuristic values** |

*Table 4 – Business logic layer: offered functionalities by ranking calculation block*

The **Data Source Handler Block** comprises functionalities to handle external service data coming to the *openlaws* platform. Tables 5a-c offer an overview of the functionalities offered from this block.

| No. | Functionality |
|---|---|
| 1 | **Batch import from new external sources** |
| 2 | **Incremental update import from already known source** |

*Table 5a – Business logic layer: offered functionalities by DB importer block*

| No. | Functionality |
|---|---|
| 1 | **Validation of imported data to be compliant and consistent with internal data model** |

*Table 5b – Business logic layer: offered functionalities by DB validation block*

| No. | Functionality |
|---|---|
| 1 | **Update internal database with validated data from DB validation block** |
| 2 | **Link new data entry with (if any) previous versions of the legal object** |

*Table 5c – Business logic layer: offered functionalities by DB updater block*

The **Portal Block** incorporates all functionalities associated to user/community interaction and activities. Tables 6a-c offer an overview of the functionalities offered from this block.

| No. | Functionality |
|---|---|
| 1 | **Send and receive private messages to/from other users** |

*Table 6a – Business logic layer: offered functionalities by messaging block*

| No. | Functionality |
|---|---|
| 1 | **Mark important dates/events in private/group calendar** |

*Table 6b – Business logic layer: offered functionalities by calendar block*

| No. | Functionality |
|---|---|
| 1 | **Private/group folders to organize documents** |
| 2 | **Intelligent search folder to contain a pre-defined search query. Updates automatically when new content is imported into the platform** |

*Table 6c – Business logic layer: offered functionalities by folders block*

The **Citation Block** incorporates functionalities to work with legal documents and extract interesting parts (see Tab. 7).

| No. | Functionality |
|---|---|
| 1 | **Reader to open legal documents** |
| 2 | **Possibility to mark parts of the text and export them as citation** |
| 3 | **Citation information refers to the document as well as the page number where the citation has been taken from** |

*Table 7 – Business logic layer: offered functionalities by citation block*

### 4.1.3 Data integration layer

The **External Data Sources Block** includes all external data sources to be embedded during the first initial prototype. A detailed description of the external resources can be found in **Del. 2.1.d1 – Analysis of legal networks**.

The **Internal Data Source Block** includes all internal data sources of the platform. Specific details for the portal-related repository can be found in section 4.2.1 and 4.4.2. In regard to legal objects, please refer to section 4.2.1 and 4.2.2.

### 4.1.4 Infrastructure layer

The Citation Block incorporates functionalities to work with legal documents and extract interesting parts (see Tab. 8).

| No. | Functionality |
|---|---|
| 1 | **Monitoring of system functionalities** |
| 2 | **Logging** |
| 3 | **Search optimization functionalities** |

*Table 8 – Infrastructure layer: offered functionalities by openlaws and monitoring services block*

### 4.1.5 Community integration layer

The functional blocks in the community integration layer provide desired community integration during the development of the platform. Details can be taken from Tab. 9.

| No. | Functionality |
|---|---|
| 1 | **Bug and feature reporting possibilities** |
| 2 | **Basic architectural stack to develop add-on modules and services** |

*Table 9 – Community integration layer: offered functionalities for community engagement and feedback during development*

### 4.1.6 Security layer

The functional blocks in the security layer provide all functionalities regarding security management aspects within the platform (see Tab. 10).

| No. | Functionality |
|---|---|
| 1 | **Authentication of users** |
| 2 | **Authorisation of user actions** |
| 3 | **Handling of user/group management within the platform** |

*Table 10 – Security layer: offered functionalities for security considerations of the platform*

## 4.2 Information view

The information view describes the information flow within the architecture. It is not only described how information comes in and out to the architecture, also information handling between system components is explained.

### 4.2.1 Information flow through architectural layers

The overall information flow of the architecture is presented in Fig. 5 and Fig. 6 respectively. The figures depict the connection of modules, the associated technologies the modules are realized with, as well as the type of interfaces used to connect the components with each other. A description of the particular technologies can be found in section 4.4.2.

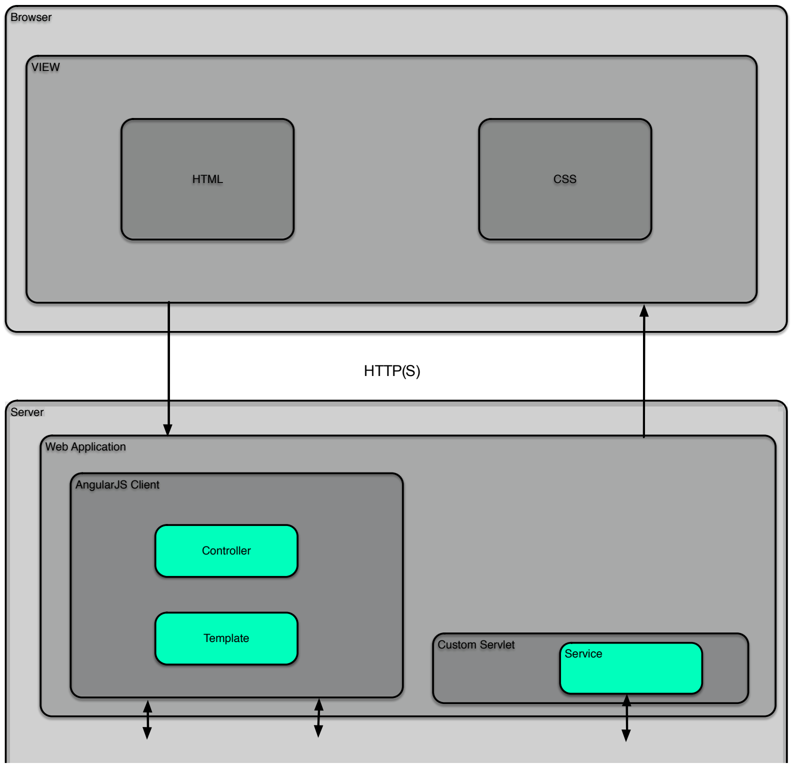


*Figure 5 – Web browser connection to the Web application server*

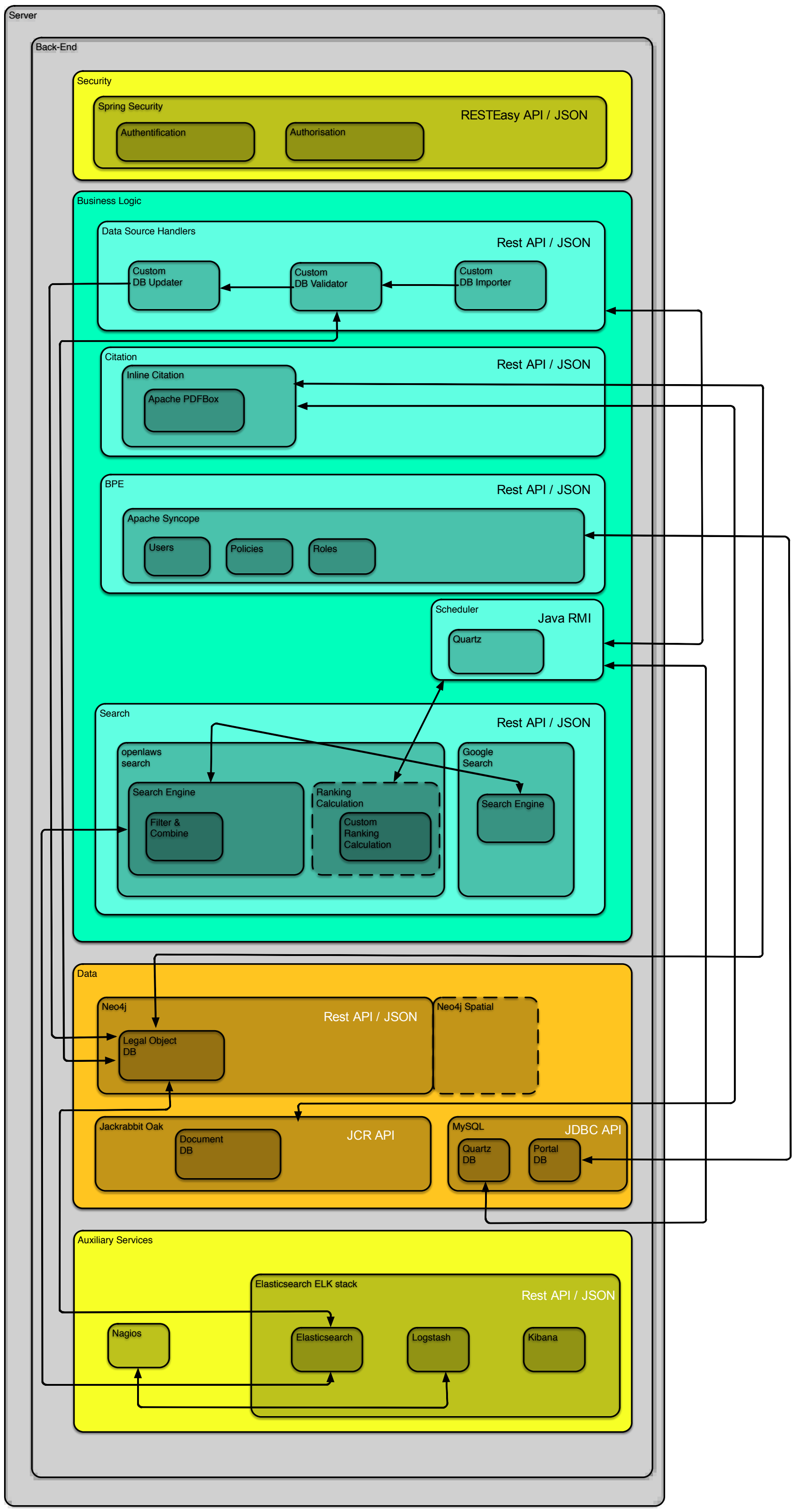


*Figure 6 – Initial draft of the openlaws backend architecture*

### 4.2.2 Preliminary Data Model Description

This section of the deliverable describes in detail the first initial draft of the data model employed within the *openlaws* platform. As the portal-related database schema is already specifically designed to map the requirements of the core module, this part of the deliverable focuses on the design of the graph database holding the open legal data. A schematic overview of the suggest data model can be seen in Fig. 5.

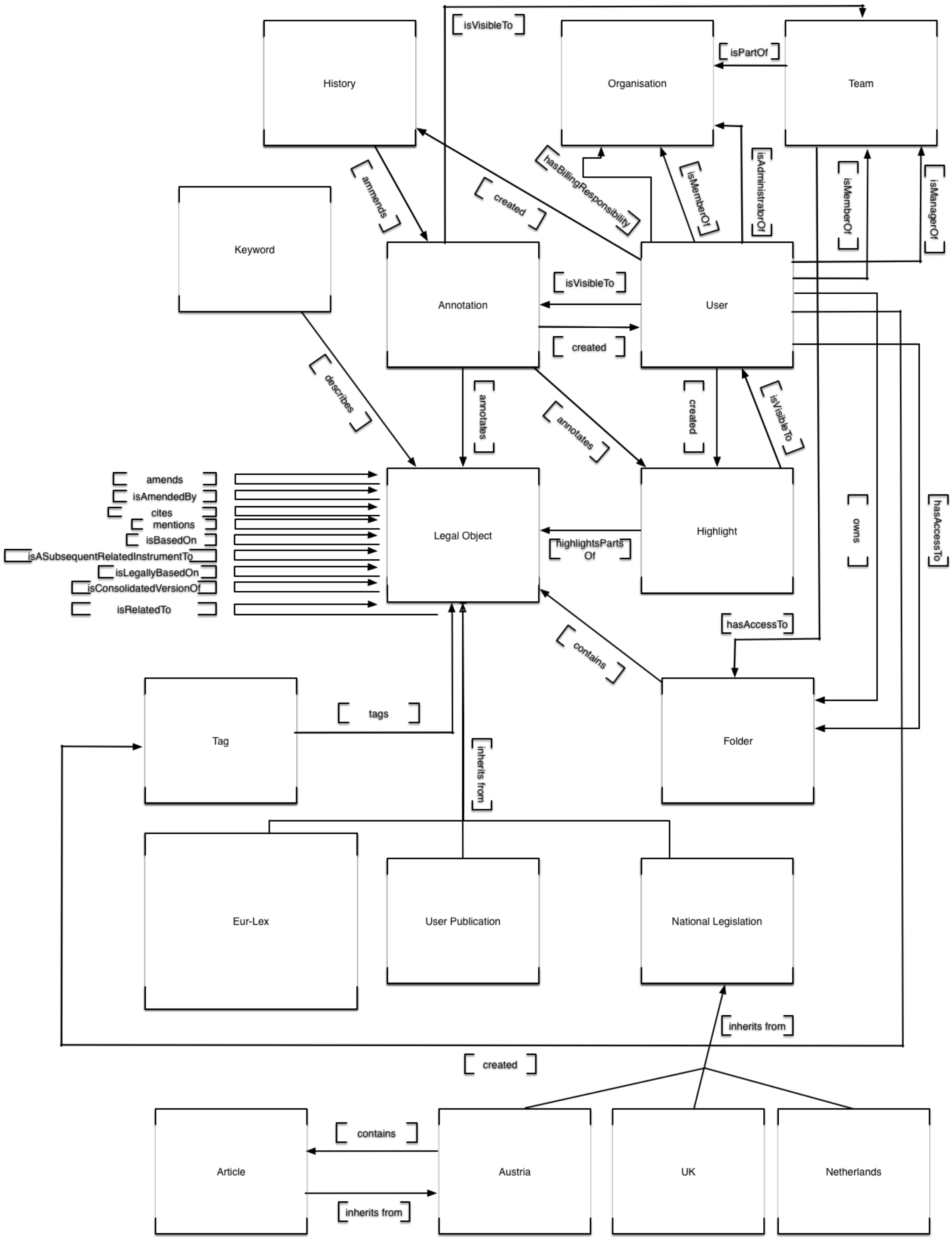
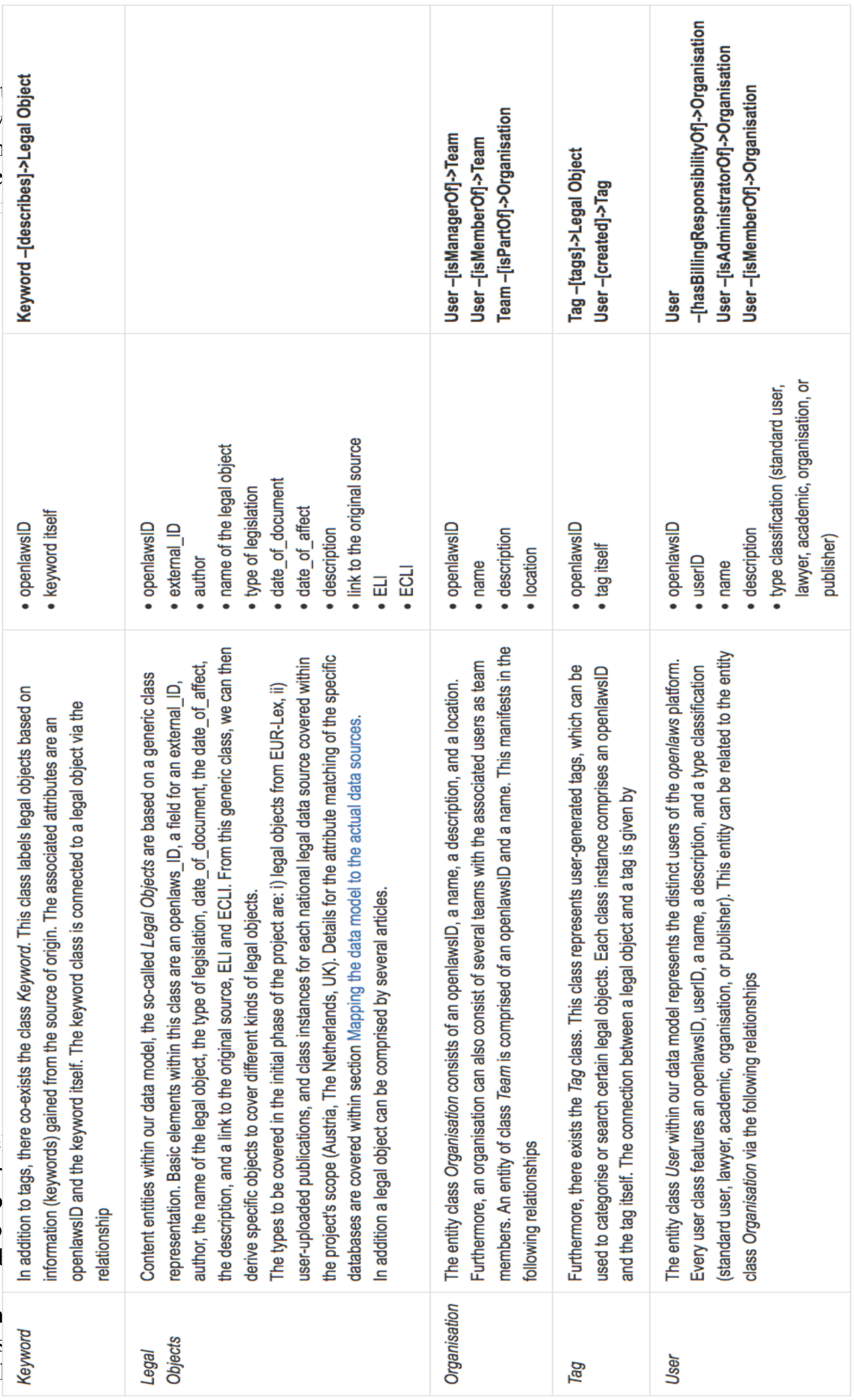


*Figure 7 – Initial draft of the openlaws graph database model*

For a be
now a ty
ters at th
links the
conducti



| | | | |
|---|---|---|---|
| *Keyword* | In addition to tags, there co-exists the class *Keyword*. This class labels legal objects based on information (keywords) gained from the source of origin. The associated attributes are an openlawsID and the keyword itself. The keyword class is connected to a legal object via the relationship | • openlawsID<br>• keyword itself | **Keyword –[describes]->Legal Object** |
| *Legal Objects* | Content entities within our data model, the so-called *Legal Objects* are based on a generic class representation. Basic elements within this class are an openlaws_ID, a field for an external_ID, author, the name of the legal object, the type of legislation, date_of_document, the date_of_affect, the description, and a link to the original source, ELI and ECLI. From this generic class, we can then derive specific objects to cover different kinds of legal objects.<br>The types to be covered in the initial phase of the project are: i) legal objects from EUR-Lex, ii) user-uploaded publications, and class instances for each national legal data source covered within the project's scope (Austria, The Netherlands, UK). Details for the attribute matching of the specific databases are covered within section Mapping the data model to the actual data sources.<br>In addition a legal object can be comprised by several articles. | • openlawsID<br>• external_ID<br>• author<br>• name of the legal object<br>• type of legislation<br>• date_of_document<br>• date_of_affect<br>• description<br>• link to the original source<br>• ELI<br>• ECLI | |
| *Organisation* | The entity class *Organisation* consists of an openlawsID, a name, a description, and a location. Furthermore, an organisation can also consist of several teams with the associated users as team members. An entity of class *Team* is comprised of an openlawsID and a name. This manifests in the following relationships | • openlawsID<br>• name<br>• description<br>• location | **User –[isManagerOf]->Team**<br>**User –[isMemberOf]->Team**<br>**Team –[isPartOf]->Organisation** |
| *Tag* | Furthermore, there exists the *Tag* class. This class represents user-generated tags, which can be used to categorise or search certain legal objects. Each class instance comprises an openlawsID and the tag itself. The connection between a legal object and a tag is given by | • openlawsID<br>• tag itself | **Tag –[tags]->Legal Object**<br>**User –[created]->Tag** |
| *User* | The entity class *User* within our data model represents the distinct users of the *openlaws* platform. Every user class features an openlawsID, userID, a name, a description, and a type classification (standard user, lawyer, academic, organisation, or publisher). This entity can be related to the entity class *Organisation* via the following relationships | • openlawsID<br>• userID<br>• name<br>• description<br>• type classification (standard user, lawyer, academic, organisation, or publisher) | **User –[hasBillingResponsibilityOf]->Organisation**<br>**User –[isAdministratorOf]->Organisation**<br>**User –[isMemberOf]->Organisation** |

he searc
Therefor
to this to
*object* to
topic to l
the new
statemen
and links
est and l
changed his mind slightly, and annotations the document with a comment (creates

*annotation object* and links it to *highlight object* and *user*). Again, one day later, Peter changes some minor aspects in his comments (a *history object* is created and linked to the *annotation object* as well as the *user*). In order to foster the searchability of his document collection, Peter adds several tags to his documents (*tag object* is created and linked to the *legal object* as well as to the *user*).

### 4.2.3 Mapping the data model to the actual data sources

In this section, the before-described data model is mapped to the particular data schemata from the EU and national level databases incorporated into the initial *openlaws* platform. The four databases to be included are: i) Austria, ii) the Netherlands, iii) United Kingdom, and iv) the Eur-Lex platform. Figures 8-11 depict how the generic legal object model is mapped to the attributes of individual databases to be included. In addition, a real-world example is given do demonstrate, how the actual data will look like.

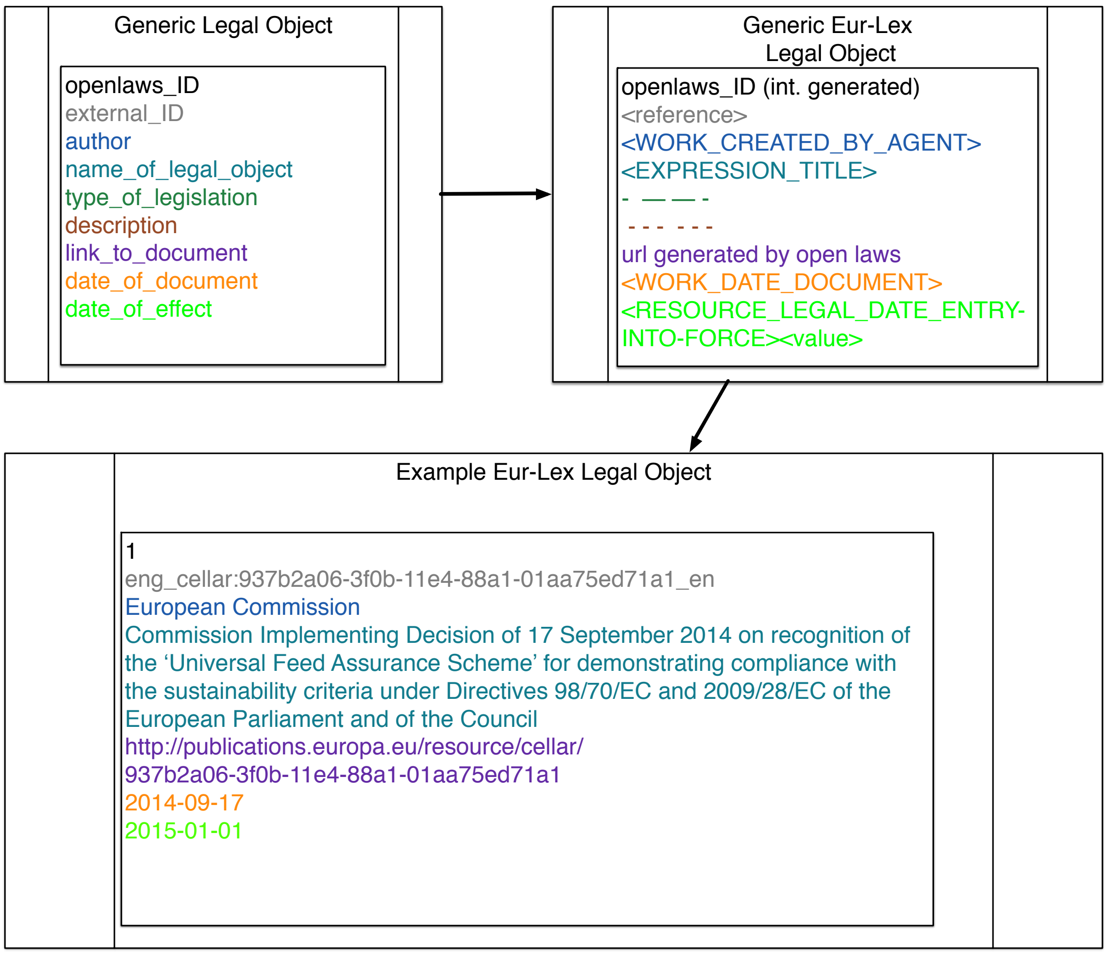

Generic Legal Object
openlaws_ID
external_ID
author
name_of_legal_object
type_of_legislation
description
link_to_document
date_of_document
date_of_effect
Generic Eur-Lex Legal Object
openlaws_ID (int. generated)
<reference>
<WORK_CREATED_BY_AGENT>
<EXPRESSION_TITLE>
- — — -
- - -  - - -
url generated by open laws
<WORK_DATE_DOCUMENT>
<RESOURCE_LEGAL_DATE_ENTRY-INTO-FORCE><value>
Example Eur-Lex Legal Object
1
eng_cellar:937b2a06-3f0b-11e4-88a1-01aa75ed71a1_en
European Commission
Commission Implementing Decision of 17 September 2014 on recognition of the 'Universal Feed Assurance Scheme' for demonstrating compliance with the sustainability criteria under Directives 98/70/EC and 2009/28/EC of the European Parliament and of the Council
http://publications.europa.eu/resource/cellar/937b2a06-3f0b-11e4-88a1-01aa75ed71a1
2014-09-17
2015-01-01

*Figure 8 – Data model mapping of EUR-Lex*

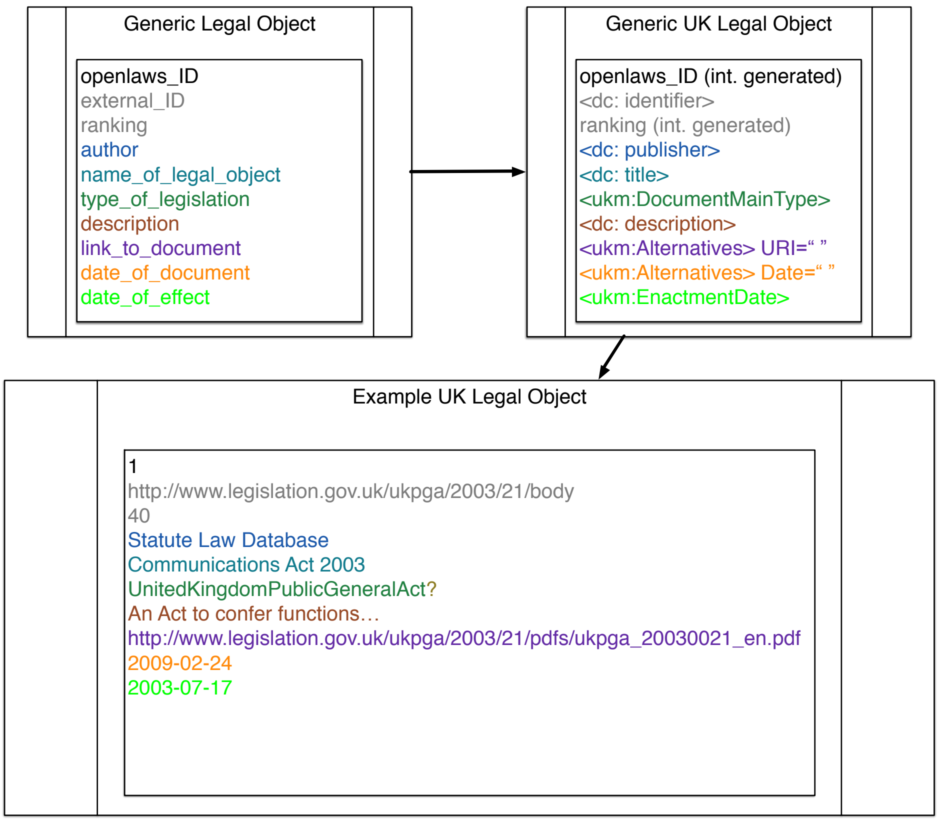


*Figure 9 – Data model mapping of UK*

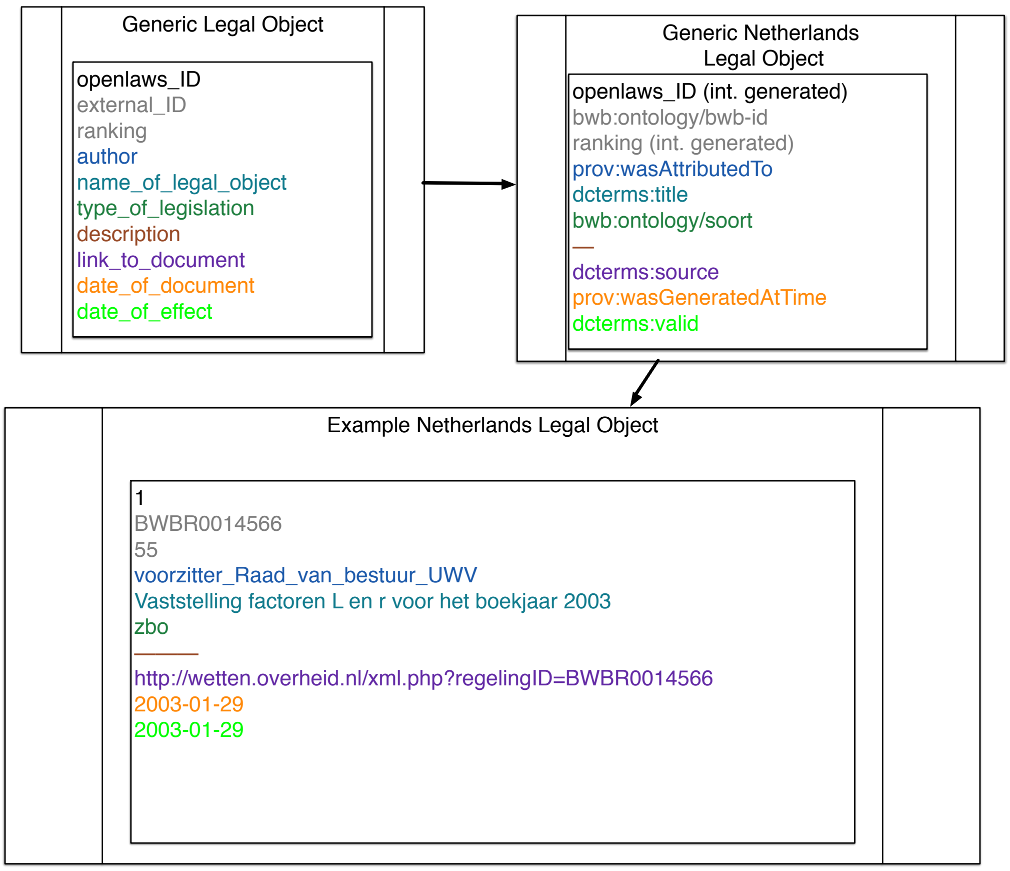
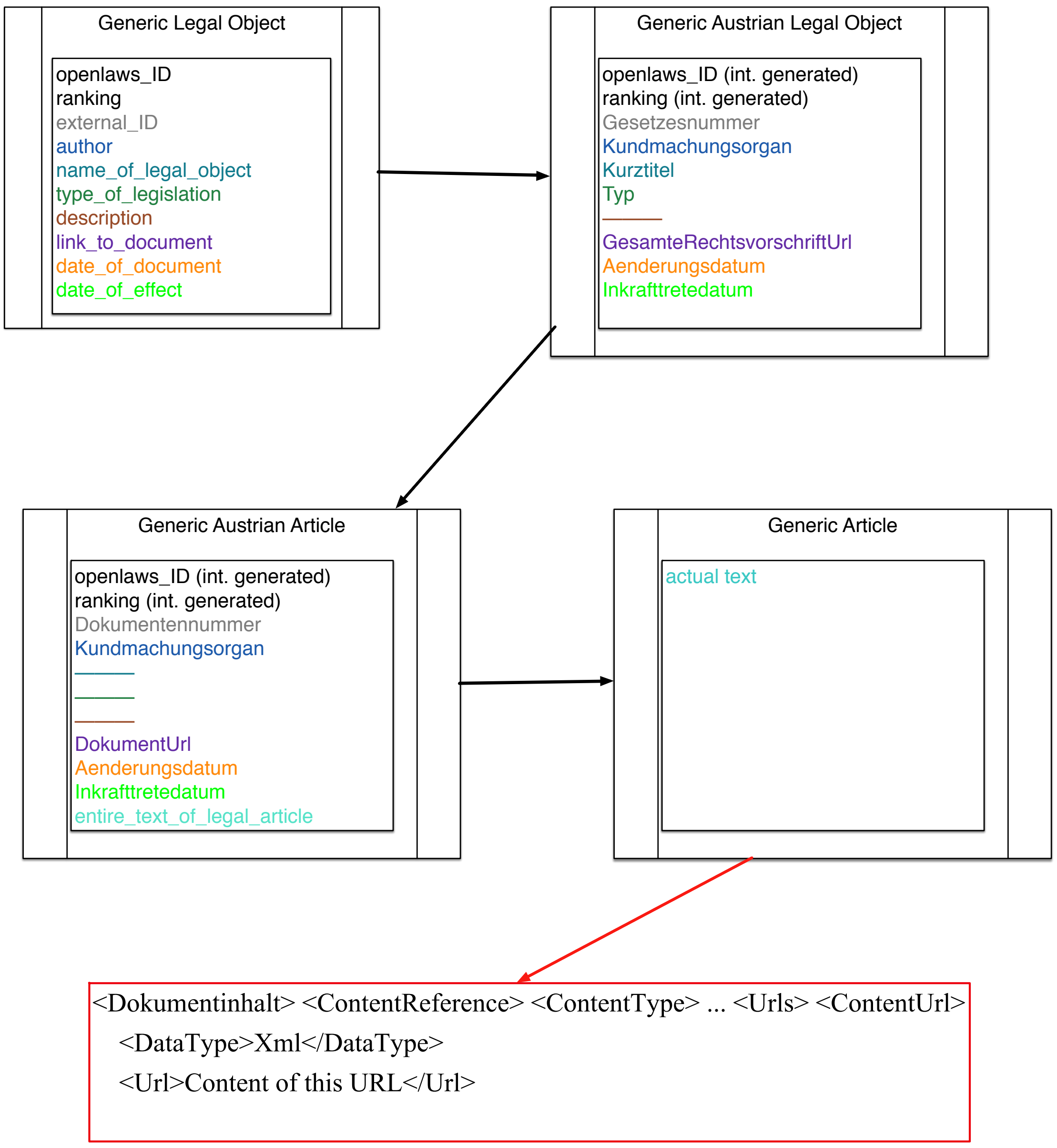


*Figure 10 – Data model mapping of the Netherlands*



```
<Dokumentinhalt> <ContentReference> <ContentType> ... <Urls> <ContentUrl>
   <DataType>Xml</DataType>
   <Url>Content of this URL</Url>
```

*Figure 11 – Data model mapping of Austria*

## 4.3 Development view

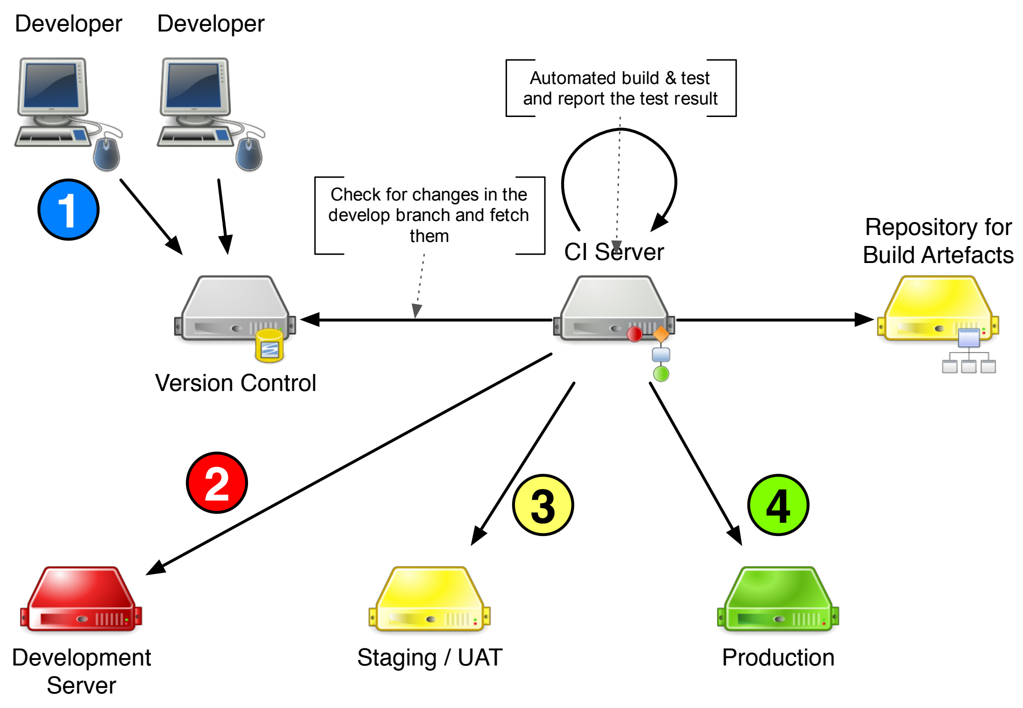
This view is dedicated to process and architectural facets that impact the software development process. These processes comprise building, testing, deployment, maintenance, and continuous development of the entire system. To aim for the highest level possible, the openlaws platform development is based on a combination of i) continuous integration, ii) continuous delivery, and iii) continuous deployment.

The idea of **continuous integration** is to merge output from the current development streams (point 1 in Fig. 12) into the main branch, so adaptations can be tested with other modifications from other streams. A generalisation of this method can be seen in Fig. 13. Potential issues and pitfalls can therefore be detected in a faster way.

An automated build server takes care of compiling the code and of performing the necessary unit tests. This server can be seen in the centre of Fig. 12.

**Continuous delivery** focuses on the delivery away from the developer towards the next steps within the engineering process. This delivery can be towards user acceptance testing (UAT), staging, or production (see points 2-4 in Fig. 12). The main concept here is to involve the user community to check, whether design issues arose during the coding phase, which could not be detected by functional/syntactical testing via unit tests.

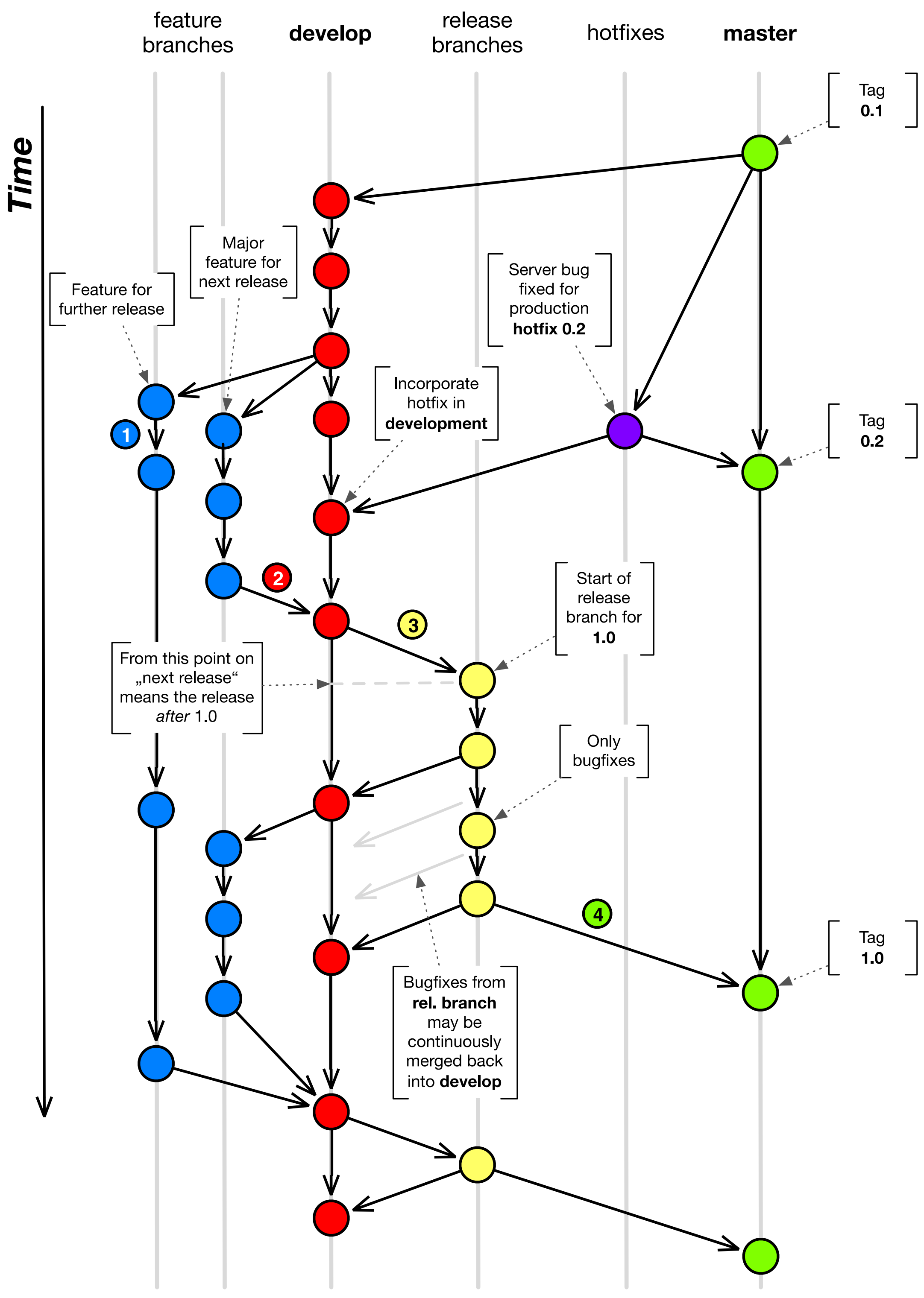
The third approach is **continuous deployment**. When the code is ready for production, it is release as soon as possible. The code on the production machine (point 4 in Fig. 12) is always stable and ready to use. The build mechanism has to guarantee that compilation of this stable version is available fast and that the resulting metrics associated to the build fulfil the business requirements.


Developer
Developer
1
Version Control
Check for changes in the develop branch and fetch them
Automated build & test and report the test result
CI Server
Repository for Build Artefacts
2
3
4
Development Server
Staging / UAT
Production

| Environment | Version control Branch | Comment |
|---|---|---|
| DEV-Computer | feature | |
| Development Server | develop | used for automated continues guid & test |
| Staging / UAT | release | |
| Production | master | |

| Type | Tool | Comment |
|---|---|---|
| Version Control | Git | can be accessed @ git.openlaws.eu |
| CI Server | Jenkins / Hudson | as alternative Bamboo could be used - in case we use all other tools from Atlassian |
| Build | Maven | |
| Repository for build artifacts | Nexus | |

*Figure 12 – Schematics of continuous integration/delivery/deployment*


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Major feature for next release
Feature for further release
Server bug fixed for production **hotfix 0.2**
Incorporate hotfix in **development**
Tag 0.2
1
2
3
Start of release branch for **1.0**
From this point on „next release“ means the release *after* 1.0
Only bugfixes
4
Tag 1.0
Bugfixes from **rel. branch** may be continuously merged back into **develop**

General branching model used for git (http://nvie.com/posts/a-successful-git-branching-model/)

*Figure 13 – Branching and merging phases over the entire development process*

## 4.4 Deployment view

This view upon the architecture defines the environment the *openlaws* platform will be embedded in. This includes on the one hand the physical hardware, on the other hand a description of the actual technologies employed to provided the before described functionalities.

### 4.4.1 Orchestration of hardware

As discussed within the requirements section of this document, scalability and high-availability are key non-functional requirements to the *openlaws* platform. While the schematics in Fig. 14 display the structure of the underlying server environment on the production side, it is possible to fully operate the entire platform on a single server for development purposes. The overall server infrastructure is planned to be redundant so to not allow a single point of failure within the design. Both instances of the Web server are within a demilitarized zone (DMZ), which protects the actual *openlaws* application server and the associated data repository.

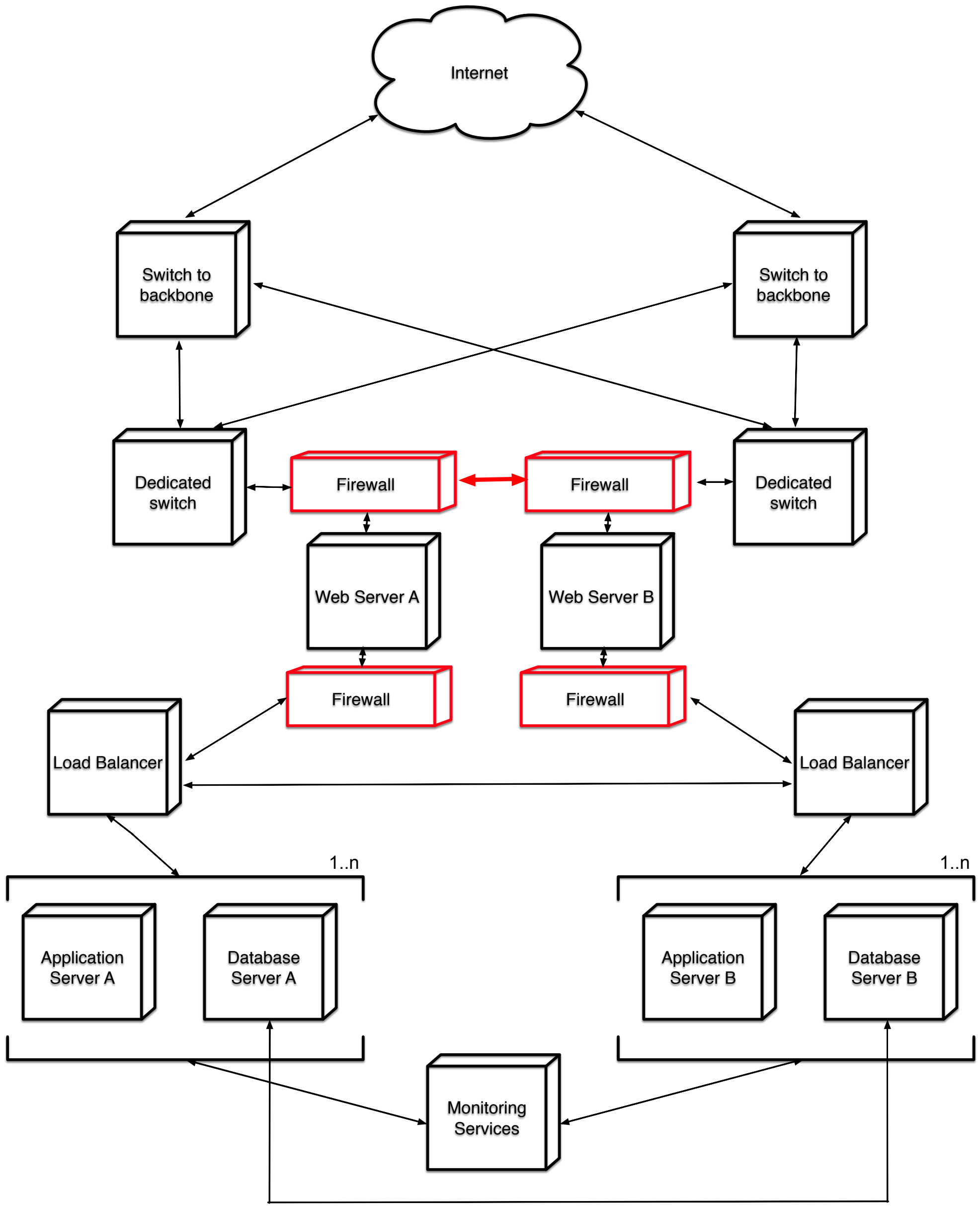


*Figure 14 – Physical server infrastructure to host the openlaws platform*

The in-cooperated load balancers distribute incoming requests equally between the instances of the *openlaws* platform. The monitoring services provide a 24/7 health status of the *openlaws* service so in case of a technical issue, countermeasures can be taken quickly. The servers as such can be provided by virtual machines. This approach provides several benefits. First, rapid deployment of backups is possible. Second, if a higher rate of computational power is needed, the configuration of virtual machines can be easily adapted to match the requirements. Furthermore, virtual machine utilization does not restrict the openlaws platform to one physical server, but allows for distribution over a cloud infrastructure if necessary.

### 4.4.2 Employed software technologies

In this section, the actual technologies employed to provide the functionalities described in the functional view are presented.

| Resource | Mapping |
|---|---|
| **Developer** | Google Inc. |
| **Name** | AngularJS |
| **Software type (Web application, standalone application, service, library, framework etc.)** | Framework |
| **Software License** | MIT License |
| **Operating System** | Cross-platform |
| **Programming Environment** | JavaScript |
| **Communication technologies or protocols to other system components** | HTTP(s)/JSON |
| **URL** | https://angularjs.org/ |

*Table 13 - User interface mapping*

| Resource | Mapping |
|---|---|
| **Developer** | GoPivotal, Inc. |
| **Name** | Spring Security |
| **Software type (Web application, standalone application, service, library, framework etc.)** | Framework |
| **Software License** | Apache License Version 2.0 |
| **Operating System** | Cross-platform |
| **Programming Environment** | Java |
| **Communication technologies or protocols to other system components** | RESTEasy API/JSON |
| **URL** | http://projects.spring.io/spring-security/ |

*Table 14 - Security mapping*

| Resource | Mapping |
|---|---|
| **Developer** | Openlaws consortium |
| **Name** | Data Source Handlers |
| **Software type (Web application, standalone application, service, library, framework etc.)** | Service |
| **Software License** | Apache License Version 2.0 |
| **Operating System** | Cross-platform |
| **Programming Environment** | Java |
| **Communication technologies or protocols to other system components** | REST API/JSON |
| **URL** | --- |

*Table 15 – Data source handler mapping*

| Resource | Mapping |
|---|---|
| **Developer** | Apache Software Foundation |
| **Name** | Apache PDFBox |
| **Software type (Web application, standalone application, service, library, framework etc.)** | Library |
| **Software License** | Apache License Version 2.0 |
| **Operating System** | Cross-platform |
| **Programming Environment** | Java |
| **Communication technologies or protocols to other system components** | REST API/JSON |
| **URL** | https://pdfbox.apache.org/ |

*Table 16 – Citation mapping*

| Resource | Mapping |
|---|---|
| **Developer** | Apache Software Foundation |
| **Name** | Apache Syncope |
| **Software type (Web application, standalone application, service, library, framework etc.)** | Framework |
| **Software License** | Apache License Version 2.0 |
| **Operating System** | Cross-platform |
| **Programming Environment** | Java |
| **Communication technologies or protocols to other system components** | REST API/JSON |
| **URL** | http://syncope.apache.org/ |

*Table 17 – Business process engine mapping*

| Resource | Mapping |
|---|---|
| **Developer** | Terracotta, Inc. |
| **Name** | Quartz |
| **Software type (Web application, standalone application, service, library, framework etc.)** | Library |
| **Software License** | Apache License Version 2.0 |
| **Operating System** | Cross-platform |
| **Programming Environment** | Java |
| **Communication technologies or protocols to other system components** | RMI |
| **URL** | http://quartz-scheduler.org/ |

*Table 58 – scheduler mapping*

| Resource | Mapping |
|---|---|
| **Developer** | Openlaws consortium |
| **Name** | Search engine |
| **Software type (Web application, standalone application, service, library, framework etc.)** | Service |
| **Software License** | Apache License Version 2.0 |
| **Operating System** | Cross-platform |
| **Programming Environment** | Java |
| **Communication technologies or protocols to other system components** | REST API/JSON |
| **URL** | --- |

*Table 69 – openlaws search mapping*

| Resource | Mapping |
|---|---|
| **Developer** | Openlaws consortium |
| **Name** | Ranking calculation |
| **Software type (Web application, standalone application, service, library, framework etc.)** | Service |
| **Software License** | Apache License Version 2.0 |
| **Operating System** | Cross-platform |
| **Programming Environment** | Java |
| **Communication technologies or protocols to other system components** | REST API/JSON |
| **URL** | --- |

*Table 20 – openlaws ranking mapping*

| Resource | Mapping |
|---|---|
| **Developer** | Google, Inc. |
| **Name** | Google Search |
| **Software type (Web application, standalone application, service, library, framework etc.)** | Service |
| **Software License** | Proprietary |
| **Operating System** | Cross-platform |
| **Programming Environment** | --- |
| **Communication technologies or protocols to other system components** | REST API/JSON |
| **URL** | https://developers.google.com/custom-search/docs/overview |

*Table 21 – Google search mapping*

| Resource | Mapping |
|---|---|
| **Developer** | Neo Technology |
| **Name** | Neo4J |
| **Software type (Web application, standalone application, service, library, framework etc.)** | Service |
| **Software License** | Dual-licensed: GPLv3 and AGPLv3 / commercial |
| **Operating System** | Cross-platform |
| **Programming Environment** | Java |
| **Communication technologies or protocols to other system components** | REST API/JSON |
| **URL** | http://www.neo4j.org/ |

*Table 22 – Legal object repository mapping*

| Resource | Mapping |
|---|---|
| **Developer** | Apache Software Foundation |
| **Name** | Jackrabbit Oak |
| **Software type (Web application, standalone application, service, library, framework etc.)** | Standalone application/Web application |
| **Software License** | Apache License Version 2.0 |
| **Operating System** | Cross-platform |
| **Programming Environment** | Java |
| **Communication technologies or protocols to other system components** | JCR API |
| **URL** | http://jackrabbit.apache.org/oak/ |

*Table 23 – Document repository mapping*

| Resource | Mapping |
|---|---|
| **Developer** | Oracle |
| **Name** | MySQL |
| **Software type (Web application, standalone application, service, library, framework etc.)** | Service |
| **Software License** | GPL (version 2) |
| **Operating System** | Cross-platform |
| **Programming Environment** | --- |
| **Communication technologies or protocols to other system components** | JDBC API |
| **URL** | http://www.mysql.com/ |

*Table 24 – Portal repository mapping*

| Resource | Mapping |
|---|---|
| **Developer** | Nagios Enterprises |
| **Name** | Nagios |
| **Software type (Web application, standalone application, service, library, framework etc.)** | Service |
| **Software License** | GPL (version 2) |
| **Operating System** | Cross-platform |
| **Programming Environment** | --- |
| **Communication technologies or protocols to other system components** | Depending on service to be monitored |
| **URL** | http://www.nagios.org/ |

*Table 25 – Auxiliary service mapping 1*

| Resource | Mapping |
|---|---|
| **Developer** | Elasticsearch BV |
| **Name** | Elastic search ELK stack |
| **Software type (Web application, standalone application, service, library, framework etc.)** | Service |
| **Software License** | Apache License Version 2.0 |
| **Operating System** | Cross-platform |
| **Programming Environment** | --- |
| **Communication technologies or protocols to other system components** | REST API/JSON |
| **URL** | http://www.elasticsearch.org/overview/ |

*Table 26 – Auxiliary service mapping 2*

## References

Chung, L., Nixon, B., Yu, E., & Mylopoulos, J. (2000). Non-functional requirements in Software Engineering. Boston: Kluwer Academic.

Engers, van T. & Boer, A. (2011). A MetaLex and metadata primer: Concepts, use, and implementation. In Legislative XML for the Semantic Web, pp. 131-149. Springer.

ISO/IEC/IEEE (2011). Systems and software engineering -- Architecture description. *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)*, pp.1-46.

Liddle, S. W. (2011). Model-Driven Software Development Handbook of Conceptual Modeling (pp. 17-54): Springer.

Martin, R.C. (2003). Agile software development: principles, patterns, and practices. Prentice Hall PTR.

Rozanski, Nick, and Woods, Eoin. Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, 2nd Edition. Addison Wesley, 2011.

Wass, C., Dini, D., Eiser, T., Heistracher, T., Lampoltshammer, T.J., Marcon, G., Sageder, C., Tsiavos, P., & Winkels, R. (2013). "OPENLAWS.EU," in Proceedings of the 16th International Legal Informatics Symposium IRIS 2013, February 21-23, 2013, Salzburg, Austria, 2013, pp. 209-211.