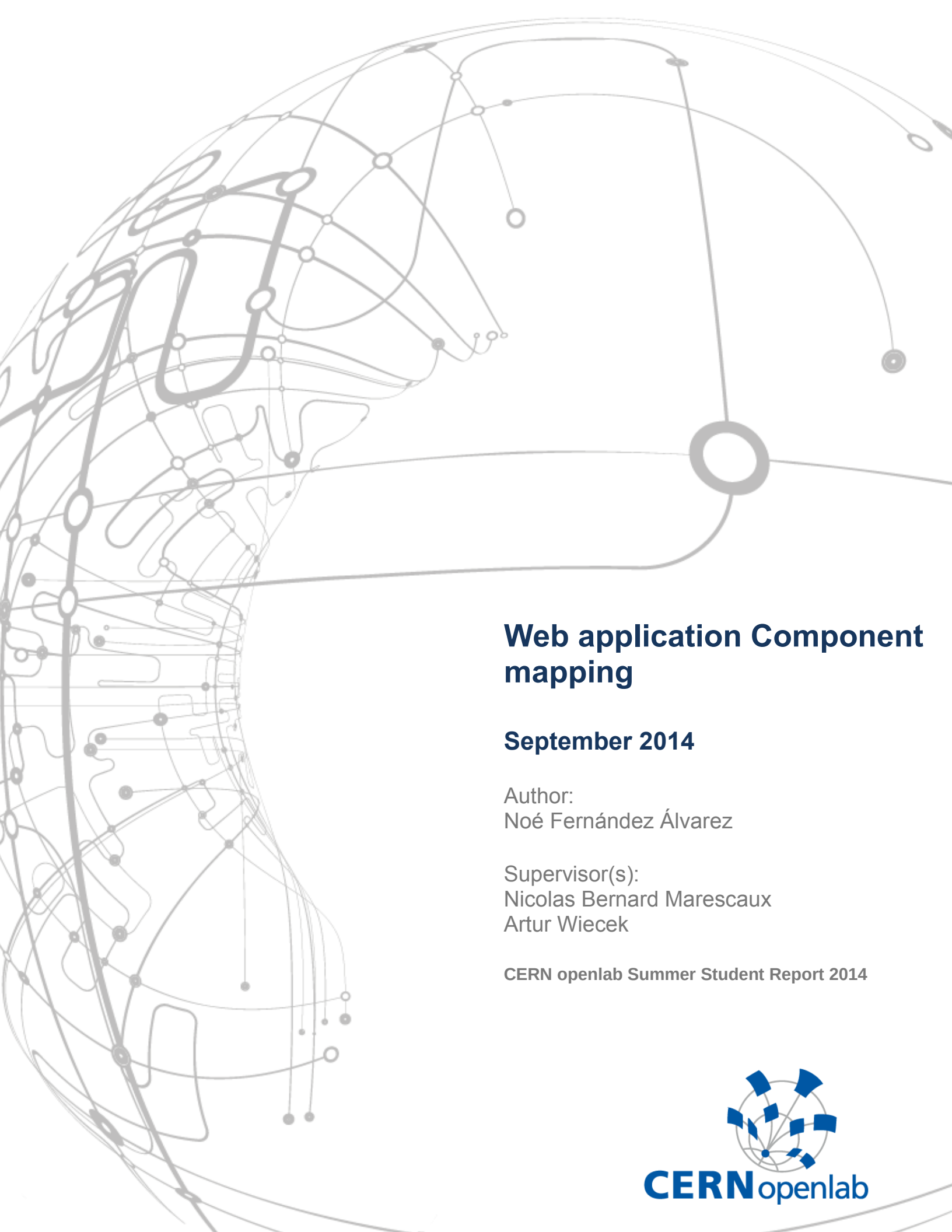


Web application Component mapping

September 2014

Author:
Noé Fernández Álvarez

Supervisor(s):
Nicolas Bernard Marescaux
Artur Wiecek

CERN openlab Summer Student Report 2014

CERN openlab

Table of Contents

Acknowledgments

I would like to express my gratitude to my supervisors, Nicolas Bernard Marescaux and Artur Wiecek, whom solved my doubts every day and gave me lots of ideas for the project. Thanks also to Luis Rodriguez, who was taking care of my work like he was my supervisor as well.

And of course, thanks to the whole *Infrastructure and Middleware Services* section, because you made me feel part of the group from the beginning.

1 Introduction

In this project we want to improve the way that the users checks the information concerning to the systems of **IT/DB** infrastructure. Giving easy-to-read representations of the environment in an optimum time.

1.1 The LDAP Server

All the information concerning the CERN's infrastructure is stored in a LDAP Server, this kind of servers store the information in a tree data structure, where each node of the tree has a set of mandatory attributes and optional attributes.

1.1.1 LDAP Entities

The different nodes of the LDAP stores different information, depending of the type of the entity. In our case, we will focus in the web components.

- **Entities**
- **Interfaces**
- **Web Locations**
- **Hardware**

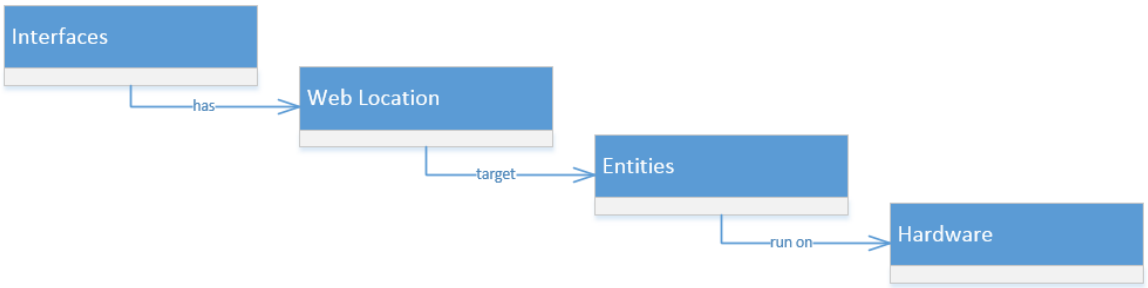


Figure 1 - Relationships between the elements

The **interfaces** represents the HTTP server interfaces, this interfaces may have zero or more **web locations**, and those defines back-end systems running applications served by a specific web server. The **entities** represents the different servers (*application servers, authentication servers, web servers, etc…*), and finally, this entities are running on a specific **hardware.**

1.2 Current tools

Currently, all the configuration files for the systems are generated out of the LDAP server and distributed on all computers via SVN to ensure that the configuration is always available, even if the LDAP server is not reachable.

To check all the information about the systems, we have a small set of tools that reads this local configuration files and shows information based on the content of this files.

2 Project overview

2.1 Why this project is worth?

When you are working with dozens of servers and several types of configurations, it´s really important to know all the information concerning to every server in order to make changes, fix problems, etcetera…

If the environment where you are working is very big, it´s a waste of time to search for all this information because the data repositories doesn´t has a human-friendly format and it´s really difficult to understand and read. Furthermore, if new people is starting to work in our environment, it´s quite difficult for them to understand the structure of the environment and how to read this information.

So we need a tool to process all this information, to make a standard and easy-to-read representation of the environment but giving the same details than the stored information, saving time and increasing the comprehension of the system.

2.2 Objectives

- **Offer real-time information**
  The information about the systems must be offered in real-time in order to give the most updated information about the different components.

- **Easy-to-read information**
  The information must be offered in a friendly format to the users in order to make easier the comprehension of the environment, but keeping the original structure of the data.

- **Multiple ways to retrieve the information**

- **API for external services**
  The information must be available for other programmers in order to create new applications based on this data.

2.3 Involved technologies

- **Java EE**
  Java Enterprise edition is the most used platform to develop web applications offering a good performance, a clear syntax and a huge API for all the purposes.

- **Mojarra (Java Server Faces)**
  Mojarra is the Apache implementation of the JSF definition, it´s a language to define the user interface and the interaction with the backend.
    - **PrimeFaces**
      PrimeFaces is an extension of JSF and it offers powerful components than the basic JSF implementations. There are more extensions like **RichFaces**, **IceFaces**, **BootFaces**, **etc…**
- **DOT Language**
  The DOT language is a Domain specific language created by GraphViz to generate graphs structures. DOT Language allows you to define every component of the graph (the shape, the colour, the label…)
- **Apache Tomcat**
  Apache tomcat (a.k.a. Tomcat) is a web server and a servlet container developed by the Apache Software Foundation. Tomcat provides their own implementation of the Oracle´s JSP and Servlet standards and makes very easy the deployment and configuration of applications.

  Furthermore, CERN IT-DB group has developed a cloud infrastructure that provides tomcat servlet containers allowing the users to manage their own applications through a web interface.

3 System analysis

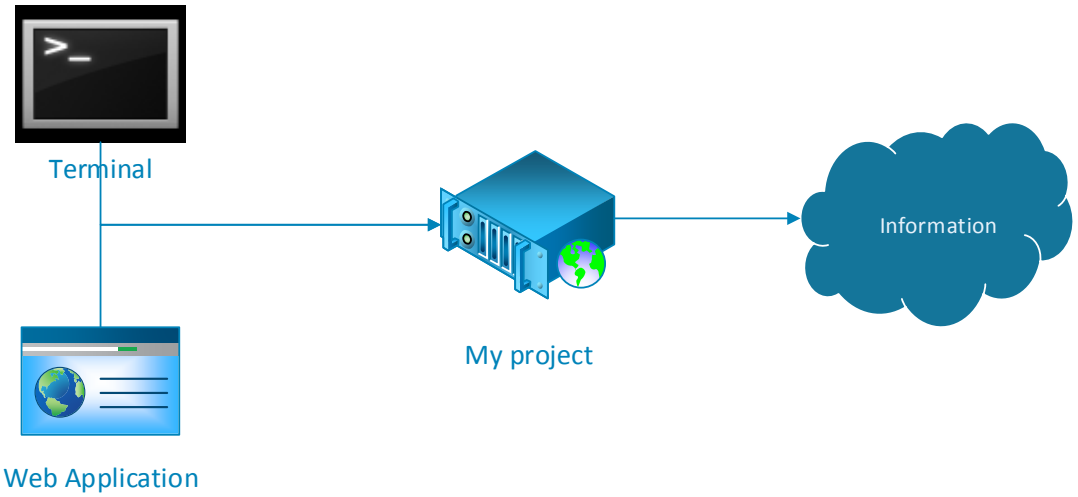


Figure 2- General architecture

3.1 System design

Based on the use cases and the environment where the application is going to work, I chose a typical 3-*layer* architecture where each layer has their own purpose allowing the programmer to isolate the different responsibilities, those are:

- **Data access**
- **Graph construction**
- **User interaction**

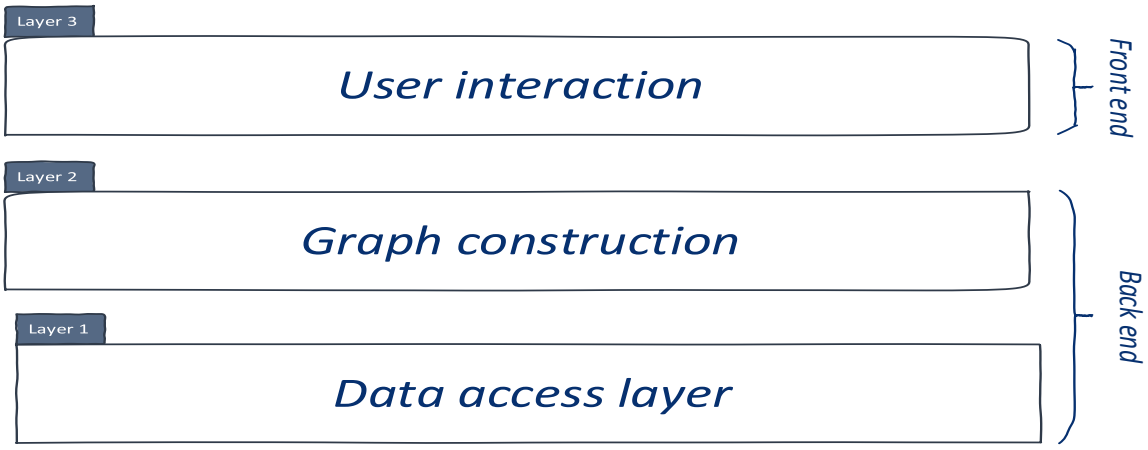


Figure 3 - Architecture overview

3.1.1 Data access layer

The data access layer is responsible of retrieve the appropriate data from the data source and offers an interface to the next layer.

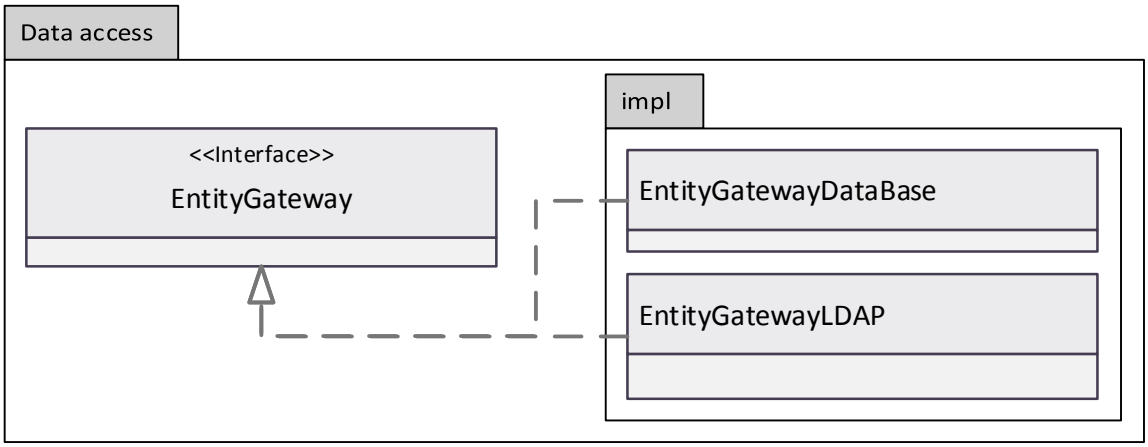


Figure 4- Gateway design overview

This design allows the programmer to use different kind of data sources in the same application because the interface for the next layer is the same, the logic for the different types of data sources is located in the implementation.

This different implementations are accessed by the next layer with a factory, where you can define the implementation for each interface

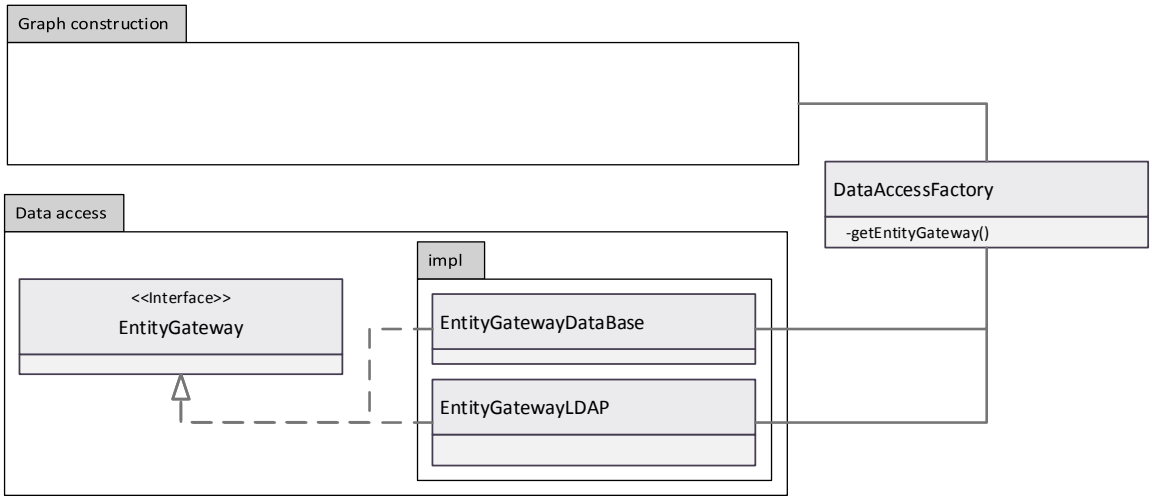


Figure 5- Relationships between the data access layer and the service layer

For example, to use a connector with a database:

```
public EntityGateway getEntityGateway() {
      return new EntityGatewayDatabase();
}
```

With an LDAP Server

```
public EntityGateway getEntityGateway() {
      return new EntityGatewayLDAP();
}
```

3.1.2 Graph construction

The **graph construction layer** is encouraged to identify and execute the proper algorithm in order to generate the representation of the desired environment.

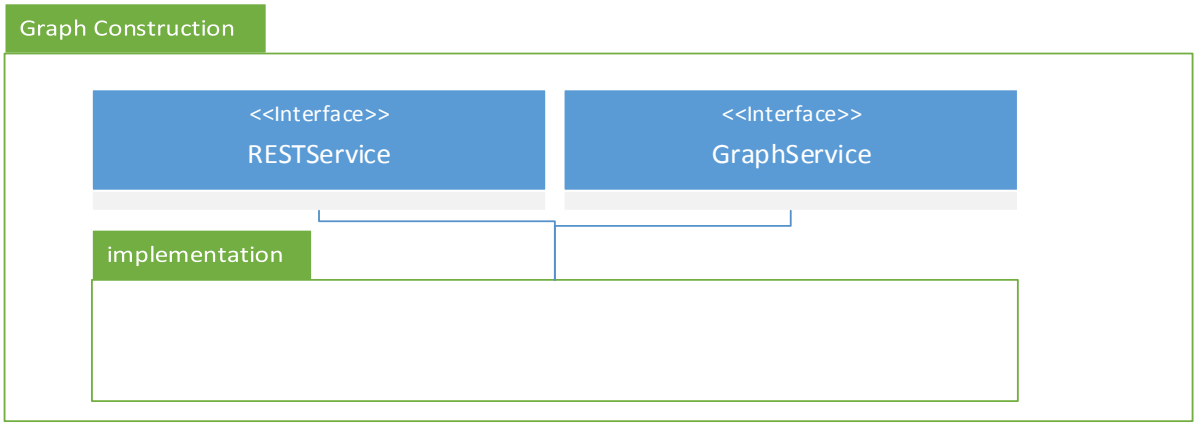
This algorithms are divided in two types:

- **Forward algorithm**: Obtain the systems from an specific URL
  - Tomcat forward algorithm
  - WebLogic forward algorithm

- **Reverse algorithm**: Obtain all the related systems with a specific system.
  - Tomcat reverse algorithm
  - WebLogic reverse algorithm
  - Apache reverse algorithm

As each algorithm has common functionality with other algorithms, I divided the algorithms in small and independent pieces of code called ***Executors***, each executor is responsible of one task and you can use the same executor in different parts of the code writing only the specific code for each algorithm, making easier the construction of new type of algorithms.

Furthermore, the graph construction layer defines the different types to access the data, in our case, we have a REST API and access via JavaBeans.



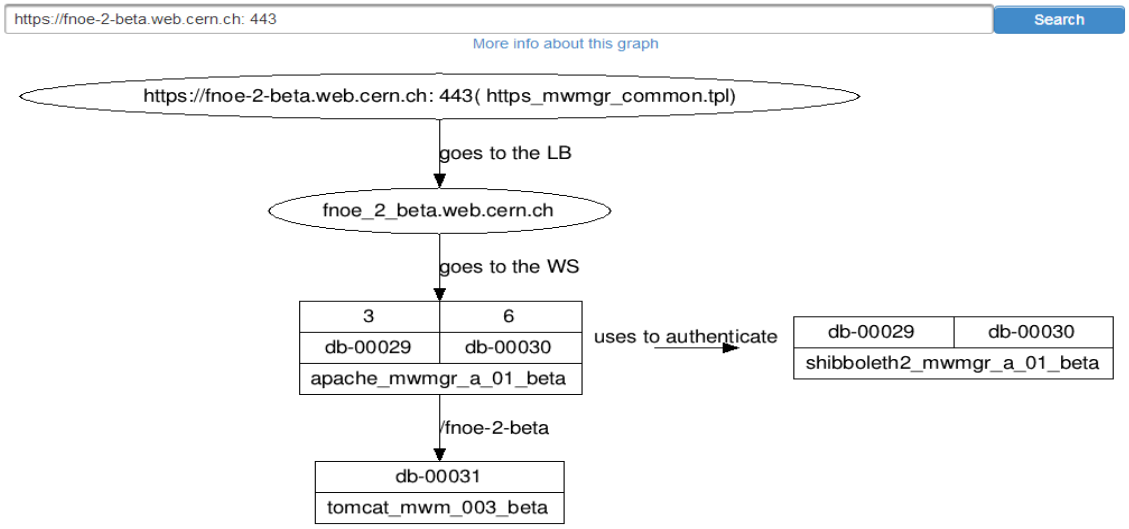
The RESTService interface offers the functionality for the request via HTTP and the GraphService interface is encouraged to manage the petitions from the web interface (via JavaBeans)

3.1.3 User interaction

The **User interaction layer** communicates the web user interface with the backend of the application and offers all the dynamic functionality of the web page.

3.2 User interface

3.2.1 Web interface



3.2.2 Shell tool

4 Future work

For the future, I suggest to complete the LDAP schema with the missing web locations in order to make easier the development of new features and tools. Moreover, all the entities of the same type in the LDAP should have the same basic attributes (i.e.: The interfaces of the applications running on web logic servers has an attribute for the load balancer, whereas the applications running on apache don´t have this attribute)

5 Results

As result of my work, I created a scalable and expandable applications that offers the information about the different environment in a reasonable time. This information generated by the application is available both in the web application and via REST.

The REST access allows to other programmers to make new types of applications based on the generated data, which is useful to extend the functionality without change the code of the original application.