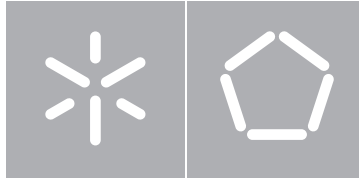




Universidade do Minho
Escola de Engenharia

Pedro José Ribeiro Moreira
Knowledge Analytics

Outubro de 2014



Universidade do Minho

Escola de Engenharia
Departamento de Informática

Pedro José Ribeiro Moreira
Knowledge Analytics

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de
Professor Pedro Rangel Henriques
Engenheiro Miguel Grade

Outubro de 2014

ACKNOWLEDGEMENTS

There are a number of people without whom this thesis might have not been written to whom I am deeply grateful.

First of all, I would like to thank the companies that provided the opportunity of developing this master work in a professional environment, Inova Ria and Maisis, without whom the internship would no be possible.

Next, I would like to thank my supervisors, Prof. Pedro Rangel Henriques and Eng. Miguel Grade, whose supervision, knowledge and guidance was fundamental the success of the project.

In addition, I would like to thank my friends, college and work colleagues for helping me when I needed.

I would specially like to thank my parents and my sister, for supporting me during this stage of my life.

Last but not least, I would like to thank my girlfriend, who was always there for me.

ABSTRACT

This document consists in a thesis report for a master work on the area of *Knowledge Analytics*. This thesis is the main component of the second year of the masters degree in Informatics Engineering at University of Minho and it was developed in collaboration with *Maisis - Information Systems*, from Aveiro, Portugal.

The main goal of this thesis was to study the knowledge management platform Oobian, property of *Maisis* and to develop and integrate software modules to extract, analyse, compute and disseminate information present in a knowledge database.

RESUMO

Este documento consiste num relatório de tese na área de *Análise de Conhecimento*. A tese é o componente principal do segundo ano do *Mestrado em Engenharia Informática* da Universidade do Minho. O trabalho de tese foi desenvolvido em colaboração com a *Maisis - Information Systems*, sediada em Aveiro, Portugal.

O principal objetivo deste trabalho de tese foi fazer um estudo sobre a plataforma de gestão de conhecimento *Oobian*, propriedade da *Maisis*, desenvolver e integrar módulos de software de modo a permitir a extração, análise, processamento e disseminação de informação presente na base de conhecimento.

CONTENTS

Contents	iii
Acronyms	v
List of figures	vi
List of tables	viii
1 INTRODUCTION	1
1.1 Context and Problem	1
1.2 Semantic Web, Ontology & Linked Data	2
1.3 Data Analytics & Visualization	7
1.4 Motivation	8
1.5 Goals	8
1.6 Contribution	9
1.7 Methodology	9
1.7.1 Development Methodology	10
1.8 Work Plan	10
1.9 Document Structure	12
2 KNOWLEDGE MANAGEMENT AND VISUALIZATION	13
2.1 Oobian platform	13
2.1.1 Oobian architecture	14
2.1.2 Oobian features	14
2.2 Data Analysis and Visualization tools	20
2.2.1 Sgvizler	20
2.2.2 Google Charts	21
2.2.3 IBM Many Eyes	22
2.2.4 Microsof Silverlight PivotViewer	23
3 PROPOSED SOLUTION	24
3.1 Requirement Specification	24
3.1.1 Analysis Methodology	24
3.1.2 Stakeholders	25
3.1.3 Actors	25
3.1.4 Requirements	27
3.2 Proposed Architecture	29
3.3 Query Format	30
3.4 Output Format	31

Contents

3.4.1	CSV Table	33
3.5	Embed component	33
4	DEVELOPMENT	35
4.1	Client side	35
4.1.1	Component Integration and Configuration	41
4.2	Server side	43
4.3	Visualization Embedding	46
5	ACHIEVED RESULTS	49
5.1	560.PT - Portugal Business Network	49
5.2	Joobian	56
6	CONCLUSION	62
6.1	Future Work	63
A	REQUIREMENT SPECIFICATION	67
A.1	Functional Requirements	68
A.2	Non-Functional Requirements	79
A.2.1	Usability	79
A.2.2	Design	79
A.2.3	Interface	80
A.2.4	Physical	80

ACRONYMS

API	Application Programming Interface
CEO	Chief Executive Officer
CMS	Content Management System
CSV	Comma Separated Values
FURPS	Functionality, Usability, Reliability, Performance, Supportability
GTD	Getting Things Done
HTML	Time Division Multiple Access
HTTP	HyperText Markup Language
JAX	Java Specification Request
JSF	Java Server Faces
JSON	JavaScript Object Notation
OWL	Web Ontology Language
OLW-DL	Web Ontology Language Description Logic
RDF	Resource Description Framework
RDFS	Resource
REST	Resource Description Framework Schema
SPARQL	SPARQL Protocol and RDF Query Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
XHTML	eXtensible Hypertext Markup Language
XML	eXtensible Markup Language

LIST OF FIGURES

Figure 1	Evolution of Web	2
Figure 2	Ontology hierarchy	4
Figure 3	Linking Open Data cloud - April 2014	6
Figure 4	Agile development process	11
Figure 5	Oobian architecture	14
Figure 6	Oobian Content Sources	15
Figure 7	Content Processing Pipeline	15
Figure 8	Oobian HTML client - Instances of a Class	16
Figure 9	Oobian HTML client - Instance details	17
Figure 10	Oobian HTML client - Relation details of an instance	17
Figure 11	Oobian HTML client - Faceted filtering	18
Figure 12	Oobian HTML client - Maps mode	19
Figure 13	Sgvizler html element example	20
Figure 14	Google Charts example	21
Figure 15	IBM Many Eyes visualization example	22
Figure 16	Silverlight PivotViewer example	23
Figure 17	Use case diagram	26
Figure 18	Architecture Block Diagram	29
Figure 19	JSF architecture	36
Figure 20	Knowledge Analytics composite component	36
Figure 21	Analytics panel	37
Figure 22	Extract from <i>Knowledge bean</i> : managed properties	38
Figure 23	Extract from <i>Knowledge bean</i>	39
Figure 24	JavaScript function to draw a visualization	40
Figure 25	Extract from the configurations file	41
Figure 26	REST web service interface	44
Figure 27	Server side block diagram	45
Figure 28	Popup with the embedding code	46
Figure 29	getElements function	47
Figure 30	Extract from the drawFromEmbed function	48
Figure 31	560.pt class hierarchy	50
Figure 32	Area chart: N ^o of companies / Creation date	51
Figure 33	Pie chart: % of companies / N ^o employees	52

List of Figures

Figure 34	Bar chart: N ^o of companies / Ratio of companies with 500+ employees	53
Figure 35	Donut chart: Relations of instance Portugal	54
Figure 36	Pie chart: family of Services related to Portugal	55
Figure 37	Joobian class hierarchy	56
Figure 38	Bar chart: 15 cities with more Job Offers	57
Figure 39	Column chart: 15 cities with more <i>Job Offers</i> in Portugal	58
Figure 40	Line chart: Evolution <i>Job Offers</i> matching Java in Porto	59
Figure 41	Table: 15 cities with more <i>Job Offers</i> in Portugal	60
Figure 42	”Dashboard” page with multiple charts	61
Figure 43	Analytics button mockup	68
Figure 44	Instance relations example	69
Figure 45	Related instance cluster example	70
Figure 46	Simple Data Table	71
Figure 47	Bar Chart	71
Figure 48	Column Chart	71
Figure 49	Line Chart	72
Figure 50	Area Chart	72
Figure 51	Pie Chart	72
Figure 52	Donut chart	72
Figure 53	Proposed grammar	75
Figure 54	Embed Html popup mockup	76

LIST OF TABLES

Table 1	Requirement Priority	27
Table 2	Functional requirements	28
Table 3	Non-Functional requirements	28
Table 4	Chart Restrictions	73

INTRODUCTION

In this first chapter of the report is made an introduction to the master work. It is exposed to the reader the main problem and presented a contextualization of the work, as well as the goals intended to be accomplished with it. It is explained the motivation behind the project from the point of view of the company as well as the student and it is also presented a general work plan and the working methodology adopted.

1.1 CONTEXT AND PROBLEM

In the context of the Master degree in Informatics Engineering of University of Minho, this master work is the main component of the second and final year. The work is supported by the InovaRia's Genius Trainee Program¹, and was developed in partnership with Maisis - Information Systems² from Aveiro, Portugal.

Maisis is a software development company, founded in 1994, whose focus is to provide products and solutions for the telecommunications market. One of those products is the Oobian³ ecosystem.

Oobian is a knowledge management platform that extracts and indexes information from structured and unstructured data inside and outside organisations, transforming it into a real knowledge database. The users of the platform can search and drill through the information in a rich user interface that creates a unique navigation experience. With this work, Maisis intends to take the next step and add features to the platform aiming to allow companies using Oobian to perform deeper data and knowledge analytics and improve their business intelligence processes.

The main problem this work proposes to solve is the lack of a formal method to extract and process information for analysis and report features. The platform already has formal methods to structure and present data from ontologies, however it doesn't have a formal solution to query the server and obtain a structured data set of results prepared for analysis. So, the general goals of this work are to specify a query format, to specify a format for

1 <http://www.bolsasgenius.pt/>

2 <http://www.maisis.pt/>

3 <http://www.oobian.com/>

1.2. Semantic Web, Ontology & Linked Data

the resultant data set and to make an analysis by generating reports about the information present in those results.

1.2 SEMANTIC WEB, ONTOLOGY & LINKED DATA

As said by Tim Berners-Lee, semantic web is "a new form of web content that is meaningful to computers" [4].

On the last few years the web went through great changes, it has gone from being a company focused information portal (web 1.0) to a social platform (web 2.0). Before, the majority of the information on-line was created by companies; now, the focus is on contents created by social communities and networks. This change led to a massive growth of the amount of information present on the web.

In spite of these major changes, the current web is still mostly based on old keyword search mechanisms, but it is changing, evolving once again. Semantic Web (web 3.0), the web of data is that evolution. From unstructured information to a real knowledge database. On figure 1 is represented the evolution of the web[7].

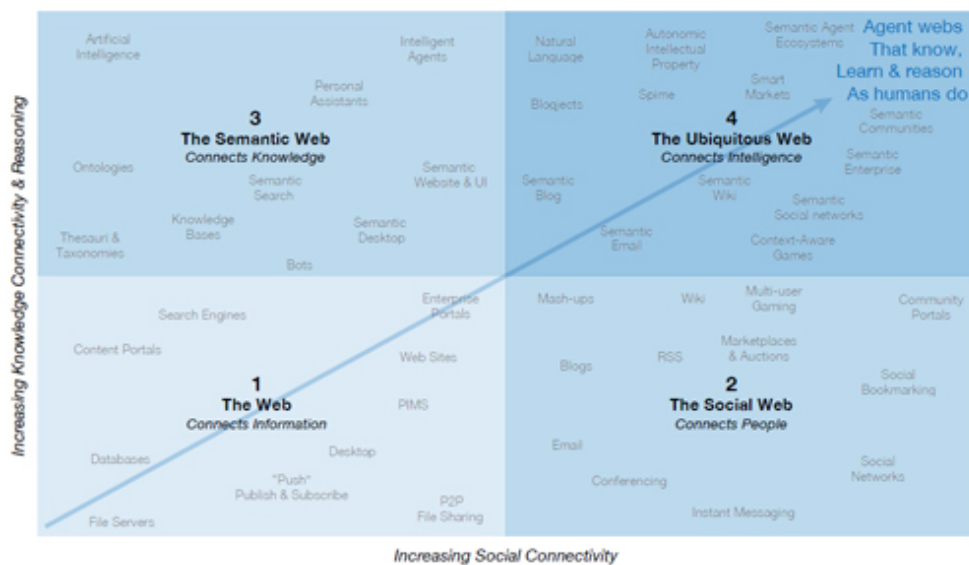


Figure 1.: Evolution of Web

1.2. Semantic Web, Ontology & Linked Data

On the basis of Semantic Web is a shift of the computational pattern, from a information centred to a knowledge centred one, enabling people and machines to connect, share and use knowledge.

The main goal of Semantic Web is to give "meaning" to the information present in the web [4], inserting machine-readable meta-data into web contents about them and how they are related to each other. By doing this it's possible to create autonomous agents that can extract and process relevant information and perform tasks on behalf of the users.

But how do these agents recognize the objects and relationships between them? Is there any standard? How can they deal with the diversity of web contents? The answer to these questions is the use of ontologies.

In philosophy, an ontology is a theory about the nature of existence, about types of things that exist. Web researchers adopted this term and for them an ontology is "formal explicit description of concepts in a domain of discourse"[13], a document or file that defines formally a set of types, properties and the relationship between types. In both computer science and philosophy, ontologies have in common the representation of entities, ideas or events, along with their properties and relations, according to a system of categories. One of the main advantages of ontologies is the fact that regardless of the language in which they are expressed, contemporary ontologies share a set of common components [12].

These components are:

- Individuals: the basic objects or instances.
- Classes: groups of individuals that belong together and share properties.
- Attributes: characteristics, properties or parameters that objects or classes can have.
- Relations: ways in which classes and individuals can be related.
- Function terms: structures with a high degree of complexity that can replace an individual in a statement.
- Restrictions: formal descriptions of what must be true in order for some assertion to be accepted as input.
- Rules: *if-then* sentences that describe logic.
- Axioms: assertions in a logical form that contain the overall theory of the ontology.
- Events: the changing of attributes or relations.

Typically, a web ontology has a taxonomy and a set of inference rules. The taxonomy is used to define classes of objects and the hierarchical relations between them. A car may be defined as a type of vehicle, and number plates may be defined as a type of identification, to

1.2. Semantic Web, Ontology & Linked Data

apply only to vehicles. This definition of classes, subclasses and relations is very useful as a large amount of knowledge can be expressed by assigning properties to classes and allowing subclasses to inherit those properties. Figure 2 depicts a simple representation of the hierarchy of a knowledge base.

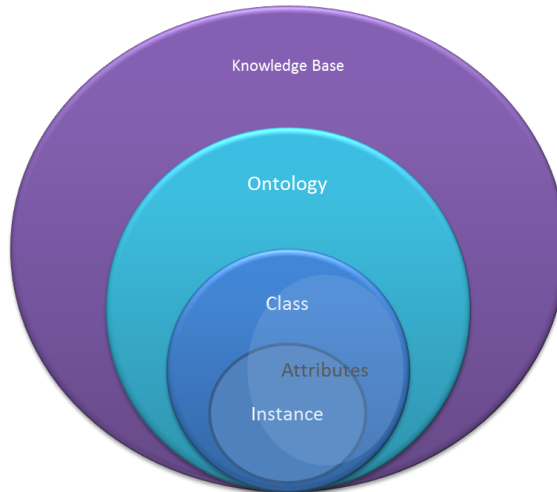


Figure 2.: Ontology hierarchy

There are some specific languages to encode ontologies. In the web context, the main ontology language is OWL - Web Ontology Language.

The W3C Consortium describes OWL as being the choice of use when "the information contained in documents needs to be processed by applications, as opposed to situations where the content only needs to be presented to humans"⁴. OWL explicitly represent the meaning of terms in vocabularies and the relations between them, and has a set of characteristics like cardinality, equality, symmetry (and other characteristics of properties), that turn it ideal to describe properties and classes.

There are three sub languages of OWL [15]:

- OWL Lite: it is the simplest, providing only the basics, for example, supports cardinality, but only permits cardinality values of 0 or 1. It is used for quick migration for thesauri and other taxonomies it's directed for those needing a classification hierarchy with simple constrains.
- OWL DL: includes all OWL language constructs with certain restrictions, for example, a class may be a subclass of many classes, but it cannot be an instance of another class. It is directed for those who want maximum expressiveness while retaining computational completeness.

⁴ <http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.2>

1.2. Semantic Web, Ontology & Linked Data

- OWL Full: it is complete kit, for those who want maximum expressiveness and syntactic freedom and don't need computational guarantees.

Each of these sub languages is an extension of its simple predecessor.

When talking about Semantic Web or Web 3.0, the words "Linked Data" appear very often. What is Linked Data after all?

Linked data is about using the web in the creation of typed links between data from different sources. To do so Berners-Lee [5] proposed a set of principals or best practices for publishing data on the web, to ensure that all data becomes part of the same data space. These principals are:

- Use URIs as names for things;
- Use HTTP URIs so that people can look up those names;
- When someone looks up a URI, use the standards (RDF, SPARQL) to provide useful information;
- Include links to other URIs, so they can discover more things.

On the base of Linked Data there are three main technologies[9], URIs (Uniform Resource Identifiers), HTTP (HyperText Transfer Protocol) and HTML (HyperText Markup Language). Entities are identified by URIs that use the `http://` scheme and can be looked up by dereferencing the URI over the HTTP protocol and represented in the widely used content format of HTML. These technologies are supplemented by RDF that provides a graph based data model to represent the world.

According to W3C recomendations [8], data encoded in the RDF model is the form of *subject*, *predicate* and *object* triples. Both the subject and the object elements are URIs that identify a resource and the predicate represents the relation between them, also in the form of a URI.

Together, RDF and OWL provide solid basis for creating vocabularies that can be used to describe real world entities and how they are related.

The most visible example of application is the Linking Open Data project⁵. Founded on 2007, the aim of the project was to bootstrap the Web of Data by identifying available existing data sets, convert them according to the Linked Data principles and publish them on the web. From small research labs to large organizations the project has grown considerably and the scale of the Web of Data can be depicted as shown in Figure 3.

⁵ <http://linkeddata.org/>

1.2. Semantic Web, Ontology & Linked Data

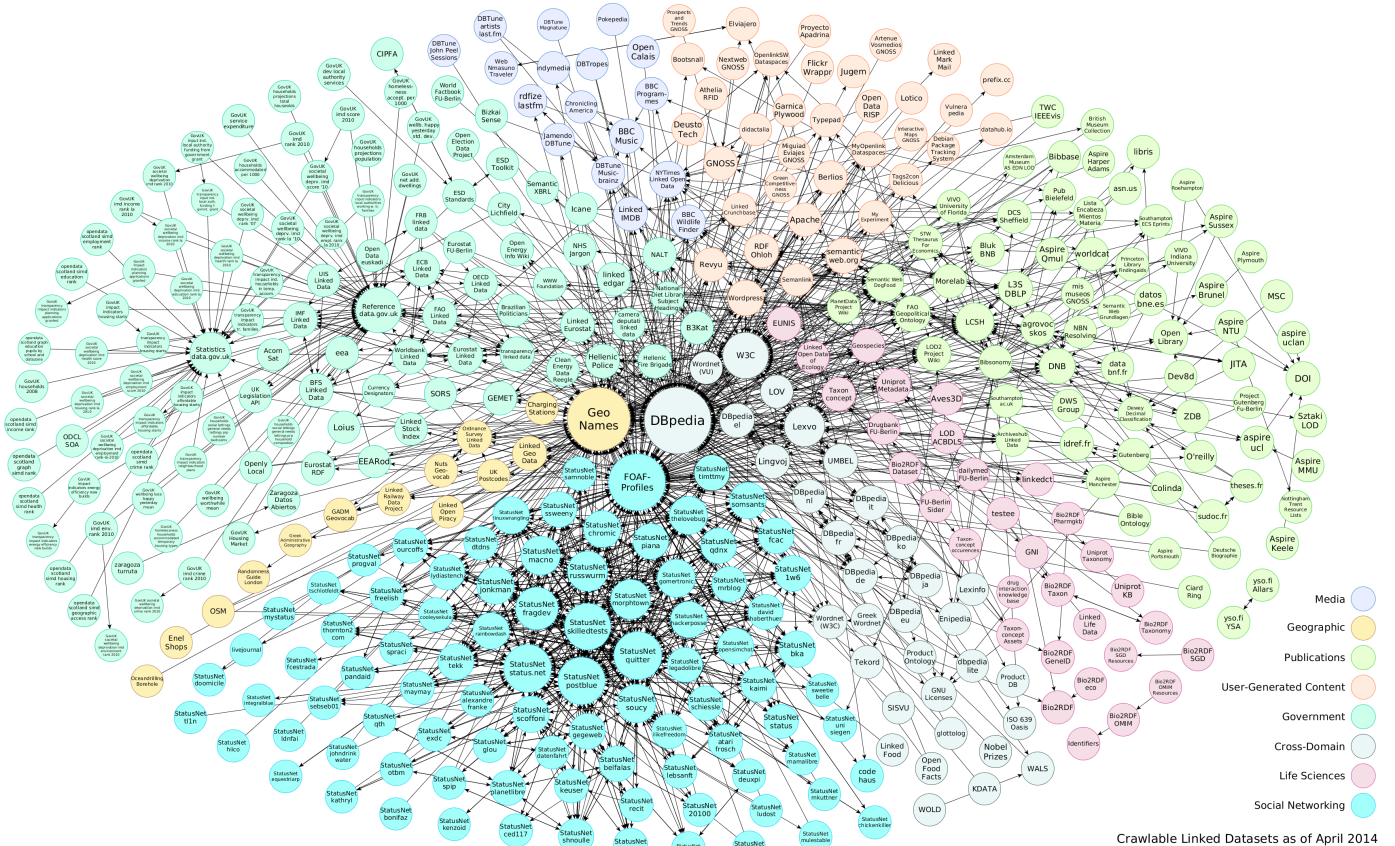


Figure 3.: Linking Open Data cloud - April 2014

Each node in the graph represents a distinct data set and each arc represents a link between items from the two connected data sets. Heavier arcs correspond to a greater number of links and bidirectional arcs indicate the outward links to the other exist in the data set.

1.3. Data Analytics & Visualization

1.3 DATA ANALYTICS & VISUALIZATION

On the past years, Business Intelligence has become an important area of study for companies. Leaders and CEOs started to wonder how to get the maximum value from the information and knowledge that they already have inside their archives and databases, in order to gain advantage on the ever more global and competitive market. This led to the development of techniques and methods to analyse and comprehend information inside companies in order to gain perspective about the company performance and predict the future of business. A well selected set of metrics about statistical data, and the use of those metrics allows companies to make informed data-driven decisions. MIT Sloan reports that companies with top performance use until five times more analytics than lower performance ones [11]. The field of web analytics has emerged and became a major field of research within business intelligence[6]. Based on data mining and statistical analysis, web analytics offers many analytical challenges and opportunities. The web based on HTTP/HTML hyper-links associated with web search engines and directory systems helped to develop technologies for web crawling, web site ranking or search log analysis. This naturally led to research about consumer behaviour, and search trends. With the growing of semantic web and the appearance of knowledge databases, it is expectable that the web analysis become even more important, as companies will be able to identify more specific needs of their clients and answer in conformity.

As important as a good business analytics process is the presentation of the conclusions of that analysis. Direct reports with charts or other visual representation of the data is half way to ensure the comprehension of the message. This leads to the visualization field.

Visualisation is the method of communicating a message using any kind of visual representation like images, charts, diagrams. This is not a new field, visualisation techniques have been used for over a thousand years, for instance in ancient cartography, however, the invention of computer graphics was a massive development on this subject.

Some areas of application:

- Data analytics: using charts or maps to create reports about sets of data.
- Scientific visualisations: representing data about experiments
- Product development: using software to create 3d models and design prototypes of products.
- Education visualisations: using simulations to aid the understanding of specific topics or subjects.

Visualization, combined with data analytics, specially in the area of multimedia, is, according to IBM's 2013 Global Technologies Outlook [10] one of the main technology trends that

1.4. Motivation

will lead to industry changing products over the next years. Due to the size and complexity of the data sets available there is a growing need of companies and particular users for visual analytics tools.

1.4 MOTIVATION

For Maisis, the main motivation behind this master work is to add value to the Oobian platform. In the economical context of today, with a very competitive market, companies have to innovate and answer with quality products to the client's needs. There is a growing need on the field of business analytics for tools that provide features to create reports about big collections of data. The Oobian platform is a horizontal knowledge management platform that provides better search and navigation through knowledge. Oobian extracts and indexes information from non-structured and structured data scattered inside and outside the organisation, transforming it into real business knowledge. One of the gaps that were identified is the lack of business analytics features in the platform. This feature could potentially improve the business intelligence processes capabilities.

For the student, the motivation is to successfully complete his master course, working and researching on an interesting field, with the added benefit of working in a professional environment.

1.5 GOALS

This section presents a general description of the main goals this master work intended to accomplish. Once this master work was developed as an internship in *Maisis*, the main goal was to develop a work that could answer to the company's problem described previously.

The work involved in this master thesis was divided in two main phases, a theoretical one, with the study of the problem and the state of the art, and a practical one with the specification and development of a software solution.

The goals for this master work were the following:

- to make a theoretical study and to understand some key concepts as:
 - Oobian platform;
 - Ontologies, Semantic Web and Linked Data;
 - Data visualization tools and techniques;
- to formalize querying mechanisms to obtain data from the Oobian core engine (Knowledge Server - Kserver).
- to specify a format for the resultant data set.

1.6. Contribution

- to propose mechanisms and techniques to analyse the data about the ontologies present on the Oobian knowledge management platform;
- to create a visual user interface to present the generated reports;
- to create a mechanism to embed those visualizations into external websites.

All the software modules should be integrated on the existent Oobian infrastructure.

The main goal of this work is to give an extra dimension of analysis of the data present in the platform, adding value to the global user experience.

1.6 CONTRIBUTION

The contributions of this master work fall within two main plans. At one side, in a theoretical plan, is a theoretical study, with reading, synthesis and analysis of bibliography related with the subjects in study. At the other side, in a practical plan, is the development of software modules to achieve the specified goals. Being so, the contributions made with this work are:

- Theoretical study on the state of the art about data analysis and visualization tools, as well as core concepts as ontologies, semantic web or linked data.
- Specification of a query mechanism to retrieve data from the Oobian knowledge server using an json table accepted by Google Visualization API as output format.
- Development and integration with Oobian HTML client, of a component to create visual representations of data present on the knowledge base.
- Specification of an HTML component structure in order to allow visualizations to be embedded into web pages.
- Development of a JavaScript library to support the embedding of visualizations created on the Oobian platform with real time data.

1.7 METHODOLOGY

Aiming to satisfy all proposed goals the adopted methodology was "Getting Things Done" (GTD)[2], to control the development of activities and documentation. This methodology consists in the idea of breaking projects into small tasks and work in one task at the time. This bottom-up way of time management leads to a stress free productivity.

This master work is composed of the following steps:

- Bibliographic search;

1.8. Work Plan

- Reading and synthesis of the selected bibliography;
- Analysis of requirements and specification of the solution;
- Development of software prototypes;
- Evaluation of the prototypes and discussion about the achieved results.

Once the work was developed in a professional context, supervision and support from the company was constant. This support was a major contribution on the process of accomplishing the proposed goals.

The supervision process translated into short meetings where the work both in development of the solution and in documentation was presented to and evaluated by Miguel Grade.

In addition, when necessary, there were meetings with Prof. Pedro Henriques, in order to update the status of the development process and to discuss the writing of the documentation.

1.7.1 *Development Methodology*

The selected methodology during the development process consisted in grouping functional requirements into small core groups. For each one of those groups, several steps were executed: implement, test, present to the supervisor, receive feedback, make necessary alterations and corrections and test again. This process translated into an agile process. Figure 4 depicts the process for each group of requirements.

After the requirement specification process, requirements were grouped according to the level of depth of the analysis, class, instance or cluster level. For each one of these groups figure 4 helps to understand the development process. As each requirement was implemented, tests were made to check if the obtained results were correct.

1.8 WORK PLAN

The duration of this master thesis was estimated in eight months. Although all main goals were well defined, initially wasn't possible to know how many time was needed to accomplish each one. Despite that, the eight months were divided in five distinct phases. The thesis report was written in parallel with all the phases described below:

1st phase - THEORETICAL STUDY: This first phase was used to study the state of the art and to learn important concepts to the development and implementation of the system.

2nd phase - SPECIFICATION AND REQUIREMENT ANALYSIS: In the second phase the main task was to specify the solution and the system requirements, which was validated by the development team and by the stakeholders.

1.8. Work Plan

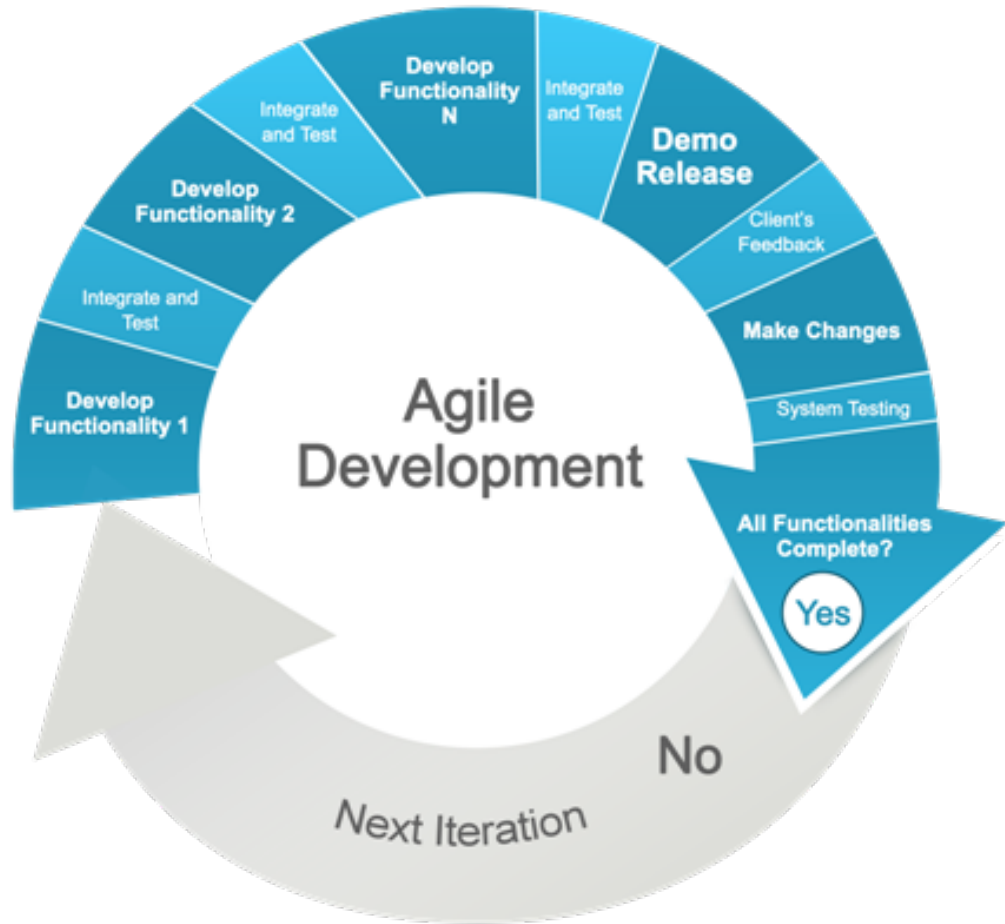


Figure 4.: Agile development process

3rd phase - DEVELOPMENT: On this phase started the development of the software prototype. This was to the longest phase.

4th phase - TESTING/BUGFIXING: After the prototype was developed the main task was to test and validate it by emulation of real world scenarios. The goal was to identify and correct possible errors or bugs to maximize the quality of the final software.

5th phase - EVALUATION AND CONCLUSION OF THE REPORT: The last phase was used to evaluate the prototype and to draw conclusions relating the results of this evaluation and all the theoretical studies done during the master work.

Once this evaluation was made, the rest of the time was used to finish this document.

1.9. Document Structure

1.9 DOCUMENT STRUCTURE

The present thesis report is organized and structured so that the reader can understand the concepts in study, the existent problem and the achieved results that aim to answer and solve that problem. Being so, this thesis was divided into several chapters that describe theoretical concepts as well as practical results of the developed components.

- **Chapter 1: *Introduction*** - In the first chapter of the document it is presented to the user the contextualization of the problem, as well as the goals intended to be accomplished and the motivation behind the project. The working methodology followed and the work plan were also introduced.
- **Chapter 2: *Knowledge Management and Visualization*** - In the second chapter it is discussed the state of the Oobian platform, in particular, its HTML client. A study about some existent data analysis and visualization tools is also presented.
- **Chapter 3: *Proposed Solution*** - The third chapter presents to the user the proposed solution to the initial problem. First it is described the requirement specification process. Then it is explained the specification of the solution itself.
- **Chapter 4: *Development*** - Chapter four describes the development stage of the master work. Some of the used technologies are presented and the main methods and choices made during the development of the different components are explained.
- **Chapter 5: *Achieved Results*** - In chapter five, readers can see the achieved results. This chapter consists of two main usage scenarios and there are several figures depicting features and behaviours of the client and the developed components.
- **Chapter 6: *Conclusion*** - Chapter six, the final chapter, presents to the reader the main conclusions drawn from this master work, as well as possible evolutions and features that can be added in the future.

Chapter 1 was an introduction to the thesis report. In this chapter the reader found an initial overview of the context of the work and the main problem it proposes to solve.

It were exposed some vital concepts to aid the understanding of the thesis, the motivation behind it and the goals Maisis intended to be accomplished.

After that it the main contributions of this work were briefly described and the work methodology was presented.

KNOWLEDGE MANAGEMENT AND VISUALIZATION

Studying the state of the art is the starting phase of a good master work. This study is important as it allows to gain a perspective of what are the works on the field and what is the state of available tools. As this master work is being developed in a specific context, part of the study of the state of the art was focused on the *Oobian* platform. In this chapter it is presented a study about Oobian, existing data analysis tools, and visualization tools or libraries.

2.1 OOBIAN PLATFORM

Companies, government agencies and other organizations maintain huge amounts of information in electronic format, including spread sheets, policy manuals, web pages, to mention just a few. Contemporary private data sets can now exceed the size of the entire Internet in the 1990s, although some organizations do not publicize their stores. The content may be stored in file shares, websites, content management systems (CMSs) or databases, but without the ability to find this corporate knowledge, managing even a small company would be hard.

*Oobian*¹ is an horizontal knowledge management platform that provides better search and navigation through knowledge. It extracts and indexes information from structured and non-structured data scattered in organizations and transforms it in organized business knowledge. *Oobian* provides a simple way of searching through all the knowledge fragments scattered all around the enterprise, offering many advantages to the common user that is trying to get his hands into specified content. *Oobian* connects people to the information they need to get their jobs done. General productivity suites, for example intranet search solutions, increase employee efficiency by connecting a broad set of people to a broad set of information. In comparison, search-driven applications drive measurable return on investment by helping a well-defined set of people accomplish specific business tasks more efficiently. Search-driven applications, such as research portals and 360° customer insight solutions aggregate information from a defined set of content repositories, add structure to unstructured information and provide a contextual, interactive and actionable experience. Booth companies and clients ben-

¹ <http://www.oobian.com/>

2.1. Oobian platform

enefit of an increase of efficiency by a better management of the information inside a company and *Oobian* provides this management.

2.1.1 *Oobian architecture*

Oobian simply putting is a platform that manages to index scattered data along companies and provide it to the users. To accomplish this, it relies on a *Client-Server* architecture as depicted in figure 5.

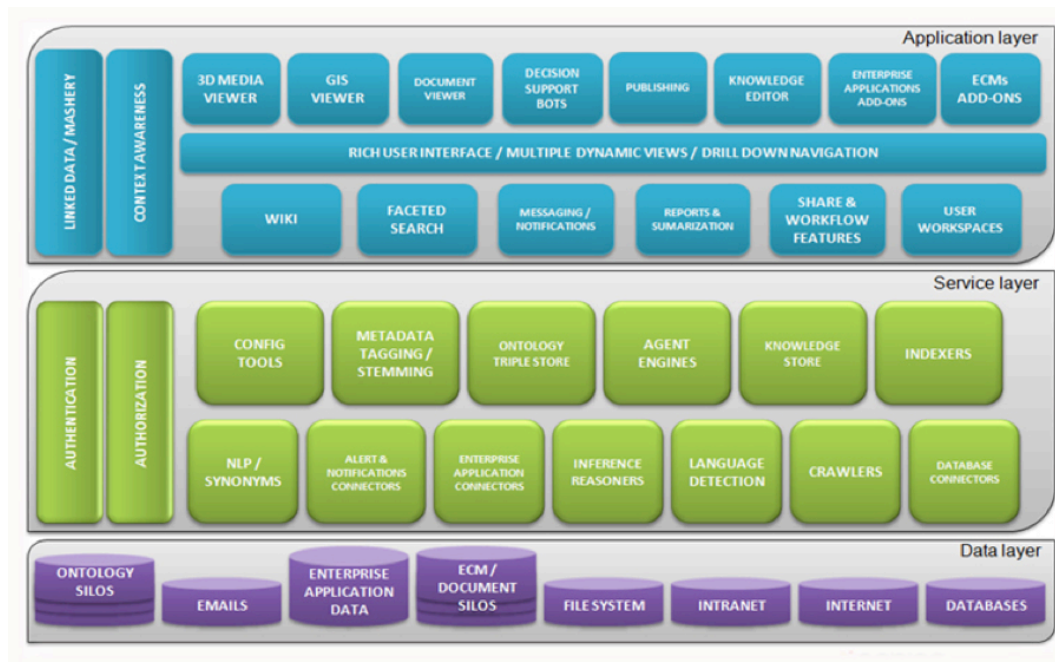


Figure 5.: Oobian architecture

The *Oobian Server* is the heart of the platform. It is responsible for all the crawling, content pipeline processing, indexing, searching and intelligent semantics. On the bottom layer of *Oobian* server there are several data connectors to feed content to the index. The top layer provides services that are consumed by the *Oobian* client. In figure 6 are listed some of the supported content sources.

Once the content is picked up from its sources, it is feed to the content processing pipeline. Figure 7 shows the main stages of that process.

2.1.2 *Oobian features*

Here are some of the *Oobian* platform key features:

- Web user interface with drill-up and drill-down features;

2.1. Oobian platform

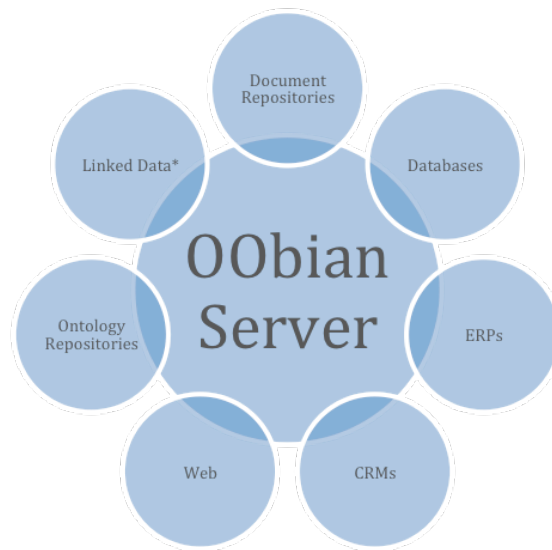


Figure 6.: Oobian Content Sources

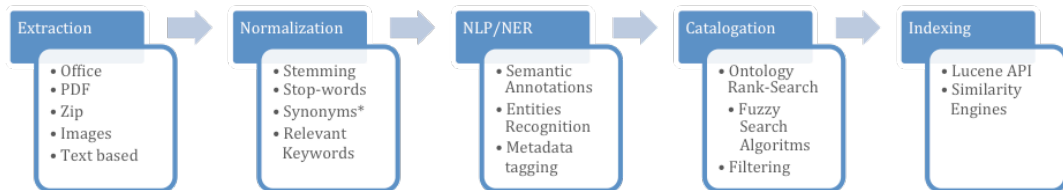


Figure 7.: Content Processing Pipeline

- Support for W3C Semantic Web recommendations like OWL or RDFS;
- Ontology based;
- Geo-referencing support;
- Highly scalable;
- SOAP and REST web service interfaces;
- Documents, Knowledge and Maps integration;

Oobian Insight is the top layer of the platform. It has two modes of presentation, two clients. One in *Silverlight* and one in *HTML5*. On this master work the focus will be on the HTML client.

2.1. Oobian platform

Oobian HTML client has two main modes:

- Navigation;
- Maps;

Navigation

Navigation mode allows users to navigate in the knowledge base, drilling up and down classes, in a rich user interface. Figure 8 depicts a navigation scenario where an user is exploring instances of a class.

Each instance is represented as a "box" with some details about it displayed below the instance icons. Users can navigate through pages of instances.

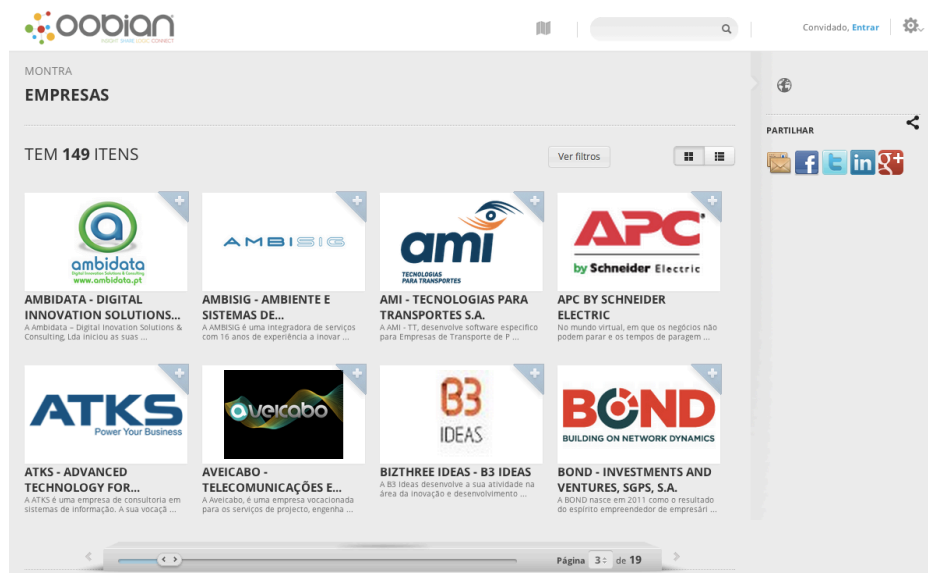


Figure 8.: Oobian HTML client - Instances of a Class

2.1. Oobian platform

Figure 9 shows some details of an instance and figure 10 depicts clusters of instances related to the current instance.

The screenshot displays the Oobian HTML client interface. At the top left is the Oobian logo. The main content area shows the details for the instance 'MAISIS - INFORMATION SYSTEMS LDA'. The text describes the company's history, its focus on knowledge management, and its services. On the right side, there is a sidebar with '41 RELACIONAMENTOS' (41 relationships), 'SUGESTÕES' (Suggestions) for related instances like ATKS and Novabase, 'PALAVRAS-CHAVE' (Keywords) such as 'Gestão Documental' and 'ECM', and 'PARTILHAR' (Share) options for social media.

PROPRIEDADES	Value
Ano de criação	1994
CAE	62010 - Actividades de programação informática
E-mail	maisis@maisis.pt
Empresa atua em (Mercado)	Portugal
Empresa atua em (Setor)	Empresas e organizações em geral Indústrias Transformadoras Transportes e armazenagem

Figure 9.: Oobian HTML client - Instance details

The screenshot shows the 'ESTÁ RELACIONADO COM' (Related to) section of the Oobian HTML client. It features a search bar and filter buttons. Below, there are eight clusters of related instances, each represented by a box with a count and a category name. The clusters are: 3 CLIENTES, 6 PRODUTOS, 3 SERVIÇOS, 4 SETORES, 3 SOLUÇÕES, and 12 ÁREAS DE ATIVIDADE. Two clusters for 'PORTUGAL' are also shown, each with a flag icon and a brief description.

Category	Count
CLIENTES	3
PRODUTOS	6
SERVIÇOS	3
SETORES	4
SOLUÇÕES	3
ÁREAS DE ATIVIDADE	12

Figure 10.: Oobian HTML client - Relation details of an instance

2.1. Oobian platform

When exploring classes or clusters of instances, it is possible to filter the instances using a mechanism of faceted search.

Faceted search, or *Faceted navigation* is a technique for accessing information organized according to a *faceted classification*. Faceted classification is a form of organizing content, by assigning multiple attributes to an object, instead of a predetermined taxonomic order. These attributes, or facets are "clearly defined, mutually exclusive, and collectively exhaustive aspects, properties or characteristics of a class or specific subject" [3]. Lets consider an example, **cars**. A **car** has a certain *brand*, it is a certain *colour*, it is listed with a certain *price* and belongs to a certain vehicle *class*. Using facets it is possible to set up a handful of categories that will combine to fully describe a car: *brand*, *colour*, *price*, *class*. Each category is populated with the right terms and organized. Then each car is classified by choosing the right terms for each category. This is a faceted classification, a set of mutually exclusive and jointly exhaustive categories, each made by isolating one perspective on the items that combine to describe all the objects in question.

Figure 11 illustrates the mechanism of faceted filtering present in *Oobian* client.

The screenshot shows the Oobian HTML client interface for searching companies. The main heading is 'MONTRA EMPRESAS'. Below it, there's a filter for 'Analytics' and a count 'TEM 130 ITENS'. The main content area displays a grid of six company cards, each with a logo, name, and a brief description. The cards are: Alcatel-Lucent Portugal S.A., Ambidata - Digital Innovation Solutions..., Ambisig - Ambiente e Sistemas de..., Atks - Advanced Technology for..., Aveicabo - Telecomunicações e..., and Bond - Investments and Ventures, SGPS, S.A. On the right side, a 'Ver filtros' (View filters) panel is open, showing a date range filter for 'Ano de criação' (Creation year) from 1974 to 2012, and a list of other facets like CAE, E-mail, Fax, Morada, NIF, Número de colaboradores, Rácio (VAB/Volume de negóc...), Telefone, Valor acrescentado bruto (€), and Volume de negócios (€). The interface also shows a page number 'Página 1 de 17' at the bottom right.

Figure 11.: Oobian HTML client - Faceted filtering

2.1. Oobian platform

Maps

Maps mode allows users to explore geotagged instances in an interactive map, as shown in Figure 12.

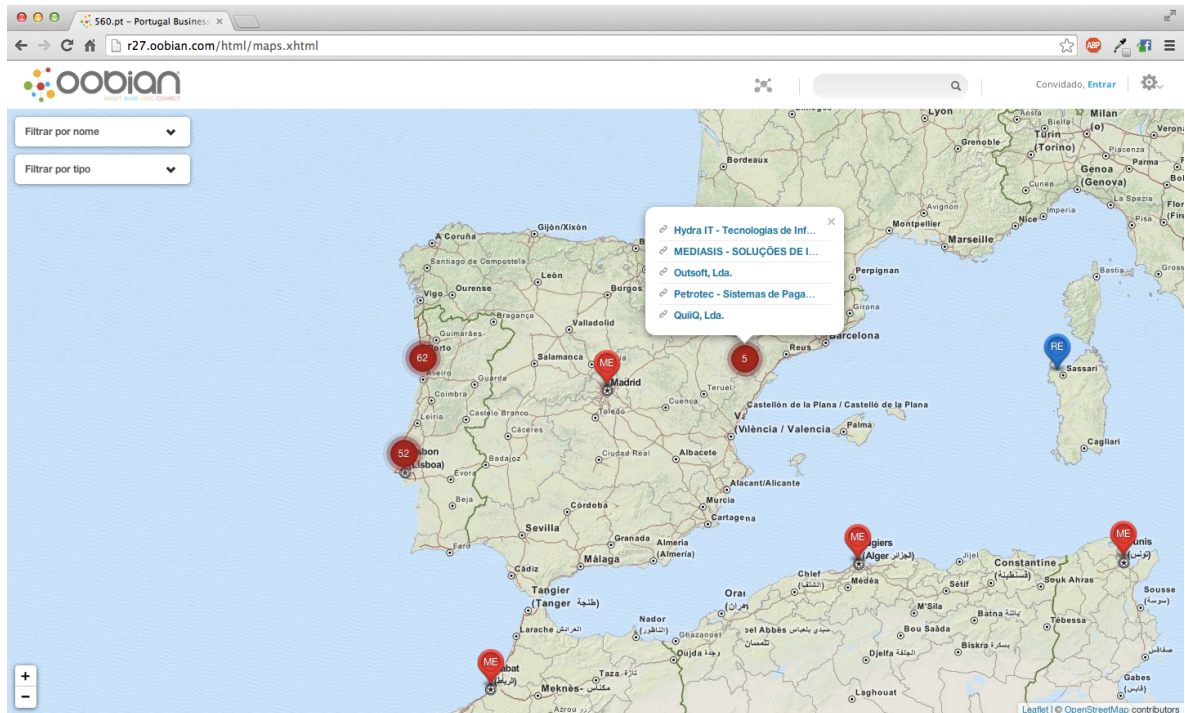


Figure 12.: Oobian HTML client - Maps mode

2.2. Data Analysis and Visualization tools

2.2 DATA ANALYSIS AND VISUALIZATION TOOLS

One of the main goals of this master work is to develop a component to analyse information with a visual interface. So, it is important to do a research about existing tools for that purpose. In this section are investigated and presented some of those tools.

2.2.1 *Sgvizler*

*Sgvizler*² is a project for a "small JavaScript wrapper" [14] which aims to create visualisations of data by rendering the result set of SPARQL queries into charts or HTML elements. This project supports a large number of visualization types. It works receiving a SPARQL query and a SPARQL endpoint from the user and translating the result set into a specific Json format. This Json is accepted by Google Charts, and the visualizations are generated using the Google Charts API. The generated visualizations can be integrated in *HTML* web pages by including the generated *HTML* elements. Figure 13 is an example of a generated *HTML* element.

```
<div id="example"
  data-sgvizler-endpoint="http://sws.ifi.uio.no/sparql/npd"
  data-sgvizler-query="SELECT ?class (count(?instance) AS ?noOfInstances)
    WHERE{ ?instance a ?class }
    GROUP BY ?class
    ORDER BY ?class"
  data-sgvizler-chart="google.visualization.PieChart"
  style="width:800px; height:400px;"></div>
```

Figure 13.: Sgvizler html element example

² <http://dev.data2000.no/sgvizler/>

2.2. Data Analysis and Visualization tools

2.2.2 Google Charts

*Google Charts*³ is a visualization API from *Google* that allows users to visualize and present many different types of information by creating interactive charts. All chart types are populated with data from a specific data table with a default format, making it easy to create different charts and visualizations from the same data source. That data table can be populated directly from the web site, from a database or a Json file in a specific format. The charts are JavaScript classes with customization properties and are rendered using *HTML5/SVG* technologies providing cross-browser compatibility. Figure 14 is an example of a chart created by this API.

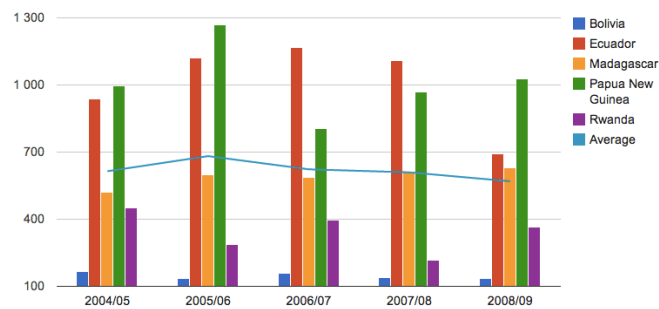


Figure 14.: Google Charts example

³ <https://developers.google.com/chart/?hl=pt-PT>

2.2. Data Analysis and Visualization tools

2.2.3 IBM Many Eyes

Many Eyes⁴ is a web based project from *IBM* that provides visualizations for data upload by the users [16]. It is developed by an open web community that includes visualization experts, practitioners, academics and enthusiasts offering their expertise, aiming to provide the best format to present the data from the users. Users upload their public data, contained in spreadsheets or text files, and are presented with a variety of visualizations recommended by *Many Eyes*. Once the visualization is selected the users can share that visualization over the web. Figure 15 shows an example of a visualization created with *Many Eyes*.

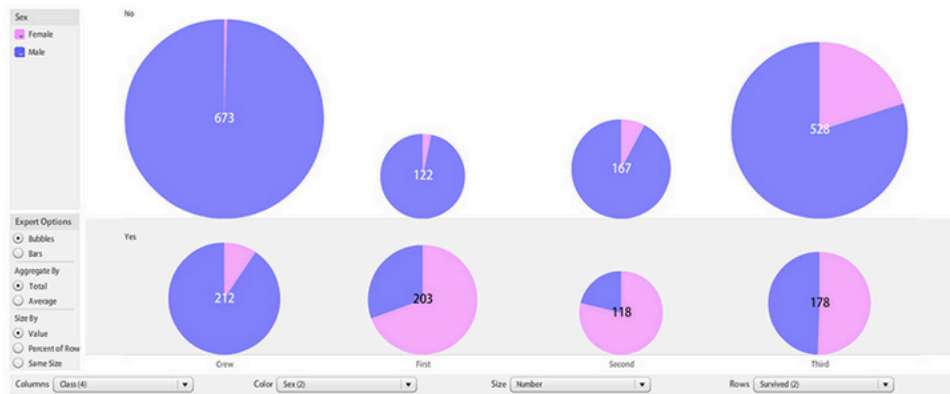


Figure 15.: IBM Many Eyes visualization example

This is an interesting tool, but it is not fitted to be used by enterprises or companies, mainly because it requires for the uploaded data to be public.

⁴ <http://www-958.ibm.com/software/analytics/manyeyes/>

2.2. Data Analysis and Visualization tools

2.2.4 Microsoft Silverlight PivotViewer

PivotViewer⁵ is a *Microsoft Silverlight* control to visualize large collections of objects at once. It has a dynamic interface with several data filters, sorting and browsing features to help users to quickly find what they are searching. Figure 16 shows an example of *PivotViewer*.

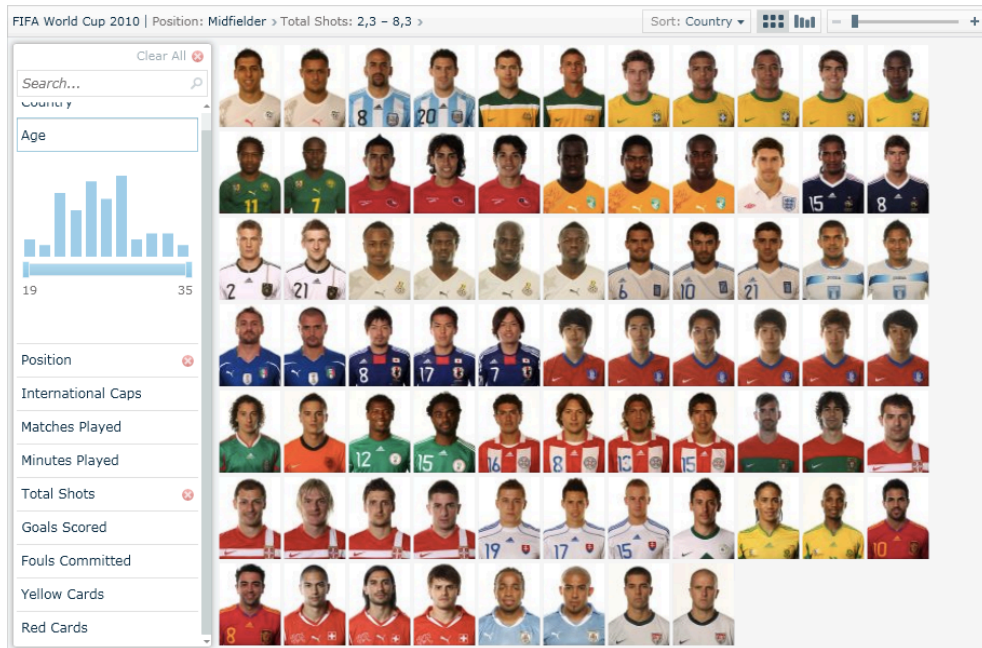


Figure 16.: Silverlight PivotViewer example

Chapter 2 presented to the reader a study on the state of the art. The main subject of study was the Oobian ecosystem, in particular, its HTML client. After that, the study focused on data analysis and visualization tools and libraries.

⁵ <http://www.microsoft.com/silverlight/pivotviewer/>

PROPOSED SOLUTION

In this chapter is explained the proposed solution. In first place, it is presented the requirement specification process as well as a general requirements list. Then the solution itself is described, starting with its general architecture followed by more detailed explanations about the several components of the project.

3.1 REQUIREMENT SPECIFICATION

A major part in the process of development of a software solution is the requirements specification. In this section it is presented that specification, the methodology followed, the stakeholders of this work and actors that will interact with the system. Two lists with the requirements are also presented. The full requirement specification is presented in appendix [A](#).

3.1.1 *Analysis Methodology*

As said before, one of the most important phases of the software development process is the process of requirement analysis. In order to maximize the quality of the analysis it is recommended to use a well defined methodology.

The selected methodology for the development of this work was FURPS/FURPS+ [1].

FURPS is an acronym for:

- **Functionality:** functional requirements describing features or capabilities that the client expects the software to have;
- **Usability:** describe the system from the user perspective and take into account factors like the user interface, the quality of documentation, the efficiency and the general quality of the system;
- **Reliability:** describe the system predictability, precision or failure recovery;

3.1. Requirement Specification

- **Performance:** describe the system's performance, like response time, throughput or resource usage;
- **Supportability:** describe factors like the compatibility of the system, maintainability or configurability.

FURPS+ adds support for some possible needs:

- **Design:** restrictions that influence the design of the solution, as the use of relational databases;
- **Implementation:** use of specific tools or standards of development;
- **Interface:** factors about how the solution interacts with other systems, for instance format restrains;
- **Physical:** hardware needed to support the final solution.

3.1.2 Stakeholders

Once the analysis methodology is defined, it is important to identify the stakeholders of the project. A stakeholder is any person or organization that have interest in the development of the system.

There are three main stakeholders related with this master work. The first one is *Maisis Information Systems*, the company that provided the subject and the possibility of an internship, represented by Eng. Miguel Grade. Other of the stakeholders is the trainee Pedro Moreira, that is developing this master work with the intent of finishing his masters degree. The final stakeholder is professor Pedro Rangel Henriques from University of Minho, who very kindly agreed to supervise and support this master work.

3.1.3 Actors

In this section are presented the actors that will interact with the platform. Identifying the actors is an important step on the requirement specification process, as it helps to identify particular needs of users.

There are two main actors interacting with the system: the common user of the platform and the external embedded component.

The common user while navigating in the interface, consults and filters information and has the ability to create visualizations, get the embed code of the visualization and export it as

3.1. Requirement Specification

a CSV table.

The external embedded component does requests directly to the server. In first place it makes a request to obtain a valid authentication token. After that it makes a request with the query that returns the data table with the results. Figure 17 depicts a simple use case diagram of the system.

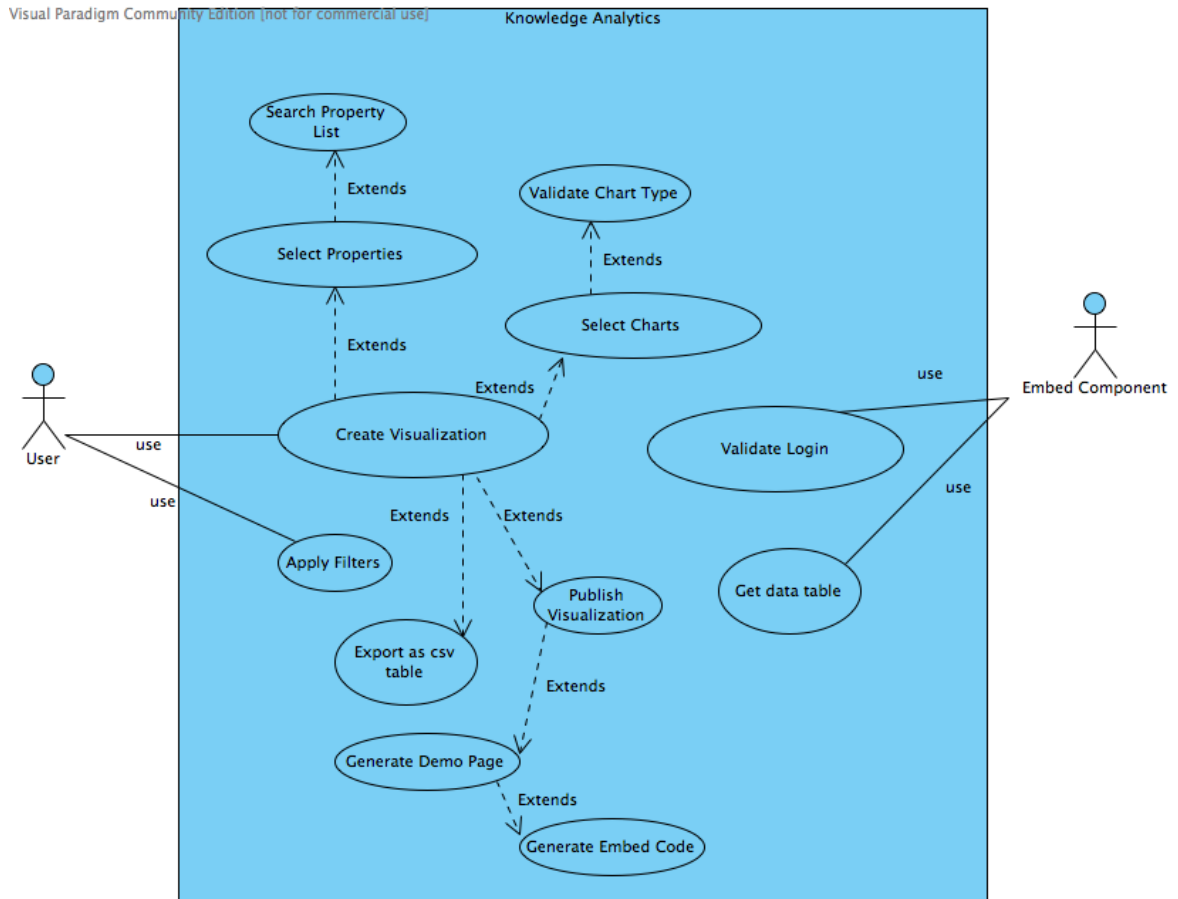


Figure 17.: Use case diagram

3.1. Requirement Specification

3.1.4 Requirements

On the process of software development time and efficiency are key factors. So, it is a good practice to prioritise the different requirements, in order to help organise the development process and obtain a final product that is as close as possible to the wanted result.

Table 1 presents the type of priority by which requirements will be listed.

Priority	Description
Must	Requirements that must be implemented in order to consider the final product a success.
Should	Requirements that should, if possible, be part of the final product.
Nice	Desirable requirements, implemented only if there is time to do it, without compromise the implementation of higher priority requirements.

Table 1.: Requirement Priority

3.1. Requirement Specification

In order to present to the reader a general view of the requirements, on this subsection are presented two lists of requirements and their priorities.

Table 2 shows the first list of requirements. Functional requirements describing features and behaviours that the final solution must have.

Requirement Id	Name	Priority
1	HTML client integrated component	Must
2	Class level analysis	Must
3	Instance level analysis	Must
4	Cluster level analysis	Must
5	Data properties picker	Must
6	Multiple visualization types support	Must
7	Visualization constrains	Must
8	Native query support	Must
9	<i>Google Visualization</i> Json table output	Must
10	Embed HTML component	Must
11	Self updating HTML component	Must
12	Interval result segmentation	Must
13	Query filter support	Must
14	Query limits support	Should
15	Query offset support	Should
16	Query operations support	Nice
17	Data table export as CSV	Nice

Table 2.: Functional requirements

Table 3 shows the second list of requirements. Non-Functional requirements considered as restrictions both to the system and the development.

Requirement type	Requirement Id	Name	Priority
Usability	18	Effectiveness	Must
Usability	19	Efficiency	Must
Design	20	Oobian platform	Must
Interface	21	Oobian client	Must
Physical	22	Hardware requirements	Must

Table 3.: Non-Functional requirements

3.2. Proposed Architecture

3.2 PROPOSED ARCHITECTURE

As said previously, the main goals of this project were to formalize a query structure, to propose an output format for the resultant data set and to represent that data set in a visual form.

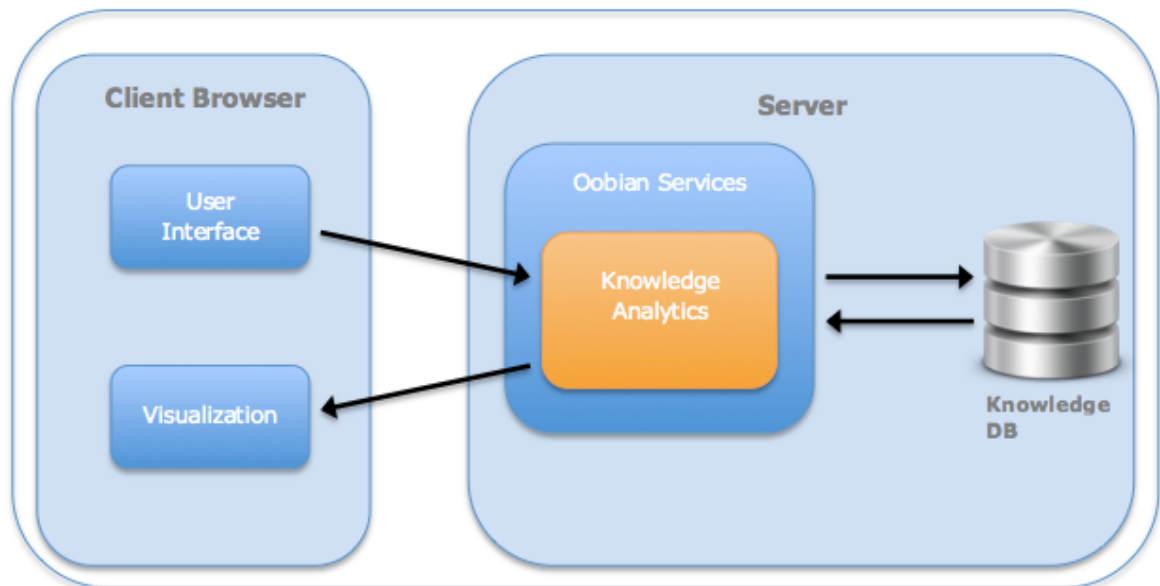


Figure 18.: Architecture Block Diagram

So, the proposed solution consists on a component, to be available in the Oobian client user interface, where the user selects data properties and chart types he wants to analyse. Once the data properties are selected, a query in the specified Json format is injected into a request that is sent to a web service present in the Kserver. That request with the Json query is processed and the resultant data set is mapped into the specified Json data table format. The web service returns that data table which is then processed by the visualisation API and the charts are presented to the user. Figure 18 represents a general view of the solution.

3.3. Query Format

3.3 QUERY FORMAT

As said before, one of the main goals of this master work was to define a query format to obtain data from the server.

The first step to accomplish this goal was to formally define a context free grammar to describe the query language.

```
query:
  '{select:'select',from:'from', where:'where', limit:INTEGER', offset:INTEGER
  ',' locale:'STRING}'
;

select:
  '{operator:'STRING',properties:['properties']}'
;

from:
  '{contextId:'STRING',('instanceId:STRING',)*', namespace:'STRING'(', objPropId
  : 'STRING')*}'
;

where:
  '{filters}'
;

properties:
  STRING(','STRING)*
;

filters:
  filter(','filter)*
;

filter:
  STRING':'['STRING(','STRING)*']'
  |STRING':'['INTEGER(','INTEGER)*']'
;
```

As shown in the listing above, the query has the following elements:

- **Select:** contains an operator(e.g., Count) and an array with a list of properties to be obtained from the server and included on the final chart. Each data property defined here corresponds to a column of the results table. The order of the data properties

3.4. Output Format

is important as it defines the order of the data columns of the result data set and ultimately the visual representation.

- **From:** identifies the location of the data, with the **contextId** representing the *id* of the context (class or instance id) of the ontology, the **instanceId** representing the *id* of the instance and the namespace where those ids are located. **InstanceId** is an optional field, as it is only needed when performing analysis of clusters related to one instance in particular **objPropId** represents the id of a relation, it is also optional as it is only needed when performing analysis of a particular relation of an instance;
- **Where:** contains a set of properties defining the criteria to filter the results of the query;
- **Limit:** an integer value to limit the number a results;
- **Offset:** an integer value to define the number of items to skip before returning results.
- **Locale:** contains the language of the data that should be returned.

Below is an example of a valid sentence containing a query:

```
{
  select:{operator:"null",properties:["city"]},
  from:{
    contextId:"Education",
    instanceId:"Person_W-A0CgjJyz",
    namespace:"http://www.owl-ontologies.com/CV.owl#"
  },
  where:{
    ":countryCodeSource" : [ "GB" ], ":city" : [ "London" ]
  },
  limit:1,
  offset:1,
  locale:"pt"
}
```

3.4 OUTPUT FORMAT

With this master work, Maisis intended to specify a standard output format for the result set of the query. So, to ensure that this format was reusable if needed, the chosen solution was to create a Json file. This Json follows the rules accepted by Google Charts API. The Json file has two main properties: cols and rows.

3.4. Output Format

Cols: consists in an array of objects matching the columns of a table that describes the ID and Type of each column.

- id: identifies the column;
- label (optional): string with a label to identify the column;
- type: data type of the column;

Supported data types:

- string;
- number;
- boolean;
- date;
- datetime;
- timeofday.

Rows: consists in an array of objects matching the rows of a table. Each row contains a array of cell (c) objects corresponding to the cells of a table.

Each cell has a v property that is the value of the cell. The data type of the cell has to match the data type of the correspondent column.

Example of a Json structure with 3 columns and 5 rows.

```
{
  "cols": [
    {"id": "col_1", "label": "year", "type": "string"},
    {"id": "col_2", "label": "sales", "type": "number"},
    {"id": "col_3", "label": "expenses", "type": "number"}
  ],
  "rows": [
    {"c": [{"v": "2001"}, {"v": "3"}, {"v": "5"}]},
    {"c": [{"v": "2002"}, {"v": "5"}, {"v": "10"}]},
    {"c": [{"v": "2003"}, {"v": "6"}, {"v": "4"}]},
    {"c": [{"v": "2004"}, {"v": "8"}, {"v": "32"}]},
    {"c": [{"v": "2005"}, {"v": "3"}, {"v": "56"}]}
  ]
}
```

3.5. Embed component

3.4.1 CSV Table

Having defined that the output of the service was going to be a Json string containing a table, it was only logic to choose *CSV* as the format to the export feature.

CSV (*Comma Separated Values* or *Character Separated Values*) is a simple file format widely supported by business, consumer and scientific applications, that stores tabular data in a plain-text format. This means that the file is a sequence of characters with no data that has to be interpreted. A *CSV* file can store any number of records, separated by line breaks. Each record consists of fields separated by a character, for example a comma or a semicolon, as illustrated in the following listing.

```
Number of employees;Companies
1-10;58
100-500;22
50-100;14
10-25;13
25-50;12
```

3.5 EMBED COMPONENT

One of Maisis initial goals for this work was to develop a mechanism to embed visualizations created by users, with the help of Oobian client, into their own websites.

To accomplish this goal, the proposed solution was to dynamically generate an HTML component that could easily be copied and integrated in a web page source code.

Below is depicted an example of a generated component.

```
<div id="knowledge1404490905747"
  data-knowledge-address="http://172.27.192.180:8080"
  data-knowledge-service="/Knowledge/rest/kanalytics/data"
  data-knowledge-user="jujn9P4BXX+U2MGahmo0Vg=="
  data-knowledge-pass="dZW+Md37p1sDdhNUaJcF6w=="
  data-knowledge-query="{ 'select' : { 'operator' : null, 'properties' : [ ':
    anoCriacao' ] },
                          'from' : { 'contextId' : ':Empresa', 'namespace' :
    'http://www.tice.pt/ontology/tice.owl#' },
                          'where' : { '' : [ ] },
                          'limit' : 0,
                          'offset' : 0,
                          'locale' : 'pt' }"
  data-knowledge-chart="BarChart"
  data-knowledge-title="Empresas / Ano de criação"
```

3.5. Embed component

```
style="width:600px; height:400px">
</div>
```

In *HTML 5* it is possible to store custom data attributes in HTML components. These stored *data-** attributes can later be used in the *JavaScript* pages to create richer and engaging user experience.

The generated *embed* component has the following structure:

- *id*: Id of the component. In order for this id to be unique, it is composed by the word *knowledge* followed by the timestamp of the time when it was created. This way it is possible to include several components in the same page without having duplicated ids;
- *data-knowledge-address*: Address where the *Kserver* is located;
- *data-knowledge-service*: Contains the specific web service signature that returns the data;
- *data-knowledge-user*: Contains an encrypted user name to login into the *kserver*;
- *data-knowledge-pass*: Contains the correspondent password;
- *data-knowledge-query*: Contains the specific query for obtaining the desired results.
- *data-knowledge-chart*: Visualization type;
- *data-knowledge-title*: Visualization title;
- *style*: Contains the styling of the component, namely the width and the height of the visualization.

Chapter 3 focused on the specification of the proposed solution. In this chapter, the solution itself was presented to the reader. Starting from the requirement specification process followed by the specification of the several proposed components, specific aspects of the proposed solution were described.

DEVELOPMENT

This chapter of the document aims to present the development stage of the master work. It is made a description of the main methods, choices and results obtained while developing the solution. This chapter is divided in several sections, in order to explain the several components that were developed, both client and server side. To aid the understanding of what was developed, this chapter is illustrated with several pictures and diagrams.

4.1 CLIENT SIDE

As said previously, the Oobian platform is based on a well defined *Client - Server* architecture. Being so, the developed solution is based on the same architecture that was introduced in the previous chapter and depicted in [18](#)

Oobian HTML client is based on the JavaServer Faces (JSF) framework. JSF is a "standard Java framework for building user interfaces for Web applications." It is based on a component-driven design model and uses XML files called view templates or *facelets*. The *FacesServlet* processes requests, loads the appropriate view template, builds a component tree, processes events and renders and HTML response to the client.

In JSF exists the concept of composite components, special types of templates that act as a component. Any component is essentially a piece of reusable code that behaves in a particular way. A composite component consists of a collection of markup tags and other existing components. This reusable component has a customized, defined functionality and can have validators, converters, and listeners attached to it like any other component. With *Facelets*, any *XHTML* page that contains markup tags and other components can be converted into a composite component. [Figure 19](#) depicts a simple diagram of *JSF* architecture.

In order to integrate the desired *Knowledge Analytics* module in the *Oobian client* the approach chosen was to develop a composite component that could be easily integrated in the interface, as the one listed in [figure 20](#).

This interface module should provide a slider panel with a menu to select which property to analyse, the type of visualization to create, both limit and offset inputs to specify the set of

4.1. Client side

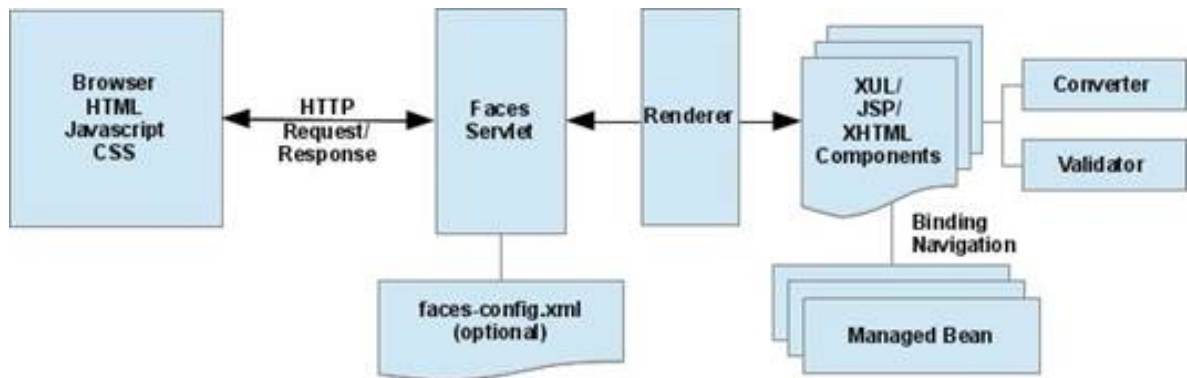


Figure 19.: JSF architecture

```

<div>
  <oob:KnowledgeFaceted rendered="#{readerBean.rootInstances!=null
    and fn:length(readerBean.rootInstances)!=0}"
    readerClass="#{readerBean.currentClass}"
    icid="#{icid}"
    objectId="#{readerBean.currentReaderClassId}"
    isInstance="false" namespaceId="#{readerBean.currentClassNamespace}"
    propertyId=""
    classId="#{readerBean.currentReaderClassId}"/>
</div>

```

Figure 20.: Knowledge Analytics composite component

results, and the number of segments. This number of segments defines the number of intervals to split the results when performing analysis of numeric or temporal properties.

Once this panel is a *JSF* composite component, the only work needed to include it in an existent page is to call it and pass context parameters.

Figure 21 shows the slider panel described above, integrated in the interface at a class level.

JSF uses *Managed Beans* to separate presentation from business logic. Program logic is contained in the bean implementation code and *JSF* simply refers to bean properties. Basically, a managed bean is a specialized Java class that synchronizes values with components, processes business logic and handles navigation between pages.

It is possible to define the scope in which beans are stored. There are the following scopes available:

- **Application** (*@ApplicationScoped*): Application scope persists across all users' interactions with a web application.
- **Session** (*@SessionScoped*): Session scope persists across multiple HTTP requests in a web application.
- **View** (*@ViewScoped*): View scope persists during a user's interaction with a single page (view) of a web application.

4.1. Client side

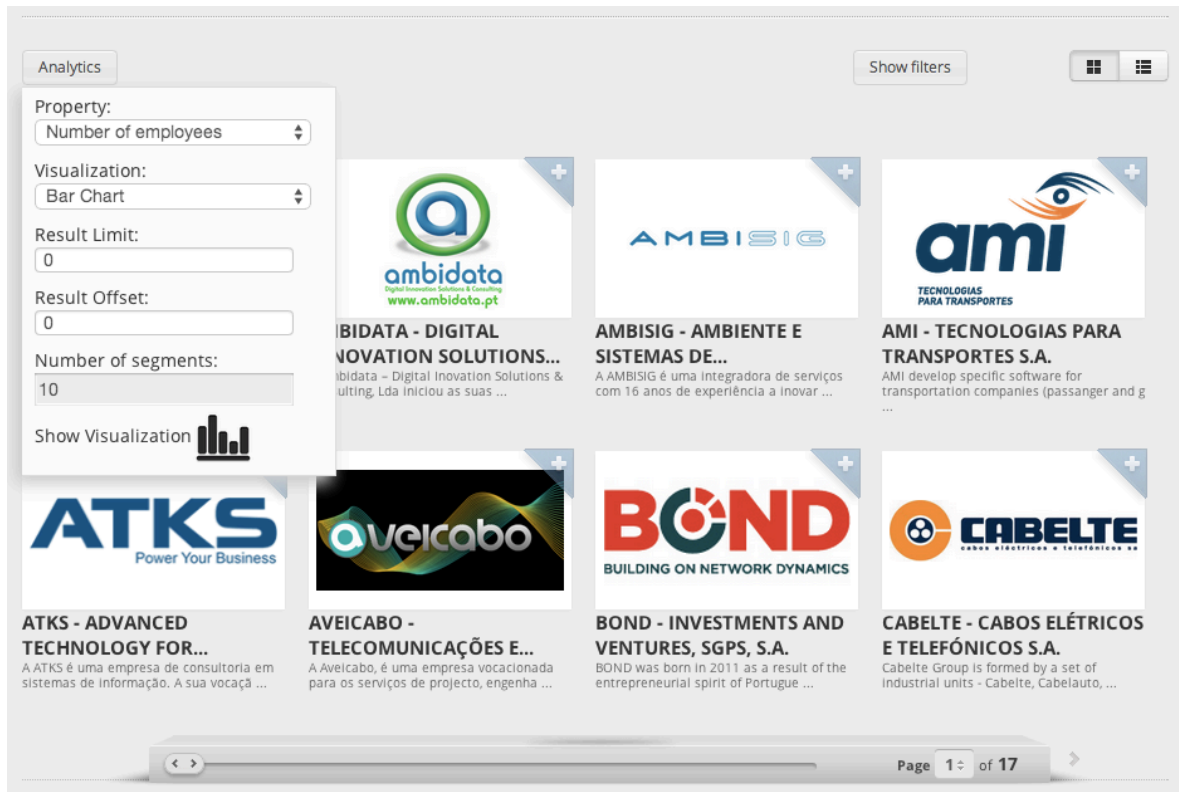


Figure 21.: Analytics panel

- **Request** (*@RequestScoped*): Request scope persists during a single HTTP request in a web application.
- **None** (*@NoneScoped*): Indicates a scope is not defined for the application.
- **Custom** (*@CustomScoped*): A user-defined, non-standard scope.

To support the composite component, it was developed a *view scoped* managed bean. This *knowledge bean* is the core of the client side as it handles all the logic of the component.

According to user's interaction with the interface, it builds a query object, and serializes it in a *Json* string. To do this serialization the choice was to use *Jackson*. *Jackson* is a suite of data processing tools for Java, among which are included libraries as the *Object Mapper* or *Object Writer* that allows to serialize Java objects to *Json* strings and deserialize *Json* strings back to Java objects.

4.1. Client side

In *JSF* it is possible to inject dependencies of managed beans into other managed beans. *Oobian HTML* client has several managed beans to handle different operations. When developing the *Knowledge Analytics* component this was taken into account and several dependencies were included in the *Knowledge* bean. Figure 22 shows an extract of the *Knowledge* bean where these dependencies are injected.

```
@ManagedBean(name = "knowledge")
@ViewScoped
public class KnowledgeBean {

    @ManagedProperty(value = "#{userSessionBean}")
    private UserSessionBean userSession;
    @ManagedProperty(value = "#{facetBean}")
    private FacetedSearch facetedSearchBean;
    @ManagedProperty(value = "#{languageBean}")
    private LanguageBean languageBean;
    @ManagedProperty(value = "#{cfg}")
    private ConfigurationBean configurationBean;
    @ManagedProperty(value = "#{readerBean}")
    private ReaderBean readerBean;
    @ManagedProperty(value = "#{navigationBean}")
    private NavigationBean navigationBean;
    @ManagedProperty(value = "#{tableBean}")
    private TableBean tableBean;
```

Figure 22.: Extract from *Knowledge* bean: managed properties

For example, *facetedSearchBean* handles the logic behind the faceted search features of the client. The best way to include filters when building a query was to use filters that users apply when navigating in the client. So when users create a visualization it is created with the filtered set of data and reflect what users define in the faceted filtering mechanism.

4.1. Client side

Once created the query object is parsed by *Jackson* into a *Json* string and posted into a web service request sent to the *Knowledge Analytics* REST web service (figure 23). As response, the web service returns a *Json* string containing a data table with the format specified in chapter 3.

```
Query query = new Query(select, from, where, limit, offset, locale);
Request request = new Request(query, visualization);
setEmbedQuery(mapper.writeValueAsString(query));
json = mapper.writeValueAsString(request);

Client client = Client.create();
String token = getUserSession().getUserToken();

String serverUrl = userSession.getServerURI() + getKnowledgeServicePath() + "query?tk=" + token;
WebResource webResource = client.resource(serverUrl);

ClientResponse response = webResource.type("application/json").post(ClientResponse.class, json);
if (response.getStatus() != 201) {
    throw new RuntimeException("Failed - Http error code: " + response.getStatus());
}
tabelaJson = response.getEntity(String.class);
```

Figure 23.: Extract from *Knowledge bean*

4.1. Client side

The data table is read by a JavaScript function (see figure 24) included in the component and the visualization is drawn in a pop-up.

```
<script type="text/javascript">
  google.load('visualization', '1', {'packages': ['corechart', 'table']});
  google.setOnLoadCallback(drawChart);
  function drawChart() {
    var data = #{tableBean.jsonTable};
    if (data !== 'empty'){
      var chartOpt = #{tableBean.chartOpt};
      var table = new google.visualization.DataTable(data);
      var chart = new google.visualization.#{tableBean.chartName}(document.getElementById('chartDiv'));
      if (chartOpt === 'donut'){
        var options = {
          title: "#{tableBean.chartTitle}",
          pieHole: 0.4,
          hAxis: {slantedText: true},
          width: #{tableBean.chartWidth},
          height: #{tableBean.chartHeight}
        };
      }
      else{
        var options = {
          title: "#{tableBean.chartTitle}",
          hAxis: {slantedText: true},
          width: #{tableBean.chartWidth},
          height: #{tableBean.chartHeight}
        };
      }
      chart.draw(table, options);
      document.getElementById('popupPanelDiv').style.display = 'block';
    }else{
      document.getElementById('popupPanelDiv').style.display = 'none';
    }
  }
</script>
```

Figure 24.: JavaScript function to draw a visualization

4.1. Client side

4.1.1 Component Integration and Configuration

As said previously, *Oobian HTML* is based on *JSF* and it is a modular platform. It can be configured to include or not to include modules in order to meet users needs. These configurations are stored on a *properties* file that is read by a configurations managed bean that defines if the modules are rendered or not in the *HTML* client.

Knowledge Analytics was developed as a composite component, intended to be a module, so its configurations are stored on that same *properties* file. Figure 25 shows an extract of that *properties* file.

```
##### CONFIGURE KNOWLEDGE ANALYTICS OPTIONS #####
knowledgeAnalytics.enabled=true
knowledgeAnalytics.exportCsvEnabled=true
knowledgeAnalytics.embedChart=enabled
knowledgeAnalytics.defaultLimit=0
knowledgeAnalytics.defaultOffset=0
knowledgeAnalytics.defaultChartWidth=900
knowledgeAnalytics.defaultChartHeight=600
knowledgeAnalytics.defaultEmbedWidth=600
knowledgeAnalytics.defaultEmbedHeight=400
```

Figure 25.: Extract from the configurations file

There are the following configurable properties:

- *enabled*: a boolean that defines if the component is enabled;
- *exportCsvEnable*: a boolean that defines if the export as *csv* feature is enabled;
- *embedChart*: a boolean that defines if the visualization embedding is enabled;
- *defaultLimit*: an integer to define the default *Limit* of the query to limit the result set;
- *defaultOffset*: an integer to define the default *Offset* the result set;
- *defaultChartWidth* & *defaultChartHeight*: integers to define the default dimensions, in pixels, of the visualization on the client;
- *defaultEmbedWidth* & *defaultEmbedHeight*: integers to define the default dimensions, in pixels, of the visualization when embedded on an external web site.

4.2. Server side

4.2 SERVER SIDE

As said previously (see Chapter 3, Figure 18), *Knowledge Analytics* component relies on a *Client-Server* architecture. In the previous section the main aspects of the development of the client side were presented to the reader; in this section are presented the main aspects of the development of the server side.

In the core of the server side are several REST web services, developed using the *Jersey* framework. *Java* defines REST support via the Java Specification Request (JSR) 311. This specification is known as *JAX-RS*. Jersey is a Java framework for developing RESTful web services, that provides support for *JAX-RS APIs* and serves as *JAX-RS* reference implementation.

JAX-RS uses annotations to define the REST behaviour of Java methods:

- *@PATH*: Path to the base URL + /created_path. The base url is defined by the application name, the servlet and the URL from the *web.xml* configuration file;
- *@POST*: Indicates that the method answer to an HTTP POST request;
- *@GET*: Indicates that the method answer to an HTTP GET request;
- *@PUT*: Indicates that the method answer to an HTTP PUT request;
- *@DELETE*: Indicates that the method answer to an HTTP DELETE request;
- *@Produces(type)*: Defines which MIME type is delivered by the method, for example "text/plain" or "application/Json";
- *@Consumes(type)*: Defines which MIME type is consumed by the method;
- *@PathParam*: Used to inject values from the URL into a method parameter.

4.2. Server side

```
public interface KAnalyticsServiceInterface {
    @GET
    @Path("getLogin")
    @Produces("application/json")
    public Response getLogin(@QueryParam("us") String user, @QueryParam("pw") String pass);

    @GET
    @Path("data")
    @Produces("application/json")
    public Response getTableForComponent (@QueryParam("tk") String token, @QueryParam("query") String query);

    @POST
    @Path("query")
    @Consumes("application/json")
    @Produces("application/json")
    public Response getTableForClient (@QueryParam("tk") String token, Request request);

    @POST
    @Path("clusterQuery")
    @Produces("application/json")
    public Response getTableForClientCluster (@QueryParam("tk") String token, RequestCluster request);

    @GET
    @Path("clusterQueryData")
    @Produces("application/json")
    public Response getTableForComponentCluster (@QueryParam("tk") String token, @QueryParam("query") String query);

    @GET
    @Path("clusterQueryObject")
    @Produces("application/json")
    public Response getTableForComponentClusterRelations (@QueryParam("tk") String token, @QueryParam("query") String query);

    @POST
    @Path("clusterQueryClientObject")
    @Produces("application/json")
    public Response getTableForClientClusterRelations (@QueryParam("tk") String token, @QueryParam("query") String query);
}
```

Figure 26.: REST web service interface

Figure 26 shows an extract of the interface of the developed services, each one has a specific address either the encrypted login credentials, or the Json data table.

- *getLogin* is used by the embedded visualization to perform login on the platform. It an encrypted user/password pair and returns a valid temporary *OAuth* access token.
- *getTableForComponent* receives, from the external embedded component, an access token and a query for a class level analysis as parameter. Returns a valid Json string containing a table with the data for the visualization.
- *getTableForClient* receives, from the HTML client, an access token and a *request* object containing a query for a class level analysis. It returns a valid Json string containing a table with the data for visualization.
- *getTableForClientCluster* receives, from the HTML client, an access token and a *request* object containing a query for a cluster level analysis. Returns a Json string containing a table with the data for visualization.

4.2. Server side

- *getTableForComponentCluster* receives, from the external embedded component, an access token and a Json string with the query for a cluster level analysis. Returns a Json string containing a table with the data for visualization.
- *getTableForComponentClusterRelations* receives, from the external embedded component, an access token and a Json string with the query for an instance level analysis. Returns a Json string containing a table with the data for visualization.
- *getTableForClientClusterRelations* receives, from the HTML client, an access token and a *request* object containing a query for an instance level analysis. Returns a Json string containing a table with the data for visualization.

Each one of these services parses the query and fills the output table with data retrieved from an Enterprise Java Bean(EJB) of the *Oobian Kserver*.

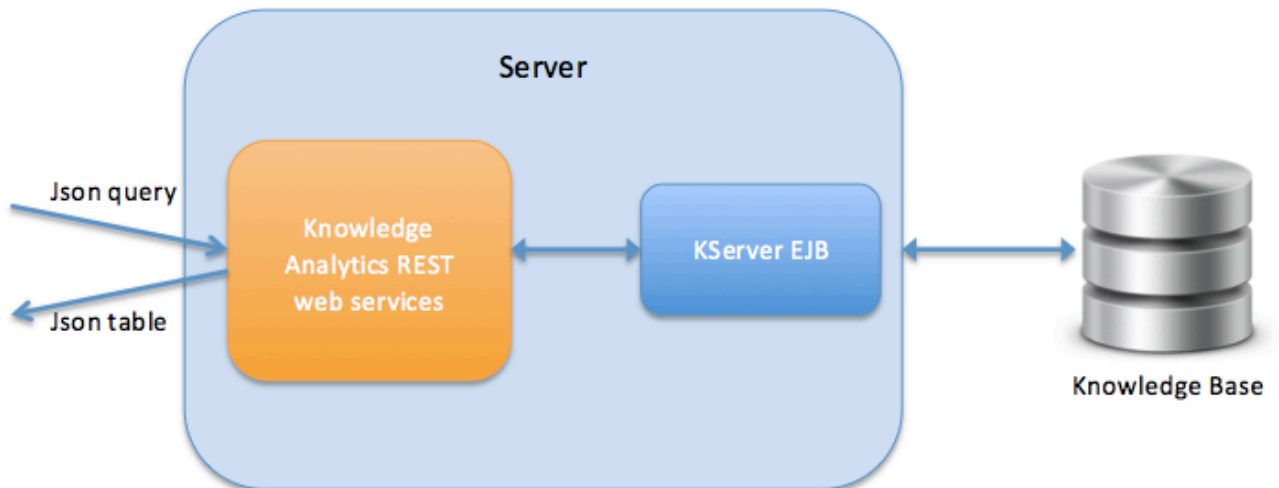


Figure 27.: Server side block diagram

4.3. Visualization Embedding

4.3 VISUALIZATION EMBEDDING

So far in this chapter it were presented and described some of the key aspects of the development stage. It was explained how the client and the server side of the component were developed. In this section it is described another one, the visualization embedding.

One of the goals Maisis had for the project was to allow users to embed visualizations, charts or tables, (created in Oobian) into external web sites for example personal, or company web pages.

As explained in Chapter 3, the proposed solution was to develop a feature to create a custom *HTML* element, a *div*, with custom *HTML 5 data-** attributes. These attributes were then read by a *JavaScript* library would retrieve the query string, make an HTTP request to the correspondent web service, obtaining the data table and then render the visualization embedded into the web page content. Figure 28 shows the developed pop-up with the HTML code of the *div* element that can be embedded into an web page.



The image shows a window titled "Embed component" with a close button (X) in the top right corner. The window contains the following text:

To include this visualization in your web page, simply add the following headers:

```
<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.9.0/jquery.js"></script>
<script type="text/javascript" src="https://www.google.com/jsapi"></script>
<script type="text/javascript" src="https://dl.dropboxusercontent.com/u/6994337/knowledge.js"></script>
```

And copy the following div element to the body of your page:

```
<div id="knowledge1405958016086" data-knowledge-address="http://172.27.192.180:8080" data-knowledge-
service="/Knowledge/rest/kanalytics/clusterQueryData" data-knowledge-user="jujn9P4BXX+U2MGahmo0Vg==" data-knowledge-pass
="dZW+Md37p1sDdhNUaJcF6w==" data-knowledge-query="{ 'select': { 'operator': null, 'properties': [ ':numEmpregados' ] }, 'from': {
'instanceld': ':Mercado_10', 'classId': ':Empresa', 'namespace': 'http://www.tice.pt/ontology/tice.owl#', 'objPropId':
':inverse_of_estaPresenteEm'}, 'where': { ': [ ] }, 'limit': 0, 'offset': 0, 'locale': 'pt' }" data-knowledge-chart="PieChart" data-knowledge-
title="Angola:Empresa / Número de colaboradores" style="width:600px; height:400px"></div>
```

For further instructions click here.

Figure 28.: Popup with the embedding code

The first problem encountered when developing this solution was about how to guarantee access to the data. Oobian server requires an *OAuth* access token as authentication. In the client, this token is created via login, with a registered user/password pair.

This seemed a good starting point, but having the user/password pair, or a token included in the component, in the "open", was obviously out of question.

The solution found to overcome this was simply to encrypt the user/password pair.

4.3. Visualization Embedding

In the process of creating the visualization, in the *Oobian Client*, users are logged in and have a valid user session. So, when generating the embed code in the client, user's credentials are encrypted using an *AES Cipher* algorithm with *Maisis* own secret key.

Having the credentials encrypted allowed to include them in the embed code. In the developed *JavaScript* library, a *login()* function reads the encrypted user/password pair from the embed component, makes a request to the login web service on the server. This web service decrypts the credentials, generates a valid access token and returns it in the response. Once obtained the token, it is just a question of including it in the HTTP with the query in order to authenticate the *KServer* and retrieve the Json data table to draw de visualization.

The next problem was how to deal with duplicate id's and how to allow the embedding of multiple visualizations into the same page.

To deal with the duplicate id problem, the choice was to append a *timestamp* to the id of the div, when generating the embedding code. This way, each generated component has a unique id.

To draw multiple visualizations in the same page, the solution was to include a function in the *JavaScript* library to create an array containing all the elements with a specific attribute, in this case, the custom *data-knowledge-query* attribute. Figure 29 shows this function.

```
function getElements(attr)
{
    var matchingElements = [];
    var allElements = document.getElementsByTagName('*');
    for (var i = 0, n = allElements.length; i < n; i++)
    {
        if (allElements[i].getAttribute(attr))
        {
            // Element exists with attribute. Add to array.
            matchingElements.push(allElements[i].id);
        }
    }
    return matchingElements;
}
```

Figure 29.: getElements function

4.3. Visualization Embedding

Once the array with the correct id's for the elements is built, for each one of those elements, the main *JavaScript* function gets the needed values from the custom *data-** attributes. It makes an HTTP request and draws the visualization with the data table presented on the HTTP response. Figure 30 shows an extract of that function.

```
function drawFromEmbed(){
    var token = login();

    for (var i= 0; i < divIds.length; i++){

        var element = document.getElementById(divIds[i]);
        var address = element.getAttribute("data-knowledge-address");
        var service = element.getAttribute("data-knowledge-service");
        var chartType = element.getAttribute("data-knowledge-chart");
        var chartTitle = element.getAttribute("data-knowledge-title");
        var query = element.getAttribute("data-knowledge-query");
        query = query.replace(/\'/g, '\');

        var url = address+service + "?tk=" + token;
        var jsonData = $.ajax({
            url: url,
            cache: false,
            type: "GET",
            data: {query: query},
            dataType: "json",
            async: false
        }).responseText;
    }
}
```

Figure 30.: Extract from the drawFromEmbed function

Chapter 4 focused on the development stage of the master work. Here, the reader found a technical perspective of the solution, with the exposition and explanation of some of the key aspects and choices made during the development process.

ACHIEVED RESULTS

The purpose of this chapter is to present some results that were obtained with *Knowledge Analytics* component to test its development. As this master work was developed in a professional environment, two real world scenarios, that use *Oobian platform* and contain real data were selected to test the Analytics module. Being so, this chapter is divided in two sections describing each one of those usage scenarios. In each of the sections is presented a brief description of the test scenario and a set of visualizations created in order to test the various features of the querying process, including simple queries and more complex queries with filters about the data present on the correspondent knowledge base.

5.1 560.PT - PORTUGAL BUSINESS NETWORK

560.PT - Portugal Business Network is an on-line platform that houses information about Portuguese Communities abroad, the *Portuguese Diáspora*. Part of it is a showcase of Portuguese companies present on foreign markets.

Some of the goals of this platform are:

- Promote the internationalization of companies and exportation of Portuguese products and services;
- Promote the contact between registered companies;
- Create business opportunities for Portuguese entrepreneurs abroad;
- Create networks between Portuguese people of different areas to promote Portugal's image, culture, interests and communities around the world.

The platform is based on the Oobian platform. In this section, several visualizations created with the *Knowledge Analytics* module will be presented.

5.1. 560.PT - Portugal Business Network

For each one of the visualizations it is also included the correspondent query that was sent to the server.

The data present in the knowledge base is structured according to the hierarchy shown in figure 31.

Class level

To start testing at the class level, the class *Empresa* was chosen. This class has 130 instances.

The properties of class *Empresa* are the following:

- Business volume;
- Creation year;
- Gross volume added;
- NACE code;
- Number of employees;
- Ratio (gross value added/business volume) %.

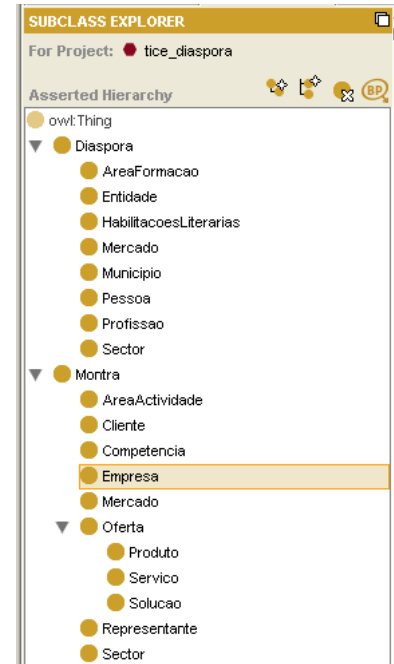


Figure 31.: 560.pt class hierarchy

5.1. 560.PT - Portugal Business Network

A simple visualization that can be created is the distribution of the number of companies by their creation date. Figure 32 shows an *area chart* exhibiting that evolution. To get that visualization, a simple query asking for the creation date without filters is enough, as listed below.

```
select : {
  operator : "null",
  properties : [ ":anoCriacao" ] },
from : {
  contextId : ":Empresa",
  namespace : "http://www.tice.pt/ontology/tice.owl#"
},
where : { '' : [ ] },
limit : 0,
offset : 0,
locale : "en"
```

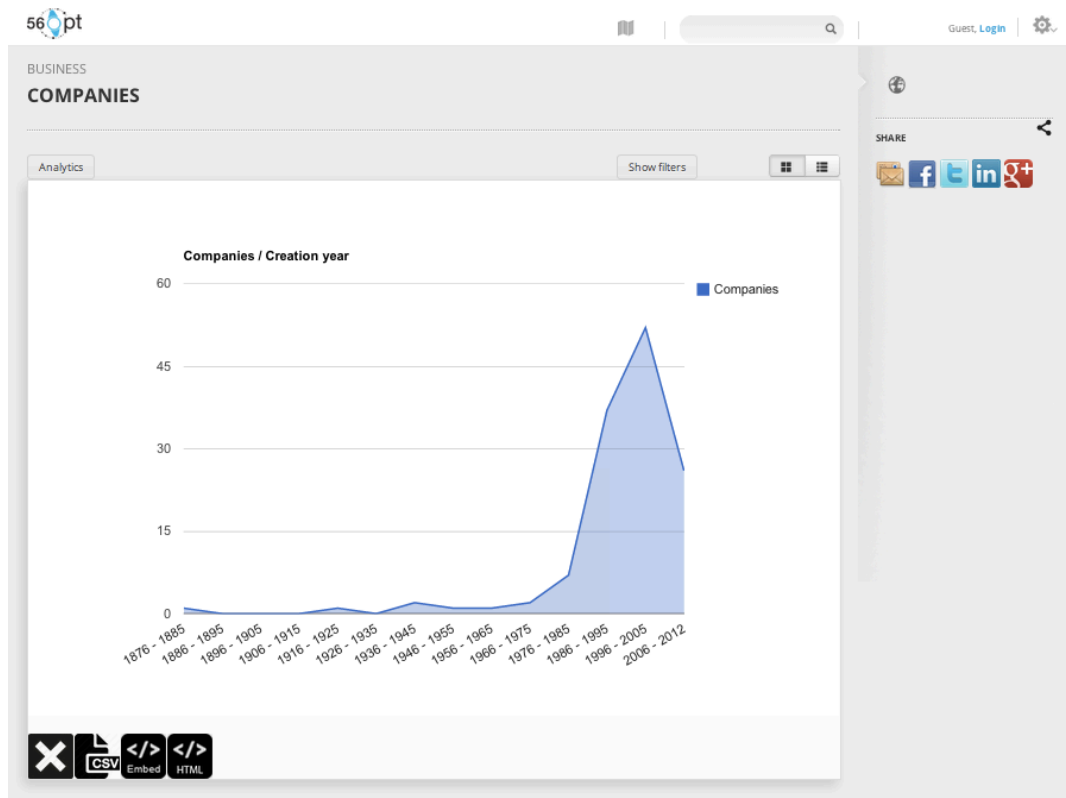


Figure 32.: Area chart: N^o of companies / Creation date

5.1. 560.PT - Portugal Business Network

A simple pie chart can be used to visualize the size of the companies, based on the number of employees, as depicted in Figure 33. Moving the mouse over each "slice" of the chart it is possible to see the exact number of companies.

The query, once again is simple:

```
select : {
  operator : "null",
  properties : [ ":numEmpregados" ] },
from : {
  contextId : ":Empresa",
  namespace : "http://www.tice.pt/ontology/tice.owl#"
},
where : { '' : [ ] },
limit : 0,
offset : 0,
locale : "en"
```

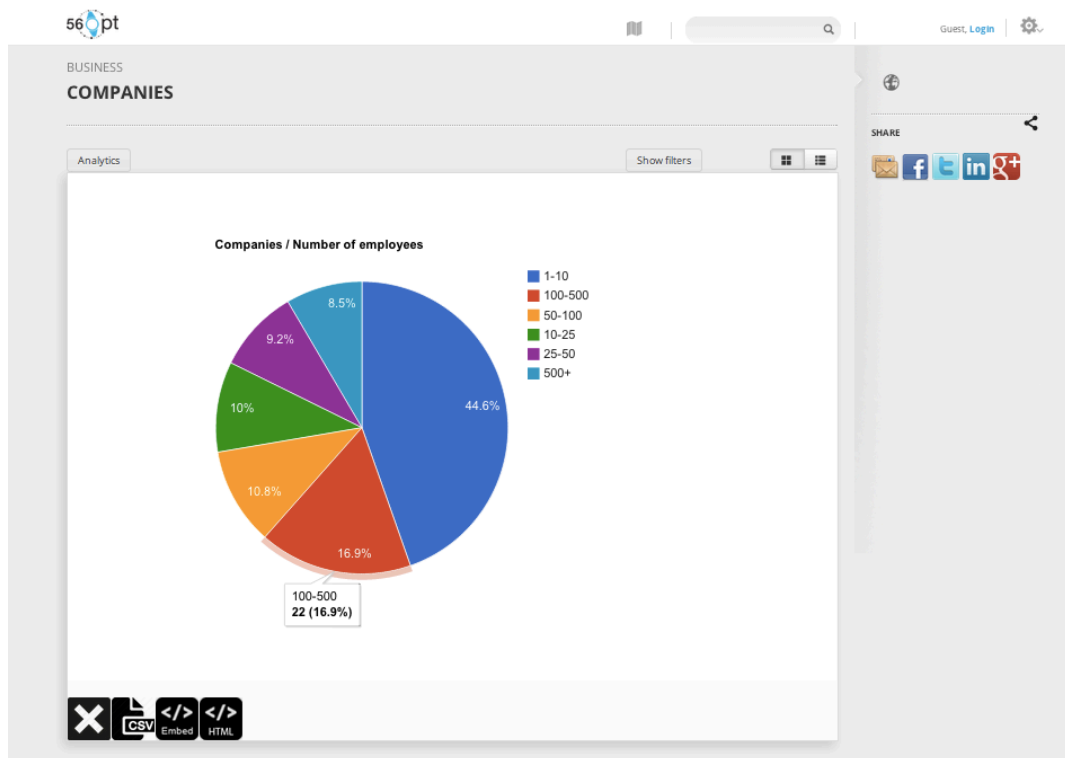


Figure 33.: Pie chart: % of companies / N^o employees

5.1. 560.PT - Portugal Business Network

Using filters it is possible to perform a deeper analysis of the data. For example, figure 34 shows the Ratio (gva/business volume)% of companies with 500 or more employees. This ratio can be explained as the margin of profit of the biggest companies present in the knowledge base.

The query that returns this result contains a filter in the *where* field:

```
select : {
  operator : "null",
  properties : [ ":racio" ] },
from : {
  contextId : ":Empresa",
  namespace : "http://www.tice.pt/ontology/tice.owl#"
},
where : { ":numEmpregados" : ["500+"] },
limit : 0,
offset : 0,
locale : "en"
```

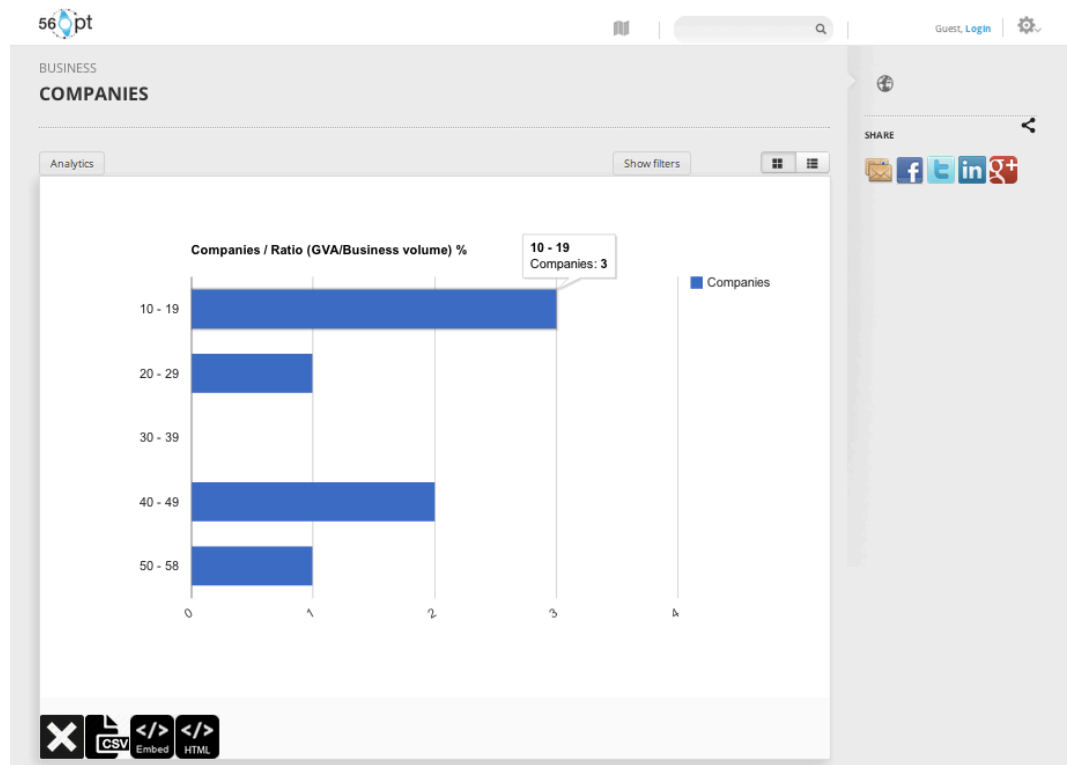


Figure 34.: Bar chart: N^o of companies / Ratio of companies with 500+ employees

5.1. 560.PT - Portugal Business Network

Instance level

At the instance level, it is possible to create visualizations about the relationships of an instance. For example, when analysing Portugal as a market, it is possible to create a *Donut* with all the relationships of *Portugal*, as depicted in Figure 35, that shows that most of the instances related to Portugal are from the *Solutions* class.

When creating a chart about the relations of an instance, the field *properties* of the query is empty because the aim of analysis is whole instance and not any property in particular.

```
select : {
  operator : "null",
  properties : [ ] },
from : {
  contextId : ":Mercado_1",
  namespace : "http://www.tice.pt/ontology/tice.owl#"
},
where : { '' : [ ] },
limit : 0,
offset : 0,
locale : "en"
```

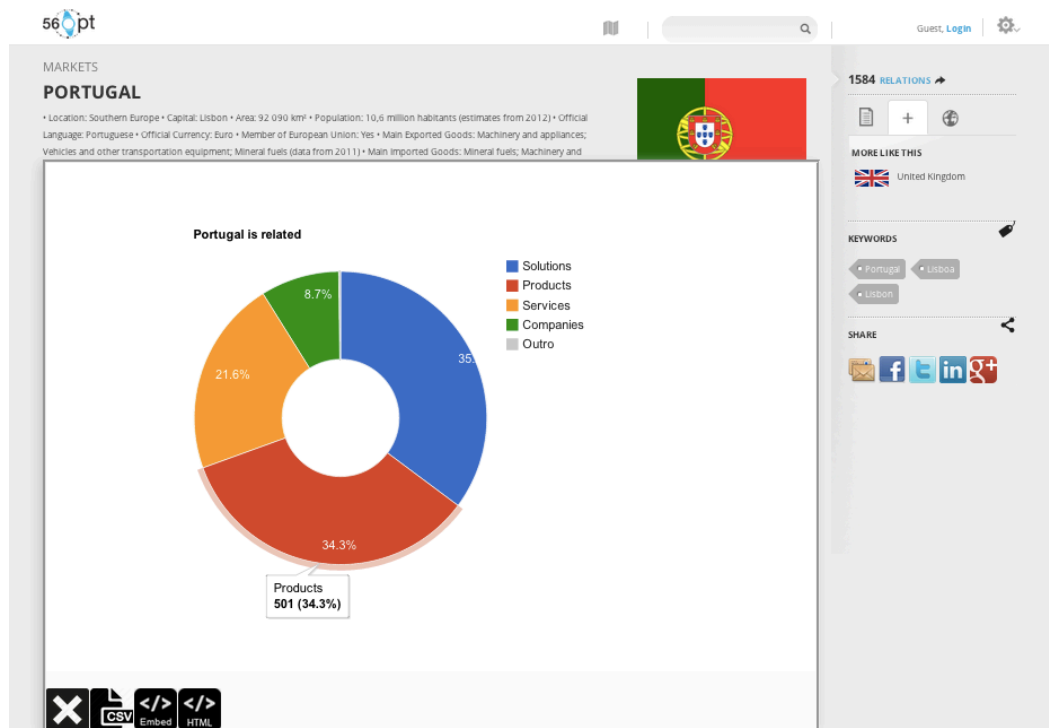


Figure 35.: Donut chart: Relations of instance Portugal

5.1. 560.PT - Portugal Business Network

Cluster level

Oobian HTML client groups instances related to a particular instance in clusters. In the previous chart, it was presented a "donut" with the relations of Portugal as a market. It is possible to drill into each one of those clusters and create visualizations about them.

Figure 36 shows a pie chart about the family of *Services* that are related to *Portugal*, that is returned as the answer to the query below.

```
select : {
  operator : "null",
  properties : [ ":familiaServico" ] },
from : {
  instanceId : ":Mercado_1",
  classId : ":Servico",
  namespace : "http://www.tice.pt/ontology/tice.owl#",
  objPropId : ":temOferta_Servico'"
},
where : { '' : [ ] },
limit : 0,
offset : 0,
locale : "en"
```

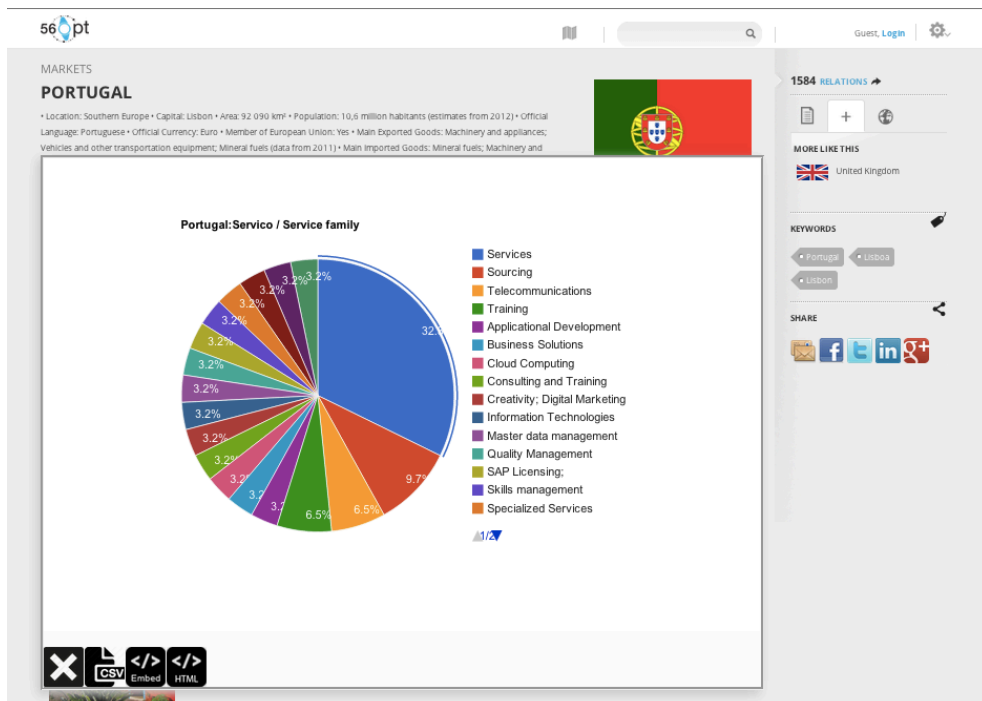


Figure 36.: Pie chart: family of Services related to Portugal

5.2. Joobian

5.2 JOOBIAN

Joobian is a mobile application, developed by *Maisis* that extracts and indexes job offers and opportunities from several sources and for several countries (Portugal, Brasil, Uk, USA, among others) and matches them according to user's skills.

Those skills can either be gathered from the user's *LinkedIn* account or manually inserted into the application. This matching translates into an app that personalises job search and presents job offers that are made for the user.

On the basis of the application is the Oobian platform.

This test scenario was chosen due to the massive amount of data present on the knowledge base. On the tested knowledge base it was present a portion of the index used by the application, containing information gathered during nearly one month, from 2014-06-18 to 2014-07-17. In this porting there are approximately 350 000 job offers (the entire index has over 5 million), matching around 2 300 skills, from about 37 000 cities and published by more than 60 000 companies. Figure 37 shows the class hierarchy of the data on the knowledge base. Particularly, the class *Job Offer* has the following data properties:

- City;
- Company;
- Country;
- Creation date;
- Formated location;
- Job date;
- Skills.

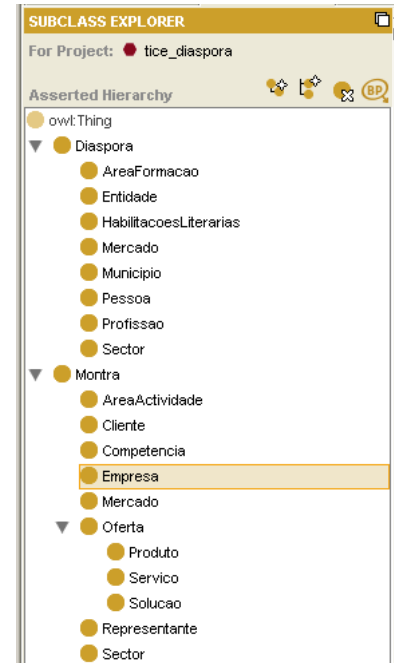


Figure 37.: Joobian class hierarchy

5.2. Joobian

The first test consists in a bar chart depicting the fifteen cities with more job offers in the knowledge base. The query to obtain this data is simple, with a limit of 15 to set the number of results to get. Figure 38 shows the result of the following query:

```
select : {
  operator : "null",
  properties : [ ":city" ] },
from : {
  contextId : ":JobOffer",
  namespace : "http://www.owl-ontologies.com/CV.owl#" },
where : { ' ' : [ ] },
limit : 15,
offset : 0,
locale : "en"
```

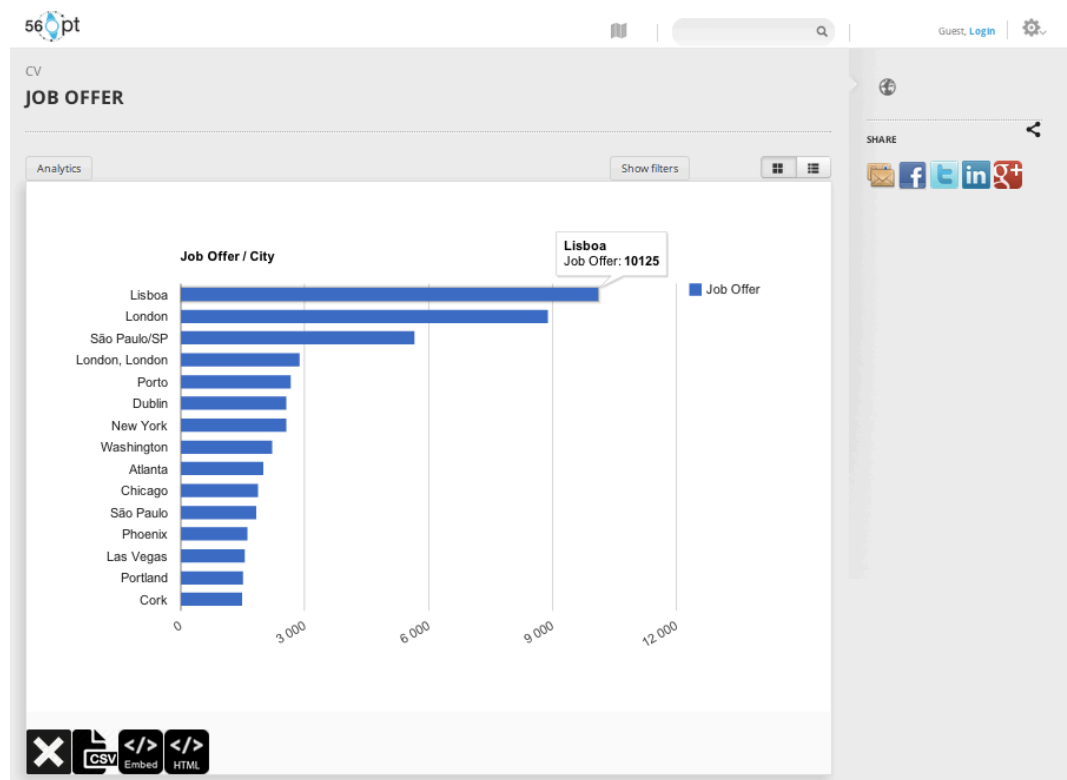


Figure 38.: Bar chart: 15 cities with more Job Offers

5.2. Joobian

Adding a filter to the previous query, as listed below, it is possible to create a chart for example with the fifteen cities with more job offers in Portugal, as depicted in figure 39.

```
select : {
  operator : "null",
  properties : [ ":city" ] },
from : {
  contextId : ":JobOffer",
  namespace : "http://www.owl-ontologies.com/CV.owl#" },
where : { ":country" : [ "Portugal" ] },
limit : 15,
offset : 0,
locale : "en"
```

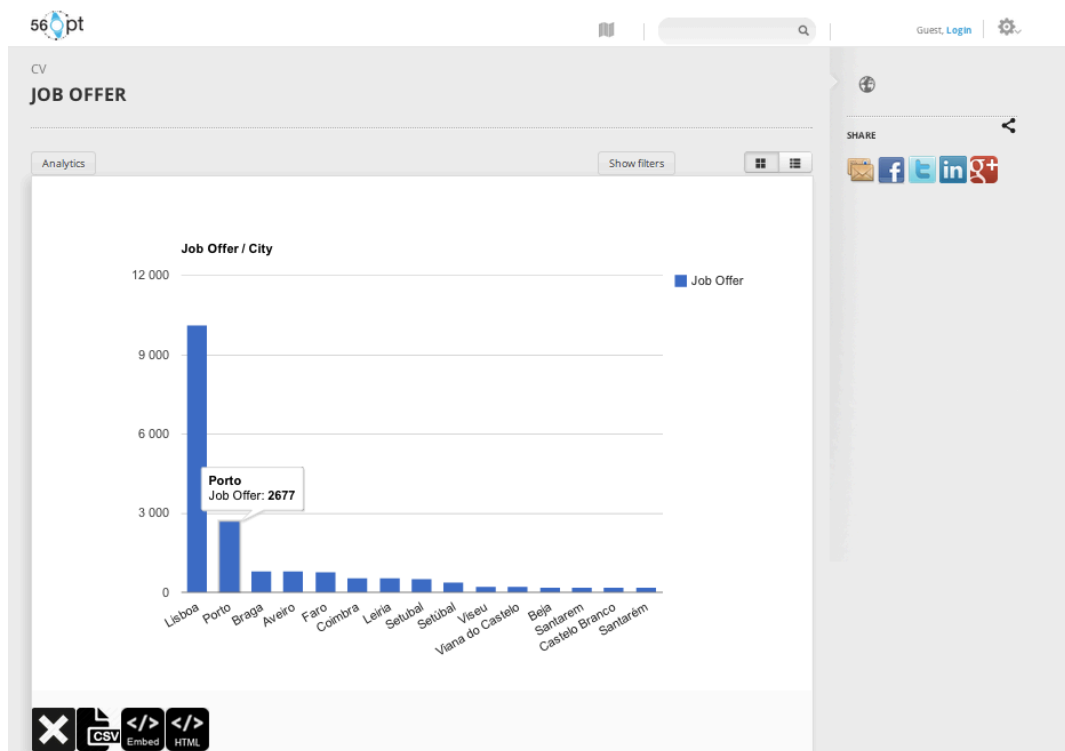


Figure 39.: Column chart: 15 cities with more *Job Offers* in Portugal

5.2. Joobian

Using multiple filters allows users to perform deeper analysis and reach more concrete results. For example, figure 40 shows the evolution of the number of job offers matching the skill *Java* over a period of one month, from 2014-06-30 to 2014-07-29 for the city of Porto. The chart in figure 40

```
select : {
  operator : "null",
  properties : [ ":creationDate" ] },
from : {
  contextId : ":JobOffer",
  namespace : "http://www.owl-ontologies.com/CV.owl#",
where : { ":skills" : [ "Java" ], ":city" : [ "Porto" ] },
limit : 0,
offset : 0,
locale : "pt"
```

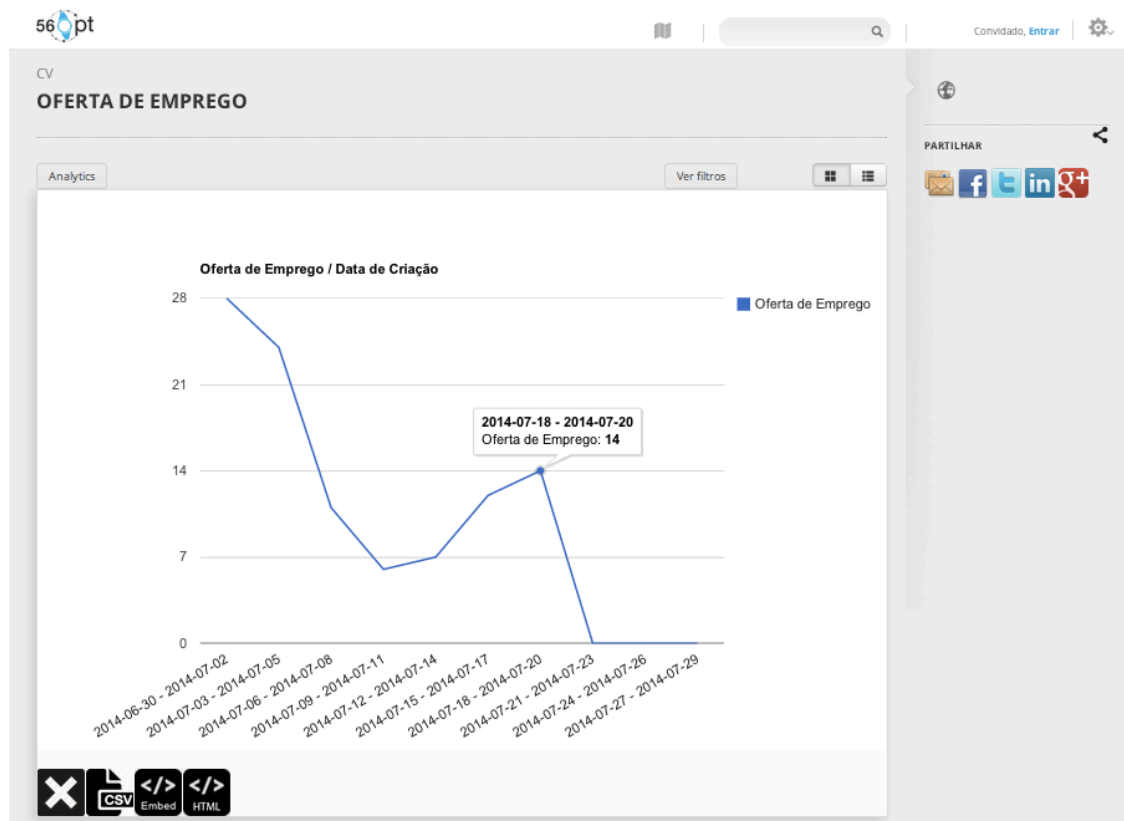
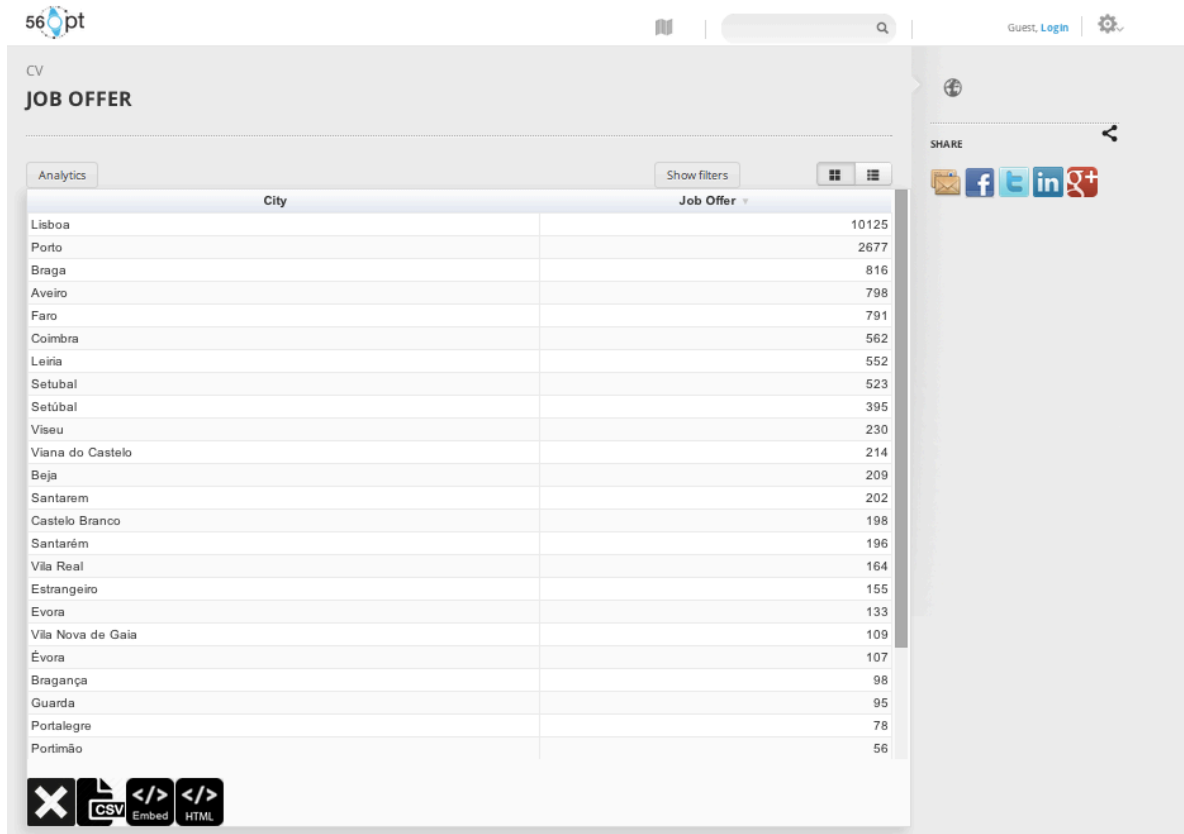


Figure 40.: Line chart: Evolution *Job Offers* matching Java in Porto

5.2. Joobian

Every one of the charts showed so far are based on the Json data table specified at Chapter 3, which means that every analysis made so far can simply be depicted by a simple table, as the one shown in Figure 41.



The screenshot shows a web application interface for '56pt'. The main content area displays a table titled 'JOB OFFER' with two columns: 'City' and 'Job Offer'. The table lists 15 cities and their corresponding number of job offers. The interface also includes a search bar, user options (Guest, Login), and social sharing buttons (Facebook, Twitter, LinkedIn, etc.).

City	Job Offer
Lisboa	10125
Porto	2677
Braga	816
Aveiro	798
Faro	791
Coimbra	562
Leiria	552
Setubal	523
Setúbal	395
Viseu	230
Viana do Castelo	214
Beja	209
Santarem	202
Castelo Branco	198
Santarém	196
Vila Real	164
Estrangeiro	155
Evora	133
Vila Nova de Gaia	109
Évora	107
Bragança	98
Guarda	95
Portalegre	78
Portimão	56

Figure 41.: Table: 15 cities with more *Job Offers* in Portugal

5.2. Joobian

Embedding multiple charts

One of the main goals of this work was the capability of embedding multiple charts, created in the Oobian client, in an external web site.

Figure 42 depicts a web page where multiple different charts were embedded, creating a live dashboard that allows users to keep track of evolutions or changes in information present on the knowledge base.

In this example it is presented a pie chart with the distribution of job offers by country, a bar chart with the fifteen cities with more job offers in Portugal, a column chart with the fifteen skills most requested in Porto, and the evolution of the number of job offers matching Java in Porto over a period of about one month.

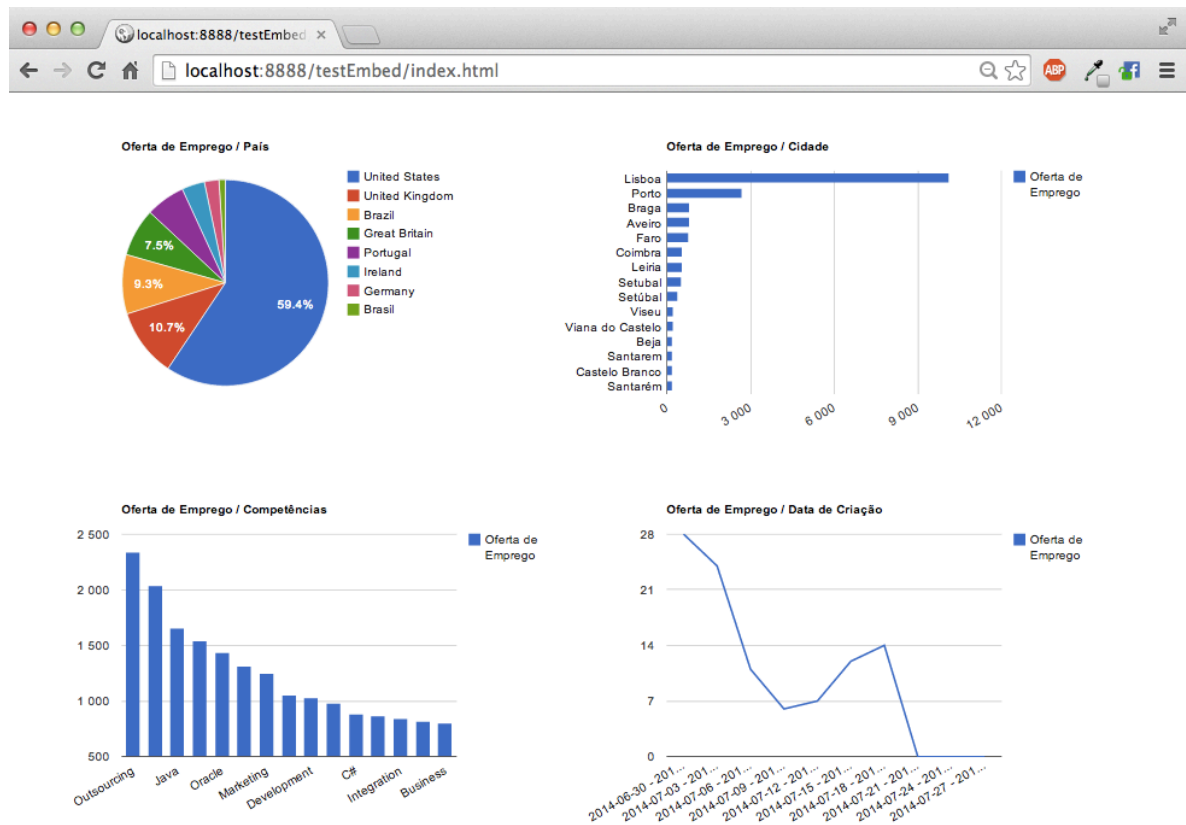


Figure 42.: "Dashboard" page with multiple charts

In chapter 5, it was made an exposition of the achieved results. Two case studies emulating real world scenarios were tested and the results were achieved presented to the reader with the help of several illustrations depicting each test that was made.

CONCLUSION

Thanks to InovaRia's Genius Trainee Program, this master was developed in a professional context, inside a company, focused on a real scenario. As said in the first chapter of this document, the master work is composed of two main parts, a theoretical one, and a practical one.

The starting point was the integration in the company, with the study of the company process of work and an overview of the developed products. Then, it was important to define the real problem addressed by this work and which goals were intended to be accomplished. These goals were:

- to make a theoretical study and to understand some key concepts as:
 - Oobian platform;
 - Ontologies, Semantic Web and Linked Data;
 - Data visualization tools and techniques;
- to formalize querying mechanisms to obtain data from the Oobian core engine (Knowledge Server - Kserver).
- to specify a format for the resultant data set.
- to propose mechanisms and techniques to analyse the data about the ontologies present on the Oobian knowledge management platform;
- to create a visual user interface to present the generated reports;
- to create a mechanism to embed those visualizations into external websites.

Once the problem and goals were defined, it was important to make a study about some key concepts like semantic web, ontologies, linked data, visualization tools and techniques. Understanding these concepts was fundamental to develop of the master project. After that, the study focused on existent tools and platforms that offer similar features as those intended to be developed. Studying these tools allowed to gain a perspective about the state of the

6.1. Future Work

art on this field, about what already exists and what is or is not possible to be done. After this, a raw solution was proposed. It was defined the general architecture of the solution, the query and output formats where specified, the final visualization types where chosen and the restrictions of each type where defined.

This solution was reviewed and approved by the company supervisor and the specification process began.

The requirement specification was aligned with the proposed solution and requirements themselves were defined always under the supervision and approval of the team. Chapters 2 and 3 reported and discussed these first working steps.

Once the specification phase was completed the development phase began. This was the longest stage of the master work. As the core of the development was completed the testing began, first in small scale, with small data sets followed by testing with bigger data sets until the component was stable and optimized. Chapters 4 and 5 were dedicated to expose the development details and the testing, showing the queries and their results in graphical mode. It was a very interesting work. There has been a growth in the field of business analytics during the last years, and the use of semantic web and linked data on the analysis process gives a real edge to an analytics platform. Another interesting part of this work was the chance to work on a professional environment. It was demanding, but rewarding at the same time, as it was a way of learning a lot, in a real-world scenario.

6.1 FUTURE WORK

With the development of this master work, Maisis attained the general goal of adding visualization and analytics features to *Oobian*. More than creating only a visual representation of data, it was important to specify and develop solid basis for the future. The developed solution accomplished this goal and with a query format defined and web services answering to those queries and returning data in a standard format on the server side, it is possible to evolve the client side and develop new features with low effort.

Some of the evolutions and features that can be developed in the future include:

- *Drilling features*: When presenting for example a bar chart about segmented data, users could click on the bar to re-define the intervals of analysis, drilling up and down the data. To do this, one possible approach is to reset the query according to the user interaction, retrieve a new data table from server and re-draw the visualization.
- *Property evolution tracking*: With the development of analysis scheduling features, a good improvement is the addition of a feature for setting a scheduler to run a query from time to time in order to register and keep track of the evolution of a pre-determined property over a pre-determined period of time.

6.1. Future Work

- *Google Spreadsheets integration:* As the visualizations are created using *Google Visualization API* and are based on *Json* data tables with the correspondent structure, an interesting evolution of the platform would be the integration with *Google Spreadsheets*, allowing users to save data tables and charts in their personal cloud space.

BIBLIOGRAPHY

- [1] Rafa E Al-qutaish. Quality models in software engineering literature : An analytical and comparative study. 6, 2010.
- [2] D Allen. *Getting things done: The art of stress-free productivity*. Penguin Books, 2002.
- [3] Ori Ben-Yitzhak, Nadav Golbandi, Nadav Har'El, Ronny Lempel, Andreas Neumann, Shila Ofek-Koifman, Dafna Sheinwald, Eugene Shekita, Benjamin Sznajder, and Sivan Yogev. Beyond basic faceted search. In *Proceedings of the 2008 International Conference on Web Search and Data Mining, WSDM '08*, pages 33–44, New York, NY, USA, 2008. ACM.
- [4] Tim Berners-Lee, James Hendler, and Ora Lassila. *The Semantic Web*, 2001.
- [5] Christian Bizer, T Heath, and T Berners-Lee. Linked data-the story so far. *International Journal on Semantic ...*, 2009.
- [6] Jacques Bughin, Michael Chui, and James Manyika. Clouds, big data, and smart assets: Ten tech-enabled business trends to watch. *McKinsey Quarterly*, 2010:1, 2010.
- [7] Mills Davis. Semantic Wave 2008 Report: Industry Roadmap to Web 3.0 & Multibillion Dollar Market Opportunities. *Executive Summary*, page 29, February 2008.
- [8] Patrick Hayes. RDF Semantics. *W3C Recommendation*, 10:1–45, 2004.
- [9] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*, 2011.
- [10] IBM Research. *Global Technology Outlook 2013*. Technical report, IBM Corporation, 2013.
- [11] Steve Lavalle, Eric Lesser, Rebecca Shockley, Michael Hopkins, and Krushwitz Nina. Big Data , Analytics and the Path From Insights to Value Big Data , Analytics and the Path From Insights to Value. *MIT Sloan Management Review*, 52:21–32, 2011.
- [12] Phillip Lord. *Components of an ontology*, 2010.
- [13] Natalya Noy and Deborah L. McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology*. Technical report, Knowledge Systems Laboratory Stanford University, 2001.

Bibliography

- [14] M Skjaeveland. Sgvizler: A javascript wrapper for easy visualization of sparql result sets. *Extended Semantic Web Conference*, pages 2–6, 2012.
- [15] Steffen Staab and Rudi Studer. *Handbook on Ontologies*, volume 2. Springer, second edition, 2009.
- [16] Fernanda B. Viegas, Martin Wattenberg, Frank van Ham, Jesse Kriss, and Matt McKeon. Manyeyes: A site for visualization at internet scale. *IEEE Transactions on Visualization and Computer Graphics*, 13:1121–1128, 2007.



REQUIREMENT SPECIFICATION

According to the FURPS+ methodology, in this appendix are presented, with bigger detail, the requirements of this master work.

When necessary, for each functional requirement it is included a graphic mockup, in order to help the total understanding of the intended final product. Some of the mockups represent a set of requirements, once the Oobian interface allows to represent many information within the same window. These mockups were presented to and validated by Maisis, and represent an approach to the aspect of the final solution.

The requirement presentation along this appendix follows a template, created to help on the comprehension of each requirement. The template is the following:

<Name of the requirement>

- **Requirement Id:** identification of the requirement;
- **Dependencies:** list of dependent requirements;
- **Priority:** priority of the requirement;
- **Description:** brief description of the requirement.

A.1. Functional Requirements

A.1 FUNCTIONAL REQUIREMENTS

The main goal of the final product is to represent information present on the Oobian platform, in a visual format. So, functional requirements are focused on the success of that presentation.

Html client integrated component

- **Requirement Id:** 1;
- **Dependencies:** -;
- **Priority:** Must;
- **Description:** One of the goals of this work was to develop and integrate a component in the existent Oobian client. The Html5 client is one of the existent clients of the Oobian platform, so the developed component should be integrated with the existent interface. Figure 43 depicts the inclusion of an "Analytics" button into the interface;



Figure 43.: Analytics button mockup

Class Level Analysis

- **Requirement Id:** 2;
- **Dependencies:** -;
- **Priority:** Must;

A.1. Functional Requirements

- **Description:** the component must allow users to perform an analysis of a class, and create a visualization, for instances counts about instances of that class;

Instance Level Analysis

- **Requirement Id:** 3;
- **Dependencies:** –;
- **Priority:** Must;
- **Description:** As depicted by figure 44, when viewing the details of an instance, the Oobian client presents details of the relations of that instance. The component must allow users to create a visualization of those relations, for example, a pie or a column chart;

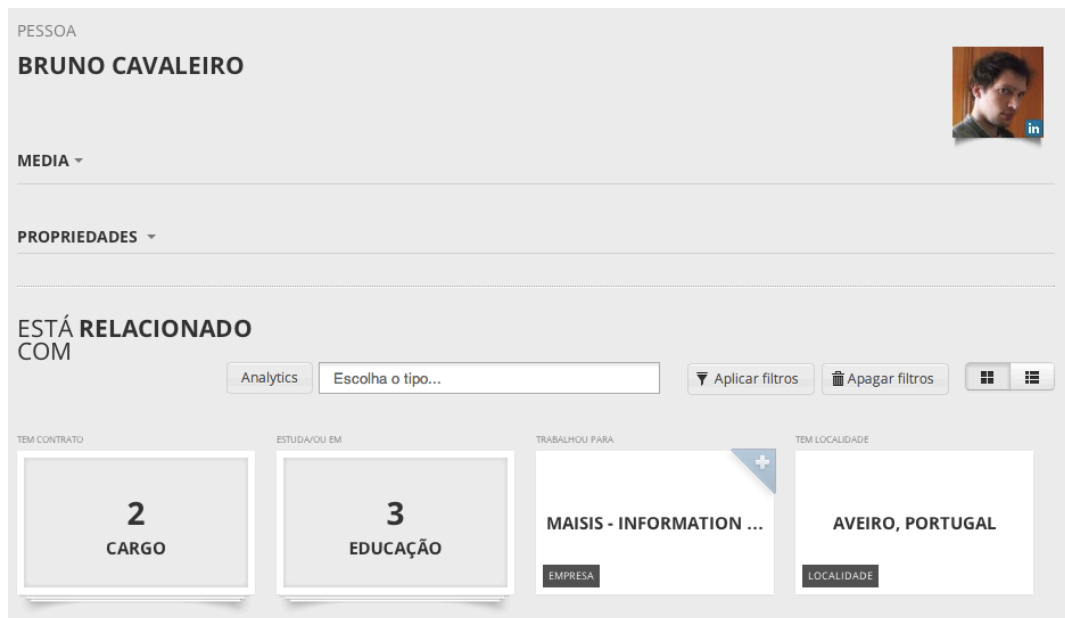


Figure 44.: Instance relations example

A.1. Functional Requirements

Cluster Analysis

- **Requirement Id:** 4;
- **Dependencies:** –;
- **Priority:** Must;
- **Description:** When navigating in an instance, Oobian client groups instances related to the current instance in clusters, as depicted by figure ?? so the component must allow users to make an analysis of the instances in that cluster and create a visualization about them.

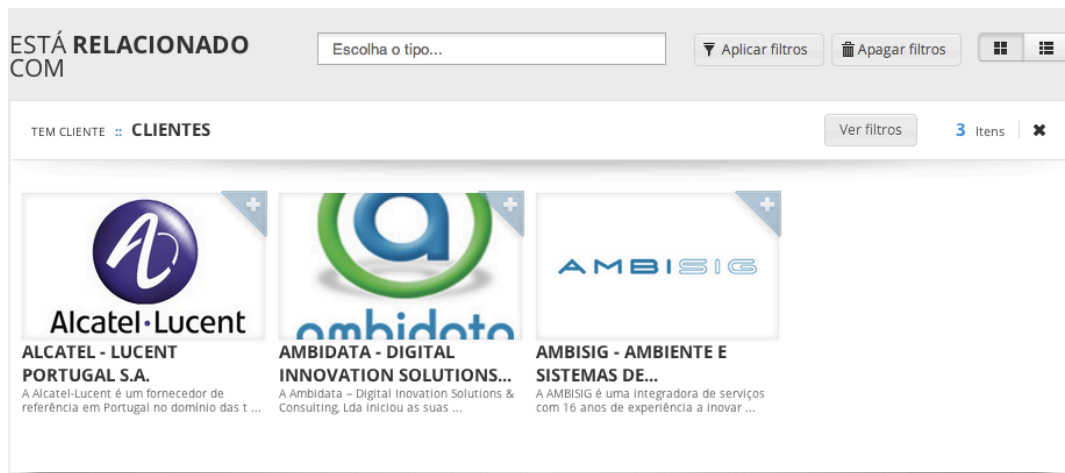


Figure 45.: Related instance cluster example

Data properties picker

- **Requirement Id:** 5;
- **Dependencies:** 1;
- **Priority:** Must;
- **Description:** In order to allow users to select which property to analyse, the component must have a picker that shows the data properties of the class or cluster.

A.1. Functional Requirements

Multiple visualization types support

- **Requirement Id:** 6;
- **Dependencies:** 2, 3, 4;
- **Priority:** Must;
- **Description:** The component must allow users to create different types of visualizations. The system must support the following types:
 - Column & Bar charts;
 - Line & Area charts;
 - Pie & Donut charts;
 - Data table;

Oobian is a platform prepared to deal with a large variety of information and knowledge, from data about company incomes to locations of historical places. To represent such variety of information it is important that the platform supports several types of chart.

A simple data table is the starting point of representing any set of data, as illustrated by figure 46.

	Name ▲	Salary	Full Time Employee
1	Alice	\$12,500	✓
2	Bob	\$7,000	✓
3	Jim	\$8,000	✗
4	Mike	\$10,000	✓

Figure 46.: Simple Data Table

To represent statistical data between properties, bar and column charts are always a good option, figures 47 48 are examples of those types of chart.

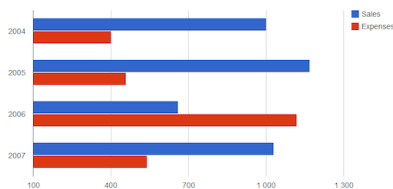


Figure 47.: Bar Chart

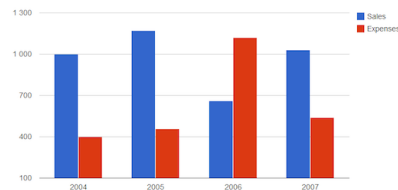


Figure 48.: Column Chart

A.1. Functional Requirements

Line and Area charts are both good to represent series of data related to time. The ideal usage for them is to represent trends or evolutions. Figures 49 and 50 are examples of these types.

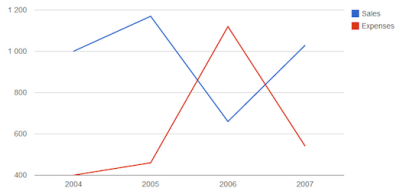


Figure 49.: Line Chart

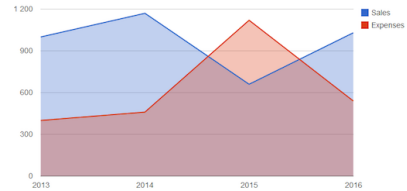


Figure 50.: Area Chart

To represent the numerical proportions the ideal type of visualization is a Pie, or Donut Chart, as illustrated by figure 51 and 52.

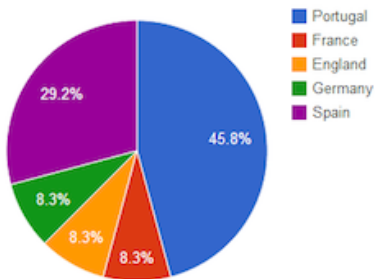


Figure 51.: Pie Chart

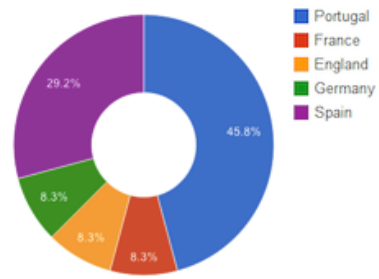


Figure 52.: Donut chart

A.1. Functional Requirements

Visualization Constrains

- **Requirement Id:** 7;
- **Dependencies:** 6;
- **Priority:** Must;
- **Description:** According to the selected property data type and chart type, the system must have mechanisms to validate the creation of that chart.

Each one of the previously presented types of visualization has specific characteristics. In order to be able to be represented, the data must follow a set of restrictions, according to the selected chart type, restrictions like the number of columns of the data table or the data type. These restrictions are presented in Table 4.

Chart Type	Result table format	X-Axis	Y-Axis	Optional
Bar chart	2 - N columns	label data type: any	value data type: number	–
Column chart	2 - N columns	label data type: any	value data type: number	–
Pie chart	2 columns	slice label data type: any	slice value data type: number	–
Line chart	2 - N columns	label data type: any	value data type: number	–
Area chart	2 - N columns	label data type: any	value data type: number	–

Table 4.: Chart Restrictions

A.1. Functional Requirements

Native Query Support

- **Requirement Id:** 8;
- **Dependencies:** –;
- **Priority:** Must;
- **Description:** Other of the main goals of this project was to formally define a querying mechanism. On chapter 3 is present the proposed solution, query language described by the grammar depicted by figure 53. So, according to the selected property and context of navigation, the component must create a valid sentence, containing the query. Below is an example of a valid sentence:

```
{
  select:{operator:"null",properties:["city"]},
  from:{
    instanceId:"Person_W-A0CgjJyz",
    contextId:"Education",namespace:"http://www.owl-ontologies.com/CV.owl#"
  },
  where:{
    ":countryCodeSource" : [ "GB" ], ":city" : [ "London" ]
  },
  limit:1,
  offset:1,
  locale:"pt"
}
```

A.1. Functional Requirements

```
1 grammar query_instance_grammar;
2
3 query:
4     '{"select':select,"from':from',"where':where',"limit':INTEGER','offset':INTEGER',"locale':STRING}'
5     ;
6 select:
7     '{"operator':STRING',"properties':["properties"]}'
8     ;
9 from:
10    '{"contextId':STRING,'(instanceId':STRING,)*',"namespace':STRING('objPropId':STRING)*}'
11    ;
12 where:
13     '{"filters}'
14     ;
15 properties:
16     STRING(','STRING)*
17     ;
18 filters:
19     filter(','filter)*
20     ;
21 filter:
22     STRING::["STRING(','STRING)*"]
23     |STRING::["INTEGER(','INTEGER)*"]
24     ;
```

Figure 53.: Proposed grammar

Google Visualization json table output

- **Requirement Id:** 9;
- **Dependencies:** –;
- **Priority:** Must;
- **Description:** Once chosen library for creating visualizations was *Google Charts Api*, the system must encode a json string according to the format accepted by the API. Below is an example of a json in that format.

```
{
  "cols": [
    {"id": "col_1", "label": "year", "type": "string"},
    {"id": "col_2", "label": "sales", "type": "number"},
    {"id": "col_3", "label": "expenses", "type": "number"}
  ],
  "rows": [
    {"c": [{"v": "2001"}, {"v": "3"}, {"v": "5"}]},
    {"c": [{"v": "2002"}, {"v": "5"}, {"v": "10"}]},
    {"c": [{"v": "2004"}, {"v": "8"}, {"v": "32"}]},
    {"c": [{"v": "2005"}, {"v": "3"}, {"v": "56"}]}
  ]
}
```

A.1. Functional Requirements

Embed HTML component

- **Requirement Id:** 10;
- **Dependencies:** 9;
- **Priority:** Must;
- **Description:** The system must be able to generate an HTML snippet to allow users to include the created chart into an external website. Figure 54 is a mockup of an embed popup where the generated html snippet is presented allowing users to copy it.

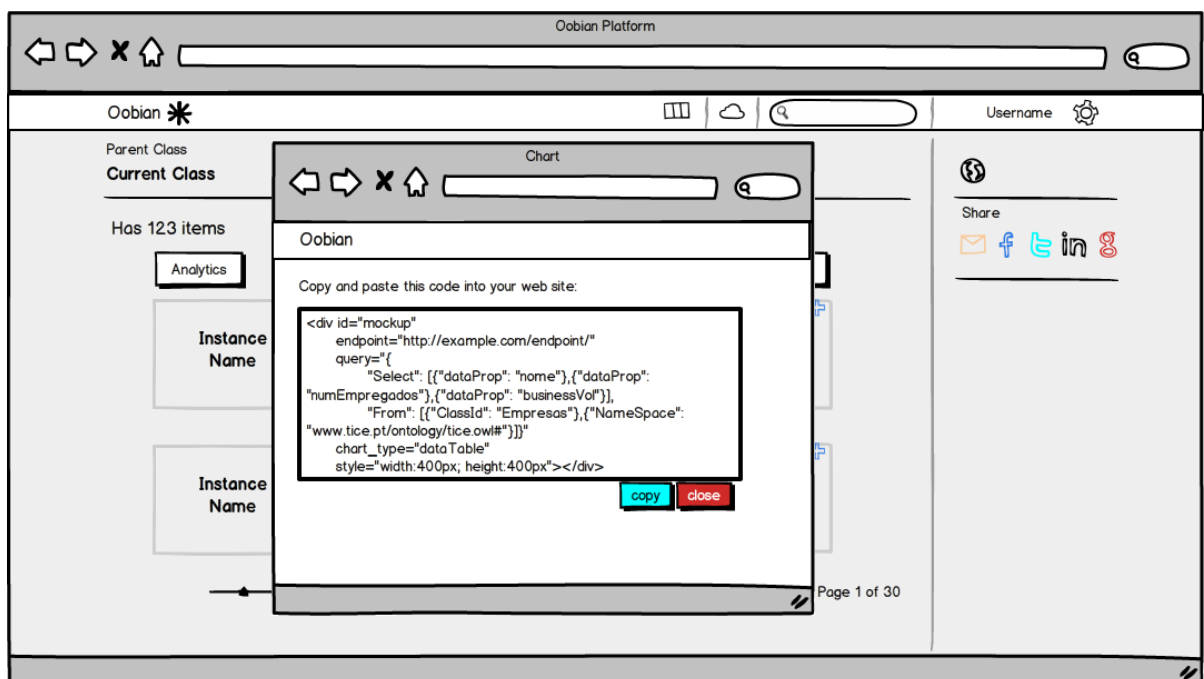


Figure 54.: Embed Html popup mockup

A.1. Functional Requirements

Self updating HTML component

- **Requirement Id:** 11;
- **Dependencies:** –;
- **Priority:** Must;
- **Description:** As said in the previous requirement, the system must be able to create an HTML component to allow users to embed a chart into a website.
Once this component is created it must be "alive" and reflect any changes that occur on the server, always providing an updated analysis.

Interval Result Segmentation

- **Requirement Id:** 12;
- **Dependencies:** 9;
- **Priority:** Must;
- **Description:** Depending on the data present on the platform, it is possible to have classes or clusters with data properties with numeric data types, as "xsd:int" or "xsd:float". Its also possible to have data properties with "xsd:date" or "xsd:datetime". For these numeric/date typed properties, the system must be able to segment the results in a number of intervals specified by the user. By doing this, it will allow to create visualizations that show an evolution, from a minimum to a maximum value.

Query Filter Support

- **Requirement Id:** 13;
- **Dependencies:** 8;
- **Priority:** Should;
- **Description:** In order to filter the number of results, the system should take in account the filters applied on the client, build a query with those filters and return a set of filtered results.

Query Limits Support

- **Requirement Id:** 14;
- **Dependencies:** 8;
- **Priority:** Should;

A.1. Functional Requirements

- **Description:** The component should be able to allow users to limit the number of results to obtain from the server. So, the query mechanism should be compatible with limits.

Query Offset Support

- **Requirement Id:** 15;
- **Dependencies:** 8;
- **Priority:** Nice;
- **Description:** It would be nice if the system allow users to set a number of items to skip before returning the results.

Query Operations Support

- **Requirement Id:** 16;
- **Dependencies:** 8;
- **Priority:** Nice;
- **Description:** It would be nice for the component to offer the possibility of including operations on the querying process. The querying mechanism should be compatible with operators as *Count*, *Sum*, between others.

Data table export as CSV

- **Requirement Id:** 17;
- **Dependencies:** 9;
- **Priority:** Nice;
- **Description:** All of the created visualizations are based on the data table output presented previously. It would be nice if the system allows users to export that data in the CSV format.

A.2. Non-Functional Requirements

A.2 NON-FUNCTIONAL REQUIREMENTS

In this section the non-functional requirements of the software component developed in the master work are presented. Non-functional requirements reflect qualities of the system and according to the FURPS methodology, act as constraints both for the system and for the development.

A.2.1 *Usability*

The pair *Effectiveness/Efficiency* is always an important pair of usability requirements for the satisfaction of the user. On the subject of user interface, this module is intended to be included on the Oobian Html client interface, so it has to be simple and user friendly, in line with the rest of the client.

Effectiveness

- **Requirement Id:** 18;
- **Priority:** Must;
- **Description:** The system must be able to answer the needs of the various users. Ultimately it must allow users to create charts about data present on the platform.

Efficiency

- **Requirement Id:** 19;
- **Priority:** Must;
- **Description:** In order for a system to be efficient, it has to provide results in a way that requires a low amount of effort and time from the user.
On the *Knowledge Analytics* module, this way is the quick "Analytics" button that will be present on the interface.

A.2.2 *Design*

Oobian Platform

- **Requirement Id:** 20;
- **Priority:** Must;
- **Description:** Oobian platform is the ecosystem where the information that the *Knowledge Analytics* module consumes is located, so the module must be designed in conformity with it.

A.2. Non-Functional Requirements

A.2.3 Interface

Oobian Client

- **Requirement Id:** 21;
- **Priority:** Must;
- **Description:** *Knowledge Analytics* is not a standalone component, it is always part of Oobian Html client, so, users must have an Oobian Client in order to use the final Knowledge Analytics module.

A.2.4 Physical

Hardware Requirements

- **Requirement Id:** 22;
- **Priority:** Must;
- **Description:** The minimum hardware requirements to support an Oobian client are the following:
 - Mac OS 9.0 or higher, Windows 2008 or higher;
 - Microsoft Silverlight (for the Silverlight Insight client);
 - Intel Dual Core 1.7 or higher;
 - 2GB of ram memory;
 - 200 Megabytes of free HDD space;
 - Monitor;
 - Mouse and Keyboard;