



**Rui Maia Barreto  
Sacchetti**

**Plataforma para visualização de dados clínicos  
baseada em software open source**

**Framework for clinical data visualization based on  
open-source software**





**Rui Maia Barreto  
Sacchetti**

**Plataforma para visualização de dados clínicos  
baseada em software open source**

**Framework for clinical data visualization based on  
open-source software**

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica do Doutor José Luís Oliveira, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do Engenheiro Renato Pinho Rodrigues, da empresa BMD Software.



Aos meus pais.



**o júri / the jury**

presidente / president

**Prof. Doutor Augusto Marques Ferreira da Silva**

Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

**Prof. Doutor Rui Pedro Sanches de Castro Lopes**

Professor Coordenador do Instituto Politécnico de Bragança

**Prof. Doutor José Luis Guimarães Oliveira**

Professor Associado da Universidade de Aveiro





**palavras-chave**

Análise de dados, Visualização de Informação, Business Intelligence, Gestão de Conhecimento na Saúde.

**resumo**

Os profissionais de saúde enfrentam constantemente o desafio de analisar grandes quantidades de dados médicos na procura de respostas para questões complexas. Além disso, estes profissionais não têm conhecimentos de linguagens para consultas de bases de dados. Portanto, é necessário providenciar aos profissionais de saúde ferramentas que lhes permitam visualizar relatórios, gráficos e *dashboards*, sem que seja necessário interagir com linguagens de consulta, idealmente nas plataformas onde os dados são gerados.

Integrar soluções de *Business Intelligence* (BI) em aplicações já existentes surge como uma possível solução para este problema, providenciando recursos aos utilizadores finais que lhes permitam explorar dados de uma forma simplificada. Deste modo, não só é valorizada a aplicação onde ocorre a integração, como é eliminada a necessidade de plataformas destacadas para análise de dados.

Esta dissertação começa por analisar se a integração de soluções BI na área da saúde é viável, apresentando as principais características de dados clínicos, bem como um estado de arte das soluções *Business Intelligence* existentes. De seguida, é apresentada uma proposta de solução, onde o desenvolvimento de uma solução integrável é detalhadamente descrito. Por fim, serão apresentados casos de estudo onde a integração do trabalho desenvolvido ocorreu, de forma a avaliar a sua aplicabilidade e integrabilidade.



**keywords**

Data analysis, Information visualisation, Business Intelligence, Knowledge management in healthcare.

**abstract**

Healthcare providers are faced with the challenge of going through large amounts of data in search of answers to complex questions. On top of that, medical staff typically don't have database knowledge, and so, it is necessary to provide them with tools that allows the visualization of reports, charts and dashboards, without having to interact or even see query languages, ideally on the same platforms where data is generated.

Embedding Business Intelligence (BI) has emerged as a possible solution to this issue, providing end-user with tools that enables them to explore data in a simplified way. Thus, the application where the solution was embedded not only is enhanced, but the necessity for data analysis stand-alone tools is eliminated.

This thesis starts by analyzing if integrating a BI solution into medical platforms is plausible, by presenting the main characteristics of clinical data, followed by an overview of the state of the art on open-source Business Intelligence software. Then, a solution proposal is presented where the development of an embedded solution is pragmatically addressed. Finally, case studies where the integration of the work developed was integrated will be presented, in order to evaluate its applicability and integrability.



# Contents

<b>Contents</b>	<b>i</b>
<b>List of Figures</b>	<b>iii</b>
<b>List of Tables</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivating . . . . .	2
1.2 Objectives . . . . .	2
1.3 Document organization . . . . .	3
<b>2 Concepts</b>	<b>5</b>
2.1 Data Visualization . . . . .	5
2.2 Business Intelligence . . . . .	6
2.3 Business Intelligence in Healthcare . . . . .	8
2.4 Data in Healthcare . . . . .	9
2.5 Open Source Software . . . . .	11
2.6 Licensing . . . . .	13
2.7 Conclusion . . . . .	15
<b>3 State-of-Art</b>	<b>17</b>
3.1 Business Intelligence Suites . . . . .	17
3.2 Business Intelligence Tools . . . . .	21
3.3 Tools Comparison . . . . .	22
3.4 Conclusion . . . . .	24

<b>4</b>	<b>Solution Analysis</b>	<b>27</b>
4.1	Requirements . . . . .	28
4.2	First Approach - Metabase . . . . .	30
4.3	Second Approach - Pentaho . . . . .	36
<b>5</b>	<b>Pentaho Implementation</b>	<b>45</b>
5.1	Pentaho Report Designer . . . . .	45
5.2	Pentaho BI Server . . . . .	55
5.3	Integration . . . . .	60
5.4	Conclusion . . . . .	63
<b>6</b>	<b>Results and discussion</b>	<b>65</b>
6.1	Use cases . . . . .	65
6.2	Summary and contributions . . . . .	72
6.3	Limitations . . . . .	73
<b>7</b>	<b>Conclusion</b>	<b>75</b>
7.1	Future work . . . . .	77
	<b>Bibliography</b>	<b>79</b>
	<b>Appendix A: MDTeam Report</b>	<b>83</b>
	<b>Appendix B: CardioBox Report</b>	<b>84</b>
	<b>Appendix C: PACScenter Report</b>	<b>85</b>

# List of Figures

2.1	Decision-making life cycle . . . . .	6
2.2	Business Intelligence elements overview. . . . .	7
2.3	Framework for BI in healthcare . . . . .	11
4.1	Metabase dashboard example. . . . .	31
4.2	Screenshot of Metabase methodology for dashboard embedding. . . . .	32
4.3	Metabase database search engine. . . . .	33
4.4	Screenshot of the developed export button. . . . .	34
4.5	Screenshot of the developed Pivot table features. . . . .	35
4.6	Pentaho platform simplified overview and content action flow. . . . .	38
4.7	Proposed solution architecture. . . . .	40
4.8	<i>BI-commons-plugin</i> instantiation example. . . . .	43
5.1	Pentaho Report Designer architecture overview. . . . .	46
5.2	Screenshot of a blank report in Pentaho Report Designer. . . . .	48
5.3	Screenshot of the ‘Label’ element properties. . . . .	49
5.4	Screenshot of the Edit Parameter window in Report Designer. . . . .	51
5.5	Query code example with parameter injection. . . . .	52
5.6	Screenshot of PRD when a parameterizable report is requested. . . . .	53
5.7	Screenshot of the Query Designer Editor in Report Designer. . . . .	54
5.8	Screenshot of a pivot table example in Report Designer. . . . .	55
5.9	Pentaho Server HTTP request flow. . . . .	56
5.10	Pentaho Server operations flow for a report request. . . . .	57
5.11	Pentaho Home page. . . . .	58

5.12	Pentaho integration overview. . . . .	61
6.1	Diagram representing part of MDTeam database schema. . . . .	67
6.2	Screenshot of the MDTeam report first half. . . . .	68
6.3	Diagram representing part of CardioBox database schema. . . . .	69
6.4	Screenshot of CardioBox with the embedded solution. . . . .	70
6.5	Diagram representing part of PACScenter database schema. . . . .	71
6.6	Screenshot of PACScenter with embedded solution. . . . .	72



# List of Tables

2.1	Differences and Similarities of healthcare and other sectors. . . . .	9
2.2	Open source license characteristics. . . . .	14
3.1	Business Intelligence platform features. (June 2018) . . . . .	23
5.1	Description of the main report elements in Pentaho Report Designer. . . . .	50
5.2	Description of the main API endpoint in the File Management category. . . . .	60



# Chapter 1

## Introduction

The healthcare sector collects daily an enormous amount of data, both clinical and administrative. With this increase of healthcare data, our knowledge should be greater and our practice should be more effective. However, the quantity of data is surpassing the capacity of its use to leverage better results in the quality of care [1].

Making effective and accurate decisions with data supporting evidence is crucial in any sector, and healthcare is no different. Through quality decisions that are made, the care provided to patients has the potential to become safer and more effective, resulting in better patient's health outcomes [2]. Yet, making decisions in healthcare is often complicated by several factors such as potentially conflicting data, the heightened sense of privacy and security, and the fact that medicine is not an exact science [3].

Healthcare providers are faced with the challenge of going through huge amounts of data in search of answers to complex questions. It is necessary to deliver tools to healthcare providers that allow them to swiftly move around data from different sources in an easy manner. It is believed that Business Intelligence (BI) solutions can help with this problem, facilitating the translation of data into effective decision-making [1, 4].

Through its end-user orientation, Business Intelligence tools provide solutions that allows the user to easily have access to data visualizations, like reports, charts and dashboards, which can, accurately and meaningfully, be decoded.

As BI is becoming increasingly relevant for the healthcare sector [5], the basis of this work will be a revision on existing BI solutions, followed by an evaluation on the hypothesis if they can be integrated into existing medical web applications.

## 1.1 Motivating

Business intelligence is a constantly evolving area that attempts to find the easiest and most flexible solutions to present information for the end-user. A topic that has recently been discussed is embedded BI and analytics. This topic reflects the idea of integrating parts of a BI engine into already developed solutions. With this approach, it is no longer necessary for organizations to acquire specific, stand-alone, software to perform reports and dashboards, since these functionalities are already embedded in the applications used by the organizations.

There are two ways of integrating embedded BI software into the company solutions. The first is to develop an internal BI solution from scratch, and later embed it to the remaining company solutions. The second approach is to acquire an existing BI solution, either free or commercial, and later embed it. Typically, software developing companies don't have the expertise, time or money to build a Business Intelligence solution from scratch, and so, the present work aims to study the feasibility of the second approach.

In the healthcare area there is the constant necessity of providing physicians with patient's reports and state-of-art analytics. Using Business Intelligence solutions is a common approach, though they are used as a stand-alone solution. There is the urgent necessity of verifying if integrating BI solutions into medical software is plausible, in order to eliminate the necessity of multiple platforms and to provide physicians with a better and faster workflow.

## 1.2 Objectives

The main objective of this work is the study of existing Business Intelligence solutions and the subsequent development of an embedded solution intended for a medical context.

In the solutions study phase, the main free and open-source solutions will be analyzed. Particular attention will also be given to the licenses to which these solutions are subjected, since it may be necessary to do intrinsic or extrinsic changes to the solution.

After analyzing which platform can best fulfill the requirements, an important goal is to scrutinize how this solution can be integrated into the already developed software.

The ideal solution will enable end-users of existing software to have access to reports and dashboards generated by a BI engine, yet transparent to them. The goal is to create a smooth flow of events such that when the user requests certain content, he doesn't need to be aware

that they were actually generated by a BI engine.

This solution must be developed in such a way that allows easy integration with existing solutions, facilitating developer's work. Besides that, it must have a strong usability component that allows full customization on the buttons and labels styles, so that the embedded part can be smoothly integrated in the existing layouts.

### 1.3 Document organization

The following chapters of this thesis are organized according to the following structure:

**Chapter 2** presents some concepts that are necessary to be clarified for a better understanding of the work that follows. Concepts such as Business Intelligence, Data Visualization, Medical Data and Software Licenses will be addressed.

**Chapter 3** presents an analysis of the state-of-art of existing Business Intelligence solutions. The most relevant open-source solutions will be reviewed in an attempt to gain a better understanding of the possibilities and limits of each one. An empirical evaluation of the solutions will later be made.

**Chapter 4** starts with a requirements gathering. At this stage, functional and non-functional requirements that the final platform would have to contemplate will be pragmatically addressed. Then, the development process is described. A first approach will be carried out with one of the platforms, discussing its limitations. Then, a second approach with another platform is presented, in which it was possible to develop a system that fulfilled all the requirements.

**Chapter 5** describes the necessary methodology to run an instance of the solution. It also provides a walkthrough, from the development of content, to its embedded presentation.

**Chapter 6** presents three application scenarios where the solution was deployed. A reflection will be made on the contributions that the solution offers, as well as its limits and obstacles.

**Chapter 7** presents a conclusive reflection on this thesis and highlights directions for future work.



# Chapter 2

## Concepts

In this chapter, important key concepts for the present work are described. Throughout this work, they are constantly referred, and so, it is important to have a clear definition of their meaning and what constitutes them.

### 2.1 Data Visualization

Data visualization refers to the graphical presentation of information, in order to help the viewer gain insight of data. It is about understanding the relationships among numbers as patterns, correlations and trends that can go undetected in groups of numbers [6].

The process of visualization relies on the end-user cognitive capacities and interactive techniques to detect, measure and compare data, in order to acquire new insight. Data visualization is only as good as our eyes can discern and our brain can grasp. The fundamental objective is to translate data into visual representations that can be easily, efficiently, accurately and meaningfully decoded [7].

As organizations continue to make data-driven decisions, the demand to understand and see through the collected data is quickly growing. Treating data as a real asset can become a competitive differentiator. In fact, recent studies have proven that companies who are mostly data-driven had a higher productivity and higher profits than the average [8]. Before the “information age”, data was scarce and sometimes neglected. There was no interest in collecting huge amounts of information and with such detail. Organization’s decisions had a high level of intuition on it. Nowadays, with the technological advance and the competitiveness

of the market, organizations are compelled to make the best out of their data.

Thus, there is a need to offer technological solutions that help organizations easing the process of data visualization and optimize business. Business Intelligence (BI) tools surge to solve this problem. With its end-user orientation, BI platforms have the great advantage of not demanding programming skills to create rich, interactive visualizations. Popular data visualizations tools, while very powerful, require programming skills that organizations do not have or can't afford to have. Normally, healthcare providers don't have database knowledge, so it is necessary to provide them with tools that simplifies the creation of charts, dashboards, heat maps and performance metrics without having to interact or even see query languages or programming code.

## 2.2 Business Intelligence

Business intelligence (BI) is a collection of methods and technologies for collecting, storing, accessing and analyzing data from multiple sources, offering organizations managers the right tools to make better decisions [9]. It is a heterogeneity of tools that allow decision-making from data that, at first sight, has no meaning. Figure 2.1 represents the decision-making life cycle.

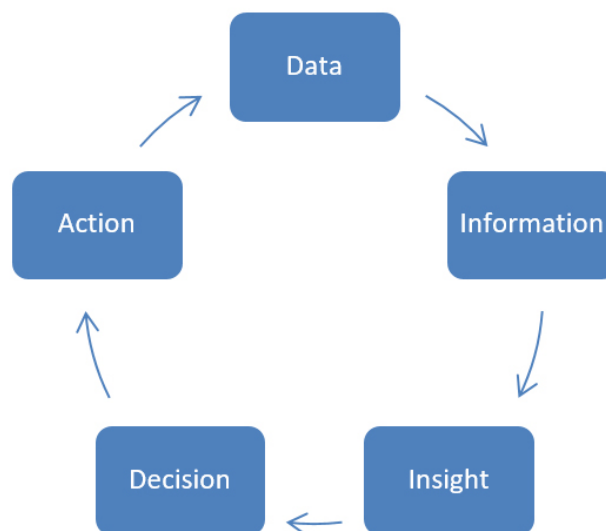


Figure 2.1: Decision-making life cycle



Data is the feedstock of any BI eco-system. BI tools offers the possibility to cross data and act on it swiftly by providing the right tools to do so. From there, is up to the user to interpret and decipher the data. The insight that one might gain can later be translated into highly informed decisions, completing the BI purpose.

Business Intelligence platforms can have many different elements, where each one is targeted to perform a different task. Figure 2.2 shows an architecture that comprises most of these elements.

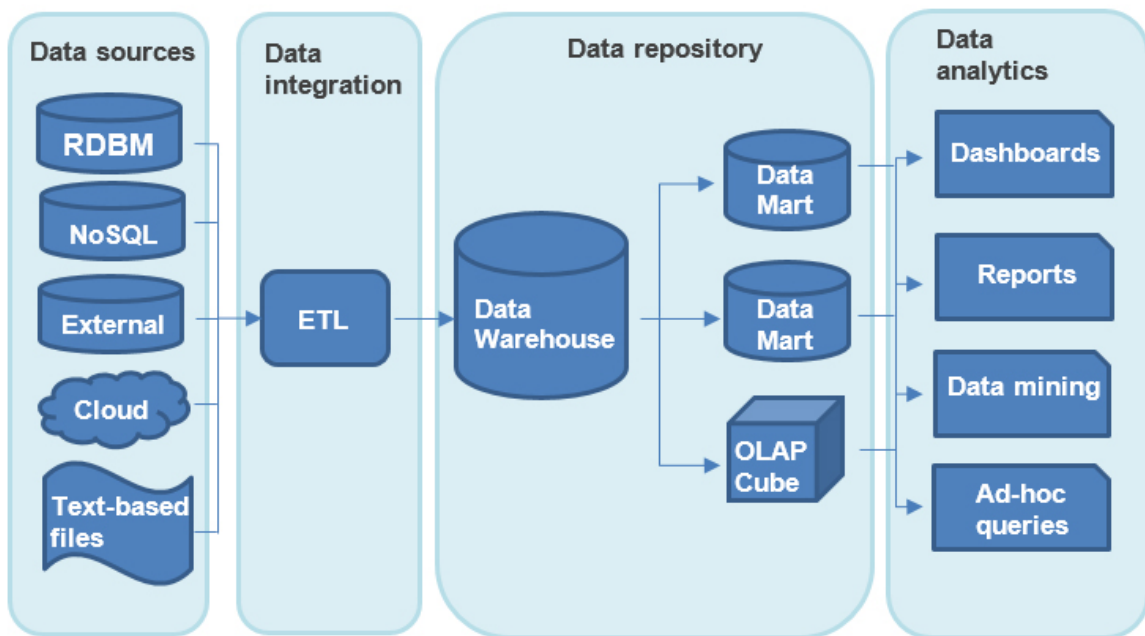


Figure 2.2: Business Intelligence elements overview.

First, organization’s data might be spread in different formats, such as rational/non-relational databases or in text-based formats like XML/CSV. Data must then be extracted from different sources, transformed and uniformed into one clean format, and later be loaded into a new database. This operation is called Extract, Transform and Load (ETL). A database with the correct and clean data is also known as Data Warehouse (DW). A DW can later be divided into smaller databases called Data Marts (DM).

Next, database queries and Online Analytical Processing Cubes (OLAP Cube) are essential to build data visualizations. OLAP is computer processing that enables users to swiftly move around data, through fast, consistent and interactive access. It is characterized by being multi-dimensional. Whereas a relational database can be considered a two-dimensional, in

OLAP each data attribute (column) is a separated dimension. This characteristic supports end-user analytical and navigational activities.

Last, business performance management (BPM) using dashboards, scorecards or key performance indicators (KPI) helps the end-user visualize and analyze multiple performance metrics.

Some BI platforms go further and, in addition to these characteristics just mentioned, they implement statistical and data mining features. It allows several more functionalities like data segmentation, classification, anomaly detection and predictive modeling. Leading enterprise BI software, like SAP and IBM, have already integrated the majority of these features.

## 2.3 Business Intelligence in Healthcare

The health community is experiencing an exponential data growth. Whether it is from health or health-care related, information is being generated from multiples sources such as sophisticated medical instruments, Internet of Things (IoT) in healthcare, patient care points of contact – hospitals, clinics and pharmacies - and web-based health communities [10].

Making informed and successful decisions is crucial in any sector, but particularly important in the medical area as they can make a significant difference in people’s lives. It is believed that, with the use of BI, healthcare organizations can convert huge amounts of data into information that can improve treatments outcome, increase safety and improve overall efficiency in clinical processes [1]. Healthcare decision-making can be extremely dynamic and complex [11]. Through consistent quality decisions, the health system has the potential to positively and progressively improve patient’s health outcomes [12].

However, healthcare sector shows underdeveloped information system structure and the accumulation of information generated is surpassing the capacity of its use to leverage better results in the quality of care. BI solutions can help with this problem, converting data into knowledge that improves patient treatment and operational efficiency [1, 5].

The existing literature in BI has largely been done from retail, manufacturing and finance point-of-view [5]. When organizations choose to adopt BI solutions they need to understand the main characteristics of their business model are and how they can influence the choice of solutions. When addressing the problem from a healthcare position, it is necessary to take into account the major differences that it can bring due to the complexity of health systems.

Table 2.1 [1, 4, 13, 14] highlights the main characteristics when comparing healthcare BI infrastructures with other sectors.

Table 2.1: Differences and Similarities of healthcare and other sectors.

Healthcare and other sectors	
Differences	Similarities
Management is unified in most sectors, but in healthcare, there are two types of reporting: Clinical and Operational.	Healthcare is becoming patient-centered, following product-centered and customer-centered success in other sectors.
Actors are a small set in most areas, but healthcare there are a huge set of actors with different permissions (e.g. doctors, patients, nurses, insurance companies, governmental authorities...).	Although healthcare systems are typically larger, more complex and employ more people than other systems, they still benefit from whole-system analysis.
Most industrial systems have hard metrics. In healthcare, people's feelings and choices matter.	All sectors seek improvements in cost, quality and delay through integrated processes.
Healthcare decision-making is often complicated by the need to integrate ill-structured, uncertain and potentially conflicting data from different sources.	The output of information should be presented in the format that gives a quick and self-elaborative snapshot to decision makers in graphic representation.
Healthcare is both an art and a science; not every patient will react the same way to a treatment.	A well-built data warehouse is essential, as it is the center of knowledge creation.
Healthcare data carries a heightened sense of privacy and security issues.	Decisions may depend on the function of the task and the expertise of the decision maker.

## 2.4 Data in Healthcare

Developing a healthcare Data Warehouse is a complex and time-consuming process, but it is crucial for delivering quality health services. Extracting knowledge from healthcare services requires the integration of different healthcare data sources [15]. The data generated by a health organization can be classified into 3 different types: Clinical, Administrative and External [5].

- **Clinical data** refers to data generated in the act of treating the patient. It can be in the form of electronic health records (EHR), laboratory results or exams. It is important that all treatment applied to the patient is documented by all the actors involved, so that data is as complete as possible.
- **Administrative data** sources contain all the business data generated in a health-care organization. It includes information like human resources (HR) data, operations scheduling and financial data. Administrative data is needed to measure, assess, control, and improve the quality and productivity of operations at the organizational level.
- **External data** can be either clinical or administrative, but it is provided by an external entity. It can be data from other departments, from insurance companies or medical reports.

BI has gained a lot of interest amongst healthcare providers especially due to its potential use in electronic health records (EHR) [16]. An electronic health record is an official repository of information that keeps track of all health status and treatments that a patient has been submitted to. It is stored and transmitted securely between authorized users [17]. This definition comprises three fundamental characteristics: it acts as a repository of medical information; the content is private and it can be shared between authorized entities. EHR contains information such as patient's demographics, medical history, current and past medications, radiology reports, laboratory data and several others. It is used primarily for planning patient care, documenting the delivery of care and assessing the outcomes [18]. The EHRs have created massive amounts of clinical information and they are loaded with valuable data that can be studied for decision-making purposes. Granting healthcare providers tools that they can use to effectively detect patterns, trends and cause-and-effect relationships among patients, can make medical decision-making more pragmatic and less subjective, translating into better and safer outcomes. This philosophy meets the fundamentals of EHR records: Providing patients with a better service.

Synthesizing what has been discussed in this chapter, Figure 2.3 comprises various elements that make up a possible Business Intelligence solution in a healthcare environment.

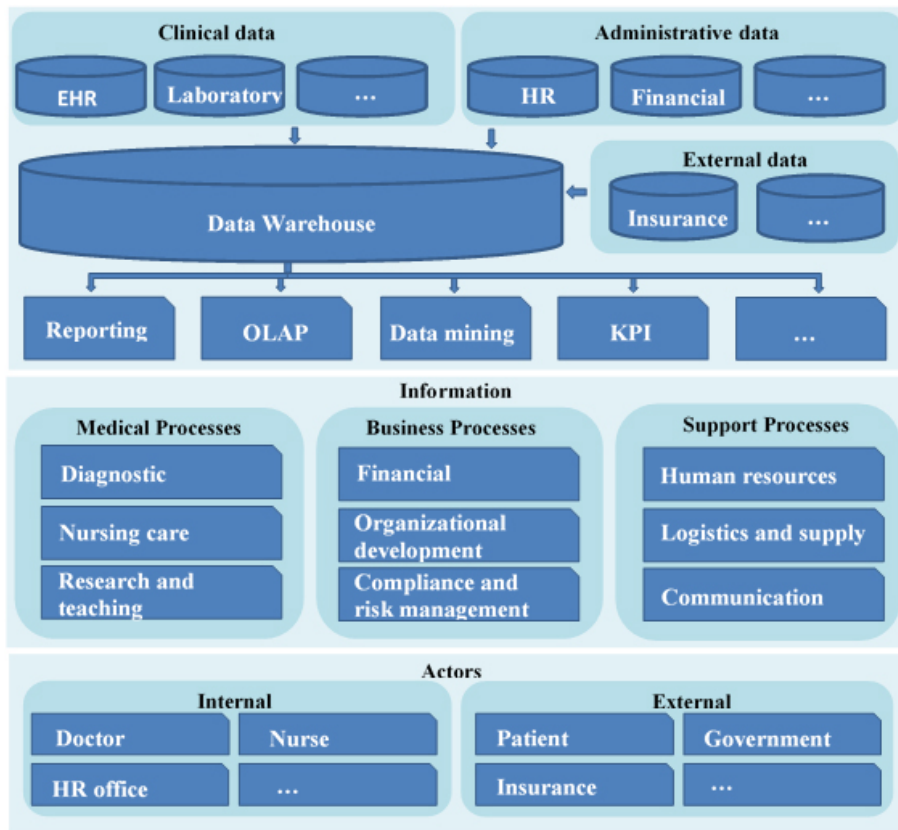


Figure 2.3: Framework for BI in healthcare (Adapted from [5])

## 2.5 Open Source Software

Open Source Software (OSS) is growing fast; what was once the domain of hobbyists and hackers, has gained acceptance with end-users, corporations and governments. Linux and Android operating systems are the most known and used exemplars of open source, operating in millions of devices [19].

Open source software implies that the source code is made available, so that anyone can inspect, modify, enhance and share. It is important to highlight that the term does not necessarily mean unpaid software. Open source software is made by many people, and distributed under licenses. Those licenses must comply with the Open Source Definition (OSD), which means following ten ground rules [20]:

- **Free redistribution** – The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing

programs from several different sources. It shall not require a royalty or other fee.

- **Source Code** – The program must include the source code. Deliberately obfuscated source code is not allowed.
- **Derived works** – The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.
- **Integrity of the author’s source code** – The license may restrict source-code from being distributed in modified form only if the license allows the distribution of “patch files” with the source code for the purpose of modifying the program at build time.
- **No discrimination against persons or groups** – The license must not discriminate against any person or group of persons.
- **No discrimination against fields of endeavor** – The license must not restrict anyone from making use of the program in a specific field of endeavour. For example, it may not restrict the program from being used in a business, or from being used for genetic research.
- **Distribution of license** – The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.
- **License must not be specific to a product** – The rights attached to the program must not depend on the program’s being part of a particular software distribution.
- **License must not restrict other software** – The license must not place restrictions on other software that is distributed along with the licensed software.
- **License must be technology-neutral** – No provision of the license may be predicated on any individual technology or style of interface.

The ongoing growth of the OSS idealism has opened a window of opportunity for organizations. The use of OSS made it possible to start from already developed software and develop new features on top of them. This contributes to a huge cost reduction, but it is not all. The independence of suppliers and the transparency of the code are major advantages. The main advantages are as follows [21]:

- The low or zero cost of licenses.
- Frequent updates.
- Exchange of experiences, questions and information in the community of the respective product.
- Easy to split the application into modules, using only the necessary ones.
- Integration with other tools, interconnecting available components through the reuse of sources.

The main disadvantages are:

- The software is still far from being user-friendly in general, and highly dependent on technical experts to program new features.
- It is difficult to find the source of errors and correcting them, as the code is not familiar. Sometimes it lacks documentation.
- The absence of a stable version. The entire community can add value, leading to the existence of a set of modules that evolve independently.
- There are no official contracts.

## 2.6 Licensing

There are over 50 different types of licenses that Open Source Initiative has certified as being open source. The most important characteristic is what kind of restriction the license puts upon derivate works [22]. Four types of licenses will be briefly addressed as they will be important throughout this work. They are GPL, AGPL, MPL and Apache. Table 2.2 summarizes the main characteristics of each one.

GNU General Public License (GPL) is the most used license. With a strong copyleft position, it requires that any modification to the source code must be notified and shared. Also, any derivate work must be released under the same GPL license. This prevents any competitor from building a proprietary and improved version from someone else's work, as they are compelled to share and stage changes [22].

Table 2.2: Open source license characteristics.

		AGPL-3.0	GPL-2.0	MPL-2.0	Apache-2.0
Permissions	Commercial	✓	✓	✓	✓
	Distribution	✓	✓	✓	✓
	Modification	✓	✓	✓	✓
	Patent use	✓		✓	✓
	Private use	✓	✓	✓	✓
Conditions	Disclose source	✓	✓	✓	
	License and copyright notice	✓	✓	✓	✓
	Network use is distribution	✓			
	Same license	✓	✓	✓	✓
	Stage changes	✓	✓		✓
Limits	Liability	✓	✓	✓	✓
	Trade mark use			✓	✓
	Warranty	✓	✓	✓	✓

GNU Affero General Public License (AGPL) is a modified version of the ordinary GPL, only with one additional clause. When a modified version is used to provide a service over the internet, the complete source code of the modified version must be made available. This closes a loophole on the GPL license over the internet. When a software is running on a local and private server, providing services to the internet, under a GPL license, providers don't have to share the code because they didn't convey the software. With AGPL license, sharing services online from derivate work counts as conveying, and modified versions must be made available [23].

Mozilla Public License (MPL) is a license that combines MPL software with proprietary code. Users can develop their own private code alongside with the original source code, as long as they keep MPL code in separated files.

Apache license is the most business-friendly. This permissive license doesn't require derivate work to be distributed under the same license. Also, the source code from derivate works doesn't have to be disclosed [24].



## 2.7 Conclusion

The concept of Business Intelligence suggests a complex set of methods and technologies that allows working on raw data up to the formation of visualizations. The present work focuses precisely on the last layer of Business Intelligence, the analytical layer.

Business Intelligence is usually associated with retail and financial organizations, where the goal is to maximize profits through decision-making based on the results produced by analytical processes. This work aims to understand if the concept of Business Intelligence can be applied in the area of health, more precisely, to clinical data. Therefore, in this chapter, the main differences between the health sector and other sectors were addressed. A brief description of the type of data that exists in the health area was also described, as well as how BI platforms can translate this data into discernment for the healthcare provider.

As one of the main objectives is to find an open-source solution that satisfies the requirements, the open-source concept was explored. It was possible to analyze some rules that a project has to obey in order to be considered open-source, as well as the advantages and disadvantages that follow.

Licensing comes hand in hand with open-source, as it is the only legal way to condition the use of this type of software. It was possible to study several types of licenses where what changes is the permissiveness of each one. In the next chapter, a state-of-art revision will be carried out on existing BI solutions. These solutions may have one or more associated licenses. By studying the permissiveness of each license, it is now undoubtedly easier to have a more judgmental character, as it may be necessary to make changes to existing platforms.



## Chapter 3

# State-of-Art

After defining some key concepts and framing the problem, in this chapter some existing BI platforms that were tested and evaluated will be described. First, Business Intelligence solutions that are more complete and mature will be presented. They are known as BI suites. Then, some BI solutions that have fewer features when compared to the suites will also be presented, as they were considered relevant for the purpose of this work. These solutions are more targeted to data visualization. Later in this chapter, an empirical evaluation of the solutions will be carried out, based on several factors.

The platforms that are about to be presented were chosen following important criteria. Any paid solutions was kept out of scope, as they are typically very expensive and targeted for bigger financial organizations. Free platforms were primarily focused, and, conveniently, most free solutions are open source. Some platforms, although initial free, were very limited in functionality and extra features were behind a paywall. They were also kept out of scope. The platform's popularity, their reviews and their online community were also some key criteria for consideration.

### 3.1 Business Intelligence Suites

The term "suites" means that the platform is composed of a set of different modules that are delivered in a single package. As Business Intelligence is formed by a set of technologies that can get very detailed and complex, vendors opt to divide what would be a very large and confusing platform into multiple smaller modules. Each module is focused solely on a BI

process, like OLAP or Data mining. For instance, Pentaho, a BI platform, is divided into four modules. One of them is named Kettle, which is completely oriented towards ETL process. Each of the remaining modules is targeted to other BI processes. This methodology allows a better pragmatism in the development flow, and it allows more complete and focused tools.

The Business Intelligence suits that are about to be analyzed are Pentaho, Jaspersoft and SpagoBI.

### **3.1.1 Pentaho**

Founded in 2004, Pentaho is the most used BI platform in the market. It was recently bought by Hitachi Data Systems Corporation on June 4, 2015. The current version of Pentaho CE is 8.0 and it works under a GPL-2.0 Licence.

This platform is distributed in two editions: The Enterprise Edition (EE) and the Community Edition (CE). The Community Edition is free, open source, and only contains part of the functionalities, while the Enterprise Edition is a commercial and a paid product that includes all the Community edition features, and, in addition, has an enhanced user interface. All the platform is written in Java.

The Community Edition has the following modules: Pentaho Data Integration, Pentaho Reporting, Pentaho Analysis Services and Pentaho Data Mining. They are local desktop engines that work separately and the results produced are sent to a server that acts as a repository, known as Pentaho Server.

Pentaho Data Integration, also known as Kettle, is the engine responsible for ETL. It allows the creation of data integration streams, from multiple sources to a data warehouse. It contains multiple features for data migration, data cleaning and rules applications.

Pentaho Analysis Services is based on Mondrian, an OLAP server with a Relational OLAP (ROLAP) architecture. It allows multidimensional analysis and common OLAP operations such as slice and dice, roll-up, drill-down and pivot. It uses multidimensional expressions (MDX) query language. Pentaho offers a program called Schema Workbench to visually create and test OLAP cube schemas. Once the cube is created, the user can upload it to the server.

Pentaho Reporting is a reporting tool that allows the creation of data visualization in the form of graphs, tables or reports. It supports standard data sources like SQL, MDX and also

Kettle transformations on the fly. The results can later be exported to multiple formats, like CSV, XML, Excel or PDF, or uploaded to the Pentaho Server.

Pentaho Data Mining, known as Weka, is a collection of machine learning algorithms and evaluation methods for data mining tasks.

The Pentaho Server is a web-based interface that hosts the content created by the described modules. It has several plugins and more can be found in the Market, a community plugin repository. It allows creation and management of users and the creation of dashboards and graphs, although not as complete as Pentaho Reporting.

### **3.1.2 Jaspersoft**

Jaspersoft BI platform was created in 2006. Like Pentaho, it is distributed in two editions: A Community Edition - which is free, open-source and works under an AGPL license - and a paid Commercial Edition available in three different editions: Reporting, AWS, Professional and Enterprise.

It follows the same architecture as Pentaho, meaning that it is composed of several modules, where each one is focused on one task: Jaspersoft ETL, Jaspersoft OLAP, JasperReports Library, Jaspersoft Studio, iReport and JasperReports Server.

Jaspersoft ETL is a data integration platform for extract, load and transform data. It has an overall pleasant and detailed GUI environment.

JasperReports Library is an open source reporting engine, written in Java. It produces any kind of visualization - dashboards, tables, charts and reports - from multiple sources. This engine is at the core of any other Jaspersoft module that needs to generate data visualization.

Jaspersoft Studio and iReport are similar, but Jaspersoft Studio is an Eclipse-based report designer, while iReport is NetBeans-based. Both use the library just described to render visualization and the results can be sent to the JasperReports Server. These tools are somewhat limited in the Community Edition when comparing to the Commercial Edition. Still, it is possible to use them to produce simple visualization and they have the ability to export the results in a wide range of formats.

Jaspersoft OLAP is an OLAP analyses platform. It runs within JasperReports Server and it uses Mondrian as a server. JasperReports Server gives the user the tools to interact with the cube and create visualizations, including an enhanced user interface, streamlined toolbars

and icons, expand/collapse features, and consistent option panes.

JasperReports Server is a web application that acts as a central information hub for the organizations. It is optimized to share, secure and centrally manage the reports and analytic views generated by the other modules.

### 3.1.3 SpagoBI

Developed by SpagoWorld, an organization founded in 2006, SpagoBI is a full Business Intelligence platform, completely free and open source. According to their website, they claim to be a consulting company, thus having no interesting in commercializing the solution.

Unlike the solutions that have been seen so far, where there are two versions of the product (Community and Enterprise), SpagoBI only has one complete and free version, developed in Java. This platform works under the terms of the Mozilla Public Licence v2.

This platform is seen as an integration platform rather than product platform because it is not built around a predefined set of tools. It has a modular structure in which all modules are related to the core system, ensuring the harmony of the platform. For instance, if the user wants to develop a new report, there are multiple engines he can choose from Jasper Report Engine (from Jaspersoft), BIRT Report Engine and BO Engine. The engines can be installed depending on the user needs.

Similar to the other platforms, SpagoBI is also divided into modules. There are 4 main modules that constitute the platform: Meta, Studio, SDK and Server.

The Meta module is responsible for all the data integration and metadata management. There are three different engines from which the user can choose from to perform data migrations.

The Studio module is an Eclipse-based environment that allows the developer to design and modify all analytical documents, such as reports, OLAP, dashboards and data mining. All the produced documents can be later uploaded to the Server module, through the SDK module.

The SDK module is the specific tool used for the integration of the services provided by the Server. In particular, it is used by SpagoBI Studio in order to allow users to download or upload their analytical documents from or onto the server, through web services. It also offers web services for the communication between the Server and the Meta module.

The Server module is a Java web application that offers a unified access point, following the same characteristics as the other platforms. It has a very detailed user modeling, allowing a fully customizable roles management.

## 3.2 Business Intelligence Tools

The three platforms that have been discussed – Pentaho, Jaspersoft and SpagoBI – integrate most of the BI cycle functionalities, from ETL, through OLAP, to complex analytical document building. The solutions that are about to be described are more simple and lightweight. They do not offer the complete vertical stack of features needed for BI, as the previous ones. In fact, they only provide database connectors and the tools to build analytical documents. Nevertheless, these tools are fully web-based, with modern UI and very user-friendly. They don't demand a laborious workflow to build dashboards or other visualizations.

### 3.2.1 Metabase

Metabase is a free, open source analytical tool founded in 2014. It is developed in Clojure, distributed under an AGPL-3.0 license.

The main idea around Metabase is the ability to ask questions about data without knowing SQL, through a very simple web interface. This work is done by a translator that converts the user's research intention into SQL code. The results are retrieved in table format, where the user can proceed to re-filter the result or translate the table into several different graphs. The result of one single graph is called "Question". Later, it is possible to build dashboards by dragging and resizing multiple "Questions" into a mesh. It has a simple approach to building geographic maps.

The results, individual questions or complete dashboards, can later be sent by email or Slack and it is also possible to share the results by embedding them into any web page.

Metabase allows fully customizable roles management and has a unique and modern feed wall, where the activities of work colleagues can be visualized.

### 3.2.2 Saiku

Saiku is an OLAP analytical engine. Created in 2010, it is an open source software that works under the Apache v2 License .

Saiku is distributed in 2 editions: Saiku Enterprise edition – a paid version as a standalone server that offers all the functionalities - and Saiku Community edition – a free, community version that offers significantly fewer features. Saiku Community edition doesn't have crucial features such as dashboard designing or advanced data filters.

With the ability to connect to multiple OLAP servers, including Mondrian, Saiku offers a user-friendly web interface that allows the users to perform analytical research, creating and sharing reports. With drag-and-drop features to interact with measures and dimensions, the users don't need to understand MDX or related OLAP query languages.

After defining the measures and dimensions, the results are returned in table format where the user can perform OLAP operations – such as drill down, filter or pivot – or translate the results into several different types of charts. The results can then be saved, shared or exported to Excel or PDF.

Saiku Server provides a REST API with JSON data payload, meaning that it is possible to interact with the engine without the Saiku UI.

## 3.3 Tools Comparison

All the mentioned platforms were installed and explored to get a feel of their capacities and usability, keeping in mind that the tools are intended for healthcare providers. While the platforms were being experimented, an empirical and analytical evaluation of the main functionalities and characteristics was made. This evaluation is presented in Table 3.1.

This table should not be considered as a way to evaluate which platform is better, but rather an informative way to quickly recognize the main characteristics of each platform. In fact, it wouldn't make sense to compare such different tools, as they can be used to serve different purposes.

The first impression when analyzing BI suites was that the whole visualization creation process required a laborious effort. The interfaces in which they are based, mostly NetBeans or Eclipse, are complex and requires an extensive know-how, even for developing simple graphs or



Table 3.1: Business Intelligence platform features. (June 2018)

	Suites			Data Visualization	
Functionalities/Criteria	Pentaho	Jaspersoft	SpagoBI	Metabase	Saiku
Functionalities					
Reporting	✓	✓	✓	✓	✓
Dashboards	✓	✓	✓	✓	
Microsoft Office Integration	✓	✓	✓	✓	✓
OLAP	✓	✓	✓		✓
Interactive Visualization	✓	✓	✓	✓	
Data Mining	✓		✓		
Collaboration	✓		✓	✓	
KPI	✓		✓	✓	
Geo-referencing	✓	✓	✓	✓	
ETL	✓	✓	✓		
Software					
Version type	Community	Community	Complete	Complete	Community
Version	8.0	6.0	6.1	0.26.2	3.7
Language	Java	Java	Java	Clojure	Java
License	GPL-2.0	AGPL-3.0	MPL-2.0	AGPL-3.0	Apache-2.0
Code					
# of commits	7 625	8 240	10 841	13 205	5 508
# of minor releases last year	12	9	3	19	6
# of bugs opened last year	517	143	110	217	?
# of bugs solved last year	402	119	37	355	?
Evaluation					
Easiness to Acquire	5	3	5	5	5
Easiness to Install	4	4	3	5	3
Easiness to Use	3	2	2	4	4
UI appearance	3	2	3	5	4
Processing Speed	4	3	3	3	3
User Documentation	4	4	4	4	2
Developer Documentation	4	3	3	4	3
Forum	4	2	3	3	2

reports. Furthermore, technical configurations, like setting up a database connection, can get complicated and it has to be done in every module. None of the suites have mechanisms that facilitate the search of information, it is almost mandatory to have some SQL knowledge. It is perceivable that in order to have so many detailed functionalities, it is necessary to abdicate from simplicity. The overall opinion is that these BI suites have a very steep learning curve and they are targeted for more technical users.

As for Metabase, its simplicity was captivating. It was clearly developed to enable the average user with the ability to deep dive into data, without even see SQL code. Translating table search results into graphs or other data visualizations, and configure them, is also very simple. Their API is well documented as it explains all the endpoints with a small description. As for the user documentation, they demonstrate some use cases to explain the visualization process. The fact that it is written in Clojure can be a disadvantage as it is not a popular programming language. Moreover, it is not a mature platform, and some features continue to be developed.

Saiku enables the users to swiftly move across data from an OLAP cube, with all the operations made easy. The whole interface is very easy to understand and, with familiar drag and drop features, the user doesn't need to understand MDX or related OLAP query languages. The user documentation, updated for every version, is appropriated as it details most use cases, although their API documentation is not very detailed, as most information is missing. The Community edition is very limited, for instance, it is not possible to build dashboards from multiple graphs. Saiku entire stack - server and UI - was recently switched to an Apache-2.0 license, as they believe it will help them grow as a product and as a community.

### 3.4 Conclusion

None of the existing platforms fully satisfied our initial goal. Either the solution of creating visualization was too meticulous and sometimes complicated, or it is was too simple, compromising important features.

Assigning any of the BI suites directly to healthcare providers would be unreasonable, as they are very complex and have a very steep learning curve. On top of that, they don't have mechanisms that facilitate the search on databases for users that don't have SQL know-how. In order to consider BI platforms as possible solutions, the processes of building a graph or a

report would have to be assigned to technicians and not to end-users, in this case, healthcare providers.

Metabase is a good example of what the intended platforms should look like, but not entirely. It has a good mechanism of translating users' search intentions into SQL code that allows the graph generation, but an ideal platform would also have the possibility to generate reports in a paper sheet style and the possibility of configuring graphs, as Metabase graph axes cannot be modified.

Saiku alone would not satisfy this work requirements. The community edition is very limited and most functionalities are left off. Even though it is a powerful OLAP searcher, using Saiku and building a solution on top of it would be unreasonable, as it would require a huge effort to take advantage of only one functionality.

In order to make a conscious and weighted decision on the path one should proceed, the requirements that the final solution should contemplate are going to be pragmatically established at the beginning of the next chapter. This will help to define a shape of the intended platform. Then, the analyzed solutions will be confronted with the established requirements and the solution that best fits will be chosen. If some requirements are missing, it is necessary to analyze in each way they could be improved and build a solution on top of it.



## Chapter 4

# Solution Analysis

This chapter starts by describing the most important requirements that the chosen platform should contemplate. By establishing functional and non-functional requirements, it becomes possible to better screen the solutions considered and the choice of the platform to work will be more well-founded. It also provides a more clear idea of what the intended platform should be.

Next, the different approaches that were made to find the best solution to the given problem will be described. As it will be possible to read next, the first approach was not successful, and therefore a second attempt was necessary. As the present chapter focuses on the methodology and pragmatism of an exploratory journey of finding a solution, not the solution itself, the whole reasoning of both the failed and the successful solution will be described.

At each approach, the reasons for choosing the solution are detailed and a brief analyses of the architecture behind is made. Next, it is scrutinized in which way the chosen platform may serve better, and if it is necessary to modify the source code. If any modification was made, they are thoroughly detailed, as well as the work that the associated license forces. Finally, each approach is concluded with an evaluation of the work done and the problems that the solution solves and raises.

## 4.1 Requirements

A functional requirement is a brief description that specifies a function that a system should be able to perform, regarding technical functionality. A non-functional requirement places constraints on *how* the system will do so, by specifying criteria that can be used to judge the operations of a system.

In the following sub-chapters, some requirements will be enumerated and a brief description of its importance to the platform will be given.

### 4.1.1 Functional Requirements

#### **Embedding**

The embedding feature is a crucial functional requirement of this work. After developing a report or a dashboard, the content created should be easily integrated into an already developed web application. The content, provided by the BI platform, should be directly placed within the application user interface and, if possible, display controls that allow the final user to manipulate the content.

#### **Report and Dashboard Building**

The final solution should have the tools to build highly refined content such as a complete dashboard with different types of graph, or an A4 report style, that provide insightful data views. It also should have the possibility to fetch data from multiple data sources. If possible, the process of creating content should be as easy and simple as possible, but without compromising functionality. Preferably, it should allow end-users to develop their own reports and dashboards. If this is not possible, then the content building phase will be delivered to a technician who will construct the content in such a way that it is possible for the end-users to change the information shown through filters.

#### **Report and Dashboard Filters**

Filters, or parameters, are variables that the content builder can set up. Then, the end-user can navigate through the data within the limits established by the builder on those variables. These variables, after being defined, are injected into the databases queries and

manipulate the information presented on the dashboard or report. For instance, a filter could be the ID of a given patient, that after being inserted by the end-user, database queries only fetch data to the given patient ID. The main objective is to give the end-user a prompted user interface with all the possible parameters that can be modified.

## **User Management**

It is important that the solution has the mechanism to support multiple users, each one with individual authorization access. As previously analyzed, medical data is a very sensitive topic. Adding roles management that constrains the visualization of content between users adds a layer of privacy and security to the platform. Preferably, the users that are in the database of the BI engine will be the same as the application where it is embedded.

### **4.1.2 Non-Functional Requirements**

#### **Performance**

Performance is a key non-functional requirement to the platform. The server that is behind the embedded content should provide a fast answer, even when handling large database queries. Performance should happen at different levels on an embedded system. The first level would be the time that the BI server spends on fetching database content. The second level would be the actual time the server needs to render a report or dashboard with the correct information. The third and final level is the transmission of content from the server to the embedded application.

#### **Usability**

When embedding the system into a web application, it is important that the integrated part is modular enough so that the developer has the freedom to arrange it in the best way possible. This goes from style attributes – buttons and labels conventions, regarding size, naming and position – to actual internal functionality – such as error handling.

#### **Compatibility and Portability**

The solution should be easily integrated into already developed web applications. Building a solution that is easily portable decouples it from specific environments and allows a more

compatible solution. Portability should come hand in hand with reusability, translating into less time needed to integrate the solution.

### 4.1.3 Conclusion

In this sub chapter, the requirements of the envisioned solution were gathered and discussed. With the requirements established it is possible to have a more technical overview of the final solution and therefore it gives us more criteria to choose a solution out of the considered ones. It would be ambitious to consider that a solution could fulfill all the established requirements. The next step, described in the next sub chapter, is to choose a solution that meets the greatest possible number of requirements and proceed to evaluate whether it is possible to modify or add elements to the solution so that it meets the stabilized objectives.

## 4.2 First Approach - Metabase

Metabase was the first approach in trying to develop a solution. In this subchapter, it will be described the whole process that has been done around it. Several obstacles were found that prevented the success of this platform to the given problem. However, with this attempt, it was possible to learn immensely about the entire paradigm of the desired solution, and therefore was undoubtedly a process that helped in the construction of the final solution.

### 4.2.1 Why Metabase

Metabase was chosen as the first approach due to a variety of reasons. First, Metabase stands out for being a solution that meets almost all the requirements established, with the particular advantage of being fully web-based. Enchanted with a very pleasant user interface, it smooths even more technical operations such as setting up a database connection or administrating user's permissions.

The dashboards built on Metabase can contain parameters, or filters, that the builder can set up. Then, the end-user can navigate through the data within the limits established by the builder. This is a very promising feature as it can enhance any application analytics view and it can be used to a variety of ends.

The Figure 4.1 shows one dashboard where it is possible to visualize different types of



question cards. It also contemplates two filters, “Quarter and Year” and “City”, which the end-user can input values in order to navigate through the data.

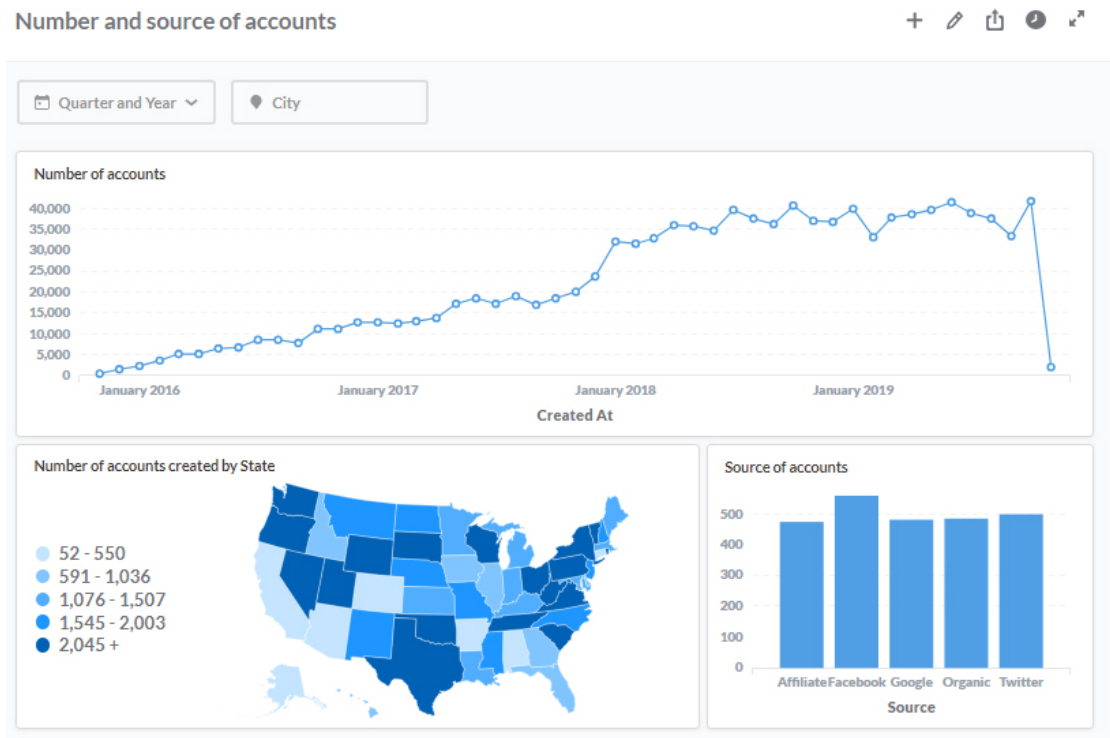


Figure 4.1: Metabase dashboard example.

Metabase includes an application embedding feature that allows developers to easily embed saved dashboards into 3rd party web applications. It provides both the server side and client side code that needs to be inserted, in a variety of different languages. It also allows the customization of the parameters and dashboard style. Figure 4.2 shows a screenshot of the embedding setup feature.

On a more technical point of view, Metabase is built and packaged as a Java jar file. It can also be deployed as a Docker image or into most cloud platforms, such as Amazon Web Services or Heroku. On the website, there are in-depth tutorials to assist in every deploy possibility.

Metabase is an open community that encourages developers to introduce new features or fix founded bugs. Its code repository is published at GitHub.

Preview
Code

To embed this dashboard in your application:

Insert this code snippet in your server code to generate the signed embedding URL Node.js ▾

```

1 // you will need to install via 'npm install jsonwebtoken' or in your package.json
2
3 var jwt = require("jsonwebtoken");
4
5 var METABASE_SITE_URL = "http://localhost:3000";
6 var METABASE_SECRET_KEY = "1bcee16c32a5e937b9343e0ad8db93f55e157d54ff4020519f6d0d73dd513186";
7
8 var payload = {
9   resource: { dashboard: 1 },
10  params: {}
11 };
12 var token = jwt.sign(payload, METABASE_SECRET_KEY);
13
14 var iframeUrl = METABASE_SITE_URL + "/embed/dashboard/" + token + "#bordered=true&titled=true";

```

Then insert this code snippet in your HTML template or single page app. Mustache ▾

```

1 <iframe
2   src="{{iframeUrl}}"
3   frameborder="0"
4   width="800"
5   height="600"
6   allowtransparency
7 ></iframe>

```

**Style**

Border

Title

Light
  Dark

---

**Parameters**

Which parameters can users of this embed use?

Quarter and Year Editable ▾

City Editable ▾

---

**Danger zone**

This will disable embedding for this dashboard.

Unpublish

Figure 4.2: Screenshot of Metabase methodology for dashboard embedding.

## 4.2.2 Architecture

Metabase frontend is developed in React, a JavaScript library for building user interfaces. React is used in the development of single-page applications and aims primarily to provide speed, simplicity, and scalability. React philosophy is based on the concept of “Lego bricks” and, being a library and not a framework, it is possible to add any required JavaScript libraries as add-ons. Metabase also uses Redux, a commonly used library used alongside with React. Redux is a small library for state managing and it is used because controlling the state of one-page applications can get very complicated. Redux smooths that problem by providing one single object that contains the entire application state.

The backend of Metabase is entirely written in Clojure, a commonly used language in the west USA, where Metabase Company is based. The backend publishes a REST API – representational state transfer application programming interface –, with a JSON payload, that is the channel of all the communication between and frontend and the backend.

### 4.2.3 Analysis

As the work on Metabase started to deepen, a lot of useful features, as well as some limitations, were noticed.

On one hand, it has an unbelievably easy way of questioning a database data, producing attractive and functional visualizations and combining them into a dashboard. Figure 4.3 shows the query translation system. It has built-in email and slack automation, dashboard public sharing or embedding, filters, parameters and geo graphs.

The screenshot shows the Metabase query builder interface. At the top, it says "New question" and "SAVE". The query is defined by the following filters:

- DATA: People
- FILTERED BY: State is GA
- Birth Date between January 1, 1960 and January 1, 1969
- VIEW: Raw data
- GROUPED BY: Add a grouping

The visualization is set to "Table". A "Refresh" button is visible. Below the query builder, a table displays the results:

ID	Name	Address	Birth Date	Created At
70	Martin Jenkins	2109 Block Grove	Monday, August 29, 1960 12:00 AM	Tuesday, September 25, 2018 9:20 AM
109	Ottilie Gutkowski	0237 Pasquale Exte...	Sunday, May 22, 1966 12:00 AM	Wednesday, October 17, 2018 4:15 PM
279	John Kutch	17603 Wisozk Flat	Tuesday, May 2, 1967 12:00 AM	Tuesday, August 28, 2018 1:07 PM
333	Shannon Nolan	2989 Marks Trail	Monday, May 6, 1968 12:00 AM	Monday, August 6, 2018 5:27 PM

Figure 4.3: Metabase database search engine.

On the other hand, as for limitations, there were some features that Metabase was lacking. The most important missing feature was a report building engine, in an A4 style. Then, there were some minor but also crucial lacking features, for instance, after building a single question or a dashboard, an important aspect would be the possibility of PDF exporting, something Metabase did not have. Another important limitation was pivot tables. As they provide a very complete and simple overview of data, pivot tables are a crucial form of visualization. However, Metabase pivot tables, as well as every other visualization form, were auto-generated and completely immutable. For instance, modifying axes range on graphs was not possible.

These were some key features that needed to be added or changed and, after some search,

it was noticed that these features were highly requested in the Metabase community.

#### 4.2.4 Development

It was decided that it would be worth to add these features so development started by forking Metabase GitHub project, version 0.26.2. A changelog was added to keep track of all major software changes, as the AGPL-3.0 license obliges.

Because Metabase is a large project with several contributors and with a frontend developed under React, it took some time to get used to the coding methodology, components architecture and actions flow. The code was approached by modifying small things, such as the stylesheets. The implementation of the necessary features began when the familiarization with the code was adequate.

The development started on the PDF export feature. A JavaScript open-source library was added, named *DOM-to-Image*, which can turn arbitrary DOM – Document Object Model – node into a vector or an image. Next, another JavaScript library, named *jsPDF*, was added. It allows to dynamically generate PDF documents. Both of these libraries were MIT licensed. Finally, the execution flow was created: Render the div in which the dashboard is placed into an image, retrieve the dashboard title and other information, feed them the PDF builder and download it. A PDF export button was added on the dashboard toolbar, showed at Figure 4.4, and the execution flow was bounded to it.



Figure 4.4: Screenshot of the developed export button.

The development continued on the configuration of graphs and pivot tables. All the visualizations forms were unmodifiable, and so it was not possible to configure any graph axes or pivot cells. Tools that would allow the final user to have more freedom on the generated content were created. In the pivot tables, a feature that allows the user to toggle the rows and columns was added. The possibility to add a “Total” final row was also added, as showed in Figure 4.5.

## Customize this table

Pivot the table

Toggle axes

Add 'Total' row

State	Aaliyahfurt	Abbigailchester	Abb
AA	1	-	
AE	-	1	
AK	-	-	
AL	1	-	
AP	-	-	
AR	-	-	
AS	-	-	

Figure 4.5: Screenshot of the developed Pivot table features.

As the functionality of integrating dashboards into existing solutions was being further explored, a hidden license was discovered. This license is specially developed by Metabase just for content embedding and goes beyond the AGPL Metabase Licence. It is called “Premium Embedding”, and it removes the “Powered by Metabase” attribution from the embedded charts. It had a monthly or yearly cost.

### 4.2.5 Problems and conclusion

Although Metabase seemed like a very promising approach, the problems that were being faced quickly became higher than one’s expectations.

Firstly, the progress on the most demanding feature, the A4 report building engine, was insufficient. There were only a few JavaScript libraries for online reporting but they were too immature and any implementation experience failed. On top of that, there were license difficulties when trying to merge both Metabase Licences and any new one.

Next, the “Premium Embedding” license was unexpected. Only when further exploration was made on the embedding feature, this special license came across. Embedding developed dashboards into 3rd party applications was one top priority on this work and having a special, restrictive, license for this feature goes against the basics of this current work, as it tries to find a clean, open-source and free solution.

Then, Metabase backend is written in Clojure, a dialect of the Lisp programming language. It has a very steep learning curve and it would be very difficult to do any modification if necessary.

Lastly, there were minor, but still relevant, problems. Metabase started to show significantly less performance when querying a large database. As medical databases tend to be very large, it would definitely result in a performance bottleneck. Besides, if Metabase is hosted in a cloud service, where requests timeouts can be enforced by the host, large queries can never return a response.

All these problems combined, allying with the fact that this work still was on an early development stage, raised the question whether Metabase path would be the correct one. Metabase is a powerful BI platform, designed for enabling otherwise complex queries for everyday business users and it is extremely well thought out. Surely seemed like the correct approach, however, this work requirements demanded a more technical, fine-detailed, platform, where a user can develop more personalized visualizations forms, and afterward, freely embed them into a web application. Even if the work and time needed were used to improve this solution, the embed license was an obstacle impossible to get around. The Metabase approach was closed by pushing all the changes to a public GitHub repository, as some developed features were highly requested. An additional document where it states all the changes made to the code was also added, as the AGPLv3 license demands.

### **4.3 Second Approach - Pentaho**

In this sub-chapter, a discussion around the Pentaho solution will take place. As it will be possible to read, this approach successfully gave a response to this work presented problems. Unlike Metabase, Pentaho is a very complex platform compound by several modules.

As this chapter focuses on the development stage, it will be addressed all the code that was necessary to develop so that it was possible to integrate Pentaho into a web application. In the next chapter, a more careful analysis will be done on the platform itself, describing the whole process from content development to its implementation in a web application.

### 4.3.1 Why Pentaho

After revisiting the state-of-art and consider the remaining options, Pentaho was chosen as the second approach. Although it was not the first choice, primarily because a complete web solution was preferred, Pentaho always stood out from others solutions due to a number of reasons.

Pentaho is a prominent, open-source, project that covers the full spectrum of Business Intelligence life cycle. When Pentaho was first evaluated, it seemed a rich end to end solution, highly configurable and very modular. It certainly gives a lot of different architecture implementation possibilities.

It is divided into four main desktop modules and each module is targeted to a specific Business Intelligence task. For the purpose of this work, it was only necessary to work with one of them, the Pentaho Report Designer – PRD. This module is a reporting tool that enables the creation of relational and analytical reports. It was also necessary to work with Pentaho Server, which serves as a web repository for the content created in PRD.

With high performance and low memory consumption, PRD easily handles even large database queries and therefore it is widely used across all industries, ranging from big corporations to smaller business. Its styling is flexible, pixel-perfect, and allows the report designer to manipulate almost every detail aspect, something Metabase was lacking. Although primarily used for reports, it is also possible to build parameterizable dashboards.

### 4.3.2 Architecture

The Pentaho Report Designer (PRD) architecture was designed to achieve a flexible, yet simple, report engine. It is composed by two main modules (Designer and Engine), and it is released under a GPLv2 license, meaning it allows content developers to freely integrate generated content into commercial or open-source solutions.

Report Designer is a desktop reporting tool that serves as a visual design environment. It is where a report designer builds reports, with a drag-and-drop feature. Users need to have a basic understanding of databases and query building in order to get the most out of Report Designer. It has a moderate learning curve.

Report Engine is a Java reporting library that renders reports based on the parameters set on Report Designer. It is possible to connect virtually any data-source, and reports can

be exported to HTML, Excel, PDF, Text or CSV and HTML.

The content developed in PRD can later be uploaded into the Pentaho BI Server. This server is a web application for sharing and managing content. It has some interesting features, for instance, users management with permissions and email automation. It provides a REST API, with a JSON payload, that is the channel of all communication, both for internal modules, like PRD, and external clients. The Figure 4.6 shows the necessary steps to build a report and publish a report, as well as the steps for requesting it from the Pentaho Server. In the next chapter, all the elements presented in the image will be described in detail, as well as the intermediate steps.

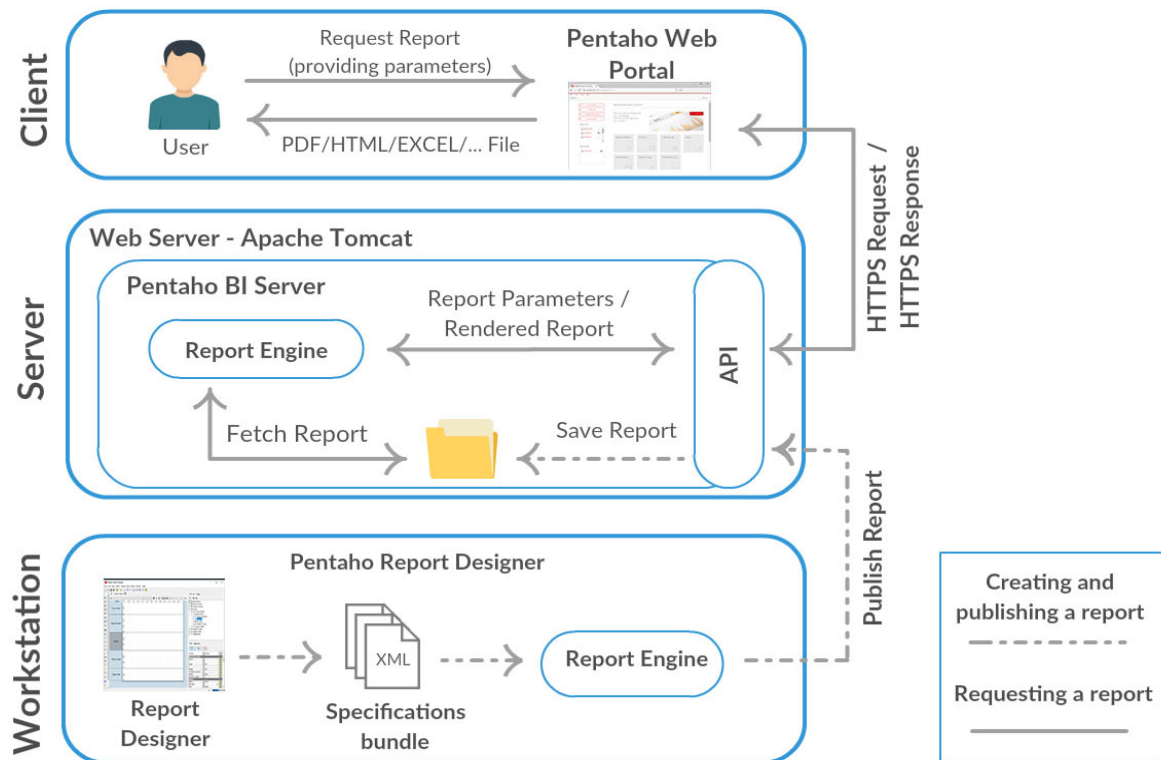


Figure 4.6: Pentaho platform simplified overview and content action flow.

### 4.3.3 Analysis

In an initial stage, a few mock reports were developed and published to understand the processing logic and understand the platform's limitations.

The first thing that was immediately perceived was that online documentation regarding



report creating was lacking. Several technical challenges were faced that were solved using a trial-and-error approach, for example, adding a new JDBC driver – Java Database Connectivity, or creating and manipulating a pivot table. Nevertheless, PRD was extremely versatile and offered the opportunity to develop personalized visualizations forms.

One of the most interesting features, that opened many doors, was the ability to create report parameters. A parameter is a variable that is defined by a value type – String, Boolean, Date, Integer, Double, etc. – and a value boundary. Its value can be injected into the database queries and therefore used to filter the report information. There is also the possibility to use parameters to modify the report layout, by injecting the parameter value into a graph configuration, for instance. Parameters can be set as hidden, normal or mandatory.

When the report building is finished, it can be directly exported into the already mentioned formats, or it can be uploaded to the server. In order to do that, the designer must provide the server location and his login credentials. Afterward, he chooses the directory path and exports the report in a raw format, a *prpt* file. Pentaho Server has a built-in report render, meaning it can also export the report to all the possible formats. To render a report on Pentaho Web Portal, a small user interface is prompted to the user for him to insert the mandatory parameters. Then, after the user filled the parameters and chooses the output format, the report will be displayed or directly downloaded.

Although there is a serious lack of information on PRD, the Pentaho BI Server has a great community support and documentation. The API is partially documented in the source code.

Unfortunately, Pentaho platform does not offer any solution to embed developed reports into a 3rd party web application.

#### **4.3.4 Development**

Report embedding feature is a key requirement and, carefully analyzing the Pentaho Server API, it seemed that it had everything needed to develop such feature.

The main goal was to develop a solution that could be embedded into 3rd party web applications and, given a certain report, a user interface would be prompted for the user to fill the parameters. Afterward, the report would be rendered and displayed, or downloaded, according to the user input. The content could also be directly downloaded, according to the developer needs.

The solution would have to be easily embedded into already developed web applications. As most modern web application follow a Model-View-Controller (MVC) architecture, the solution was developed so that it could be easily integrated into this paradigm. The ambition was to create a system described in Figure 4.7, where the content displayed in *page.html* is actually provided by the BI engine.

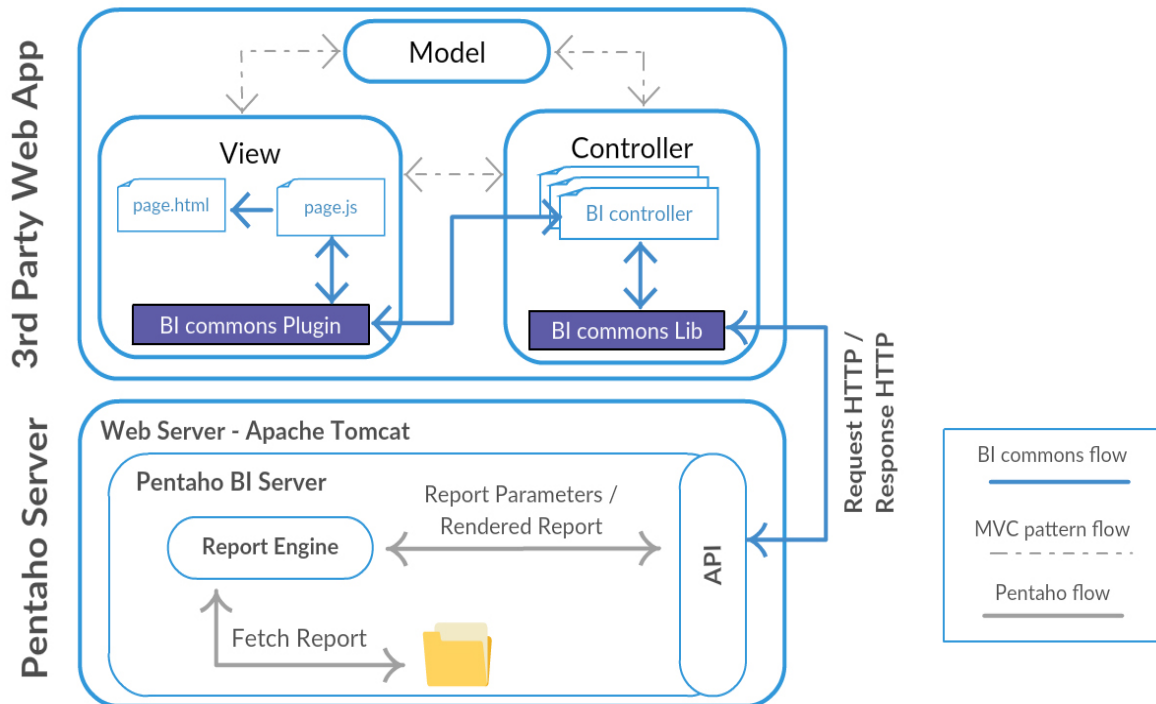


Figure 4.7: Proposed solution architecture.

Before the development started, the crucial aspects of the pretended project, such as the architecture and design patterns, were outlined. Throughout the development, an effort was made to obtain the greatest possible abstraction, as it decouples software elements and makes it easier to extend the application. In this case, a solution was developed targeted for Pentaho BI. However, there may surge the need to change BI solution, so the developed solution could not be anchored to this engine.

The first step in the development stage was to create the server side library, the *BI-commons-Lib* (Fig 4.7). The main idea was to develop a Java API client library that communicates with the Pentaho Server API. This client library, sometimes also called helper library, provides methods that encapsulate the communication with an API. It helps applica-

tion developers with their development projects by hiding the explicit API HTTP requests. It provides one single point of communication, ensuring a better support and best practices. It also reduces the amount of code that developers need to write.

This library was developed in a Java Maven project. To assist with the development of REST methods, we used Unirest, a lightweight HTTP client, with many features that smooths the communication with any API endpoint.

The development of *BI-commons-Lib* started with the API authentication implementation. Pentaho BI Server supports Basic authentication. As the REST API is stateless, each request that is made to the server requires an *Authorization* header, with the value *Basic* followed by a Base64 encoded *username* and *password*, separated by a *colon* (example: admin:password). The information transmitted in the API calls is normally in plaintext. To ensure the confidentiality of all the transferred content, it is necessary for the Pentaho Server to use an HTTPS protocol. In this manner, all the communication will be protected by the SSL/TSL security protocol, over TCP/IP.

Afterwards, the development continued with the implementation of a method that retrieves a given report. To do so, a POST request must be made to the following REST URL:

*https://example:port/pentaho/api/repos/<path>/generatedContent*

Where *<path>* is the location of a given report on the server repository directory. The POST request body must contain all the parameters the report expects, plus an additional rendering parameter called *output-target*, which controls the rendering format (HTML, PDF, EXCEL etc.). The returned result is an *InputStream* object.

Several other functions were developed. Methods regarding the information of a report, such as the verification of mandatory parameters, were first addressed. Then, methods regarding the structure of the repository were developed, such as the possibility of obtaining the complete tree of content or the possibility of uploading content.

The development of this library was an iterative activity and more methods were developed depending on the implementation needs. Some methods were more difficult to implement than others, as documentation regarding many API endpoints was lacking. Sometimes it was necessary to deep dive into the source code and understand exactly how the request was processed.

After the server-side library was partially completed, the development of the client-side started, the *BI-commons-Plugin*.

As this feature must be easily embedded into any number of web applications, it was decided that the development of a JQuery plugin was the best solution. A JQuery plugin provides a layer of abstraction by omitting the functionality logic contained. It helps the developers focus on what's important while writing clean and readable code. Packaging common function within a JQuery plugin also provides portability by easily allowing its integration into already developed web applications.

The developed plugin provides a simple way of interacting with the previously developed library, at the front-end level. To instantiate the plugin, the developer must provide the report path on a Business Intelligence server, as well as the location of some essential methods built around the Java library. The plugin will immediately fetch all the given report information, and if the report has mandatory parameters, a user interface will be prompted to the final user. From here, the final user can modify any parameter, within the limits established by the report builder, and preview the results.

As this plugin must be smoothly integrated into already developed web pages, one of its key aspects is the customizability. The developer can set any styling to the prompted user interface by defining a style to the form component, to the input fields, or to 'submit' button, giving it a more appropriated appearance. Error messages were also left to the developer. When an error occurs, an error event is triggered, and it is up to the developer to handle it in the best way he considers. Figure 4.8 is an instantiation example of the *BI-commons-Plugin*.

Finally, documentation has been created describing all the steps necessary to integrate the components developed in an existing web application. This document details all the methods in the Java library, as well as on the JQuery plugin. An example of integration in a MVC architecture was also developed.

### **4.3.5 Problems and conclusion**

The development stage reached the end when the developed solution seemed to smoothly work on any tested situation. By analyzing the final solution, it is possible to conclude that it certainly meets the established requirements, but it also raises some problems that can be seen as limitations.

```

$('#BIdiv').BIcommonsPlugin({
  biServerUrls: {
    biServerBaseUrl: 'http://localhost:8080/',
    getReportUrl: PentahoController.getReport(),
    getReportMandatoryParamsUrl: PentahoController.getReportMandatoryParameters(),
    reportHasMandatoryParamsUrl: PentahoController.reportHasMandatoryParameters()
  },
  reportSettings: {
    reportname: 'home/MDTeam/report.prpt',
    parameters: {
      'output-target': 'pageable/pdf',
      patientID: 1
    }
  },
  btnIcon: 'fa fa-search fa-fw',
  inputClass: 'form-inline',
  error: function (message) {
    console.error('Error: ', message)
  }
});

```

Figure 4.8: *BI-commons-plugin* instantiation example.

A major technical problem with this solution is the necessity of a running instance of the Pentaho BI Server at any time. In order to fetch any report, the server must be up and running. Furthermore, if the server instance is not hosted on the same local network where the web application is, there are multiple external factors that could compromise the effectiveness of the solution. However, this problem was known from the start and any embedded BI solution would have to undergo with this condition.

Another limitation that this solution brings is the fact that Pentaho Report Designer is more oriented to technical users, with SQL knowledge. Delivering the report building work to a healthcare provider would be unreasonable. Creating or modifying a report requires delegating the work to a technician. However, with the use of parameters, the report can become quite versatile, giving to the end-users the possibility to make huge changes to the report, both in the arrangement of the elements and in their content, within the established limits.

Nevertheless, the developed solution opens the door to many possibilities. Giving the end-user the versatility to manipulate a report or a dashboard in an easy manner was a crucial

objective of this work. On one hand, Pentaho Report Design allows the development of tremendous detailed report aspects, which can be further refined through the use of parameters. On the other hand, the developed solution offers the possibility to integrate the Pentaho ecosystem into already developed web applications, allowing developers to easily enhance their website with an analytical component and ultimately providing the end-users with the possibility of developing personalized visualization forms.

Throughout the development of case studies, where the solution was tested, several situations were found where it was necessary to modify some minor aspects of the solution, such as adding methods that a particular case study required, or modifying existing ones.

## Chapter 5

# Pentaho Implementation

This chapter addresses *how* Pentaho should be integrated into an existing environment and how it should be used. First, the module where it is possible to develop reports and graphs, the Pentaho Report Designer, will be analyzed. Here, the necessary steps to develop a report, as well as the architecture behind it, will be described in detail. Next, Pentaho BI Server will be presented, along with all its main features and characteristics. A description of how to have a configured running instance of the server will be made. Finally, it will be shown how Pentaho can be integrated into an existing application. In this Chapter, the person who builds a report will be addressed as a technician.

The development was based on Pentaho CE version 8.0, which can be downloaded free of charge at <https://sourceforge.net/projects/pentaho>. A different version may differ in some characteristics.

### 5.1 Pentaho Report Designer

The Pentaho Report Designer (PRD) is a desktop graphical tool for developing reports. This subchapter introduces the main characteristics of PRD while providing a walk-through on *how* a report should be conceived.

It is said that in a Business Intelligence environment, about 75 to 80 percent of the final displayed content is presented in the form of a report [25]. Therefore, in the present work, there was a higher effort in studying this approach.

Pentaho Report Designer allows technicians to build reports tailored to the end-user needs.

With a drag-and-drop feature, it allows the creation of pixel-perfect, dynamic reports, where the displayed content can easily change according to parameters values. For instance, the technician can create a report where it displays a clinical history for a given *patientID*. The doctor, as a final user, just needs to input the desirable *patientID* to get the corresponding report. In this simple use case, it is necessary to have dynamic elements – that change according to the input value, a dataset – which is the associated data after querying the database –, and finally a parameter, that is the *patientID*. In this subchapter a brief introduction to all these crucial elements will be addressed.

### 5.1.1 Architecture

The architecture behind most reporting solutions is similar. Figure 5.1 shows an overview of the elements that compose the PRD reporting tool.

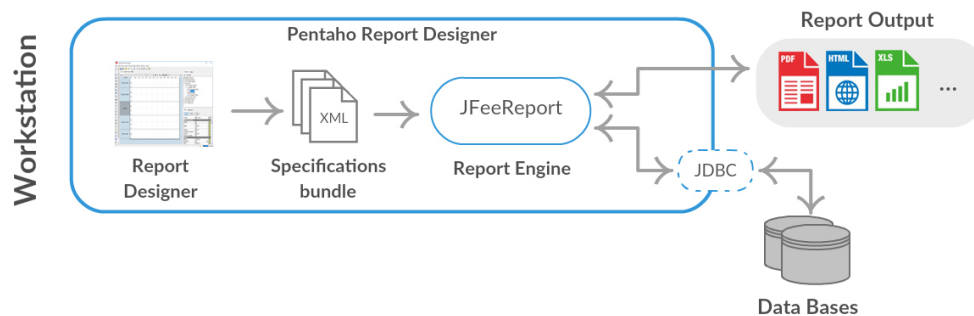


Figure 5.1: Pentaho Report Designer architecture overview.

First, there is a graphical application where the technician can build a report and specifications are defined, in this case it is called Report Designer. A closer screenshot of the application interface can be seen in the next subchapter, in Figure 5.2.

Then, all the developed work on the Report Designer is translated into a *.prpt* bundle file. This file contains several XML files with the report specifications, such as *layout.xml*, where all the layout information is described, or *\*-ds.xml*, where query definitions are presented. All the JDBC connections settings included in this bundle, such as the username, password, host and port, are in plain text. So, on a security point-of-view, it is preferable to use JNDI connections. In a Unix-like Operating System environment, JNDI connections can be added at:



*\$HOME/.pentaho/simple-jndi/default.properties*

These configurations will not be included in the *.prpt* bundle file, only the JNDI name, and so, it is necessary to define the same JNDI connections, or JDBC, on the Pentaho Server side. Also, if any JDBC is missing, it should be added at the *lib* directory, that is located on the root installation directory of PRD.

Finally, the *.prpt* is fed to the Reporting Engine. This engine is a Java reporting library, named JFreeReport, that can be used both in the desktop application and in the Pentaho Server. It is responsible to render the developed report based on the XML files specifications. It starts by fetching the data from the data sources and then, accordingly to the output format, renders the report.

The developed reports can be saved on the local machine or published into the Pentaho BI Server, by providing the server URL, port and user credentials.

### 5.1.2 Report Structure

When a new report is started, it is automatically divided into multiple sections. Figure 5.2 shows a screenshot of a blank report structure as an example. This sections contain different attributes where each one plays a different role when a report is rendered. Some of them are mandatory, such as Page Header/Footer or Report Header/Footer, and cannot be removed. Others, like Details, are flexible and can be freely added or removed from the report structure.

At the left side of the Figure, it is possible to visualize a palette of all the elements that could be added to the report. In the next sub-chapter they will be addressed. At the right side of the Figure, it is possible to see the Structure of the report on a nested layers schema. All the elements added to the report will be positioned inside their parent.

Every section or report element have properties that are divided into Style and Attributes (Figure 5.2). Attributes control the behavior and the content, while the Style controls the appearance, such as positioning, colors and fonts. Most of the properties are, by default, inherited from the parent, similar to a CSS file.

All the sections have special purposes. The following list briefly describes each one, by presenting the main characteristics and typical uses:

- Page Header/Footer – All the elements located in this section will be replicated in

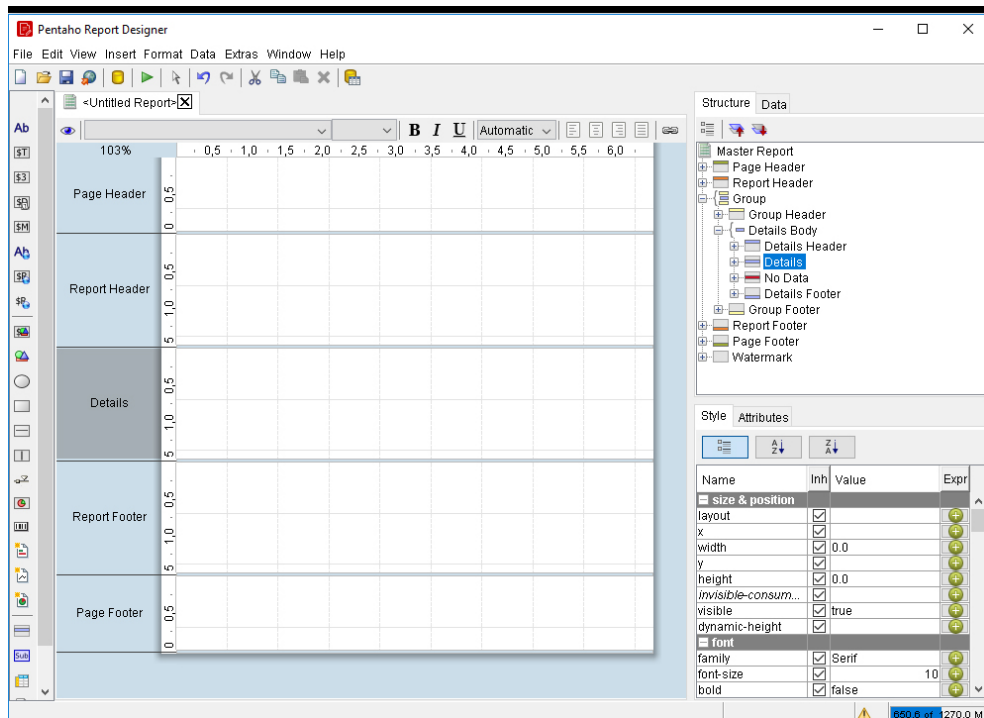


Figure 5.2: Screenshot of a blank report in Pentaho Report Designer.

every page of the report. This section is used to place static information such as the organization name, logo, dates or page numbering.

- Report Header/Footer – All the elements positioned on this section will be displayed only on the first report page. This section is typically used to give a brief report introduction or summary. Information placed on this section is normally the report title, report parameters and report totals.
- Group Header/Footer – A group is used to organize actual content. The previous Figure shows, on the Structure tab, the group Details. It is composed by a Header, a Footer, a Body and a No Data section. The Group Header/Footer is used to place static information such as table headers, or subtotals. Both the Details Header and Footer are hidden by default when a new report is created. At any section of this Group it is possible to nest another group, creating a hierarchical structure.
- Group Body – This section is used to place individual rows from a query result. It is only necessary to place one text-field, with the same name as the query field alias.

When the report is rendered, all the rows from the query result will be displayed.

- No Data – This section is rendered when queries from the Group Body return no results. It is typically used to place informative messages.
- Watermark – This section is used to place a message or an image at a fixed location on the report. It will be rendered in the background and it is typically used to display an informative message such as “Confidential” or the company logo.

### 5.1.3 Report Elements

As already mentioned, on the left side of Figure 5.2, the palette with all the elements that can compose a report is present. It is possible to drag-and-drop any element to a section of the report. Table 5.2 gives a brief description of some available options, ordered from top to bottom. Some of them are very similar, such as “Vertical Line” and “Horizontal Line”, and in those cases, only one of them is described.

Although each element has special properties, most of them are similar. In Figure 5.3 it is possible to see a *Label* element’s properties.

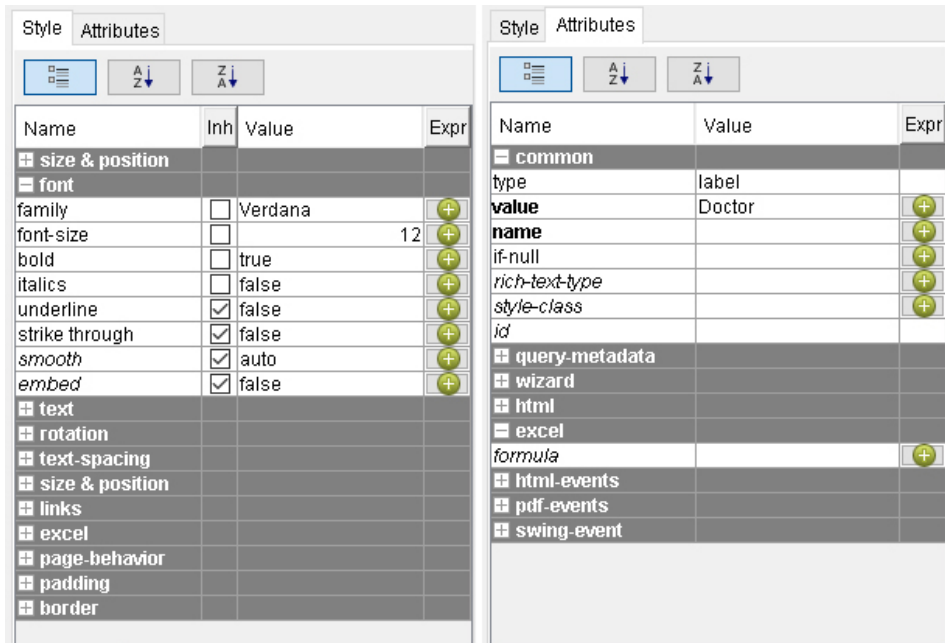


















Figure 5.3: Screenshot of the ‘Label’ element properties.

The element’s properties are divided in Style and Attributes. Style controls the appearance

Table 5.1: Description of the main report elements in Pentaho Report Designer.

Icon	Name	Description
	Label	A static text String that cannot be changed.
	Text Field	A dynamic text field where the value comes from a query or function.
	Number Field	A dynamic numerical field where the value comes from a query or function.
	Date Field	A dynamic date field where the value comes from a query or function
	Message Field	A combination of static and dynamic content. Used to combine text labels with other dynamic fields. For instance: Patient: $\$(firstname)$ $\$(lastname)$ . Birthdate: $\$(birthdate, dd/MM/yyyy)$ .
	Resource Label	A static string that comes from a resource file. A resource file is a text file in a key:value structure, conceived by the technician. When referring a key, the value will be displayed. Used mainly for translations.
	Image Field	A dynamic image where the content comes from the database or an URL.
	Image	A static image that is embedded into the report.
	Rectangle	A rectangle vector.
	Vertical Line	A vertical line vector.
	Survey Scale	A progress chart. The goal and the current value can be static or dynamic.
	Chart	A chart that can be edited in the Chart Editor. There are several chart types to choose from. Data comes from a query.
	Line Sparkline	A small line chart used inline, i.e. in a table cell.
	Band	A group, containing a Header, Footer and Details, as mentioned on the previous sub-chapter.
	Sub-Report	A separated report page that can be added to a parent report, creating a hierarchal structure. It is used to create separated queries, since it is only possible to have one query per report.
	Crosstab	A pivot table.

of the element, while attributes control the behavior. The “correct” symbol on the Style tab means that the value of that property is inherited from the parent. As it is possible to see, there is a high refinement degree for each report element, resulting in a very versatile report.

It is possible to associate an expression to each element's property by clicking on the "+" symbol. An expression is a formula that changes the *value* of the associated property.

Formulas can range from simple mathematical operations, to more advanced task, such as counting the number of elements in a list. It is possible to create conditional structures (if/else) and there is a list of more than 140 functions to choose from, on the Expression Editor. It is also possible to inject a parameter into the element value, through an expression.

#### 5.1.4 Parameters

Parameters are variables that the technician creates and where its value is given by the end-user, every time the report renders.

The technician creates a parameter and defines a range of possible values from where the end-user can later choose from. The definition of possible values is given by a database query. Parameters can also be in the form of free text. Figure 5.4 shows a screenshot of the Edit Parameter window. As it is possible to see, the figure portrays the edition of a date parameter.

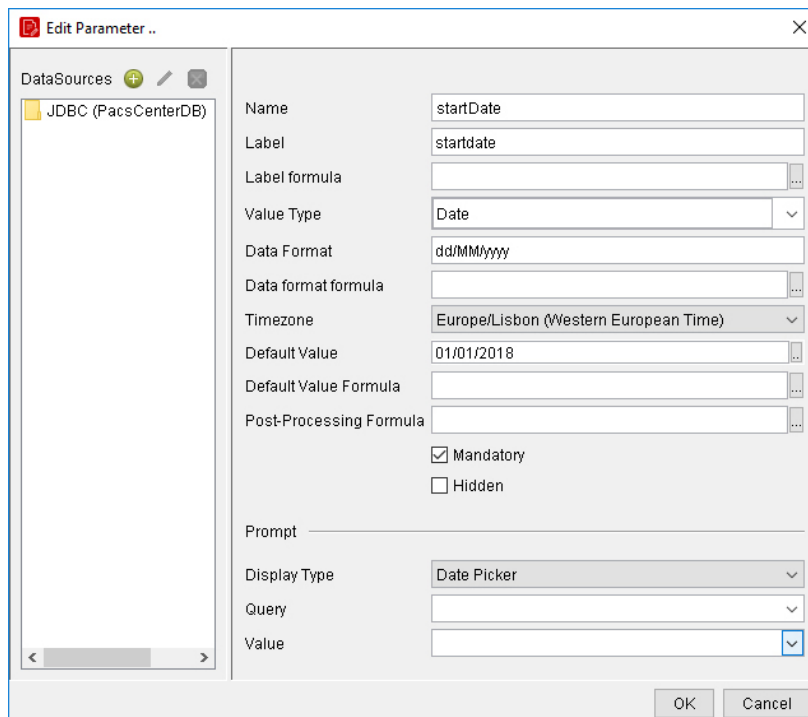


Figure 5.4: Screenshot of the Edit Parameter window in Report Designer.

Parameters can be set as mandatory, where the report doesn't render until the parameter has a value, as hidden, or as normal.

After the parameter has been defined, it is possible to attribute its value to an element's property. If a value from an element can take the value of a parameter given by the final user, it becomes clear that the final user can have the authority to make deep changes to a report without much effort, both on the aesthetic and behavior side. In theory, it is even possible that a technician builds fully parameterizable report, where every element's property is controlled by the final user input.

A parameter can also be used to inject its value in a database query. In order to do this, in the *Where* clause of a query, it is necessary to place the parameter name, enclosed with curly braces and preceded by a dollar sign, for example: `#{startDate}`. Figure 5.5 shows a simple query example with the use of parameters.

```
SELECT  action.id,
        action.date,
        action.type

FROM viewer.action action

WHERE (action.date Between #{startDate} and #{endDate})
```

Figure 5.5: Query code example with parameter injection.

Logically, in order to execute this query, *startDate* and *endDate* must have values. This can be achieved by defining these parameters as mandatory or attributing a default value. Thus, the report will always render correctly.

When the report is finished, and the technician desires to render the report, a user-interface will be prompted with all the parameters labels and input fields in order to be filled (not the hidden parameters). Figure 5.6 shows a screenshot of the parameters that need input values so that the report can be rendered. Mandatory parameters are noted with a “\*”. The selection of a date time is assisted by a *date picker* when the three dots are clicked. This is automatically done because the *Display Type* of this parameter was set as *Date Picker*.

When the report is uploaded to the server, the same parameters will be requested to the end-user. Only after the mandatory parameters have been filled, the report is rendered.

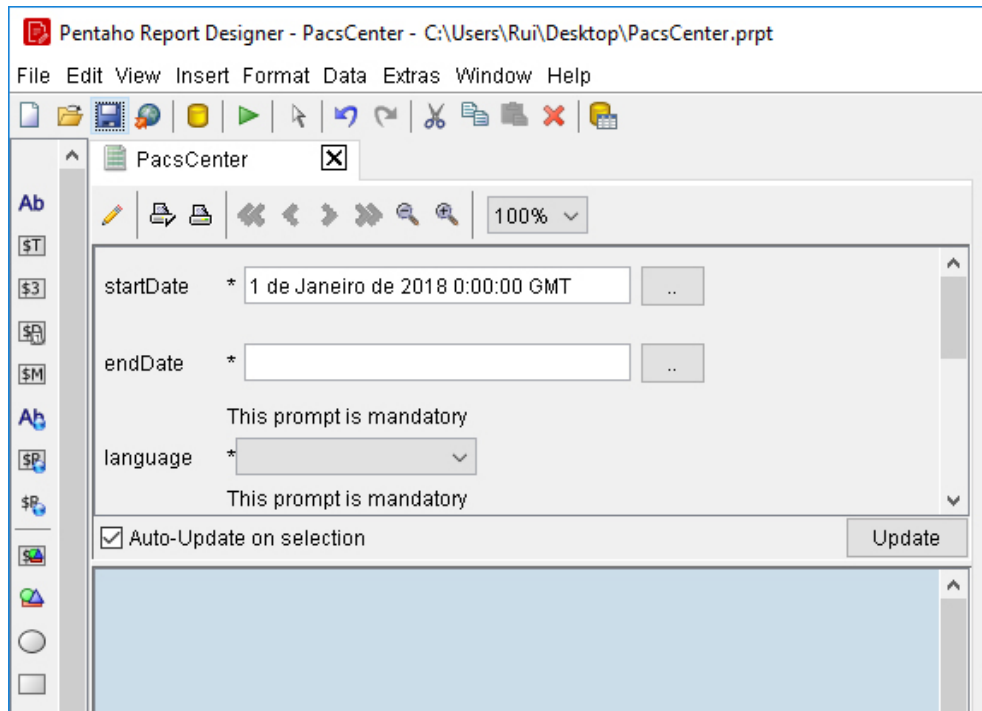


Figure 5.6: Screenshot of PRD when a parameterizable report is requested.

### 5.1.5 Data Sets

Choosing the correct data to be displayed in the report is one of the most important steps. A Data Set is nothing more than the resulting data that is fetched after querying a database.

A page report only supports one Data Set, meaning that only the variables fetched by a single query can be placed in a page report. However, a report can contain a sub-report, with their own Data Set. At Table 5.2, the button sub-report was addressed.

Every time Pentaho Report Designer is started, it automatically fetches all the JDBC connectors as well as JNDI connections. Then, when a technician desires to create a new database connection, it is necessary to select the JDBC and input all the necessary configurations (host, port, credentials, database name...). Afterward, he can freely build the database queries.

To help the technicians with the construction of Data Sets, PRD has a Query Designer Editor that works with a simple drag-and-drop approach, as shown in Figure 5.7. The desirable tables can be dragged to the black canvas and the Editor will list all the available columns and connect the associated tables. The technician just needs to cherry-pick the required columns,

by clicking on the respective checkbox. Afterward, all the information on the black canvas will automatically be translated into SQL code.

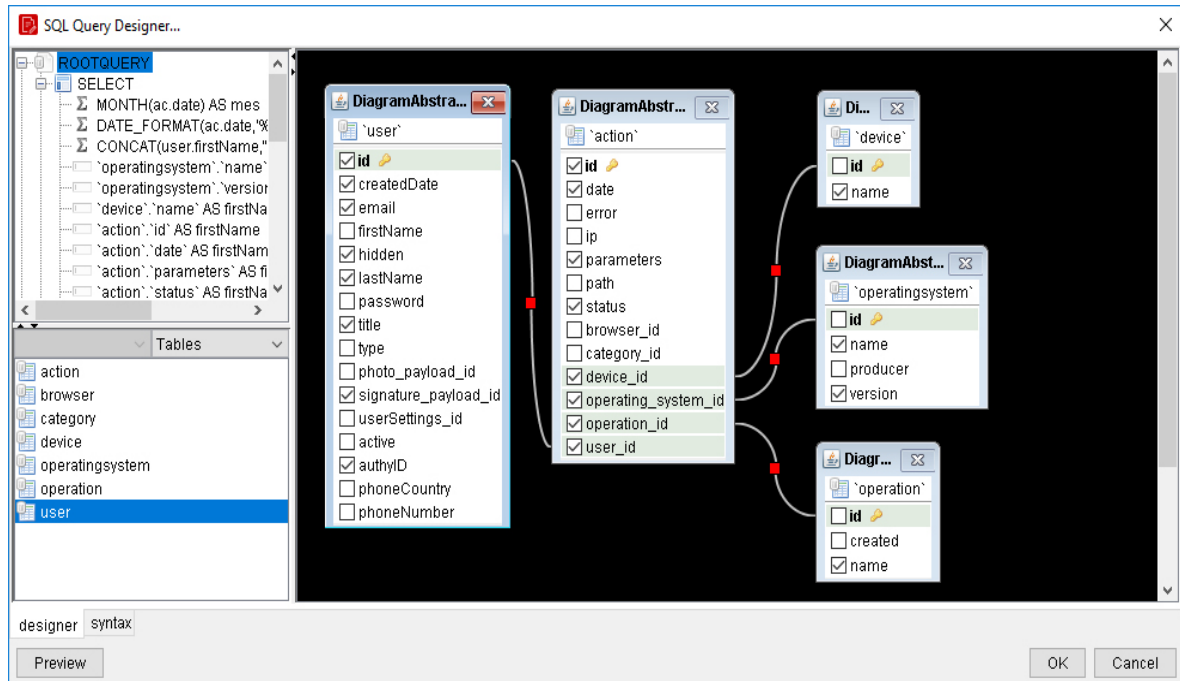


Figure 5.7: Screenshot of the Query Designer Editor in Report Designer.

This editor has limited features. More advanced database operations, such as concatenating strings or date time formats, needs to be done by someone with database query knowledge.

After the query has been completed, all the columns selected on the query will be displayed in the Data tab, as shown on the right side in Figure 5.8. All queries must have a unique name. In the same Figure, the query name is “Reports per Doctor, per Year and per Month”, which is under the JDBC named “PacsCenterDB”.

Afterward, the technician just needs to drag-and-drop the available columns into the report canvas. If the dataset in that column is just one row, it will be displayed. If the dataset contains more than one row, it is necessary to place the column into a “Details” group, in order to display all the column content.

In case of being a pivot table, the content can be placed just as shown in the previous Figure, in a Crosstab section. Crosstabs, behave just like sub-reports, where they have their unique report page and unique Data Sets.



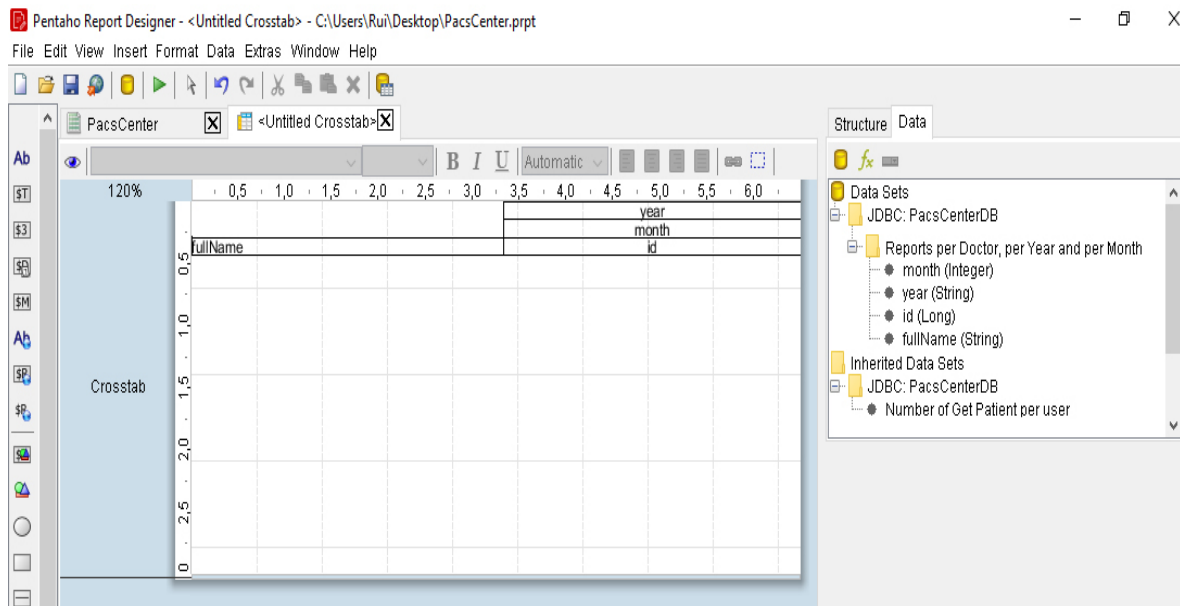


Figure 5.8: Screenshot of a pivot table example in Report Designer.

## 5.2 Pentaho BI Server

The Pentaho BI Server consists of a set of smaller programs, implemented as Java servlets, which are responsible for receiving requests from a client, processing them and returning a response.

In this sub-chapter, the core functionalities of Pentaho BI Server will be addressed. First, an analysis of the technology on which the server is based will be carried out, as well as the methodology of the requests. Next, a brief overview of the server's main services and components will be made. Then, the client-side interface will be addressed. Finally, the Pentaho Server API will be analyzed, as well as its main end-points.

### 5.2.1 Technology

When a client requests certain task to the Pentaho Server, the work to be done is delegated to a servlet. A Java servlet is a class that handles the server logic and conceives a response object to a certain request.

To handle the network logic, such as parsing the HTTP request and handling connections, it is necessary to use a *servlet container*. This container is responsible for interpreting a client

request, delegating the work to a servlet, and returning the response object, created by the servlet, to the client. This paradigm of Java web applications is called *Java Servlet Technology*.

The *servlet container* is hosted on a centralized server, available to a local network, or to the entire internet, depending on the configuration. The Pentaho BI Community Edition comes with Apache Tomcat that behaves both as a *servlet container* and a web server. Figure 5.9 shows a simple overview of a request handling in the Pentaho Server architecture.

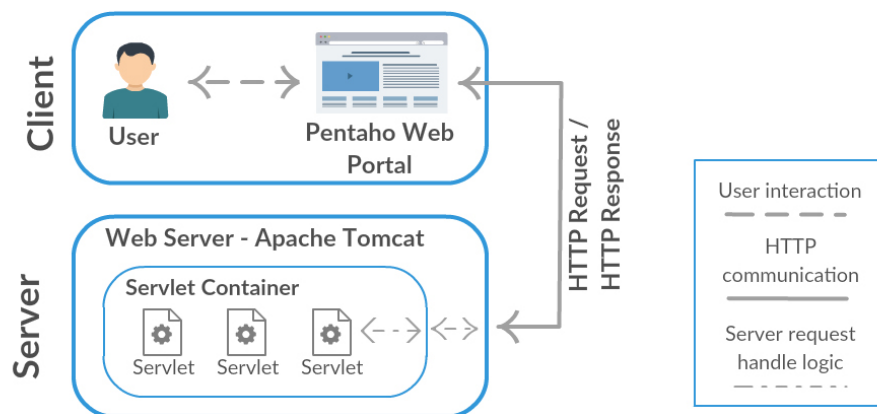


Figure 5.9: Pentaho Server HTTP request flow.

## 5.2.2 Logic

Pentaho Server offers several components that are implementations of already existing solutions. For example, the component that handles the user authentication and authorization is built upon Spring Security.

The server mainly works as a repository, where the content generated by the desktop modules, such as Pentaho Report Designer, can be stored.

Pentaho Server contains the reporting engines necessary to render reports. So, when a client requests a given content, the server fetches the file from its directory, and delivers it to the corresponding engine, as well extra content request parameters that might have been given. Then, after the engine produces the result, a response object is returned to the client.

Figure 5.10 shows a concrete example of all the logic behind a request. Consider that a client request a report hosted on the server, and provides a parameter. The server fetches

the *.prpt* bundle file that was published in the server by the technician and delivers both the bundle file and the given report parameter to the JFreeReport engine, hosted within the server. From here, the JFreeReport request a connection from the server database connection pool and fetches the necessary data set from the database. After the engine finishes rendering the report, the content is encoded into a Binary Large Object – BLOB – and sent to the client in a response object. On the client-side, the BLOB content is built according to the request format (PDF, HTML, EXCEL...).

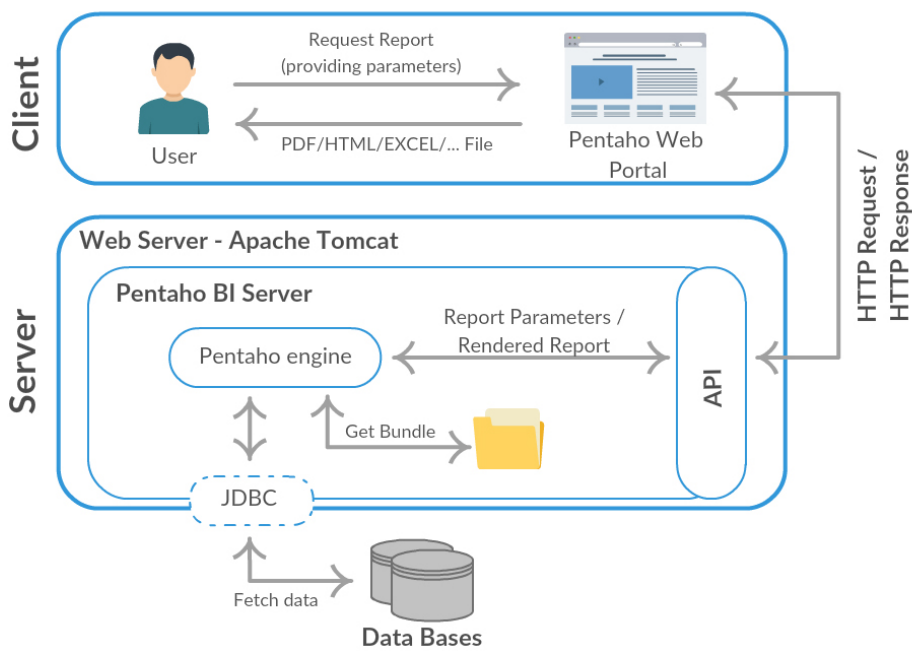


Figure 5.10: Pentaho Server operations flow for a report request.

Although it works mainly as a content repository, it is also possible to execute some work on the server. For instance, it is possible to schedule a task that can be used in the background for execution of reports or email scheduling. Pentaho uses Quartz, an open source job scheduling library.

The Pentaho Server can be used to send e-mails, after a standard SMTP server has been configured in the file *email-config.xml*, located in the following path:

*<installation-directory>/pentaho-solutions/system/smtp-email*

It is necessary to restart the server after changing any configuration file.

### 5.2.3 User Interface

Pentaho is shipped with a client-side interface called Pentaho User Console. It is mainly used to explore and open existing content. Figure 5.11 shows a screenshot of the interface home page, after a successful login.

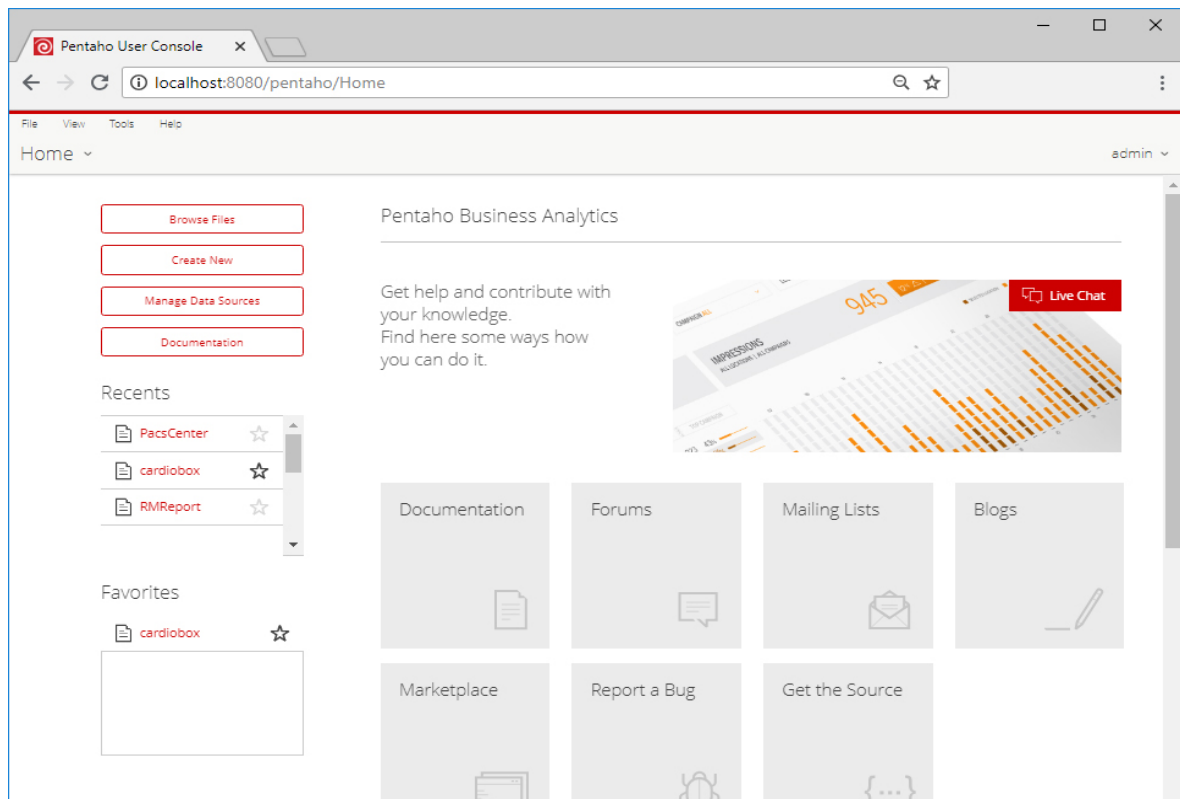


Figure 5.11: Pentaho Home page.

With this web interface it is possible to manage the whole file system, and doing operations such as upload, delete, check properties and visualize content. It can also be used to create content, although very limited. When a user opens a report on the web interface, a small user interface will be prompted to the user for him to insert the parameters values. When all parameters have been given, the report will be rendered to the requested format. If the user requested an HTML or PDF format, the report will be rendered on the same page. Any other format, such as EXCEL, will be directly downloaded.

## 5.2.4 API

The Pentaho Server has a Representational State Transfer Application Programming Interface – REST API - with a JSON or XML payload. The list of existing end-points is extensive, but they can be categorized into four different topics:

- File Management – End-points that allows the management of the server directory. They provide methods to upload, download, view and modify properties of existing files and folders.
- Data Sources Managing – End-points for uploading, downloading and managing data sources on the server.
- Scheduling – Allows the management of schedules, such as creating, uploading, deleting and listing schedules.
- User Role Management – End-points to user’s management. They are used to check permissions, add/remove users or modify roles.

Any HTTP request to the API needs a token that represents the authentication of the user that is requesting the service. Pentaho BI Server supports Basic Authentication, and so, each request that is made to the server requires an *Authorization* key in the HTTP request header, with the value *Basic* followed by a Base64 encoded *username* and *password*, separated by a colon [*Base64(admin:password)*]. For example: *Authorization - Basic YWRtaW46cGFzc3dvcmQ=*.

All the content transmitted in any API call is in plaintext. If a package is captured, it is easy to decode the username and password from the HTTP header, since Base64 is a very weak encryption algorithm. Furthermore, all the content sent and received can also be reconstructed, compromising the confidentiality of data. To ensure the security of all transferred content, it is necessary for the Pentaho Server to use an HTTPS protocol. In this manner, all the communication will be protected by the SSL/TSL security protocol, over TCP/IP.

Table 5.2 briefly describes some useful end-points. The pathId is the path to the desired file on the server directory, where the forward slash is replaced by a colon punctuation mark.

For instance “*/home/admin/report.prpt*” translates to “*:home:admin:report.prpt*”. All the end-points presented belong to the “File Management”.

Table 5.2: Description of the main API endpoint in the File Management category.

End-point Path	Method	Description	Response
<i>/repo/files/{pathId}/tree</i>	GET	Retrieves a recursive list of children – files and folders – of the given path, with properties.	XML/ JSON
<i>/repo/files/{pathId}/canAccess</i>	GET	Checks if the current user has permission on the given file.	Boolean
<i>/repos/{pathId}/parameter</i>	GET	Retrieves a list of all the parameters a report expects. Each parameter is highly detailed, including data type and possible values.	XML/ JSON
<i>/repos/{pathId}/generatedContent</i>	POST	Retrieves a rendered report. The POST body must contain all parameters the report expects, plus an additional parameter that controls the format.	BLOB

Most end-points are documented in the source code, complemented with examples. This smooths the development of a client library when the goal is to integrate Pentaho into existing applications.

### 5.3 Integration

In this sub-chapter, it will be discussed the typical integration of Pentaho in an organization. Additionally, it will also be presented the intended way that Pentaho should be used when embedded in a web application with the modules developed in the previous chapter.

When a Business Intelligence solution is integrated into an organization, it is imperative that it connects to one or more existing data sources, as the foundation of a BI solution is to work in the data generated in an organization. The data sources can vary from single files to entire databases, generated by the most diverse activities.

In a hypothetical company that has a web application that sells goods or services to the users, all the data generated, both logical and behavioral, can be stored in a database. When a BI solution is integrated into this environment, it is typically used by the business managers to swiftly move across data, in order to gain insights. All the work around the BI solution is

decoupled from the actual web application, and it is therefore used as a stand-alone tool.

But there are cases where a web application would benefit from a data analytic component upon the data it generates, providing users with powerful visualization tools on their own data. It would certainly enhance the application, and users would have the access to personalized content.

There are two hypothesis on how to add such analytical component to a web application. The first would be to develop an analytical tool from scratch. The second would be to integrate an analytical BI component. The second approach is the basis of this work, where the goal was to create a strong relationship between the web application and the Business Intelligence solution. Figure 5.12 suggests what has been said so far, where the analytical content in the web application is delivered by the Pentaho Server.

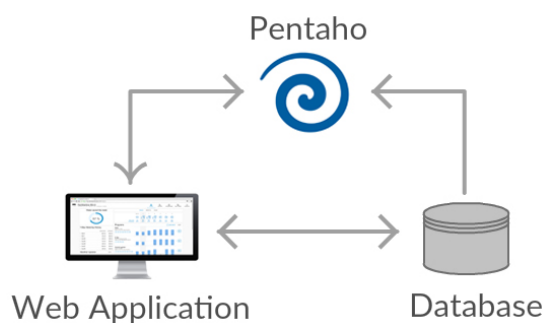


Figure 5.12: Pentaho integration overview.

Looking back to the Figure 4.7 on Chapter 4, it is possible to recall the proposed integration process of Pentaho in a web application. The first step to integrate the Pentaho solution into an application is to have a running instance of the Pentaho Server. The server can be running on the same machine where the application is hosted or in an independent one. Nowadays, most applications follow a Model-View-Controller – MVC – architecture. The modules developed in the previous chapter allow an easy integration of the Pentaho Server into an already developed web application that follows the MVC architecture.

To integrate Pentaho into an application, developers just need to first import the *BI-commons-Lib* into the project. This library encapsulates all the communication with the Pentaho Server API, resulting in a cleaner and simpler integration. Requesting content from the Pentaho Server becomes as easy as calling a function with the report path as a parameter. Developers just need to develop a Controller around the available methods, according to their

needs.

The second step is to import the *BI-commons-Plugin* into the View module. This plugin was developed to communicate with previously developed controllers and it is responsible to interact with the end-user at the front-end layer.

As an example of communication between the integrated components, consider the following execution steps when a report is requested to the plugin:

1. Given a report path, the *BI-commons-Plugin* request all the expected parameters from a controller.
2. The controller calls the method in the *BI-commons-Lib* responsible for fetching the report parameters.
3. The library calls the `/repos/pathId/parameter` API end-point and receives an XML object response. The library parses the XML and returns a list to the controller.
4. The controller returns the list to the plugin.
5. The plugin parses the list, creating labels, input fields and available options for all the necessary parameters. They are prompted to the user.
6. The user fills the parameters as he wishes, with an additional 'output-format' parameter that controls the rendering format. User presses 'submit'.
7. The plugin collects all the given input and requests the report with all the parameters from the controller. Parameters are sent in a Map object.
8. The controller calls the method in the *BI-commons-Lib* responsible for fetching a report.
9. The library calls `/repos/pathId/generatedContent` API end-point and receives a BLOB object response. Returns the object to the controller.
10. The controller returns the object to the Plugin.
11. The plugin parses the BLOB file according to the 'output-format'. Displays or downloads the report.



## 5.4 Conclusion

In order to carry out the present work, it was necessary to go through a learning stage on the Pentaho BI platform. This chapter tried to resume the main aspects of the platform, while providing a continuous walkthrough of all the necessary steps to develop content, from the creation to its embedded web visualization.

The Pentaho Report Designer has a steep learning curve, but as familiarization with the tool increased, the process of creating content became smoother. Logically, in order to work with such a refined tool, it is necessary to give up simplicity. Thanks to its properties, it offers the user an enormous versatility in the type of document to create. It contains some features that help a user who does not have database knowledge to develop content. However, if something more advanced is required, knowledge of query language is undoubtedly necessary. But perhaps the most important feature is the use of parameters, and what can be achieved with them. They can be used to filter data, by injecting its value into queries, or to make deep layout changes.

As for the Pentaho Server, thanks to its strong documentation, making changes at the system level is a smooth process. Its API is quite extensive, containing methods for various purposes. For the present work, it was only necessary to work with end-points responsible for the file management.

It is possible to conclude that Pentaho provides the necessary tools so that its application is easily integrated into other solutions. In this particular case, the main objective was the integration of analytical content into an existing solution, with the special characteristic of being the common user to benefit from this component, based on the data themselves generate.



## Chapter 6

# Results and discussion

This section presents the different use cases where the developed solution was implemented. Firstly, the applications where the solution was embedded are described, as well as the methodologies necessary to integrate the solution. Next, the reports or dashboards developed are presented, as well as how they achieve a solution to the existing needs. Afterward, a general analysis of the solution will be made taking into account the challenges encountered in each case. Finally, the limitations of this solution will be discussed.

### 6.1 Use cases

Three case studies will be presented, each one on a different platform. These platforms were developed by BMD Software, a company, founded in 2011, that offers technological solutions for different branches in the field of biomedical informatics. BMD Software produces and maintains solutions for medical imaging and bioinformatics data management. Currently, the company has four main products: PACScenter, CardioBox, Savvy and NeoScreen. The first two products will be used as integration case studies, as well as another product, MDTeam, a not so widespread but equally important solution. All products are modern web applications targeted for healthcare institutions, researchers and hospitals. Their main goal is to provide solutions for better decision-making and knowledge management in healthcare.

### 6.1.1 MDTeam

MDTeam is a patient's information aggregator platform for heart surgery purposes. It allows acquisition, storage and reporting of a patient health information and it is specially designed for heart valve surgery.

A heart valve surgery is a very delicate operation. Before it takes place, there is a reunion where several doctors from different health areas meet to discuss the procedure and exchange inputs and health data such as reports, blood tests, electrocardiograms and magnetic resonance imaging (RMI). MDTeam comes into place to provide a simple web platform where it is possible to upload and store all the exchanged data on a given patient for a heart valve surgery.

On this platform, there was the necessity of providing all the involved doctors with a meeting summary report, where the most important information shared can be quickly organized on an A4 report style.

Figure 6.1 is a diagram that represents part of the MDTeam database schema. The main table (Multidisciplinary) aggregates all information shared on the meeting, on a given *Patient*, but there are several complementary tables with one-to-many relations, such *SurgicalPrior* and *Medication*, or many-to-many relations such as *SpicoStatus* and *Rhythm*. The main goal was to build a report that, given a patient ID, concentrates in an A4 page all the essential information.

The report was developed on Pentaho Report Designer. In order to specifically fetch all data for a given patient, it was necessary to create a parameter named patientID. This mandatory parameter is injected in all database queries, in the Where clause. Then, the report layout was created. On the top layer, all basic patient information is displayed, such as the name, age, hospital and responsible doctor. On the middle layer is displayed the patient prior health history and other crucial information. Lastly, on the bottom layer, blood and heart critical data are presented. On Figure 6.2 it is possible to visualize the first half of the report. A full report was attached as Appendix A. When the report was completed, it was uploaded to Pentaho BI Server.

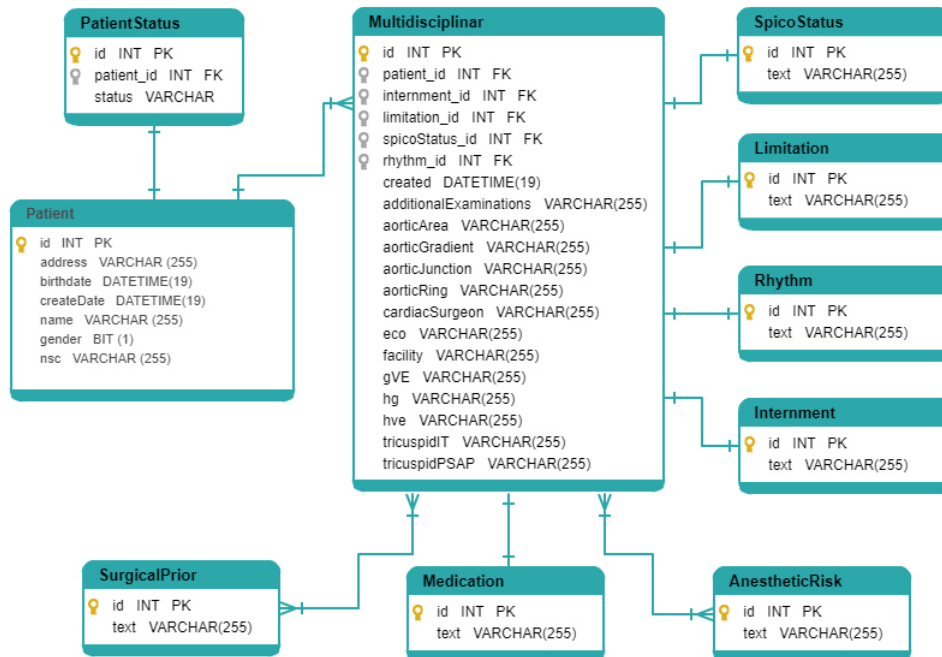


Figure 6.1: Diagram representing part of MDTeam database schema.

MDTeam is a web platform developed in Play, a framework to build web applications with Java or Scala. It is lightweight, stateless and has a web-friendly architecture. A Play application follows the Model-View-Controller (MVC) architectural pattern.

Integrating the developed work into the MDTeam application required two steps, as addressed in the previous chapter. The first step was to import the *BI-commons-Lib* to the server-side of the project and then develop a controller’s methods around this library, such as the authentication and fetching a report. The second step was to integrate the plugin into the client-side and add the necessary JavaScript code to instantiate it.

In this case study, the responsible developer desired to integrate the plugin in such a way that when the end-user, in this case a doctor, was visualizing a patient page, there was a button where he could download the developed report for that particular user.

In order to do that, the plugin execution flow was bound to a ‘download’ button. The patientID, that is the mandatory parameter for the report, was automatically fetched by JavaScript, since the end-user was already in the desirable patient page.

<b>Reunião Multidisciplinar</b> Válvulas Aórticas Trans-catéter		Nome: MARQUES MARIA LUCAS Idade: 49 anos Data de Nascimento: 08/05/1969	
<b>Informação</b>			
Cirurgia Cardíaca Recusada: Sim		Cirurgião Cardíaco: Dr. Miguel Ribeiro Pinto	
Motivo da Recusa Cirúrgica: Elevado Risco Cirúrgico		Centro Hospitalar: C.H.Baixo Vouga	
Via Aérea (Mallampati) Classe II	Antecedentes Anestésicos 13 de Junho de 2012 9 de Novembro de 2015	Risco Anestésico Excitação ou Delírio	Alergias Lactose Carbamazepina
Estado Psicossocial/Neurológico: Autónomo		Escala de Rankin: Category E	
Antecedentes Dislipidemia Obesidade Diabetes		Antecedentes Cirúrgicos Bypasses Patentes Cirurgia Valvular Cirurgia Vascular	
Clinica Sincope DPN Ortopneia	Medicação Hipocoagulação Antiagregação Corticosteróides	Internamento Urgência Etiologia Cardiovascular/Valvular (Nº de episódios no último ano) 1 episódio	
Ritmo: 70 bpm			
Limitação nas actividades da vida diária pela clinica cardíaca: Ligeira			

Figure 6.2: Screenshot of the MDTeam report first half.

### 6.1.2 CardioBox

CardioBox is a centralized ECG – electrocardiogram – management platform. It allows acquisition, storage, reporting and delivery of ECGs, all in a modern and simple web platform. It acts as a centralized electrocardiogram revision platform, where cardiologists have the tools to remotely write, sign, deliver, print and download reports.

With CardioBox it is possible to build a network of institutions that collaborate to provide a better and faster workflow. These institutions are divided into three layers. The first layer is constituted by health organizations. The second layer are facilities, such as hospitals, clinics or diagnostic centers that belong to a given organization. The final layer is constituted by cardiologists that can belong to one or more facilities.

Figure 6.3 represents part of the diagram database structure of the CardioBox platform. Patient's ECGs are generated at the facilities by healthcare professionals, in this context called *provenience*. Then, they are sent to the platform to later be viewed and reported by a

cardiologist, which in this case is called a *user*. Each *ECG* can only contain one *report* written by one cardiologist. In the database logical model, it is possible to recognize the network of institutions, with the tables *Organization*, *Facility*, and *Users*, where *Users* are the doctors.

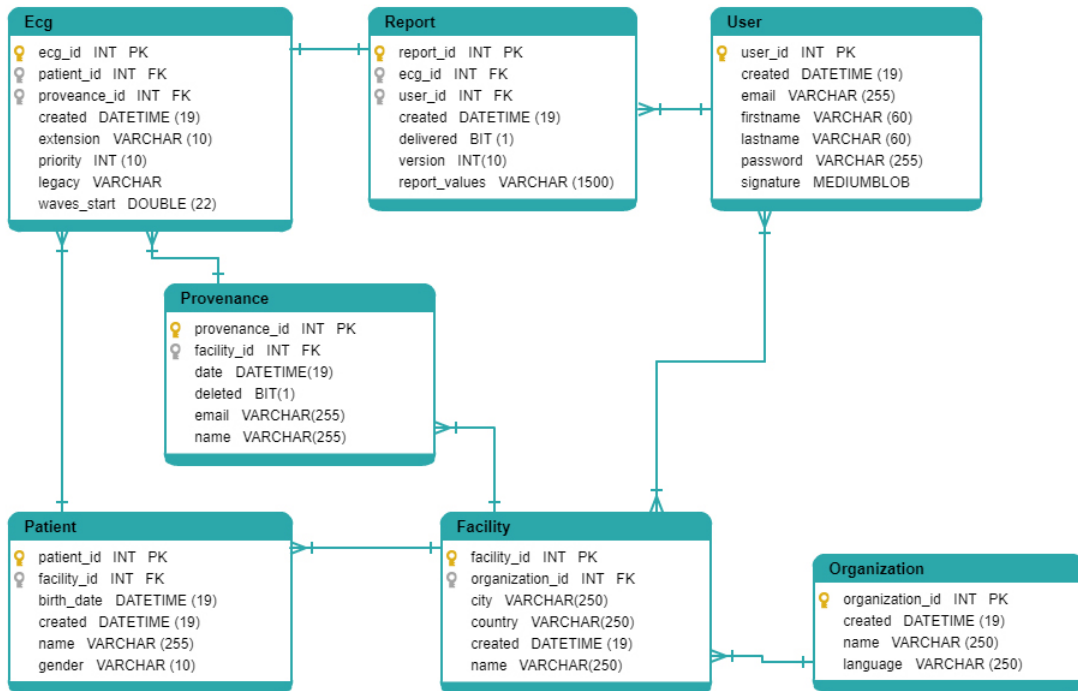


Figure 6.3: Diagram representing part of CardioBox database schema.

Two reports were requested for this platform. Both of them are targeted to an organization level, where it is necessary to provide managers with state-of-art reporting and analytics of the entire sub-system. The reports were developed in the Pentaho Report Designer and later uploaded to the Pentaho BI Server.

In the first report, it was created a view over the number of reports per cardiologist and facility, to a given organization. This view should be periodically and for a given period of time. The development of this report began with the definition of four parameters: a *start date*; an *end date*; a *periodic time*, that could be *Monthly*, *Quarterly* or *Yearly*; and the *report language*. After developing the database queries with the aggregation of the required data, a pivot table visualization was completed. Afterwards, the report was embellished with the addition of a header, where it states the organization name and the report title, and a footer, where CardioBox logotype and the page number were added.

In the second report, it was requested the number of ECGs send per provenance for a

given period of time. Four parameters were created. Two of them were a start and an end date to the data search. Another parameter specifies the desirable facility id and the last parameter was the report language. On the report, a header was created where it states the facility name, as well as the organization to where it belongs. Then, on the report body, the representation of data is given in two forms: a table and a horizontal bar graph.

The integration of the developed solution into CardioBox platform was similar to the MDTeam, as both of them are developed in Play. But on this case study, due to a developed request, it was necessary to implement an upload method on the *BI-commons-Lib*, as the developer wanted to programmatically send the developed reports into the Pentaho BI Server. Figure 6.4 represents a screenshot of the integrated solution onto the CardioBox Platform, connected to a small test database. A full report example with the pivot table approach can be seen in Appendix B.

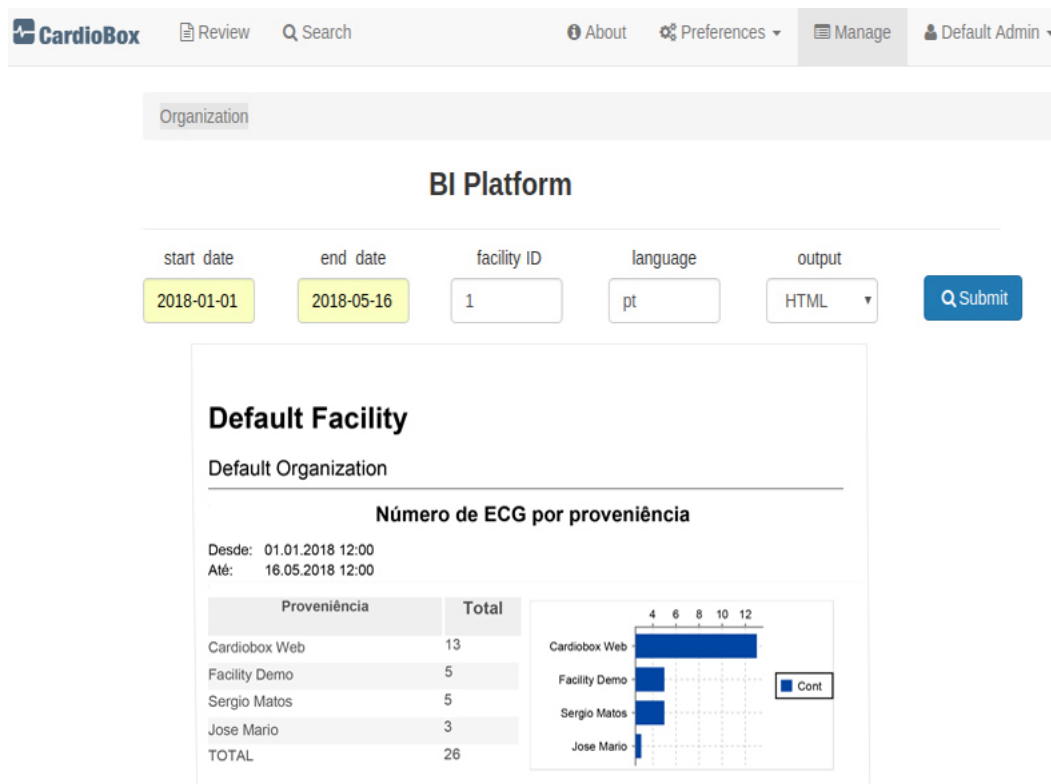


Figure 6.4: Screenshot of CardioBox with the embedded solution.



### 6.1.3 PACScenter

PACScenter is a medical imaging platform for patient studies storage, visualization and sharing. It allows the acquisition and storage of medical images such as cardiovascular, musculoskeletal or ultrasound imaging. Then, with a view and report component, it allows healthcare professionals to perform diagnosis on the collected images, for several imaging centers. It is also possible to collaborate with other physicians by sharing patient studies. All these steps happen in a fully web workstation.

Similar to CardioBox, with PACScenter it is also possible to build a network of institutions and healthcare professionals that collaborate. After facilities collect the medical images, they are submitted in the application for later analysis by a physician.

PACScenter database stores most of the actions that the users perform on the platform. For example, whenever a user requests a report or submits a patient's exam, these actions are stored. Figure 6.5 shows part of the database diagram, where there is a detailed collection of data around one single action, e.g. the browser in which the user is accessing, or the operation performed.

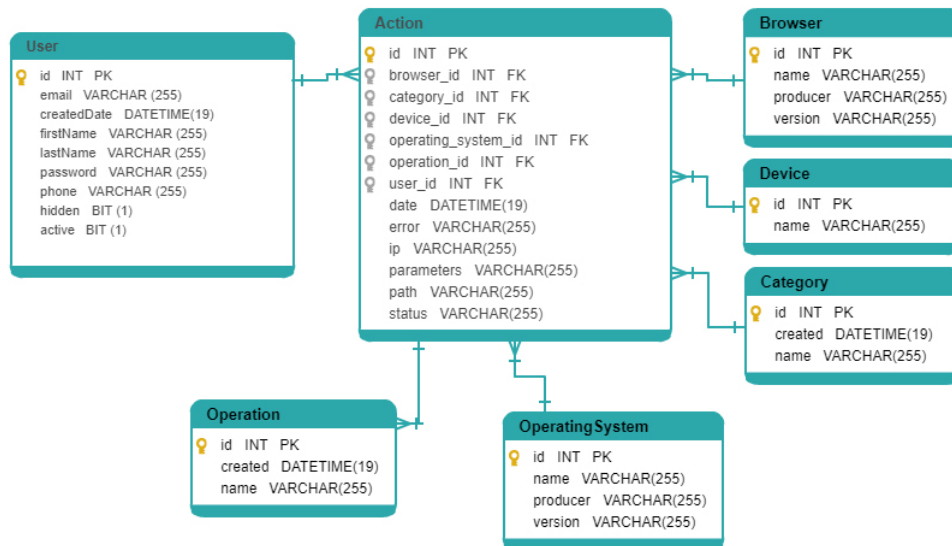


Figure 6.5: Diagram representing part of PACScenter database schema.

On this case study, we also wanted to add an analytical view on the platform where it would be possible to an organization manager to follow the number of reports requested by each User, for a given period of time. The report, developed on Pentaho Report Designer, has

six parameters: a *start date*; an *end date*; the *language*; the *date period*, that can be *Monthly*, *Quarterly* or *Yearly*; a *visualization type*, that can be toggled between *Pivot* or *Table and Graph*; and lastly an *user array*, where the end-user can filter the results by selecting the desirable *Users*. On this particular case study, both the pivot table and the graphs are in the same report. However, only one is shown at a time. This was achieved by injecting the visualization type parameter into the ‘visible’ attribute of each representation. PACScenter is also developed in the Play framework, and so the integration part was similar to the other case studies. All the parameters are displayed to the end-user and the Figure 6.6 represents a final result example, connected to a test database. A full report example with the *Table and Graph* approach can be seen in Appendix C.

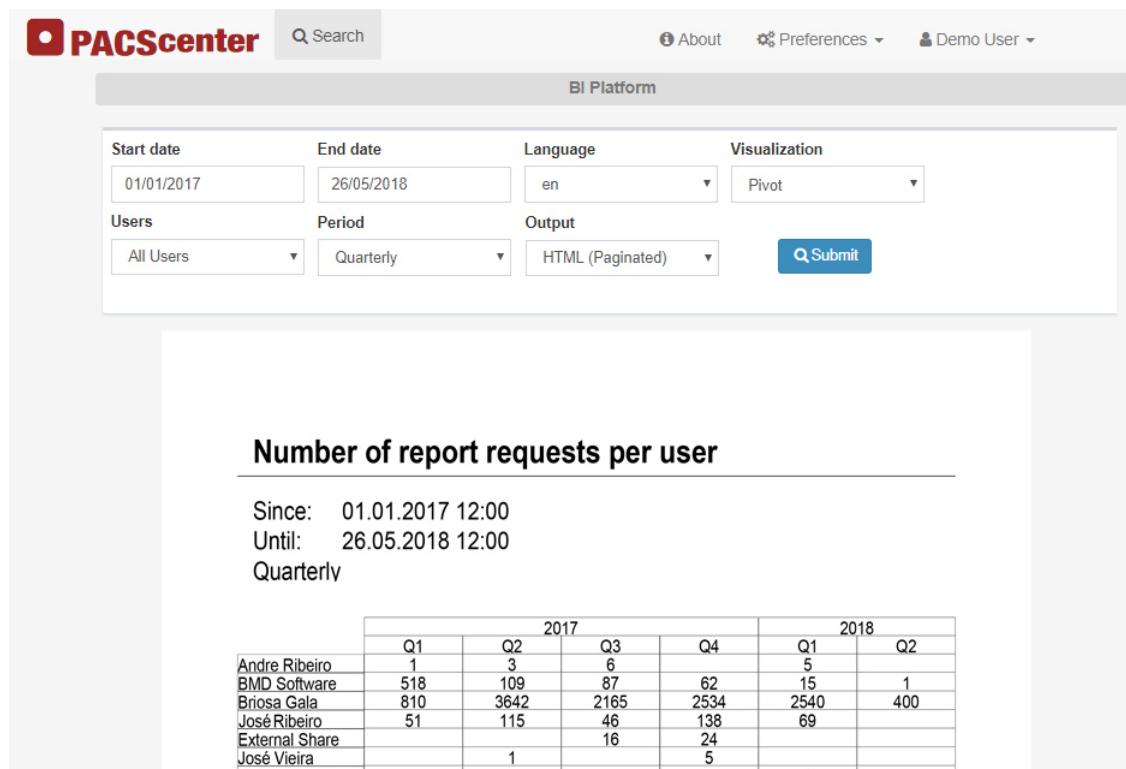


Figure 6.6: Screenshot of PACScenter with embedded solution.

## 6.2 Summary and contributions

With the development of case studies, it was possible not only to test the integration of the developed solution in a real context but also to test the whole logic of Pentaho as the

chosen solution, from the development of reports to its embedded presentation.

By analyzing the integration of the components developed in the existing applications, it can be said that the requirements of compatibility and portability have been fulfilled. The integration process was always easy and clear and the existence of strong documentation as well as an example of integration on a MVC architecture has greatly helped all developers to incorporate the solution. In an overall assessment on the implementation of the solution as a whole, by analyzing the case studies, one could say that Pentaho certainly satisfies all the established requirements across all the implementations, due to a variety of reasons.

First, the reports requested by the developers had a very high degree of detail, which confirmed that the change of platform from Metabase to Pentaho was undoubtedly the most correct, since it would not be possible to match the expectations of developers with Metabase dashboards. It has been demonstrated that Pentaho Report Designer can be very flexible in creating content tailored to the requirement's needs. At MDTeam, a complete structured A4 style report was created, with diverse dynamic fields, that contemplates important clinical information about a patient. While at PACScenter and CardioBox, a more analytic content was generated, making use of the administrative data to create interactive dashboards.

Secondly, in all cases of study, the Pentaho BI Server proved to be very efficient, generating reports almost instantaneous and without failure. These requirements were fundamental as on all platforms developed by BMD there is an important focus on the performance of the solutions.

Finally, the usability component of to the developed solution was fully validated. As different applications have their own interface, developers can easily choose a similar in the integrated solution.

It is possible to conclude that the developed solution offers the possibility of easily implementing a powerful BI engine in an already developed web platform, which complements any application with a strong data analysis component and allows presenting said data to users in a neat and efficient manner.

### **6.3 Limitations**

Some limitations had already been anticipated before the start of the development phase, when Pentaho was being considered, and so the solution tried to get around those problems.

However, after the implementation of the cases of study, a few more problems were raised that can also be seen as limitations.

First off, the *BI-commons-Lib* is fully developed in Java. In order to implement the solution, the server side of the application must be able to compile Java code.

Then, a running instance of the Pentaho BI Server is necessary at any time. This is not an issue by itself, but it can be the source of several minor problems. Unexpected communication problems or intrinsic server problems may arise.

Next, in order to create professional looking content on Pentaho Report Designer, it takes some practice and time invested into learning the details of this tool. This can be seen as a limitation because, in order to integrate this solution, developers need to take some time to build their content, tailored to the user needs.

Finally, a minor limitation is the fact that the graphs generated on PRD are old-fashioned. As there is more and more effort in creating attractive websites with modern visualizations, integrating a solution that does not follow the same parameters can be an obstacle.

## Chapter 7

# Conclusion

The main goal of this thesis was the study, evaluation and implementation of a BI solution that would enhance an already developed clinical web solution by delivering state-of-art reporting and analytics to the final user.

The work began with a horizontal analysis of all existing open-source solutions in an attempt to gain a better understanding of the possibilities and limits of each one. It was concluded that no platform analyzed could meet all the demands and therefore it would be necessary to choose a solution and complement it, by developing additional components around the solution.

In order to make a reasoned choice of the platform that was going to be adopted, a requirements gathering was first made. At this stage, the functional and non-functional requirements that the final platform would have to contemplate were pragmatically addressed.

Supported by the requirements analyzed, the Metabase solution was chosen. Firstly, a more in-depth analysis of the platform was carried out, where its architecture, code and flow of actions were studied. Subsequently, several modifications were implemented in an attempt to make this solution better fit the established requirements. However, during this process, a number of problems began to emerge and even though Metabase seemed like a very promising approach, the problems that were faced by tailoring this solution to fit the requirements were heavy enough to discard this solution.

After the first approach failed, the state-of-art was revised and Pentaho was chosen as the next solution to be adopted. It proved to be a robust and efficient solution. However, despite having some functionalities that Metabase did not have, it lacked an important feature,

namely the possibility to integrate generated content into third-party web applications.

To complement this requirement, the development part of this work focused on the elaboration of a mechanism that would allow the integration of the content generated in Pentaho into the frontend of a web application.

The work developed is made up of two parts. The first part is a Java library, called *BI-commons-Lib* that communicates with the Pentaho Server API, acting as a client library. Several methods have been implemented that facilitate communication with the server. With this library, it becomes possible to request parameterizable reports and dashboards or browse in the Pentaho Server repository. It is intended to be integrated into the server side of an application, however, it can be used for several other purposes. It was developed with the maximum of abstraction so that if it is necessary to change the Business Intelligence solution, the integration process happens smoothly and without compromising the solutions that depend on it.

The second part focused on the development of a JQuery plugin, called *BI-commons-Plugin*. The purpose of this plugin is to provide mechanisms that enables the end-user to manipulate the contents of a report or dashboard, as well as its rendering or download. The plugin does this by prompting a user interface to the user with the parameters fields that need input, as designed by the report builder.

In order to evaluate the applicability of the whole solution, three case studies were developed where the entire application was integrated into existing solutions. Each case study had its own particularities, and the content developed for each case was quite different. However, the solution developed proved to be able to respond to all the individual needs. In an overall assessment of the implementation of all case studies, both Pentaho and the developed work presented substantial results in terms of performance, reliability and usability.

One of the biggest problems that was anticipated in this work was the applicability of Business Intelligence tools to clinical data. As it was possible to observe in the case studies, health organizations require tools that can manipulate both clinical data, such as MDTeam, and administrative data, such as CardioBox and PACScenter. The chosen platform, as well as the developed work, proved to be transversal to all type of data, providing solutions that met the requirements.

With the present work, it was also possible to study the feasibility of integrating an existing

BI community solution. Oftentimes, software companies are faced with the doubt whether to develop a BI solution in-company from scratch, or to acquire an existing one. With all the work done, it can be said that free Business Intelligence solutions that exist, at the moment, in the market are excellent tools with multiple purposes and that, very hardly will compensate the work and the money invested into developing a solution from scratch.

In conclusion, the present work can be seen as a small increment in the domain of embedded Business Intelligence tools into existing applications. Rather than using a dashboard or reporting software for purposes of internal usage as a stand-alone tool, in this work it was proved that it is possible to integrate advanced analytics and reporting into already developed products, providing the clients with a powerful but simplistic data visualization experience. More precisely, it has been demonstrated that it is possible to enrich a medical application by integrating a free, secure and efficient solution that uses the data healthcare providers generate and translates it into visualizations that can be interpreted and used in decision-making.

## 7.1 Future work

In this work, the bases of a channel that allows the user to obtain the requested document were created, by developing file management methods around the Pentaho Server API endpoints. In future work, it would be interesting to strengthen this channel by embedding more Business Intelligence server functionality into the application interface, to the point of giving to the end-user a true Business Intelligence experience through mechanisms as easy as those developed. These functionalities can be:

- Simple interface for ad-hoc queries;
- Email automation;
- Data sources management;
- User role management;
- Reactive, proactive and predictive alerting.

The recent growth in embedded BI solutions will not slow down. Providing the users with tools that they can use to explore and visualize data that themselves create in a software, not only benefits their experience but also makes the software much more valuable.

Even though there are excellent dashboards or report solutions on the market, having to alternate between solutions to produce visualizations and data insights it is not feasible for typical users. It is easier for them to work on the data that is generated inside the same application rather than to buy and learn a new tool.

For all the mentioned reasons, and supported by the developed work, it is possible to perceive that future work on this field will be the development of more modular and embeddable BI features that can be easily integrated into applications, accordingly to the developer's needs.



# Bibliography

- [1] Patti Brooks, Omar El-Gayar, and Surendra Sarnikar. Towards a business intelligence maturity model for healthcare. *Proceedings of the Annual Hawaii International Conference on System Sciences*, pages 3807–3816, 2013.
- [2] Hamzah Osop and Tony Sahama. Electronic Health Records : Improvement to Healthcare Decision-making. *IEEE 18th International Conference on e-Health Networking, Applications and Services*, 2016.
- [3] Andre W. Kushniruk. Analysis of Complex Decision-Making Processes in Health Care: Cognitive Approaches to Health Informatics. *Journal of Biomedical Informatics*, 34(5):365–376, 2001.
- [4] Ally Selemani and Nawaz Khan. Data Warehouse and BI to Catalize Information Use in Health Sector for Decision Making : A Case Study. *International Conference on Computational Science*, pages 92–97, 2016.
- [5] Tobias Mettler and Vivian Vimarlund. Understanding business intelligence in the context of healthcare. *Health Informatics Journal*, 15(3):254–264, 2009.
- [6] Kamran Parsaye and Mark Chignell. *Intelligent Database Tools and Applications: Hyperinformation Access, Data Quality, Visualization, Automatic Discovery*. 1993.
- [7] Mads Soegaard and Rikke Friis Dam. The Encyclopedia of Human-Computer Interaction, 2nd Ed. <https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed>. (Accessed on 2017-11-03).

- [8] Andrew McAfee and Erik Brynjolfsson. What Makes a Company Good at IT? - WSJ. <https://www.wsj.com/articles/SB10001424052748704547804576260781324726782>, 2011. (Accessed on 2017-11-03).
- [9] Jianwen Su and Yan Tang. Business Intelligence Revisited. *2017 Fifth International Conference on Advanced Cloud and Big Data (CBD)*, pages 1–6, 2017.
- [10] Hsinchun Chen and Veda C Storey. Business Intelligence and analytics: From Big Data To Big Impact. 36(4):1165–1188, 2018.
- [11] Martin Dawes, William Summerskill, Paul Glasziou, Antonino Cartabellotta, Janet Martin, Kevork Hopayian, Franz Porzolt, Amanda Burls, and James Osborne. Sicily statement on evidence-based practice. *BMC Medical Education*, 5(1):1, dec 2005.
- [12] T. Greenhalgh, J. Howick, and N. Maskrey. Evidence based medicine: a movement in crisis? *BMJ*, 348(jun13 4):g3725–g3725, jun 2014.
- [13] David Avison and Terry Young. Time to rethink health care and ICT? *Communications of the ACM*, 50(6):69–74, jun 2007.
- [14] ET Chen. Implementation Issues of Enterprise data Warehousing and Business Intelligence in the Healthcare Industry. *Communications of the IIMA*, 12(2):39–50, 2012.
- [15] Shahidul Islam Khan and Abu Sayed Md. Latiful Hoque. An analysis of the problems for Health Data integration in Bangladesh. In *2016 International Conference on Innovations in Science, Engineering and Technology (ICISSET)*, pages 1–4. IEEE, oct 2016.
- [16] Wilfred Bonney. Applicability of Business Intelligence in Electronic Health Record. *Procedia - Social and Behavioral Sciences*, 73:257–262, feb 2013.
- [17] ISO/TR 20514:2005 - Health informatics – Electronic health record – Definition, scope and context. <https://www.iso.org/standard/39525.html>. (Accessed on 2017-11-03).
- [18] K HAYRINEN, K SARANTO, and P NYKANEN. Definition, structure, content, use and impacts of electronic health records: A review of the research literature. *International Journal of Medical Informatics*, 77(5):291–304, may 2008.

- [19] Sheen S. Levine and Michael J. Prietula. Open Collaboration for Innovation: Principles and Performance. *Organization Science*, 25(5):1414–1433, oct 2014.
- [20] The Open Source Definition (Annotated) | Open Source Initiative. <https://opensource.org/osd-annotated>. (Accessed on 2017-11-11).
- [21] Sinfic SA. <http://www.sinfic.pt/SinficWeb/displayconteudo.do?numero=24957>. (Accessed on 2017-11-12).
- [22] Arnoud Engelfriet. Choosing an open source license. *IEEE Software*, 27(1):48–49, 2010.
- [23] Georgia M Kapitsaki and Georgia Charalambous. Find your Open Source License now ! pages 1–8, 2016.
- [24] Juho Lindman, Anna Paajanen, and Matti Rossi. Choosing an open source software license in commercial context: A managerial perspective. *Proceedings - 36th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2010*, pages 237–244, 2010.
- [25] Roland Bouman and Jos Van Dongen. *Pentaho Solutions - Business Intelligence and Data Warehousing with Pentaho and MySQL*. 1 edition, 2009.



# Appendix A: MDTeam Report

**Reunião Multidisciplinar**  
Válvulas Aórticas Trans-catéter

**Nome:** MARQUES MARIA LUCAS

**Idade:** 49 anos

**Data de Nascimento:** 08/05/1969

Informação			
<b>Cirurgia Cardíaca Recusada:</b> Sim		<b>Cirurgião Cardíaco:</b> Dr. Miguel Ribeiro Pinto	
<b>Motivo da Recusa Cirúrgica:</b> Elevado Risco Cirúrgico		<b>Centro Hospitalar:</b> C.H.Baixo Vouga	
<b>Via Aérea (Mallampati)</b> Classe II	<b>Antecedentes Anestésicos</b> 13 de Junho de 2012 9 de Novembro de 2015	<b>Risco Anestésico</b> Exitação ou Delírio	<b>Alergias</b> Lactose Carbamazepina
<b>Estado Psicossocial/Neurológico:</b> Autónomo		<b>Escala de Rankin:</b> Category E	
<b>Antecedentes</b> Dislipidemia Obesidade Diabetes		<b>Antecedentes Cirúrgicos</b> Bypasses Patentes Cirurgia Valvular Cirurgia Vasculuar	
<b>Clínica</b> Síncope DPN Ortopneia		<b>Medicação</b> Hipocoagulação Antiagregação Corticosteróides	<b>Internamento Urgência</b> Etiologia Cardiovascular/Valvular (Nº de episódios no último ano) 1 episódio
<b>Ritmo:</b> 70 bpm			
<b>Limitação nas actividades da vida diária pela clínica cardíaca:</b> Ligeira			
<b>Informação clínica adicional:</b> Sem informação adicional.			

Exames Complementares	Lab: Cr:	Hg: 42,1	Plq: 68,3	Outros: Sem informação.
Ecocardiograma: TT Massa VE(g): 4,6 <b>Válvula Aórtica</b> Gradiente 3,5 mm/Hg Área 2,3 cm <sup>2</sup> Anel Ao 1,3 mm Junção ST 0.9 mm	Data: 01/2018 HVE: 7,0 <b>Válvula Mitral</b> IM 64,3 /IV <b>Válvula Tricúspide</b> IT 2,3/IV PSAP:5,3 FEVE: 5,5 %	<b>Caterismo Coronário</b> Lesão crítica de 3 vasos	<b>Angio-TAC</b> Femorais OK Axilares OK Ilíacas/Aorta OK Aorta ascendente em Porcelana	
<b>Outros Exames:</b> Electrocardiograma: 4045628				

**Note:** simulated data.

1

# Appendix B: CardioBox Report

## Default Organization

---

### Número de ECG por instalação e proveniência

Desde: 12.06.2014 12:00

Até: 28.06.2016 12:00

**Mensalmente**

Facility	Provenance	2014	2016				
		Agosto	Janeiro	Fevereiro	Março	Julho	Agosto
Default Facility	Cardiobox Web	1		1	1	6	2
	Facility Demo		2		1		
	Jose Mario						3
	Sergio Matos		3		2		
Default Facility3	Luis Lemos					5	
Facility Default	Jose Mario					4	1

**Note:** simulated data.



BMD Software

1 / 1

# Appendix C: PACScenter Report

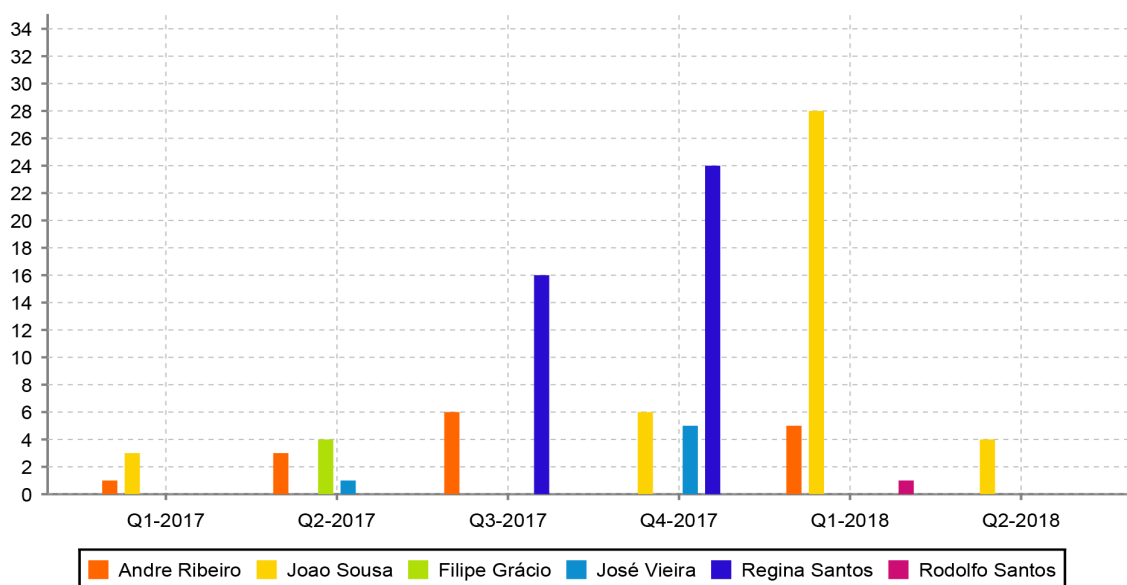
## Número de relatórios pedidos por utilizador

Desde: 01.01.2017 12:00

Até: 23.06.2018 12:00

Trimestralmente

Utilizador	Total
Joao Sousa	41
Regina Santos	40
Andre Ribeiro	15
José Vieira	6
Filipe Grácio	4
Rodolfo Santos	1



**Note:** simulated data.



BMD Software

