1 Introduction

This project took place for my last year internship at TELECOM Nancy to get my Master's Degree in Computer Science. It was directed by a small start-up company from Wisconsin, Prime Resources.

The main goal of the internship was to apply in a specific project all the skills and knowledge I learned during my training as a graduate student. Though challenging, it helped me develop new skills as a software developer, and also helped me to improve my analysis and conception skills.

The task that was given to me was to develop a Configuration Management Database (more commonly referred to as CMDB in this report) application aimed at small and medium-sized companies. The objective was mainly to write something simple, user-friendly, and efficient in terms of rendering, based on the notion of Minimally Viable Product (MVP), meaning that the development is focused on the main functionalities of the application.

The goal was to create a CMDB that users would want to use. Without competing with major CMDBs available on the market such as OneCMDB, our application was aimed for people who would like to access quickly to the different dependencies of their system, and who would feel the need to use it without spending too much time on it. Thus, the software was designed as a cloud application for an easy access, whether on a phone, tablet, or PC. The advantage of not having to download and install something on the computer or phone was essential for us, as it improved the attractiveness of the product and the time that a user would spend to configure the application.

A CMDB consists mainly of listing all the different resources that a company has (servers, databases...) in order for a specific user to know which of these resources are used by a specific department or network. It is really useful when an administrator would like to upgrade one of his servers, but first would want to make sure that the different dependencies linked to the server would not create a version problem for a specific application linked to that server. Thus, it enables not only administrators, but every single user to know exactly how the company resources are implemented, and to have an idea of which parts would be impacted by a specific change.

The problem about CMDBs though, is that it can soon become too heavy to keep up-to-date. For example, big retail companies like Kohl's or Macy's require heavy CMDB software in order to support their vast network spread across the country. It soon becomes a big charge to make sure the CMDB is up-to-date and running with the latest versions of the different components of the company. Administrators do not always have time to update the software, and users can get discouraged to use it since it does not really reflect the state of the company at the present time.

CMDBs can then become neglected, which is a financial loss since thousands of dollars are usually invested in getting a license. Obviously, small and medium-sized companies would have no interest in investing such a big amount of money in products that are complicated to use and that would require too much time to keep up-to-date.

Although CMDBs on the market are apparently complex, the idea of knowing exactly what is going on in the company with all the different resources is an important, if not necessary, measure every company should strive to have. We believe that such companies are looking for a simple and easy way to manage their network, without having to spend too much time learning how to do it and actually doing it.

That is where the subject of my internship comes up. My objective was to write a CMDB specifically aimed at such companies. It should be easy to use by any employees, no matter what their position could be, and easily maintainable as well.

In order to present the subject in more detail, I am going to first introduce the company in which I did my internship, Prime Resources. I ~~will~~ then describe the project in more detail. Then, we will take a look at the different solutions available on the market. Next, I ~~will~~ analyze the problem and describe the solution that was implemented before exposing the different technologies used to develop the product. We will then study in more detail the implementation of the solution before talking about the economic strategy behind it. Finally, after describing the future potential of the application, I will expose the different work and management methods used during the project.

~~Now, I will start in the following part by introducing the company in which I did my internship.~~

Par nécessaire !

Eviter le futur !

I or we :

You have to choose !

2 Presentation of the company

In this part, I will present the company in which I had the opportunity to do my internship.

Created in 2014, Prime Resources LLC is a start-up company focusing on providing top-notch services to other companies. Its main goal is to recruit people having the skills necessary to answer one of the clients' most challenging problems. The different domains of expertise of the company do not only include information technology but also manufacturing, executive search, and engineering. [1]

Although the company was created very recently, its story is interesting to tell as it illustrates the spirit of American entrepreneurship that I was able to experience a lot during my internship. In 2014, John and Karen Houdek, residents of Sussex, WI, created their own company, Prime Resources. Owned by Mrs. Houdek and led by her husband, it was their desire, after spending 25 years in the industry leadership, to run their own business. The company has grown since then and answers the challenges of many other businesses who come to Prime Resources in order to find the best professionals available to solve their problems.

The company is registered as an LLC (Limited Liability Company). According to the IRS (Internal Revenue Service), an LLC company is "a business structure allowed by state statute. Each state may use different regulations, and you should check with your state if you are interested in starting a Limited Liability Company." [2]

This legal structure is the most convenient to adopt for an entrepreneur as the registration process at state level is fairly quick and does not include a lot of constraints. An owner of an LLC is called a member, and there is no limit on the number of members that can be included in an LLC. Actually, a single person can also be the sole member of his LLC, and different entities (such as other companies) can be members as well.

The type of activities a company can do as an LLC usually depends on state law, but most of the time bank and insurance companies cannot claim this status. On the other hand, this business structure is very popular for IT services start-ups.

Prime Resources recently moved its headquarters, originally located in the small town of Hartland, to the nearby city of Sussex. It is located on Main Street, right in the heart of the city.

Regarding the different work methods, Prime Resources will usually place their employees at a client's location. Most of the time, people will not work from the headquarters but from each client's site. In my case, it was a little different. Since I was working on a project that was not for a specific client but for a later release on the market as a software solution, I had the opportunity to work from different locations throughout my internship.

Although I worked from the headquarters, I also had the opportunity to work from a business lounge located in downtown Milwaukee: the Hudson Business Lounge. The Hudson is a meeting, work, and event center located in the heart of Milwaukee's Third Ward, one of the oldest parts of the city. The purpose of this location is to provide a work environment for start-ups, as well as a meet-up area with other like-minded professionals. Offering over 11,000 square feet of office space, it is a futurist concept that gives people the opportunity to grow their network, either by meeting new clients or potential employers/employees.[3]

I was able to meet new people working on different IT projects. Some were just developers like me who worked on a specific project for their company, and others were managers and recruiters who hold conferences and events in order to broaden people's minds in the business world.

Working from this location influenced me deeply. It helped me better understand the world of start-up companies and the spirit of entrepreneurship that is so specific to the United States. It was also Prime Resources' goal for me to meet other entrepreneurs, as it gave me an opportunity to discuss my project and get ideas for the development.

~~I will now present the different objectives of this project, along with the problems I had to solve.~~

Présentation de l'entreprise très succincte !

- Peut inclure plus d'info. Par exemple l'organigramme, une carte pour situer l'entreprise, ...

3 Description of the project

The goal of my internship at Prime Resources was to develop a simple version of a Configuration Management Database, commonly referred to as CMDB. A CMDB is a software that allows businesses to keep track of their IT infrastructure to know exactly what resources are available to the company, whether it be hardware like servers, software like applications, or even human resources like the number of database administrators.

In today's market, companies can use a CMDB in order to keep their different resources up-to-date. While not all businesses can invest time and money in such tools, most big companies like Kohl's, General Motors, or Walmart will usually invest in a CMDB so they can have a better view of all resources available in each department of the company. Although this practice is well-defined in companies' strategic development, the use of CMDBs can rapidly become overwhelming and much too time-consuming, thus making employees less inclined to update them, or even to use them at all.

Experience proved to my manager, currently a vice-president at Kohl's Corporate, that people never use the CMDBs because of their complexity and their difficulty of access. A first problem is the time required to learn how to use a CMDB. Although different systems are available on the market, they all require some training to know how to research or update a specific item in the database. Most of the time, employees only need to know about one specific function, but the complexity of the system makes it impossible to speed up the learning process or to specialize the training in a specific area of the software.

A second problem that comes to mind (and actually results from the first one) is the inaccuracies caused by the data in the CMDB. When the structure of the company changes (for example when a new device is added to the network or when a specific department is reworked), it becomes necessary to take into account these changes into the CMDB. They need to be registered in the software by either an administrator or a qualified employee who owns the rights to modify the data according to his own department.

Those changes being frequent, it can be fairly hard to keep track of them, which ultimately leads to some inaccuracies in the CMDB. Employees will then not feel they can rely on a database that could potentially be out-of-date. Thus CMDBs become seldom updated and used.

Although bigger companies can find time and money to invest in getting a CMDB and training some employees to keep it updated, small- and medium-sized companies do not have this luxury. They usually have to use simpler methods to access the different resources of their company. The principles of CMDBs could however be applied to any company, the size not being a decisive factor for a CMDB to be efficient. It is clear, though, that with the current products available on the market, those companies would not be likely to invest time and money in purchasing a license for one of these applications.

This is where the project takes its place. What if there was a simple CMDB on the market that would meet all the requirements regarding what a small- or medium-sized company could expect from a CMDB? Such an application should, of course, be simple to use and very intuitive, while enjoying the perks of having a top-notch interface and using some of the most recent technologies available on the market. This could be easily implemented if the conception and development always kept in mind the following aspect: **Minimally Viable Product (MVP)** *(see glossary)*.

My task was then to develop a software that could answer most flaws described earlier. The application would be divided into three parts:

- Configuration items listing
- Applications listing
- Data center mapping

The first part would show the list of all configuration items from the company. A configuration item, as we will see later in the state of the art, is either a virtual element in the company like a server, a database, or a virtual storage service. The application should give the user the possibility to view all configuration items in the company, as well as their properties. An administrator should also be able to add new items or properties to existing configuration items. Moreover, the interface should display these items in a hierarchical tree.

The second listing should present the different applications used in the company. Each application can reference one or multiple configuration items. Thus, in this listing, there will be two parts: first, a file explorer tree that shows the hierarchy of the applications grouped into folders; second, a graph that will list all configuration items used by a specific application. As with the previous part, there will be two views: user and administrator. The user should be able to access all applications and visualize the tree. The administrator should be able to organize the different applications into folders and be able to rename, add, or delete applications. He also can add configuration items to a specific application and reorganize them into different folders.

The third part is the most ambitious in the whole project. It should present the company's different data-centers. As with the two previous points, a hierarchical tree will list all the data centers. After clicking on a specific node, the user will access a map of the data center that shows the position of all the elements included in the room. If the user is an administrator, they can modify the map by editing the room's characteristics (such as the number of rows and columns and the position of the different servers). They can also add or delete cabinets. Now, when a user clicks on a cabinet, they will be redirected to a closer view that will show them the different configuration items (servers) present on that cabinet. By clicking on it, the user will be redirected to the properties of that configuration item, as shown in the first part. It will also allow the user to see the path to all the applications that are using this configuration item.

The application will also feature other functionalities like a login and sign up page allowing users to access the application and register. A user will be able to register either as a user or administrator. This distinction will determine if the user logged in is allowed to make changes to the application.

Another feature included will be an information page showing the different settings for the connected user. The settings included will be the username, the email address, the administrator status, as well as the possibility to change the current password.

Now that we have seen the problem the application is trying to solve, as well as the different requirements that it should have, we will focus on the work that was done before regarding CMDBs as well as the different software available on the market.

State of the Art

Maj : remarque valable pour tous les titres de sections

4 State of the art

Configuration Management Databases, commonly referred to as CMDBs, are part of the Information Technology Infrastructure Library framework, also called ITIL. Before describing what the purpose of a CMDB is, it is important to understand what ITIL is and what its specifications are in order to better understand the big picture.

4.1 Information Technology Infrastructure Library (ITIL)

The Information Technology Infrastructure Library framework, more commonly referred to as ITIL framework, consists of different tools and practices to "help individuals and organizations use IT to realize business change, transformation, and growth." [4] More specifically, it focuses on providing different requirements in order to help IT services to focus on business needs.

ITIL consists of a succession of procedures and tasks to be completed by the IT department of each company in order to master the model described in a set of books. The first requirements of this kind appeared in the 1980s in the United Kingdom, before being released into a major version in 2000 as ITIL v2. The current version in use as today is ITIL v3, with an update which came in July 2011.

The advantage of following this model is to help companies achieve their goals in a more efficient and less costly way, as well as to grow and develop the size and budget of the company over the years. To achieve this goal, five key points are incorporated into the ITIL specifications:

- Strategy management for IT services
- Service portfolio management
- Financial management for IT services
- Demand management
- Business relationship management

Although understanding the specific role of each process is not necessary in the continuation of this report, it is nonetheless interesting to note where CMDBs are positioned in this larger scale.

The IT services strategy management includes two core points: service support and service delivery. CMDBs are part of the first one. This first service focuses on the user; it makes sure that they, whether a final consumer or a developer in the company, can have a positive experience and

will be able to ask information, request update, or notify of an issue. The CMDB should be able to process any request and, if it fails, notify a third party which will analyze the problem and likely propose a solution.

4.2 Configuration Management Database (CMDB)

Part of the ITIL process chain, a Configuration Management Database, is, as the name suggests, a database containing the references of the different assets of a company (servers, databases, networks, applications...). Each single item in this database is referred to as a configuration item. A CMDB's task is to list all these assets along with their properties, thus allowing any user in the company to know which assets are available in a specific department.

Most of the time, there are two different views in a CMDB: administrator and user. The first one must be able to manage the different items, either globally or in a specific department. That includes being able to add different configuration items in the database when the company is purchasing new hardware, but also being able to update properties on specific objects. An administrator will also grant permissions of access to certain types of users according to the department in which they work. A global administrator can also be used to manage the different administrators at each department's level.

A user, on the contrary, is only interested in viewing the contents of the database, and especially in knowing specific properties relative to his department. Most CMDBs will then grant a view permission only to certain types of items related to the user's department.

Having these two main roles in mind, it is important to note the different interests motivated by using a CMDB. On one hand, we have a user only interested in knowing certain properties on a specific configuration item. The CMDB must provide a very fast and intuitive interface so the search can be done quickly. On the other hand, there is an administrator that needs to frequently update the database and make sure each update will not have a negative impact on another configuration item. A CMDB implementation needs then to make sure each of these specifications is respected.

One might wonder what should be included in a CMDB. The answer to this question is obviously both large and subjective. Most of the time, the configuration items will be virtual objects such as a database instance, a server host, a network, a virtual storage instance, etc. Basically, it can be pretty much everything included in the company's IT infrastructure. The different human resources can be added to this as well.

Visually, the structure of a CMDB uses trees and graphs to show the different interactions between the components. For example, if an administrator wants to change a setting on a certain server, the CMDB gives them the possibility to see the different configuration items that would be impacted

by this change. Visually, this solution is most of the time rendered under a tree or graph to be read in an easier way by the user.

The ITIL specifications describe the four major tasks that a CMDB must fulfill:

- Identification of configuration items to be included in the CMDB
- Control of data to ensure that it can only be changed by authorized individuals
- Status maintenance, which involves ensuring that the current status of any configuration item is consistently recorded and being kept up to date
- Verification, through audits and reviews of the data to ensure that it is accurate. [5]

Today, the market presents different implementations for CMDBs, both in private and public domain. Although CMDBs were initially being developed as commercial products requiring licensing, some open-source CMDBs started to appear on the market by the mid-2000s; the main reason explaining this phenomena was the desire for some smaller companies to customize the CMDB to their needs, a difficult and even sometimes impossible task to achieve without the help of a consultant from the developing company on commercial products. Although not being as complete as commercial applications, open-source CMDBs give certain advantages to users. Below is a description of two of the most popular open-source CMDBs. [6]

The first one, Itop, is developed by Combodo, a French company. It offers a full customizable CMDB, along with a service desk, allowing users to create and access IT products in their company. The software also includes an error and incident management system allowing the administrator to keep track of the different errors in the system.

Another popular open-source CMDB is OneCMDB. Aimed at small- and medium-sized businesses, it was developed in 2006 by Lokomo Systems AB, a company based in Sweden. The version currently available is a stand-alone edition coming with source code, and released under a GNU General Public License. The code is accessible on Source Forge and requires Java installed on the system in order to function. [7]

Although those two CMDBs are free to use and more easily accessible for small companies, they lack the ability to manage bigger flows of data that bigger companies require. In early 2015, Capterra, a website dedicated to helping companies find the best business software available on the market, published an article entitled "Top ITSM Software Products". For information, ITSM, or IT Service Management, refers to all the procedures in a company to manage the IT department. ITIL is part of these procedures. [8]

The article pointed to Freshservice as the most popular solution used by companies. Developed by Freshdesk, a company based in San Francisco, it features a whole helpdesk system along with the

"Première vue CMDB satisfont 5 critères avancés : n'est pas question de "big companies" parmi vos objectifs"

⇒ Il faut rajouter plus d'éléments (cloud-based sol...) pour motiver et démontrer l'apport de la solution même si elle n'est pas encore présentée.

basics of CMDBs described earlier. This solution is used by a number of big companies, such as Sony or Honda. As with the other commercial software, the annual licensing can get very expensive depending on the companies' needs.

There are many other solutions on the market. Some are being developed for an internal use only, meaning that the company that is developing it will be the one to use it. That solution can be a nice alternative to buying a license to classic CMDB provider, as it will be created to fit the company's needs. However, it is first important to show how the CMDB can be a real advantage to the company, as it will take time and money to develop it.

In brief, there are a lot of different CMDBs on the market to choose from, but most of them have the same problems: they are designed for big companies who can afford a high annual licensing cost, and they require a special training to use it as it takes time to ensure that everything is kept up to date. Thus, the complexity of the system can discourage users to take advantage of it, sometimes leading to cases where the CMDB is not kept up-to-date, or is not even used at all.

All of these different problems warrant a reconsideration of how to use and/or develop CMDBs. The product that I developed during my internship comes as an alternative solution in order to prove that a CMDB can be both simple and useful to use. I will now describe the solution that I created.

5 Problem analysis and the solution: a cloud-based CMDB

In this part, I will describe the different steps of the development of the cloud-based CMDB.

During the first days of my internship, my manager explained to me the main flaws of CMDBs. As a vice president at Kohl's, he understood the importance of having CMDBs in the company, both from a corporate and user point of view. But the reality of the market makes it harder to find a tool that is affordable, maintainable, and easy to use, both for administrators and for users. To him, the most important point of a CMDB is summed up with three key words: Minimally Viable Product. This notion of MVP is very important to understand for the rest of this report, and was the main thing to remember as I developed the application.

Basically, a minimally viable product is, as the name suggests, a product that is viable so it can do what it is supposed to in an efficient way with as few flaws as possible, but at the same time a product that is conceived in pure simplicity, or simple enough that a user should not need a manual or an explanatory note to know what to do with it. These notions should apply both to the final product, i.e. what the user will see, and the code, so that other developers who come after can understand precisely what was done. I now describe what this implies for both point of views.

5.1 From a user point of view

For a user, as was previously written, the solution needs to be user-friendly. This goes from the main functionalities of the product to the simplest and tiniest details such as where the option bar is located or how a user signs up in the application. While developing an application that can fulfill these requirements, there are five main points to keep in mind. [9] According to Walker Fenton, CEO of Sepia Labs, the company behind the professional social network Glassboard, these are:

- **Understanding the context of the users that will use the application**

  This point may seem obvious, but too often in the corporate world engineers and developers conceive applications and keep adding features to them while losing the view of what the application was originally intended for. It is really important to ask the following questions: what is the user expecting from the application? In which circumstances will he use the application? What main functionalities is the user expecting? With this in mind, it becomes easier to target the different options to include in the application and the ones to leave along.

- **Having a clean presentation**

  The size of the screen on which the application will run does not really matter, what matters is how to visually organize the different graphic elements of the application so that the

The reason is that this cloud-based CMDB is not designed for a specific company; actually, the product was not the request of a client, or even a group of clients. It came up as an idea and, like every idea in the business world, needed to be verified.

Selling a MVP is more about selling a concept or a prototype than a real product. The product, though conceived for users, will not be sold to users, but to investors, IT managers, and other businessmen who would welcome a solution to help them be more productive in their company. Steve Blank, a manager who specialized in start-up management methods, puts it in this way: "*You're selling the vision and delivering the minimum feature set to visionaries not everyone.*" [10]

That is a key concept to keep in mind for the rest of this paper. The cloud-based CMDB that I developed is not only an application, but a new vision of using CMDBs that has not been tried before. The goal is then to write a solution with all the functionalities needed to prove to potential investors the efficiency and potential of the solution. In the case where the product is not convincing enough, then time will not have been wasted in developing something unsuccessful.

As stated earlier, developing a MVP is not common for most companies, but it can be an advantage for start-ups, especially while looking for new investors. Those investors, who could also be called "earlyvangelists" for early adopter and internal evangelist, are people with five major concerns that can be schemed by the pyramid represented on figure 1. [10]

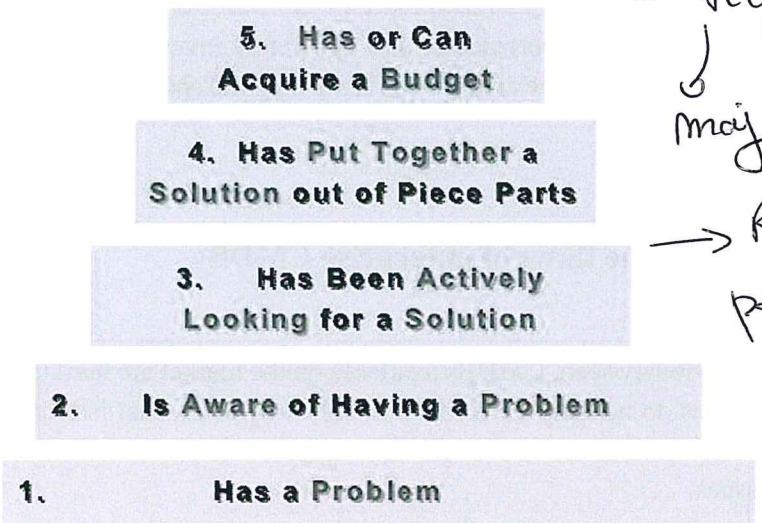

Figure 1: Earlyvangelist pyramid

The pyramid addresses the concerns of these potential investors:

- They have a problem.
- They understand they have a problem.

- They are actively searching for a solution and have a timetable for finding it.
- The problem is painful enough that they have cobbled together an interim solution.
- They have, or can quickly acquire dollars to purchase the product to solve their problem.

These people should be the target of the product, because they have the means to provide financial support to help the company pursue its development. They do not necessarily care if the product is ready to be sold, but care more about what the different available functions are, because they can see the potential the product can have on the market.

Let's take a well-known example in the business world: Facebook. At its beginning, it was just a simple social network that allowed people to access photos and basic information on people registered on the network. It was just a minimally viable product: it allowed people to be on the same network and share data. Rapidly, though, investors started being interested in the product and using it. At first it was mostly universities and college campuses, then the network expanded into the corporate world, until finally being able to reach everybody aged 13 years old and older with a valid email address. During all this evolution, developers kept adding features to the social network because it worked well. Investors were convinced, so they sponsored the product, thus allowing Facebook to grow and become the billion-user network we know today. [11]

This example proves the importance of a MVP. Before investing time and money in a product, it is first necessary to check if the concept can work and find sponsors to allow its growth into a larger application.

5.4 Answering the flaws of enterprise CMDBs

As we have previously seen, CMDBs available on the market are hard to keep up-to-date and, as a result, employees do not rely on them. The only way to fix that is to create a simple application that people will want to use so it can stay updated most of the time. There are, however, two important issues.

First, it is evident that a minimally viable product will never replace an enterprise CMDB published by Oracle. Even if features would progressively be added to the product, creating an enterprise CMDB will require time and money. Plus, if the solution was as ambitious as most software found on the market, the application will lose simplicity and we would end up with another "big" CMDB which would have the same flaws that were supposed to be fixed.

Second, according to the size of the company, having a CMDB is sometimes not an option. Most big companies will require one, but this implies having an enterprise solution as described in the

previous paragraph. Regarding smaller companies, a CMDB could be really useful but such companies obviously will not need an enterprise application like bigger companies do, so they usually end up with no CMDB at all.

With the application, the main focus was to answer these two flaws with two strategies targeting both environments. First, we decided to primarily target small- and medium-sized companies as potential customers for our CMDB. Such companies are looking for something simple, which perfectly described the kind of application we want to give them. They just want to focus on the core functionalities of a CMDB, so this describes perfectly the minimally viable product.

The other target could be units and departments of bigger companies, and even employees. While we do not expect huge businesses to replace their complex CMDB by a simple prototype, the solution can be advertised as a local alternative in a smaller scale and be used in different departments. Instead of representing the structure of the whole company, the CMDB could simply show the architecture of a specific department and its subcomponents. Thus, keeping the CMDB up-to-date will not require a lot of time, and employees will be able to know exactly what the different interactions between configuration items are.

This strategy of targeting two types of users with different goals could actually work if it helps people become more efficient and productive. But for this to be done, it is important to develop the solution using the latest technologies to help ensure a smooth experience for the user.

5.5 The importance of cloud-based applications

After targeting the audience for the application, it was important to determine which platform would support the application. Originally, my manager proposed a Java application working with Google App Engine. The advantage of this solution would be the opportunity to run the solution on any operating system. This solution would have followed OneCMDB, an open-source CMDB that was developed in Java.

When I started to select all the tools I would need to start the Java development, I encountered an issue with the Java version. Google App Engine did not support the latest Java version (Java 8) and it was necessary to develop the application using Java 7. Although it was not a problem to download and install that version, it made me think of another problem I encountered when I tested OneCMDB, the open-source application written in Java. This solution offered a visualization of the configuration items that I could not try on my computer, because the Java plug-in was made unavailable in the latest version of Google Chrome for security issues. [12] [13] In order for it to work, it was necessary to use a previous version of Chrome.

This little problem made me realize how crucial compatibility problems are in applications. It would be very unfortunate if users could not use the product just because the technologies used

encounter compatibility problems or are outdated. Using another browser could have been a solution, but if we take a look at the statistics given by the Digital Analytics Program (DAP) from the Federal Government, we realize that Chrome is the most used browser, all devices included, with 34.7% of all visitors. [14] This means that it was crucial to make the solution work on Chrome if we wanted to keep using the same technologies.

Intra-browsers compatibility can be one of the toughest parts of web development. Indeed, it is important to ensure that the application being developed will be working with the majority of browsers on the market. Figure 2 shows the market shares among browsers in the United States. It is interesting to note that Chrome is the most used one today, with a difference of 14 points with its main competitor, Internet Explorer, as of September 4, 2015. [15]

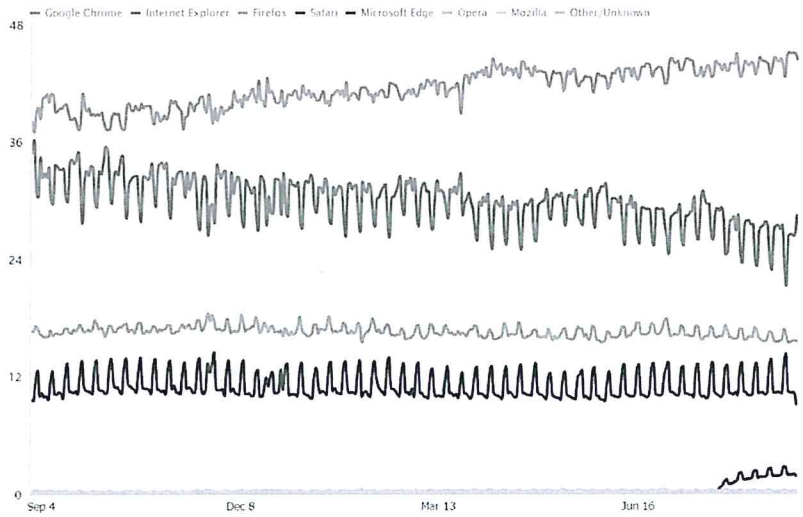

Figure 2: Web browsers market shares in the United States

This schema shows the importance of taking into account the cross-browser compatibility during the development. Although Chrome is the most used browser, IE, Firefox, and Safari are nonetheless widely used and it is necessary to make sure the application that will be developed will be compatible on these browsers as well.

In our original plan, although most part of the application would be for desktop, managing the database had to be done through Google App Engine, so using a browser was required anyways. The other consideration I had were the restrictions offered by a desktop application. The original application, as imagined by my manager and I, would be run only on a desktop computer. That could dramatically limit how the application is used.

According to the Pew Research Center, nearly two-thirds of Americans own a smartphone. [16] This does not only mean that there is an important market available regarding smartphone applications, but it also implies that Americans will most likely use their smartphone more frequently than they would use their laptop, due to having permanent access to their phone. The