ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO DE INGENIERÍA DE TECNOLOGÍAS Y
SERVICIOS DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE MADRID

TRABAJO FIN DE GRADO 2015

Design and implementation of an adaptive dashboard and visualization interface for the dynamic optimization and control of data centers

AUTOR
MARÍA GARCÍA FERNÁNDEZ

TRABAJO DE FIN DE GRADO

TÍTULO

DESIGN AND IMPLEMENTATION OF AN ADAPTATIVE DASHBOARD AND VISUALIZATION INTERFACE FOR THE DYNAMIC OPTIMIZATION AND CONTROL OF DATA CENTERS

AUTOR

D. MARÍA GARCÍA FERNÁNDEZ

TUTOR

D. FÉLIX MENCÍAS MORANTE

PONENTE

D. JOSE MANUEL MOYA FERNÁNDEZ

DEPARTAMENTO

INGENIERÍA ELECTRÓNICA

TRIBUNAL

Presidente: D. Rubén San Segundo Hernández

Vocal: Juan Antonio López Martín

Secretario: José Manuel Moya Fernández

Suplente: Álvaro Gutiérrez Martín

FECHA DE LECTURA Y DEFENSA: 22 de Diciembre de 2015

CALIFICACIÓN OBTENIDA:

People will forget what you said, people will forget what you did, but people will never forget how you made them feel.

Some say the world will end in fire,
Some say in ice.
From what I've tasted of desire
I hold with those who favor fire.
But if it had to perish twice,
I think I know enough of hate
To say that for destruction ice
Is also great
And would suffice.

Life consists of expectations or dreams, fears and time.
How would you manage those, it's up to you.

These violent delights have violent ends
And in their triumph die, like fire and powder,
Which, as they kiss, consume.

ACKNOWLEDGEMENTS

A mi familia, por estar siempre a mi lado, por quererme y lo más importante por aguantarme! Gracias, gracias y gracias.

Gracias porque vosotros me habéis hecho ser la persona que soy hoy.

Gracias por ser mi salvavidas y mi motor día a día.

Gracias por ser mis coaches personales y no dudar de mí ni un solo momento.

Os quiero.

También quiero dar las gracias a todas las personas de las que he aprendido, las que me han hecho crecer, las que me hacen reír y también vivir, aquellas personas que consideraron que merecía la pena, que vieron algo en mí y que me han animado a luchar por mis metas. Simplemente, gracias.

ABSTRACT

Over the last few years, the Data Center market has increased exponentially and this tendency continues today. As a direct consequence of this trend, the industry is pushing the development and implementation of different new technologies that would improve the energy consumption efficiency of data centers. An adaptive dashboard would allow the user to monitor the most important parameters of a data center in real time. For that reason, monitoring companies work with IoT big data filtering tools and cloud computing systems to handle the amounts of data obtained from the sensors placed in a data center.

Analyzing the market trends in this field we can affirm that the study of predictive algorithms has become an essential area for competitive IT companies. Complex algorithms are used to forecast risk situations based on historical data and warn the user in case of danger.

Considering that several different users will interact with this dashboard from IT experts or maintenance staff to accounting managers, it is vital to personalize it automatically. Following that line of though, the dashboard should only show relevant metrics to the user in different formats like overlapped maps or representative graphs among others. These maps will show all the information needed in a visual and easy-to-evaluate way.

To sum up, this dashboard will allow the user to visualize and control a wide range of variables. Monitoring essential factors such as average temperature, gradients or hotspots as well as energy and power consumption and savings by rack or building would allow the client to understand how his equipment is behaving, helping him to optimize the energy consumption and efficiency of the racks. It also would help him to prevent possible damages in the equipment with predictive high-tech algorithms.

KEY WORDS

Data Processing Centers (CPD), Data Center Infrastructure Management (DCIM), visualization, energy consumption, efficiency, optimization, sensors, metrics, monitoring, overlapped map, layers, gradients, hotspots, Graphite, APIs.

ACRONYMS

AJAX	Asynchronous JavaScript and XML
APAC	Asia Pacific
API REST	Application Programming Interface Representational State Transfer
AWS	Amazon Web Services
CF	Consolidation Function
CLI	Common Language Infrastructure
CPD	Data Center Processing
CPU	Central Processing Unit
CSRF	Cross Site Request Forgery
CSS	Cascade Style Sheet
D3	Data Driven Documents
DCIM	Data Center Monitoring Infrastructure
DHCP	Dynamic Host Configuration Protocol
DHMTL	Dynamic HTML
DOM	Document Object Model
DWR	Direct Web Remoting
EC2	Elastic Compute Cloud
ECR	EC2 Container Registry
ECS	EC2 Container Service
GNU	Gnu's Not Unix
GUI	Graphical User Interface
HDFS	Hadoop Distributed File System
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
I/O	Input Output
IaaS	Interface as a Service
IAM	Identity Access Management
IETF	Internet Engineering Task Force
IoT	Internet of Things

IT	Information Technology
JS	Java Script
JSON	JavaScript Object Notation
KVM	Kernel-based Virtual Machine
LAMP	Linux Apache MySQL Perl/PHP/Pyton
LGPL	Lesser General Public License
MLlib	Machine Learning Library
MPL	Mozilla Public License
MySQL	My Structured Query Language
NFC	Near Field Communication
PC	Personal Computer
PHP	Hypertext PreProcesor
RCP	Remote Procedure Call
RDBMS	Relational Data Base Management System
RFID	Radio Identification
RMI	Remote Method Invocation
RRD	Round Robin Database
SCADA	Supervisory Control And Data Acquisition
SO	Operating System
SOAP	Simple Object Access Protocol
SVG	Scalable Vector Graphics
TWh	Terawatt-hour
UML	Unified Modelling Language
URL	Uniform Resource Locator
VML	Vector Markup Language
VPN	Virtual Private Network
W3C	WWW Consortium
WAN	Wide Area Network
WWW	World Wide Web
XCB	Extended Codebook
XHTML	Extensible Hyper Text Markup Language
XML	Extensible Markup Language
YARN	Yet Another Resource Negotiator

LIST OF FIGURES

Table of contents

1. INTRODUCTION

1.1. MOTIVATION AND CONTEXT

A data processing center consists of thousands of servers located in the same building. These servers offer different services to multiple users such as financial and accounting software applications, e-mail hosting systems or cloud computing storage services among others. Almost a20% of European enterprises used cloud computing services in 2014 [1]. In the last years, the data center market has increased exponentially and according to different experts on the field, this tendency will continue. This trend has a direct effect on energy consumption. In 2013, U.S. data centers consumed approximately 90 TWh of electricity (enough electricity to power all the households in New York City twice over a year). Some estimates indicate that it will increase to 140 TWh annually by 2020 which means a carbon footprint of 150 million tons every year [2].

Due to this huge industry demand, industry and academia have focused on developing complex algorithms to optimize data center energy consumption. As an interesting fact, only in Google Scholar there are more than 2.300.000 results when searching for "data centers energy". The purpose of this project is to develop a sophisticated dashboard that will allow the user to visualize and control inefficiencies and optimise energy consumption of data centers.

1.2. CLOUD COMPUTING & INTERNET OF THINGS

Cloud computing offers undisputed benefits in terms of agility and cost-effectiveness. It is a fact that cloud computing is dominating IoT application market. More than a 50% of IoT developers primarily connect devices through the Cloud and almost a 26% of IoT developers associate cloud computing with IoT and work with the Cloud as a development environment [3]. In Figure 1.1, it is shown a forecast study of the number of devices in use based on the last ten years. In the next two years, the number of these devices will grow exponentially mainly due to IoT development.

Cloud technology is maturing and improving its security and data integration. It offers scalable resources on demand to business users. It also is a sophisticated tool that helps to analyze vast amounts of data efficiently and cost-effectively. Cloud services offer exceptional flexibility, enabling IT to evaluate the best approach to business users' request. Using cloud infrastructure to analyze big data makes sense for the companies due to several reasons. First of all, investments in big data analysis can be significant and drive a need for efficient and cost-effective infrastructure. Private clouds may offer a more efficient and cost-effective model to implement analysis of big data, than just increasing internal resources with public cloud services. However, a hybrid cloud option enables companies to use on-demand storage space and computing power via public cloud solutions for certain analytics initiatives and provide added capacity and scale as needed. Secondly,

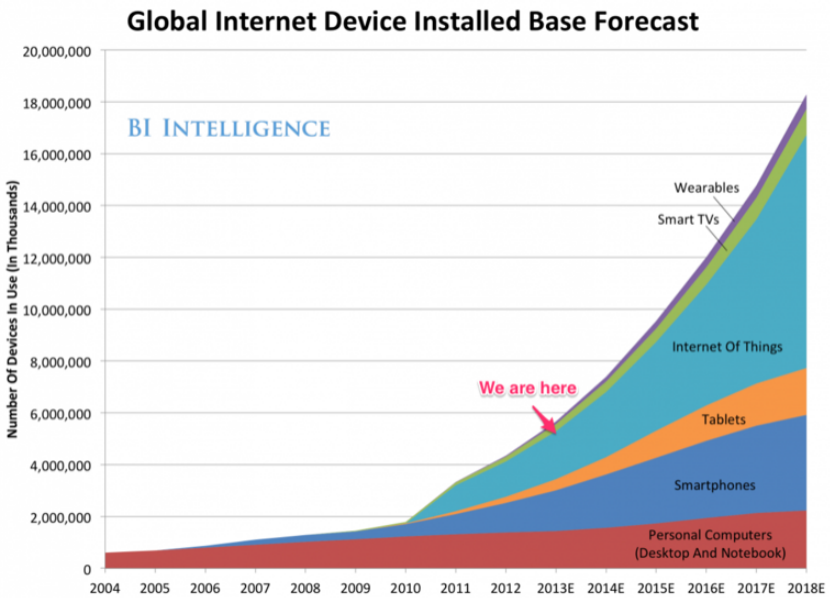


Figure 1.1. Global Internet Device Installed Base Forecast - Garner, IDC, Strategy Analytics, Machina Research, company fillings, BII estimates [4]

big data may mix internal and external sources. Enterprises often keep their most sensitive data in-house. However, huge volumes of big data generated by third-parties or public providers may be located externally. Analyzing the data wherever it resides often makes more sense, either in internal or public cloud data centers or in edge systems and client devices, than moving relevant data sources behind your firewall, as the latter implies a significant commitment of resources. This holds true regardless the data is in internal storages or in public data centers or even in edge systems and client devices. Finally, data services are needed to extract value from big data [5].

It is a fact, that the use of IoT and cloud computing is increasing exponentially thus, a real need to visualize the data generated will come up. This need will be completely satisfied with our dashboard.

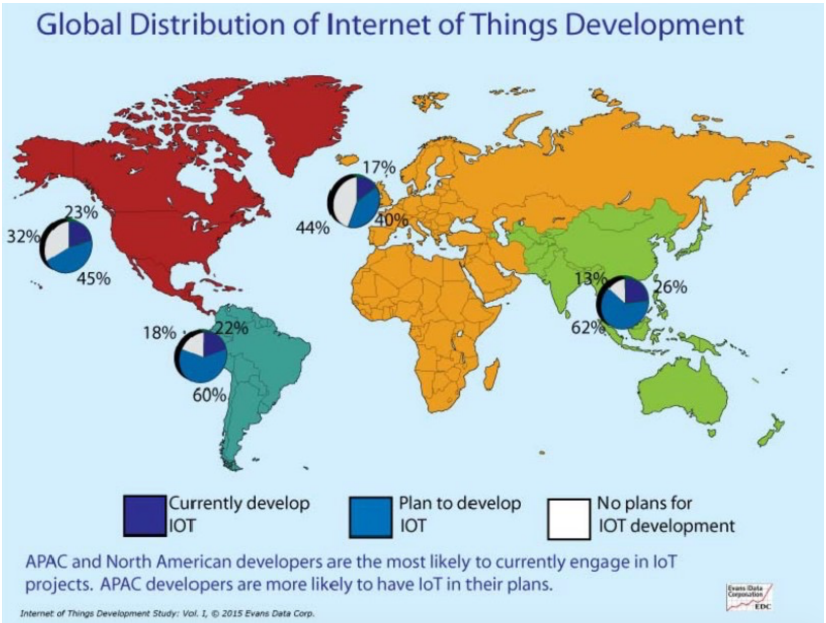


Figure 1.2. Global distribution of IoT development by IoT Development Study Vol I - Evans Data Corp.

1.3. OVERVIEW OF THE STATE-OF-THE-ART

It is a fact that the real value of big data resides in the insights it produces when analyzed: enabling the discovery of patterns or identifying vital indicators for decisions. Big data analytics are a set of advanced technologies designed to work with large volumes of heterogeneous data. It uses sophisticated quantitative methods such as machine learning, neural networks, robotics, computational mathematics and artificial intelligence to explore the data and to discover interrelationships and patterns [6].

In today's market, there are some companies that provide data center infrastructure management services or that develop software tools or applications focus on IT processes management. Optimisee is a Spanish company born in 2013 that offers a real solution to save energy and improve the efficiency and capacity of data centers. By analysing data from appropriate sensors, they generate predictive models for critical variables in the data center. These models evolve continuously to dynamically suggest proposals for action with an estimation of the impact on business.

In this field, we can find some Optimisee direct competitors like Schneider Electric which is a Spanish company that provides energy management and industrial automation systems. However, they are more focused on domestic or home automation systems.

We also have SynapSense, a Panduit company since 2014, that provides wireless data center monitoring and cooling control solutions. While Panduit is known as a developer and provider of general solutions that connect, manage and automate the physical infrastructure, SynapSense is focused on data center solutions, delivering unparalleled visibility, reliability and energy efficiency among others. They offer a DCIM service with some extras such us power and environmental wireless monitoring [7].

Additionally, OpenStack is a free and open source software platform for cloud computing that was born in 2010. It is considered an IaaS *(see Appendix I)*. The software platform consists of interrelated components that control hardware pools of processing, storage and networking resources throughout a data center. Users manage them through a web-based dashboard, through command-line tools or through a RESTful API. It was released under Apache License terms [8].

Furthermore, VMware offers virtualization and automation of compute, networking and storage resources from the PC desktop to the data center and to the cloud. They provide efficiency, agility and control with their cloud infrastructure and business mobility solutions enabled by "One cloud, any application, any device" [9].

Moreover, Amazon is developing a new product that is known as ECR, that will be available in the market in the next months. It is basically a fully-managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images. Amazon ECR is integrated with ECS, simplifying the user's development to production workflow. It eliminates the need to operate your own container repositories or worry about scaling the underlying infrastructure. It hosts your images in a highly available and scalable architecture, allowing the user to reliably deploy containers for your applications. Integration with AWS Identity and IAM provides resource-level control of each repository. Therefore, there are no upfront fees or commitments with Amazon ECR [10]. Among its benefits, we could find that:

- Amazon ECR eliminates the need to operate and scale the infrastructure required to power the container registry. It is fully managed, there is no software to install or infrastructure to scale. The user just needs to push his container images to Amazon ECR and pull them when he needs to deploy.
- It transfers the user container images over HTTPS and automatically encrypts them at rest. The user can also configure policies to manage permissions and control access to his images using AWS Identity and IAM users and roles without having to manage credentials directly on your EC2 instances.
- It has a highly scalable, redundant, and durable architecture. User's container images are highly available and accessible, allowing the user to reliably deploy new containers for his applications.
- It integrates with Amazon ECS and the Docker CLI, allowing the user to simplify his development and production workflows.

Microsoft is also expanding into this market with Microsoft Azure *(see Appendix I)*. However, its product is not already mature enough to widely spread.

1.4. OBJECTIVES OF THE PROJECT

The main purpose of this project is to develop a dashboard that will allow the client to monitor and control different parameters of a data center. In order to achieve that, a list with the multiple objectives to cover appears above:

1. Analyse, study and deeply understand Optimisee's architecture.
2. Acquire knowledge of the relevant technologies involved.
3. Develop a specific API to monitor relevant data in the dashboard with data updates every hour.

4. Design an overlapped map that will show the information listed below of a data center room:
    - Temperature
    - Pressure
    - Humidity
    - Hotspots
5. Generate the map images in the client side instead of create them in the server side.
6. Add new functionalities to the dashboard that would allow the client to control actuators in the data center room.

1.5. STRUCTURE OF THE PROJECT

This project is structured in three main parts. The first one consists of a state-of-the-art and a brief description of what Optimisee and its competitors are developing (section 1.3), capped by an analysis of how the market is likely to evolve over the next few years (sections 1.1 and 1.2). The second stage is characterized fundamentally by a deep analysis, study and description of the Optimisee's services architecture (section 2.1) and an overview of the major technologies used in this field (section 2.2). In addition, this section also encompasses an overall explanation about the workspace architecture used in the implementation (section 2.3). Finally, a complete description of the new functionalities introduced and the libraries used will be explained (section 2.3.4 and chapter 3).

2. DESIGN

2.1. ARCHITECTURE

The complete architecture adopted during the development of the adaptive dashboard and visualization interface for the dynamic optimization and control of data centers is described below.

As it is shown in *Figure 2.1*, a user connected by a VPN for security reasons, will sign in the system. Once the user has already crossed the firewall that protects the server from external and untrusted users petitions, he or she may send a query petition using a collection of Rest APIs. This request will ask, for example, for specific external information collected by different sensors. This request is mainly divided into three steps. First of all, when the request is sent a response from the Storage Cache DB will be sent back with all the objects and parameters of the data center room. This response is managed by the Storage Docker container. Secondly, once the GUI Docker Container knows which parameters have to be displayed in the web page, it will ask for them to Graphite through Graphite's API, who will respond with the data requested for each metric. Finally, using complex algorithms and a compatible API it is possible also to forecast the energy consumption, the temperature and humidity inside a CPD, among others. Once all the information is collected and prepared it is displayed in the frontend application.

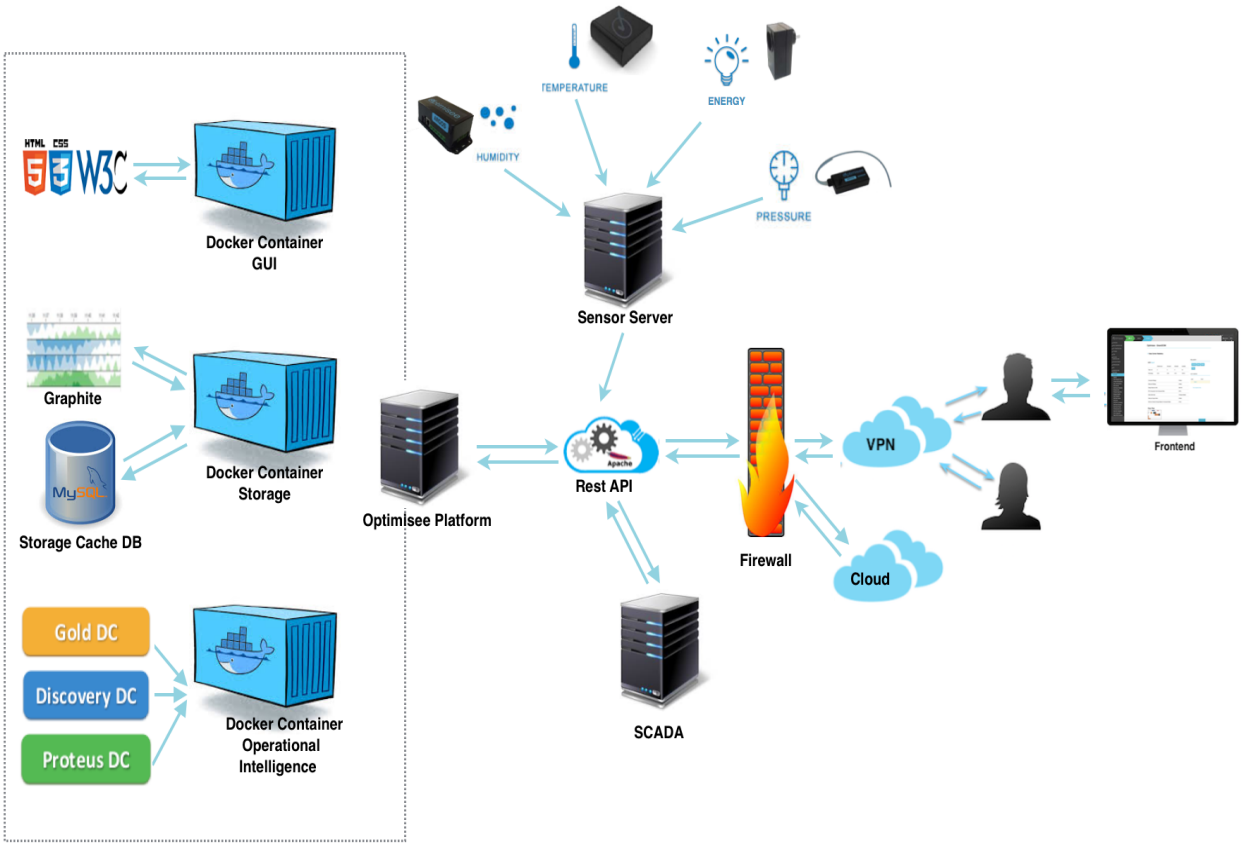


Figure 2.1. Platform architecture.

2.2. RELAVANT TECHNOLOGIES

In order to design and implement the dashboard we work with the technologies explained below.

2.2.1. HTML5 & CSS3

HTML5 is the fifth revision of the HTML standard, a markup language used for structuring and presenting content on the WWW. It was finalized and published on 28 October 2014 by the W3C. HTML5 is a response to the fact that the HTML and XHTML in common use on the WWW have a mixture of features introduced by various specifications, along with those introduced by software products such as web browsers and those established by common practice.

It is also an attempt to define a single markup language that can be written in either HTML or XHTML. It includes detailed processing models to encourage more interoperable implementations. It extends, improves and rationalizes the markup available for documents and introduces markup and APIs for complex web applications. For the same reasons, HTML5 is also a potential candidate for cross-platform mobile applications. Many features of HTML5 have been built with the consideration of being able to run on low-powered devices such as smartphones and tablets.

HTML5 adds many new syntactic features. These include the new <video>, <audio> and <canvas> elements, as well as the integration of SVG content (replacing generic <object> tags), and MathML for mathematical formulas. These features are designed to make it easy to include and handle multimedia and graphical content on the web without having to resort to proprietary plugins and APIs. HTML5 also includes new page structure elements such as <main>, <section>, <article>, <header>, <footer>, <aside>, <nav> and figure and some elements such as <a>, <cite> and <menu> have been changed, redefined or standardized. The APIs and DOM are no longer afterthoughts, but are fundamental parts of the HTML5 specification. HTML5 also defines in some detail the required processing for invalid documents so that syntax errors will be treated uniformly by all conforming browsers and other user agents [11].

Since the inception of web pages, there has always been a significant difference between document content and document presentation. For this reason, CSS was created, being its latest standard CSS3 which is completely backwards-compatible with earlier versions of CSS [12].

A CSS is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document and is applicable to rendering in speech or on other media. CSS is designed to take care of the document presentation, focusing on the layout, colors and fonts. Along with HTML and JavaScript, CSS is

used by most world wide websites to create visually engaging webpages, user interfaces for web applications and mobile applications [13].

This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content, such as semantically insignificant tables that were widely used to format pages before consistent CSS rendering was available in all major browsers.

This separation of formatting and content makes it possible to present the same markup page in different styles for different rendering methods. It can also be used to display the web page differently depending on the screen size or device on which it is being viewed. Another advantage of CSS is that aesthetic changes to the graphic design of a document or more than one can be applied quickly and easily by editing a few lines in one file, rather than by a laborious and expensive process of crawling over every document line by line, changing markup.

The CSS specification describes a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this scheme called cascade, priorities or weights are calculated and assigned to rules in order to make the results predictable.

During the design of the front-end web page we work with Bootstrap, which is the most popular HTML, CSS and JS framework for developing responsive, mobile first projects on the web[14].

2.2.2. GRAPHITE

Graphite is an enterprise-scale monitoring tool. It was originally designed by Chris Davis in 2006. In 2008, Graphite was released under the open source Apache 2.0 license and today many large companies use it, as it is a pillar of the nowadays e-commerce monitoring systems [15].

It has two main functionalities highly related: it stores numeric time-series data and it renders graphs of these data on demand. However, Graphite does not collect data, for this reason some tools have appeared in the market that send data to Graphite.

Moreover, from a CPU perspective, Graphite scales horizontally on both the frontend and the backend, allowing to add more machines to get more throughput. It is also fault tolerant considering that losing a backend machine will cause a minimal amount of data loss and will not disrupt the system if you have enough capacity remaining to handle the load.

On the other hand, from an I/O perspective, under load Graphite performs lots of tiny I/O operations on lots of different files very rapidly. This is because each distinct metric sent to

Graphite is stored in its own database file, similarly to how many tools (Drraw[1], Cacti[2], Centreon[3], etc) built on top of RRD work. In fact, Graphite originally did use RRD for storage until fundamental limitations arose that required a new storage engine.

To end up with, Graphite is capable of updating thousands of different metrics every minute. Just in case the disks cannot keep up with the large number of small write operations that occur (each datapoint is only a few bytes: however, most standard disks cannot do more than a few thousand I/O operations per second), Graphite's backend catches the incoming data and Graphite's database engine, Whisper, allows Carbon to write multiple data points at once, thus increasing overall throughput only at the cost of keeping excess data cached in memory until it can be written.

GRAPHITE'S ARCHITECTURE

If one user writes an application that collects numeric time-series data that would be interesting to graph, first of all, the information collected is sent to Graphite's processing backend, Carbon, which stores the data in Graphite's specialized database. Afterwards, the data may be visualized through graphite's web interfaces. From this, we can conclude that Graphite consists of 3 software components intrinsically related as we can see in *Figure 2.2*:

- **CARBON:** One or more of various daemons with different functionalities that build up the storage backend of a Graphite installation. In simple installations, there is typically only one daemon, carbon-cache.py. All of the carbon daemons listen for time-series data and can accept them over a common set of protocols. However, they differ in what they do with the data once they receive them [19].
- **WHISPER:** Whisper is a fixed-size database, similar in design to RRD (*see Appendix II*). It provides fast, reliable storage of numeric time-series data over time.
- **GRAPHITE WEBAPP:** The Graphite web app is built on the Django (*see Appendix III*) web framework and uses the ExtJS javascript GUI toolkit[4]. However the graph rendering is done using the Cairo graphics library and finally the backend and database are written in Python.

---

[1] *Drraw is a simple web based presentation front-end for RRDtool that allows you to interactively build graphs of your own design [16].*

[2] *Cacti is a complete network graphing solution designed to harness the power of RRDTool's data storage and graphing functionality [17].*

[3] *Cetreon is a real-time IT performance monitoring and diagnostics management open source tool [18].*

[4] *Ext JS is a pure JavaScript application framework for building interactive cross platform web applications using techniques such as Ajax, DHTML and DOM scripting [20].*

Once you collect the data and send the data points to Carbon, they become immediately available for graphing in the web app. The web app offers several ways to create and display graphs including a simple URL API for rendering that makes it easy to embed graphs in other webpages.

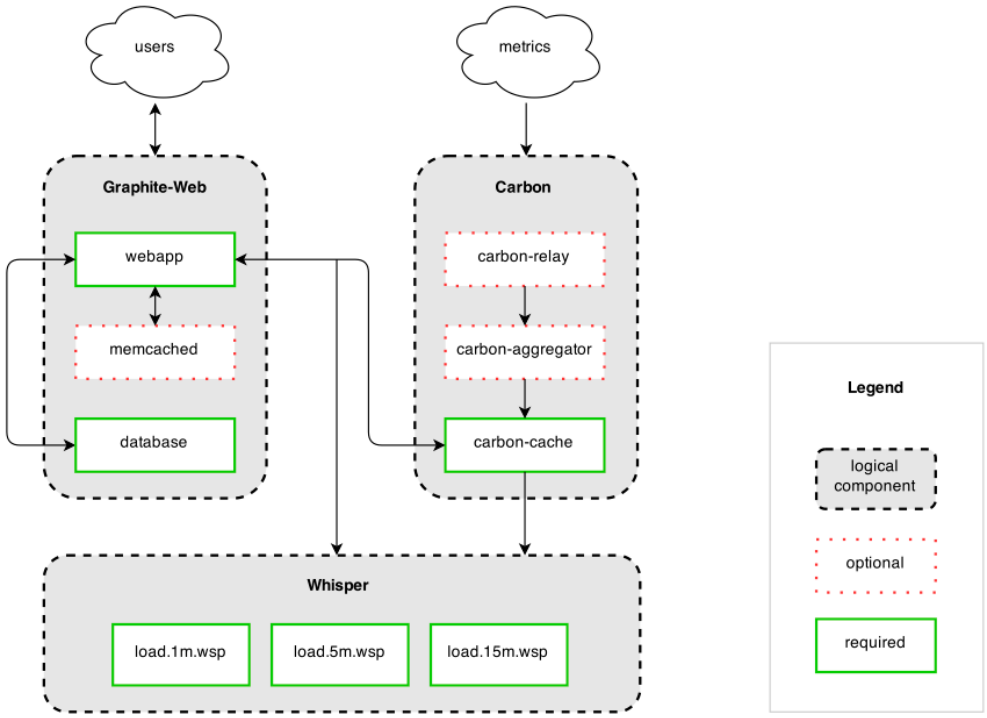


Figure 2.2. Graphite Architecture.

2.2.3. MySQL

MySQL is a mature open-source RDBMS owned by Oracle Corporation, it was the world's second most widely used RDBMS and the most widely used open source client-server model RDBMS. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements.

It is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack. MySQL is also used in many high-profile, large-scale websites, including Google (though not for searches), Facebook, Twitter, Flickr and YouTube [21].

MySQL has a GUI tool to administer MySQL databases or manage data contained within the databases for multiple SO. Users may use the included command line tools, install MySQL Workbench via a separate download or use a third party GUI tool [22].

2.2.4. DOCKER

Docker is a platform specially designed for developers and administrators of multiple systems that helps them to develop, ship, deploy and run applications easily. Docker is an exclusive tool that supports different users (developers, system admins, release engineers) to assemble applications, test and deploy them into production as fast as possible. Docker containers are formats that allow developers to care about their applications inside them. The use of these containers substantially simplifies the management and deployment of code. They are lightweight and fast. Docker runs almost in every 64bits SOs and it is extremely attractive for data centers [23].

Some of the most appealing Docker's characteristics are the deployment and easy scalability that it provides. Since Docker runs on so many platforms, it is easy to move an application from a testing environment into the cloud and back when it is necessary. In addition, it is lightweight what makes faster and easier scaling up and down. The user can quickly launch and shut containers when needed. Moreover, Docker containers are not virtualizations. They do not need a hypervisor (as you can see in *Figure 2.3*) thus, users can pack more of them onto their hosts.

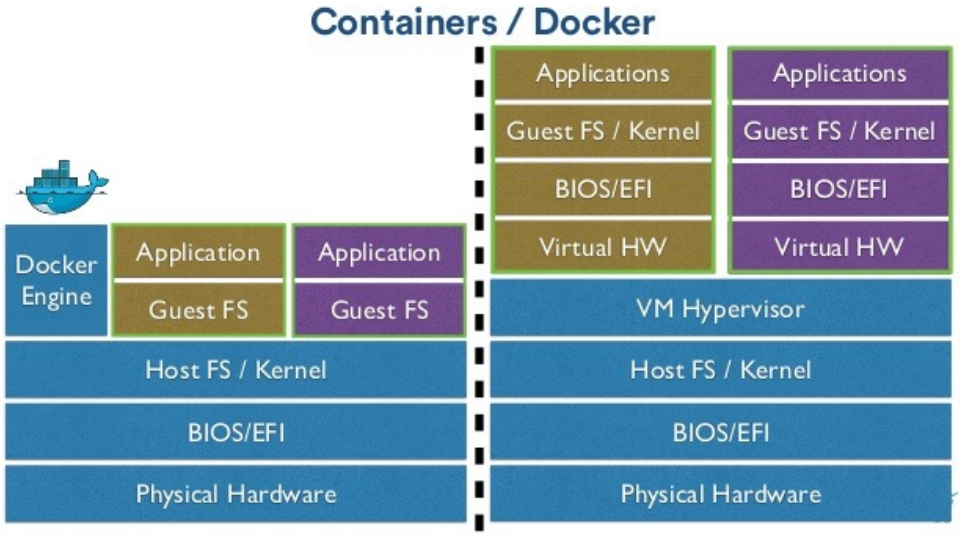


Figure 2.3. Docker vs VMs

And finally, faster deployment makes management easier. As Docker speeds up your work flow, it gets easier to make lots of small changes instead of huge updates. Smaller changes mean reduced risk and more uptime.

DOCKER'S ARCHITECTURE

Docker uses a client-server architecture. The Docker client and daemon communicate via sockets or through a RESTful API. Both of them could be run on the same system or the Docker client could be connected remotely to a Docker daemon.

- **THE DOCKER DAEMON**

The Docker daemon runs on a host machine. It builds, runs and distributes the Docker containers. However, the user directly interacts with the Docker client, instead of the Docker Daemon.

- **THE DOCKER CLIENT**

The Docker client is the primary user interface to Docker. It accepts commands from the user and communicates back and forth with the Docker daemon.

To understand Docker's internal architecture, it is important to distinguish between the following three components: Docker images, registries and containers. In order to do understand the differences between them, it is very enlightening the *Figure 2.3*.

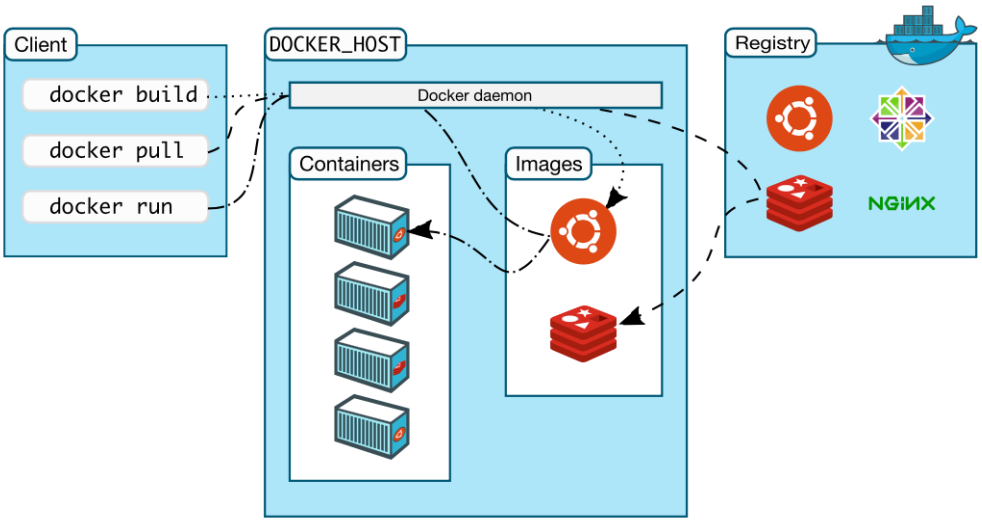


Figure 2.4. Docker architecture

✺ **DOCKER IMAGES**

A Docker image is a template in which the user is only allowed to read. It contains an operating system with some functionalities already installed. These images are used to create Docker containers. With Docker, any user may build new images, update existing ones or download "open source" images that other people have already created.

✺ **DOCKER REGISTRIES**

Docker registries hold images. These are public or private stores from which you upload or download images. They may also create their own images or use images that other users have previously created.

✺ **DOCKER CONTAINERS**

Docker containers are similar to a directory. A Docker container holds everything that is needed for an application to run. Each container is created from a Docker image. Docker containers can be run, started, stopped, moved and deleted. Each container is an isolated and secure application platform.

2.3. SUBSYSTEM DESIGN

2.3.1. INTRODUCTION

Usually, when a user requests an image that contains real time information, the picture is immediately loaded from the server where the image was pre-generated. These images are refreshed and stored continuously. This means an exhaustive and inefficient use of the server due to the storage of all the possible combination of images.

However, Optimisee generates these images in the client side due to its dynamism. This tendency is becoming more common since when generating images in the client side you obtain directly the data and is actually the user who chooses how to display it.

We needed to develop API's that will work together with the existing ones in order to send the data necessary to generate these images.

2.3.2. APIS

As it has been mentioned before, we work with Graphite's API in order to obtain the requested data from specific metrics. Graphite's API fetches metrics from a time-series database and renders graphs or JSON data out of these time series [26]. However, we need to handle different APIs compatible with Graphite's because our design includes more functionalities such us:

- Data graphing: One of the main functionalities of Optimisee is to forecast and predict the energy consumption, temperature and pressure using complex algorithms.
- Managing objects in a CPD: In this case the HTTP API is used and together with the frontend application, it allows the user to create, modify or eliminate different RoomObjects like racks or cabinets, temperature, humidity, pressure inside of a data center room or energy sensors among others.
- Warning: It would be interesting to alert the user by mail, SMS and directly to the frontend interface when some extreme or danger situations occur, that could damage the equipment. The user may choose how he or she wants to be informed and in which situations by the web page.
- Actuators: In case there would be any risk situation in our CPD, with some actuators we may ensure the effective control of any rack and its processes. In addition, the client may pre-set some maximum values and the actions to take if these thresholds are exceed like,

for example, start the cooling system if the temperature in the room or in a hotspot is up to 40ºC, or switch the rack off if it is higher than 70ºC.

- Profiles: Another API will manage a variety of different users. Information about these users and their visualization preferences will be stored. When a user logs in the web page there would be a request of their favorite metrics and all the information obtained will be displayed.

In the next figure, we will see the requests sent between Docker containers and the explanation of these processes when the user registers a new device and asks for the data that it collects.

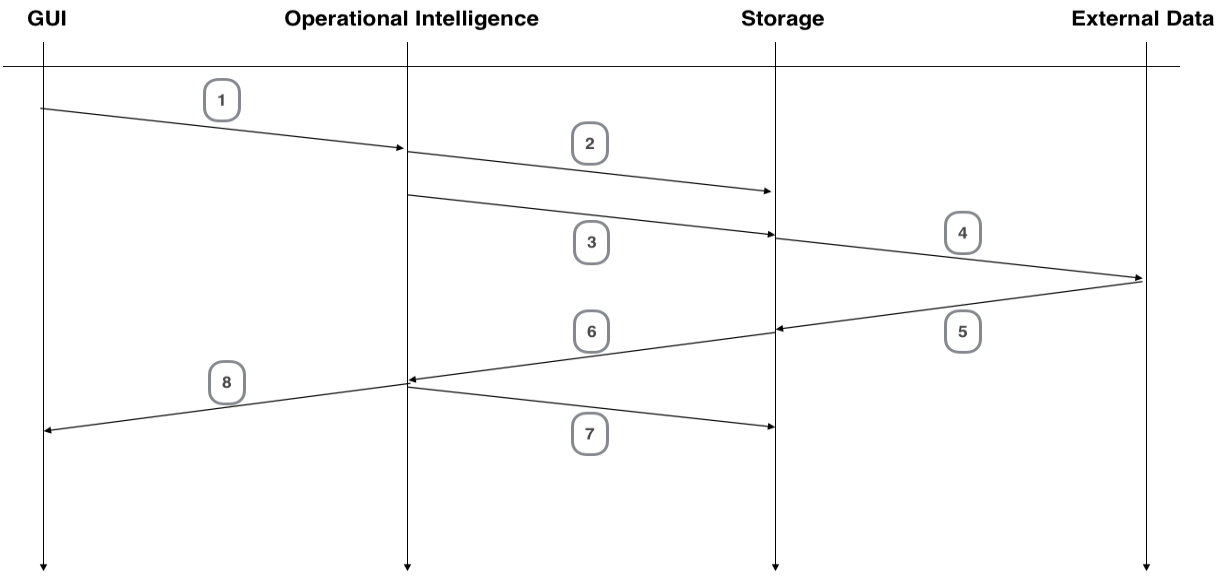


Figure 2.5 Requests between Docker containers

1. The user registers a new room object in the system.
2. All the new information is stored in the database
3. At the same time, another request delayed a bit is sent to the database. Here, we find all the device parameters that will be filled in with data in the next request.
4. A request is sent to obtain real time information directly from the sensors.
5. A response with all the information requested is sent back.
6. The same response is sent back to the Operational Intelligence Docker container.
7. All the information requested is stored in the database.
8. The Operational Intelligence Docker container notifies to the GUI that the data are ready to be displayed.

These APIs are necessary to communicate effectively between Docker Containers, notify the satisfactory read of external data or to create new devices such as sensors or racks.

2.3.3. SERVER SIDE

INTRODUCTION

The server side runs in GNU/Linux with an Apache PHP server. To develop PHP webapps, we first need to setup Apache, MySQL and PHP in order to work in a LAMP environment. We will program the database in the SQL language, the sever-side in PHP and the client-side in HTML, CSS and JavaScript.

DATABASE STORAGE

We access to MySQL (see section 2.2.3) as a storage system in order to save all the information collected from the requests.

In order to prepare our scenario, we needed to create two database tables: fac_DataCenter and fac_RackSensor. The first one stores the two data center that we have been working with whereas the second one stores all the sensor placed in different racks. We need to create these tables to store new data center or new sensors and its metrics. Once these metrics were stablished, we may use them to read data from graphite.

It is shown below some of the MySQL expressions that we write to create and fill the tables presented in *Figure 2.7* and *Figure 2.8*.

```
CREATE TABLE fac_RackSensor (
    SensorId INT(11) NOT NULL PRIMARY KEY,
    SensorMetric VARCHAR(80) NOT NULL,
    RackId INT (11) NOT NULL,
    SensorType VARCHAR(20) NOT NULL,
    SensorZ INT(11) NOT NUL);

INSERT INTO fac_RackSensor VALUES
    (1,‘optimisee. .discovery. .hq.sala1.rack.r01.temperature.outlet.62’,1,’Temperature’0),
    (1, ‘optimisee. .discovery. .hq.sala1.rack.r01.pressure.1’,1,’Pressure’, 10),
    (1, ‘optimisee. .discovery. .hq.sala1.rack.r01.pressure.2’,1,’Pressure’, 10),
                                        …
    (1, ‘optimisee. .discovery. .hq.sala1.rack.r02.humidity.6’,2,’Humidity’, 20);
```

```
mysql> select*from fac_DataCenter;
+--------------+-------------------------------------+---------+---------------+-----------------+---------------+--------+-----------------+---------------+-------------+---------+---------+
| DataCenterID | DataCenterMetric                    | Name    | SquareFootage | DeliveryAddress | Administrator | MaxkW  | DrawingFileName | EntryLogging | ContainerID | MapX    | MapY    |
+--------------+-------------------------------------+---------+---------------+-----------------+---------------+--------+-----------------+---------------+-------------+---------+---------+
|            1 | optimisee.GT1M1-A                   | GT1M1-A |           500 |                 |               |  2500  | GT1M1-A.png     |             0 |           2 |   458   |   482   |
|            2 |                                     | GT1M1-B |           150 |                 |               |   150  | example.png     |             0 |           2 |   788   |   560   |
|            3 | optimisee.       discovery.       hq.sala1 |         |             0 |                 |               |     0  |                 |             0 |           3 |     0   |     0   |
+--------------+-------------------------------------+---------+---------------+-----------------+---------------+--------+-----------------+---------------+-------------+---------+---------+
3 rows in set (0.00 sec)

mysql>
```

Figure 2.6. fac_DataCenter database table.

```
mysql> select*from fac_RackSensor;
+----------+-----------------------------------------------------------------------------+--------+-------------+---------+
| SensorId | SensorMetric                                                                | RackId | SensorType  | SensorZ |
+----------+-----------------------------------------------------------------------------+--------+-------------+---------+
|        1 | optimisee.       .discovery.       .hq.sala1.rack.r01.temperature.outlet.62 |      1 | Temperature |       0 |
|        2 | optimisee.       .discovery.       .hq.sala1.rack.r01.pressure.1            |      1 | Pressure    |      10 |
|        3 | optimisee.       .discovery.       .hq.sala1.rack.r01.pressure.2            |      1 | Pressure    |      10 |
|        4 | optimisee.       .discovery.       .hq.sala1.rack.r01.temperature.inlet.65  |      1 | Temperature |       0 |
|        5 | optimisee.       .discovery.       .hq.sala1.rack.r02.temperature.inlet.59  |      2 | Temperature |       0 |
|        6 | optimisee.       .discovery.       .hq.sala1.rack.r02.temperature.inlet.64  |      2 | Temperature |       0 |
|        7 | optimisee.       .discovery.       .hq.sala1.rack.r02.temperature.outlet.1  |      2 | Temperature |       0 |
|        8 | optimisee.       .discovery.       .hq.sala1.rack.r02.temperature.outlet.61 |      2 | Temperature |       0 |
|        9 | optimisee.       .discovery.       .hq.sala1.rack.r01.pressure.3            |      1 | Pressure    |      10 |
|       10 | optimisee.       .discovery.       .hq.sala1.rack.r02.pressure.4            |      2 | Pressure    |      10 |
|       11 | optimisee.       .discovery.       .hq.sala1.rack.r02.pressure.5            |      2 | Pressure    |      10 |
|       12 | optimisee.       .discovery.       .hq.sala1.rack.r02.pressure.6            |      2 | Pressure    |      10 |
|       13 | optimisee.       .discovery.       .hq.sala1.rack.r01.humidity.1            |      1 | Humidity    |      20 |
|       14 | optimisee.       .discovery.       .hq.sala1.rack.r01.humidity.2            |      1 | Humidity    |      20 |
|       15 | optimisee.       .discovery.       .hq.sala1.rack.r01.humidity.3            |      1 | Humidity    |      20 |
|       16 | optimisee.       .discovery.       .hq.sala1.rack.r02.humidity.4            |      2 | Humidity    |      20 |
|       17 | optimisee.       .discovery.       .hq.sala1.rack.r02.humidity.5            |      2 | Humidity    |      20 |
|       18 | optimisee.       .discovery.       .hq.sala1.rack.r02.humidity.6            |      2 | Humidity    |      20 |
+----------+-----------------------------------------------------------------------------+--------+-------------+---------+
18 rows in set (0.00 sec)

mysql>
```

Figure 2.7. fac_RackSensor database table.

GRAPHITE PANEL

Graphite works with the Cairo library to display real-time data in the server side.

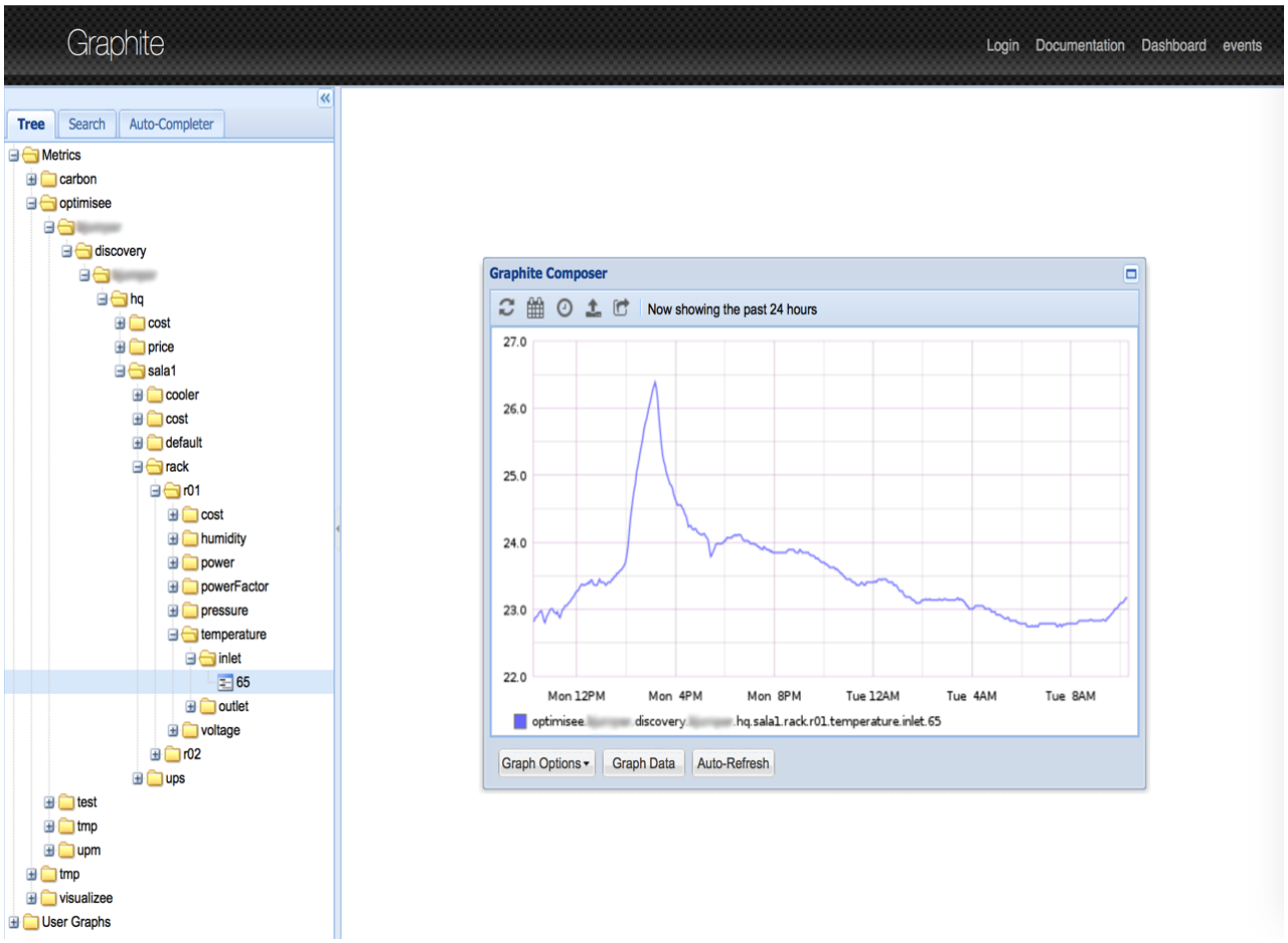


Figure 2.8. Graphite metric

CAIRO

It is a free software library and is available to be redistributed or modified under the terms of either the GNU LGPL version 2.1 or the MPL version 1.1. We use Cairo, a to collect data from Graphite, which is a Graphite 2D graphics library that is written in C [30].

Cairo is designed to produce consistent output on all output media while taking advantage of display hardware acceleration when available. The Cairo API provides several operations including stroking and filling cubic Bézier splines, transforming and compositing translucent images and antialiased text rendering. All drawing operations can be transformed by any affine transformation (scale, rotation, shear, etc.).

This graphic library helps us to display all the information collected directly from Graphite. Optimisee also developed another API to store all these parameters and predict with complex algorithms future situations thanks to this historic data.

PHP

PHP is a server scripting language designed by Rasmus Lerdorf, a powerful tool to create dynamic and interactive websites. It is a fast, flexible and widely-used scripting language. We have chosen PHP among other web languages such us Ruby[5], Phyton[6] or Javascript (Node.js)[7] due to several reasons that are shown below.

	PHP	Ruby	Phyton	Javascript (Node.js)
Free Software	Yes	Yes	Yes	Yes
Learning Curve	Short	Too long	Short	Short
Community of users and developers	Large	Small	Medium	Medium
Provides extensive database support	Medium	Bad	Good	Medium
Offers extension API	Yes	C	C, C++	No
Great number of available extensions and source codes	A lot	A few	A few	A lot

[5] Ruby is a programming language that runs with Ruby on Rails or simply rails, an open source, full-stack web application framework. It is a dynamic, imperative object-oriented programming language developed by Yukihiro Matsumoto in Japan. It also has dynamic type system and automatic memory management.

[6] Python is a widely-used high level design for programmers to express concepts with fewer lines of code. It was conceived in the late 1980s and was implemented by Guido van Rossum. It supports multiple ways of building the structure and elements of computer programs, including object-oriented and functional programming [27].

[7] Node.js is an open-source, cross-platform runtime environment for developing server-side web applications. Node.js applications are written in JavaScript and can be run within the Node.js runtime on many different SOs [28], [29].

	PHP	Ruby	Phyton	Javascript (Node.js)
Can be deployed on most web servers	Yes	Yes	Yes	Yes
Works on almost every OS	Yes	Yes	Yes	Yes
Object Oriented	Yes	Yes	Yes	Yes
Multi-processor work	Good	Good	Bad	Medium
Processing time	Medium	Slow	Slow	Good
Security	Regular security features	Good security features		

Figure 2.9. PHP, Ruby, Phyton and Node.js comparison

Optimisee works with multiple PHP objects that are interrelated. First of all, we should distinguish these objects with which we will work. We start from the data center room where we may find different room objects as well as controlling systems. Focusing on what interest us the most, the room objects, we may find racks or cabinets and sensors. Each rack is directly related to a server, that will store the information collected from the devices placed on the rack. Each rack will have multiple sensors to measure the most relevant parameters of it. There are many different kinds of sensors: we may find temperature, humidity, energy and pressure sensors among others, as it is shown in the next figure.

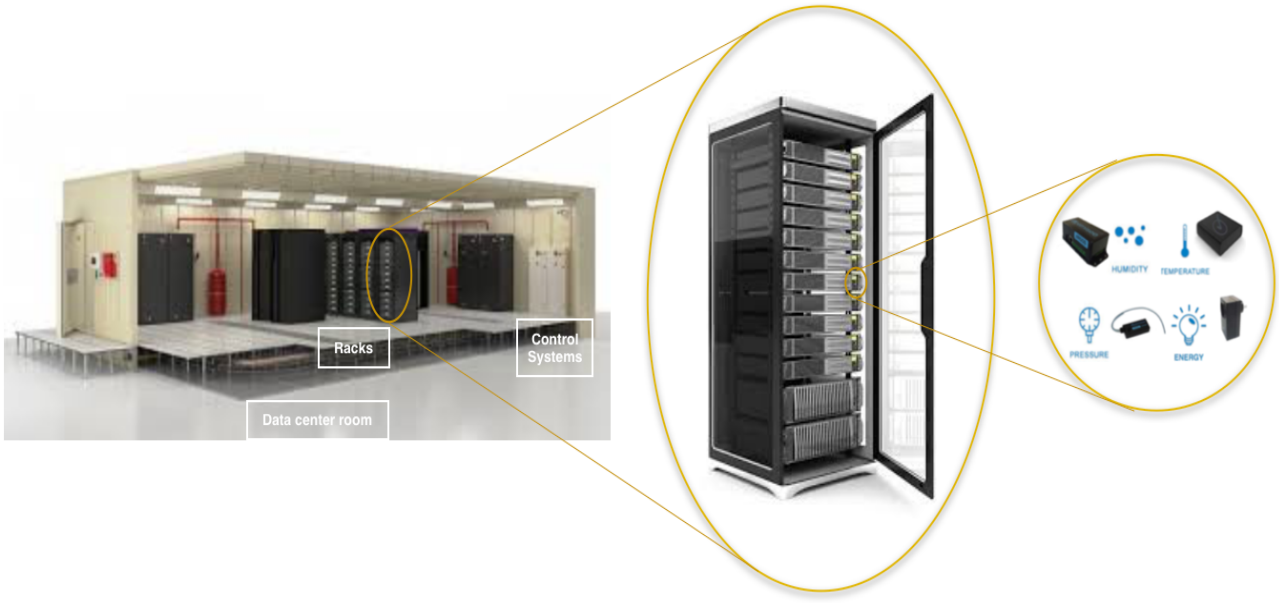


Figure 2.10. Objects in a data center room

In *Figure 2.11* it is shown the UML Diagram followed to develop the dashboard.

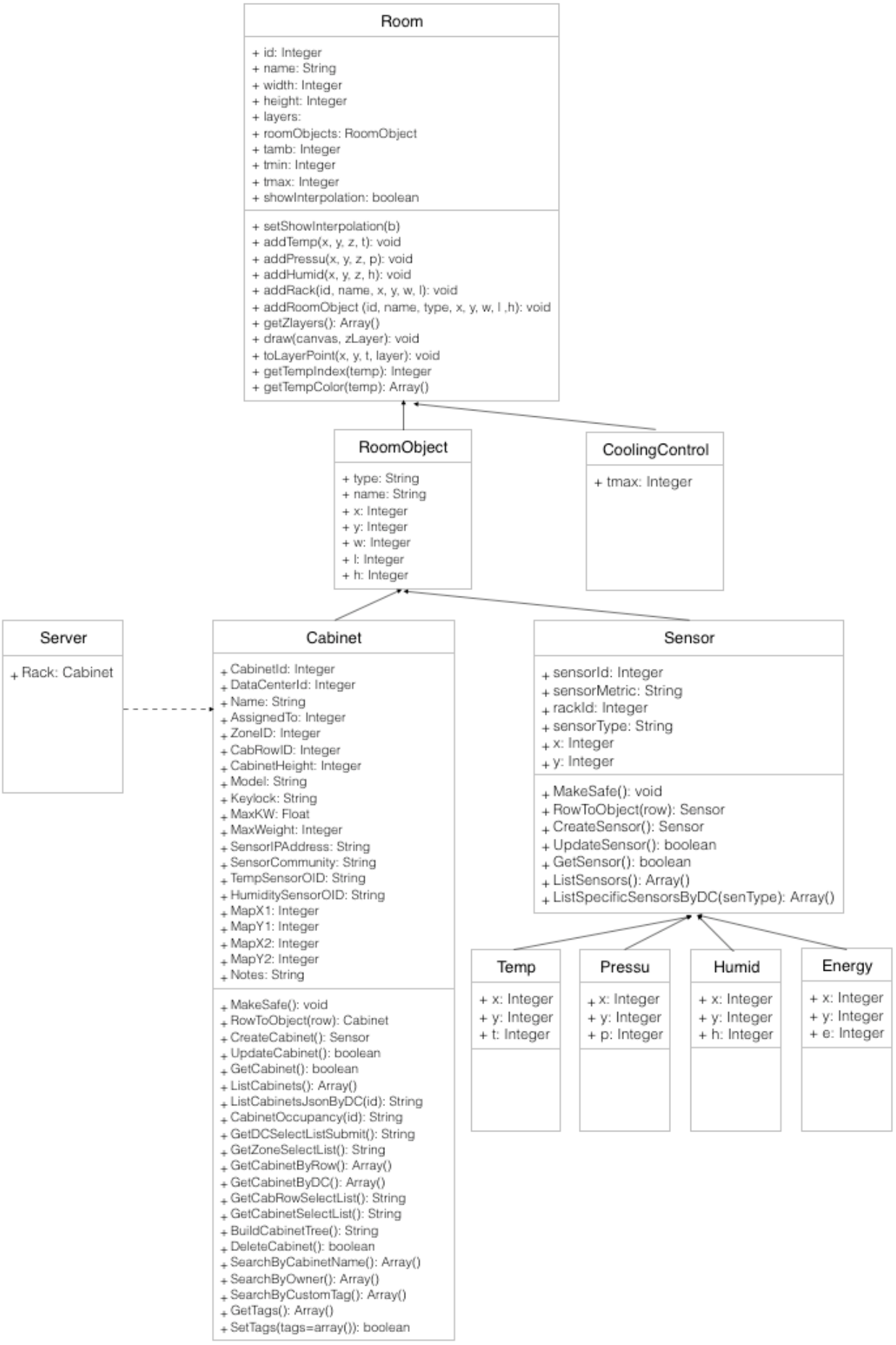


Figure 2.11. UML diagram

2.3.4. CLIENT SIDE

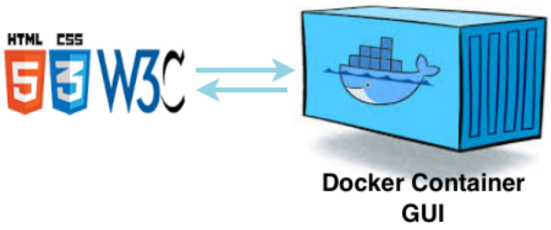


Figure 2.12. Client side architecture

In the client side, we read and represent graphically real-time data. In order to do that we studied different graphic libraries and chose some of them to work with.

GRAPHIC LIBRARIES

We manage different graphic libraries due to the multiple functionalities we offer.

HIGHCHARTS

Highcharts is a charting library written in pure JavaScript, offering an easy way of adding interactive charts to a web site or web application. Highcharts currently supports line, spline, area, areaspline, column, bar, pie, scatter, angular gauges, arearange, areasplinerange, columnrange, bubble, box plot, error bars, funnel, waterfall and polar chart types and many of these can be combined in one chart. It works in all modern mobile and desktop browsers including the iPhone, iPad and Internet Explorer from version 6. On iOS and Android, multitouch support provides a seamless user experience. Standard browsers use SVG for the graphics rendering. In legacy Internet Explorer graphics are drawn using VML. Moreover, Highcharts is a free for non-commercial application [31].

One of the key features of Highcharts is that under any of the licenses, user are allowed to download the source code and make their own edits. This allows for personal modifications and a great flexibility. Highcharts is solely based on native browser technologies and does not require client side plugins like Flash or Java. Furthermore, the user does not need to install anything on his server. Highcharts needs only two JS files to run: the highcharts.js core and either the jQuery, MooTools or Prototype framework. One of these frameworks is most likely already in use in your web page. It has a simple configuration syntax. Setting the Highcharts configuration options requires no special programming skills. The options are given in a JavaScript object notation structure, which is basically a set of keys and values connected by colons, separated by commas and grouped by curly brackets.

Through a full API the user can add, remove and modify series and points or modify axes at any time after chart creation. Numerous events supply hooks for programming against the chart. In combination with jQuery, MooTools or Prototype's Ajax API, this opens for solutions like live charts constantly updating with values from the server, user supplied data and more.

Sometimes the user wants to compare variables that are not of the same scale. Highcharts lets him assign an y axis for each series or an x axis if the user wants to compare data sets of different categories. Each axis can be placed to the right or left, top or bottom of the chart. All options can be set individually, including reversing, styling and position.

Highcharts takes the data in a JavaScript array, which can be defined in the local configuration object, in a separate file or even on a different site. Furthermore, the data can be handled over to Highcharts in any form and a callback function can be used to parse the data into an array.

D3.JS

D3 is a JavaScript library for producing dynamic, interactive data visualizations in web browsers. It makes use of the widely implemented SVG, HTML5 and CSS standards. In contrast to many other libraries, D3.js allows great control over the final visual result. Its development was noted in 2011. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation [32].

D3 takes care to the efficient manipulation of documents based on data. This avoids proprietary representation and affords extraordinary flexibility, exposing the full capabilities of web standards such as HTML, SVG and CSS.

With minimal overhead, D3 is extremely fast, supporting large datasets and dynamic behaviours for interaction and animation. D3's functional style allows code reuse through a diverse collection of components and plugins.

Modifying documents using the W3C DOM API is tedious due to several reason like the method names are verbose and the imperative approach requires manual iteration and bookkeeping of temporary state. D3 employs a declarative approach, operating on arbitrary sets of nodes called selections. Selectors are defined by the W3C Selectors API and supported natively by modern browsers.

D3 provides numerous methods for mutating nodes that will cover the majority of needs like setting attributes or styles, registering event listeners, adding, removing or sorting nodes and

changing HTML or text content. Direct access to the underlying DOM is also possible, as each D3 selection is simply an array of nodes.

D3 does not introduce a new visual representation. D3's vocabulary of graphical marks comes directly from web standards such us HTML, SVG and CSS. The user may use new expressions introduced by the browsers immediately without any toolkit update. Moreover, D3 is easy to debug using the browser's built-in element inspector because the nodes that the user manipulates with D3 are exactly those that the browser understands natively.

CUBISM

Cubism.js is a D3 plugin for visualizing time series. It is a extremely interesting option when designing realtime dashboards, pulling data from Graphite, Cube and other sources. Cubism is available under the Apache License on GitHub [33].

Cubism fetches time series data that is rendered incrementally. After the initial display, Cubism reduces server load by polling only the most recent values. It also renders incrementally, using Canvas to shift charts one pixel to the left. This makes Cubism easily scalable, updating hundreds of metrics every ten seconds. Despite asynchronous fetching, rendering is synchronized so that charts update simultaneously, further improving performance and readability.

In addition, Cubism scales in terms of perception. Cubism's horizon charts make better use of vertical space than standard area charts, allowing the users to see multiple metrics at a glance and helping them to discriminate small changes. These horizon charts reduce vertical space without losing resolution. Larger values are overplotted in successively darker colors, while negative values are offset to descend from the top. As the user increases the number of colors, he reduces the required vertical space. By combining position and color, horizon charts improve perception. Position is highly effective at discriminating slight changes, while color differentiates large changes.

Furthermore, Cubism is considered a flexible tool for several reasons. It has built in support for Graphite and Cube and it can be readily extended to fetch data from other sources. Client-side metric arithmetic allows for further flexibility by combining metrics from multiple sources. Moreover, Cubism's modular components are designed for extensibility. You can add new chart types and different modes of interaction. Cubism builds on D3, making it highly customizable via CSS and JavaScript.

DUCKSBOARD

Ducksboard was acquired by New Relic, Inc. in October 2014. They offer a monitoring product called New Relic Browser that provides the user full visibility into the complete web page lifecycle for modern browser-side applications. Around 90% of a web page's load time is accounted for on the frontend. New Relic Browser takes advantage of this opportunity to improve browser-side performance and offers powerful performance insights [34].

It usually runs custom queries to analyze users by location, browser, device and any other metrics vital to user application's success. It also tracks the pageviews, response times and sessions of specific accounts or users to give support and operations teams insight into customer experience. It determines the content and features that users are engaging in a web page to help prioritize which parts of an application need optimization. Although, it supports many different and common formats, this library was finally dismissed in preference for Cubism because Cubism library works better with Docker containers.

CARTODB

The CartoDB Editor is a self-service mapping and analysis tool that combines an intuitive interface with powerful discovery features. It was born on the web, for this reason there is no need to install any additional software and the user can access the latest features anytime and anywhere. It is easy to learn, there are great tutorials and documentation as well as a gallery with example visualizations. CartoDB also connects to the places where user data already lives, so their analysis is always up-to-date. It is designed to answer big questions. Even with millions of data points and a huge audience, CartoDB stays speedy. CartoDB has powerful APIs to build location-intelligent applications [35].

The CartoDB Editor filters, clusters and explores location-based trends. Also, it tests the user's hunches and gain new perspective by incorporating its public, market-specific and specialty data. It does advanced analysis on the fly, thus the user may see the results in real time. However, as it happens with Duckboard, this library was dismissed because CartoDB only works with maps and it will affect the dashboard scalability.

CLIENT-SERVER WEB TECHNOLOGIES

AJAX

Asynchronous JavaScript and XML, a browser feature accessible in JavaScript for creating fast and dynamic web pages. It allows a script to make an HTTP request to a website or send data to a server in background, without the need for a page reload. It also may request or receive information from a server after the page was loaded. Ajax has been around for more than a decade.

Though the name includes XML, the user can transfer nearly anything in an Ajax request. The most commonly used data is JSON, which is close to JavaScript syntax and consumes less bandwidth.

AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page. It is based on internet standards, and uses a combination of XMLHttpRequest object to retrieve data from a web server and JavaScript/DOM to display or use the data [36].

In *Figure 2.13*, it is explained how Ajax requests and responses work in a web page and how they communicate with the server.

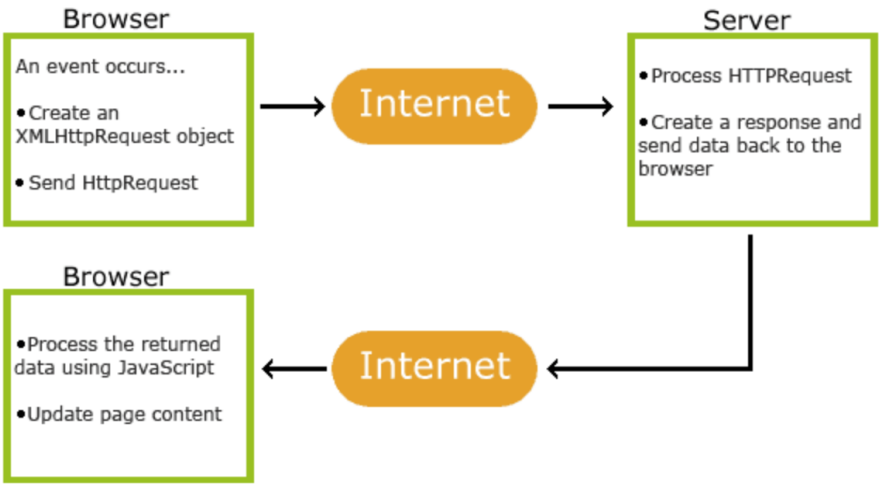


Figure 2.13. Ajax Diagram

Ajax itself is mostly a generic term for various JavaScript techniques used to connect to a web server dynamically without necessarily loading multiple pages. In a more narrowly-defined sense, it refers to the use of XmlHttpRequest objects to interact with a web server dynamically via JavaScript. Some of the main benefits of using Ajax in web applications are:

1. Ajax is used to perform a callback, making a quick round trip to and from the server to retrieve and/or save data without posting the entire page back to the server. By not performing a full postback and sending all form data to the server, network utilization is minimized and quicker operations occur. In sites and locations with restricted bandwidth, this can greatly improve network performance. Most of the time, the data being sent to and from the server is minimal. By using callbacks, the server is not required to process all form elements. By sending only the necessary data, there is limited processing on the server. There is no need to process all form elements, process the ViewState, send images back to the client or send a full page back to the client.

2. Ajax allows the user to make asynchronous calls to a web server. This allows the client browser to avoid waiting for all the data to arrive before allowing the user to act once more.

3. Because a page postback is being eliminated, Ajax enabled applications will always be more responsive, faster and more user-friendly.

4. The main purpose of Ajax is to improve the speed, performance and usability of a web application.

Ajax callbacks can be done by instantiating an XMLHttpRequest object in the client-side JavaScript. The XMLHttpRequest object can be used to directly call server-side objects like pages and web services. These pages and web services will either save and/or return data. Once a client initializes an Ajax callback to the server, the client will not need to wait for a response and can continue to use the web application while the request is being processed. Once done, the server will send a response back to the client and the client will process it as necessary [37].

JavaScript is the client-side programming language and XML is a markup language to define data. JSON is another markup language to define data. JSON is much easier to use with JavaScript than XML. When it comes to Ajax and JavaScript, JSON Web Services are replacing XML Web Services. Another major advance to JavaScript and Ajax is the JavaScript object library called jQuery. This free, open-source software is a wrapper around JavaScript. jQuery is used to easily write client-side JavaScript to navigate and manipulate a page and make asynchronous Ajax callbacks. By using jQuery and JSON Web Services, Ajax callbacks have become standard programming practices for designing and developing web applications.

REVERSE AJAX

Reverse Ajax is the biggest new feature in DWR *(see Appendix IV)* 2.0. It gives the user the ability to asynchronously send data from a web-server to a browser. In a standard HTTP Ajax request, data is sent to the server [38]. Reverse Ajax can be simulated to issue an Ajax request. However, the server can send events to the client. In other words, they maintain a low-latency communication [39]. DWR supports 3 methods of pushing the data to the browser: Piggyback, Polling and Comet.

- Polling is the most obvious solution to the problem. The browser makes a request to the server at regular and frequent intervals looking for updates in the page.
- Comet allows the server to start answering the browser's request for information very slowly and to continue answering on a schedule dictated by the server *(see Appendix V)*.
- With the piggyback option, the server, having an update to send, waits for the next time the browser makes a connection and then sends its update along with the response that the browser was expecting.

Each method has some benefits. Polling is simple to implement, however it can easily overload a server. In contrast, comet is much easier on servers but it is complex. Another comet characteristic is its very low latency: there is no need to wait for the next time the browser connects. Both polling and comet require extra network connectivity. For this reason, piggyback is the version with the lowest overhead, however it does have very high latency. DWR allows you to use all three, with only configuration changes to switch implementation.

DWR can be configured to use Comet or Polling when extra load is acceptable but faster response times are needed. This mode is called active Reverse Ajax. By default DWR starts with active Reverse Ajax turned off, allowing only the piggyback transfer mechanism.

WebSocket

WebSocket is a recent technique that comes from HTML5. Many browsers already support it such us Firefox, Google Chrome or Safari. It enables bidirectional, full-duplex communication channels. The connection is opened through a sort of HTTP request, called WebSockets handshake that has special headers. The connection is kept alive and the user can write and receive data in JavaScript.

WebSocket represent a long awaited evolution in client-server web technology. It allows a long held single TCP socket connection to be established between the client and server which allows for bi-directional, full duplex or messages to be instantly distributed with little overhead resulting in a very low latency connection [40].

Both the WebSocket API and the WebSocket protocol are standardized which means the web now has an agreed standard for realtime communication between Internet clients and servers. Originally considered a browser technology, WebSocket is becoming a cross platform standard for realtime communication between client and server. There are now WebSocket library implementations in Objective-C, .NET, Ruby, Java, node.js, ActionScript and many other languages.

Finally, WebSocket represents a standard for bi-directional realtime communication between servers and clients. Initially in web browsers, but ultimately between any server and any client. The standards first approach means that developers may finally create functionality that works consistently across multiple platforms. Connection limitations are no longer a problem since WebSocket represents a single TCP socket connection.

WebSocket vs Ajax

WebSocket offers new application domains to browser applications that were not really possible using HTTP and AJAX such as interactive games, dynamic media streams or bridging to existing network protocols [41].

However, there is certainly an overlap in purpose between WebSocket and AJAX/Comet. For example, when the browser wants to be notified of server events then Comet techniques and WebSocket are certainly both viable options. However, WebSocket would be the best choice if the application needs low-latency push events or if it has huge connection constraints. On the other hand, we would use Comet techniques, if the application has to work with existing frameworks and deployed technologies like RESTful APIs or proxies. However, we work with Ajax due to it is supported in all the current web browsers and it is a mature technology with several documentation.

2.5. MOCKUP

In *Figure 2.14*, it is shown a web-page mockup example of how the dashboard would look like.

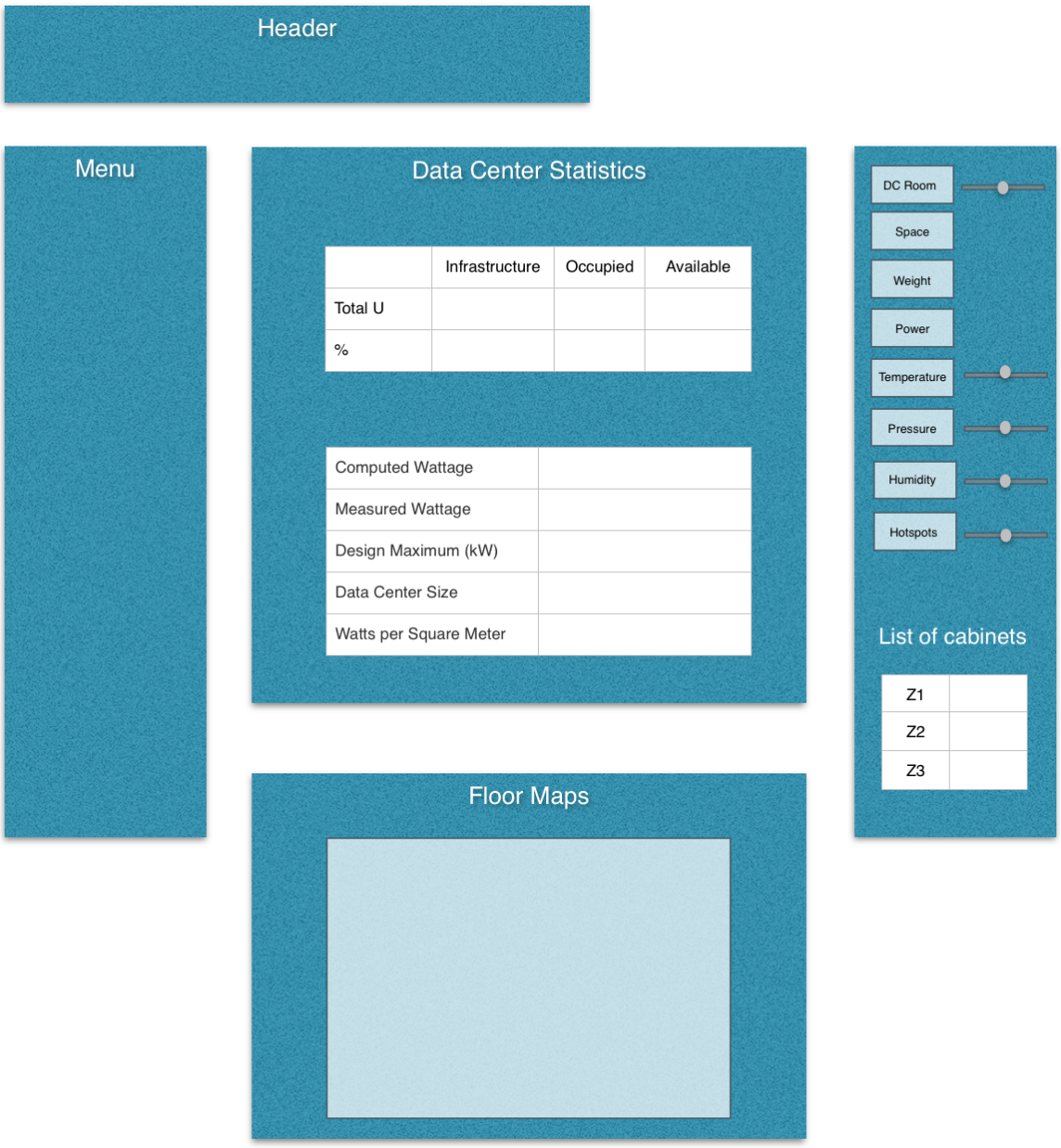


Figure 2.14 Web page mockup

3. IMPLEMENTATION

In this section, it is explained the user requirements as well as the procedures followed to create an overlapped gradient map. One of the most appealing characteristics of these maps is that many different information may be displayed at the same time. Moreover, the user may personalize it, and thus only relevant data will be displayed in a easy-to-understand layout.

The overlapped map consists of a group of layers overlapping one another as it is shown in *Figure 3.1*.

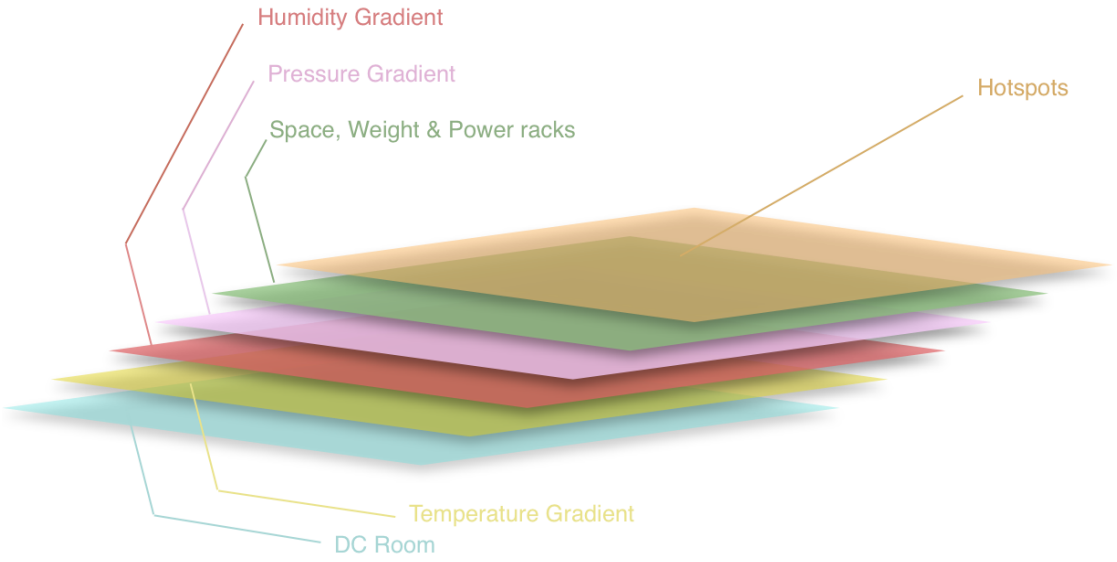


Figure 3.1. Map layers

3.1. USER REQUIREMENTS

The user requirements that have been taken into consideration during the implementation of the map are listed below:

- Bidimensional visualization design
- Real time data available
- Graphical display of different parameters such us:
  - Temperature
  - Hotspots
  - Humidity
  - Pressure
- Generation of map images in the client side
- Control cooling systems through the dashboard
- Easily-Scalable for further designs

3.2. DASHBOARD DESIGN

As you can see in the next figures, there is a number of buttons in the right section of the dashboard. The user may configure his own map by selecting these buttons. Each option has a different function as it is shown below.

- If the user presses the DC Room button, a floor plan of the selected data center room is displayed.
- With the next three buttons (Space, Weight and Power), all the racks of the room appear in different green tonalities to provide information of the parameter selected.
- If the user chooses one of the next three buttons (Temperature, Humidity and Pressure) it will appear the respective temperature, humidity or pressure gradients of the data center room.
- Finally, the hotspot button shows the temperature sensors that monitor temperatures above 40ºC.

The user may change the transparency of the maps moving the cursor that appears next to the corresponding button. However, the buttons that provide information about the rack location do not have this option enabled because we thought it would be interesting to place these racks or not place them but it does not make sense to display them in the map with transparency parameters. Also, some additional features, that are not included in the user requirements, were incorporated to the dashboard like hiding the maps when the button is deselected. We thought that would be more appealing for the user to select a button and choose the image transparency rather than the automatically display the image with last transparency value selected.

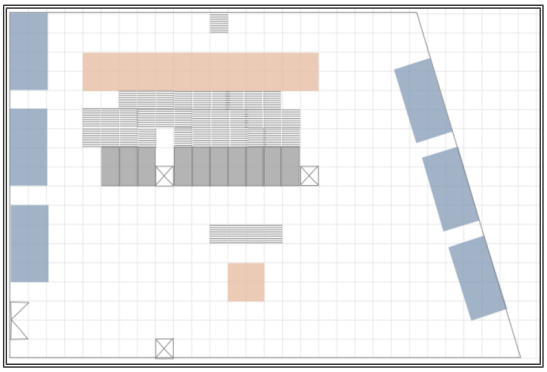
Depending on the space specifications of the rack they would appear in different tonalities. In case the rack takes up a lot of space it will appear in the map with a strong green tonality. The same occurs with the buttons weight and power, the higher the rack power or weight specifications are, the stronger the tonality will be, as it is shown in the figures below.

GT1M1-A Export

	Infrastructure	Occupied	Allocated	Available
Total U 115	19	7	26	63
Percentage	16.5%	6.1%	22.6%	54.8%

Computed Wattage	7200 Watts
Measured Wattage	14841 Watts
Design Maximum (kW)	2500 kW
BTU Computation from Computed Watts	24566 BTU
Data Center Size	500 Square Meters
Watts per Square Meter	14 Watts
Minimum Cooling Tonnage (Based on Computed Watts)	2 Tons

Floor Maps



Map gradient

DC Room 47%

Space

Weight

Power

Temperature

Pressure

Humidity

Hotspots

List of cabinets

Label	Action
Z1	
Z2	
Z3	

Front Cabinets View

Optimisee

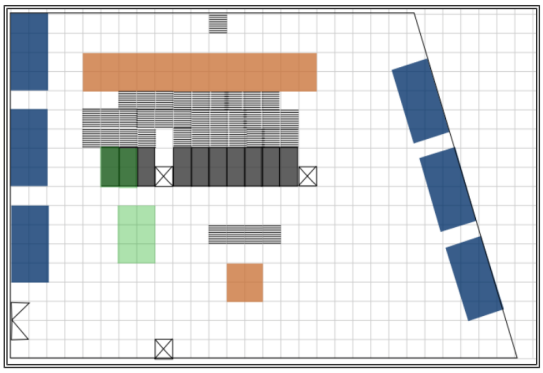
Figure 3.2. Floor room map - transparency 47%

GT1M1-A Export

	Infrastructure	Occupied	Allocated	Available
Total U 115	19	7	26	63
Percentage	16.5%	6.1%	22.6%	54.8%

Computed Wattage	7200 Watts
Measured Wattage	14841 Watts
Design Maximum (kW)	2500 kW
BTU Computation from Computed Watts	24566 BTU
Data Center Size	500 Square Meters
Watts per Square Meter	14 Watts
Minimum Cooling Tonnage (Based on Computed Watts)	2 Tons

Floor Maps



Map gradient

DC Room 100%

Space

Weight

Power

Temperature

Pressure

Humidity

Hotspots

List of cabinets

Label	Action
Z1	
Z2	
Z3	

Front Cabinets View

Optimisee

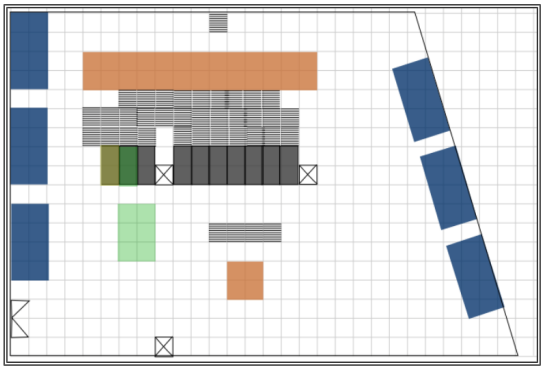
Figure 3.3. Floor room map and space

GT1M1-A Export

	Infrastructure	Occupied	Allocated	Available
Total U 115	19	7	26	63
Percentage	16.5%	6.1%	22.6%	54.8%

Computed Wattage	7200 Watts
Measured Wattage	14841 Watts
Design Maximum (kW)	2500 kW
BTU Computation from Computed Watts	24566 BTU
Data Center Size	500 Square Meters
Watts per Square Meter	14 Watts
Minimum Cooling Tonnage (Based on Computed Watts)	2 Tons

Floor Maps



Map gradient

DC Room 100%
Space
Weight
Power
Temperature
Pressure
Humidity
Hotspots

List of cabinets

Label	Action
Z1	
Z2	
Z3	

Front Cabinets View

Optimisee

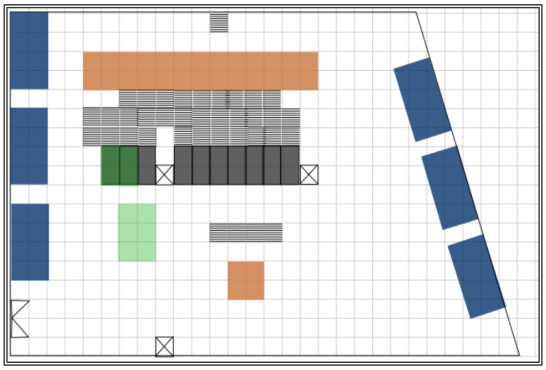
Figure 3.4. Floor room map and weight

GT1M1-A Export

	Infrastructure	Occupied	Allocated	Available
Total U 115	19	7	26	63
Percentage	16.5%	6.1%	22.6%	54.8%

Computed Wattage	7200 Watts
Measured Wattage	14841 Watts
Design Maximum (kW)	2500 kW
BTU Computation from Computed Watts	24566 BTU
Data Center Size	500 Square Meters
Watts per Square Meter	14 Watts
Minimum Cooling Tonnage (Based on Computed Watts)	2 Tons

Floor Maps



Map gradient

DC Room 100%
Space
Weight
Power
Temperature
Pressure
Humidity
Hotspots

List of cabinets

Label	Action
Z1	
Z2	
Z3	

Front Cabinets View

Optimisee

Figure 3.5. Floor room map and power

To design the gradient maps below we use bicubic interpolation equations (see section 3.2) to display the data received in a smoother and more appealing way.

TEMPERATURE GRADIENT LAYER

As it is shown in *Figure 3.6,* near to the racks the temperature in the data center room is higher (high temperatures are represented by red). As we move away form them, the temperature starts to decrease (first we see a yellow area and then when the temperature is lower than 20ºC a blue area is displayed). We can see a clearly progressive variation of the room's temperature below. The range of temperatures considered is between 0-100ºC.

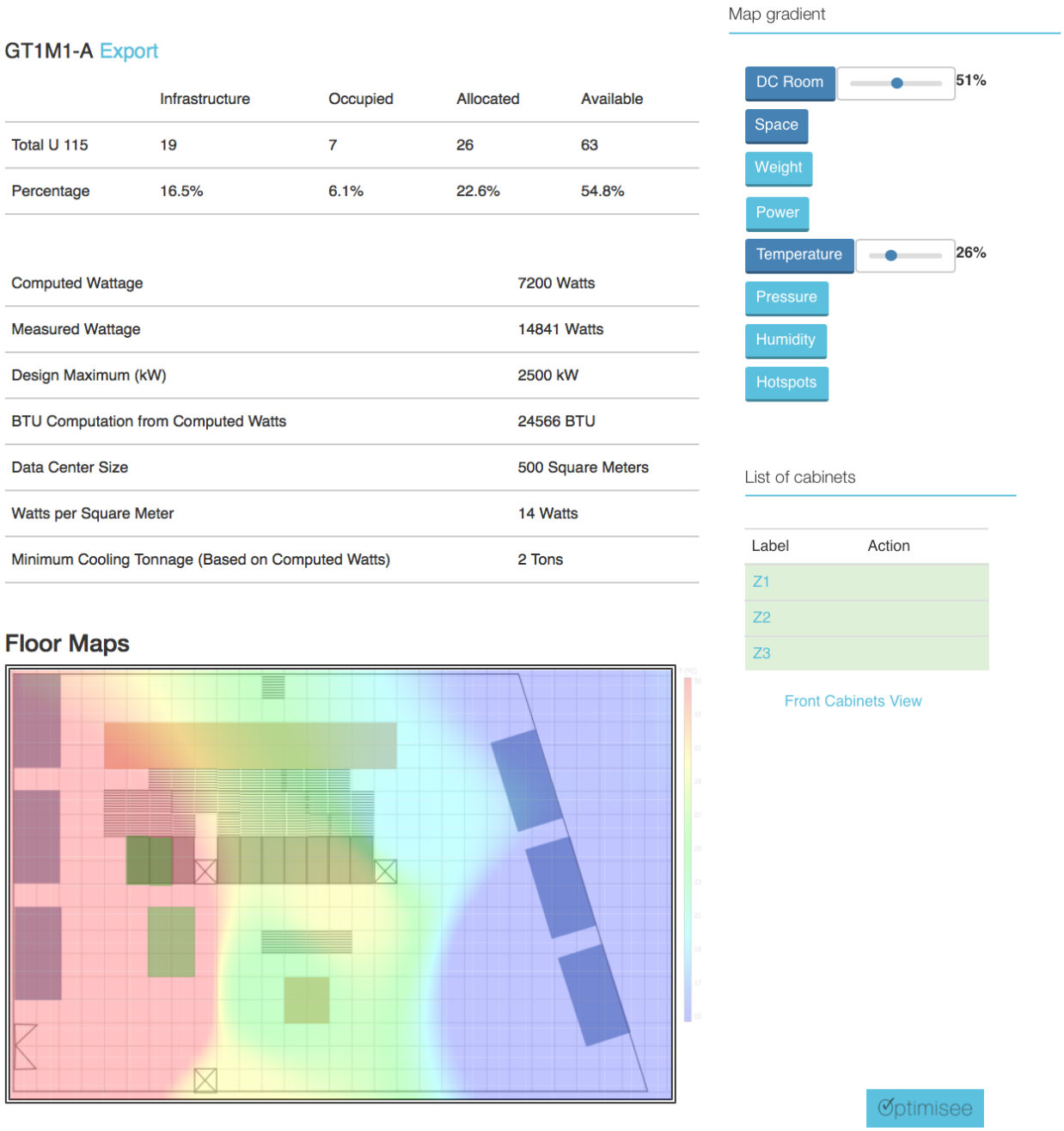


Figure 3.6. Temperature gradient, floor map and space

PRESSURE GRADIENT LAYER

As you can see in *Figure 3.7,* there are not variations of the pressure inside the data center room, as it is a closed-system area. It is considered a pressure value range from 0 to 10 atm (atmospheres). Considering that the average atmospheric pressure is about 1013 mbar or approximately 1 atm, I think that range of values chosen is completely appropriate.

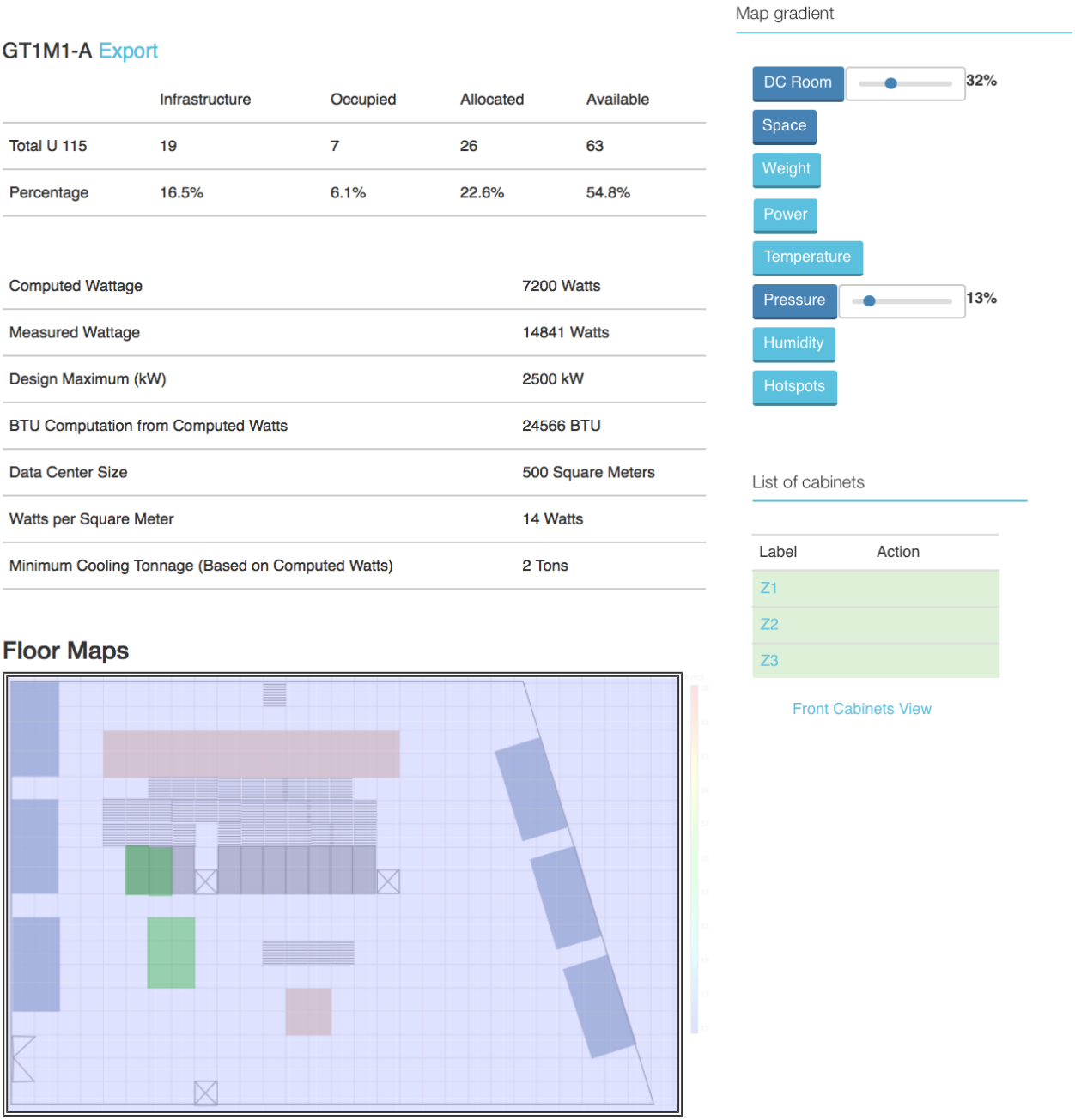


Figure 3.7. Pressure gradient over floor map and space

HUMIDITY GRADIENT LAYER

The humidity range goes from 0 to 100%. *Figure 3.8* shows the room's humidity is almost regular. However, near the racks where, as we mentioned before, the temperature is higher, the area gets drier. This drier area is represented by yellow in the figure below.

Figure 3.8. Humidity gradient over floor map and space

HOTSPOTS LOCATOR LAYER

A map to determine where are the hotspots in a data center room has been design and implemented, as you can see in *Figure 3.9*. The hotspots appear as red spots when the temperature read from a temperature sensor rises up to 40ºC, we can find two different hotspots in the picture near to the racks. This temperature threshold of 40ºC could be modify for the users depending on their requirements.

Figure 3.9. Hotspot locator over temperature gradient, floor map and space.

3.3. API IMPLEMENTATION

First of all in the web page the metrics of the data center room are requested and sent back with the information collected form the database. However, in order to obtain the maps described before, it was necessary to implement an first API functionality in which different ajax requests followed by its responses were sent as it is shown in the next figure as well as a second API functionality that allows the user to create and modify sensors in a data center room.

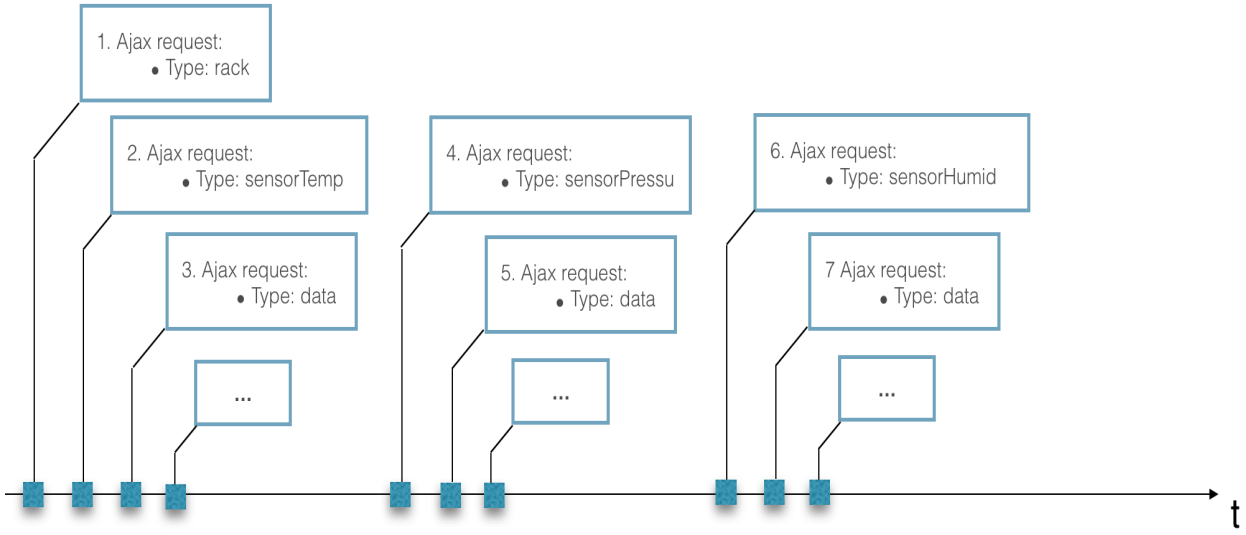


Figure 3.10. Dashboard Ajax requests

1. In the first Ajax petition, a list with all the racks in the room is requested.
2. Once we obtain, the racks that are placed in the room, a second Ajax petition is sent. The response received contains the temperature sensors of each rack.

   At this point we use the second API functionality developed to create, in the client side, the sensors with the parameters obtained from the previous Ajax response.
3. Then, several Ajax requests are sent in order to obtain the temperature of each of the sensors created before.
4. We repeat the second step but in this case with the pressure sensors.
5. We send multiple Ajax requests to obtain the pressure values of all the pressure sensors of each rack.
6. We repeat the second or fourth step with the humidity sensors.
7. We send multiple Ajax requests again in order to obtain the humidity values of all the humidity sensors of each rack.

In the next figure, we can see an example of an different Ajax requests and responses in which the metrics of each rack placed in a specific data center room are collected.

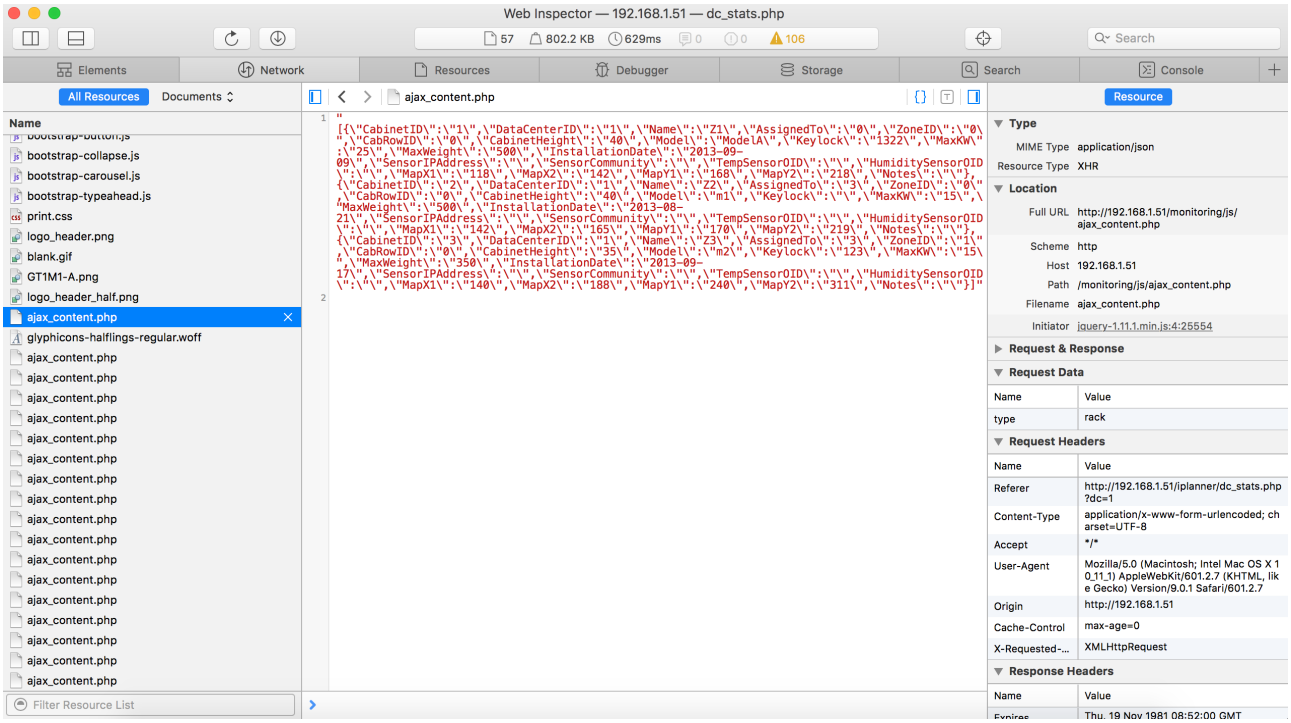


Figure 3.11. Ajax request example in a test scenario.

3.4. BICUBIC INTERPOLATION

We used bicubic interpolation equations to design the gradient overlapped map. They are an extension of cubic interpolation equations for interpolating data points on a two dimensional regular grid. With these algorithms, the interpolated surface is smoother than with other kinds of interpolation. In addition, in image processing, when speed is not an issue, bicubic interpolation becomes a very interesting algorithm mainly because it works with 16 pixels (4×4) more than others interpolation algorithms.

Considering a unit square (1x1), where in its corners the function values f and the derivatives fx, fy and fxy are known in its corners, we can express any interpolated surface as follows:

$$f(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} a_{ij}\, x^i y^j$$

*Where n = 3 and m = 3, to obtain the 16 pixels
*It must contain an integer number of unit squares

The interpolation problem consists of determining the coefficients a_{ij}. We obtain 4 equations matching p(x,y) with the function corner values, eight equations for the derivates in the x-direction and y-direction and four more for the cross derivative xy [42].

$$f_x(x,y) = \sum_{i=1}^{n}\sum_{j=0}^{m} a_{ij}\, i\, x^{i-1} y^j \qquad f_y(x,y) = \sum_{i=0}^{n}\sum_{j=1}^{m} a_{ij}\, x^i j\, y^{j-1} \qquad f_{xy}(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}\, i\, x^{i-1} j\, y^{j-1}$$

This set of linear equations form a matrix equation of the form: $M \cdot \alpha = \beta$ where:

$$\alpha = [a_{00}\ a_{10}\ a_{20}\ a_{30}\ a_{01}\ a_{11}\ a_{21}\ a_{31}\ a_{02}\ a_{12}\ a_{22}\ a_{32}\ a_{03}\ a_{13}\ a_{23}\ a_{33}]$$

$$\beta = [f(0,0)\ f(1,0)\ f(0,1)\ f(1,1)\ f_x(0,0)\ f_x(1,0)\ f_x(0,1)\ f_x(1,1)\ f_y(0,0)\ f_y(1,0)\ f_y(0,1)\ f_y(1,1)\ f_{xy}(0,0)\ f_{yx}(1,0)\ f_{xy}(0,1)\ f_{xy}(1,1)]^T$$

Inverting the matrix, $\alpha = M^{-1} \cdot \beta$, allows us to calculate α.

M =

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	2	0	0	0	3	0	0	0
0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	2	3	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	2	0	0	0	3	0	0
0	0	0	0	0	1	2	3	0	2	4	6	0	3	6	9

Figure 3.12. Coefficient Matrix for a values

M^{-1} =

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
-3	3	0	0	-2	-1	0	0	0	0	0	0	0	0	0	0
2	-2	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	-3	3	0	0	-2	-1	0	0
0	0	0	0	0	0	0	0	2	-2	0	0	1	1	0	0
-3	0	3	0	0	0	0	0	-2	0	-1	0	0	0	0	0
0	0	0	0	-3	0	3	0	0	0	0	0	-2	0	-1	0
9	-9	-9	9	6	3	-6	-3	6	-6	3	-3	4	2	2	1
-6	6	6	-6	-3	-3	3	3	-4	4	-2	2	-2	-2	-1	-1
2	0	-2	0	0	0	0	0	1	0	1	0	0	0	0	0
0	0	0	0	2	0	-2	0	0	0	0	0	1	0	1	0
-6	6	6	-6	-4	-2	4	2	-3	3	-3	3	-2	-1	-2	-1
4	-4	-4	4	2	2	-2	-2	2	-2	2	-2	1	1	1	1

Figure 3.13. Inverted Coefficient Matrix

This procedure yields a surface f(x,y) on the unit square which is continuous and with continuous derivates. Bicubic interpolation on an arbitrarily sized regular grid can then be accomplished by patching together such bicubic surfaces, ensuring that the derivatives match on the boundaries [43].

4. CONCLUSIONS

The computational and cooling power demand of enterprise servers is increasing at an unsustainable rate. Higher chip power densities brought about by new process technologies cause temperatures to increase. Moreover, as data center cooling becomes more efficient, the contribution of server clients become more significant.

Understanding the relationship between computational power, temperature and cooling systems is crucial to enable energy-efficient operations at server and data center levels. It would be extremely interesting to develop action models, with different actuators to carry out specific actions when risk situations occur, all based on client requirements.

Big data services together with Internet of Things and Cloud Computing technologies are the new trend of analysis and what competitive enterprises use for modelling tendencies in data centers. These forecast models are precise and complex but the benefits are significant for monitoring companies.

Our dashboard solution enables the usage of proactive optimization strategies both at server and data center levels. If we review the objectives of the project (section 1.4) we may affirm that:

- The first objective about the analysis, study and understanding of Optimise's architecture has been accomplished.
- Secondly, I have made an analysis of the current technologies involved in web design where I considerably increased my knowledge on this field.
- The third goal was almost fulfilled, a specific API to request and send data into the dashboard was developed. However, the data update when the web page is loaded again.
- An overlapped map has been completely implemented an it does display:
  - A temperature gradient
  - A pressure gradient
  - A humidity gradient
  - A hotspot location layer.
- The fifth objective has been also achieved, we generate the map images in the client side because we can work directly with the data collected and display it in more efficient formats.

We have also take into consideration, during the implementation of the dashboard almost all of the user requirements described in the section 3.1:

- We developed a bidimensional visualization design, more precisely we designed an overlapped map instead of plotting a graph with the different sensors and their values because we considered that a gradient map will display all the information that the user needs in a more easy-to-understand way.
- We display the most important parameters of a data center: temperature, humidity, pressure and hotspots.
- We generate the map images in the client side, as we mention before.
- We implement an easily and scalable system for further designs. It not complex to add more layers in the maps or increase the number of sensors.

To conclude with, this dashboard allows users to realise about inefficiencies about equipment location or cooling systems. The temperature gradient will be very useful to distribute the racks in a data center room, to analyse if the cooling systems are efficient or/and it will help the user to balance the work-load between racks.

5. Future Design Lines

In a short term, the future lines to design would be:

- To automatically update the data displayed in the maps every hour. As it is currently implemented, the sensor values are always read when the web page loads. It should be considered to request these data only when the user asks for it in order to be more time-efficient.

- To include an energy gradient layer when placing energy sensors in the data center room.

In a more long term, it could be interesting to consider the implementation of the dashboard on AJAX reverse due to all the benefits that it offers, mentioned before on section 2.3.4. Also, it would be relevant to develop a U's rack location system inside of a data center room. Designing a map with a coordinate system that will display in real time what Us are stored in each rack would be very useful for maintenance staff and it is a fact that for companies, improving staff time efficiency will represent huge savings. The location system could be implemented by RFID or NFC technologies, which are both mature technologies that are already present in many devices.

To conclude with, I would like to introduce the possibility of developing a dynamic cooling system that will adjust automatically the room's temperature. In order to achieve that, it would be necessary to assure an effective communication between the frontend and the SCADA *(see Figure 2.1)* which is the system that controls all the actuator and controlling systems in Optimisee data center rooms.

6. APPENDICES

6.1. APPENDIX I: MICROSOFT AZURE

Microsoft Azure is a growing collection of integrated cloud services such us analytics, computing, database, mobile, networking, storage and web, for moving faster, achieving more and saving money [44].

Its integrated tools, pre-built templates and managed services make it easier to build and manage enterprise, mobile, Web and IoT applications faster, using skills that the user has and technologies that he already knows.

Azure supports the broadest selection of operating systems, programming languages, frameworks, tools, databases and devices. The user may run Linux containers with Docker integration, build apps with JavaScript, Python, .NET, PHP, Java and Node.js or build back-ends for iOS, Android and Windows devices.

Some cloud providers make the user choose between his data center and the cloud. However, Azure integrates easily with his existing IT environment through the largest network of secure private connections, hybrid database and storage solutions, and data residency and encryption features. The user can even run Azure in his own data center. Azure's pay-as-you-go services can quickly scale up or down to match demand, allowing the user to pay for what he or she uses.

Some organizations are still wary of the cloud, for this reason Microsoft has made an industry-leading commitment to the protection and privacy of user's data. It was the first cloud provider recognized by the European Union's data protection authorities for its commitment to rigorous EU privacy laws. Microsoft was also the first major cloud provider to adopt the new international cloud privacy standard, ISO 27018. They also launched Azure Government, a stand-alone version of Azure designed to meet the rigorous compliance requirements of U.S. public agencies.

Azure runs on a worldwide network of Microsoft-managed data centers across 22 regions (more countries and regions than Amazon Web Services and Google Cloud combined). This fast-growing global footprint gives the user lots of options for running applications and ensuring great customer performance. Azure is also the first multinational cloud provider in mainland China.

Azure's predictive analytics services, including Machine Learning, Cortana Analytics and Stream Analytics, are redefining business intelligence. It will help the user to make smarter

decisions, improve customer service and uncover new business possibilities from your structured, unstructured and streaming Internet of Things data. The user can also build and deploy modern, cross platform web and mobile applications, run his enterprise applications on Azure, run large scale compute jobs and perform powerful predictive analytics, encode, store and stream audio and video at scale or build intelligent products and services leveraging Internet of Things services.

6.2. Appendix II: IaaS

In the most basic cloud service model, according to the IETF, providers of Infrastructure as a Service offer computers, physical or virtual machines and other resources. IaaS refers to online services that abstract the user from the details of the infrastructure like physical computing resources, location, data partitioning, scaling, security or backups. A hypervisor, such as Xen, Oracle VirtualBox, KVM, VMware ESX/ESXi or Hyper-V runs the virtual machines as guests [45].

Pools of hypervisors within the cloud operational system can support large numbers of virtual machines and the ability to scale services up and down according to customers' requirements. IaaS clouds often offer additional resources. The providers of this services supply these resources on demand from their equipments installed in data centers.

To deploy their applications, cloud users install operating system images and their application software on the cloud infrastructure. In this model, the cloud user patches and maintains the operating systems and the application software.

6.3. Appendix III: RRD

A Round Robin Database is a specialized storage system with a constant footprint over time. In a RRD large amounts of time series information such as temperatures, network bandwidth, CPU load or stock prices are stored [46].

In the short term, each data point is significant, they want an accurate picture of every event that has occurred in the last 24 hours, which might include small transient spikes in disk usage or network bandwidth. However, in the long term, only general trends are necessary. For this reason, in order to save space, they compact the older data using a CF, which performs some computation on many data points to combine it into a single point over a longer period. Though they have lost precision, the data is still tremendously useful for demonstrating general trends over time.

6.4. Appendix IV: Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web

development, making easier to build quickly and with less code better Web apps. It is free and open source [47].

Django framework is characterized by:

- Its scalability, some of the busiest sites on the Web leverage Django's ability to quickly and flexibly scale.
- It is exceptionally fast, Django was designed to help developers take applications from concept to completion as quickly as possible.
- Its security, Django takes security seriously and helps developers avoid many common security mistakes.
- It is incredibly versatile, companies, organizations and governments utilize Django to build content management systems, social networks, scientific computing platforms, etc.

6.5. Appendix V: DWR

DWR is a Java library that enables Java on the server and JavaScript in a browser to interact and call each other as simply as possible. It has a large user-base, active mailing list and has been used in many projects [48].

DWR has a number of features like call batching, marshalling of virtually any data-structure between Java and Javascript, including binary file uploading and downloading, exception handling, advanced CSRF protection and deep integration with several Java server-side technologies.

DWR consists of two main parts:

- A Java Servlet running on the server that processes requests and sends responses back to the browser.
- JavaScript running in the browser that sends requests and can dynamically update the webpage.

DWR works by dynamically generating Javascript based on Java classes. Although, the user may think that the execution is happening on the browser, the server is executing the code and DWR is marshalling the data back and forwards. This method of remoting functions from Java to JavaScript gives DWR users a feel much like conventional RPC mechanisms like RMI or SOAP, with the benefit that it runs over the web without requiring web-browser plug-ins.

Since DWR 2.0, Reverse Ajax is available. It allows Java code running on the server to find out what clients are viewing certain pages and to send to them JavaScript, generated either manually or using a Java API. These JavaScript generating APIs generally match a client-side APIs.

6.6. APPENDIX VI: COMET

Comet is a web application model where a request is sent to the server and kept alive for a long time, until a time-out or a server event occurs. When the request is completed, another long-lived Ajax request is sent to wait for other server events. With Comet, web servers can send the data to the client without having to explicitly request it [49].

The big advantage of Comet is that each client always has a communication link open to the server. The server can push events on the clients by immediately committing the responses when they arrive or it can even accumulate and send bursts. Because a request is kept open for a long time, special features are required on the server side to handle all of these long-lived requests.

Implementations of Comet can be separated into two types:

- Comet using HTTP streaming mode.

In this case, one persistent connection is opened. There will only be a long-lived request since each event arriving on the server side is sent through the same connection. Thus, it requires on the client side a way to separate the different responses coming through the same connection. There are two common techniques for streaming the first one include Forever Iframes or the one that includes the multi-part feature of the XMLHttpRequest object used to create Ajax requests in JavaScript.

  - Forever Iframes technique

It involves a hidden Iframe tag put in the page with its src attribute pointing to the servlet path returning server events. Each time an event is received, the servlet writes and flushes a new script tag with the JavaScript code inside. The iframe content will be appended with this script tag that will get executed. It is simple to implement and it works in all browsers supporting iframes. However, there is no way to implement reliable error handling or to track the state of the connection, because all the connections and data are handled by the browser through HTML tags. It is imposible to detect when the connection is broken on either side.

  - Multi-part XMLHttpRequest technique

It is more reliable than the mentioned before. In this case the multi-part flag supported by some browsers is used on the XMLHttpRequest object. An Ajax request is sent and kept open on the server side. Each time an event comes, a multi-part response is written through the same connection. With this technique only one persistent connection is opened. This is the Comet technique that saves the most bandwidth usage. However, the multi-part flag is not supported by all browsers. Some widely used libraries, such as CometD in Java, reported issues in buffering.

- Comet using HTTP long polling mode

It involves techniques that open a connection. The connection is kept open by the server and as soon as an event occurs, the response is committed and the connection is closed. Then, a new long-polling connection is reopened immediately by the client waiting for new events to arrive.
You can implement HTTP long polling by using:

- Script tags

In this case, as with iframes, the goal is to append a script tag in the user page to get the script executed. The server will suspend the connection until an event occurs, send the script content back to the browser and then reopen another script tag to get the next events. This technique is very easy to implement and works across domains because it is based on HTML tags. However, similar to the iframe technique, error handling is missing and the user can not have a state or the ability to interrupt a connection.

- XMLHttpRequest long polling object

It is the second and most recommended method to implement Comet. It consist of opening an Ajax request to the server and wait for the response. The server requires specific features on the server side to allow the request to be suspended. As soon as an event occurs, the server sends back the response in the suspended request and closes it. The client then consumes the response and opens a new long-lived Ajax request to the server. It is easy to implement on the client side with a good error-handling system and timeout management. This reliable technique also allows a round-trip between connections on the server side, since connections are not persisten. It also works on all browsers, the user only makes use of the XMLHttpRequest object by issuing a simple Ajax request. However, this technique still relies on a stateless HTTP connection, which requires special features on the server side to be able to temporarily suspend it.

7. BIBLIOGRAPHY

[1] «Eurostat,» [En línea]. Available: http://ec.europa.eu/eurostat/statistics-explained/index.php/Cloud_computing_-_statistics_on_the_use_by_enterprises#Data_sources_and_availability

[2] «Anthesis Group,» [En línea]. Available: http://anthesisgroup.com/latest-research-by-anthesis-americas-data-centers-consuming-massive-and-growing-amounts-of-electricity/.

[3] Marina Zapater Sancho, «Proactive and reactive thermal aware optimisation techniques to minimise the environmental impact of data centers,» de *Tesis Doctoral ETSIT UPM*, 2015.

[4] «Global Internet Device Installed Base Forecast,» [En línea]. Available: http://www.philoso.de/de/00004debatten/00001Big%20Data/00005Chancen%20und%20Gefahren/index.php

[5] «Forbes,» [En línea]. Available: http://www.forbes.com/sites/louiscolumbus/2015/07/26/analytics-cloud-computing-dominate-internet-of-things-app-developers-plans/

[6] «Intel,» [En línea]. Available: http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/big-data-cloud-technologies-brief.pdf

[7] «SynapSense,» [En línea]. Available: http://www.panduit.com/en/panduit-acquires-synapsense

[8] «OpenStack,» [En línea]. Available: https://en.wikipedia.org/wiki/OpenStack

[9] «VMware,» [En línea]. Available: http://www.vmware.com/es/cloud-computing/overview.html#sthash.UjfrSd7R.dpuf

[10] «Amazon EC2,» [En línea]. Available: https://aws.amazon.com/ecr/?nc1=h_ls

[11] «HTML5,» [En línea]. Available: https://en.wikipedia.org/wiki/HTML5

[12] «CSS3,» [En línea]. Available: http://www.w3schools.com/css/css3_intro.asp

[13] «CSS,» [En línea]. Available: https://en.wikipedia.org/wiki/Cascading_Style_Sheets

[14] «Bootstrap,» [En línea]. Available: http://getbootstrap.com

[15] «Graphite,» [En línea]. Available: http://graphite.readthedocs.org/en/latest/overview.html

[16] «Drraw,» [En línea]. Available: http://web.taranis.org/drraw/

[17] «Cacti,» [En línea]. Available: http://www.cacti.net

[18] «Cetreon,» [En línea]. Available: https://www.centreon.com/en/

[19] «Carbon,» [En línea]. Available: http://graphite.readthedocs.org/en/latest/carbon-daemons.html

[20] «ExtJS,» [En línea]. Available: https://en.wikipedia.org/wiki/Ext_JS

[21] «MySQL,» [En línea]. Available: https://www.mysql.com

[22] «Wikipedia MySQL,» [En línea]. Available: https://en.wikipedia.org/wiki/MySQL

[23] «Docker,» [En línea]. Available: http://docs.docker.com/engine/misc/

[24] «Firewall,» [En línea]. Available: https://en.wikipedia.org/wiki/Firewall_(computing)

[25] «VPN,» [En línea]. Available: https://en.wikipedia.org/wiki/Virtual_private_network

[26] «Graphite's API,» [En línea]. Available: http://graphite-api.readthedocs.org/en/latest/

[27] «Php, Ruby & Phyton comparison,» [En línea]. Available: http://www.1stwebdesigner.com/php-vs-ruby-vs-python/

[28] «Node.js,» [En línea]. Available: https://en.wikipedia.org/wiki/Node.js

[29] «Javascript advantages and disadvantages,» [En línea]. Available: https://www.quora.com/What-the-advantages-and-disavantages-of-using-Java-J2EE-for-both-web-front-end-and-app-layer-nowadays-for-big-sites-as-apposed-to-other-major-frameworks

[30] «Cairo,» [En línea]. Available: http://cairographics.org

[31] «Highcharts,» [En línea]. Available: http://www.highcharts.com/products/highcharts

[32] «D3,» [En línea]. Available: http://d3js.org

[33] «Cubism,» [En línea]. Available: https://square.github.io/cubism/

[34] «New Relic,» [En línea]. Available: http://newrelic.com/browser-monitoring

[35] «CartoDB,» [En línea]. Available: https://cartodb.com

[36] «Ajax,» [En línea]. Available: http://www.w3schools.com/Ajax/ajax_intro.asp

[37] «Ajax,» [En línea]. Available: http://www.seguetech.com/blog/2013/03/12/what-is-ajax-and-where-is-it-used-in-technology

[38] «Reverse Ajax,» [En línea]. Available: http://www.ibm.com/developerworks/library/wa-reverseajax1/

[39] «Reverse Ajax,» [En línea]. Available: http://directwebremoting.org/dwr/documentation/reverse-ajax/index.html

[40] «WebSockets,» [En línea]. Available: https://pusher.com/websockets

[41] «Ajax vs WebSockets,» [En línea]. Available: http://stackoverflow.com/questions/10377384/why-use-ajax-when-websockets-is-available

[42] «Bicubic Interpolation,» [En línea]. Available: https://en.wikipedia.org/wiki/Bicubic_interpolation

[43] «Bicubic Interpolation,» [En línea]. Available: http://www.giassa.net/?page_id=371

[44] «Microsoft Azure,» [En línea]. Available: https://azure.microsoft.com/es-es/

[45] «IaaS,» [En línea]. Available: https://en.wikipedia.org/wiki/Cloud_computing#Infrastructure_as_a_service_.28IaaS.29

[46] «RRD,» [En línea]. Available: https://jawnsy.wordpress.com/2010/01/08/round-robin-databases/

[47] «Django,» [En línea]. Available: https://www.djangoproject.com

[48] «DWR,» [En línea]. Available: http://directwebremoting.org/dwr/index.html

[49] «Comet,» [En línea]. Available: https://en.wikipedia.org/wiki/Comet_(programming)